Machine Learning Approaches for Wireless Spectrum and Energy Intelligence

by

Keyu Wu

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Communications

Department of Electrical and Computer Engineering
University of Alberta

Abstract

Cognitive radio and energy-harvesting technologies improve the efficiency of spectrum use and energy use in communication networks. However, due to the randomness and dynamics of spectral and energy resources, wireless nodes must intelligently adjust their operating configurations (radio frequency and transmission power). With machine learning as primary tools, this thesis addresses three spectrum and energy management problems.

First, we consider a single-channel energy-harvesting cognitive transmitter, which attempts to maximize data throughput with harvested energy and dynamically available channel. The transmitter needs to determine whether or not to perform spectrum sensing and channel probing, and how much power for transmission, subject to energy status and wireless channel state. The resulting control problem is addressed by a two-stage reinforcement learning algorithm, which finds the optimal policy from data samples when the statistical distributions of harvested energy and channel fading are unknown.

Second, we consider energy-harvesting sensor, which strives to deliver packets with finite battery capacity and random energy replenishment. A selective transmission strategy is investigated, where low priority packets can be dropped to save energy for high priority data packets. The optimal transmission policy, which determines whether or not a packet should be transmitted, is derived via training an artificial neural network with data samples of packet priorities, wireless channel gains, and harvested energy levels.

Third, we investigate cooperation among cognitive nodes for reliable spectrum sensing given spectrum heterogeneity (i.e., spatial dependence of spectrum availability). Sensing cooperation can mitigate it. However, the challenge is how to model and exploit spatial correlations to fuse sensing data. To overcome this, spatial correlations among cognitive nodes are modeled as a Markov random field; and given data observations, sensing cooperation is achieved by solving a maximum posterior probability estimator over the Markov random field. Under this methodology, three cooperative sensing algorithms are designed for centralized, cluster-based, and distributed cognitive radio networks. These algorithms offer improved computational efficiency and reduced communication overhead.

Preface

This thesis is an original work by Keyu Wu. Parts of the thesis have been published or submitted to journals or conferences, which are indicated below.

Partial work of Chapter 3 is represented at IEEE ICC 2017 as "K. Wu, H. Jiang and C. Tellambura, 'Sensing, Probing, and Transmitting strategy for Energy Harvesting Cognitive Radio,' *2017 IEEE ICC*, Paris, 2017". The full length version is submitted to IEEE Transaction of Vehicular Technology in May 2018 as "K. Wu and H. Jiang, and C. Tellambura, 'Sensing, Probing and Transmitting Policy for Energy Harvesting Cognitive Radio with two-stage After-state Reinforcement Learning' ".

Partial work of Chapter 4 is represented at IEEE ICC 2017 as "K.Wu, C. Tellambura and H. Jiang, 'Optimal Transmission Policy in Energy Harvesting Wireless Communications: A Learning Approach,' *2017 IEEE ICC*, Paris, 2017". The full length version is submitted to IEEE Internet of Things Journal in July 2018 as "K. Wu, F. Li, C. Tellambura, and H. Jiang, 'Optimal Selective Transmission Policy for Energy-Harvesting Wireless Sensors via Monotone Neural Networks' ".

A tutorial version of Chapter 5 is published as "K. Wu, M. Tang, C. Tellambura and D. Ma, 'Cooperative Spectrum Sensing as Image Segmentation: A New Data Fusion Scheme,' in *IEEE Communications Magazine*, vol. 56, no. 4, pp. 142-148, April 2018".

Acknowledgements

I sincerely appreciate my supervisors, Dr. Chintha Tellambura and Dr. Hai Jiang, for their guidance, encouragement, advice and support. In the course of this thesis research, I had been greatly benefited from their profound knowledge and expertise. Their research attitude and standard have set me a valuable example, and will continue influencing me in my future career.

I would like to thank my thesis examining committee members Dr. Yindi Jing, Dr. Ehab Elmallah, and Dr. Dongmei Zhao for their time and efforts in reviewing my thesis and providing me valuable comments. Their feedbacks helped me to improve the quality of the thesis.

I also like to thank all professors who delivered courses during my study, which makes me prepared for the research. Especially, I would like to thank Dr. Richard Sutton for his excellent lecture on reinforcement learning, which continuously inspired throughout the research.

Especial gratitude also goes to present and former colleagues of iCore Wireless Communications Lab W5-070 and Advanced Wireless Communications and Signal Processing Lab W5-077 for their continual help and companion.

Most importantly, I am very thankful to my family, including my parents, my brother, my parents-in-law and especially my wife, Mrs. Tiantian Huang. It is their understanding and unconditional support that help me to get through all difficulties in the four years' path.

Last but not least, I would like to thank all funders throughout my PhD study, including the CSC Scholarship, Alberta Innovates Technology Futures (AITF) Top-up Scholarship, FGSR Travel Awards and IEEE ICC Student Travel Grants. Without their finding support, I could not have completed the PhD study.

Contents

List of Tables

List of Figures

List of Symbols

Symbols in Chapter 3

a	action of an MDP
b	battery level
$B_{\max}$	battery capacity
C	PU channel status
d	endogenous component of a state
e_H	harvested energy
e_S	energy required for sensing
e_P	energy required for probing
e_T	energy required for transmitting
FB	feedback indicator form SU receiver
f_E	probability density function of harvested energy
f_H	probability density function of channel gain
h	wireless channel gain
J^*	after-state value function
p	belief of channel availability
r	reward function
s	state
V^*	state value function
x	exogenous component of a state
π^*	optimal policy
β	after-state
Θ	spectrum sensing result

Symbols in Chapter 4

a	action of an MDP
b	battery level
c	total energy consumption for standing by, data reception and channel estimation
d	data importance value
e	harvested energy
f_B	transition probability of battery level given an after-state
f_D	probability density function of data importance values
f_E	probability density function of harvested energy
f_H	probability density function of required energy for tranmission
h	required energy for transmission
J^*	after-state value function
p	after-state
r	reward function
s	state
V^*	state value function
z	channel power gain
$\boldsymbol{\theta}$	parameter vector of MNN

Symbols in Chapter 5

c	edge cost of a BF graph
d_i	distance from the ith SU to the PU
$f_{Y\|X}$	data likelihood function
K	a graph cut
$\boldsymbol{x}$	all SUs' spectrum statuses
x_i	the ith SU's spectrum status
$\boldsymbol{y}$	all SUs' sensing observation
y_i	the ith SU's sensing observation
$\mathcal{E}$	the edge set of a graph
$\mathcal{G}$	a graph
$\mathcal{V}$	the node set of a graph

α	channel propagation factor
ψ_i	complex channel gain of a fading channel
ϕ	potential function between a pair of neighboring SUs
Φ_X	MRF over SU spectrum statuses
$\boldsymbol{\lambda}$	Lagrange multiplier

Glossary of Terms

Acronyms	**Definition**
ANN	artificial neural network
BP	belief propagation
CSS	cooperative spectrum sensing
CSI	channel status information
MAC	medium access control
MAP	maximum a posterior
MDP	Markov decision process
MNN	monotone neural network
MRF	Markov random field
PDF	probability density function
PrR	protection range
PU	primary user
SU	secondary user

Chapter 1

Introduction

1.1 Communications with Recycled Spectrum and Energy

1.1.1 The next generation wireless systems

The next generation (5G) wireless communications system will begin their deployment in 2020. 5G is designed to support various applications subject to stringent technical requirements [1]. For example, to support data-intensive applications, including steaming video, online gaming, and virtual reality, 5G promises 1000 times higher data rate (1 to 10 Gb per second) compared with current 4G systems. Besides human-centric applications, machine-type communications, including Internet of Things and self-driving cars, are also important application scenarios of 5G. Since machine-type communications may involve a massive number of devices, such as sensors or actors, it is expected that 5G should be able to support 1 million connections per square km. In addition, for securing latency-critical applications (such as car-to-car communications or industrial control), 5G is required to provide 10 times less latency (1 millisecond) than 4G.

1.1.2 Spectrum and energy considerations

1.1.2.1 Spectrum efficiency is a key

Radio spectrum is probably the most important resource for wireless communications. However, wireless spectrum has almost been assigned to different wireless applications (such as telecommunications, televisions, radio, radar, navigation, etc.) [2]. It is very difficult and expensive to obtain additional spectrum bands for 5G use. For example, a 70 MHz spectrum band (at 600 MHz) that was utilized for television broadcasting

has been reassigned for 5G use with 19.8 billion dollars cost in spectrum auction [3]. Although it is promising[1] to obtain several new spectrum bands from mmWave range (24-84 GHz), due to poor propagation characteristics, mmWave bands are mainly exploited to meet extreme bandwidth requirements from local traffic "hot spots" (such as stadiums or urban areas). Many 5G application scenarios, such as connecting widely deployed devices, providing necessary capacity and supporting user mobility, mainly rely on spectrum below 6 GHz [5]. In summary, the development of 5G is constrained under spectrum crunch (especially sub 6 GHz), and therefore, improving spectrum use efficiency is a key for the success of 5G.

1.1.2.2 Energy efficiency ensures affordability and sustainability

The efficient use of energy is another important design consideration of 5G for decreasing costs and environmental impacts. Specifically, it is reported that "currently 3% of the world-wide energy is consumed by the Information & Communications Technology infrastructure that causes about 2% of the world-wide CO_2 emissions, which is comparable to the world-wide CO_2 emissions by airplanes or one quarter of the world-wide CO_2 emissions by cars" [6]. Under the dramatic increase of services and data demands, if not carefully designed, 5G may further increase energy consumption and carbon footprint. For example, it is expected that, without energy efficient plan and management, communication industry may use as much as 51% of global electricity in 2030 under the current growth trend of devices, infrastructures and data traffic [7]. Therefore, improving energy use efficiency is crucial for affordability and sustainability of 5G.

1.1.3 Cognitive radio: recycle spectrum from primary users

Although an apparent spectrum scarcity is indicted in [2], deeper investigations reveal encouraging results. Specifically, several field experiments suggest that, at given time and location, large portion (between 80% and 90%) of the licensed spectrum is idle [8]. The result is an underutilization of radio spectrum. These temporally unused spectrum bands, a.k.a., spectrum holes, arise because most of primary (licensed) user access is bursty.

[1] As an example, the Federal Communications Commission is working on allocating 5 licensed spectrum bands (24.25-24.45, 24.75-25.25, 27.5-28.35, 37-38.6 and 38.6-40 GHz) for 5G applications [4].

Cognitive radio, proposed in [9], is a software-defined radio that has the ability to perceive environments, adjust operating configuration and learn from experience. These features make cognitive radio an ideal solution to recycle temporally available spectrum holes from primary systems (radar, television systems, etc.), which is known as dynamic spectrum access. Specifically, by sensing spectrum bands (i.e., channels), cognitive/secondary nodes detect primary user activities, and access a channel if it is not occupied. This dynamic access scheme would provide 5G an efficient and flexible solution to further exploit underutilized spectrum and improve overall spectrum use efficiency and communication capacity.

1.1.4 Energy-harvesting: recycle energy from environments

One of the energy consumption drivers of 5G will be the massive number of machine type communication devices, such as wireless sensors (which are "eyes" and "ears" for the Internet of Things and other applications). Although these devices generally operate with low power, their massive population (expected to reach 21 billions in 2020) will make the total energy consumption non-negligible. Furthermore, compared with the operation stage, production and deployment of these devices may cause the same order (if not higher) energy consumption. Specifically, semiconductor devices need highly complicated manufacturing process. Therefore, (unlike other electrical machines, such as cars and refrigerators) for these high-tech devices, the energy consumption in the production stage could be even higher than that of the operation stage [10]. What's worse, since in many application, these devices are deployed extensively in large areas for environment monitoring, replacing or recharging battery may not be an option. Therefore, new devices have to be periodically redeployed to compensate node losses due to drained batteries.

Energy-harvesting technology provides a solution to the aforementioned problems. Energy-harvesting devices are able to recycle energy from ambient environments, such as sunlight, indoor illumination, electromagnetic energy, etc [11]. The key component of an energy-harvesting device is called an energy-harvesting transducer, which can convert other types of physical quantities, such as pressure or brightness, into electricity. As reported in [12], an energy harvester is capable to produce 140 mW of power with small enough size for portable electronic devices (note that the trans-

mit power of a wireless sensor node generally varies from 100 μW to 100 mW [13]). Hence, equipping low-power communication devices with energy harvester leads to the self-sufficiency of energy [13], which could cut off the energy expenditure to power these devices. Further, empowered with energy-harvesting, the life time of sensors is not constrained by battery capacity, but reaches the limits of hardware [14], which greatly reduces the energy cost for reproduction and redeployment.

1.2 Management of Spectrum Holes

The core idea of cognitive radio (dynamic spectrum access) is to access primary user spectrum without causing interference. However, this goal is challenging due to the dynamic activities of primary users, as the spectrum holes created by the inactivity of primary users thus change over time and location. Therefore, the use of spectrum holes by cognitive nodes must be managed carefully.

1.2.1 Spectrum sensing and access

Reliable spectrum sensing enables secondary nodes to correctly identify spectrum holes while minimizing interference to primary users. For example, in IEEE standard 802.22, the mis-detection probability for detecting licensed television transmitters should be not larger than 0.1 [15]. In addition, agility is also desired. Faster spectrum sensing provides more transmission time over sensed spectrum holes, which increases data throughput [16]. Various signal processing techniques have been considered [17] for handling the reliability and agility trade-off.

In multi-channel systems, it is generally impractical to sense all channels simultaneously due to hardware limitations. When a cognitive node can only sense and access one channel at a moment, the node needs to decide by what order the channels should be scanned, and in addition, given a sensed spectrum hole, whether or not to stop scanning and exploit the spectrum hole [18–20]. When primary activities have temporal correlation [21, 22], sensing observations can be used for predicting primary activities (i.e., future channel status). In this case, spectrum sensing decision may need to balance between the two goals: identifying idle channel for immediate use and tracking primary activities to guide future decisions.

1.2.2 Cooperative spectrum sensing

Due to wireless propagation impairments such as shadowing and fading, spectrum sensing by an individual node may not meet reliability requirements within an acceptable sensing time duration. Specifically, due to the shadowing effect caused by objects obstructing, primary signal may be largely blocked at certain cognitive nodes. In addition, when wireless channel (between a primary transmitter and a cognitive node) undergoes deep fading, the received primary signal can be weak. Moreover, when the coherence time is large, the fading may last for long time. All these factors considerably increase required time to accumulate week signal for reliable detection.

Fortunately, the sensing reliability and agility can be improved by intelligently fusing sensing observations from multiple, spatially-separate secondary nodes, which is known as cooperative spectrum sensing (CSS) [23]. The fundamental reason is that the simultaneous deep fading and shadowing of channels to multiple users is highly unlikely. Various data fusion techniques have been considered [16, 24–26], which can be roughly categorized as soft-combining schemes (combining raw received signals) and hard-combining schemes (combining binary decisions).

1.2.3 Cooperative transmission

Cognitive radio networks can be benefited by relaying data with intermediate nodes, known as cooperative transmission. First, for large networks, cognitive nodes at different locations may observe different spectrum holes. Two cognitive nodes cannot establish direct communication link, if there is no common spectrum hole between them. However, in the context of cooperative transmission, communication is possible whenever there exists a multi-hop-multi-channel path between two communicating nodes. This significantly reduces the likelihood of experiencing communication breakdown due to "spectrum outage" [27]. In addition, with clever relay selection and signal combining, a cognitive node is able to transmit with a lower power compared with the direct-link case [28–30]. This improves energy efficiency, and also reduces potential interference to primary users.

1.3 Management of Harvested Energy

Energy-harvesting-empowered wireless nodes are generally energy-constrained (i.e., with finite battery capacity). Therefore, a node must carefully control its energy consumption to satisfy certain quality-of-service requirement while avoiding energy depletion.

1.3.1 Handling dynamic battery status

In conventional energy-constraint systems (without energy-harvesting), the purpose of energy management is to efficiently use a finite energy budget. In this case, it is relatively easy to predicate the evolution of battery status, and to design a power control policy. However, in energy-harvesting case, randomness of energy replenishment leads to dynamic changes of battery status even with constant power allocation. This considerably complicates energy management. Specifically, a control policy needs to be designed under long-term performance criteria while taking energy randomness into consideration [31–37].

1.3.2 Incorporating data-centric consideration

One of the major energy-harvesting application scenarios is to power wireless sensor networks, which include spatially-dispersed autonomous sensors for monitoring physical or environmental conditions (such as temperature, moisture, etc.) [38, 39]. Wireless sensor networks are intrinsically "data-centric". For instance, data packets may be temporally and spatially redundant, since packets collected within a narrow temporal-spatial-window may correspond to an identical environmental condition or event. In addition, for many applications, data packets may associate with different priorities. For example, data packets containing information of enemy attacks [40] or fire alarms [41] may have higher priority. In summary, rather than maximizing throughput or reducing transmission time (which are targets in conventional communication systems), energy should be efficiently utilized from a data-centric point of view [42], and possible solutions are:

- prioritization: Prioritization considers the differentiation of data transmission by paying more attention to high priority packets. As an example, packet priority

is incorporated in a medium access control protocol [43], which provides low-latency delivery for high priority packets.

- aggregation: Aggregation reduces the total transmission load by compressing data redundancy when packets are routing from sources to destinations. For example, in a multi-source-single-destination case, data aggregation is achieved by searching an aggregation tree that is with as small amount of edges as possible [44].

1.3.3 Simultaneous information and power transfer

Radio frequency-based energy-harvesting technology harvests energy from ambient radio signal, which provides a convenient solution to power ultra-low-power devices. Compared with other sources (such as solar and wind), harvested energy from radio signal tends to be more stable and predictable [45]. Interestingly, since radio signal is exactly the information carrier for wireless systems, a radio frequency-based energy-harvesting node is able to replenish energy and get information from the same radio signal, known as simultaneous information and power transfer. However, due to circuit limitation, energy harvester is not yet able to extract energy from decoded radio signal [46]. That is, signal energy is lost after information is decoded, and vice versa. This introduces a fundamental information-power trade-off, which has been considered in receiver architecture design, user scheduling and others (see [45] and references therein).

1.4 Thesis Motivation and Contributions

In the following, we first briefly introduce machine learning. Then the potentials of applying machine learning for wireless spectrum and energy management is discussed, from which we propose the research subjects of this thesis.

1.4.1 Brief introduction of machine learning

Machine learning is a methodology of solving problems with data-driven computer programs. In other words, problem solutions are not hard-coded but incrementally learned from training data, which provides flexibility and adaptability. Depending

on targeting problems, machine learning can be roughly classified as three categories: supervised learning, unsupervised learning and reinforcement learning.

1.4.1.1 Supervised learning

Supervised learning can be considered as fitting a function that best matches given input-output examples and generalizes to unseen data. When the values of output only take from a finite set, the learning task is called classification, and each output is interpreted as the associated "class" of the input. When outputs take continuous values, the learning task is called regression. There are various supervised learning algorithms, such as support vector machines (for classification) [47] and artificial neural networks (ANNs, for both classification and regression) [48, Chapter 4] (also see Chapter 2.2)

1.4.1.2 Unsupervised learning

Unsupervised learning considers the discovery of data structures, where (unlike supervised learning) the training data does not contain targeted outputs. The output of supervised learning is certain "interesting pattern" of training data. Clustering analysis is one of the most common unsupervised learning tasks. It reveals data samples' underlying structure via clustering data samples into several groups based on certain similarity metric (see Chapter 2.3).

1.4.1.3 Reinforcement learning

Reinforcement learning (RL) considers optimal decision making under uncertainty (stochastic dynamic setting). An agent (decision maker) may receive random rewards from an "environment". The amount of reward depends on the taken action and "state" of the environment. But this action also affects the environment, i.e., a state changes and randomly transits to a next state after an action applied (defined by state transition probability). The goal of the agent is to collect as many rewards as possible in a given period, by considering the stochastic and dynamic feature of the environment. This problem setup can be mathematically modeled as Markov decision processes (MDPs) [49] (Chapter 2.1).

1.4.2 Machine learning approaches for spectrum and energy intelligence

Trading with computation and data, machine learning provides a flexible and generic paradigm for solving complicated problems, which perfectly matches the need of spectrum and energy management in cognitive and energy-harvesting wireless systems. First, due to the evolution of computing technologies, computing hardwares and softwares are increasingly powerful. Even mobile devices are able to execute reasonably complex algorithms. In addition, many application scenarios of cognitive and energy-harvesting wireless systems are data-intensive. For example, an energy-harvesting-powered cognitive node may need to handle various types of information, including wireless channel state, packet buffer, battery status, harvested energy, spectrum sensing observations, primary activities, etc. These data offer opportunities to exploit learning algorithms for better analysis, prediction and control of wireless systems. Lastly, since in many cases, learning algorithms can directly process raw data (without knowing underlying distributions), this could free system designer from estimating and validating stochastic models of related random variables.

With aforementioned motivations, this thesis addresses related spectrum and energy management issues with machine learning as primary tools. Three research contributions are made.

- Joint sensing-probing-transmission control for a single-channel energy-harvesting cognitive node:

  Spectrum sensing is essential for a cognitive node to discover spectrum holes. In addition, to achieve high data rate with an identified spectrum hole, the node may like to transmit as large power as possible. However, when the cognitive node is (solely) powered by energy-harvesting devices, constantly performing above operations may quickly drain the node's energy.

  For example, when the channel is not likely to be free, the node may decide not to sense for energy saving. Similarly, the node may want to transmit when the channel condition is good, and not to transmit if the channel condition is poor. That is, the node may need to adapt its transmit power depending on channel state information (CSI), which can be known via CSI estimation, referred to as channel probing. Channel probing involves the cognitive node transmitting

a pilot sequence, which enables the receiving node to evaluate the channel and provide CSI feedback to the transmitter.

In summary, subject to energy status and belief on channel availability, the node needs to decide whether or not to perform spectrum sensing and channel probing; and given probed CSI, the node needs to further decide on the transmit power. This control problem is modeled as a two-stage MDP, and the optimal control policy is further solved via a learning algorithm.

- Optimal selective transmission for an energy-harvesting sensor:
  In the context of wireless sensor networks, the prioritization paradigm (see Section 1.3.2) is considered for data-centric transmission design. Specifically, a sensor can drop low-priority packets (for saving and accumulating energy) when current available energy is limited, which allows the sensor to transmit more important packets in a long term. This is known as a selective transmission strategy.

  Obviously, to decide whether a packet should be sent or dropped, the packet's priority and the node's energy status should be considered. In order to incorporate wireless channel's effect on decision making, the third factor, CSI, is further exploited. Specifically, if a sensor decides to transmit a packet, it adjusts transmission power depending on CSI to avoid transmission failure. If channel fading is deep according to CSI, which increases the transmit power necessary to achieve reliable communication, the sensor may choose not to transmit.

  That is, we study the optimal selective transmission, where a node decides to send or not to send by considering energy status, data priority and fading status. This transmission control problem is modeled with MDP, and the optimal policy is derived via training an ANN.

- Cooperative spectrum sensing under heterogeneous spectrum availability:
  Most of existing works on CSS assume that all cognitive nodes experience the same spectrum availability (spectrum homogeneity assumption). This spectrum homogeneity assumption is reasonable, if primary users have very large transmission power, such as television broadcast systems, and/or small-scale cognitive

networks where all cognitive nodes are co-located.

Here, we consider CSS with heterogeneous spectrum availability, i.e., secondary nodes at different locations may have different spectrum statuses, which may occur when transmission power of primary users is small, and/or secondary networks have large geographical expansion. Under heterogeneous spectrum availability, there is still positive gain with sensing cooperation, since spatially proximate secondary nodes are likely to experience the same spectrum status. The challenge is how to model and exploit spatial correlation for efficient and effective sensing cooperation.

To address the above challenge, a cooperative spectrum sensing methodology is proposed. Specifically, spatial correlations among secondary users are approximately modeled as a Markov random field (MRF, see Chapter 2.3), and given cognitive nodes' data observations, sensing cooperation is achieved by solving a maximum posterior probability (MAP) estimator over the MRF. Under this methodology, three cooperative sensing algorithms are proposed, which are, respectively, designed for centralized, cluster-based, and distributed cognitive networks.

1.5 Thesis Outlines

The thesis is organized as follows. Chapter 2 provides the basic background of relevant machine learning approaches, including MDP, after-state, ANN and MRF. Chapters 3-5, respectively, address the three research topics. Chapter 6 concludes this thesis and discusses possible future research.

Chapter 2

Background

2.1 MDP and After-state

MDP and after-state are exploited in Chapters 3 and 4. These concepts are briefly discussed in this section.

2.1.1 Problem setting of MDP

MDP considers the optimal decision making under a stochastic dynamic environment. It is assumed that the environment can be fully described via a state s, with all states defined as state space $\mathbb{S}$. Facing a state s, an agent, i.e., a decision maker, can interact with the environment by applying an action a, with all available actions at s denoted as $\mathbb{A}(s)$. Therefore, for all states, the total available actions are $\{\mathbb{A}(s)\}_s$, which is named as action space. After an action a is applied on the environment of state s, the agent can get an instantaneous reward, which can be random and its expected value can be expressed as a reward function $r(s, a)$, which only depends on (s, a). The applied action, in return, affects the environment, and therefore, the state of the environment changes and transits to some other state s'. It is assumed that this transition is Markovian, i.e., the probability of transiting to a certain state only depends on current state and the action taken, which can be expressed as a state transition probability $p(s'|s, a)$. Therefore, the 4-tuple informa-

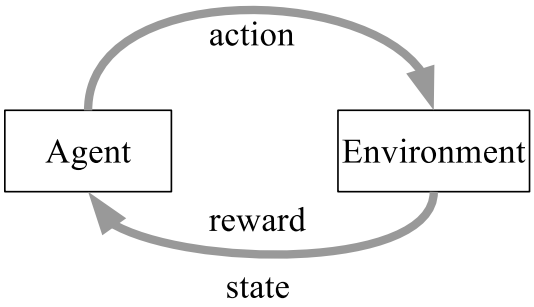


Figure 2.1: The problem setting of MDP

tion $\{\mathbb{S}, \{\mathbb{A}(s)\}_s, r, p\}$, namely, state space, action space, reward function and state transition probability, defines an MDP.

2.1.2 Standard results for MDP control

Let Π denote all stationary deterministic policies, which are mappings from $s \in \mathbb{S}$ to $\mathbb{A}(s)$. Given an MDP, it is sufficient to consider policies within Π. For any $\pi \in \Pi$, a function $V^\pi : \mathbb{S} \to \mathbb{R}$, representing accumulated rewards, for π is defined as follows,

$$V^\pi(s) \triangleq \mathbb{E}[\sum_{\tau=0}^{\infty} \gamma^\tau r(s_\tau, \pi(s_\tau)) | s_0 = s], \tag{2.1}$$

where s_τ denotes the state of time τ, $\gamma \in [0, 1]$ is a constant known as discounting factor, and the expectation $\mathbb{E}[\cdot]$ is defined by the state transition probability.

Among Π, there is an optimal policy $\pi^* \in \Pi$ that attains the highest value of V^π at all s, i.e.,

$$V^{\pi^*}(s) = \sup_{\pi \in \Pi} \{V^\pi(s)\}, \ \forall s.$$

In addition, π^* can be identified by the Bellman equation [49, p. 154], which is defined as follows,

$$V(s) = \max_{a \in \mathbb{A}(s)} \{r(s, a) + \gamma \mathbb{E}[V(s')|s, a]\}, \tag{2.2}$$

where s' means the random next state given current state s and the taken action a. Let $V^*(s)$, known as state value function, be the solution to (2.2). Then, the optimal policy $\pi^*(s)$ can be defined as

$$\pi^*(s) = \arg\max_{a \in \mathbb{A}(s)} \{r(s, a) + \gamma \mathbb{E}[V^*(s')|s, a]\}. \tag{2.3}$$

Furthermore, it is shown [49, p. 152] that

$$V^*(s) = V^{\pi^*}(s), \quad \forall s. \tag{2.4}$$

Therefore, $V^*(s)$ and $V^{\pi^*}(s)$ are used interchangeably.

2.1.3 MDP control based on after-states

Standard MDP results of Section 2.1.2 deal with the problems from the viewpoint of "state", which provides a generic solution for solving MDPs (i.e., solving the optimal

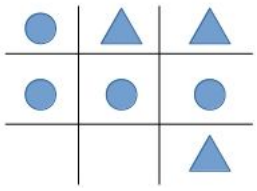


Figure 2.2: Paper-and-pencil game Tic-Tac-Toe

policy π^*). However, for certain problems, it is more natural and useful to define policies in terms of after-state, which is explained with the Tic-Tac-Toe (a child paper-and-pencil game) in the following.

Tic-Tac-Toe is a two-player game (also see [50, p. 10]), where two players mark a 3-by-3 grid in turn, and the player first places three of his/her marks in a horizontal, vertical, or diagonal row wins the game. Fig. 2.2 shows the case that the player with the "O" mark wins the game at his/her fourth marking.

From either player's point of view, the playing of Tic-Tac-Toe can be modeled as an MDP. Specifically, before the player's each marking, the configuration of marks in the grid can be viewed as a state s. At a state, empty spaces defined all possible action $\mathbb{A}(s)$ for the player. After the player's action applied, his opponent replies, which gives another state. Define the reward of final winning marking (the action immediately leads to a win) as 1; while, all other immediate actions have reward 0. Also set γ to 1 and treat winning states as absorbing states. Then, we can interpret $V^\pi(s)$ (2.1) as the probability to win by following policy π at state s. Furthermore, $V^*(s)$ (2.4) can be interpreted as the highest probability to win given state s (by considering all possible policies within Π). Finally, the optimal action $\pi^*(s)$ (2.3) reduces to

$$\pi^*(s) = \arg\max_{a \in \mathbb{A}(s)} \{\mathbb{E}[V^*(s')|s, a]\},$$

suggesting that, at each state, the player should take the action that, in expectation, gives the best next state (that has the highest chance to win).

Above analysis is reasonable. However, when playing Tic-Tac-Toe, we rarely determine our actions by dealing with states. In fact, we evaluate our strategies in terms of mark positions (defined as after-states) after our action applied, but before our opponent's marking (also see [50, p. 156]). The reason is that we know exactly what after-state will be after our actions at a certain state. (See Fig. 2.3 for two

examples of the relationship between after-states and state-action pairs for the player with the "O" mark.) In addition, we have a sense of the wining chance of different resulted after-states (which is estimated with our experience and reasoning). Let's denote the estimated chance of winning at after-state p as $J^*(p)$. Therefore, when human play the game, at a state s, we simply choose the action that leads to the after-state with highest value $J^*(\cdot)$, i.e.,

$$\pi^*(s) = \arg\max_{a \in \mathbb{A}(s)} \{J^*(\varrho(s,a))\},$$

where $\varrho(s,a)$ means the after-state after action a is applied upon state s.

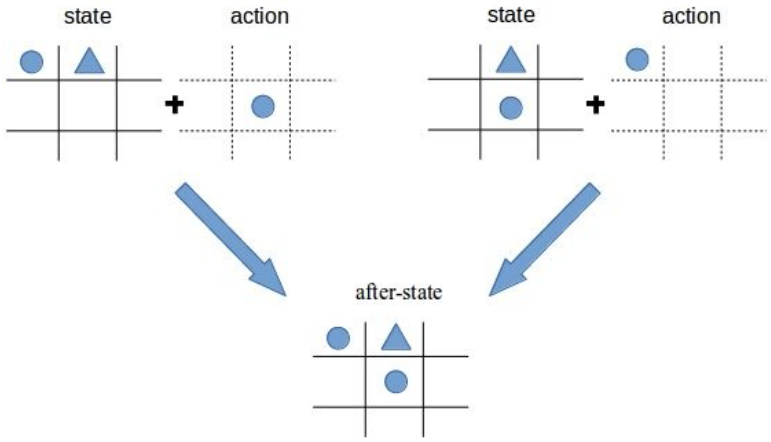


Figure 2.3: From state-action pairs to after-states

Furthermore, from Fig. 2.3, we can see that multiple state-action pairs may correspond to one after-state, which can potentially reduce storage space and simplify the problem (as we will show in Chapters 3 and 4). Besides this, in Chapter 3, we show that after-states are useful in theoretically establishing the optimal policy and developing learning algorithms. In Chapter 4, we show that after-states can facilitate the problem analysis and the discovery of optimal policy structure.

2.2 Artificial Neural Network

ANNs [48, Chapter 4] are exploited in Chapter 4 (for estimating a one-dimensional differentiable function). The problem setting of ANNs is to learn a function that best matches given input-output examples. In the following, we discuss how to train an ANN given multi-dimensional input-output pairs $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_i$, where $\boldsymbol{x}_i \in \mathbb{R}^M$ and

$\boldsymbol{y}_i \in \mathbb{R}^N$.

2.2.1 Neural network as a function

An ANN is a weighted directed graph (consisting of nodes and edges). Normally, the graph has a layered structure. That is, nodes (known as neurons) are grouped into L ordered layers. Neurons of the lth layer are connected to neurons of the $(l+1)$th layer, for $1 \leq l \leq L-1$, with weight w_{ij}^{l+1} between the ith neuron of the lth layer and the jth neuron of the $(l+1)$th layer. The first layer, called input layer, has M neurons. The last layer, called output layer, has N neurons. All layers in between are called hidden layers, where the lth layer $(1 < l < L)$ has K^l neurons ($\{K^l\}_l$ are hyperparameters). As an example, a 3-input-2-output ANN is shown in Fig. 2.4, which has a single hidden layer with 4 neurons.

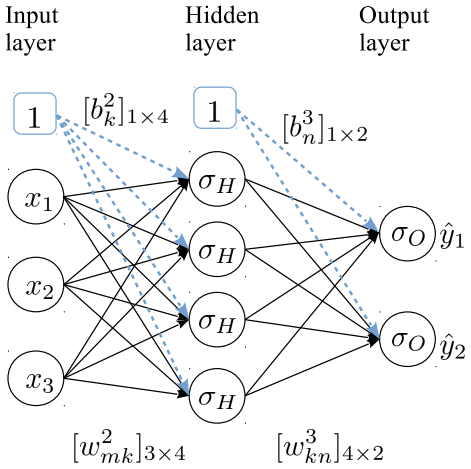


Figure 2.4: A three-layer neural network

Given graph structure and parameters, an ANN presents a function $f(\cdot)$. That is, for a given input $\boldsymbol{x} \in \mathbb{R}^M$, an ANN estimates the associated output as $\hat{\boldsymbol{y}} = f(\boldsymbol{x}) \in \mathbb{R}^N$, which is detailed as follows. First, neurons of the input layer output a "signal" vector equaling $\boldsymbol{x}$. Specifically, the mth neuron of the input layer outputs a "signal" x_m, where x_m is the mth component of $\boldsymbol{x}$. Signals generated from the input layer pass through edges (and weighted by corresponding weights) and reach the 2nd layer. Neurons of the 2nd layer regenerate signals based on received signals (which is discussed later), and pass them to the 3rd layer. The process continues until output-layer neurons generate signals $\{\hat{y}_i\}_{i=1}^N$. Then, vector $\hat{\boldsymbol{y}} = [\hat{y}_1, \cdots, \hat{y}_N]$ is

treated as the ANN's estimated output associated with the input $\boldsymbol{x}$.

Here, we present the details of signal passing and regeneration in ANNs. Denote z_i^l as the output of the ith neuron of the lth layer. Obviously, we have $z_i^1 = x_i$, $\forall\, 1 \leq i \leq M$. For $l \geq 2$, the ith neuron of the lth layer receives signal vector $\{w_{ki}^l \cdot z_k^{l-1}\}_k$. It regenerates a signal as $\sigma_i^l(\sum_k w_{ki}^l \cdot z_k^{l-1} + b_i^l)$, where $\sigma_i^l(\cdot)$ and b_i^l are so-called activation function and bias parameter associated with the ith neuron of the lth layer. Theoretically, different neurons (of hidden layers and the output layer) can have different activation functions. In practice, it is usually sufficient to assign all neurons of hidden layers a function $\sigma_H(\cdot)$, which is required to be non-linear and differentiable. A widely used $\sigma_H(\cdot)$ is the sigmoid function [48, Chapter 4], i.e.,

$$\sigma_H(x) = \frac{1}{1 + e^{-x}}, \tag{2.5}$$

which is shown in Fig. 2.5. The choices of activation functions of output-layer neurons

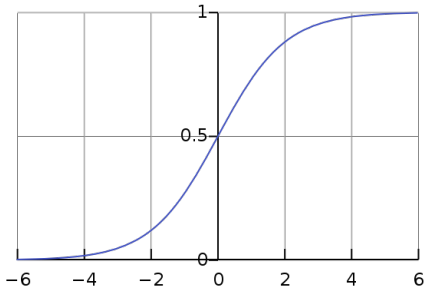


Figure 2.5: Sigmoid function

depend on the desired range of output values. In the simplest case, where each component of output $\hat{y}_i$ takes all real values, we can set $\sigma_O(x) = x$ (known as linear activation function).

2.2.2 Train neural networks with labeled data

In ANNs, the types of activation functions, layers of hidden layers, and number of neurons of each hidden layers are all hyperparameters (parameters do not change during learning process). The choices of hyperparameters are empirical, and usually done via trials. The parameters of ANNs are weights $\{\boldsymbol{w}^l\}_l$ and biases $\{\boldsymbol{b}^l\}_l$, which are adjusted to make outputs match given data.

Specifically, for a batch of given input-output pairs $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_i$, an ANN generates a batch of outputs $\{\hat{\boldsymbol{y}}_i\}_i$, where $\hat{\boldsymbol{y}}_i$ is the output given $\boldsymbol{x}_i$. A loss function $\mathcal{L}$ is

constructed to measure differences between $\{\hat{\boldsymbol{y}}_i\}_i$ and $\{\boldsymbol{y}_i\}_i$. For example, a quadratic loss function is commonly used, i.e.,

$$\mathcal{L}(\boldsymbol{w}^l, \boldsymbol{b}^l) = \sum_i ||\boldsymbol{y}_i - \hat{\boldsymbol{y}}_i||_2^2$$

(for given data $\mathcal{L}$ is a function of parameters $\{\boldsymbol{w}^l\}_l$ and bias $\{\boldsymbol{b}^l\}_l$).

Therefore, the goal of learning is to minimize $\mathcal{L}$ by adjusting weights $\{\boldsymbol{w}^l\}_l$ and bias $\{\boldsymbol{b}^l\}_l$. Note that the exact minimization is difficult, since $\mathcal{L}$ is not convex. Nevertheless, it has been empirically observed that good performance can be achieved with gradient based searching (for example, with gradient descent algorithms). Due to the layered structure of ANNs, derivatives $\partial\mathcal{L}/\partial\boldsymbol{w}^l$ and $\partial\mathcal{L}/\partial\boldsymbol{b}^l$ can be efficiently computed via applying the chain rule (see backpropagation algorithms for details [51]).

2.3 Markov Random Field

In Chapter 5, MRF is exploited in the context of CSS. In this section, the basis of MRF is illustrated with an image segmentation example.

MRFs are widely used in computer visions. Its applications often arise from a common feature that an image pixel is usually correlated with others in a local sense. For simplicity, let us consider gray-valued images, and the task is to segment an image into two parts: "object" and "background". That is, the segmentation process returns a binary image. When object and background have distinct gray values, it is a good idea to segment the image via finding a gray value threshold. For example, it is expected that we can get a good segmentation of Fig. 2.6(northwest)[1] with a single threshold.

However, if the image is contaminated with noise, thresholding segmentation may perform poorly. As an example, Fig. 2.6(northeast) shows an image that is contaminated by salt-and-pepper noise. Fig. 2.6(southwest) shows the thresholding segmentation result with gray value threshold equaling 154 (an optimal value obtained via the Otsu's method [52]).

To deal with the noise and improve the segmentation result, there exist many methods. One idea is to incorporate an intuition that spatially close pixels are likely

[1] This picture, named as "Rubycat.jpg", is obtained at `http://stupididy.wikia.com/wiki/File:Rubycat.jpg`.

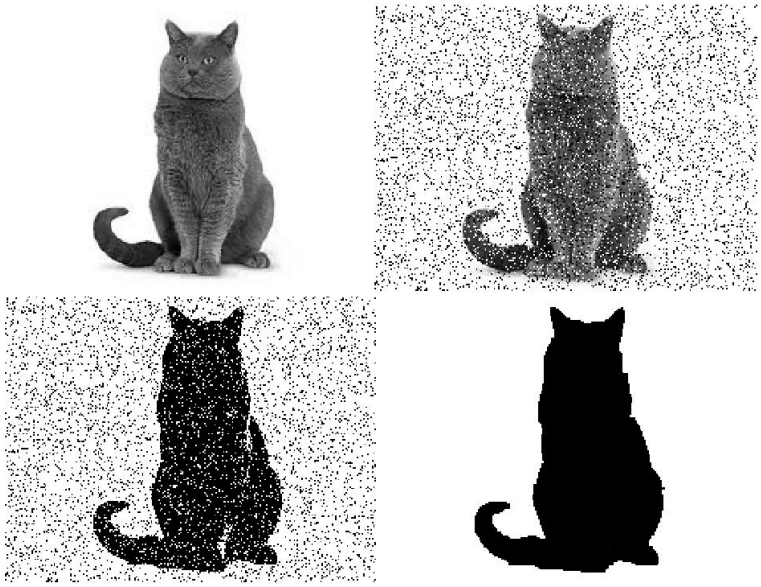


Figure 2.6: An binary image segmentation example; northwest: original image; northeast: image contaminated by salt-and-pepper noise; southwest: thresholding segmentation; southeast: segmentation incorporating MRF.

to belong to the same category (object or background). As a simple but useful method, MRF can be used to model this intuition, which is described as follows.

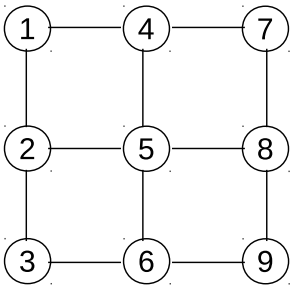


Figure 2.7: Graph $\mathcal{G}$ for a nine-pixel image

Let x_i denote the label of ith pixel, and $x_i = 1/0$ represents that the ith pixel belongs to object/background. Then, for each pixel, we define its neighbors as its above, below, left and right pixels (known as 4-neighbors relationship). From this neighboring relationship, we can define a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$: the set of nodes $\mathcal{V} = \{1, \cdots, N\}$, respectively, represent pixel labels $[x_1, \cdots, x_N] \triangleq \boldsymbol{x}$; the set of edges $\mathcal{E} = \{(i, j) | \text{if the } i\text{th and } j\text{th pixels are neighbors}\}$. An example of $\mathcal{G}$ for a nine-pixel image is shown in Fig. 2.7.

Here, an MRF is constructed from $\mathcal{G}$. For each edge of $(i, j) \in \mathcal{G}$, we heuristically

define a potential function $\phi(x_i, x_j)$ as Table 2.1, which reflects our belief that $x_i = x_j$

Table 2.1: Pairwise potential function $\phi(x_i, x_j)$

x_i \ x_j	0	1
0	36	14
1	14	36

is more likely to happen than $x_i \neq x_j$. Finally, an MRF $\Phi(\boldsymbol{x})$ over $\boldsymbol{x}$ is defined as (2.6)

$$\Phi(\boldsymbol{x}) = \prod_{(i,j)\in\mathcal{E}} \phi(x_i, x_j). \tag{2.6}$$

$\Phi(\boldsymbol{x})$ is used to approximately model the (unnormalized) joint prior distribution over $\boldsymbol{x}$.

Now, we incorporate the constructed MRF (2.6) with the thresholding segmentation for better result. Denoting the gray value of the ith pixel as y_i, we define a data likelihood function $f(y_i|x_i)$ as

$$f(y_i|x_i) = \begin{cases} 1, & \text{if } x_i = 1,\ y_i < 154; \\ 1, & \text{if } x_i = 0,\ y_i \geq 154; \\ 0, & \text{otherwise.} \end{cases}$$

Then, we define an optimization problem as[2] (see Chapter 5 for solving methods)

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{x}} \left\{ \prod_{i\in\mathcal{V}} f(y_i|x_i) \prod_{(i,j)\in\mathcal{E}} \phi(x_i, x_j) \right\}, \tag{2.7}$$

which computes the MAP given data likelihood functions and MRF as a prior. The result $\boldsymbol{x}^*$ from (2.7) then is taken as image segmentation result, which is shown in Fig. 2.6(southeast). It can be seen that the noise has been perfectly eliminated.

2.4 Summary

In chapter, we briefly introduced several concepts in machine learning field, including MDPs, after-state, ANNs and MRFs, which will be exploited in following chapters for spectrum and energy management of wireless systems.

[2] Note that, if $\phi(1,1) = \phi(1,0) = \phi(0,1) = \phi(0,0)$, $\boldsymbol{x}^*$ computed via (2.7) reduces to the thresholding segmentation.

Chapter 3

Sensing-Probing-Transmission Control for Energy-Harvesting Cognitive Radio

3.1 Introduction

Energy-harvesting and cognitive radio aim to improve energy efficiency and spectral efficiency of wireless networks. However, the randomness of energy-harvesting process and uncertainty of spectrum access introduce unique challenges on the optimal design of such systems.

Specifically, rapid and reliable identification of spectrum holes [53, 54] is essential for cognitive radio. Furthermore, when accessing spectrum holes, a cognitive node or secondary user (SU) must adapt its transmit power depending on channel fading status, which is indicated by channel state information (CSI) [18–20]. The CSI estimation process is referred to as channel probing: it involves the SU transmitting a pilot sequence, which will enable the secondary receiving node to estimate the channel and provide a CSI feedback to the transmitter node[1]. Note that this channel probing takes place on a perceived spectrum hole, but due to spectrum sensing errors, the SU may have misidentified the spectrum hole. In that case, the PU will be harmed. In other words, interference on PUs can occur during both channel probing and data transmission stages. In summary, an SU not only must minimize the harm to PUs but also perform spectrum sensing, channel probing, and adaptive transmissions subject to the available harvested energy.

[1]See [55–57] and references therein for pilot designs.

Therefore, with low energy availability, an SU may not perform all of these operations. For instance, if the channel is likely to be occupied, the SU may decide to not sense it and save the energy expense. Moreover, in a deep fading channel, the SU decides not to transmit. Furthermore, since sensing, probing, and transmitting consume the harvested energy, these operations are coupled. Therefore, in energy-harvesting cognitive radio systems, it is important to jointly control the processes of sensing, probing, and transmitting while adapting to fading status, channel occupancy, and energy status.

3.1.1 Related works

Sensing and/or transmission policies for energy-harvesting cognitive radios have been extensively investigated [58–70]. We next categorize and summarize them.

3.1.1.1 Optimal sensing design

Works [58–62] focus on optimal sensing, but not data transmission. Sensing policy (i.e., to sense or not) and energy detection are considered for single channel systems under an energy causality constraint [58, 59]. Specifically, in [58], the stochastic optimization problem for spectrum sensing policy and detection threshold with energy causality and collision constraints is formulated. In [59], sensing duration and energy detection threshold is jointly optimized for a greedy sensing policy. Work [60] considers multi-user multi-channel systems where the SUs have the capability to harvest energy from radio-frequency signals. Thus, energy can be harvested from primary signals. Balancing between the goals of harvesting more energy (from busy channels) and gaining more access opportunities (from idle channels), work [60] considers the optimal SU sensing scheduling problem. In cooperative spectrum sensing, the joint design of sensing policy, selection of cooperating SU and optimization of the sensing threshold has been studied [61]. This work is extended in [62] where 1) SUs are able to harvest energy from both radio frequency and conventional (solar, wind and others) sources, and 2) SUs have different sensing accuracy.

3.1.1.2 Optimal transmissions

If SUs have side information to access spectrum, optimal transmission control is desirable [63, 64]. Specifically, work [63] considers data rate adaptation and channel allocation for an energy-harvesting cognitive sensor node where channel status is provided by a third-party system (which does not deplete energy from the sensor node). Joint optimization of time slot assignment and transmission power control in a time division multiple access (TDMA) system is considered [64]. Here, the SUs use the underlay spectrum access (they can transmit even if the spectrum is occupied, provided interference on PUs is below a certain threshold [71]).

3.1.1.3 Joint design with static channels

Joint sensing and transmission design for static wireless channels has been considered [65–68]. Specifically, joint optimization of sensing policy, sensing duration, and transmit power is considered [65]. Similarly, joint design of sensing energy, sensing interval, and transmission power is considered in [66]. In [67], an energy half-duplex constraint (an SU cannot sense or transmit while harvesting energy) is assumed. To balance energy-harvesting, sensing accuracy, and data throughput, work [67] optimizes the durations of harvesting, sensing, and transmitting. In [68], a similar harvesting-sensing-transmission ratio optimization problem was considered, where SUs can harvest energy from radio frequency signals and the primary users' transmissions do follow a times-slot structure (thus, channel occupancy status may change anytime).

3.1.1.4 Joint design with fading channels

Work [69] considers multiple channels and energy-harvesting cognitive nodes. This work takes an unusual turn in that channel probing takes place before channel sensing. This approach runs the risk of probing busy channels. When that happens, the pilots transmitted for the purpose of channel estimation will be corrupted due to primary signals and the pilots in turn may cause interference to primary receivers.

Reference [70] investigated a secondary sensor network with (1) multiple spectrum sensors (powered by harvested energy) (2) multiple battery-powered data sensors for data transmission, and (3) a sink node for data reception. The first problem is to

optimally schedule in order to assign spectrum sensors over channels for maximizing channel access. When the sensing operation identifies the free channels, the second problem is to allocate transmission time, power and channels among data sensors for minimizing energy consumption. In this work, CSI availability is assumed *a priori* (which implies an always-on channel probing without costing energy).

3.1.2 Problem statement and contributions

Joint design of energy-harvesting, channel probing, sensing and transmission, especially under fading channels has not been reported widely. For instance, to adapt the transmit power depending on fading status, channel probing is necessary, which can be conducted only if the channel is idle. Thus, the SU does not know fading status when it decides whether or not to perform spectrum sensing. However, this *sensing-before-probing feature* has not been captured in existing works.

To fill this gap, we investigate a single-channel energy-harvesting cognitive radio system. The single channel may be occupied by the PU at a time. If this is true, the SU has no access. At each time slot, the SU decides whether to sense or not, and if the channel is sensed to be free, the SU needs to decide whether to probe the channel or not. After a successful probe, the SU obtains CSI. With that, the SU needs to decide the transmit power level. To maximize the long-term data throughput, we consider the joint design of sensing-probing-transmitting actions over a sequence of time slots.

In order to make optimal actions, the SU must track and exploit energy status, channel availability and fading status. These variables change randomly and are also affected by the previous sensing, probing and transmitting actions. We cast this stochastic dynamic optimization problem as an infinite time horizon discounting MDP (see Chapter 2.1).

Although MDP is a standard tool, it should be carefully designed to capture the sensing-before-probing feature of the problem. Moreover, because the node may not have the statistical distributions of energy-harvesting process and channel fading, the optimal policy must be solved in face of this informational deficiency. Our main results are summarized as follows.

1. We devise a time-slotted protocol, where energy-harvesting, spectrum sensing,

channel probing and data transmission are conducted sequentially. We formulate the optimal decision problem as a two-stage MDP. The first stage deals with sensing and probing, while the second with the control of transmit power level. To the best of our knowledge, this is the first time for using a two-stage MDP (which can better capture the sensing-before-probing feature than a one-stage MDP ([65,72]) to model the control of sensing, probing and transmitting actions of the SU.

2. The optimal policy is developed based on an *after-state* (also called post-decision state, see Chapter 2.1) value function. The use of the after-state confers three advantages. First, it facilitates a theoretical establishment of the optimal policy. Second, storage space needed to represent the optimal policy is largely reduced. Third, it enables the development of learning algorithms.

3. The wireless node often lack the statistical distributions of harvested energy and channel fading. Thus, it must learn the optimal policy without this information. To do so, we propose a reinforcement learning algorithm. Reinforcement learning algorithms do not assume knowledge of an exact mathematical model of the MDP. The proposed reinforcement learning algorithm exploits samples of harvested energy and channel fading in order to learn the after-state value function. The theoretical basis and performance bounds of the algorithm are also provided.

The rest of this chapter is organized as follows. Section 3.2 describes the system model. The optimal control problem is formulated as a two-stage MDP in Section 3.3. In Section 3.4, the structure of the MDP is analyzed, and an after-state value function is introduced to simplify the optimal control of the MDP. In Section 3.5, a reinforcement learning algorithm is proposed to learn the after-state value function. The performance of the proposed algorithm is investigated in Section 3.6.

3.2 System Model

Primary user model: We consider a single-channel system, where the operation of PU is time-slotted. It may correspond to system with TDMA scheme embedded, such as

wireless cellular networks. The SU synchronizes with the PU, and also acts in the same time-slotted manner. The channel occupancy is modeled as an on-off Markov process (Fig. 3.1), which has been justified by field experiments, (see, e.g., [73]). The state $C = 1/0$ denotes channel availability/occupation. The state transition probabilities are p_{ij} for $i \in \{0, 1\}$ and $j \in \{0, 1\}$. We assume that the SU knows the transition probability matrix.

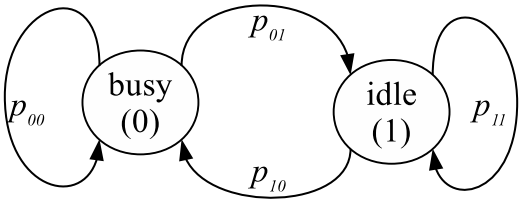


Figure 3.1: The PU's channel occupancy model

Channel sensing model: An energy detector senses the channel for a fixed sensing duration τ_S with a predefined energy threshold. The sensing result Θ infers the true channel state C. The performance energy detector is characterized by a false alarm probability $p_{FA} \triangleq \Pr\{\Theta = 0|C = 1\}$ and a miss-detection probability $p_M \triangleq \Pr\{\Theta = 1|C = 0\}$. Furthermore, $p_D \triangleq 1 - p_M$ and $p_O \triangleq 1 - p_{FA}$ represent the probability of correct detection of the PU and and of access to the spectrum hole, respectively. In practice, p_M must be set low enough to protect the primary system. For example, in cognitive access to television channels, p_M is less than 0.1 [15]. The true values of p_{FA} and p_M are known to the SU. Finally, each sensing operation consumes a fixed amount of energy e_S.

Sufficient statistic of channel state: Because the channel is monitored infrequently and there can be sensing errors, the true state C is unknown. At best, the SU can make decisions based on all observed information (e.g., sensing results and others). All such information can be summarized as a scaled sufficient statistic, known as the belief variable $p \in [0, 1]$, which represents the SU's belief in the channel's availability [22].

Energy-harvesting model: The SU harvests energy from sources such as wind, solar, thermoelectric and others [74]. The harvested energy arrives as an energy package at the beginning of each time slot. This package E_H has a probability density function (PDF) $\mathrm{f}_E(x)$. Across different time slots, E_H is an independent and identically distributed random variable. The SU node does not know this PDF. The SU is equipped

with a finite battery, with capacity $B_{\max}$. The amount of remaining energy in the battery is denoted as b.

Data transmission model: Here are the working assumptions. The SU always has data to send, and the standard block fading model applies. The channel gain between the SU and the receiving node is H, with PDF $\mathrm{f}_H(x)$, which is unknown to the SU. The SU adapts its transmission rate to different channel states by a choosing transmit power from a finite set of power levels. Channel probing is implemented as follows, and the SU sends channel estimation pilots if it senses that the channel is free, i.e., $\Theta = 1$.

- If the channel is indeed free ($C = 1$), the secondary receiving node will get the pilot sequence, estimate the channel state information (CSI) and send the CSI back to the SU through an error-free dedicated feedback channel. This receiver feedback (FB) is assumed to be always successful ($FB = 1$).
- If the channel is actually occupied, ($C = 0$), the pilot and primary signals will collide. This results in a failed CSI estimation, resulting in there being no feedback from the receiver ($FB = 0$).

The energy cost of channel probing is fixed at e_P, and the fixed time duration of probing is τ_P, whether $FB = 1$ or $FB = 0$.

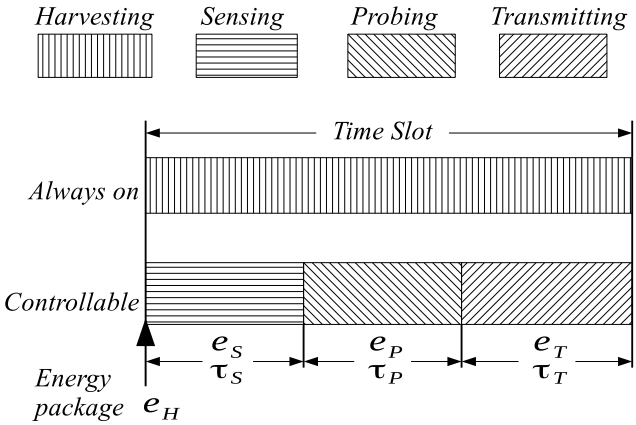


Figure 3.2: Time slot structure

MAC protocol: The time slot is divided into sensing, probing, and transmitting sub-slots (Fig. 3.2). At the beginning of the sensing sub-slot, the SU gets an energy package (harvested during the previous time slot). Based on the harvested energy e_H,

current belief p, and battery level b, the SU decides whether to sense the channel or not. If yes, and if the sensing output indicates a free channel, the SU decides whether or not to probe the channel. If yes, it will transmit channel estimation pilots to the receiver. And if the FB from the receiver is received, the SU gets the CSI. Then it needs to decide the transmission energy level to use, e_T taken from set $\mathbf{E_T}$ of a finite number of energy levels. And if any of the above conditions is not satisfied, the SU will remain idle during the remaining time slot, and then repeat the procedure at the next time slot.

Note: For the sake of presentation simplicity, we consider the case with single-channel and continuous data traffic. Our subsequently developed optimal control scheme and learning algorithms can be generalized to systems with multiple PU channels and bursty data traffic, which is discussed in Section 3.3.2.1.

3.3 Two-stage MDP Formulation

3.3.1 Finite step machine for MAC protocol

Here, we will use a finite step machine (see Fig. 3.3) to elaborate on the MAC protocol introduced in Section 3.2.

1. At the sensing step of slot t, the SU, initially with battery level b_t^S,[2] belief p_t^S, and harvested energy e_{Ht}, needs to decide whether to sense or not. If the SU chooses not to sense, it remains idle until the beginning of slot $t+1$, at which time it has energy $b_{t+1}^S = \phi(b_t^S + e_{Ht})$, where $\phi(b)$ is defined as:

$$\phi(b) \triangleq \max\{\min\{b, B_{\max}\}, 0\},$$

and the belief on channel occupancy changes to $p_{t+1}^S = \psi(p_t^S)$, where $\psi(p)$ is defined as:

$$\psi(p) \triangleq \text{prob}\{C_{t+1} = 1 | p_t = p\} = p \cdot p_{11} + (1-p) \cdot p_{01}.$$

2. If the SU chooses to sense, it will get a negative sensing result (i.e., $\Theta = 0$) with probability $1 - \text{p}_\Theta(p_t^S)$, where $\text{p}_\Theta(p)$ is defined as:

$$\text{p}_\Theta(p) \triangleq \Pr\{\Theta = 1 | p\} = p \cdot p_O + (1-p) \cdot p_M.$$

[2]Superscript S represents sensing, and subscript t means slot index.

Then it will remain idle until the beginning of slot $t+1$, and we have $b_{t+1}^S = \phi(\phi(b_t^S + e_{Ht}) - e_S)$, and $p_{t+1}^S = \psi(\mathrm{p}_N(p_t^S))$, where $\mathrm{p}_N(p)$ means the probability that the channel is idle given belief p and negative sensing result, i.e.,

$$\mathrm{p}_N(p) \triangleq \Pr\{C = 1 | p, \Theta = 0\} = \frac{p \cdot p_{FA}}{p \cdot p_{FA} + (1-p) \cdot p_D}.$$

3. If the SU chooses to sense, a positive sensing result ($\Theta = 1$) occurs with probability $\mathrm{p}_\Theta(p_t^S)$. It then reaches the probing step, and at this moment, the battery level is $b_t^P = \phi(\phi(b_t^S + e_{Ht}) - e_S)$,[3] and the belief transits to $p_t^P = \mathrm{p}_P(p_t^S)$, where $\mathrm{p}_P(p)$ is the probability that channel is idle, given belief p and positive sensing result, i.e.,

$$\mathrm{p}_P(p) = \Pr\{C = 1 | p, \Theta = 1\} = \frac{p \cdot p_O}{p \cdot p_O + (1-p) \cdot p_M}.$$

Next, the SU gets into the probing step.

1. At the probing step of slot t, if the SU with (p_t^P, b_t^P) chooses not to probe, it will remain idle until the beginning of slot $t+1$, and the battery level remains the same $b_{t+1}^S = b_t^P$, and the belief becomes $p_{t+1}^S = \psi(p_t^P)$.

2. If the SU chooses to probe, and after sending the pilots, there is probability $1 - p_t^P$ that the channel is busy, which will preclude FB from the receiver. And then the SU remains idle until the beginning of slot $t+1$ with battery $b_{t+1}^S = \phi(b_t^P - e_P)$ and belief $p_{t+1}^S = p_{01}$.

3. Having sent the pilots, the SU can get FB with probability p_t^P, and observe the channel gain information, $h_t \geq 0$. The SU then reaches the transmitting step. At this moment, the SU knows that the channel is free, i.e., $p_t^T = 1$,[4] and the remaining energy is $b_t^T = \phi(b_t^P - e_P)$.

Finally, at the transmitting step of slot t, the SU decides the amount of energy $e_T \in \mathbf{E_T}$ to use for transmission. After data transmission, it goes to the beginning of slot $t+1$ with battery $b_{t+1}^S = \phi(b_t^T - e_T)$ and belief $p_{t+1}^S = p_{11}$. Note that if $e_T = 0$, there will be no transmission.

[3] Superscript P represents probing.
[4] Superscript T represents transmitting.

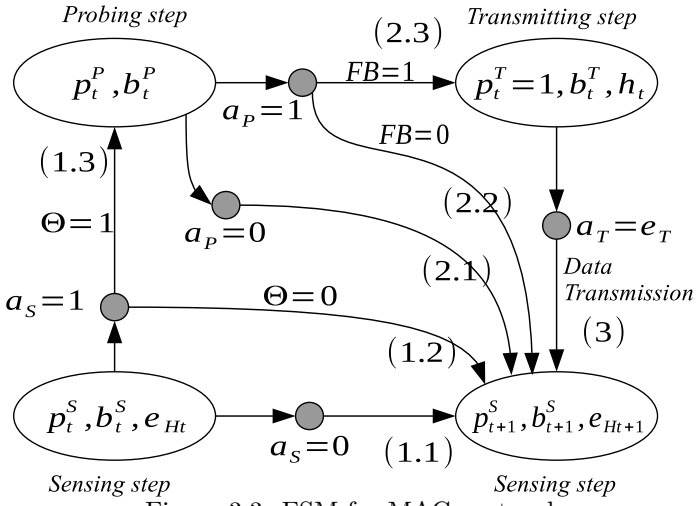


Figure 3.3: FSM for MAC protocol

3.3.2 Two-stage MDP

Based on the finite step machine, we will use an MDP to model the control problem. With s denoting a "state", a denoting an "action", an MDP is fully characterized by specifying the 4-tuple ($\mathbb{S}$, $\{\mathbb{A}(s)\}_s$, $\mathrm{f}(\cdot|s,a)$, $r(s,a)$), namely state space, allowed actions at different states, state transition kernel, and reward associated with each state-action pair, which are described as follows.

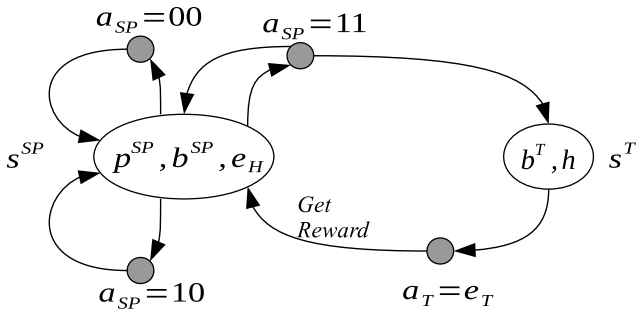


Figure 3.4: Two-stage MDP

1. To reduce the state space, we merge the sensing and probing steps into one stage (superscript SP) via jointly deciding these actions at the beginning of the sensing step. We also observe that, at the transmitting step, the belief is always equal to 1, and it is not necessary to represent it. Therefore, the state space $\mathbb{S}$ is divided into two classes: 1) sensing-probing state $s^{SP} = [b^{SP}, p^{SP}, e_H]$, with $b^{SP} \in [0, B_{\max}]$, $p^{SP} \in [0,1]$ and $e_H \in [0,\infty)$; and 2) transmitting state $s^T = [b^T, h]$, with $b^T \in [0, B_{\max}]$ and $h \in [0,\infty)$.

2. At a sensing-probing state s^{SP}, the full set of available actions are "not to sense", "to sense but not to probe", and "to sense and to probe if possible", i.e., we have $a_{SP} \in \mathbb{A}(s^{SP}) = \{00, 10, 11\}$. Here, the first digit presents the sensing decision, and the second digit presents the probing decision. If the available energy $\phi(b^{SP} + e_H)$ is less than $e_S + e_P$, the available action set $\mathbb{A}(s^{SP})$ is limited to $\{00, 10\}$; and if it is less than e_S, we have $\mathbb{A}(s^{SP}) = \{00\}$. And at a transmitting state s^T, the available actions are "transmission energy level to use", i.e., $a_T \in \mathbb{A}(s^T) = \mathbf{E_T}$.

3. $\mathrm{f}(\cdot|s, a)$ is a PDF of the next state[5] s' over $\mathbb{S}$ given initial state s and the taken action a. Denote $\delta(\cdot)$ as the Dirac delta function, which is used to generalize $\mathrm{f}(\cdot|s, a)$ to include discrete transition components. We can derive the state transition kernel following the description of the finite step machine. Starting from $s_t^{SP} = [p_t^{SP}, b_t^{SP}, e_{Ht}]$, it may transit to $s_{t+1}^{SP} = [p_{t+1}^{SP}, b_{t+1}^{SP}, e_{Ht+1}]$ or $s_t^T = [b_t^T, h_t]$ depending on chosen actions, with $\mathrm{f}(\cdot|s_t^{SP}, a_{SP})$ shown in (3.3), (3.4), (3.5) and (3.6) (on the top of next page). From transmitting state $s_t^T = [b_t^T, h_t]$, it can only transit to $s_{t+1}^{SP} = [p_{t+1}^{SP}, b_{t+1}^{SP}, e_{Ht+1}]$, with $\mathrm{f}(\cdot|s_t^T, a_T)$ shown in (3.7) (on the top of next page). Note that we treat $\mathrm{f}_H(x)$ and $\mathrm{f}_E(x)$ as generalized PDF's, which cover discrete or mixed random variables model for H and E_H.

4. At a sensing-probing state, because no data transmission has occurred yet, the reward is set to 0:

$$r(s_t^{SP}, a^{SP}) = 0. \tag{3.1}$$

At a transmitting state, the reward is achieved data rate, which is given by the Shannon formula. Therefore, the immediate reward is given by

$$r(s_t^T, a_T = e_T) = \tau_T W \log_2(1 + \frac{e_T h_t}{\tau_T N_0 W})\mathbf{1}(b_t^T \geq e_T), \tag{3.2}$$

where W is the channel bandwidth, N_0 is the thermal noise density and $\mathbf{1}(\cdot)$ is an indicator function.

We next place a technical restriction on the random variable H. Its interpretation is that, with any battery level and chosen transmission energy, the expected amount (and also squared amount) of sent data is bounded.

[5]Throughout this chapter, y' stands for the notation of y after one state transition in an MDP model.

$$\mathrm{f}(s_{t+1}^{SP}|s_t^{SP}, a_{SP}=00) = \delta(p_{t+1}^{SP} - \psi(p_t^{SP}))\delta(b_{t+1}^{SP} - \phi(b_t^{SP} + e_{Ht}))\,\mathrm{f}_E(e_{Ht+1}), \tag{3.3}$$

$$\begin{aligned}\mathrm{f}(s_{t+1}^{SP}|s_t^{SP}, a_{SP}=10) &= [(1-\mathrm{p}_\Theta(p_t^{SP}))\delta(p_{t+1}^{SP} - \psi(\mathrm{p}_N(p_t^{SP}))) + \mathrm{p}_\Theta(p_t^{SP})\delta(p_{t+1}^{SP} - \psi(\mathrm{p}_P(p_t^{SP})))] \\ &\quad \times \delta(b_{t+1}^{SP} - \phi(\phi(b_t^{SP} + e_{Ht}) - e_S))\,\mathrm{f}_E(e_{Ht+1}),\end{aligned} \tag{3.4}$$

$$\begin{aligned}\mathrm{f}(s_{t+1}^{SP}|s_t^{SP}, a_{SP}=11) &= \mathrm{p}_\Theta(p_t^{SP})(1-\mathrm{p}_P(p_t^{SP}))\delta(p_{t+1}^{SP} - p_{01})\delta(b_{t+1}^{SP} - \phi(\phi(b_t^{SP} + e_{Ht}) - e_S - e_P)) \\ &\quad \times \mathrm{f}_E(e_{Ht+1}) + (1-\mathrm{p}_\Theta(p_t^{SP}))\delta(p_{t+1}^{SP} - \psi(\mathrm{p}_N(p_t^{SP})))\delta(b_{t+1}^{SP} - \phi(\phi(b_t^{SP} + e_{Ht}) - e_S))\,\mathrm{f}_E(e_{Ht+1}),\end{aligned} \tag{3.5}$$

$$\mathrm{f}(s_t^T|s_t^{SP}, a_{SP}=11) = \mathrm{p}_\Theta(p_t^{SP})\,\mathrm{p}_P(p_t^{SP})\delta(b_t^T - \phi(\phi(b_t^{SP} + e_{Ht}) - e_S - e_P))\,\mathrm{f}_H(h_t). \tag{3.6}$$

$$\mathrm{f}(s_{t+1}^{SP}|s_t^T, a_T = e_T) = \delta(p_{t+1}^{SP} - p_{11})\delta(b_{t+1}^{SP} - \phi(b_t^T - e_T))\,\mathrm{f}_E(e_{Ht+1}). \tag{3.7}$$

Assumption 3.1. *For any $b^T \in [0, B_{\max}]$ and any $e_T \in \mathbf{E_T}$, $\mathbb{E}[r(s^T, e_T)]$ and $\mathbb{E}[r^2(s^T, e_T)]$ exist and are bounded by some constants L_1 and L_2, respectively, with $\mathbb{E}[\cdot]$ being the expectation operation over random variable H.*

3.3.2.1 Possible generalizations

We also discuss possible generalization of the formulated two-stage MDP model to multi-channel and bursty traffic cases. Subsequent developed after-state based control and learning algorithms apply similarly with generalized MDPs.

- Multi-channel cases: Assume that the SU is able to sense and transmit over one of multiple channels. In this case, at a sensing-probing state, the SU has to decide whether or not to sense; if yes, to decide which channel to sense; if sense a free channel, to decide whether or not to probe. In addition, instead of maintaining a scale channel belief variable, a state (both s^{SP} and s^T) should include a belief vector respectively representing the SU's belief on these channels, which can be updated based on corresponding channel's occupancy model and sensing-probing observations (see [22, 72]).

- Bursty traffic cases: With bursty traffic, data traffic arrives randomly, and the data buffer fluctuates randomly. In this case, besides the amount of transmitted data, reducing packet loss due to data packet's overflow is also of interest. Therefore, we can include current length of data buffer into states, and redefine the reward function (3.2) as a weighted combination of sent data and (negative) buffer length (see [75]).

3.3.3 Optimal control via state value function V^*

Let Π denote all stationary deterministic policies, which are mappings from $s \in \mathbb{S}$ to $\mathbb{A}(s)$. We limit the control within Π. For any $\pi \in \Pi$, we define a function $V^\pi : \mathbb{S} \to \mathbb{R}$ for π as follows,

$$V^\pi(s) \triangleq \mathbb{E}[\sum_{\tau=0}^{\infty} \gamma^\tau r(s_\tau, \pi(s_\tau))|s_0 = s], \tag{3.8}$$

where the expectation is defined by the state transition kernel (3.3)-(3.7). Therefore, by setting γ to a value that is close to 1, $V^\pi(s)$ can be (approximately) interpreted as the expected data throughput achieved by policy π over infinite time horizon with initial state s.

From the discussions of Chapter 2.1.2, we know that, $V^*(s)$ is the solution to the following equation

$$V(s) = \max_{a \in \mathbb{A}(s)} \{r(s,a) + \gamma \mathbb{E}[V(s')|s,a]\}, \tag{3.9}$$

the optimal policy $\pi^*(s)$, which attaches the maximum value among all policies Π, can be defined as

$$\pi^*(s) = \arg\max_{a \in \mathbb{A}(s)} \{r(s,a) + \gamma \mathbb{E}[V^*(s')|s,a]\}. \tag{3.10}$$

In other words, the task is to find a policy $\pi^* \in \Pi$ such that its expected (discounted) throughput is maximized.

Remark: Although the optimal policy $\pi^*(s)$ can be obtained from the state value function $V^*(s)$, there are two practical difficulties for using (3.9) and (3.10) to solve our problem. First, the SU does not know the PDF's $\mathrm{f}_E(x)$ and $\mathrm{f}_H(x)$. The $\max\{\cdot\}$ operation outside of $\mathbb{E}[\cdot]$ operation in (3.9) will impede us in using samples to estimate[6] V^*. Second, $\mathbb{E}[\cdot]$ operation for the action selection in (3.10) will impede us in achieving the optimal action, even if V^* is known.

Remark: In addition, there is another theoretical difficulty. In discounting MDP theory, the existence of V^* is usually established from the contraction theory, which

[6] This difficulty can be illustrated with a simpler task. Given V^1 and V^2 are two random variables, suppose that we wish to estimate $\max\{\mathbb{E}[V^1], \mathbb{E}[V^2]\}$. And we can only observe a batch of samples $\{\max\{v_i^1, v_i^2\}\}_{i=1}^L$, where v_i^1 and v_i^2 are realizations of V^1 and V^2, respectively. However, the simple sample average of the observed information is not able to provide an unbiased estimation of $\max\{\mathbb{E}[V^1], \mathbb{E}[V^2]\}$, since $\lim_{L\to\infty} \frac{1}{L}\sum_{i=1}^L \max\{v_i^1, v_i^2\} \geq \max\{\mathbb{E}[V^1], \mathbb{E}[V^2]\}$.

requires the reward function $r(s,a)$ to be bounded for all s and all a [49, p. 143]. However, this is not satisfied in our approach, since we allow the channel gain h to take all positive values, and hence, r is unbounded over the state space. Therefore, in this case, the existence of V^* is not easy to establish.

As we will show in Section 3.4, both the practical and theoretical difficulties can be solved by transforming the value function into an after-state setting. Moreover, this transformation reduces space complexity via eliminating the explicit need for representing E_H and H processes.

3.4 After-state Reformulation

Here, we first analyze the structure of the two-stage MDP (Section 3.4.1). Second, Section 3.4.2 reformulates the optimal control in terms of after-state value function J^*. Finally, the solution of J^*, and its relationships with the state value function V^* are given in Section 3.4.3.

3.4.1 Structure of the MDP

The structural properties of the MDP given in the 4-tuple $(\mathbb{S}, (\mathbb{A}(s))_s, \mathrm{f}(\cdot|s,a), r(s,a))$ are as follows.

1) We divide each state into endogenous and exogenous components. Specifically, for a sensing-probing state s^{SP}, the endogenous and exogenous components are $d^{SP} = [p^{SP}, b^{SP}]$ and $x^{SP} = \{e_H\}$, respectively. All possible d^{SP} and x^{SP} are defined as $\mathbb{D}^{SP}$ and $\mathbb{X}^{SP}$, respectively.

Similarly, for a transmitting state s^T, the endogenous and exogenous components are $d^T = \{b^T\}$ and $x^T = \{h\}$, respectively. All possible d^T and x^T are $\mathbb{D}^T$ and $\mathbb{X}^T$, respectively.

Finally, let $d \in \mathbb{D} = \mathbb{D}^{SP} \cup \mathbb{D}^T$ and $x \in \mathbb{X} = \mathbb{X}^{SP} \cup \mathbb{X}^T$.

2) The number of available actions $\mathbb{A}(s)$ at each state s is finite.

3) Checking the state transition kernel (3.3), (3.4), (3.5), (3.6) and (3.7), we can see that, given state $s = [d, x]$, and action $a \in \mathbb{A}(s)$, the transition to next state $s' = [d', x']$ has following properties.

- The stochastic model of d' is fully known. Specifically, for a given action a

taken at state $s = [d, x]$, we have $\mathcal{N}(a)$ possible cases depending on sensing observations after the action, which leads to $\mathcal{N}(a)$ possible values of d'. And at the ith case, which happens with probability $p_i(d, a)$, the value of d' takes the value $\varrho_i(s, a)$. Functions $\mathcal{N}$, ϱ_i and p_i are known, and listed in Table 3.1 for different d, x, a and observations.

- The x' is a random variable whose distribution depends on $\varrho_i(s, a)$, i.e., if $\varrho_i(s, a) \in \mathbb{D}^{SP}$, x' has PDF $\mathrm{f}_E(x)$; and if $\varrho_i(s, a) \in \mathbb{D}^T$, x' has PDF $\mathrm{f}_H(x)$ (see Table 3.1). This relationship is described by conditional PDF $\mathrm{f}_X(x'|\varrho_i(s, a))$.

With these notations, the state transition kernel $\mathrm{f}(s'|s, a)$ can be rewritten as:

$$\begin{aligned}\mathrm{f}(s'|s, a) &= \mathrm{f}((d', x')|(d, x), a) \\ &= \sum_{i=1}^{\mathcal{N}(a)} p_i(d, a)\delta(d' - \varrho_i(s, a))\, \mathrm{f}_X(x'|\varrho_i(s, a)).\end{aligned} \tag{3.11}$$

4) The reward $r([d, x], a)$ is deterministic, defined via (3.1) and (3.2).

Table 3.1: Structured state transition model

	d	x	$a \in \mathbb{A}(d, x)$	$\mathcal{N}(a)$	Observation	$p_i(d, a)$	$d' = \varrho_i([d, x], a)$	$\mathrm{f}_X(x'\|\varrho_i)$
s^{SP}	$[b, p]$	e_H	00	1	none	1	$[\psi(p), \phi(b + e_H)]$	f_E
			10	2	$\Theta = 1$	$\mathrm{p}_\Theta(p)$	$[\psi(\mathrm{p}_P(p)), \phi(\phi(b + e_H) - e_S)]$	f_E
					$\Theta = 0$	$1 - \mathrm{p}_\Theta(p)$	$[\psi(\mathrm{p}_N(p)), \phi(\phi(b + e_H) - e_S)]$	f_E
			11	3	$\Theta = 1$, FB =1	$\mathrm{p}_\Theta(p)\,\mathrm{p}_P(p)$	$\phi(\phi(b + e_H) - e_S - e_P)$	$\mathrm{f}_H()$
					$\Theta = 1$, FB =0	$\mathrm{p}_\Theta(p)(1 - \mathrm{p}_P(p))$	$[p_{01}, \phi(\phi(b + e_H) - e_S - e_P)]$	f_E
					$\Theta = 0$	$1 - \mathrm{p}_\Theta(p)$	$[\psi(\mathrm{p}_N(p)), \phi(\phi(b + e_H) - e_S)]$	f_E
s^T	b	h	e_T	1	none	1	$[p_{11}, \phi(b - e_T)]$	f_E

3.4.2 Introducing after-state based control

Based on the above structural properties, we now show that optimal control can be developed based on "after-states" (see Chapter 2.1). Physically, an after-state is the endogenous component of a state. However, for ease of presentation, we consider it as a "virtual state" appended to the original MDP (Fig. 3.5).

Specifically, after an action a applied on a state $s = [d, x]$, it randomly transits to an after-state β. The number of such transitions is $\mathcal{N}(a)$. At the ith transition, the after-state is $\beta = \varrho_i([d, x], a)$ with probability $p_i(d, a)$. From β, the next state is $s' = [d', x']$ with $d' = \beta$ and x' has PDF $\mathrm{f}_X(\cdot|\beta)$.

We next introduce after-state based control. The main ideas are as follows. From β, the next state $s' = [d', x']$ only depends on β. Therefore, ***starting from an***

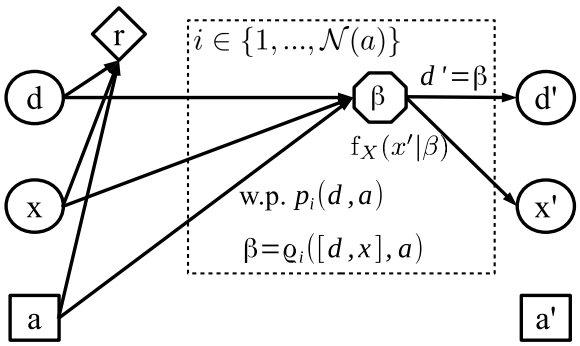


Figure 3.5: Augmented MDP model with after-state

after-state β, the maximum expected discounted reward only depends on β . We denote it by an *after-state value function* $J^*(\beta)$. The key is that if $J^*(\beta)$ is known for all β, the optimal action at a state $s = [d, x]$ can be determined as

$$\pi^*([d,x]) = \arg\max_{a \in \mathbb{A}([d,x])} \{r([d,x],a) + \sum_{i=1}^{\mathcal{N}(a)} p_i(d,a) J^*(\varrho_i([d,x],a))\}. \tag{3.12}$$

The equation (3.12) is intuitive: the optimal action at a state $s = [d, x]$ is the one that maximizes the sum of the immediate reward $r([d, x], a)$ and the expected maximum future value $\sum_{i=1}^{\mathcal{N}(a)} p_i(d, a) J^*(\varrho_i([d, x], a))$. The solving of J^* and the formal proof of (3.12) are provided in Section 3.4.3.

Remark: Unlike (3.10), if J^* is known, generating actions with (3.12) is easy, since $\mathcal{N}(a)$ and $|\mathbb{A}(s)|$ are finite, and $p_i(d, a)$ and $\varrho_i([d, x], a)$ are known. Furthermore, the space complexity of J^* is lower than that of V^*, since $\mathbb{X}$ does not need to be represented in J^*.

3.4.3 Establishing after-state based control

The development of this subsection is as follows. First, we define a so-called after-state Bellman equation as

$$J(\beta) = \gamma \mathop{\mathbb{E}}_{X'|\beta} \left[\max_{a' \in \mathbb{A}([\beta, X'])} \{r(\beta, X', a') + \sum_{i=1}^{\mathcal{N}(a')} p_i(\beta, a') J(\varrho_i([\beta, X'], a'))\} \right], \tag{3.13}$$

where $\mathop{\mathbb{E}}_{X'|\beta}[\cdot]$ means taking expectation over random variable[7] X', which has PDF $f_X(\cdot|\beta)$. Then, Theorem 3.1 shows that (3.13) has a unique solution J^*, and also provides a value iteration algorithm for solving it. Note that, at this moment, the

[7] Given that the after-state of current slot is β, X' denotes the random exogenous variable of the next slot (see Fig. 3.5).

meaning of J^* is unclear. Finally, Theorem 3.2 and Corollary 3.1 show that J^* is exactly the after-state value function defined in Section 3.4.2, and the policy defined with (3.12) is equivalent with (3.10), and therefore, is the optimal policy.

Theorem 3.1. *Given Assumption 3.1, there is a unique J^* that satisfies (3.13). And J^* can be calculated via a value iteration algorithm: with J_0 being an arbitrary bounded function, the sequence of functions $\{J_l\}_{l=0}^{L}$ defined by the following iteration equation: for all $\beta \in \mathbb{D}$,*

$$J_{l+1}(\beta) \leftarrow \gamma \mathop{\mathbb{E}}_{X'|\beta}[\max_{a' \in \mathbb{A}([\beta, X'])} \{r([\beta, X'], a') + \sum_{i=1}^{\mathcal{N}(a')} p_i(\beta, a') J_l(\varrho_i([\beta, X'], a'))\}], \quad (3.14)$$

converges to J^* when $L \to \infty$.

Proof. See Section 3.8.1. □

Remark: Unlike the classical Bellman equation (3.9), in the after-state Bellman equation (3.13), the expectation is outside of the reward function. While this is unbounded, its expectation is bounded due to Assumption 3.1. Therefore, the solution to (3.13) can be established by contraction theory.

Remark: Comparing with (3.9), equation (3.13) exchanges the order of (conditional) expectation and maximization operators. And inside the maximization operator, functions r, $\mathcal{N}$, p_i, and ϱ_i are known. These are crucial in developing a learning algorithm that uses samples to estimate the after-state value function J^*.

Theorem 3.2. *The existence of a solution V^* to* (3.9) *can be established from J^*. In addition, their relationships are*

$$V^*([d, x]) = \max_{a \in \mathbb{A}([d,x])} \left\{ r([d, x], a) + \sum_{i=1}^{\mathcal{N}(a)} p_i(d, a) J^*(\varrho_i([d, x], a)) \right\} \quad (3.15)$$

and

$$J^*(\beta) = \gamma \mathop{\mathbb{E}}_{X'|\beta} \left[V^*([\beta, X']) \right]. \quad (3.16)$$

Proof. See Section 3.8.2. □

Corollary 3.1. *J^* is the after-state value function, and the policy defined with* (3.12) *is optimal.*

Proof. From (3.16) and the physical meaning of V^* (see (2.4) in Chapter 2.1), $J^*(\beta)$ represents the maximum expected discounted sum of reward, starting from after-state β. Therefore, J^* is the after-state value function.

The equation (3.12) can be derived from the optimal policy (3.10) as follows: first decompose the expectation with (3.11), and then plug in (3.16). Therefore, (3.12) is the optimal policy. □

Corollary 3.1 shows that optimal control can be achieved equivalently through value function J^*. And Theorem 3.1 establishes the existence of J^* and also provides a value iteration algorithm for solving J^*. However, there are two difficulties in obtaining J^* using the value iteration algorithm. **Difficulty 1**: the computation of $\mathop{\mathbb{E}}_{X'|\beta}[\cdot]$ requires the knowledge of f_E and f_H, which is unknown in our setting. **Difficulty 2**: the after-state space $\mathbb{D}$ is continuous, which requires computation of expectation at infinitely many β. Through reinforcement learning, these two difficulties will be addressed in Section 3.5.

3.5 Reinforcement Learning Algorithm

In this section, we first address **Difficulty 2** via discretizing the after-state space into finite clusters, which is discussed in Section 3.5.1. In addition, a learning algorithm is proposed in Section 3.5.2 to address **Difficulty 1**. Given data samples of wireless channel and energy-harvesting process, the algorithm learns a (near) optimal policy via sample averaging, instead of taking expectation. Furthermore, the algorithm's convergence guarantee and performance bounds are analyzed in Section 3.5.3. Finally, the algorithm is modified in Section 3.5.4, for achieving simultaneous data sampling, learning and control.

3.5.1 After-state space discretization

We will divide the continuous after-state space $\mathbb{D}$ into a finite number of portions or clusters $\mathbb{K}$, which defines a mapping $\omega : \mathbb{D} \to \mathbb{K}$. In addition, all after-states assigned into the same cluster are mapped into one representative after-state. Mathematically, let $\mathbb{D}(k) \triangleq \{\beta \in \mathbb{D} | \omega(\beta) = k\}$ denote the set of after-state assigned to cluster $k \in \mathbb{K}$. Thus, $q(k) \in \mathbb{D}(k)$ represents all after-states of $\mathbb{D}(k)$. Finally, we denote $\mathbb{K}^{SP}$ as the

image of $\mathbb{D}^{SP}$ under ω with its elements denoted as k^{SP}. And we denote $\mathbb{K}^T$ as the image of $\mathbb{D}^T$ under ω with its elements denoted as k^T.

As an example, in Fig. 3.6, two-dimensional $\mathbb{D}^{SP}$ is uniformly discretized into 9 clusters $\mathbb{K}^{SP} = \{1, ..., 9\}$. The one-dimensional subset of after-state space $\mathbb{D}^T$ is uniformly discretized into 3 clusters $\mathbb{K}^T = \{10, 11, 12\}$. The association from an after-state β to the cluster k is denoted by $k = \omega(\beta)$. And the after-states assigned to the same cluster are represented by its central point, $q(k)$.

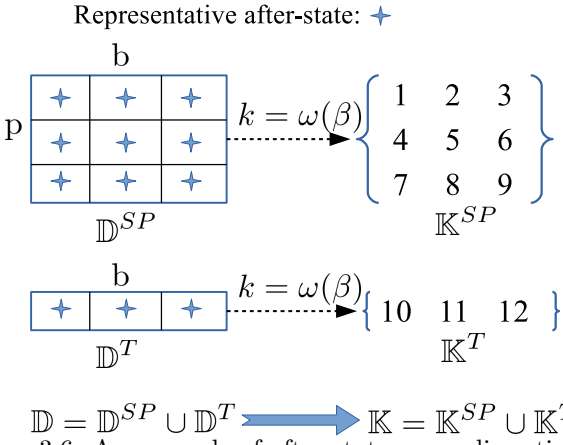


Figure 3.6: An example of after-state space discretization

3.5.2 Learn optimal policy with data samples

With this discretization, we design a reinforcement learning algorithm that learns near optimal policy from the samples of E_H and H.

The idea is to learn a function $g(x)$ over $\mathbb{K}$ to approximate $J^*(x)$ such that $g(\omega(\beta))$ is close to $J^*(\beta)$ for all $\beta \in \mathbb{D}$. Then a near-optimal policy can be constructed as

$$\hat{\pi}([d,x]|g) = \underset{a \in \mathbb{A}([d,x])}{\arg\max} \{r([d,x],a) + \sum_{i=1}^{\mathcal{N}(a)} p_i(d,a) g(\omega(\varrho_i([d,x],a))\}. \tag{3.17}$$

Comparing (3.17) with (3.12), we observe that if $g(x)$ approximates $J^*(x)$ accurately, $\hat{\pi}(\cdot|g)$ is close to π^*.

The function $g(x)$ is learned by iteratively updating with data samples. Each update uses only one data sample. This facilitates the tailoring of the algorithm for online applications (Section 3.5.4). Next, we present the algorithm and some intuitive reasons.

3.5.2.1 Algorithm

Initial with arbitrary bounded function $g_0(x)$. Calculate $g_{l+1}(x)$ from $g_l(x)$ and x_l, the lth data sample. Since x_l can be either an energy or fading sample, there are two cases:

- if x_l is a sample of E_H, randomly choose N non-repeated clusters from $\mathbb{K}^{SP}$;
- if x_l is a sample of H, randomly choose N non-repeated clusters from $\mathbb{K}^{T}$.

For either case, we denote the set of chosen clusters as $\bar{K}_l$. Given x_l and $\bar{K}_l$, we have the updating rule as

$$g_{l+1}(k) = \begin{cases} (1-\alpha_l(k)) \cdot g_l(k) + \alpha_l(k) \cdot \delta_l(k) & k \in \bar{K}_l \\ g_l(k) & \text{otherwise,} \end{cases} \tag{3.18}$$

where $\alpha_l(k) \in (0,1)$ is the step size of cluster k for the lth iteration, and $\delta_l(k)$ is constructed with x_l as

$$\delta_l(k) \triangleq \gamma \max_{a \in \mathbb{A}([q(k),x_l])} \{r([q(k),x_l],a) + \sum_{i=1}^{\mathcal{N}(a)} p_i(q(k),a) g_l(\omega(\varrho_i([q(k),x_l],a)))\} \tag{3.19}$$

(see Section 3.5.2.2 for the interpretation of $\delta_l(k)$).

Section 3.5.3 will show that if energy and fading can be sampled infinitely often, the step size $\alpha_l(k)$ decays and the sequence of functions $\{g_l(x)\}_{l=1}^{\infty}$ converges such that $g_\infty(\omega(\beta))$ is close to $J^*(\beta)$, and the policy $\hat{\pi}(\cdot|g_\infty)$ defined (3.17) is close to π^*.

Algorithm 3.1 Learning of control policy

Require: Data samples $\{x_l\}_l$
Ensure: Learned control policy $\hat{\pi}(\cdot|g_L)$
  Initialize $g_0(k) = 0,\ \forall k$
  for l from 0 to $L-1$ **do**
    if x_l is a data sample of E_H **then**
      Choose N clusters from $\mathbb{K}^{SP}$ and get $\overline{K}_l$
    else if x_l is a data sample of H **then**
      Choose N clusters from $\mathbb{K}^{T}$ and get $\overline{K}_l$
    end if
    Generate g_{l+1} by executing (3.18) with $(x_l, \overline{K}_l)$
  end for
  With g_L, construct control policy $\hat{\pi}(\cdot|g_L)$ through (3.17)

The above algorithmic pieces are summarized in Algorithm 3.1. For a sufficiently large L number of iterations, the learning process can be considered complete. The

learned policy $\hat{\pi}(x|g_L)$ can then be used for sensing, probing and transmission control, just as in Algorithm 3.2 in Section 3.5.4[8].

3.5.2.2 Intuitions

Algorithm 1 is a stochastic approximation algorithm [76], which is intuitively generalization of the value iteration algorithm (3.14). Specifically, it is known from (3.14) that, given the value function $J_l(\beta)$ of the l-th iteration, a noisy estimation of $J_{l+1}(\beta)$ can be constructed as

$$\max_{a' \in \mathbb{A}([\beta, x'])} \{r([\beta, x'], a') + \sum_{i=1}^{\mathcal{N}(a')} p_i(\beta, a') J_l(\varrho_i([\beta, x'], a'))\}, \tag{3.20}$$

with x' sampled from $\mathrm{f}_X(\cdot|\beta)$, i.e., x' is a realization of E_H, if $\beta \in \mathbb{D}^{SP}$; and x' is a realization of H, if $\beta \in \mathbb{D}^T$.

Therefore, by comparing (3.20) with (3.19), we see $\delta_l(k)$ as an estimate of $g_{l+1}(k)$ for $k \in \bar{K}_l$ (with ω introduced for discretization, β approximated with $q(k)$, and J_l replaced with g_l). Hence, with $\delta_l(k)$, equation (3.18) updates g_{l+1} for chosen clusters within $\bar{K}_l$ by sample averaging. Note that, theoretically, we can set $\bar{K}_l$ to $\mathbb{K}^{SP}$ or $\mathbb{K}^T$ (x_l is energy or fading sample), which could accelerate learning speed (Section 3.6.2.2 gives an example). However, large $|\mathbb{K}^{SP}|$ or $|\mathbb{K}^T|$ leads to increased computations. Hence, instead of updating all clusters of $\mathbb{K}^{SP}$ or $\mathbb{K}^T$, we randomly update N clusters within them at each iteration, which controls the computational burden.

3.5.3 Theoretical soundness and performance bounds

In this part, we formally state the convergence requirements and performance guarantees for Algorithm 3.1.

First, for $\forall\, k \in \mathbb{K}$, we define $M(k) = \{l \in \{0, 1, ..., L-1\} | k \in \overline{K}_l\}$, which presents the set of iteration indices where k is chosen during learning. In addition, we define

$$\xi \triangleq \max_k \{ \sup_{\beta \in \mathbb{D}(k)} |J^*(\beta) - J^*(q(k))|\}, \tag{3.21}$$

which describes the "error" introduced by the after-state space discretization. Finally, in order to evaluate the performance of a policy π from after-states' point of view,

[8] Specifically, we can get an adaptive MAC routine with $\hat{\pi}(\cdot|g_L)$, by removing lines (2), (5-7), (10-16), and (20-22) of the Algorithm 3.2, and replacing $\hat{\pi}(\cdot|g_l)$ in line (9) and line (25) with $\hat{\pi}(\cdot|g_L)$.

we define

$$J^{\pi}(\beta) = \gamma \mathop{\mathbb{E}}_{X'|\beta} \left[V^{\pi}([\beta, X']) \right], \tag{3.22}$$

where V^{π} is defined in (3.8).

Given the definitions of $M(k)$, ξ and $J^{\pi}(\beta)$, we have following theorem.

Theorem 3.3. *Given that Assumption 3.1 is true, and also assuming that, in Algorithm 3.1, as* $L \to \infty$,

$$\sum_{l \in M(k)} \alpha_l(k) = \infty, \, \forall k \tag{3.23}$$

$$\sum_{l \in M(k)} \alpha_l^2(k) < \infty, \, \forall k \tag{3.24}$$

then we have:

(i) the sequence of functions $\{g_l\}_{l=0}^{L}$ *generated in* (3.18) *converge to a function* g_∞ *with probability 1 as* $L \to \infty$*;*

(ii) $||J^* - J_\infty|| \leq \frac{\xi}{1-\gamma}$*, where function*

$$J_\infty(\beta) \triangleq g_\infty(\omega(\beta)), \tag{3.25}$$

and $||\cdot||$ *denotes the maximum norm;*

(iii) $||J^* - J^{\pi_\infty}|| \leq \frac{2\gamma\xi}{(1-\gamma)^2}$*, where*

$$\pi_\infty \triangleq \hat{\pi}(\cdot|g_\infty). \tag{3.26}$$

Proof. See Section 3.8.3. □

Remark: Assumptions (3.23) and (3.24) actually put constraints on both $\{x_l\}_l$ and $\alpha_l(k)$:

(a) Energy-harvesting and wireless fading processes need to be sampled infinitely often in $\{x_l\}_{l=0}^{L-1}$, as $L \to \infty$;

(b) for any k, the sequence of step sizes $\{\alpha_l(k)\}_{l \in M(k)}$ should delay at a proper rate (neither too fast nor too slow).

For Algorithm 3.1 to converge, the constraint (a) is needed to gain sufficient information on random processes; the constraint (b) is needed to properly average out randomness (small enough step sizes) and make sufficient changes to functions $\{g_l\}_l$ via updating (large enough step sizes). The reasoning from assumptions (3.23) and (3.24) applying to these two constraints is as follows.

First, $\sum_{l\in M(k)} \alpha_l(k) = \infty$ requires $|M(k)| = \infty$, where $|M(k)|$ denotes the size of $M(k)$. Because, otherwise, we have $\sum_{l\in M(k)} \alpha_l(k) \leq |M(k)|$ (as $\alpha_l(k)$ is upper bounded by 1). This further implies the constraint (a), due to the definition of $M(k)$ and the way that Algorithm 3.1 constructs $\bar{K}_l$.

Second, in order to satisfy $\sum_{l\in M(k)} \alpha_l^2(k) < \infty$, the sequence of step size $\{\alpha_l(k)\}_{l\in M(k)}$ should start to delay after certain l with sufficient delay rate. However, the delay rate should not too large, in order to satisfy $\sum_{l\in M(k)} \alpha_l(k) = \infty$. In summary, we have the constraint (b) for step size sequences. There are various step size rules that satisfy this constraint [77, Chapter 11]. For example, we can set $\alpha_l(k) = \frac{1}{|M(k,l)|}$, where $M(k,l)$ is the set of slots that cluster k is chosen before the lth iteration.

Remark: The statement (i) of Theorem 3.3 demonstrates the convergence guarantee of Algorithm 1. The statement (ii) shows that the learned function g_∞ is close to the J^*, and their difference is controlled by the error ξ caused by after-state space discretization. The statement (iii) claims that asymptomatically, the performance of policies $\{\hat{\pi}(\cdot|g_l)\}_l$ approaches that of the optimal policy π^*, and that the performance gap is proportional to the error ξ.

3.5.4 Simultaneous sampling, learning and control

Algorithm 3.1 operates offline — batch learning which generates the best predictor by learning on the entire training data set of energy-harvesting and channel fading data at once. Thus, the optimal policy cannot be used until learning is complete. However, for some applications, an online learning scheme may be more desirable. In online machine learning, sequential data is used to update the best predictor for future data at each step. It is also used in situations where it is necessary for the algorithm to dynamically adapt to new patterns in the data.

One intuitive idea to tailor Algorithm 3.1 for online learning is as follows. Supposing that current learned function is g_l, we can use $\hat{\pi}(\cdot|g_l)$ to generate actions and

interact with the environment in real-time. Thus, we can collect a data sample from energy-harvesting or channel fading process, which can be further used to generate g_{l+1}. As the loop continues, g_l approaches g_∞, and the policy $\hat{\pi}(\cdot|g_l)$ approaches π_∞, which implies that generated actions during the process will be more and more likely to be optimal. In this way, simultaneous sampling, learning and control can be achieved.

However, the problem is that the above method cannot guarantee to sample the wireless fading process infinitely-often (i.e., cannot satisfy assumptions (3.23) and (3.24) of Theorem 3.3). Note that the wireless fading process can be sampled only if $\hat{\pi}(\cdot|g_l)$ chooses $a^{SP} = 11$. But, the above method may enter a deadlock such that $a^{SP} = 11$ will never be chosen. The deadlock can be caused by: (1) insufficient battery energy that results from the learned policy's consistent aggressive use of energy; and/or (2) persistently locking in $a^{SP} = 00$ or $a^{SP} = 10$. In order to break this possible deadlock during the learning process, with some small probability ϵ (named as the exploration rate), we force the algorithm to deviate from $\hat{\pi}(\cdot|g_l)$ to either accumulate energy ($a^{SP} = 00$) or to probe channel gain information ($a^{SP} = 11$) (e.g., exploration).

Based on the above points, Algorithm 3.2 is provided for always-on sampling, learning and control. Here, we argue that g_l generated by Algorithm 3.2 converges to g_∞ when $t \to \infty$. First, at each time slot, there is probability $\epsilon/2$ that the algorithm will choose $a^{SP} = 00$ to accumulate energy. Therefore, given the battery level b_t^{SP} of slot t, we can[9] find a finite T such that $\text{prob}\{b_{t+T}^{SP} \geq e_S + e_P\} > 0$. In other words, at any slot $t \geq T$, we have $\text{prob}\{b_t^{SP} \geq e_S + e_P\} > 0$. Thus, having sufficient energy for sensing and probing, the algorithm will choose $a^{SP} = 11$ with probability $\epsilon/2$. In addition, at any time slot, the channel will be free with a non-zero probability. Therefore, there is a non-zero probability that the algorithm can reach the transmitting stage. Thus, the wireless fading process can be sampled infinitely often for $t \to \infty$. In summary, the assumptions (3.23) and (3.24) of Theorem 3.3 are satisfied (under properly delayed step size), and $\{g_l\}_l$ converges to g_∞ asymptotically.

[9] If this condition cannot be satisfied, the underlying energy harvest process is not sufficient to power secondary nodes.

Algorithm 3.2 Simultaneous sampling, learning and control

Note: $\beta_t^{SP} \in \mathbb{D}^{SP}$ presents an after-state in slot t. β_t^T is defined similarly.
Initialize: battery b_0, channel belief p_0, and after-state $\beta_0^{SP} = [b_0, p_0]$
Initialize: $g_0(k) = 0,\ \forall k$, and set $l = 0$
for t from 1 to ∞ **do**
    Observe arriving harvested energy amount e_{Ht}
    Set $x_l = e_{Ht}$ and choose $\overline{K}_l$ with N clusters from $\mathbb{K}^{SP}$
    Generate g_{l+1} by executing (3.18) with $(x_l, \overline{K}_l)$
    $l \leftarrow l + 1$
    Construct state $s_t^{SP} = [\beta_{t-1}^{SP}, e_{Ht}]$
    Generate sensing-probing decision $a_t^{SP} = \hat{\pi}(s_t^{SP}|g_l)$ via (3.17)
    if random() $\leq \epsilon$ **then** ▷ Exploration
        if random() $\leq 1/2$ **then**
            $a_t^{SP} = 00$
        else if $11 \in \mathbb{A}(s_t^{SP})$ **then** ▷ Energy sufficiency
            $a_t^{SP} = 11$
        end if
    end if
    Apply sensing and probing actions based on a_t^{SP}
    if $a_t^{SP} = 11$ & $\Theta = 1$ & $FB = 1$ **then**
        Observe the channel gain h_t from FB
        Set $x_l = h_t$, and construct $\overline{K}_l$ by choosing N clusters from $\mathbb{K}^T$
        Generate g_{l+1} by executing (3.18) with $(x_l, \overline{K}_l)$
        $l \leftarrow l + 1$
        Derive after-state β_t^T with s_t^{SP} via Table 3.1
        Construct state $s_t^T = [\beta_t^T, h_t]$
        Generate transmit decision $a_t^T = \hat{\pi}(s_t^T|g_l)$ via (3.17)
        Set transmission power based on a_t^T, and transmit data
        Derive after-state β_t^{SP} from (s_t^T, a_t^T) via Table 3.1
    else
        Derive after-state β_t^{SP} with s_t^{SP} and a_t^{SP}, Θ and FB via Table 3.1
    end if
end for

3.5.4.1 Choices of exploration rate

Although the convergence is guaranteed for any $\epsilon \in (0, 1)$, the choice of ϵ affects the performance of the algorithm. Large ϵ helps channel acquisition, which may in turn accelerate the learning process. But too large ϵ will make the algorithm act too randomly, and cause significant loss to the achievable performance. Section 3.6.2.1 discusses the choice of ϵ in detail.

3.5.4.2 Complexity analysis of Algorithm 3.2

For each t, major computations are the two embedded function updates for g_l (line 6 and line 21). Each update needs to compute (3.19) N times. And each computation requires $|\mathcal{N}(a)|$ multiplications, $|\mathcal{N}(a)|$ summations and one maximization over a set with size $|\mathbb{A}(a)|$.

3.5.4.3 Energy burden for running Algorithm 3.2

In this part, we consider the energy burden for executing Algorithm 3.2 within each time slot, since it is running on an energy limited node. The exact amount of energy consumption is difficult to compute, since it depends on hardware platforms and algorithm implementation details. Hence, we roughly estimate the order of energy consumption (rather than its exact value).

Reference [78] shows that the energy consumption for executing of an algorithm is determined as

$$\text{Engy}_{\text{alg}} = P_{\text{pro}} \cdot T_{\text{alg}},$$

where P_{pro} is the operating power of the processor (where the algorithm is executed) and T_{alg} is the time needed for executing the algorithm. In addition, T_{alg} can be modeled as

$$T_{\text{alg}} = C_{\text{alg}} \cdot \frac{1}{f_{\text{pro}}},$$

where f_{pro} is the clock frequency of the processor, and C_{alg} denotes the number of clocks that the processor needs to execute to the algorithm within each time slot.

Assuming that $|\mathbf{E_T}| = 6$ and $N = 1$, from Section 3.5.4.2 and Table 3.1, we know that, in worst case, executing Algorithm 3.2 requires 4 multiplications, 4 summations,

one maximization over a set with size 3 and one maximization over a set with size 6. It is sensible to assume that C_{alg} is with order 10^2. Furthermore, reference [79] shows that, for modern processors, $\frac{P_{\text{pro}}}{f_{\text{pro}}}$ is with order 10^{-9} Joule. In summary, the order of Engy_{alg} should be around 10^{-7} Joule.

Therefore, within each time slot, energy consumption for executing learning algorithm is negligible compared with that for sensing, probing and transmitting. For example, a typical transmission power is around 10 mW, and packet duration is around 10 ms, which means that the typical transmission energy is around 10^{-4} Joule.

3.6 Simulation Results

3.6.1 Simulation setup

The secondary signal power attenuation h consists of path loss h_s and channel fading h_f. We assume a Rayleigh fading model. Thus, h_f has PDF $f(x) = e^{-x}, \quad x \geq 0$. The path loss h_s is distance-dependent and is assumed to be fixed.

Then, with above channel model, the amount of transmitted data can be rewritten as

$$\tau_T W \log_2(1 + \frac{e_T h_s h_f}{\tau_T N_0 W}) = \tau_T W \log_2(1 + \frac{e_T h_f}{\eta})$$

where $\eta \triangleq \tau_T N_0 W / h_s$. We assume that $W = 1$ MHz, $\tau_T = 10$ ms, and $\eta = 1$ (for energy normalization). Normalizing with respect to η, we set battery capacity $B_{\max} = 10$, sensing energy $e_S = 1$, probing energy $e_P = 1$, and the set of transiting energy levels $\mathbf{E_T} = \{0, 1, 2, 3, 4, 5\}$.

In addition, we assume that energy is harvested from wind power. Thus, E_H is well characterized by the Weibull distribution [80], with shape and mean parameters k_E and μ_E. Throughout the simulation, we set $k_E = 1.2$, and change μ_E to model different harvested energy supply rates.

Furthermore, the channel occupancy Markov model is described by $p_{00} = 0.8$ and $p_{11} = 0.9$ and spectrum sensing are set as: $p_{FA} = 0.1$ and $p_M = 0.01$.

Finally, simple uniform grid is used for discretization with 10 levels each for both belief and battery dimensions. Thus, $|\mathbb{K}^{SP}| = 100$ and $|\mathbb{K}^{T}| = 10$.

3.6.2 Characteristics of online learning algorithm

3.6.2.1 Learning under various exploration rate ϵ

With $\mu_E = 1$ and the update size $N = 1$, we investigate Algorithm 3.2 for $\epsilon \in \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ and for ϵ adapting with $\sqrt{1/t}$.

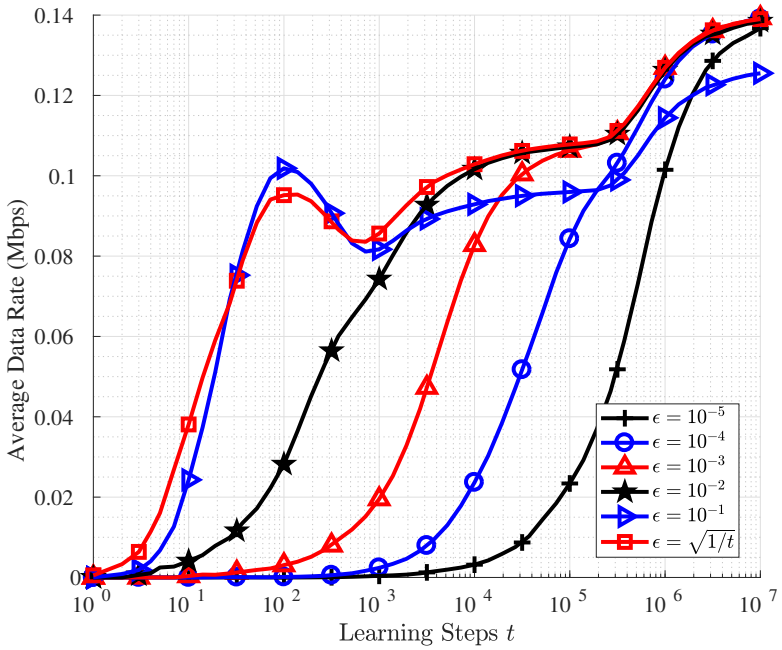


Figure 3.7: Learning curves under various exploration rates

Fig. 3.7 shows the learning curves with logarithmic time index. Note that smaller ϵ takes longer time to initiate the learning process. In addition, some learning curves (with ϵ equal to 10^{-1}, 10^{-2}, 10^{-3} and $\sqrt{1/t}$) consist of an initial phase (from 0 to around 10^5 steps) and a final phase, which begins (at around the 10^5-th step) with a steep performance increase. These phenomena are explained as follows.

First, note that ϵ determines the frequency of sampling of the wireless fading process, which further determines the updating rate of $g_l(x)$ over $\mathbb{K}^T$. In addition, the learning complexity for transmitting policies is much lower than that for sensing-probing policies, since the size of $\mathbb{K}^T$ (e.g., 10) is much smaller than that of $\mathbb{K}^{SP}$ (e.g., 100). Therefore, with large enough ϵ (equal to 10^{-1}, 10^{-2}, 10^{-3} and $\sqrt{1/t}$), the transmitting policy learns significantly faster than the sensing-probing policy. Hence, the learning curve manifests a two-phase process. And since larger ϵ implies obtaining CSI more often, an algorithm with larger ϵ initiates faster.

However, having too large a value for ϵ can cause performance loss due to too aggressive exploration. For example, for $\epsilon = 10^{-1}$, the system achieves 10% less throughput than in other cases. As well, $\epsilon = \sqrt{1/t}$, which starts with large value and decreases over time, provides fast start-up and also almost-lossless asymptotic performance.

3.6.2.2 Learning with update size N

With $\mu_E = 1$ and $\epsilon = \sqrt{1/t}$, we investigate Algorithm 3.2 for $N \in \{1, 2, 5, 10\}$. We observe that all algorithms converge to the same limit (Fig. 3.8), but larger N requires fewer learning steps. This suggests a trade-off between computational load and learning speed.

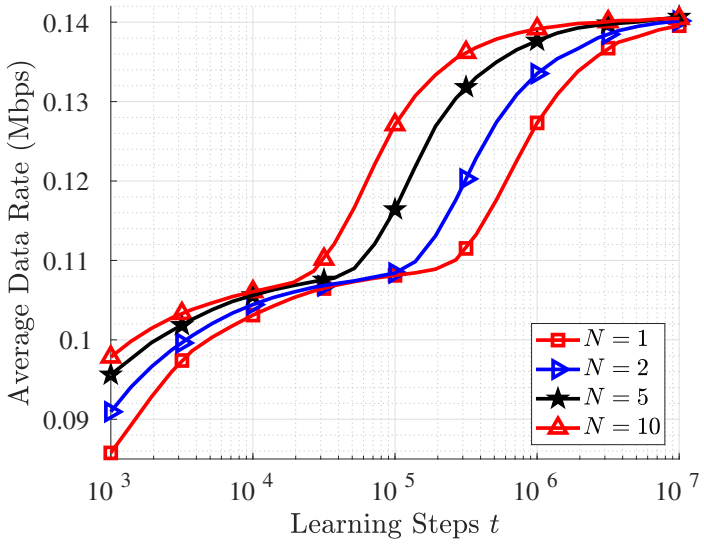


Figure 3.8: Learning curves under various update sizes

3.6.3 Myopic versus holistic

We next compare Near Opt (the policy learned by Algorithm 3.2), Greedy-Sensing-Probing (GSP), Greedy-Transmitting (GT), and Greedy-Sensing-Probing-Transmitting (GSPT). GSP always senses and probes the channel, but adapts the transmit power, which is learned by constraining the action space $\mathbb{A}(s)$ in (3.19) and (3.17) to have only the greedy action at sensing-probing states. In GT, which is learned similarly, the node transmits at maximum power level whenever the energy is sufficient, but carefully chooses the sensing and probing action. GSPT is a pure greedy policy with

myopic actions at both stages. We compare how these policies perform with different rates of energy harvests (vs. mean harvested energy μ_E).

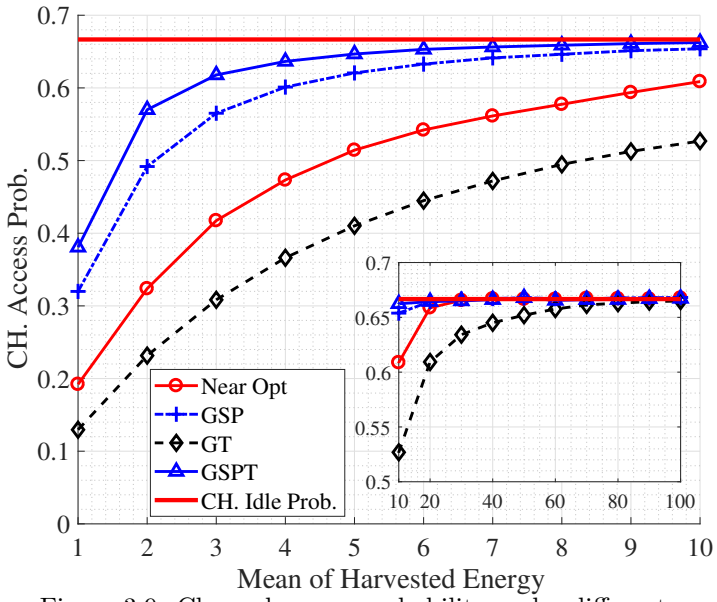


Figure 3.9: Channel access probability under different μ_E

First, we consider the ability of a policy to exploit channel access opportunities. This is measured by channel access probability, which is the probability that the channel is free while a sensing action is chosen. As a benchmark, this probability is upper bounded by the channel's idle probability $p_{01}/(p_{01}+p_{10}) = 0.2/0.3 \approx 0.67$. Fig. 3.9 shows the measured channel access probabilities of different policies. In addition, the data rate is measured and presented in Fig. 3.10, which is upper-bounded by $p_{01}/(p_{01}+p_{10}) \cdot p_O \cdot \mathbb{E}[W \log_2(1+e_T^M h_f)] \approx 1.285$ Mbps, where $e_T^M = \max\{\mathbf{E_T}\} = 5$.

Overall, all the policies exploit the increasing harvested energy for more channel access opportunities and higher data rates. And with a high enough energy supply, all the policies achieve the upper bounds.

Note that GSPT is most efficient at exploiting channel access opportunities because it is aggressive in sensing and probing. However, the drawback is that these actions are energy intensive, resulting in a lack of energy for data transmission. Thus, GSPT has the worst performance in terms of data rate.

Compared with GSPT, GSP has lower channel access probability but achieves a higher data rate. The main reason is that GSP adapts transmit power based on channel fading status and the greedy sensing-probing strategy. Therefore, GSP uses

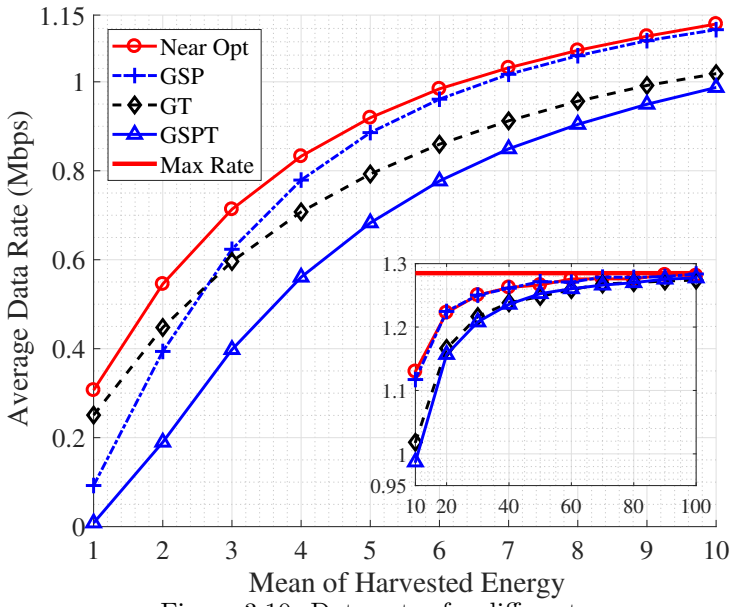


Figure 3.10: Data rates for different μ_E

energy more efficiently for data transmitting rather than simply sensing and probing.

In contrast, GT always uses the highest transmit power without considering the channel status. And this policy strives to make intelligent sensing and probing decisions based on battery energy level, channel belief and the aggressive transmission strategy. When the harvested energy is low, GT outperforms GSP in terms of data rate. That is because making proper sensing and probing decision to save energy for transmission is far more important. However, when the harvested energy supply increases, GSP outperforms GT. The reason is that energy expenses due to greedy sensing and probing action are marginal relative to the available energy.

With full adaptation, Near Opt achieves favorable energy trade-offs between sensing-probing and transmission stages. Therefore, Near Opt achieves the best data rate.

3.7 Summary

In this chapter, with goal of maximizing data throughput, we studied the optimal design of spectrum sensing, channel probing and transmitting for energy-harvesting cognitive nodes. The optimal control problem was formulated with a two-stage MDP model, and further simplified via after-state technique. For addressing the difficulty of lacking distribution information, we proposed to use reinforcement learning to solve

the optimal policy, whose theoretical basis and performance bounds were analyzed.

3.8 Appendix

3.8.1 Proof of Theorem 3.1

Theorem 3.1 is proved with the use of contraction theory. Specifically, we show the solution to (3.13) uniquely exists, and it is the fixed point of a contraction mapping. Furthermore, the value iteration algorithm (3.14) converges to the fixed point.

First, define the set of bounded functions $J : \mathbb{D} \to \mathbb{R}$ as $\mathbb{F}$. Then, let T^* be an operator on $\mathbb{F}$, and for any $J \in \mathbb{F}$, T^*J is another function with domain $\mathbb{D}$, whose value at β is defined as

$$(T^*J)(\beta) = \gamma \mathop{\mathbb{E}}_{X'|\beta}[\max_{a' \in \mathbb{A}([\beta, X'])} \{r([\beta, X'], a') + \sum_{i=1}^{\mathcal{N}(a')} p_i(\beta, a') J(\varrho_i([\beta, X'], a'))\}].$$

By Assumption 3.1, it is easy to check that, given J is bounded, T^*J is bounded (i.e., $T^*J \in \mathbb{F}$). Therefore, T^* is a mapping from $\mathbb{F}$ to $\mathbb{F}$. It is shown in [81, p. 211] that $\mathbb{F}$ is complete under the maximum norm. Furthermore, as shown in the following, T^* is a contraction mapping under the maximum norm with modulus γ. Therefore, the contraction theory applies to T^*.

Due to the contraction theory [81, p. 209], there exists a unique fixed point for T^*, denoted as J^*, such that $T^*J^* = J^*$, i.e., function J^* does not change under operator T^*. Note that equation $T^*J^* = J^*$ is exactly the after-state Bellman equation (3.13). Therefore, we have shown that there is a unique solution to (3.13).

In addition, the contraction theory [81, p. 209] states that, for arbitrary function $J_0 \in \mathbb{F}$, $\lim_{l \to \infty} T^{*l} J_0 = J^*$. Note that $T^{*l} J_0$ means the function that is generated by, starting from J_0, iteratively applying operator T^* on previously generated function for l times, which exactly describes the value iteration algorithm (3.14). This has proved the value iteration algorithm (3.14) converges to J^*.

Hence, there only remains to show that T^* is a contraction mapping. Given any two functions J_1, $J_2 \in \mathbb{F}$, for β that satisfies $(T^*J_1)(\beta) \geq (T^*J_2)(\beta)$, we have

$$\begin{aligned} 0 &\leq (T^*J_1)(\beta) - (T^*J_2)(\beta) \\ &= \gamma \mathop{\mathbb{E}}_{X'|\beta} \Big[\max_{a_1 \in \mathbb{A}([\beta, X'])} \big\{ r([\beta, X'], a_1) + \sum_{i=1}^{\mathcal{N}(a_1)} p_i(\beta, a_1) J_1(\rho_i([\beta, X'], a_1)) \big\} \end{aligned}$$

$$
\begin{aligned}
&\quad - \max_{a_2 \in \mathbb{A}([\beta, X'])} \Big\{ r([\beta, X'], a_2) + \sum_{i=1}^{\mathcal{N}(a_2)} p_i(\beta, a_2) J_2(\rho_i([\beta, X'], a_2)) \Big\} \Big] \\
&\leq \gamma \mathop{\mathbb{E}}_{X'|\beta} \Big[r([\beta, X'], a_1^*) + \sum_{i=1}^{\mathcal{N}(a_1^*)} p_i(\beta, a_1^*) J_1(\rho_i([\beta, X'], a_1^*)) \\
&\quad - r([d, X'], a_1^*) - \sum_{i=1}^{\mathcal{N}(a_1^*)} p_i(\beta, a_1^*) J_2(\rho_i([\beta, X'], a_1^*)) \Big] \\
&= \gamma \mathop{\mathbb{E}}_{X'|\beta} \Big[\sum_{i=1}^{\mathcal{N}(a_1^*)} p_i(\beta, a_1^*) \times \big(J_1(\rho_i([\beta, X'], a_1^*)) - J_2(\rho_i([d, X'], a_1^*)) \big) \Big] \\
&\leq \gamma \mathop{\mathbb{E}}_{X'|\beta} \Big[\sum_{i=1}^{\mathcal{N}(a_1^*)} p_i(a_1^*) ||J_1 - J_2|| \Big] = \gamma ||J_1 - J_2||, \qquad (3.27)
\end{aligned}
$$

where

$$
a_1^* = \mathop{\arg\max}_{a_1 \in \mathbb{A}([\beta, X'])} \{ r([\beta, X'], a_1) + \sum_{i=1}^{\mathcal{N}(a_1)} p_i(\beta, a_1) J_1(\rho_i([\beta, X'], a_1)) \},
$$

and $|| \cdot ||$ is the maximum norm.

For β that satisfies $(T^* J_1)(\beta) < (T^* J_2)(\beta)$, we can get

$$
0 < (T^* J_2)(\beta) - (T^* J_1)(\beta) \leq \gamma ||J_1 - J_2||, \qquad (3.28)
$$

following similar procedure by replacing J_1 to J_2, and vice versa. Therefore, combining (3.27) with (3.28) gives $|(T^* J_1)(\beta) - (T^* J_2)(\beta)| \leq \gamma ||J_1 - J_2||$ for all $\beta \in \mathbb{D}$, i.e., $||T^* J_1 - T^* J_2|| \leq \gamma ||J_1 - J_2||$. It has proved that T^* is a contraction mapping on $\mathbb{F}$ with modulus γ. And the proof of Theorem 3.1 is completed.

3.8.2 Proof of Theorem 3.2

With $s = [d, x]$, define a function

$$
G(s) \triangleq \max_{a \in \mathbb{A}(s)} \big\{ r(s, a) + \sum_{i=1}^{\mathcal{N}(a)} p_i(d, a) J^*(\varrho_i(s, a)) \big\}. \qquad (3.29)
$$

Expanding $J^*(\varrho(s, a))$ from equation (3.13) gives

$$
\begin{aligned}
G(s) = \max_{a \in \mathbb{A}([d,x])} \Big\{ & r(s, a) + \sum_{i=1}^{\mathcal{N}(a)} p_i(d, a) \times \\
& \gamma \mathop{\mathbb{E}}_{X'|\varrho_i(s,a)} \big[\max_{a' \in \mathbb{A}([\varrho_i(s,a), X'])} \{ r(\varrho_i(s, a), X', a') + \sum_{j=1}^{\mathcal{N}(a')} p_j(d', a') J(\varrho_j(\varrho_i(s, a), X', a')) \} \big] \Big\}
\end{aligned}
$$

$$= \max_{a\in\mathbb{A}([d,x])} \Big\{ r(s,a) + \gamma \sum_{i=1}^{\mathcal{N}(a)} p_i(d,a) \mathop{\mathbb{E}}_{X'|\varrho_i(s,a)} [G(\varrho_i(s,a), X')] \Big\}$$
$$= \max_{a\in\mathbb{A}([d,x])} \Big\{ r(s,a) + \gamma \mathbb{E}\left[G(S') | s, a \right] \Big\}, \tag{3.30}$$

where the definition of G implies the second equality, and (3.11) implies the last equality. Note that (3.30) is exactly the state Bellman equation (3.9). Therefore, function $G = V^*$ solves (3.9), and the relationship (3.15) is established. Finally, with (3.29) and the definition of the after-state Bellman equation (3.13), the relationship (3.16) is established, which completes the proof.

3.8.3 Proof of Theorem 3.3

For Algorithm 3.1, we define two operators H and $\hat{H}$. Let H be an operator on functions $\mathbb{K} \mapsto \mathbb{R}$. Applying H on a function g, i.e., Hg, gives another function with domain $\mathbb{K}$, and its value at k is defined as

$$(Hg)(k) = \gamma \mathop{\mathbb{E}}_{X'|q(k)} [\max_{a'\in\mathbb{A}([q(k),X'])} \{ r([q(k),X'],a') + \sum_{i=1}^{\mathcal{N}(a')} p_i(q(k),a') g(\omega(\varrho_i([q(k),X'],a'))) \}].$$

Similarly, define another operator on functions $\mathbb{K} \mapsto \mathbb{R}$ as

$$(\hat{H}g)(k) = \gamma \max_{a'\in\mathbb{A}([q(k),X'])} \{ r([q(k),X'],a') + \sum_{i=1}^{\mathcal{N}(a')} p_i(q(k),a') g(\omega(\varrho_i([q(k),X'],a'))) \},$$

where X' is a random variable with PDF $\mathrm{f}_X(\cdot|q(k))$. Note that the outcome of $\hat{H}g$ is random, and depends on the realization of X'.

Note that, in Algorithm 3.1, at any iteration l, $g_l(k)$ does not change for $k \notin \bar{K}_l$. Therefore, the step size value $\alpha_l(k)$, $\forall k \notin \bar{K}_l$, does not affect the algorithm. By defining $\alpha_l(k) = 0$, $\forall k \notin \bar{K}_l$, and with the operators H and $\hat{H}$, the updating (3.18) can rewritten as, $\forall k \in \mathbb{K}$:

$$g_{l+1}(k) = (1 - \alpha_l(k)) g_l(k) + \alpha_l(k)((Hg_l)(k) + w_l(k)) \tag{3.31}$$

where $w_l(k) = (\hat{H}g_l)(k) - (Hg_l)(k)$.

3.8.3.1 Proof of statement (i):

Due to the proposition 4.4 of [82, p. 156], we have following lemma.

Lemma 3.1. *Given following conditions,*

(a) H is a contraction mapping under maximum norm;

(b) for all k, $\sum_{l=0}^{\infty}\alpha_l(k)=\infty$, and $\sum_{l=0}^{\infty}\alpha_l^2(k)<\infty$;

(c) for all k and l, $\mathbb{E}[w_l(k)|g_l]=0$;

(d) there exist constant C_1 and C_2 such that $\mathbb{E}[w_l^2(k)|g_l]\le C_1+C_2||g_l||^2$;

the sequence of functions $\{g_l\}_l$ generated from iteration (3.31) converges to a function g_∞ with probability 1, and the limiting function g_∞ satisfying $Hg_\infty=g_\infty$.

We prove the statement (i) of Theorem 3.3 by checking the four conditions of Lemma 3.1 as follows. First, the contraction mapping condition (a) of H can be established in a similar procedure as the proof of Theorem 3.1, and is omitted here. Then, due to assumptions (3.23) and (3.24) of Theorem 3.3, the condition (b) about α_l is satisfied. In addition, we have $\mathbb{E}[w_l(k)|g_l]=0$ via the definition of H and $\hat{H}$. Therefore, the condition (c) is satisfied. Finally, we have to show the condition (d): the bounded variance property of w_l. For given k and l, we define a function as

$$I(x)=\gamma\max_{a'\in\mathbb{A}([q(k),x])}\{r([q(k),x],a')+\sum_{i=1}^{\mathcal{N}(a')}p_i(q(k),a')g_l(\omega(\varrho_i([q(k),x],a')))\}.$$

With the notation $I(x)$, we have

$$\begin{aligned}
&\mathbb{E}[w_l^2(k)|g_l]=\mathop{\mathbb{E}}_{X'|q(k)}\left[\left(I(X')-\mathop{\mathbb{E}}_{Y'|q(k)}[I(Y')]\right)^2\middle|g_l\right]=\mathop{\mathbb{E}}_{X'|q(k)}\left[\left(\mathop{\mathbb{E}}_{Y'|q(k)}[I(X')-I(Y')]\right)^2\middle|g_l\right]\\
&\le\mathop{\mathbb{E}}_{X'|q(k)}\left[\left(\mathop{\mathbb{E}}_{Y'|q(k)}[2\max\{\big|I(X')\big|,\big|I(Y')\big|\}]\right)^2\middle|g_l\right]\\
&\le\mathop{\mathbb{E}}_{X'|q(k)}\left[\left(\mathop{\mathbb{E}}_{Y'|q(k)}[2|I(X')|]\right)^2\middle|g_l\right]+\mathop{\mathbb{E}}_{X'|q(k)}\left[\left(\mathop{\mathbb{E}}_{Y'|q(k)}[2\big|I(Y')\big|]\right)^2\middle|g_l\right]\\
&\overset{\text{ⓐ}}{\le}\mathop{\mathbb{E}}_{X'|q(k)}\left[\left(2\big|I(X')\big|\right)^2\middle|g_l\right]+\mathop{\mathbb{E}}_{X'|q(k)}\left[\left(2L_1+2||g_l||\right)^2\middle|g_l\right]\\
&\overset{\text{ⓑ}}{\le}8L_2+8||g_l||^2+8L_1^2+8||g_l||^2=8(L_2+L_1^2)+16||g_l||^2,
\end{aligned}$$

where the inequalities ⓐ and ⓑ holds from the Assumption 3.1 and the fact that $(x+y)^2\le 2x^2+2y^2$ for any real value x and y. Therefore, it is proven that $\mathbb{E}[w_l^2(k)|g_l]$ is bounded by $8(L_2+L_1^2)+16||g_l||^2$, which completes the proof of the statement (i) of Theorem 3.3.

3.8.3.2 Proof of statement (ii):

First, define an partial order for functions $\mathbb{K} \mapsto \mathbb{R}$ as follows. If $g_1(k) \leq g_2(k)$, $\forall k$, we say $g_1 \leq g_2$. It is easy to check that, given any two functions g_1 and g_2 satisfying $g_1 \leq g_2$, we have $Hg_1 \leq Hg_2$.

Then, define a function $\bar{g}(k) \triangleq \inf_{\beta \in \mathbb{D}(k)} J^*(\beta) + \frac{\xi}{1-\gamma}$. Applying H on $\bar{g}$ gives

$$
\begin{aligned}
(H\bar{g})(k) &= \gamma \mathop{\mathbb{E}}_{X'|q(k)} \Big[\max_{a' \in \mathbb{A}([q(k),X'])} \big\{ r([q(k),X'],a') + \sum_{i=1}^{\mathcal{N}(a')} p_i(q(k),a') \bar{g}(\omega(\varrho_i([q(k),X'],a')) \big\} \Big] \\
&\overset{\text{ⓐ}}{\leq} \gamma \mathop{\mathbb{E}}_{X'|q(k)} \Big[\max_{a' \in \mathbb{A}([q(k),X'])} \big\{ r([q(k),X'],a') + \sum_{i=1}^{\mathcal{N}(a')} p_i(q(k),a') \big(J^*(\varrho_i([q(k),X'],a')) + \frac{\xi}{1-\gamma} \big) \big\} \Big] \\
&\overset{\text{ⓑ}}{=} J^*(q(k)) + \frac{\gamma\xi}{1-\gamma} \overset{\text{ⓒ}}{\leq} \inf_{\beta \in \mathbb{D}(k)} J^*(\beta) + \xi + \frac{\gamma\xi}{1-\gamma} = \bar{g}(k),
\end{aligned}
$$

where equality ⓐ is due to the definition of $\bar{g}(k)$, equality ⓑ comes from the after-state Bellman equation (3.13), and inequality ⓒ is due to the definition of ξ in (3.21). Therefore, we have $(H\bar{g})(k) \leq \bar{g}(k)$ for all k, i.e., $H\bar{g} \leq \bar{g}$.

Combining the fact that $Hg_1 \leq Hg_2$, if $g_1 \leq g_2$, with the fact that $H\bar{g} \leq \bar{g}$, we have $H^k\bar{g} \leq \bar{g}$, where H^k means applying H operator k times. Then, due to Lemma 3.1 in the proof of statement (i), we have $\lim_{k\to\infty} H^k\bar{g} = g_\infty \leq \bar{g}$, which means $g_\infty(k) \leq \inf_{\beta \in \mathbb{D}(k)} J^*(\beta) + \frac{\xi}{1-\gamma}$, $\forall\, k$. Therefore, we get $J^*(\beta) \geq g_\infty(\omega(\beta)) - \frac{\xi}{1-\gamma}$, $\forall\, \beta$. From the definition of J_∞ in (3.25), $J^*(\beta) - J_\infty(\beta) \geq -\frac{\xi}{1-\gamma}$, $\forall\, \beta$, follows.

On the other hand, defining $\underline{g}(k) = \sup_{\beta \in \mathbb{D}(k)} J^*(\beta) - \frac{\xi}{1-\gamma}$ and following the similar procedure, we can prove $H\underline{g} \geq \underline{g}$, and therefore, get $J^*(\beta) \leq g_\infty(\omega(\beta)) + \frac{\xi}{1-\gamma}$. In turn, it implies $J^*(\beta) - J_\infty(\beta) \leq +\frac{\xi}{1-\gamma}$, which completes the proof of statement (ii) in Theorem 3.3.

3.8.3.3 Proof of statement (iii):

For any policy π, define an operator T^π on $\mathbb{F}$ as[10]

$$
(T^\pi J)(\beta) = \gamma \mathop{\mathbb{E}}_{X'|\beta} [r([\beta,X'],\pi) + \sum_{i=1}^{\mathcal{N}(\pi)} p_i(\beta,\pi) J(\varrho_i([\beta,X'],\pi))], \tag{3.32}
$$

[10] $\mathbb{F}$ is defined in Section 3.8.1.

where π inside r, $\mathcal{N}$, p_i and ϱ_i denoting $\pi([\beta, X'])$. And from the state transition kernel (3.11), J^{π_∞} as defined by (3.22) can be recursively rewritten as

$$J^{\pi_\infty}(\beta) = \gamma \mathop{\mathbb{E}}_{X'|\beta}[r([\beta, X'], \pi_\infty) + \sum_{i=1}^{\mathcal{N}(\pi_\infty)} p_i(\beta, \pi_\infty) J^{\pi_\infty}(\varrho_i([\beta, X'], \pi_\infty))].$$

By comparing with T^π in (3.32), we have

$$T^{\pi_\infty} J^{\pi_\infty} = J^{\pi_\infty}. \tag{3.33}$$

In addition, similar to the proof of Theorem 3.1, T^π can be shown to be a contraction mapping with modulus γ, which means

$$||T^{\pi_\infty} J_1 - T^{\pi_\infty} J_2|| \leq \gamma ||J_1 - J_2|| \tag{3.34}$$

for any J_1 and J_2. Besides, from the definitions of $\hat{\pi}(\cdot|g_\infty)$ (i.e., π_∞) in (3.17) and J_∞ in (3.25), we have (T^* defined in Section 3.8.1)

$$T^{\pi_\infty} J_\infty = T^* J_\infty. \tag{3.35}$$

Furthermore, from statement (ii) of Theorem 3.3, we have

$$||J^* - J_\infty|| \leq \frac{\xi}{1-\gamma}. \tag{3.36}$$

Finally, it is shown in the proof of Theorem 3.1 that

$$T^* J^* = J^*, \tag{3.37}$$

and

$$||T^* J_1 - T^* J_2|| \leq \gamma ||J_1 - J_2|| \tag{3.38}$$

for any J_1 and J_2.

By combining the above results, we have

$$\begin{aligned}
||J^{\pi_\infty} - J^*|| &\overset{\text{(a)}}{=} ||T^{\pi_\infty} J^{\pi_\infty} - J^*|| \\
&\overset{\text{(b)}}{\leq} ||T^{\pi_\infty} J^{\pi_\infty} - T^{\pi_\infty} J_\infty|| + ||T^{\pi_\infty} J_\infty - J^*|| \\
&\overset{\text{(c)}}{\leq} \gamma ||J^{\pi_\infty} - J_\infty|| + ||T^* J_\infty - T^* J^*|| \\
&\overset{\text{(d)}}{\leq} \gamma ||J^{\pi_\infty} - J^*|| + \gamma ||J^* - J_\infty|| + \gamma ||J_\infty - J^*|| \\
&\overset{\text{(e)}}{\leq} \gamma ||J^{\pi_\infty} - J^*|| + \frac{2\gamma\xi}{1-\gamma},
\end{aligned} \tag{3.39}$$

where ⓐ is due to (3.33); ⓑ is the triangle inequality; ⓒ comes from (3.34), (3.35) and (3.37); ⓓ is due to the triangle inequality and (3.38), and ⓔ is due to (3.36). Finally, from (3.39), we have $||J^{\pi_\infty} - J^*|| \leq \frac{2\gamma\xi}{(1-\gamma)^2}$, which proves the statement (iii) of Theorem 3.3.

Chapter 4

Optimal Selective Transmission for Energy-Harvesting Wireless Sensors

4.1 Introduction

Wireless sensor networks, consisting of spatially-dispersed autonomous sensors, can monitor physical or environmental conditions [38, 39]. Important applications include environmental monitoring, industrial process monitoring and control, machine learning, data gathering and many more. Ubiquitous deployment of wireless sensor networks is thus a key enabler of the Internet of Things [83], where the sensors must be autonomous with limited energy and computational resources. To aid this goal, the sensor nodes may be powered by harvested energy [84] from ambient sources (solar, wind and others [74]). This enhances the energy self-sustainability of the nodes and consequently, the life time of the network is constrained by hardware limits, not by battery capacity [14]. Hence, energy-harvesting promises enhanced network lifetime, as well as a more sustainable evolution of wireless sensor networks.

However, due to finite battery capacity and the randomness of harvested energy, energy depletion can still occur when a sensor node attempts to transmit packets. To avoid this, a sensor node can evaluate/quantify the **priority** of data packets and then decide whether to send or not. For example, data packets containing information of enemy attacks [40] or fire alarms [41] may have higher priority. So low-priority packets may be dropped when available energy is limited, which will allow the sensor to transmit more important packets in a long term. Such *selective transmission* strategies

were studied by [85, 86] in conventional wireless sensor networks and extended to energy-harvesting settings in [87–90].

In [85], a sensor node considers its available energy and the priority of a packet to decide whether the packet should be transmitted or not. This policy maximizes the expected total priority of transmitted packets by the sensor. To enhance this process, work [86] adds a success index, a measure of the likelihood that a transmitted packet reaches its destination, e.g., a sink node. Therefore, when making its transmission decision, each sensor takes decisions of other sensors into consideration through the use of the success indices, which may improve overall performance. Nevertheless, the use of success indices introduces communication overhead, as the success index for each packet has to be passed from the sink node to all sensors along the packet's routing path.

Works [87–90] studied selective transmission in energy-harvesting wireless sensor networks. In [87], the harvested energy of a sensor is modeled as a random variable that takes value of 0 or 1, and the energy expense for each packet transmission is always defined as one. Given these assumptions, the sensor's battery dynamic can be analyzed by the Markov chain theory, which is then used to develop the transmission policy. Using the same assumptions on energy-harvesting and energy expense, work [88] derived the optimal transmission policy. In addition, work [88] proposed a low-complexity balanced policy: if the priority of a packet exceeds a predefined threshold, it is transmitted. Balanced policy is designed to ensure that the expected energy consumption equals the expected energy-harvesting gain, leading to energy neutrality. This ensures energy use efficiency while reducing energy outage risks. Work [89] extended the result of [88] to the case where there exists temporal correlation in the energy-harvesting process.

However, in order to find optimal and/or heuristic policies, the statistical distributions of data priority and/or energy-harvesting process are needed in [85–89]. In addition, works [87–89] assumed one unit of energy for both energy replenishment and energy consumption, which may not be practical. These two limitations are resolved in [90]. Specifically, this work models harvested energy and wireless fading as general random variables. In addition, based on the Robbins-Monro algorithms [76], work [15] learns the optimal transmission policy from the observed data, without their

statistical distributions.

4.1.1 Motivation, problem statement and contributions

In existing works [85–90], the selective transmission control is made based on energy status and packet priority, whereas the effect of fading on optimal decision making has not been considered. Thus, when it is decided to transmit, the transmission may fail due to wireless fading.

On the other hand, it can be beneficial via incorporating channel status into decision making. Specifically, based on channel state information (CSI), a node can estimate the necessary transmission power for achieving reliable communication. The node may choose not to transmit for energy saving, if CSI indicates the occurrence of deep fading. The node may also take the advantage of good channel status by transmitting with lower power. Therefore, selective transmission with CSI exploited may have more efficient use of energy, compared with those policies in [85–90].

Note that, for obtaining CSI, the node must estimate the wireless channel and thus sends pilot signals to the receiver side and receives feedback from it. Considering that the length of the pilot signals is much shorter than that of the data packets [91], the energy required for channel estimation is small compared to that required for data transmission. Moreover, for slowly-fading quasi-static channels, which are quite typical in wireless sensor networks, the channel estimation can be done less frequently.

With aforementioned motivations, we consider selective transmissions of a wireless sensor node, to optimize its energy usage and to exploit CSI. The node decides to send or not to send by considering battery status, data priority and fading status, whereas only the first two factors are considered in [85–90]. In addition, we assume that the node does not know the statistical distribution of the battery status, data priority, or fading status. This lack of distributional knowledge must therefore be reflected in the solving of optimal transmission policy. Considering all the aforementioned challenges, we make the following contributions.

a) We model the selective transmission problem as a continuous-state MDP [49] (also see Chapter 2.1). The optimal policy is derived from an after-state value function. This approach transforms the three-dimensional control problem (i.e., on battery

status, priority, and fading status) to a one-dimensional after-state function problem (i.e., on the "after-state" battery status). As a result, control of transmission is greatly simplified.

b) The structural properties of the after-state value function and the optimal policy are analyzed. We prove that the after-state value function is differentiable and non-decreasing, and that the optimal policy is threshold-based with respect to data priority and channel status.

c) To address the difficulty of representing the continuous after-state value function, we find a representational approximation. For preserving the discovered structural properties, we propose to use a monotone neural network (MNN) [92], and prove that it is a well-designed approximation for the after-state value function, which is differentiable and non-decreasing.

d) We develop a learning algorithm to train the proposed MNN to approximate the after-state value function. The learning process exploits data samples (but not the distribution information, which is unknown in our problem setting). The trained MNN can construct a near-optimal transmission policy. With simulations, we demonstrate the learning efficiency of the proposed algorithm, and also the performance achieved by the learned policy.

The rest of chapter is organized as follows. Section 4.2 describes the system model and formulates the selective transmission control problem. Section 4.3 derives and analyzes the optimal transmission policy based on an after-state value function. Section 4.4 proposes an MNN to approximate the after-state value function, and develops a learning algorithm to train the proposed MNN. Section 4.5 provides simulations of the proposed algorithm and learned policy.

4.2 System Model and Problem Formulation

4.2.1 Operation cycles

We consider a single link with one wireless sensor node (transmitter) and its receiver. In the sequel, when we say "the node", it means the sensor node. The time is partitioned into cycles, where the duration of a cycle is random (Fig. 4.1). A cycle

(say cycle t) begins with a silent period, in which the node waits until a data packet arrives. When that occurs, the silent period ends and an active period starts. During this active period, the node has to decide whether to transmit the received packet or discard it. After the packet is transmitted or discarded, cycle t ends and the silent period of cycle $t+1$ starts. At the same time, the node obtains energy replenishment with amount e_t, which is harvested during cycle t. Note that the duration of a cycle is the interval between two successive data packets' arrivals, which can be random, but is assumed to be long enough to perform necessary tasks in an active period, i.e., data reception, channel estimation, and possible transmission.

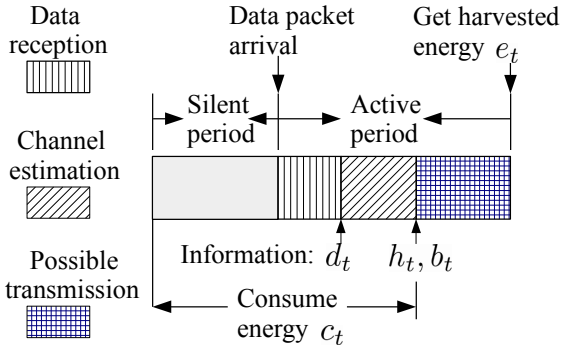


Figure 4.1: Cycle structure

4.2.2 States and actions

In the active period of cycle t, the node makes a transmission decision based on state $s_t = [b_t, h_t, d_t]$, where b_t is the remaining energy, h_t is the energy needed for transmission and d_t is the packet priority. These quantities are detailed below.

- The node receives and decodes a data packet. It is assumed that the node is able to evaluate the priority d_t of the packet via, for example, reading the packet contents. Here a higher priority value d_t means more importance.

- The node sends a suitable pilot signal to the receiver, and obtains the channel power gain z_t (CSI) from the receiver's feedback. The node then uses z_t to estimate the required transmit energy h_t based on the full channel inversion power control scheme [93], which ensures a certain targeted signal power at the receiver. Without loss of generality, we assume a unit-targeted receiving power and a unit transmission duration, and thus, the required energy for transmission

can be given as $h_t = 1/z_t$.

- $b_t \in [0, 1]$ represents the remaining energy in the node's battery after the energy expenditure (denoted as c_t) in cycle t for standing by (in the silent period of cycle t), data reception and channel estimation (in the active period of cycle t). Note that the battery's capacity is set to normalized unit energy.

The decision variable $a_t = 1$ represents "transmit" and $a_t = 0$ represents "discard". If $a_t = 0$, the packet is dropped with zero energy consumption. On the other hand, if $a_t = 1$ is chosen, and

- if energy is sufficient ($b_t \geq h_t$), the node consumes energy h_t and consequently the packet will be delivered successfully.

- if the energy is not sufficient, packet delivery fails, and the remaining energy is exhausted, i.e., the energy consumption is b_t.

4.2.3 State dynamics

This subsection models the relationship between s_{t+1} and (s_t, a_t). We assume that $\{h_t\}_t$ are independent and identically distributed (i.i.d.) continuous random variables with a PDF $f_H(x)$. Similarly, $\{d_t\}_t$ are i.i.d. continuous random variables with PDF $f_D(x)$. Therefore, h_{t+1} and d_{t+1} are independent of (s_t, a_t).

However, b_{t+1} is affected by (s_t, a_t), since different combinations of (b_t, h_t, a_t) cause different energy consumptions (Section 4.2.2). Moreover, b_{t+1} also depends on e_t. Finally, during cycle $t + 1$, the waiting in the silent period, and the data reception and channel estimation in the active period all consume energy, whose total amount is denoted as c_{t+1}. Therefore, b_{t+1} is further affected by c_{t+1}. In summary, we have

$$b_{t+1} = ((\varrho(s_t, a_t) + e_t)^- - c_{t+1})^+, \tag{4.1}$$

where $(x)^- \triangleq \min\{x, 1\}$, $(x)^+ \triangleq \max\{x, 0\}$, and

$$\varrho(s_t, a_t) = (b_t - h_t \cdot a_t)^+. \tag{4.2}$$

We assume that $\{e_t\}_t$ and $\{c_t\}_t$ are, respectively, i.i.d. continuous random variables with PDF $f_E(x)$ and PDF $f_C(x)$. In Lemma 4.1 of Section 4.3.3, we will show

that, given the value of $\varrho(s_t, a_t)$, b_{t+1} is a time-independent continuous random variable, i.e., its conditional PDF can be written as $f_B(\cdot|\varrho(s_t, a_t))$.

Therefore, given state s and action a at current cycle, the state $s' = [b', h', d']$ at next cycle can be characterized by the following conditional PDF, named as state transition kernel,

$$f(s'|s, a) = f_H(h') \cdot f_D(d') \cdot f_B(b'|\varrho(s, a)). \tag{4.3}$$

In the sequel, we use $(\cdot)'$ to denote a variable in the next cycle.

4.2.4 Rewards

At cycle t, a packet is successfully transmitted, if and only if $a_t = 1$ and $b_t \geq h_t$. Also considering that the packet's priority is quantified by d_t, the immediate reward of deciding on action a_t in presence of state s_t is defined as

$$r(s_t, a_t) \triangleq \mathbb{1}(a_t = 1) \cdot \mathbb{1}(b_t \geq h_t) \cdot d_t, \tag{4.4}$$

where $\mathbb{1}(\cdot)$ is an indicator function.

4.2.5 Problem formulation

A policy is designed to maximize the expected total rewards over an infinite time horizon. We consider only the set of all deterministic stationary policies, denoted as Π. A deterministic stationary policy $\pi \in \Pi$ is a time-independent mapping from states to actions, i.e., $\pi : \mathbb{S} \mapsto \mathbb{A}$, where $\mathbb{S} = \{ s = [b, h, d] | \, b \in [0, 1], h \in \mathbb{R}_+, d \in \mathbb{R}_+ \}$ denotes the state space, and $\mathbb{A} = \{0, 1\}$ denotes the action space.

Since the node continuously harvests energy from the environment, potentially over many cycles, the total rewards can be infinite. To avoid this, discounting is perhaps the most analytically tractable and most widely studied approach. A discounting factor $0 < \gamma < 1$ is used to ensure the infinite summation is bounded, and therefore, for each π, the objective value obtained following policy π is defined as

$$V^\pi = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, \pi(s_t))], \tag{4.5}$$

where the expectation $\mathbb{E}[\cdot]$ is defined over the distribution of initial state s_0 and state trajectory $\{s_t\}_{t=1}^{\infty}$ induced by actions $\{\pi(s_t)\}_{t=0}^{\infty}$. Note that if $\gamma \approx 1$, V^π can be

(approximately) interpreted as the expected total priority of sent packets by policy π.

Our target is to solve an optimal policy π^* such that

$$\pi^* = \arg\sup_{\pi \in \Pi} \{V^\pi\}. \tag{4.6}$$

Therefore, via choosing transmission decision $\pi^*(s_t)$ at each cycle t, the expected total priority value of transmitted packets is maximized. In addition, since the node does not know the PDFs f_H, f_D, f_C and f_E, the solution of π^* must involve samples of the corresponding random variables.

4.3 Optimal Selective Transmission Policy

4.3.1 Standard results from MDP theory

The 4-tuple $< \mathbb{S}, \mathbb{A}, r, f >$, namely the state space, action space, reward function, and state transition kernel, defines an MDP. From basic MDP theory (see Chapter 2.1), policy π^* (4.6) can be constructed from state-value function $V^* : \mathbb{S} \mapsto \mathbb{R}$ as

$$\pi^*(s) = \arg\max_a \left\{ r(s,a) + \gamma \cdot \mathbb{E}\left[V^*(s') | s, a \right] \right\}, \tag{4.7}$$

where the expectation is taken over the next state s' given current s and a. In addition, V^* is a solution to the Bellman equation

$$V(s) = \max_a \left\{ r(s,a) + \gamma \cdot \mathbb{E}\left[V(s') | s, a \right] \right\}. \tag{4.8}$$

Finally, V^* can be computed recursively[1] by using (4.8).

Remark: Although V^* can be solved via (4.8), it is hard to compute π^* via (4.7). Specifically, (4.7) requires a conditional expectation over a random next state s', a computationally expensive task. We thus address this difficulty through a reformulation based on the after-state value function.

4.3.2 Reformulation based on after-state value function

An after-state (also known as post-decision state), which is an intermediate variable between two successive states, can be used to simplify the optimal control of cer-

[1] The recursive computation scheme is known as the value iteration algorithm. Section 4.3.2 provides an example of using it to compute the after-state value function. V^* can be similarly computed.

tain MDPs (see Chapter 2.1). The physical interpretation of after-state is problem-dependent.

We next define after-state for our problem. We also show that π^* can be defined over an after-state value function, which can be solved by a value iteration algorithm.

Physically, an "after-state" p_t of cycle t is the remaining energy after action a_t is performed but before harvested energy e_t is stored in the battery. Therefore, given state s_t and action a_t, the after-state is $p_t = \varrho(s_t, a_t)$. Recall that $\varrho(s_t, a_t) = (b_t - h_t \cdot a_t)^+$ (as defined in (4.2)). Hence, deriving from (4.3), the conditional PDF of state $s' = [b', h', d']$ of next cycle given after-state p at current cycle is

$$q(s'|p) \triangleq f_H(h') \cdot f_D(d') \cdot f_B(b'|p). \tag{4.9}$$

Hence, the term $\mathbb{E}\left[V^*(s')|s, a\right]$ inside (4.7) and (4.8), where the conditional expectation is defined with PDF (4.3), can be written as $\mathbb{E}\left[V^*(s')|\varrho(s, a)\right]$ whose expectation is defined with PDF (4.9) with $p = \varrho(s, a)$. Keeping this observation in mind, π^* is redefined as follows.

We define the after-state value function $J^* : [0, 1] \mapsto \mathbb{R}$ as

$$J^*(p) = \gamma \mathbb{E}[V^*(s')|p]. \tag{4.10}$$

Plugging (4.10) into (4.7), we have

$$\pi^*(s) = \arg\max_a \{r(s, a) + J^*(\varrho(s, a))\}. \tag{4.11}$$

Therefore, (4.11) provides an alternative formulation of the optimal policy. We next present a value iteration algorithm to solve for J^*.

Plugging (4.10) into (4.8), we have $V^*(s) = \max_a \{r(s, a) + J^*(\varrho(s, a))\}$. By replacing a with a', replacing s with s' and taking (γ-weighted) conditional expectation $\gamma \cdot \mathbb{E}[\cdot|p]$ on both sides, we further have $\gamma \cdot \mathbb{E}\left[V^*(s')|p\right] = \gamma \cdot \mathbb{E}\left[\max_{a'} \{r(s', a') + J^*(\varrho(s', a'))\}|p\right]$. Noticing that $\gamma \cdot \mathbb{E}\left[V^*(s')|p\right]$ on the left hand side is exactly the definition of $J^*(p)$, we have that J^* satisfies the following equation

$$J^*(p) = \gamma \cdot \mathbb{E}\left[\max_{a'} \{r(s', a') + J^*(\varrho(s', a'))\}|p\right]. \tag{4.12}$$

Finally, following a similar procedure as Theorem 3.1 in Chapter 3.4.3, J^* can be solved by a value iteration algorithm (under a technique assumption that random

variable d_t has finite mean). Specifically, initially with a bounded function J_0, the sequence of functions $\{J_k\}_{k=1}^K$ computed via, $\forall p \in [0,1]$,

$$J_{k+1}(p) \leftarrow \gamma \cdot \mathbb{E}\left[\max_{a'}\{r(s',a') + J_k(\varrho(s',a'))\}\middle| p\right], \tag{4.13}$$

converges to J^* when $K \to \infty$.

Remark: Different from (4.7) by which the optimal decision making needs conditional expectation, equation (4.11) shows that the optimal decision making can be directly made with J^* (without making expectation).

4.3.3 Properties of J^* and π^*

This subsection shows the properties of J^* and π^*. We begin with Lemma 4.1, whose proof is provided in Section 4.7.1.

Lemma 4.1. *Given that $p_t = p$, b_{t+1} is a continuous random variable whose distribution does not depend on t. In addition, denoting its conditional cumulative distribution function as $F_B(b|p)$, we have $F_B(b|p_1) \leq F_B(b|p_2)$, if $p_1 \geq p_2$. Finally, $F_B(b|p)$ is differentiable with respect to p.*

The after-state value function J^* can be theoretically derived from the value iteration algorithm (4.13). The results of Lemma 4.1 provide us a tool to analyze the conditional expectation operation $\mathbb{E}[\cdot|p]$ in (4.13). Via exploiting Lemma 4.1, Theorem 4.1 analyzes the structure of J^* with (4.13). The proof is provided in Section 4.7.2,

Theorem 4.1. *The after-state value function J^* is a differentiable and non-decreasing function with respect to battery level p.*

Note that π^* can be defined via J^* (4.11). Therefore, Theorem 4.1 can be used to analyze the structures of π^*, as shown in Theorem 4.2.

Theorem 4.2. *The optimal policy π^* has the following structure*

$$\pi^*([b,h,d]) = \begin{cases} 1 & \text{if } b \geq h \text{ and } d \geq J^*(b) - J^*(b-h), \\ 0 & \text{otherwise.} \end{cases} \tag{4.14}$$

Proof. From (4.11), we know that, $\pi^*([b,h,d]) = 1$ is equivalent to

$$\mathbb{1}(b \geq h) \cdot d + J^*((b-h)^+) \geq J^*(b). \tag{4.15}$$

Furthermore, (4.15) requires $b \geq h$, since otherwise we have $J^*(0) > J^*(b)$, which cannot hold as J^* is non-decreasing. Therefore, (4.15) is equivalent to $b \geq h$ and $d \geq J^*(b) - J^*(b-h)$. □

Corollary 4.1. *The optimal policy π^* is threshold based non-decreasing with respect to d and $-h$. To be specific, 1) given any b and h, if $\pi^*([b,h,d_1]) = 1$, then $\pi^*([b,h,d_2]) = 1$, for any $d_2 \geq d_1$; 2) given any b and d, if $\pi^*([b,h_1,d]) = 1$, then $\pi^*([b,h_2,d]) = 1$, for any $h_2 \leq h_1$.*

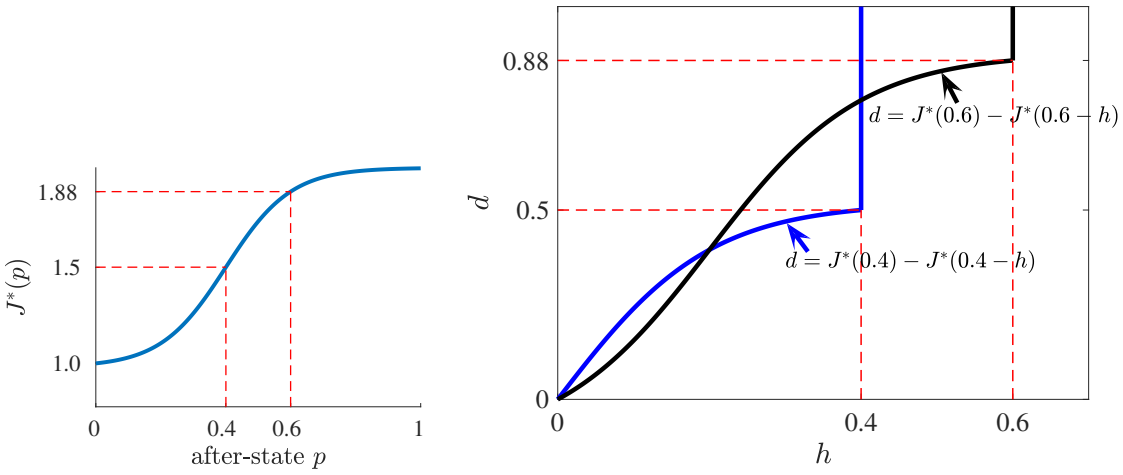
Proof. From Theorem 4.2, $\pi^*([b,h,d_1]) = 1$ implies $h \leq b$ and $d_1 \geq J^*(b) - J^*(b-h)$. Therefore, we have $d_2 > J^*(b) - J^*(b-h)$ for any $d_2 \geq d_1$, which implies $\pi^*([b,h,d_2]) = 1$.

Similarly, $\pi([b,h_1,d]) = 1$ implies $h_1 \leq b$ and $d > J^*(b) - J^*(b-h_1)$. And because J^* is non-deceasing, we have $h_2 \leq b$ and $d > J^*(b) - J^*(b-h_2)$ for any $h_2 \leq h_1$, which implies $\pi^*([b,h_2,d]) = 1$. □

Remark: Corollary 4.1 states that, for a given battery level, the optimal policy is to send, if the data priority and channel quality exceed certain thresholds.

4.3.4 An example of J^* and π^*

We now present an example of J^* and π^* in Fig. 4.2.



(a) An example of J^* (b) Decision boundaries of π^* at $b = 0.4$ and $b = 0.6$

Figure 4.2: Examples for after-state value function and optimal policy

$J^*(p)$ is the function shown in Fig. 4.2(a), which is non-decreasing and differentiable (Theorem 4.1). Based on $J^*(p)$, the optimal policy $\pi^*([b,h,d])$ is then determined based on (4.11). From Theorem 4.2, we know that, in the (h,d) space given battery level b, a decision boundary consisting of curve "$d = J^*(b) - J^*(b-h)$" and line "$h = b$" partitions the (h,d) space into two sub-spaces: in the sub-space on the upper-left side of the boundary, the decision is $\pi^*([b,h,d]) = 1$; in the sub-space on the bottom-right side of the boundary, the decision is $\pi^*([b,h,d]) = 0$. In Fig. 4.2(b), we show two examples for decision boundaries with $b = 0.4$ and $b = 0.6$, respectively. It is easily seen that $\pi^*([0.4,h,d])$ and $\pi^*([0.6,h,d])$ are threshold-based non-decreasing with respect to d and $-h$, as proved in Corollary 4.1. However, the threshold structure does not hold in dimension b. As one can see in Fig. 4.2(b), there is an area of (h,d) that $a = 1$ is chosen with $\pi^*([0.4,h,d])$, but $a = 0$ is chosen with $\pi^*([0.6,h,d])$.

4.4 Neural Network for Optimal Control

Section 4.3.2 shows that π^* can be effectively constructed by J^*, which, in turn, can be solved by the value iteration algorithm (4.13). However, the implementation of (4.13) is challenging. First, as the PDFs $f_E(\cdot)$, $f_D(\cdot)$ and $f_B(\cdot|p)$ are not available, we cannot compute $\mathbb{E}[\cdot|p]$. Second, because after-state p is a continuous variable over $[0,1]$, each iteration of (4.13) has to be computed over infinitely many p values.

Reinforcement learning provides a useful solution to approximately address both difficulties. Specifically, instead of exactly solving J^*, reinforcement learning targets an approximation of J^* via learning a parameter vector (i.e., a set of real values), while the learning process exploits data samples (rather than underlying distributions). In other words, the design of a reinforcement learning algorithm involves:

- parameterization: Parameterization decides how a parametric function $\hat{J}(p|\boldsymbol{\theta})$ is determined from a given parameter vector $\boldsymbol{\theta}$;
- parameter learning: Given a batch of data samples, a parameter vector $\boldsymbol{\theta}^*$ is learned, and $\hat{J}(p|\boldsymbol{\theta}^*)$ is used to approximate J^*.

With learned $\boldsymbol{\theta}^*$, we can construct the transmission policy as

$$\hat{\pi}(s|\boldsymbol{\theta}^*) = \arg\max_a \{r(s,a) + \hat{J}(\varrho(s,a)|\boldsymbol{\theta}^*)\}. \tag{4.16}$$

Comparing (4.16) with (4.11), we see that, if $\hat{J}(p|\boldsymbol{\theta}^*)$ approximates $J^*(p)$ well, the performance of $\hat{\pi}(s|\boldsymbol{\theta}^*)$ is close to that of $\pi^*(s)$ ([82, Chapter 6] provides rigorous statements).

In this section, we propose a reinforcement learning algorithm, which exploits monotone neural network (MNN) [92] for parameterization (Section 4.4.1) and learns the associated parameter vector via iteratively executing least square regression (Section 4.4.2). The learned parameter vector is applied for transmission control in Section 4.4.3.

4.4.1 Monotone neural network approximation

It is desired that a function parameterization provides sufficient representation ability, i.e., we can find a parameter vector $\boldsymbol{\theta}$ such that $\hat{J}(p|\boldsymbol{\theta})$ is close to $J^*(p)$. ANN [48, Chapter 4] (also see Chapter 2.2) seems to be a good option, as the universal approximation theorem [94] states that a three-layer ANN is able to approximate a continuous function to arbitrary accuracy.

However, we know that J^* is non-decreasing from Theorem 4.1, whereas (classical) ANNs include all types of continuous functions (not necessarily non-decreasing). This would make the learning of parameters inefficient, as a learning algorithm needs to search over a not-necessarily large function space.

With this motivation, we propose the use of an MNN [92] for parameterization. Mathematically, the parameterized function $\hat{J}(p|\boldsymbol{\theta})$ with the MNN is expressed as

$$\hat{J}(p|\boldsymbol{\theta}) = \left(\sum_{i=1}^{N} u_i^2 \sigma_H(w_i^2 p + \alpha_i) \right) + \beta, \tag{4.17}$$

with parameter vector

$$\boldsymbol{\theta} = [w_1, \cdots, w_N, \alpha_1, \cdots, \alpha_N, u_1, \cdots, u_N, \beta],$$

and function $\sigma_H(x) = 1/(1+e^{-x})$.

Function $\hat{J}(p|\boldsymbol{\theta})$ (4.17) is depicted in Fig. 4.3. It is actually a three-layered single-input-single-output ANN (with small modifications). Specifically, there is an input-layer with one single node, whose output represents the value of after-state p. In addition, there is a hidden-layer with N nodes. The input of the i-th node is the sum of weighted after-state value $w_i^2 \cdot p$ and hidden-layer bias α_i. And the input-output

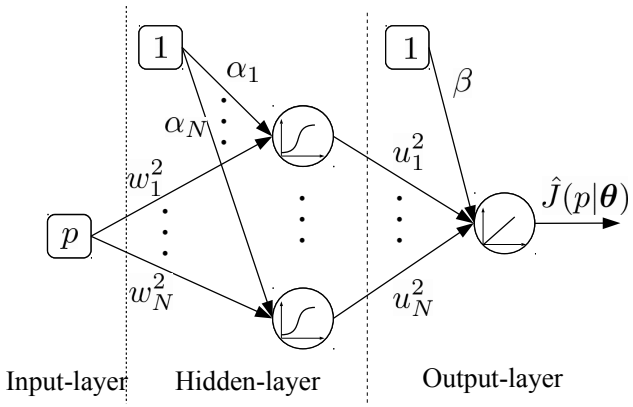


Figure 4.3: Monotone neural network

relationship of each hidden-layer node is defined by $\sigma_H(\cdot)$ (known as the sigmoid activation function, see Chapter 2.2). Finally, there is an output-layer with one node, whose output represents the ultimate approximated function value $\hat{J}(p|\boldsymbol{\theta})$. Its input is the summation of weighted outputs from the hidden-layer and the output-layer bias β. And the output of the output-layer node is equal to its input.

Note that, the key difference between MNN and classical ANN is the sign of weights. The MNN has non-negative weights, which is not necessary for classical ANN. But MNN ensures $\hat{J}(p|\boldsymbol{\theta})$ to be a non-decreasing function, which is proved in Theorem 4.3.

Theorem 4.3. *For any parameter θ, $\hat{J}(p|\boldsymbol{\theta})$ is a differentiable non-decreasing function.*

Proof. We have $\frac{d}{dp}\hat{J}(p|\boldsymbol{\theta}) = \sum_{i=1}^{N} u_i^2 \cdot w_i^2 \cdot \sigma_H(w_i^2 p + \alpha_i) \cdot (1 - \sigma_H(w_i^2 p + \alpha_i))$. It is easily verified that $\frac{d}{dp}\hat{J}(p|\boldsymbol{\theta}) \geq 0$, which completes the proof. □

In addition, Theorem 4.4 states that the proposed MNN has sufficient ability to represent any continuous and non-decreasing function.

Theorem 4.4. *For any continuous non-decreasing function $J : [0,1] \mapsto \mathcal{R}$, there exists an MNN with N hidden nodes and parameter vector $\boldsymbol{\theta}$, such that $J(p) - \hat{J}(p|\boldsymbol{\theta}) \leq \epsilon$, for any $p \in [0,1]$ and $\epsilon \geq 0$.*

Proof. It can be proven by [92, theorem 3.1], which considered a general case of representing a multiple-input-single-output non-decreasing function. □

Remark: Theorem 4.4 states that $\hat{J}(p|\boldsymbol{\theta})$ is able to approximate $J^*(p)$ to arbitrary accuracy via optimizing the parameter vector $\boldsymbol{\theta}$.

4.4.2 Fitted value iteration to train MNN

Fitted value iteration [95] is a state-of-the-art learning methodology that is especially useful for training neural networks for optimal control. When working with complex neural networks (with multiple hidden layers), it is entitled with the name of deep reinforcement learning [96, 97]. Here, we develop an algorithm, called FMNN (fitted value iteration with monotone neural network), via tailoring the fitted value iteration method into our problem. FMNN trains an MNN to approximate $J^*(p)$ via exploiting a batch of data samples. In the following, we first specify the required data. Then, the training process is presented.

4.4.2.1 Collecting training data

The required training data for FMNN is a batch of samples $\mathcal{F} = \{(p_m, s_m = [b_m, h_m, d_m])\}_{m=0}^{M-1}$, where $b_m \sim f_B(\cdot|p_m)$, $h_m \sim f_H(\cdot)$ and $d_m \sim f_D(\cdot)$. In the following, We provide two possible methods for collecting $\mathcal{F}$.

Suppose that there is a simulator that is able to generate data samples of random variables c_t, e_t, h_t and d_t that obey PDFs $f_C(\cdot)$, $f_E(\cdot)$, $f_H(\cdot)$ and $f_D(\cdot)$, respectively. In this case, we can first obtain $\{p_m\}_m$ by uniformly sampling over $[0, 1]$. Then, for a given p_m, we sample those random variables and get realizations (c, e, h, d). With these realizations, a valid data sample (p_m, s_m) can be obtained by setting $s_m = [((p_m + e)^- - c)^+, h, d]$. Finally, $\mathcal{F}$ is constructed by repeating the procedure for all p_m.

When such a simulator is not available, we can construct $\mathcal{F}$ via physically interacting with environments. Specifically, we can run a certain sampling policy $\pi_S(s)$ (such as the greedy policy, i.e., always choose to send if energy is sufficient). And during the execution of $\pi_S(s)$, we can observe a sample path of random variables $(..., s_{t-1} = [b_{t-1}, h_{t-1}, d_{t-1}], p_{t-1} = \varrho(s_{t-1}, \pi_S(s_{t-1})), s_t = [b_t, h_t, d_t], p_t = \varrho(s_t, \pi_S(s_t)), ...)$. Then, by setting $p_m = p_{t-1}$ and $s_m = [b_t, h_t, d_t]$, we are able to collect a valid sample (p_m, s_m). Finally, $\mathcal{F}$ is constructed by sweeping from $t = 0$ to M.

4.4.2.2 Fitting MNN Iteratively

Here, we present FMNN, which trains an MNN to approximate $J^*(p)$ with $\mathcal{F}$ via imitating the iterative computing scheme of (4.13).

Specifically, similar to (4.13), FMNN works iteratively. At the kth iteration, suppose that the current MNN parameter vector is $\boldsymbol{\theta}_k$, which defines a function $\hat{J}(p|\boldsymbol{\theta}_k)$ with (4.17). As suggested by (4.13), given current value function $J_k(p) = \hat{J}(p|\boldsymbol{\theta}_k)$, the updated value function should be

$$J_{k+1}(p) = \gamma \cdot \mathbb{E}\left[\max_a \left\{r(s', a) + \hat{J}(\varrho(s', a)|\boldsymbol{\theta}_k)\right\} \middle| p\right]. \tag{4.18}$$

Therefore, we wish to update the MNN's parameters to obtain a new function $J(p|\boldsymbol{\theta}_{k+1})$ that is close to $J_{k+1}(p)$.

To do so, from $\mathcal{F}$ and $\boldsymbol{\theta}_k$, we construct a batch of data

$$\mathcal{T}_k = \{(p_m, o_m)\}_{m=0}^{M-1}, \tag{4.19}$$

where

$$o_m = \gamma \cdot \max_a \left\{r(s_m, a) + \hat{J}(\varrho(s_m, a)|\boldsymbol{\theta}_k)\right\}. \tag{4.20}$$

Note that by comparing (4.20) with (4.18), o_m can be seen as a noisy realization of $J_{k+1}(p_m)$. In other words, given p_m as an input value, o_m defines the corresponding output of J_{k+1} (plus certain noise). Therefore, we may get a function that is close to $J_{k+1}(p)$ by training the MNN to fit the (noisy) input-output patterns contained in $\mathcal{T}_k$.

Specifically, given $\mathcal{T}_k$, the MNN parameter is updated as

$$\boldsymbol{\theta}_{k+1} = \arg\min_{\boldsymbol{\theta}} \{\mathcal{L}(\boldsymbol{\theta}|\mathcal{T}_k)\}, \tag{4.21}$$

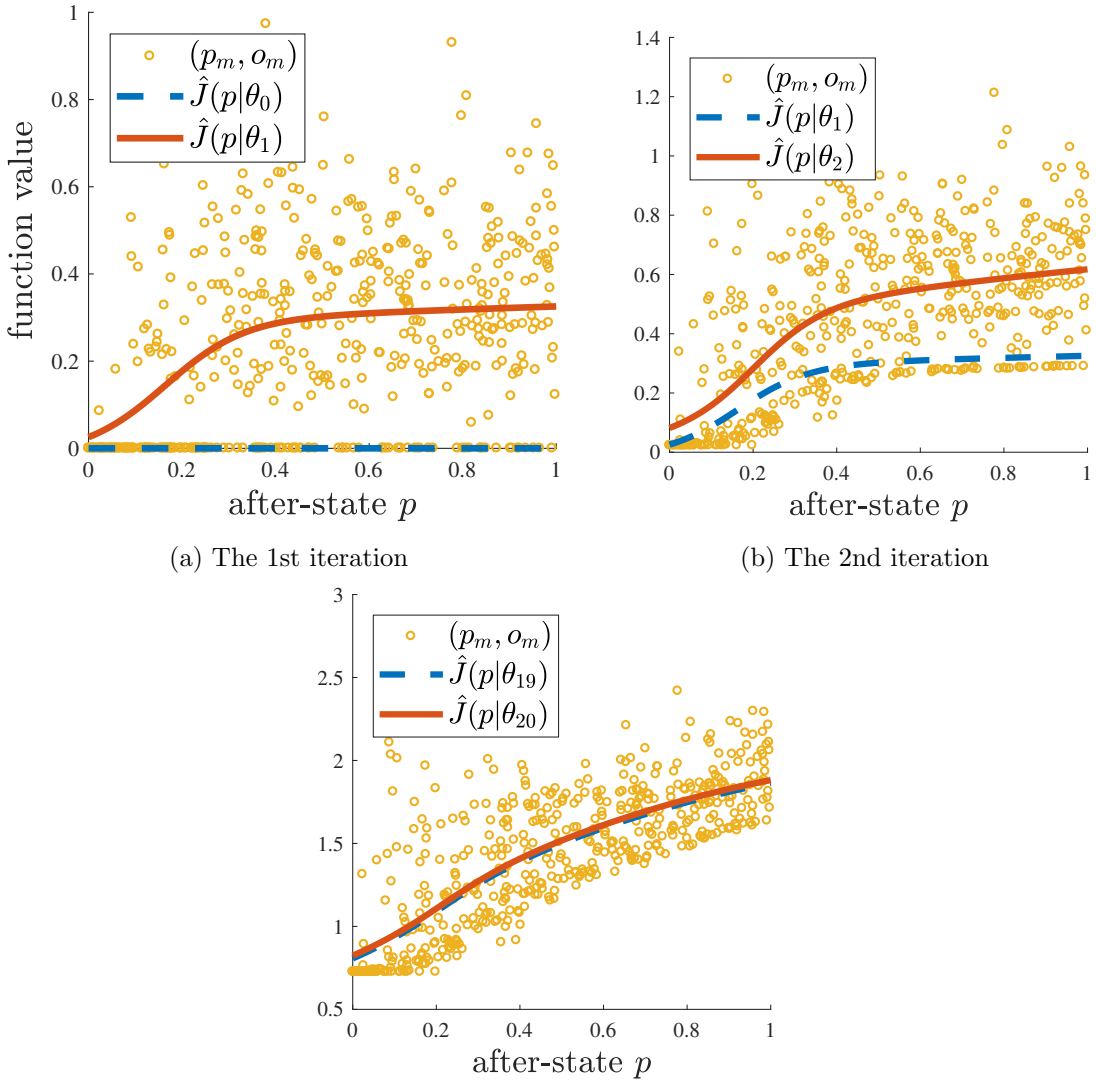
where

$$\mathcal{L}(\boldsymbol{\theta}|\mathcal{T}_k) = \frac{1}{2M} \sum_{m=0}^{M-1} \left(\hat{J}(p_m|\boldsymbol{\theta}) - o_m\right)^2 \tag{4.22}$$

(the solving of (4.21) is discussed in Section 4.4.2.3). That is, the output of updated function $\hat{J}(p|\boldsymbol{\theta}_{k+1})$ minimizes the square error with respect to data set $\mathcal{T}_k$, i.e., least square regression. Given sufficiently large M, this regression process can efficiently

average out data's randomness. Hence, the approximation error between $J_{k+1}(p)$ and $\hat{J}(p|\boldsymbol{\theta}_{k+1})$ should be small.

With $\boldsymbol{\theta}_{k+1}$, we can generate $\mathcal{T}_{k+1}$ from $\boldsymbol{\theta}_{k+1}$ and $\mathcal{F}$ (similar as (4.19)), and then solve $\boldsymbol{\theta}_{k+2}$ by fitting the MNN to $\mathcal{T}_{k+1}$ (similar as (4.21)). Iterations continue by repeating the procedure. The ultimate learned function $\hat{J}(p|\boldsymbol{\theta}_K)$ after K iterations should be close to $J^*(p)$ with sufficiently large K. For illustration, Fig. 4.4 shows the 1st, 2nd and 20th iterations during FMNN's execution. Summarizing above concepts, FMNN algorithm is presented in Algorithm 4.1.



(a) The 1st iteration

(b) The 2nd iteration

(c) The 20th iteration

Figure 4.4: Illustration of FMNN with $|\mathcal{F}| = 500$

Algorithm 4.1 FMNN: Approximate $J^*(p)$

Require: Data samples $\mathcal{F} = \{(p_m, s_m)\}_{m=0}^{M-1}$
Ensure: Learned MNN $\hat{J}(p|\boldsymbol{\theta}_K)$
procedure
  Randomly initialize parameter $\boldsymbol{\theta}_0$
  for k from 0 to $K-1$ **do**
    for m from 0 to $M-1$ **do**
      Get (p_m, s_m) as the mth element of $\mathcal{F}$
      Compute o_m from $\boldsymbol{\theta}_k$ and s_m via (4.20)
      Collect $\mathcal{T}_k(m) = (p_m, o_m)$
    end for
    Regression: $\boldsymbol{\theta}_{k+1} = \text{Fit}(\mathcal{T}_k, \boldsymbol{\theta}_k)$ (see Algorithm 4.2)
  end for
  Learned MNN is determined from (4.17) with $\boldsymbol{\theta} = \boldsymbol{\theta}_K$
end procedure

4.4.2.3 Train MNN with gradient descent

Here, we apply gradient descent to address (4.21) for solving MNN parameter $\boldsymbol{\theta}_{k+1}$ such that the represented function $\hat{J}(p|\boldsymbol{\theta}_{k+1})$ fits an input-output pattern $\mathcal{T}_k$ in least square error sense.

Gradient descent works by iteratively searching over the parameter space. Denote $\boldsymbol{\theta}^{(0)}$ as the initial search point, which can be intuitively set as current MNN parameter $\boldsymbol{\theta}_k$. By gradient descent, the parameter searching is conducted as follows

$$\boldsymbol{\theta}^{(l+1)} = \boldsymbol{\theta}^{(l)} - \xi^{(l)} \cdot \nabla \mathcal{L}(\boldsymbol{\theta}^{(l)}), \tag{4.23}$$

where $\xi^{(l)}$ is the updating step size and $\nabla \mathcal{L}(\boldsymbol{\theta}^{(l)})$ is the gradient of $\mathcal{L}$ (4.22) at $\boldsymbol{\theta}^{(l)}$. Given properly decreasing $\xi^{(l)}$ and sufficient number of iterations L, we set $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}^{(L)}$, which is considered as an approximated solution of (4.21).

Lastly, we derive $\nabla \mathcal{L}(\boldsymbol{\theta})$. With $\mathcal{T}_k = \{(p_m, o_m)\}_{m=0}^{M-1}$, the partial derivatives of $\mathcal{L}$

can be obtained as follows

$$\frac{\partial \mathcal{L}}{\partial \beta} = \frac{1}{M}\sum_{m=0}^{M-1} \epsilon_m, \tag{4.24}$$

$$\frac{\partial \mathcal{L}}{\partial u_i} = \frac{1}{M}\sum_{m=0}^{M-1} \left(\epsilon_m \cdot 2 \cdot u_i \cdot \sigma_H(w_i^2 \cdot p_m + \alpha_i)\right), \tag{4.25}$$

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \alpha_i} = \frac{1}{M}\sum_{m=0}^{M-1} \Big(&\epsilon_m \cdot u_i^2 \cdot \sigma_H(w_i^2 \cdot p_m + \alpha_i) \\ &\times \left(1 - \sigma_H(w_i^2 \cdot p_m + \alpha_i)\right)\Big),\end{aligned} \tag{4.26}$$

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial w_i} = \frac{1}{M}\sum_{m=0}^{M-1} \Big(&\epsilon_m \cdot u_i^2 \cdot \sigma_H(w_i^2 \cdot p_m + \alpha_i) \\ &\times \left(1 - \sigma_H(w_i^2 \cdot p_m + \alpha_i)\right) \cdot 2 \cdot w_i \cdot p_m\Big),\end{aligned} \tag{4.27}$$

where

$$\epsilon_m = \hat{J}(p_m|\boldsymbol{\theta}) - o_m. \tag{4.28}$$

Therefore, the gradient of of $\mathcal{L}$ is

$$\nabla \mathcal{L}(\boldsymbol{\theta}) = \left[\tfrac{\partial \mathcal{L}}{\partial w_1}, \cdots, \tfrac{\partial \mathcal{L}}{\partial w_N}, \tfrac{\partial \mathcal{L}}{\partial \alpha_1}, \cdots, \tfrac{\partial \mathcal{L}}{\partial \alpha_N}, \tfrac{\partial \mathcal{L}}{\partial u_1}, \cdots, \tfrac{\partial \mathcal{L}}{\partial u_N}, \tfrac{\partial \mathcal{L}}{\partial \beta}\right]. \tag{4.29}$$

Summarizing above results, we provide Algorithm 4.2, which works as an inner loop of FMNN for training an MNN to fit data set $\mathcal{T}_k$.

Algorithm 4.2 Inner loop of FMNN: Fit input-output pattern

Require: Input-output pattern $\mathcal{T}_k$, initial search point $\boldsymbol{\theta}_k$
Ensure: Trained parameter $\boldsymbol{\theta}_{k+1}$
1: **procedure**
2: $\quad\boldsymbol{\theta}^{(0)} = \boldsymbol{\theta}_k$
3: $\quad$**for** l from 0 to $L-1$ **do**
4: $\quad\quad$Compute $\nabla \mathcal{L}(\boldsymbol{\theta}^{(l)})$ with $\mathcal{T}_k$ via (4.24)-(4.27)
5: $\quad\quad$Obtain $\boldsymbol{\theta}^{(l+1)}$ with $\boldsymbol{\theta}^{(l)}$ and $\nabla \mathcal{L}(\boldsymbol{\theta}^{(l)})$ via (4.23)
6: $\quad$**end for**
7: $\quad\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}^{(L)}$
8: **end procedure**

4.4.3 Apply learned MNN for transmission control

With the generated parameter $\boldsymbol{\theta}_K$, $\hat{\pi}(\cdot|\boldsymbol{\theta}_K)$ is constructed from (4.16) by setting $\boldsymbol{\theta}^* = \boldsymbol{\theta}_K$. For large N, M and K, $\hat{\pi}(s|\boldsymbol{\theta}_K)$ (4.16) should be close to $\pi^*(s)$ [98], and can be applied for selective transmission control, which is presented in Algorithm 4.3.

Algorithm 4.3 Transmission control with learned MNN

Require: Learned MNN $\hat{J}(\cdot|\boldsymbol{\theta}_K)$

procedure
  for t from 0 to ∞ **do**
    A packet arrives, and the node ends the silent period of cycle t
    Decode the packet and evaluate its priority d_t
    Probe CSI and estimate required transmit energy h_t
    Determine current remaining energy in battery b_t
    Construct state $s_t = [b_t, h_t, d_t]$
    Compute after-states $p_0 = \varrho(s_t, 0)$ and $p_1 = \varrho(s_t, 1)$
    Evaluate after-state values with trained MNN as $J_0 = \hat{J}(p_0|\boldsymbol{\theta}_K)$ and $J_1 = \hat{J}(p_1|\boldsymbol{\theta}_K)$
    if $J_0 > J_1 + \mathbb{1}(b_t \geq h_t) \cdot d_t$ **then** ▷ see (4.16)
      Discard the packet
    else
      Send the packet with energy h_t
    end if
    Battery is replenished with harvested energy e_t
    Enter silent period of cycle $t + 1$
  end for
end procedure

4.5 Numerical Simulation

We will next numerically study the learning characteristics of the proposed FMNN and the performance of the learned policy. In Section 4.5.2, we investigate the learning efficiency of FMNN. Section 4.5.3 demonstrates the performance of the learned policy.

4.5.1 Simulation setup

We model the wireless channels as Rayleigh fading, which is the most common model in wireless research. It is especially accurate for signal propagation in heavily built-up urban environments. The channel power gain z_t then has the PDF $f_Z(x) = \frac{1}{\mu_Z} e^{-x/\mu_Z}$, $x \geq 0$, where μ_Z presents the mean of z_t.

We assume that energy is harvested from wind power, which is well characterized by the Weibull distribution [80]. Hence, we model e_t with Weibull PDF $f_E(x) = \frac{k_E}{\lambda_E} \left(\frac{x}{\lambda_E}\right)^{k_E-1} e^{-\left(\frac{x}{\lambda_E}\right)^{k_E-1}}$, $x \geq 0$, with shape parameter $k_E = 1.2$ and scale parameter $\lambda_E = 0.15/\Gamma(1 + 1/k_E)$, where $\Gamma(\cdot)$ denotes the gamma function and μ_E represents the mean of e_t.

Moreover, the total energy consumption c_t during the silent period, data reception,

and channel estimation is modeled as a Gamma PDF $f_C(x) = \left(\Gamma(k_C)\theta_C^{k_C}\right)^{-1} x^{k_C-1}e^{-\frac{x}{\theta_C}},\ x > 0$, with shape parameter $k_C = 10$, and scale parameter $\theta_C = 0.02/k_C$. The combination of shape and scale parameters implies that the mean of c_t equals 0.02.

Furthermore, the model of data priority d_t depends on the specific practical application. We assume that d_t is exponentially distributed, i.e., $f_D(d) = e^{-d}, \quad d \geq 0$, which is also considered in [85,90]. This assumption is sensible, since for many applications, such as system monitoring, where high-priority packets that indicate critical events should happen with small probability, while the most of packets should have low priorities.

Finally, the number of hidden nodes N of the MNN is set to 3. The FMNN is executed with the data sample size of $|\mathcal{F}| = M = 500$, and the number of iterations $K = 20$.

4.5.2 Sample efficiency for learning π^*

To evaluate learning efficiency of FMNN, we use the *sample efficiency*, which evaluates the amount of data samples needed to be processed before an algorithm can learn a (near) optimal policy. Sample efficiency is a good proxy of an algorithm's training ability and adaptivity. We next assess the sample efficiency of FMNN and that of two alternatives, namely FNN (fitted value iteration with classical neural network) and Online-DIS (online learning with after-state space discretization), which are constructed as follows.

4.5.2.1 FNN and Online-DIS

FNN is the same as FMNN except replacing the MNN as a three-layer classical ANN (without non-negative weights constraint) and modifying the gradient descent method for ANN in Algorithm 4.2. Thus, FNN does not exploit the monotonicity of J^*, and the learning efficiency is expected to be inferior to that of FMNN.

Online-DIS algorithm is developed via applying the well-known Q-learning algorithm [50, Chapter 6.5] into our problem (Q-learning is also chosen for comparison in [90]). Online-DIS applies discretization for parameterization and an online learning scheme for learning associated parameters. Specifically, Online-DIS discretizes the after-state space into $\bar{N}$ bins, which are, respectively, associated with $\bar{N}$ param-

eters. The nth parameter presents the "aggregated" function values of $J^*(p)$ for all after-states p that fall into the nth bin. These parameters are learned via continuously updating parameters with each available sample. In order to properly average out data samples' randomness, the updating step size needs to be sufficiently small (see [50, p. 39]). Therefore, the learning generally progresses fairly slowly, and requires a large amount of data samples.

4.5.2.2 Sample efficiency comparison

The setting of FNN is the same as that of FMNN. For Online-DIS, the number of bins $\bar{N}$ is set as 20. To investigate the learning efficiency, we evaluate the performance of learned policies after consuming certain amount of data samples. The results are shown in Fig. 4.5 with logarithmic scale. Note that each iteration of FMNN and FNN consumes 500 data samples, and Fig. 4.5 shows the learning progress for the first 20 iterations.

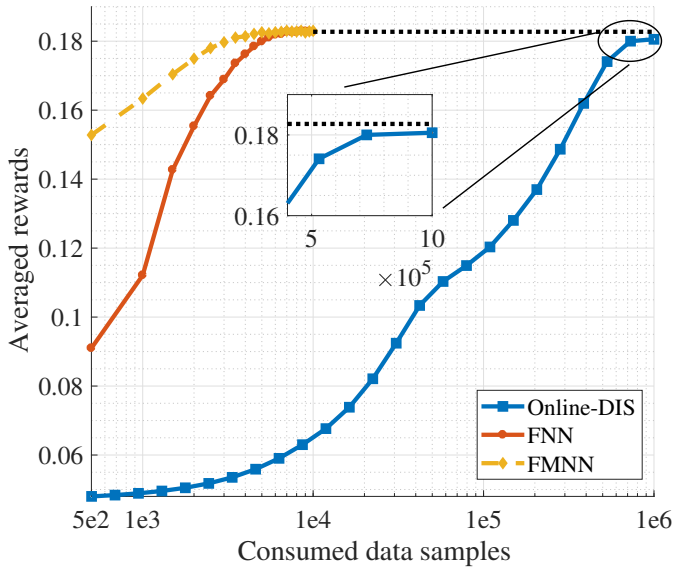


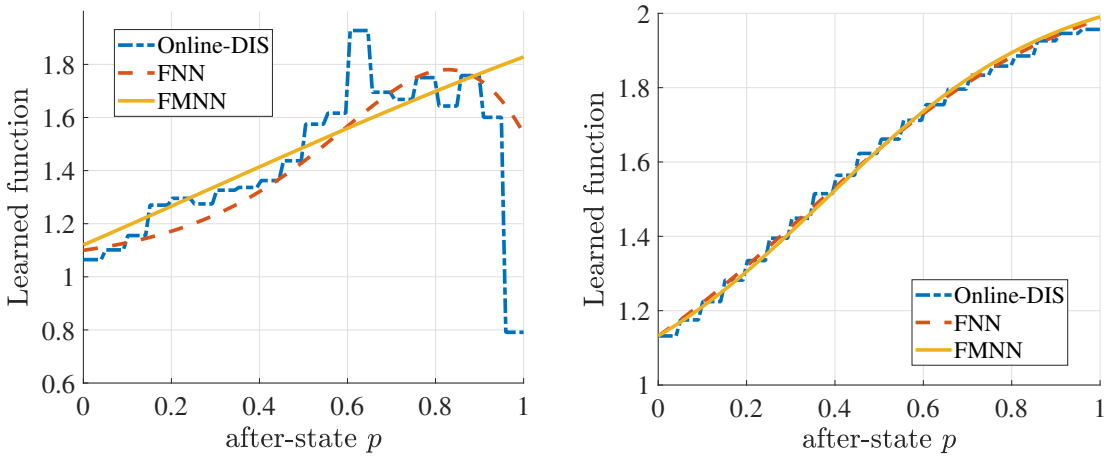
Figure 4.5: Learning curve

Firstly, we observe that FMNN and FNN are about 100 times more efficient than Online-DIS. This significant disparity occurs because FMNN and FNN directly train an MNN/ANN to fit data samples with regression, whereas Online-DIS must gradually average out randomness with a small step size.

Secondly, FMNN learns considerably faster than FNN. Because FMNN exploits

the non-decreasing property of J^*, it can learn a reasonably good policy with fewer iterations.

Finally, after processing enough data samples, all three learning curves converge. Both FMNN and FNN converge to the same value, while Online-DIS converges to a slightly inferior level. The reason is that both FMNN and FNN are able to represent a continuous non-decreasing function. Therefore, their learned policies achieve the same performance. However, the function represented by Online-DIS is piece-wise constant (see Fig. 4.6 below), and the discontinuity causes certain performance loss.



(a) FMNN and FNN after 1.5×10^3 samples, and Online-DIS after 1.5×10^5 samples

(b) FMNN and FNN after 10^4 samples, and Online-DIS after 10^6 samples

Figure 4.6: Learned value functions

The above analysis is confirmed in Fig. 4.6. Fig. 4.6(a) shows that, after processing 1.5×10^3 samples (3 iterations), FMNN learns a good value function, which is fairly close to the function learned after consuming 10^4 samples (20 iterations) as shown in Fig. 4.6(b). The function learned by FNN after processing 1.5×10^3 samples does not capture the non-decreasing structure of J^*. This fact suggests FNN's inferior sample efficiency compared with FMNN. Nevertheless, its ultimate learned function converges to that of FMNN, which provides the same limiting performance as achieved by FMNN. Finally, as one can notice, with 1.5×10^5 data samples, the function learned by Online-DIS is fluctuated, since the randomness of data samples has not be averaged out. Given 10^6 data samples and gradually decreasing step size, Online-DIS properly averages out noise and learns a non-decreasing function to fit J^*. However, due to the nature of after-state space discretization, the learned function is piece-wisely constant,

whose discontinuity causes slight performance loss of the resulted policy.

4.5.3 Achieved performance of learned policy

Given FMNN's learned parameter $\boldsymbol{\theta}_K$, the policy $\hat{\pi}(s|\boldsymbol{\theta}_K)$ defined with (4.16) can then be applied for selective transmission control via Algorithm 4.3. The major difference between the developed policy $\hat{\pi}(s|\boldsymbol{\theta}_K)$ and polices proposed in existing works [85–90] is that: polices of [85–90] make decisions based on available energy b_t and packet priority d_t, whereas $\hat{\pi}(s|\boldsymbol{\theta}_K)$ further exploits CSI h_t. To investigate the performance gain of exploiting CSI, we compare $\hat{\pi}(s|\boldsymbol{\theta}_K)$, named as DecBDH (i.e., decision based on b_t, d_t, and h_t), with the policy of work [90], named as DecBD (i.e., decision based on b_t and d_t). Note that works [85–89] do not fit energy-harvesting and/or wireless fading settings. DecBD works as follows. It first follows the scheme in [90] to decide whether or not to send based on available energy b_t and packet priority d_t. When "to send" is chosen, the node transmits with energy h_t if $b_t \geq h_t$, or drops the packet without energy consumption otherwise.

We also compare with an adaptive transmission (AdaTx) scheme, which always tries to send if a successful transmission can be achieved, i.e., the node transmits with energy h_t if $b_t \geq h_t$, or drops the packet without energy consumption otherwise.

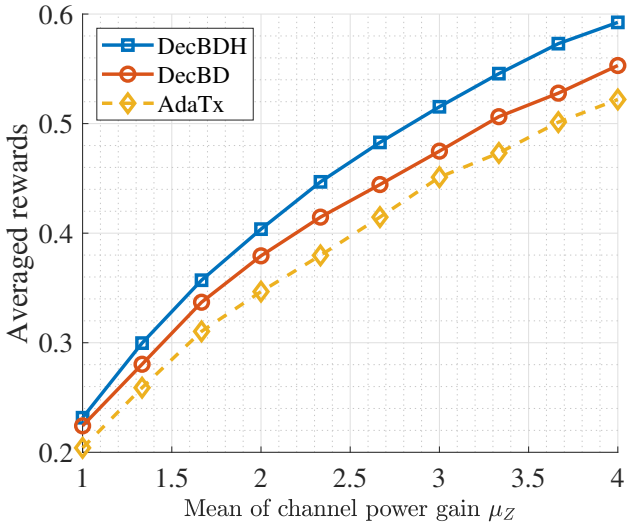


Figure 4.7: Achieved performance under different channel conditions

With $\mu_E = 0.15$, Fig. 4.7 shows the performance of DecBDH, DecBD and AdaTx under different channel conditions. As one can see, DecBD outperforms AdaTx. The

reason is that, via jointly considering d_t and b_t, DecBD is able to put more attention on high priority packets and avoid transmission when available energy level is low. Exploiting instantaneous CSI, DecBDH makes transmission decisions based on the channel status as well as the available energy and data priority, which enables it to obtain more transmission opportunities at good channel conditions, and therefore, achieve higher performance than DecBD.

4.6 Summary

In this chapter, we investigated the transmission policy of an energy-harvesting sensor, where low-priority packets can be dropped for transmitting more important packets over in a long term. Based on after-state value function, we constructed the optimal selective transmission policy, which decides whether or not to send each packet based on the sensor's battery level, the packet's priority and channel status. Then, the policy is further proved to be threshold-based with respect to data priority and channel status. Finally, exploiting the discovered structure, we proposed to learn the optimal policy with monotone neural network, which demonstrates high sample efficiency.

4.7 Appendix

4.7.1 Proof of Lemma 4.1

Since $b_{t+1} = ((p_t + e_t)^- - c_{t+1})^+$, we have, for $x \in [0, 1]$,

$$\begin{aligned}
\text{Prob}(b_{t+1} > x | p_t = p) &= \text{Prob}(((p + e_t)^- - c_{t+1})^+ > x) \\
&= \text{Prob}((p + e_t)^- > x + c_{t+1}) \\
&= \text{Prob}(1 > x + c_{t+1} \text{ and } p + e_t > x + c_{t+1}) \\
&= \int_0^{1-x} \int_{x-p}^{1} f_E(e) \cdot f_C(c)\, de\, dc, \qquad (4.30)
\end{aligned}$$

where Prob$(\cdot)$ means probability, and the last equality holds as e_t has PDF f_E and c_{t+1} has PDF f_C.

It is easy to check that Prob$(b_{t+1} > x | p_t = p)$ does not depend on t, and

$$F_B(b|p) = 1 - \text{Prob}(b_{t+1} > b | p_t = p). \qquad (4.31)$$

In addition, since f_E and f_C are both non-negative, $\text{Prob}(b_{t+1} > b|p_t = p)$ (4.30) is non-decreasing with respect to p, which implies $F_B(t|p)$ is non-increasing with respect to p from (4.31). Finally, it is obvious that $\frac{d}{dp}F_B(t|p)$ and $\frac{d}{dt}F_B(t|p)$ exist (and can be computed via the Leibniz integral rule), which completes the proof.

4.7.2 Proof of Theorem 4.1

With value iteration algorithm (4.13), we prove Theorem 4.1 by induction. Specifically, we set $J_0(p) = 0$ for all $p \in [0, 1]$, which is differential and non-decreasing. With this choice of J_0, we can prove Theorem 4.1 by showing that, if J_k is non-decreasing and differentiable, J_{k+1} is non-decreasing and differentiable.

First, from (4.13), we have

$$J_{k+1}(p) = \gamma \mathbb{E}\big[r(s', A') + J_k(\varrho(s', A'))\big|p\big], \tag{4.32}$$

where the expectation is taken over both s' and A', and A' is a function of random variable s':

$$A' = \begin{cases} 1 & \text{if } r(s', 1) + J_k(\varrho(s', 1)) \geq J_k(\varrho(s', 0)) \\ 0 & \text{otherwise.} \end{cases}$$

In addition, due to the induction assumption and the fact $\varrho(s', 1) \leq \varrho(s', 0)$, we have $J_k(\varrho(s', 1)) \leq J_k(\varrho(s', 0))$. Therefore, if $A' = 1$, we must have $b' \geq h'$, since $b' < h'$ implies $r(s', 1) = 0$, and further implies $r(s', 1) + J_k(\varrho(s', 1)) < J_k(\varrho(s', 0))$, and therefore, $A' = 0$ (a contradiction of $A' = 1$).

Denoting $S_i(b)$ as the region of (h, d) where action $A' = i \in \{0, 1\}$ given b, we have $S_1(b) = \{(h, d)|b \geq h, d + J_k(b-h) > J_k(b)\}$, and $S_0(b) = \{(h, d)|b \geq h, d + J_k(b-h) \leq J_k(b)\} \cup \{(h, d)|b < h\}$. Given $S_0(b)$ and $S_1(b)$, $J_{k+1}(p)$ (4.32) can be rewritten as

$$J_{k+1}(p) = \gamma \int_0^1 f_B(\eta|p) \cdot M(\eta)\, d\eta, \tag{4.33}$$

where

$$\begin{aligned} M(\eta) \triangleq & \iint_{(x,y)\in S_1(\eta)} \big[y + J_k(\eta - x)\big] \cdot f_D(y) \cdot f_H(x)\, dy\, dx \\ & + \iint_{(x,y)\in S_0(\eta)} J_k(\eta) \cdot f_D(y) \cdot f_H(x)\, dy\, dx \\ = & \int_0^{\eta} \int_{J_k(\eta-x)}^{+\infty} \big(y + J_k(\eta - x)\big) \cdot f_H(x) \cdot f_D(y)\, dy\, dx \end{aligned}$$

$$+\int_0^{\eta}\int_0^{J_k(\eta-x)} J_k(\eta)\cdot f_H(x)\cdot f_D(y)\,dy\,dx$$
$$+\int_t^{+\infty}\int_0^{+\infty} J_k(\eta)\cdot f_H(x)\cdot f_D(y)\,dy\,dx. \tag{4.34}$$

In the following, we present Lemma 4.2, whose proof is given in Section 4.7.3. With Lemma 4.2 and (4.33), we thereby can complete the proof by showing that $M(\eta)$ is differentiable and non-decreasing.

Lemma 4.2. *Given that function $\Omega(\eta)$ is non-decreasing and differentiable for $\eta \in [0,1]$, function $J(p)$ generated with $J(p)=\int_{\eta=0}^{1} f_B(\eta|p)\Omega(\eta)\,d\eta$ is non-decreasing and differentiable for $p\in[0,1]$.*

With the assumption that $J_k(p)$ is differentiable, the function $M(\eta)$ can be shown to be differentiable by repeatedly using the Leibniz's rule. Therefore, we are remaining to show that $M(\eta)$ is non-decreasing.

Given any $\eta_1,\eta_2\in[0,1]$ and $\eta_1\geq\eta_2$, we define $S_{ij}\triangleq S_i(\eta_1)\cap S_j(\eta_2)$ with $i,j\in\{0,1\}$. Then, from (4.34), it is easy to verify that

$$\begin{aligned}
M(\eta_1)-M(\eta_2) = &\underbrace{\mathbb{E}\left[\mathbb{1}((h,d)\in S_{11})\cdot\big(J_k(\eta_1-h)-J_k(\eta_2-h)\big)\right]}_{\triangleq\Delta_{11}}\\
&+\underbrace{\mathbb{E}\left[\mathbb{1}((h,d)\in S_{10})\cdot\big(d+J_k(\eta_1-h)-J_k(\eta_2)\big)\right]}_{\triangleq\Delta_{10}}\\
&+\underbrace{\mathbb{E}\left[\mathbb{1}((h,d)\in S_{01})\cdot\big(J_k(\eta_1)-d-J_k(\eta_2-h)\big)\right]}_{\triangleq\Delta_{01}}\\
&+\underbrace{\mathbb{E}\left[\mathbb{1}((h,d)\in S_{00})\cdot\big(J_k(\eta_1)-J_k(\eta_2)\big)\right]}_{\triangleq\Delta_{00}},
\end{aligned}$$

where the expectations are taken over h and d. With the assumption that $J_k(p)$ is non-decreasing, the following results hold. Since $\eta_1\geq\eta_2$, we have $J_k(\eta_1-H)\geq J_k(\eta_2-H)$ and $J_k(\eta_1)\geq J_k(\eta_2)$, and therefore, $\Delta_{11}\geq 0$ and $\Delta_{00}\geq 0$. And according to the definition of S_{10}, we have $d_t+J_k(\eta_1-H)\geq J_k(\eta_1)\geq J_k(\eta_2)$, which means $\Delta_{10}\geq 0$. Finally, for S_{01}, we have $J_k(\eta_1)\geq d_t+J_k(\eta_1-H)\geq d_t+J_k(\eta_2-H)$, which means $\Delta_{01}\geq 0$. In summary, we have $M(\eta_1)-M(\eta_2)\geq 0$ given $\eta_1\geq\eta_2$. Therefore, $M(\eta)$ is non-decreasing.

4.7.3 Proof of Lemma 4.2

Suppose the derivative of $\Omega(\eta)$ in $\eta \in [0,1]$ is $\omega(\eta)$. Since $\Omega(\eta)$ is non-decreasing, we have $\omega(\eta) \geq 0$. Then, using integration by parts, we have

$$J(p) = \Omega(1) - \Omega(0)F_B(0|p) - \int_0^1 \omega(\eta) \cdot F_B(\eta|p)\, d\eta, \tag{4.35}$$

where the fact $F_B(1|p) = 1$ is exploited. From Lemma 4.1, $F_B(\eta|p)$ is non-increasing respected to p for any η. This means that $-\Omega(0)F_B(0|p)$ is non-decreasing. Furthermore, because $\omega(\eta) \geq 0$, we have $-\int_0^1 \omega(\eta) \cdot F_B(\eta|p)\, d\eta$ is non-decreasing (respected to p). In summary, $J(p)$ is non-decreasing.

In addition, since $F_B(\eta|p)$ is differentiable over p, it is easy to see from (4.35) that $J(p)$ is differentiable over p.

Chapter 5

Cooperative Spectrum Sensing under Heterogeneous Spectrum Availability

5.1 Introduction

Sensing spectrum for correctly detecting PUs is crucial for cognitive radio systems, since it maximizes transmission opportunities, and also limits SUs' interference to PUs. However, due to channel propagation impairments, especially wireless fading, it is difficult for an SU to achieve reliable detection via independently sensing spectrum. Fortunately, the sensing reliability can be improved by fusing spectrum measurements from multiple, spatially-separate secondary nodes, known as cooperative spectrum sensing (CSS).

Most of existing works on CSS (see [23] and references therein) assumed a spectrum homogeneity setting, i.e., all cooperating SUs observe the same PU and experience the same spectrum availability status. This assumption is reasonable for PUs with large transmission power, such as television broadcast systems [15], and/or small-scale secondary networks where all SUs are co-located. However, in other cases (e.g., secondary networks with large geographical expansion), some SUs may be too close to PUs, and therefore, experience the busy spectrum status (i.e., do not allowed to transmit), while the other SUs are far away from PUs, and therefore, experience the free spectrum status. That is, SUs at different spatial locations may experience different spectrum availability statuses, i.e., a heterogeneous spectrum availability setting.

Considering sensing tasks in presence of heterogeneous spectrum availability, we take the view point of distance. That is, an SU is allowed to transmit, if it is certain distance far away from active PUs [99]. Such a distance, named as protection range (PrR), can be rigorously computed via considering PU interference tolerance, PU and SU transmission power and others [100, 101]. Therefore, for an SU, the goal of spectrum sensing is to decide whether it is within the PrR of any active PU.

Despite heterogeneous spectrum availability, sensing cooperation is still beneficial. It is because spatially proximate SUs are likely to fall within the PrR of the same PU, and therefore share the same spectrum status. Hence, spatial information is a key for applying CSS. We do not exploit SU location information, as it can be hard to obtain. Instead, SU neighbor information is assumed: we assume that each SU knows its neighboring SUs, i.e., SUs within the SU's transmission range. Therefore, ***the problem is, under heterogeneous spectrum availability, how to exploit neighbor information to fuse SU observations for improving sensing performance.***

As the easiest way, work [102] showed that, in presence of heterogeneous spectrum availability, an SU can improve sensing performance via combining individual hard decisions of its neighbors. However, since strong spectrum correlation may exist between SUs over several hops, it is important to consider sensing cooperation beyond one hop.

To do so, works [103–106] applied (pairwise) **Markov random fields (MRFs)** [107] (also see Chapter 2.3) to model the prior distribution of SUs' spectrum statuses. MRFs exploit a Markovian approximation: the spectrum status of an SU is independent of those of non-adjacent SUs, given its adjacent SUs' spectrum statuses. MRFs heuristically build up the prior distribution via capturing the strongest spectrum correlations among SUs.

With prior distribution, works [103–106] attempted to infer spectrum statuses via marginalization. Specifically, combining MRF prior with sensing observations (which specify data likelihood functions), a posterior distribution over all SUs' spectrum statuses can be obtained. Then, the sensing decision of an SU is made based on the marginal distribution of the SU's spectrum status, which can be computed from the posterior distribution via summing over the spectrum statuses of all other SUs. Due to

the curse of dimensionality, exactly computing the marginal distribution is impractical in large-scale secondary networks. Fortunately, thanks to MRFs' simple structure, an approximated version of the marginal distribution can be computed via belief propagation (BP) algorithms [108, Chapter 11], where SUs distributedly estimate their corresponding marginal distributions via iteratively exchanging messages with their neighbors.

5.1.1 Motivations

Approximately computing marginal distributions via BP algorithms, as considered in [103–106], provides a useful methodology to fuse SUs' sensing observations for decision making. However, there are two theoretical and one practical downsides for this methodology.

Theoretically, BP algorithms do not have convergence guarantee. In addition, when convergence is reached, the computed distribution does not equal the true marginal distribution [108, p. 392]. Practically and more importantly, BP algorithms best fit distributed networks with low SU density, but may not work well in other networks. The reason is that BP algorithms' major computation advantage comes from parallelization. This benefit vanishes in centralized secondary networks, where sensing information and decisions need to be processed in central infrastructures. Moreover, the computation complexity for an SU to execute BP algorithms is exponential in the number of its neighbors. Hence, it is computationally demanding to apply BP algorithms in secondary networks with high density.

5.1.2 Contributions

To address the above limitations, we propose a different methodology. Specifically, we still use MRFs for modeling the prior distribution, but we infer spectrum statuses via solving the maximum a posterior (MAP) problem (instead of marginal distributions). Roughly speaking, sensing decisions are jointly made by solving the configuration of SU spectrum statuses that maximizes the posterior distribution, which is named an **MAP-MRF** problem.

Note that, similar to marginalization, MAP-MRF is a meaningful formulation for inferring spectrum statuses from sensing observations. More importantly, thanks to

optimization theory, the MAP-MRF problem can be solved more efficiently and flexibly than marginalization. Based on MAP-MRF, we propose three CSS algorithms, which are, respectively, designed for centralized, cluster-based and fully distributed secondary networks.

For centralized secondary networks, we propose a CSS algorithm based on **graph cut** theory, named as GC-CSS. GC-CSS exactly solves the MAP-MRF problem with polynomial time complexity in both network size and density.

For cluster-based secondary networks (where sensing information and decision of each cluster are processed centrally), we develop a CSS algorithm based on **dual decomposition** theory, named as DD-CSS. DD-CSS is able to obtain sensing decisions distributedly (at cluster-level) via iteratively exchanging (binary) messages among clusters.

For distributed secondary networks, we obtain a CSS algorithm, named as DD1-CSS, which is a special case of DD-CSS, when each SU forms a cluster. DD1-CSS works similar to BP algorithms via exchanging messages among SUs. However, compared with BP algorithms, DD1-CSS has following advantages. First, DD1-CSS has theoretical guarantee (for solving the MAP-MRF problem). In addition, each SU's computation complexity is polynomial in the number of its neighbors. Finally, its communication overhead is low.

The rest of this chapter is organized as follows. In Section 5.2, the system model is presented. The CSS problem based on MAP-MRF is formulated in Section 5.3. GC-CSS, DD-CSS, DD1-CSS algorithms are proposed in Sections 5.4, 5.5 and 5.6, respectively. Section 5.7 demonstrates several numerical simulation results.

Notation conventions: Most of the notations in the rest of this Chapter follow the conventions below (exceptions can be easily understood from context): 1) a lowercase letter x denotes a scale variable or a random variable's realization; 2) a capitalized letter X denotes a random variable; 3) a hatted variable $\hat{x}$ denotes an estimation of x; 4) a bold variable $\boldsymbol{x}$ denotes a vector; 5) a calligraphy letter $\mathcal{X}$ denotes a set.

5.2 System Model

5.2.1 Network setup

We consider a disc with radius R (see Fig. 5.1), which represents the area of interest. There are N SUs randomly and uniformly located within the disc, and these SUs do not move. It is assumed that SUs are unaware of their own locations. The N SUs are indexed as SU_1, ..., SU_N.

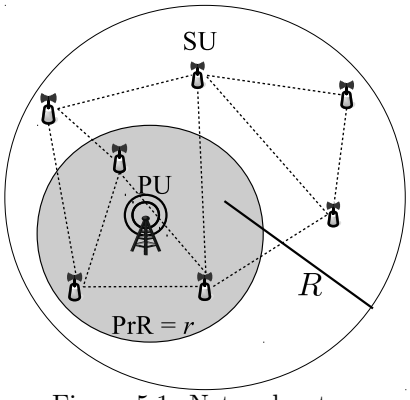


Figure 5.1: Network setup

At each sensing period, a PU appears randomly within the disc, and transmits signal. Let r denote PrR and d_i denote the distance between SU_i and the PU. Therefore, SU_i's spectrum status is "busy", denoted as $x_i = 1$, if $0 \leq d_i \leq r$; and SU_i's spectrum status is "free", denoted as $x_i = 0$, if $r < d_i \leq R$. It is easy to see that SUs at different locations may have different spectrum statuses, a heterogeneous spectrum availability setting.

5.2.2 Signal model and data likelihood functions

SU_i measures the energy of received primary signal, denoted as y_i, in order to infer its spectrum status x_i. Intuitively, if the measured energy y_i is small, SU_i is likely to be far away from the PU, and therefore, x_i is likely to be 0, and vice versa. In following, we mathematically model the relationship between (observed) energy measurement y_i and (unobserved) spectrum status X_i.

Assume that the primary signal received by SUs experiences static path loss and Rician fading. Therefore, the complex channel gain between the PU and SU_i, denoted

as H_i, is modeled as

$$H_i = (1 + D_i)^{-\alpha/2}\Psi_i,$$

where $\alpha > 0$ is the propagation factor, and Ψ_i denotes the complex gain of a Rician fading channel. Note that, the distance D_i is treated as a random variable, since the locations of SUs and the PU are unknown. We assume that $\{\Psi_i\}_i$ are independent and identically distributed, and do not change during a sensing period, i.e., a block fading assumption. Therefore, given M signal samples, the measured energy Y_i (before observation) can be expressed as

$$Y_i = \sum_{m=1}^{M} |H_i S(m) + W(m)|^2,$$

where $S(m)$, representing the mth sample of primary signal, is modeled as a random variable that obeys $\mathcal{CN}(0, 2\sigma_S^2)$ (zero-mean complex normal distribution with variance $2\sigma_S^2$), and $W(m)$, representing the mth sample of background noise, is modeled as a random variable that obeys $\mathcal{CN}(0, 2\sigma_W^2)$.

It is known that $|\Psi_i|^2$ follows a noncentral chi-square distribution. Without loss of generality, the PDF of $|\Psi_i|^2$ can be expressed as

$$\begin{aligned} f_{|\Psi|^2}(|\psi_i|^2) = {} & 2(k_H + 1)\,|\psi_i|^2 \exp\left(-(k_H + 1)|\psi_i|^4 - k_H\right) \\ & \times I_0\left(2\sqrt{k_H(k_H + 1)|\psi_i|^2}\right), \end{aligned} \tag{5.1}$$

where k_H (known as the K-factor) accounts for the power ratio between the line-of-light path and scattered paths, and $I_0(\cdot)$ is the zeroth order modified bessel function of the first kind. Note that, $f_{|\Psi|^2}(\cdot)$ does not depend on i, since $\{\Psi_i\}_i$ are assumed to be independent and identically distributed.

In addition, from [54], the PDF of Y_i given D_i and $|\Psi_i|^2$ is

$$f_{Y|\,D,|\Psi|^2}(y_i \mid d_i, |\psi_i|^2) = \frac{y_i^{M-1}\exp(-\frac{y_i}{2(1+\xi_i)\sigma_W^2})}{(2(1 + \xi_i)\sigma_W^2)^M \Gamma(M)} \tag{5.2}$$

where $\Gamma(\cdot)$ is the gamma function and $\xi_i = (1 + d_i)^{-\alpha}|\psi_i|^2\sigma_S^2/\sigma_W^2$. Note that, $f_{Y|\,D,|\Psi|^2}(\cdot|\cdot,\cdot)$ does not depend on i, since, given $D_i = d_i$ and $|\Psi_i|^2 = |\psi_i|^2$, Y_i's randomness only stems from the primary signal and background noise.

Finally, we (approximately) model the PDF of D_i given X_i as

$$f_{D|X}(d_i \mid x_i = 0) = \begin{cases} 2d_i/(R^2 - r^2) & \text{if } d_i \in (r, R], \\ 0 & \text{otherwise,} \end{cases} \tag{5.3a}$$

$$f_{D|X}(d_i \mid x_i = 1) = \begin{cases} 2d_i/r^2 & \text{if } d_i \in [0, r], \\ 0 & \text{otherwise.} \end{cases} \tag{5.3b}$$

Note that $f_{D|X}(d_i \mid x_i)$ exactly matches the network setup of Section 5.2.1 only if SU_i is located at the center of the disc. However, since SUs are unaware of their locations, it is reasonable to apply $f_{D|X}(\cdot \mid \cdot)$ to SU_i for all i.

Based on (5.1), (5.2), (5.3), we can obtain the PDF of Y_i given X_i as

$$f_{Y|X}(y_i \mid x_i) = \int_{|\psi_i|^2=0}^{\infty} \int_{d_i=0}^{R} f_{Y|\,D,|\Psi|^2}\left(y_i \mid d_i, |\psi_i|^2\right) f_{D|X}\left(d_i \mid x_i\right) f_{|\Psi|^2}\left(|\psi_i|^2\right) \,\mathrm{d}d_i \,\mathrm{d}|\psi_i|^2. \tag{5.4}$$

Given observed energy measurement y_i, $f_{Y\,|\,X}(y_i \mid x_i)$, known as data likelihood function, represents the likelihood that $X_i = x_i$. Therefore, SU_i can apply individual spectrum sensing (Ind-SS) to infer its spectrum status. That is, SU_i makes sensing decision as

$$\hat{x}_i = \begin{cases} 1 & \text{if } \gamma\, f_{Y|X}(y_i \mid 1) > f_{Y|X}(y_i \mid 0), \\ 0 & \text{otherwise,} \end{cases} \tag{5.5}$$

where weight $\gamma > 0$ is used to balance detection probability with false alarm probability. However, Ind-SS can be unreliable, especially when k_H is small (i.e., poor channel conditions). In Section 5.3, we exploit an MRF model to improve sensing reliability via fusing sensing data from other SUs.

Note: In this chapter, we consider a single PU case for simplicity. Our subsequently proposed MAP-MRF formulation (Section 5.3) and CSS algorithms (Sections 5.4-5.6) apply to more general settings, as long as a data likelihood function similar to (5.4) can be obtained for corresponding environments.

5.3 CSS with MAP-MRF Formulation

Despite the heterogeneous spectrum availability setting (section 5.2.1), proximately located SUs are likely to experience the same spectrum status. Therefore, we would like to exploit this intuition to improve sensing performance via properly fusing sensing data. For this purpose, we next construct an MRF, and then present the MAP-MRF methodology for CSS.

5.3.1 Define SU-graph and MRF

It is assumed that, upon network's setup, each SU performs neighbor discovering, from which the SU knows its neighbors (SUs within its transmission distance). With this neighbor information, we define an SU-graph, which is an undirected graph, denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $\mathcal{V}$ and $\mathcal{E}$ representing the set of nodes and edges, respectively. For the node set, we have $\mathcal{V} = \{1, \cdots, N\}$, where node i represents SU_i. The edge set $\mathcal{E}$ is defined as

$$\mathcal{E} = \{(i,j)|\ \text{SU}_i \text{ and } \text{SU}_j \text{ are neighbors}\}. \tag{5.6}$$

Note that, since edges of $\mathcal{G}$ are undirected, (i,j) and (j,i) are the same element of $\mathcal{E}$.

Here, we apply an MRF over SU-graph $\mathcal{G}$ to model the prior distribution of $\boldsymbol{X} \triangleq [X_1, \cdots, X_N]$. MRF exploits an Markovian approximation: the spectrum status of an SU is independent of those of non-adjacent SUs, if its adjacent SUs' spectrum statuses are known. Specifically, the prior belief (before observing sensing data) that $\boldsymbol{X} = \boldsymbol{x} \triangleq [x_1, \cdots, x_N]$ is modeled as

$$\Phi_{\boldsymbol{X}}(\boldsymbol{x}) = \frac{1}{B} \prod_{(i,j)\in\mathcal{E}} \phi(x_i, x_j), \tag{5.7}$$

where B is a constant for normalization, $\mathcal{E}$ is defined in (5.6), and $\phi(x_i, x_j)$, named as the potential function, is defined as

$$\phi(x_i, x_j) = \exp\left(\beta\left(x_i x_j + (1-x_i)(1-x_j)\right)\right), \tag{5.8}$$

where $\beta > 0$ is the hyperparameter of the MRF model. Note that, for the potential function, we have $\phi(0,0) = \phi(1,1) = \exp(\beta) > \phi(0,1) = \phi(1,0) = 1$, which represents the correlation between adjacent SUs, and the hyperparameter β controls the correlation strength. We assume that β has been chosen properly (see Section 5.7.2 for choosing β).

5.3.2 Fuse data over MRF

5.3.2.1 Existing scheme based on marginalization

Given the MRF prior and sensing data $\boldsymbol{y} \triangleq [y_1, \cdots, y_N]$, works [103–106] considered marginalization for CSS. Specifically, from (5.4) and (5.7), a posterior distribution of

$\boldsymbol{X}$ given $\boldsymbol{y}$ is first obtained

$$p_{\boldsymbol{X}}(\boldsymbol{X}=\boldsymbol{x}|\boldsymbol{y}) = \Phi_{\boldsymbol{X}}(\boldsymbol{x}) \prod_{i\in\mathcal{V}} f_{Y|X}(y_i \mid x_i). \tag{5.9}$$

Then, to decide the spectrum status of an SU, say SU_i, works [103–106] attempt to estimate the marginal (posterior) distribution of X_i

$$\begin{aligned} p_{X_i}(X_i = x_i \mid \boldsymbol{y}) &= \sum_{x_1} \cdots \sum_{x_{i-1}} \sum_{x_{i+1}} \cdots \sum_{x_N} p_{\boldsymbol{X}}\left([x_1, \cdots, x_{i-1}, x_i, x_{i+1}, \cdots, x_N]|\boldsymbol{y}\right) \\ &\triangleq \sum_{\boldsymbol{x}':\boldsymbol{x}\setminus\{x_i\}} p_{\boldsymbol{X}}(\boldsymbol{x}' \mid \boldsymbol{y}), \end{aligned} \tag{5.10}$$

where $\sum_{\boldsymbol{x}':\boldsymbol{x}\setminus\{x_i\}}[\cdot]$ means to "marginalize over all variables within $\boldsymbol{x}$ except x_i", which is introduced for notation convenience. Note that $p_{X_i}(x_i \mid \boldsymbol{y})$ represents the probability that $X_i = x_i$ by jointly considering the spatial correlation $\Phi_{\boldsymbol{X}}(\boldsymbol{x})$ and sensing data $\boldsymbol{y}$. Therefore, given an estimated $\hat{p}_{X_i}(x_i \mid \boldsymbol{y})$, SU_i decides $\hat{x}_i = 1$ if $\hat{p}_{X_i}(1 \mid \boldsymbol{y}) \geq \mathcal{T}$ ($\mathcal{T} \in (0,1)$ is a threshold), and decides $\hat{x}_i = 0$ otherwise. See Section 5.6.2.1 for how to compute $\hat{p}_{X_i}(x_i \mid \boldsymbol{y})$ via BP algorithms.

5.3.2.2 Proposed scheme based on maximization

Instead of following above methodology, we consider the (weighted) MAP estimation of $\boldsymbol{x}$. Specifically, given $\Phi_{\boldsymbol{X}}(\boldsymbol{x})$ and $\boldsymbol{y}$, we solve $\boldsymbol{x}^{\text{MAP}} = [x_1^{\text{MAP}}, \cdots, x_N^{\text{MAP}}]$ such that

$$\boldsymbol{x}^{\text{MAP}} = \arg\max_{\boldsymbol{x}} \left\{ \prod_{(i,j)\in\mathcal{E}} \phi(x_i, x_j) \prod_{i\in\mathcal{V}} \gamma^{x_i} f_{Y|X}(y_i|x_i) \right\}, \tag{5.11}$$

where weight $\gamma > 0$ (similar to $\mathcal{T}$) introduces trade off between detection probability and false alarm probability. It can be seen that $\boldsymbol{x}^{\text{MAP}}$ is the most probable configuration of $\boldsymbol{x}$ given spatial correlation $\Phi_{\boldsymbol{X}}(\boldsymbol{x})$ and sensing data $\boldsymbol{y}$. In other words, via deciding SUs' spectrum statuses $\hat{\boldsymbol{x}} \triangleq [\hat{x}_1, \cdots, \hat{x}_N] = \boldsymbol{x}^{\text{MAP}}$, we maximize the 'likelihood' of correctly determining SUs' spectrum statuses, based on MRF prior and sensing observations. Note that, when there is no spatial correlation among SUs, i.e., $\beta = 0$, the above decision making scheme reduces to Ind-SS (5.5).

Remark: Both (5.10) and (5.11) are meaningful sensing decision making schemes that effectively fuse data via the MRF model. It is not important to determine which one is a more theoretically proper formulation, since the MRF model is merely an

approximation of SUs' actually spatial correlation. The key point is that, (5.11) can be solved more efficiently and flexibly than (5.10), which will be discussed in Sections 5.4, 5.5 and 5.6.

5.3.2.3 Transform to posterior energy

For the convenience of CSS algorithms' development in subsequence sections, we define a function, called posterior energy, for any $\boldsymbol{x}$ as

$$\mathrm{E}(\boldsymbol{x}) = -\ln\left(\prod_{(i,j)\in\mathcal{E}}\phi(x_i,x_j)\prod_{i\in\mathcal{V}}\gamma^{x_i}f_{Y|X}(y_i|x_i)\right) = \sum_{(i,j)\in\mathcal{E}}\kappa(x_i,x_j)+\sum_{i\in\mathcal{V}}\eta_i(x_i), \tag{5.12}$$

where $\kappa(x_i,x_j)$, called the edge energy, is defined as

$$\kappa(x_i,x_j) = -\ln(\phi(x_i,x_j)) = -\beta\left(x_ix_j+(1-x_i)(1-x_j)\right) \tag{5.13}$$

and $\eta_i(x_i)$, called the unitary energy, is defined as

$$\eta_i(x_i) = -\ln(\gamma^{x_i}f_{Y|X}(y_i|x_i)). \tag{5.14}$$

It is easy to see that we have

$$\boldsymbol{x}^{\mathrm{MAP}} = \arg\min_{\boldsymbol{x}}\left\{\mathrm{E}(\boldsymbol{x})\right\}, \tag{5.15}$$

which is named as the MAP-MRF problem. In following sections, we will develop three CSS algorithms for (approximately) solving $\boldsymbol{x}^{\mathrm{MAP}}$.

5.4 GC-CSS for Centralized Secondary Networks

This section presents GC-CSS, which is designed for centralized secondary networks. Assuming SU neighbor information and sensing information (provided by a central infrastructure), the CSS decision $\boldsymbol{x}^{\mathrm{MAP}}$ is obtained by solving a so-called min-cut problem. Note that MAP-MRF problems are first formulated as min-cut problems in [109] for image segmentation task. Compared with [109], we provide a different formulation and proof.

5.4.1 BF-graph and min-cut

In this subsection, we define a BF-graph and its associated min-cut problem. BF-graph is an undirected graph with weighted edges, denoted as $\mathcal{G}^{BF} = (\mathcal{V}^{BF}, \mathcal{E}^{BF}, c(\cdot))$, with $\mathcal{V}^{BF}$, $\mathcal{E}^{BF}$, and $c(\cdot)$, respectively, represent node set, edge set and cost function associated with edges.

The node set and edge set are constructed from SU-graph $\mathcal{G}$ as follows. We define node set $\mathcal{V}^{BF} = \mathcal{V} \cup \{b, f\}$, where b and f are two nodes representing the busy and free spectrum statuses, respectively. See Fig. 5.3 for an example of BF-graph, which is constructed from the SU-graph of Fig. 5.2.

We define edge set $\mathcal{E}^{BF} = \mathcal{E} \cup \{(b,i)\}_{i\in\mathcal{V}} \cup \{(f,i)\}_{i\in\mathcal{V}}$. That is, $\mathcal{E}^{BF}$ consists of 3 types of edges: $\mathcal{E}$, $\{(b,i)\}_{i\in\mathcal{V}}$ and $\{(f,i)\}_{i\in\mathcal{V}}$ (which are respectively, represented as black, orange and blue lines in Fig. 5.3).

Edge cost function $c(\cdot)$ is specified for the 3 types of edges as follows. We assign cost for edge $(i,j) \in \mathcal{E}$ as

$$c(i,j) = \beta, \tag{5.16}$$

assign cost for edge (b,i), $\forall i$, as

$$c(b,i) = \eta_i(0) + M_i, \tag{5.17}$$

assign cost for edge (f,i), $\forall i$, as

$$c(f,i) = \eta_i(1) + M_i, \tag{5.18}$$

where $\eta_i(0)$ and $\eta_i(1)$ are defined in (5.14), and M_i is a constant ensuring both $c(b,i)$ and $c(f,i)$ are nonnegative. For example, we can define $M_i = \ln(\gamma f(y_i|1) + f(y_i|0))$.

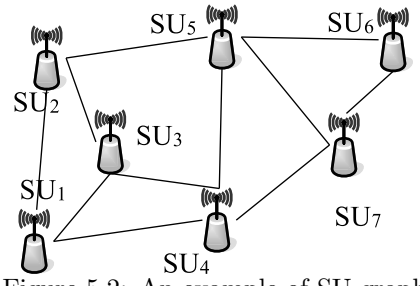


Figure 5.2: An example of SU-graph

A cut $K = (\mathcal{B}, \mathcal{F})$ of the BF-graph is a bipartition of $\mathcal{V}^{BF}$ such that $\mathcal{B} \cap \mathcal{F} = \emptyset$, $\mathcal{B} \cup \mathcal{F} = \mathcal{V}^{BF}$, $b \in \mathcal{B}$ and $f \in \mathcal{F}$. The cut-set of K is $\mathcal{E}^K = \{(i,j) \in \mathcal{E}^{BF} | i \in \mathcal{B}, j \in \mathcal{F}\}$.

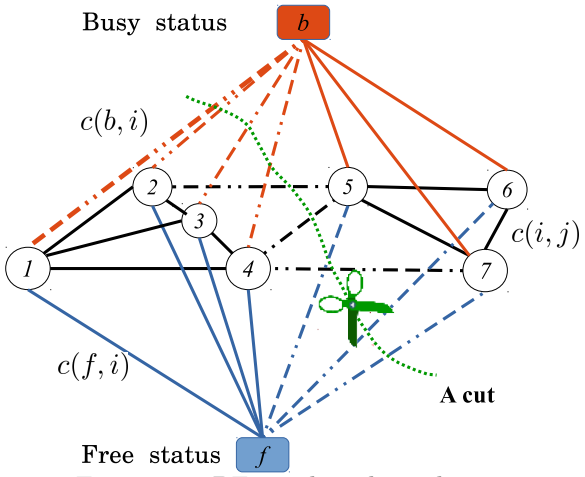


Figure 5.3: BF-graph and graph cuts

It can be seen that, after removing edges of $\mathcal{E}^K$, node b is separated from node f in graph $\mathcal{G}^{BF}$. The cost $c(K)$ of a cut K is defined as the summation of the cost of all edges within $\mathcal{E}^K$, i.e.,

$$c(K) = \sum_{e \in \mathcal{E}^K} c(e).$$

In the example of Fig. 5.3, $(\mathcal{B} = \{b, 5, 6, 7\}, \mathcal{F} = \{f, 1, 2, 3, 4\})$ defines a cut, whose cost is $\sum_{i=1}^{4} c(f, i) + \sum_{i=5}^{7} c(b, i) + 3\beta$.

Denote $\mathcal{K}$ as all possible cuts. The min-cut K^* is the cut that attains the minimum cost among $\mathcal{K}$, i.e.,

$$K^* = \arg\min_{K \in \mathcal{K}} \{c(K)\}. \tag{5.19}$$

Problem (5.19) can be solved efficiently, as stated in Theorem. 5.1, which is proved in [110, Chapter 26].

Theorem 5.1. *For an SU-graph with N nodes and $|\mathcal{E}|$ edges, problem* (5.19) *can be solved (via Ford-Fulkerson algorithm [111]) with time complexity of $\mathcal{O}(N \cdot |\mathcal{E}|^2)$.*

Remark: Developing fast algorithms for solving min-cut problems is an on-going research direction in computer science. The complexity result of Theorem 5.1 is for Ford-Fulkerson algorithm[1], the most well-known algorithm for min-cut problems. Algorithms faster than Ford-Fulkerson algorithm have been developed (see [113] and references therein).

[1] To be precise, it is a modification of Ford-Fulkerson algorithm, called Edmonds–Karp algorithm [112].

5.4.2 MAP-MRF = min-cut

This subsection shows that we can obtain the MAP solution $\boldsymbol{x}^{\mathrm{MAP}}$ from the min-cut K^*.

First, it can be easily shown that there is an one-to-one mapping between graph cut $K \in \mathcal{K}$ and sensing decision vector $\boldsymbol{x} \in \{0,1\}^N$. Specifically, for any K, we define a decision vector $\boldsymbol{x}^K = [x_1^K, \cdots, x_N^K]$ as

$$x_i^K = \begin{cases} 1, \text{ if } i \in \mathcal{B}, \\ 0, \text{ if } i \in \mathcal{F}. \end{cases} \tag{5.20}$$

On the other hand, for any sensing decision $\boldsymbol{x}$, $(b \cup \{i\}_{x_i=1}, f \cup \{i\}_{x_i=0})$ defines a cut.

Then, we show that the cost of cut K equals the posterior energy of $\boldsymbol{x}^K$ (5.20) (except for a constant).

Lemma 5.1. *For any K and its corresponding decision vector $\boldsymbol{x}^K$, we have*

$$c(K) = E(\boldsymbol{x}^K) + \textit{constant}.$$

Proof. Note that $c(K)$ can be expressed

$$c(K) = \sum_{e \in \mathcal{E}^{fi}} c(e) + \sum_{e \in \mathcal{E}^{bi}} c(e) + |\mathcal{E}^K \cap \mathcal{E}| \cdot \beta, \tag{5.21}$$

where $\mathcal{E}^{fi} = \{(f,i)\}_{i \in \mathcal{V}} \cap \mathcal{E}^K$, $\mathcal{E}^{bi} = \{(b,i)\}_{i \in \mathcal{V}} \cap \mathcal{E}^K$. Since $(f,i) \in \mathcal{E}^{fi}$ implies $i \in \mathcal{B}$, and therefore, $x_i^K = 1$ from (5.20), we have

$$\sum_{e \in \mathcal{E}^{fi}} c(e) = \sum_{i: x_i^K = 1} \eta_i(1) + \sum_{i: x_i^K = 1} M_i. \tag{5.22}$$

Similarly, we have

$$\sum_{e \in \mathcal{E}^{bi}} c(e) = \sum_{i: x_i^K = 0} \eta_i(0) + \sum_{i: x_i^K = 0} M_i. \tag{5.23}$$

Furthermore, we have

$$\begin{aligned} |\mathcal{E}^K \cap \mathcal{E}| \cdot \beta = |\mathcal{E}| \cdot \beta - |\mathcal{E} \backslash \mathcal{E}^K| \cdot \beta &\overset{\text{ⓐ}}{=} |\mathcal{E}| \cdot \beta + \sum_{(i,j) \in \mathcal{E} \backslash \mathcal{E}^K} \kappa(x_i^K, x_j^K) \\ &\overset{\text{ⓑ}}{=} |\mathcal{E}| \cdot \beta + \sum_{(i,j) \in \mathcal{E}} \kappa(x_i^K, x_j^K), \end{aligned} \tag{5.24}$$

where equality ⓐ holds, since $(i,j) \notin \mathcal{E}^K$ implies $x_i^K = x_j^K$ and $\kappa(x_i^K, x_j^K) = -\beta$; equality ⓑ holds, since if $(i,j) \in \mathcal{E}^K$, we have $x_i^K \neq x_j^K$ and $\kappa(x_i^K, x_j^K) = 0$. Summarizing (5.22)-(5.24) and also comparing with (5.12), we have

$$c(K) = \mathrm{E}(\boldsymbol{x}^K) + \sum_{i=1}^{N} M_i + |\mathcal{E}| \cdot \beta. \tag{5.25}$$

Note that $\sum_{i=1}^{N} M_i + |\mathcal{E}| \cdot \beta$ is a constant that does not depend on $\boldsymbol{x}^K$. □

We have shown that the mapping from K to $\boldsymbol{x}^K$ is bijective and $c(K) = \mathrm{E}(\boldsymbol{x}^K)$, which gives Theorem 5.2.

Theorem 5.2. *Given that K^* is a min-cut of $\mathcal{G}^{BF}$, $\boldsymbol{x}^{MAP} = \boldsymbol{x}^{K^*}$.*

Remark: Theorem 5.2 shows that the MAP-MRF problem (5.15) can be exactly solved by solving the min-cut problem (5.19), whose complexity is $\mathcal{O}(N \cdot |\mathcal{E}|^2)$, as stated in Theorem 5.1. Note that the number of edges $|\mathcal{E}|$ is a proxy of secondary network density, we can conclude that GC-CSS has polynomial time complexity versus network size and network density.

Algorithm 5.1 GC-CSS

procedure GC-CSS($\{y_i\}_{i\in\mathcal{V}}$, $\mathcal{G}$, MinCut($\cdot$))
  Construct $\mathcal{V}^{BF}$ and $\mathcal{E}^{BF}$ from $\mathcal{G}$
  Based on $\{y_i\}_{i\in\mathcal{V}}$, assign $c(e)$ for $e \in \mathcal{E}^{BF}$ via (5.16), (5.17) and (5.18)
  Get BF-Graph $\mathcal{G}^{BF} = (\mathcal{V}^{BF}, \mathcal{E}^{BF}, c(\cdot))$
  Find min-cut $K^* = \mathrm{MinCut}(\mathcal{G}^{BF})$
  Inform SUs sensing decision $\hat{\boldsymbol{x}} = \boldsymbol{x}^{K^*}$
end procedure

Summarizing above concepts, the GC-CSS algorithm is presented in Algorithm 5.1, where MinCut($\cdot$) is any min-cut algorithm (e.g., the Ford-Fulkerson algorithm, also see [113]) for solving (5.19).

5.5 DD-CSS for Cluster-based Secondary Networks

In this section, we presents DD-CSS for cluster-based secondary networks. It is assumed that SUs are grouped into several clusters (Fig. 5.4), where there is a cluster head (which can be a dedicated infrastructure or selected SU) for information col-

lecting and decision making within each cluster. Based on the dual decomposition[2] theory [114], DD-CSS distributedly (at cluster-level) estimates $\boldsymbol{x}^{\mathrm{MAP}}$ via iteratively exchanging messages among cluster heads.

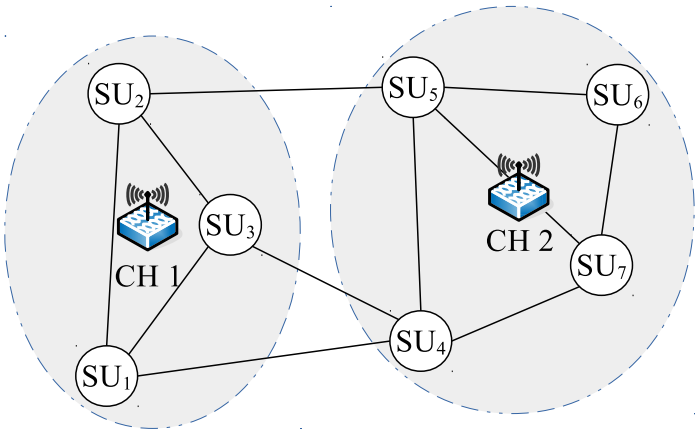


Figure 5.4: Cluster-based secondary network

In the following, we first construct subgraphs for a cluster-based secondary network (Section 5.5.1). Based on these subgraphs, dual decomposition is applied to address the MAP-MRF problem (5.15), from which DD-CSS algorithm is developed (Section 5.5.2).

5.5.1 Divide SU-graph into subgraphs

We consider a secondary network with L ($L \leq N$) clusters, whose cluster heads are, respectively, denoted as CH_1, ..., CH_L. We assume that these N SUs are completely and uniquely assigned to the L cluster heads. Specifically, denoting $\mathcal{C}_l$ as the set of SUs that are assigned to CH_l, we have $\mathcal{C}_l \cap \mathcal{C}_m = \emptyset$ if $l \neq m$, and $\cup_{l=1}^{L} \mathcal{C}_l = \mathcal{V}$.

CH_l collects sensing and neighbor information from $\mathrm{SU}_{\mathcal{C}_l}$ (i.e., $\{\mathrm{SU}_i\}_{i \in \mathcal{C}_l}$), and makes sensing decisions for these SUs. We name $\mathrm{SU}_{\mathcal{C}_l}$ as the set of member-SUs of CH_l. We further denote $\rho(i)$ as the (index of the) cluster that the ith SU belongs to. For example, in Fig. 5.4, we have $\mathcal{C}_1 = \{1, 2, 3\}$, $\mathcal{C}_2 = \{4, 5, 6, 7\}$, $\rho(i) = 1$, $\forall i \in \mathcal{C}_1$, and $\rho(i) = 2$, $\forall i \in \mathcal{C}_2$.

[2] Dual decomposition is widely used for solving MAP-MRF problems in machine learning, computer vision, natural language processing and others (see [114] and references therein). In most of these problems, the original MAP-MRF problem is NP-hard. Therefore, for ensuring the solvability of subproblems, the original problem can only be decomposed in certain specific ways. In contrast, our MAP-MRF problem can be efficiently solved by min-cut algorithms. The target of our decomposition is to minimize couplings among subproblems.

At CH_l, a subgraph $\mathcal{G}_l = (\mathcal{V}_l, \mathcal{E}_l)$ is defined (with some additional information from neighboring clusters). Specifically, the node set $\mathcal{V}_l$ is defined as

$$\mathcal{V}_l = \mathcal{C}_l \cup \{i \in \mathcal{V} | \mathcal{N}(i) \cap \mathcal{C}_l \neq \emptyset \text{ and } \rho(i) > l\}, \tag{5.26}$$

where $\mathcal{N}(i)$ denotes the set of neighboring SUs of SU_i. The edge set $\mathcal{E}_l$ of subgraph $\mathcal{G}_l$ is defined as

$$\mathcal{E}_l = \{(i,j) \in \mathcal{E} | i,j \in \mathcal{C}_l\} \cup \{(i,j) \in \mathcal{E} | i \in \mathcal{C}_l, j \in \mathcal{V}_l \backslash \mathcal{C}_l\}. \tag{5.27}$$

Note that the construction of $\mathcal{V}_l$ (5.26) and $\mathcal{E}_l$ (5.27) ensures that any edge of $\mathcal{G}$ appears exactly once in all subgraphs, as stated in Lemma 5.2.

Lemma 5.2. *Given L subgraphs constructed from* (5.26) *and* (5.27)*, we have $\mathcal{E}_l \cap \mathcal{E}_m = \emptyset$, $\forall l \neq m$, and $\cup_{l=1}^{L} \mathcal{E}_l = \mathcal{E}$.*

Proof. For an edge $(i,j) \in \mathcal{E}$, if $i,j \in \mathcal{C}_l$, we have $(i,j) \in \mathcal{E}_l$ due to (5.27). However, as $i,j \in \mathcal{C}_l$ implies $i,j \notin \mathcal{C}_m$, $\forall m \neq l$ (note that $\{\mathcal{C}_l\}_l$ are disjoint), we have $(i,j) \notin \mathcal{E}_m$, $\forall m \neq l$.

On the other hand, given the edge $(i,j) \in \mathcal{E}$ with $i \in \mathcal{C}_l$, $j \in \mathcal{C}_m$ and $l < m$ (without loss of generality), then we have $j \in \mathcal{V}_l$ and $i \notin \mathcal{V}_m$ due to (5.26), which implies $(i,j) \in \mathcal{E}_l$ and $(i,j) \notin \mathcal{E}_m$. Also considering that $\{\mathcal{C}_l\}_l$ are disjoint, we conclude that (i,j) uniquely belongs to $\mathcal{E}_l$. □

Lemma 5.2 states that there is no common edge among subgraphs. However, a subgraph may overlap with another subgraph at some nodes (i.e., SUs). Specifically, we call an SU a gate-SU, if it is involved in more than one subgraph. Denoting

$$\mathcal{L}(i) = \{l | i \in \mathcal{V}_l\} \tag{5.28}$$

as the set of (the indices of) subgraphs that SU_i is involved, the set of gate-SUs can be defined as

$$\mathcal{H} = \{i \in \mathcal{V} \mid |\mathcal{L}(i)| > 1\}. \tag{5.29}$$

Among them, the set of SUs

$$\mathcal{H}_l = \{i \in \mathcal{V}_l \mid |\mathcal{L}(i)| > 1\} \tag{5.30}$$

are called the CH_l's gate-SUs. As an example, Fig. 5.5 shows subgraphs $\mathcal{G}_1$ and $\mathcal{G}_2$ that are constructed from the cluster-based secondary network in Fig. 5.4, Note that these two subgraphs are overlapped at gate-SUs SU_4 and SU_5.

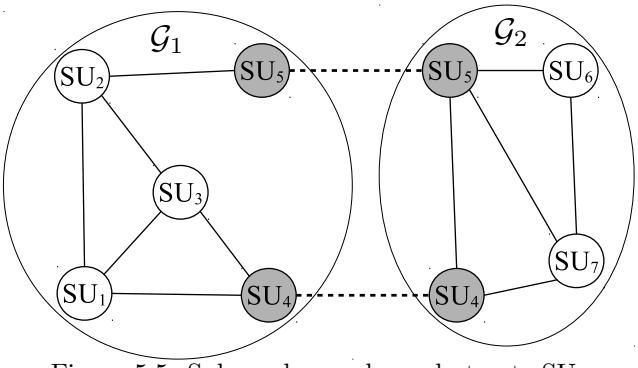


Figure 5.5: Subgraphs overlapped at gate-SUs

5.5.2 DD-CSS: inter-cluster message passing algorithm

Given subgraphs $\{\mathcal{G}_l\}_l$, dual decomposition [115] is applied to address the MAP-MRF problem (5.15). Specifically, the MAP-MRF problem is decomposed over subgraphs, which provides two things: one ***master problem*** and L ***slave problems***. These slave problems are, respectively, assigned to L cluster heads; while the master problem coordinates L cluster heads to solve the MAP-MRF problem via iteratively solving their slave problems. The above mentioned steps are detailed in this subsection.

5.5.2.1 Decompose MAP-MRF problem over subgraphs

From Lemma 5.2 and (5.28), we can decompose the posterior energy (5.12) over subgraphs as

$$\mathrm{E}(\boldsymbol{x}) = \sum_{l=1}^{L} \mathrm{E}_l(\boldsymbol{x}), \tag{5.31}$$

where $\mathrm{E}_l(\boldsymbol{x})$ is defined as

$$\mathrm{E}_l(\boldsymbol{x}) = \sum_{i \in \mathcal{V}_l} \frac{1}{|\mathcal{L}(i)|} \eta_i(x_i) + \sum_{(i,j) \in \mathcal{E}_l} \kappa(x_i, x_j). \tag{5.32}$$

Notice that $\mathrm{E}_l(\boldsymbol{x})$ can be constructed at CH_l with local topological information $\mathcal{G}_l$ and local sensing observations $\{y_i\}_{i \in \mathcal{V}_l}$. We would like to solve the MAP-MRF problem (5.15) (i.e., minimizing $\mathrm{E}(\boldsymbol{x})$) by letting each cluster head, say CH_l, to determine its sensing decision vector $\boldsymbol{x}^{(l)} \in \{0,1\}^N$ via considering $\mathrm{E}_l(\boldsymbol{x}^{(l)})$. Note that, although

the values of $\boldsymbol{x}^{(l)}_{\mathcal{V}\setminus\mathcal{V}_l}$ (i.e., $x^{(l)}_i$, $\forall i \notin \mathcal{V}_l$) do not affect $\mathrm{E}_l(\boldsymbol{x}^{(l)})$, we define $\boldsymbol{x}^{(l)}$ as an N dimensional vector for notational convenience.

However, independently minimizing $\mathrm{E}_l(\boldsymbol{x}^{(l)})$, $\forall l$, does not solve (5.15), since neighboring subgraphs are coupled at gate-SUs. Therefore, the decisions of CH_l and CH_m should be the same at $\mathrm{SU}_{\mathcal{H}_l\cap\mathcal{H}_m}$, i.e., $x^{(l)}_i = x^{(m)}_i$, $\forall i \in \mathcal{H}_l \cap \mathcal{H}_m$. Equivalently, we can introduce an auxiliary variables $\boldsymbol{z} \in \{0,1\}^N$, and require that, for all l, $\boldsymbol{x}^{(l)}_{\mathcal{H}_l} = \boldsymbol{z}_{\mathcal{H}_l}$ (i.e., $x^{(l)}_i = z_i$, $\forall i \in \mathcal{H}_l$). In summary, the MAP-MRF problem (5.15) can be reformulated as

$$\begin{aligned} \underset{\{\boldsymbol{x}^{(l)}\}_l,\, \boldsymbol{z}}{\text{minimize}} \quad & \sum_{l=1}^{L} \mathrm{E}_l(\boldsymbol{x}^{(l)}) \\ \text{subject to} \quad & \boldsymbol{x}^{(l)}_{\mathcal{H}_l} = \boldsymbol{z}_{\mathcal{H}_l},\ \forall l. \end{aligned} \tag{5.33}$$

Next, we relax the constraint of (5.33) by the method of Lagrange multiplier [116, Chapter 5]. Specifically, via introducing (Lagrange) multipliers $\{\boldsymbol{\lambda}^{(l)} \in \mathbb{R}^N\}_l$ such that

$$\lambda^{(l)}_i = 0,\ \forall l,\ \forall i \notin \mathcal{H}_l, \tag{5.34}$$

problem (5.33) is relaxed as

$$\underset{\{\boldsymbol{x}^{(l)}\}_l,\, \boldsymbol{z}}{\text{minimize}} \quad \sum_{l=1}^{L} \mathrm{E}_l(\boldsymbol{x}^{(l)}) + \sum_{l=1}^{L} \left\langle \boldsymbol{\lambda}^{(l)}, \boldsymbol{x}^{(l)} - \boldsymbol{z} \right\rangle, \tag{5.35}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product. From (5.35), a (Lagrangian) dual function $g(\{\boldsymbol{\lambda}^{(l)}\}_l)$ is defined as

$$g(\{\boldsymbol{\lambda}^{(l)}\}_l) = \min_{\{\boldsymbol{x}^{(l)}\}_l, \boldsymbol{z}} \left\{ \sum_{l=1}^{L} \mathrm{E}_l(\boldsymbol{x}^{(l)}) + \sum_{l=1}^{L} \left\langle \boldsymbol{\lambda}^{(l)}, \boldsymbol{x}^{(l)} - \boldsymbol{z} \right\rangle \right\}. \tag{5.36}$$

Note that the dual function $g(\{\boldsymbol{\lambda}^{(l)}\}_l)$ with any $\{\boldsymbol{\lambda}^{(l)}\}_l$ is actually a lower bound of the posterior energy $\mathrm{E}(\boldsymbol{x})$, since we have

$$g(\{\boldsymbol{\lambda}^{(l)}\}_l) \leq \sum_{l=1}^{L} \mathrm{E}_l(\boldsymbol{x}) + \sum_{l=1}^{L} \left\langle \boldsymbol{\lambda}^{(l)}, \boldsymbol{x} - \boldsymbol{x} \right\rangle = \mathrm{E}(\boldsymbol{x}), \tag{5.37}$$

where the inequality follows by choosing $\boldsymbol{x}^{(l)} = \boldsymbol{z} = \boldsymbol{x}$.

5.5.2.2 Define master problem and slave problems

From the dual function $g(\cdot)$, the master and slave problems are defined. Before that, we further simplify the dual function. Specifically, $g(\cdot)$ (5.36) can be rewritten as

$$g(\{\boldsymbol{\lambda}^{(l)}\}_l) = \min_{\{\boldsymbol{x}^{(l)}\}_l, \boldsymbol{z}} \left\{ \sum_{l=1}^{L} \left(\mathrm{E}_l(\boldsymbol{x}^{(l)}) + \left\langle \boldsymbol{\lambda}^{(l)}, \boldsymbol{x}^{(l)} \right\rangle \right) - \sum_{i \in \mathcal{H}} \left(\sum_{l \in \mathcal{L}(i)} \lambda_i^{(l)} \right) z_i \right\}.$$

In addition, via constraining the multipliers with

$$\sum_{l \in \mathcal{L}(i)} \lambda_i^{(l)} = 0, \ \forall i \in \mathcal{H}, \tag{5.38}$$

we can eliminate $\boldsymbol{z}$ and simplify $g(\{\boldsymbol{\lambda}^{(l)}\}_l)$ as

$$g(\{\boldsymbol{\lambda}^{(l)}\}_l) = \sum_{l=1}^{L} \min_{\boldsymbol{x}^{(l)}} \left\{ \mathrm{E}_l(\boldsymbol{x}^{(l)}) + \left\langle \boldsymbol{\lambda}^{(l)}, \boldsymbol{x}^{(l)} \right\rangle \right\}. \tag{5.39}$$

Here, the master and slave problems are defined. Specifically, the lth slave problem is defined as

$$g_l(\boldsymbol{\lambda}^{(l)}) = \min_{\boldsymbol{x}^{(l)}} \left\{ \mathrm{E}_l(\boldsymbol{x}^{(l)}) + \left\langle \boldsymbol{\lambda}^{(l)}, \boldsymbol{x}^{(l)} \right\rangle \right\}, \tag{5.40}$$

which is assigned to CH_l. The master problem is defined as

$$\underset{\{\boldsymbol{\lambda}^{(l)}\}_l \in \Lambda}{\text{maximize}} \quad g(\{\boldsymbol{\lambda}^{(l)}\}_l) = \sum_{l=1}^{L} g_l(\boldsymbol{\lambda}^{(l)}), \tag{5.41}$$

where

$$\Lambda \triangleq \left\{ \{\boldsymbol{\lambda}^{(l)}\}_l \middle| \overbrace{\sum_{l \in \mathcal{L}(i)} \lambda_i^{(l)} = 0, \ \forall i \in \mathcal{H}}^{\text{(a)}}, \overbrace{\lambda_i^{(l)} = 0, \forall l, \forall i \notin \mathcal{H}_l}^{\text{(b)}} \right\}. \tag{5.42}$$

Remind that, in (5.42), constraint (a) is from (5.38), and constraint (b) is from (5.34)

In the following, we will show that

- the master problem can be solved by iteratively solving salve problems;
- the lth slave problem can be solved at CH_l by applying GC-CSS;
- the solving of the master problem actually coordinate cluster heads to recover $\boldsymbol{x}^{\mathrm{MAP}}$.

5.5.2.3 Solve master problem with slave solutions

Since the master problem (5.41) is concave, it can be solved with projected supergradient methods by iteratively updating multipliers. Specifically, denoting $\{\boldsymbol{\lambda}^{(l,t)}\}_l$ as the multipliers of the t-th ($t \geq 0$) iteration, $\boldsymbol{\lambda}^{(l,t)}$ is iteratively updated as

$$\boldsymbol{\lambda}^{(l,t+1)} = \left[\boldsymbol{\lambda}^{(l,t)} + \alpha_t \cdot \nabla g_l(\boldsymbol{\lambda}^{(l,t)})\right]_\Lambda, \forall l, \tag{5.43}$$

where $\boldsymbol{\lambda}^{(l,0)} = \mathbf{0}$, $\alpha_t \in (0,1)$ is a step size, $\nabla g_l(\boldsymbol{\lambda}^{(l,t)})$ is the supergradient of $g_l(\cdot)$ at $\boldsymbol{\lambda}^{(l,t)}$, and $[\cdot]_\Lambda$ is a projection onto (5.42). From [117, Chapter 7], we have following theorem.

Theorem 5.3. *Given that α_t satisfies*[3]

$$\sum_{t=0}^{\infty} \alpha_t = \infty, \text{ and } \sum_{t=0}^{\infty} \alpha_t^2 < \infty, \tag{5.44}$$

$\{\boldsymbol{\lambda}^{(l,t)}\}_l$ generated via (5.43) solves the master problem (5.41) as $t \to \infty$.

The supergradient $\nabla g_l(\boldsymbol{\lambda}^{(l,t)})$ is given in Lemma 5.3.

Lemma 5.3. *Let $\boldsymbol{x}^{(l,t)}$ be a solution to (5.40) with $\boldsymbol{\lambda}^{(l)} = \boldsymbol{\lambda}^{(l,t)}$. We have $\nabla g_l(\boldsymbol{\lambda}^{(l,t)}) = \boldsymbol{x}^{(l,t)}$.*

Proof. For any $\boldsymbol{\lambda}^{(l)} \neq \boldsymbol{\lambda}^{(l,t)}$, we have

$$\begin{aligned} g_l(\boldsymbol{\lambda}^{(l)}) &\leq \mathrm{E}_l(\boldsymbol{x}^{(l,t)}) + \left\langle \boldsymbol{\lambda}^{(l)}, \boldsymbol{x}^{(l,t)} \right\rangle \\ &= \mathrm{E}_l(\boldsymbol{x}^{(l,t)}) + \left\langle \boldsymbol{\lambda}^{(l,t)}, \boldsymbol{x}^{(l,t)} \right\rangle + \left\langle \boldsymbol{\lambda}^{(l)} - \boldsymbol{\lambda}^{(l,t)}, \boldsymbol{x}^{(l,t)} \right\rangle \\ &= g_l(\boldsymbol{\lambda}^{(l,t)}) + \left\langle \boldsymbol{\lambda}^{(l)} - \boldsymbol{\lambda}^{(l,t)}, \boldsymbol{x}^{(l,t)} \right\rangle, \end{aligned}$$

which completes the proof from the definition of supergradient. □

Therefore, from Lemma 5.3 and the definition of Λ (5.42), the updating rule of $\boldsymbol{\lambda}^{(l,t)}$ (5.43) reduces to

$$\lambda_i^{(l,t+1)} = \lambda_i^{(l,t)} + \alpha_t \cdot \delta_i^{(l,t)}, \forall l, \forall i, \tag{5.45}$$

where $\delta_i^{(l,t)}$ is defined as

$$\delta_i^{(l,t)} = x_i^{(l,t)} - \frac{1}{|\mathcal{L}(i)|} \sum_{m \in \mathcal{L}(i)} x_i^{(m,t)}. \tag{5.46}$$

[3] Condition (5.44) can be satisfied by, for example, $\alpha_t = 1/t$.

From (5.45) and (5.46), we see that the master problem can be solved by iteratively solving slave problems. Next, we show that the slave solution $\boldsymbol{x}^{(l,t)}$ can be obtained by applying GC-CSS.

5.5.2.4 Solve slave problems with GC-CSS

Due to the definition of $\mathrm{E}_l(\cdot)$ (5.32), a solution $\boldsymbol{x}^{(l,t)}$ to (5.40) given $\boldsymbol{\lambda}^{(l,t)}$ can be obtained by solving

$$\boldsymbol{x}^{(l,t)} = \arg\min_{\boldsymbol{x}} \left\{ \sum_{i\in\mathcal{V}_l} \eta_i^{(l,t)}(x_i) + \sum_{(i,j)\in\mathcal{E}_l} \kappa(x_i, x_j) \right\}, \tag{5.47}$$

where $\eta_i^{(l,t)}(x_i)$ is defined as

$$\eta_i^{(l,t)}(x_i) = \begin{cases} \frac{1}{|\mathcal{L}(i)|}\eta_i(1) + \lambda_i^{(l,t)} & \text{if } x_i = 1, \\ \frac{1}{|\mathcal{L}(i)|}\eta_i(0) & \text{if } x_i = 0. \end{cases} \tag{5.48}$$

Remind that, in (5.47), $\boldsymbol{x}_{\mathcal{V}\setminus\mathcal{V}_l}^{(l,t)}$ can take arbitrary values, and only the values of $\boldsymbol{x}_{\mathcal{V}_l}^{(l,t)}$ need to be determined. Hence, via comparing with (5.15), problem (5.47) can be seen as a MAP-MRF problem defined over subgraph $\mathcal{G}_l$ with edge energy κ and unitary energy $\eta_i^{(l,t)}$. Therefore, problem (5.47) can be solved via solving a corresponding min-cut problem (see Theorem 5.2). Specifically, CH_l can obtain $\boldsymbol{x}_{\mathcal{V}_l}^{(l,t)}$ from GC-CSS (Algorithm 5.1) by 1) replacing inputs $\{y_i\}_{i\in\mathcal{V}}$ and $\mathcal{G}$ with $\{y_i\}_{i\in\mathcal{V}_l}$ and $\mathcal{G}_l$, respectively, 2) and replacing η_i with $\eta_i^{(l,t)}$ in equations (5.17)-(5.18) (also changing M_i accordingly).

5.5.2.5 Master problem coordinate clusters to recover $\boldsymbol{x}^{\mathbf{MAP}}$

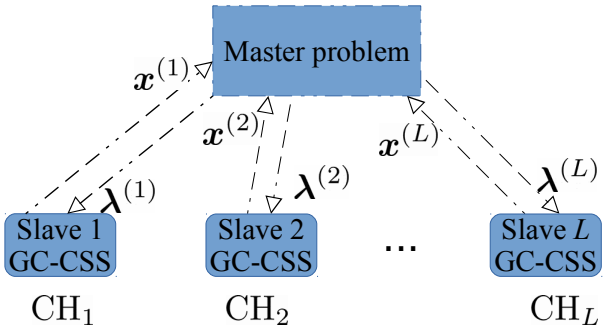


Figure 5.6: Solve master problem via iteratively solving slave problems

So far, we have shown that, for solving the master problem, slave problems need to be iteratively solved (Fig. 5.6), which is detailed as follows.

- At the t-th iteration, assume that the master problem (5.41) has multipliers $\{\boldsymbol{\lambda}^{(l,t)}\}_l$. Then, for all l,
  - Send $\boldsymbol{\lambda}^{(l,t)}$ to CH_l;
  - CH_l obtains $\boldsymbol{x}^{(l,t)}$ by solving the slave problem (5.40) with $\boldsymbol{\lambda}^{(l)} = \boldsymbol{\lambda}^{(l,t)}$ via GC-CSS;
  - CH_l sends back $\boldsymbol{x}^{(l,t)}$.
- With received $\{\boldsymbol{x}^{(l,t)}\}_l$, update multipliers and obtain $\{\boldsymbol{\lambda}^{(l,t+1)}\}_l$ via (5.45).
- Set $t = t + 1$, and repeat the procedure.

In the following, we will show that 1) the master problem works as a coordinator, which coordinates cluster heads to agree on gate-SUs; 2) given that all cluster heads agree on gate-SUs, slave solutions recover $\boldsymbol{x}^{\mathrm{MAP}}$.

At the t-th iteration, after cluster heads decide $\{\boldsymbol{x}^{(l,t)}\}_l$, assuming that not all related cluster heads (i.e., $\mathrm{CH}_{\mathcal{L}(i)}$) agree on x_i (SU_i is a gate-SU), we must have

$$0 < \frac{1}{|\mathcal{L}(i)|} \sum_{m \in \mathcal{L}(i)} x_i^{(m,t)} < 1. \tag{5.49}$$

Then, slave solutions $\{\boldsymbol{x}^{(l,t)}\}_l$ are sent back to the master problem, and multipliers are updated:

- Assume that CH_l ($l \in \boldsymbol{\mathcal{L}}(i)$) has decided $x_i^{(l,t)}$ to be 0 (or 1).
- From (5.46) and (5.49), we know that $\sigma_i^{(l,t)}$ is less (or greater) than 0.
- Therefore, (5.45) implies that the updated $\lambda_i^{(l,t+1)}$ is decreased (or increased) by $|\alpha_t \cdot \delta_i^{(l,t)}|$ (compared with $\lambda_i^{(l,t)}$).

Next, the $(t+1)$th iteration begins. Each cluster head, say CH_l, receives updated multipliers, and solves its slave problem at $\boldsymbol{\lambda}^{(l,t+1)}$:

- From (5.48), we know that, given $\lambda_i^{(l,t+1)}$, the updated unitary energy $\eta_i^{(l,t+1)}(1)$ is decreased (or increased) by $|\alpha_t \cdot \delta_i^{(l,t)}|$ (compared with $\eta_i^{(l,t)}(1)$).
- Remind that, as defined in (5.14), we have $\eta_i(1) = -\ln(\gamma f_{Y|X}(y_i|1))$, which is inversely proportional to the likelihood that $x_i = 1$.

- Hence, given a decreased (or increased) $\eta_i^{(l,t+1)}(1)$, CH_l is more likely to decide $x_i^{(l,t+1)}$ to be 1 (or 0) after solving (5.47).

Finally, with all $\mathrm{CH}_{\mathcal{L}(i)}$ adjust $\eta_i(1)$ in this way, they are more likely to agree on x_i at the $(t+1)$th iteration.

Above analysis shows that multipliers summarize the decision information of correlated clusters. From received multipliers, if a cluster head finds disagreements from correlated clusters, it "reconsiders" or changes its decisions in order to make an agreement on gate-SUs. After T iterations, if all cluster heads have agreed on gate-SUs, we can recover $\boldsymbol{x}^{\mathrm{MAP}}$ from slave solutions, as stated in following Theorem.

Theorem 5.4. *Assuming that, at the T-th iteration,*

$$x_i^{(l,T)} = x_i^{(m,T)}, \forall i, \forall l, m \in \mathcal{L}(i), \tag{5.50}$$

we can construct a decision vector $\boldsymbol{x}^{Agg}$ as

$$\boldsymbol{x}_{\mathcal{V}_l}^{Agg} = \boldsymbol{x}_{\mathcal{V}_l}^{(l,T)}, \forall l, \tag{5.51}$$

and it solves the MAP-MRF problem (5.15)*, i.e., $\boldsymbol{x}^{MAP} = \boldsymbol{x}^{Agg}$.*

Proof. Since $\cup\mathcal{V}_l = \mathcal{V}$, (5.51) assigns sensing decisions at all SUs. In addition, due to (5.50), there is no confliction during the assignment. Therefore, (5.51) uniquely define a decision vector.

From definition of $\boldsymbol{x}^{\mathrm{MAP}}$ (5.15), we must have

$$\mathrm{E}(\boldsymbol{x}^{\mathrm{Agg}}) \geq \mathrm{E}(\boldsymbol{x}^{\mathrm{MAP}}). \tag{5.52}$$

On the other hand, assuming that, $\boldsymbol{x}^{(l,T)}$ is solved at $\boldsymbol{\lambda}^{(l,T)}$ in (5.40), we have

$$\begin{aligned} g(\{\boldsymbol{\lambda}^{(l,T)}\}_l) &\overset{\text{ⓐ}}{=} \sum_{l=1}^{L} \mathrm{E}_l(\boldsymbol{x}^{\mathrm{Agg}}) + \sum_{i\in\mathcal{H}} \left(\sum_{l\in\mathcal{L}(i)} \lambda_i^{(l,T)} \right) x_i^{\mathrm{Agg}} \\ &\overset{\text{ⓑ}}{=} \sum_{l=1}^{L} \mathrm{E}_l(\boldsymbol{x}^{\mathrm{Agg}}) = \mathrm{E}(\boldsymbol{x}^{\mathrm{Agg}}) \overset{\text{ⓒ}}{\leq} \mathrm{E}(\boldsymbol{x}^{\mathrm{MAP}}), \end{aligned} \tag{5.53}$$

where ⓐ is due to (5.39) and (5.40), ⓑ is due to (5.38), and ⓒ is due to (5.37). Therefore, (5.52) and (5.53) give $\mathrm{E}(\boldsymbol{x}^{\mathrm{Agg}}) = \mathrm{E}(\boldsymbol{x}^{\mathrm{MAP}})$, which implies $\boldsymbol{x}^{\mathrm{Agg}} = \boldsymbol{x}^{\mathrm{MAP}}$.

□

Remark: We have intuitively shown that cluster heads are coordinated to make an agreement at gate-SUs (5.50). Theorem 5.4 shows that, if the agreement is achieved, the MAP-MRF problem is solved, and therefore, CH_l can accomplish its CSS task by informing its member-SUs $\text{SU}_{\mathcal{C}_l}$ of current slave solution $\boldsymbol{x}_{\mathcal{C}_l}^{(l,T)}$.

Remark: However, dual decomposition does not theoretically guarantee condition (5.50) (in general cases). Nevertheless, given sufficiently large T, even if (5.50) is not strictly satisfied, disagreements should occur at a few gate-SUs. Hence, $\boldsymbol{x}_{\mathcal{C}_l}^{(l,T)}$ should be close to $\boldsymbol{x}_{\mathcal{C}_l}^{\text{MAP}}$, and therefore, we still consider it as the sensing decision of CH_l.

In special cases, where subgraphs do not contain any loop (e.g., subgraphs are trees), there is theoretical guarantee that dual decomposition is able to solve the MAP-MRF problem. Specifically, we have Theorem 5.5, which is a result of [114, Theorem 5].

Theorem 5.5. *Assume $\{\boldsymbol{\lambda}^{(l*)}\}_l$ are the multipliers when the master problem is solved. Denoting $\{\boldsymbol{x}^{(l*)}\}_l$ as the salve solutions of (5.40) given $\{\boldsymbol{\lambda}^{(l*)}\}_l$, we have $\boldsymbol{x}_{\mathcal{V}_l}^{(l*)} = \boldsymbol{x}_{\mathcal{V}_l}^{MAP}$, $\forall l$, if subgraphs $\{\mathcal{G}_l\}_l$ do not contain any loop.*

5.5.2.6 Distributed implementation and DD-CSS

We have shown that the master problem coordinates cluster heads to estimate $\boldsymbol{x}^{\text{MAP}}$, where the key is to iteratively update slave problems with multipliers $\boldsymbol{\lambda}^{(l,t)}$ (5.48), and solve the updated slave problems (5.47).

We will show that, without explicitly solving the master problem or tracking the multipliers, slave problems can be updated equivalently as (5.48) via exchanging messages among cluster heads. Specifically, with initialization $\eta_i^{(l,0)}(x_i)$ as

$$\eta_i^{(l,0)}(x_i) = \frac{1}{|\mathcal{L}(i)|}\eta_i(x_i), \tag{5.54}$$

it is easy to verify that $\eta_i^{(l,t)}$ computed via (5.55) is equivalent to that of (5.48)

$$\eta_i^{(l,t+1)}(x_i) = \begin{cases} \eta_i^{(l,t)}(1) + \alpha_t \cdot \delta_i^{(l,t)} & \text{if } x_i = 1, \\ \eta_i^{(l,t)}(0) & \text{if } x_i = 0, \end{cases} \tag{5.55}$$

where $\delta_i^{(l,t)}$ is defined in (5.46). Note that, since $\mathcal{L}(i) = l$, $\forall i \notin \mathcal{H}_l$, (5.54) and (5.46) imply $\eta_i^{(l,t)} = \eta_i$, $\forall i \notin \mathcal{H}_l$. Therefore, at each iteration, we only need to update

$\eta_i^{(l,t)}$ at the gate-SUs of CH_l (with $\delta_i^{(l,t)}$). This can be accomplished, if $\text{CH}_{\mathcal{L}(i)\backslash\{l\}}$ inform CH_l their decisions about x_i (see (5.46)). In summary, given that all cluster heads exchange their decisions about their common gate-SUs, $\eta_i^{(l,t)}$ can be updated equivalently as (5.48).

Algorithm 5.2 DD-CSS at CH_l

1: **procedure** DD-CSS($\{y_i\}_{i\in\mathcal{V}_l}$, $\mathcal{C}_l$, $\mathcal{G}_l$, $\mathcal{H}_l$, $\{\mathcal{L}(i)\}_{i\in\mathcal{H}_l}$, GC-CSS($\cdot$))
2: $\quad\forall i \in \mathcal{H}_l$, initialize $\eta_i^{(l,0)}$ via (5.54) based on $\{y_i, \mathcal{L}(i)\}_{i\in\mathcal{H}_l}$
3: $\quad$**for** t from 0 to T **do**
4: $\quad\quad$Obtain $\boldsymbol{x}_{\mathcal{V}_l}^{(l,t)}$ = GC-CSS($\{y_i\}_{i\in\mathcal{V}_l}, \mathcal{G}_l$) via replacing η_i with $\eta_i^{(l,t)}$, for $i \in \mathcal{H}_l$, in (5.17)-(5.18)
5: $\quad\quad$**if** $t = T$ break **endif**
6: $\quad\quad$**parfor** $i \in \mathcal{H}_l$ **do**
7: $\quad\quad\quad$**parfor** $m \in \mathcal{L}(i)\backslash\{l\}$ **pardo**
8: $\quad\quad\quad\quad$Send $x_i^{(l,t)}$ to CH_m
9: $\quad\quad\quad\quad$Receive $x_i^{(m,t)}$ from CH_m
10: $\quad\quad\quad$**end parfor-pardo**
11: $\quad\quad\quad$Compute $\delta_i^{(l,t)}$ via (5.46) based on $\{x_i^{(m,t)}\}_{m\in\mathcal{L}(i)}$
12: $\quad\quad\quad$Update $\eta_i^{(l,t+1)}$ via (5.55) based on $\delta_i^{(l,t)}$
13: $\quad\quad$**end parfor-do**
14: $\quad$**end for**
15: $\quad$Inform $\text{SU}_{\mathcal{C}_l}$ sensing decisions $\hat{\boldsymbol{x}}_{\mathcal{C}_l} = \boldsymbol{x}_{\mathcal{C}_l}^{(l,T)}$
16: **end procedure**

Given that cluster heads exchange decisions and update unitary energy with (5.55), slave problems (5.47) can be solved distributedly. This results a CSS algorithm that estimates $\boldsymbol{x}^{\text{MAP}}$ via exchanging messages among cluster heads. Summarizing above concepts, DD-CSS is provided in Algorithm 5.2. Note that the exact message exchanging schemes depend on how wireless channels are shared and clusters are scheduled. Therefore, in Algorithm 5.2, we use "parfor" and "pardo" to avoid specifying the details of message exchanging. Specifically, "parfor" of line 6 and line 7 means sweeping involved gate-SUs and clusters in parallel. At line 7, "pardo" means sending and receiving messages in parallel.

The major computational burden of DD-CSS is for iteratively executing GC-CSS (line 4), where a min-cut problem defined over subgraph $\mathcal{G}_l$ needs to be solved. Therefore, from Theorem 5.1, the complexity result of DD-CSS is given in Corollary 5.1.

Corollary 5.1. *The complexity for CH_l to execute DD-CSS Algorithm 5.2 is $\mathcal{O}(T \cdot |\mathcal{V}_l| \cdot |\mathcal{E}_l|^2)$, where T is the iteration budget.*

Remark: In Corollary 5.1, the complexity result of DD-CSS is for the case where, at each iteration, the slave problem (line 4) is solved (with the Ford-Fulkerson algorithm) from scratch. However, since, at different iterations, only unitary energy at gate-SUs is updated, we are able to solve the slave problem more efficiently by exploiting the solutions of previous iterations [118].

5.6 DD1-CSS for Distributed Secondary Networks

For distributed secondary networks, a CSS algorithm can be easily obtained from DD-CSS. Specifically, we can treat a distributed network as a cluster-based network by forming a cluster at each SU. Applying DD-CSS Algorithm 5.2 on this special cluster-based secondary network, we get DD1-CSS, a fully distributed CSS algorithm, where SUs cooperatively solve the MAP-MRF problem (5.15) via iteratively exchanging their decisions (messages) with nearby SUs. This section presents details of DD1-CSS and also compares it with BP algorithms.

5.6.1 Two-hop message-passing

Here, we show that DD1-CSS is a two-hop message-passing algorithm, where an SU exchanges its decisions with (some of) SUs that are within two hops.

Firstly, we divide the SU-graph into N subgraphs following the method in Section 5.5.1. Without loss of generality, we form the ith "cluster" at SU_i, i.e., $\mathcal{C}_i = i,\ \forall i \in \mathcal{V}$. Therefore, for the ith subgraph $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$, the node set $\mathcal{V}_i$ constructed from (5.26) can be expressed as

$$\mathcal{V}_i = \{i\} \cup \mathcal{N}_L(i), \tag{5.56}$$

where

$$\mathcal{N}_L(i) \triangleq \{l \in \mathcal{N}(i) | l > i\} \tag{5.57}$$

representing the "large" neighbors of SU_i. From (5.27), the edge set $\mathcal{E}_i$ is

$$\mathcal{E}_i = \{(i,j) | j \in \mathcal{N}_L(i)\}. \tag{5.58}$$

Furthermore, from (5.26) and (5.28), we identify the set of subgraphs in which SU_i is involved as

$$\mathcal{L}(i) = \{i\} \cup \mathcal{N}_S(i), \tag{5.59}$$

where $\mathcal{N}_S(i) \triangleq \{l \in \mathcal{N}(i) | l < i\}$ representing the "small" neighbors of SU_i. From $\mathcal{L}(i)$ (5.59), we can identify the gate-SUs $\mathcal{H}_i$ via (5.30).

Secondly, given above information of $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$, $\mathcal{H}_i$, $\{\mathcal{L}(j)\}_{j\in\mathcal{H}_i}$, we get DD1-CSS by applying DD-CSS Algorithm 5.2 at SU_i, which gives following message exchanging and decision making scheme.

- At t-th iteration, SU_i gets $\{x_j^{(i,t)}\}_{j\in\mathcal{V}_i}$ via GC-CSS (similar to line 4 of Algorithm 5.2);
- If $t = T$, get sensing decision $\hat{x}_i = x_i^{(i,T)}$ and terminate DD1-CSS, otherwise, continue the following step;
- For all $j \in \mathcal{H}_i$:
  - SU_i exchanges its decision $x_j^{(i,t)}$ with all SUs in $\mathcal{L}(j)\backslash\{i\}$ (similar to lines 8 and 9 of Algorithm 5.2);
  - Given received decisions $\{x_j^{(l,t)}\}_{l\in\mathcal{L}(j)}$, SU_i computes $\delta_j^{(i,t)}$, updates $\eta_j^{(i,t+1)}$ (similar to lines 11 and 12 of Algorithm 5.2);
- Go to the next iteration with $t = t + 1$.

Finally, we will show that, with above message exchanging scheme, an SU needs to exchange decisions with all of its one-hop neighbors and some of its two-hop neighbors. It is easy to see that SU_i needs to exchange messages with SU_j, if and only if $j \in \cup_{k\in\mathcal{V}_i}\mathcal{L}(k)\backslash\{i\}$. Hence, denoting the set of communicating SUs of SU_i as $\mathcal{M}_i$, we have

$$\mathcal{M}_i = \cup_{k\in\mathcal{V}_i}\mathcal{L}(k)\backslash\{i\}.$$

Plugging (5.56) and (5.59) gives

$$\mathcal{M}_i = \Big(\{i\} \cup \mathcal{N}_S(i) \bigcup\nolimits_{k\in\mathcal{N}_L(i)} (\{k\} \cup \mathcal{N}_S(k))\Big) \backslash\{i\} = \mathcal{N}(i) \bigcup\nolimits_{k\in\mathcal{N}_L(i)} N_S(k)\backslash\{i\}.$$

That is, during the execution of DD1-CSS, an SU needs to exchange messages with all neighbors $\mathcal{N}(i)$ and large neighbors' small neighbors $\bigcup_{k\in\mathcal{N}_L(i)} N_S(k)\backslash\{i\}$.

An example is given in Fig. 5.7, where $\mathcal{V}_1 = \{1, 2\}$, $\mathcal{V}_2 = \{2, 3\}$, $\mathcal{V}_3 = \{3, 5\}$, $\mathcal{V}_4 = \{4, 5\}$, $\mathcal{V}_5 = \{5, 6\}$ and $\mathcal{V}_6 = \{6\}$ (see (5.56)). During the execution of DD1-CSS, SU_3 needs to exchange decisions of x_3 with SU_2, and exchange decisions of x_5 with SU_4 and SU_5.

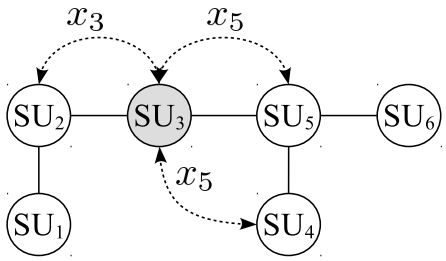


Figure 5.7: Communicating SUs of SU_3 for DD1-CSS

5.6.1.1 Theoretical guarantee and computation complexity

From (5.58), we know that subgraphs $\{\mathcal{G}_i\}_i$ do not contain loop. Actually, subgraph $\mathcal{G}_i$ has a "star" topology, where node i (as the center) connects to $|\mathcal{N}_L(i)|$ nodes. But this loop-free feature confers DD1-CSS following theoretical guarantee by applying Theorem 5.5 and Theorem 5.3.

Corollary 5.2. *DD1-CSS is guaranteed to recover $\boldsymbol{x}^{MAP}$, given α_t (in* (5.55)*) satisfying* (5.44) *and the number of iterations $T \to \infty$.*

Considering that $|\mathcal{E}_i| = |\mathcal{N}_L(i)|$ and $|\mathcal{V}_i| = |\mathcal{N}_L(i)+1|$, Corollary 5.1 gives the time complexity result of DD1-CSS as follows.

Corollary 5.3. *The complexity for SU_i to execute DD1-CSS is $\mathcal{O}(T \cdot |\mathcal{N}_L(i)|^3)$.*

5.6.2 Compared with belief propagation algorithms

Following similar procedures as [103–106], we apply BP algorithms [108, Chapter 11] to estimate $p_{X_i}(x_i|\boldsymbol{y})$ (5.10). The developed algorithm, named as BP-CSS, is then compared with DD1-CSS.

5.6.2.1 BP-CSS for estimating marginal distributions

BP-CSS consists of factor initialization and message exchanging. Initially, SU_i constructs a "factor" Θ_i as

$$\Theta_i(\boldsymbol{x}_{\mathcal{V}_i}) = f_{Y|X}(y_i|x_i) \prod_{j \in \mathcal{N}_L(i)} \phi(x_i, x_j), \tag{5.60}$$

reminding that $\mathcal{V}_i$ is defined in (5.56), $\mathcal{N}_L(i)$ is defined in (5.57), ϕ is the potential function defined in (5.8).

After factor initialization, each SU iteratively exchanges messages with its (one-hop) neighbors. Message $\mu_{i\to j}^{(t)}$ sent from SU_i to SU_j ($j \in \mathcal{N}(i)$) at the t-th iteration is a real-valued function (table) with domain $\boldsymbol{x}_{\mathcal{V}_{ij}}$, where $\mathcal{V}_{ij} = \mathcal{V}_i \cap \mathcal{V}_j$. With initialization $\mu_{i\to j}^{(0)}(\cdot) = 1$, message $\mu_{i\to j}^{(t)}$ (calculated based on factor Θ_i and previously received messages) is determined as

$$\mu_{i\to j}^{(t)}(\boldsymbol{x}_{\mathcal{V}_{ij}}) = \sum_{\boldsymbol{x}'_{\mathcal{V}_i}:\boldsymbol{x}_{\mathcal{V}_i}\setminus\boldsymbol{x}_{\mathcal{V}_{ij}}} \Theta_i(\boldsymbol{x}_{\mathcal{V}_i}) \prod_{l\in\mathcal{N}(i)\setminus\{j\}} \mu_{l\to i}^{(t-1)}(\boldsymbol{x}_{\mathcal{V}_{il}}). \tag{5.61}$$

where $\sum_{\boldsymbol{x}'_{\mathcal{V}_i}:\boldsymbol{x}_{\mathcal{V}_i}\setminus\boldsymbol{x}_{\mathcal{V}_{ij}}}[\cdot]$ means to marginalize over variables $\{x_k\}_{k\in\mathcal{V}_i\setminus\mathcal{V}_{ij}}$, whose definition is similar to that of (5.10).

Assume that, at t-th iteration, BP-CSS converges, i.e., $\mu_{i\to j}^{(t)} = \mu_{i\to j}^{(t-1)}$, $\forall i, j$. Then, SU_i gets estimated marginal posterior distribution $\hat{p}_{X_i}(x_i \mid \boldsymbol{y})$ as

$$\hat{p}_{X_i}(x_i \mid \boldsymbol{y}) = \sum_{\boldsymbol{x}'_{\mathcal{V}_i}:\boldsymbol{x}_{\mathcal{V}_i}\setminus\{x_i\}} \Theta_i(\boldsymbol{x}_{\mathcal{V}_i}) \prod_{l\in\mathcal{N}(i)} \mu_{l\to i}^{(t)}(\boldsymbol{x}_{\mathcal{V}_{il}}),$$

from which SU_i decides its spectrum status (see Section 5.3.2.1).

5.6.2.2 DD1-CSS versus BP-CSS

In BP-CSS, messages are real-valued tables, while DD1-CSS exchanges binary decisions, which suggests low communication overhead of DD1-CSS. In addition, DD1-CSS has theoretical guarantee for recovering $\boldsymbol{x}^{\mathrm{MAP}}$ (Corollary 5.2), while BP-CSS do not have guarantee to converge or obtain true marginal distributions [108, Chapter 11]. Furthermore, as stated in Corollary 5.3, each iteration of DD1-CSS is with complexity $\mathcal{O}(|\mathcal{N}_L(i)|^3)$, which is polynomial versus the number of neighboring SUs. In contrast, for BP-CSS, message updating (5.61) requires to manipulate (product-then-sum) the factor $\Theta_i(\boldsymbol{x}_{\mathcal{V}_i})$ (5.60) with incoming messages. Considering that $\Theta_i(\boldsymbol{x}_{\mathcal{V}_i})$ is a table with $2^{|\mathcal{N}_L(i)|+1}$ elements, the complexity for message updating is roughly treated as $\mathcal{O}(2^{|\mathcal{N}_L(i)|})$. Also consider that there are $\mathcal{N}(i)$ messages needed to be updated at each iteration. Therefore, BP's each iteration is of complexity $\mathcal{O}(\mathcal{N}(i)\cdot 2^{|\mathcal{N}_L(i)|})$, which grows exponentially as network density increases.

5.7 Numerical Simulations

This section presents several simulation results. Section 5.7.1 explains the simulation setup and performance metrics. In Section 5.7.2, the choosing of MRF hyperparam-

eter β is discussed. In Section 5.7.3, we demonstrate cooperation gain achieved via MAP-MRF. In Section 5.7.4 and Section 5.7.5, the sensing performance and running complexity of our proposed algorithms are investigated.

5.7.1 Simulation setup

5.7.1.1 Network setup

We set the radius of the disc R as 2 kilometers (km) and PrR r as 1 km. SUs located within the disc follow Poisson point process with density 1.6×10^{-5} node per m^2 (Sections 5.7.2 and 5.7.5 investigate other density parameters). The transmission distance of each SU is 400 meters. Therefore, in average, there are $N = 200$ SUs in the disc, and each SU has 7 neighbors.

5.7.1.2 Signal model

The K-factor $k_H = 10^{-2}$, i.e., -20 dB, for Rician fading is considered. Section 5.7.3 considers other values of k_H. In addition, we set $\alpha = 2$ as propagation factor, $\sigma_S^2 = 1$ as primary signal power, $\sigma_W^2 = 10^{-10}$ as SU sensing noise, and $M = 10$ as sampling length for energy detection.

5.7.1.3 Performance metrics

We measure the error probability P_e (algorithm' decision $\hat{x}$ is different from SU's true spectrum status x) by averaging over N SUs and $\tau_0 = 10^4$ simulation rounds, i.e.,

$$P_e = \frac{1}{\tau_0 N} \sum_{\tau=1}^{\tau_0} \sum_{i=1}^{N} \mathbb{1}(\hat{x}_i^\tau \neq x_i^\tau).$$

Whenever metric P_e is used, we set weight γ of (5.11) as 1. On the other hand, whenever varying γ is considered, we measure performance with detection probability P_d and false alarm probability P_f, which are defined as

$$P_d = \frac{1}{\tau_0} \sum_{\tau=1}^{\tau_0} \frac{1}{N_1^\tau} \sum_{i=1}^{N_1^\tau} \mathbb{1}(\hat{x}_i^\tau = 1, x_i^\tau = 1),$$

and

$$P_f = \frac{1}{\tau_0} \sum_{\tau=1}^{\tau_0} \frac{1}{N_0^\tau} \sum_{i=1}^{N_0^\tau} \mathbb{1}(\hat{x}_i^\tau = 1, x_i^\tau = 0),$$

where $N_1^\tau = \sum_{i=1}^{N} \mathbb{1}(x_i^\tau = 1)$ and $N_0^\tau = \sum_{i=1}^{N} \mathbb{1}(x_i^\tau = 0)$.

5.7.2 Choosing hyperparameter β

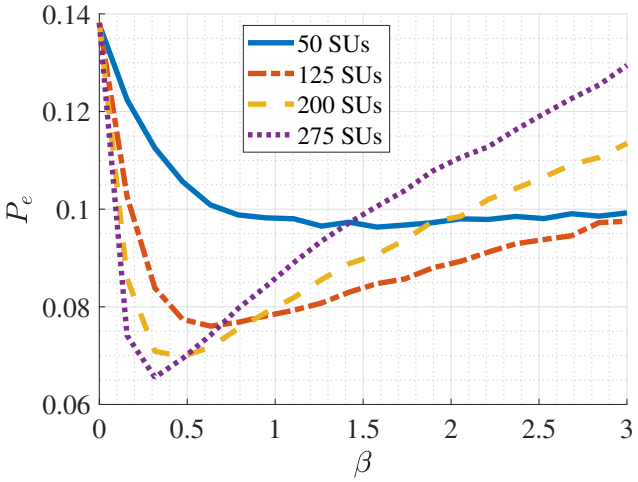


Figure 5.8: P_e for GC-CSS under different β and SU densities

Here, we choose β by evaluating P_e at different values, and select the one achieving the best performance. Fig. 5.8 shows the error probability of the GC-CSS[4] under various β and four secondary network densities, namely, 50, 125, 200 and 275 expected SUs in the disc. It can be seen that the (best achievable) performance improves as the number of cooperating SUs increases. Also note that the best value of β decreases as density increases. It is probably because, when SUs have large number of neighbors, large β causes "over-coupling" among SUs. In following simulations, we tune β to adapt different SU densities via choosing its best value (with a similar search as Fig. 5.8).

5.7.3 Performance gain and loss of MAP-MRF

With metric P_e, we demonstrate performance gain and loss of fusing sensing data via solving the MAP-MRF problem. Specifically, we compare the GC-CSS with Ind-SS, a localization algorithm and a modified version of GC-CSS, called Dist-GC-CSS. These three algorithms are explained as follows.

Ind-SS decides $x_i = 1$ if $f_{Y|X}(y|1) \geq f_{Y|X}(y|0)$, decides $x_i = 0$ otherwise. The localization algorithm is motivated from [119], which exploits SU location information

[4] We choose GC-CSS as a representative for the MAP-MRF framework.

$\{\boldsymbol{l}_i\}_i$, and estimates the PU location $\boldsymbol{l}^*$ as[5]

$$\boldsymbol{l}^* = \arg\max_{\boldsymbol{l}} \left\{ \prod_{i=1}^{N} \int f_{Y|\,D,|\Psi|^2}(y_i \mid d_i(\boldsymbol{l}), z)\, f_{|\Psi|^2}(z)\, \mathrm{d}z \right\},$$

where $d_i(\boldsymbol{l}) = ||\boldsymbol{l} - \boldsymbol{l}_i||_2$ and $f_{Y|\,D,|\Psi|^2}(\cdot|\cdot,\cdot)$ and $f_{|\Psi|^2}(\cdot)$ are defined in (5.2) and (5.1), respectively. Given $\boldsymbol{l}^*$, it decides $\hat{x}_i = 1$ if $d_i(\boldsymbol{l}) \geq r$; decides $\hat{x}_i = 0$ otherwise. Dist-GC-CSS is modified from GC-CSS by further exploiting distances between adjacent SUs. Specifically, for adjacent SU_i and SU_j, the potential function is the same as (5.8) except replacing β with $\beta_{ij} = \left(\frac{1}{d_{ij}}\right)^{0.15}$, where d_{ij} is the distance between SU_i and SU_j.

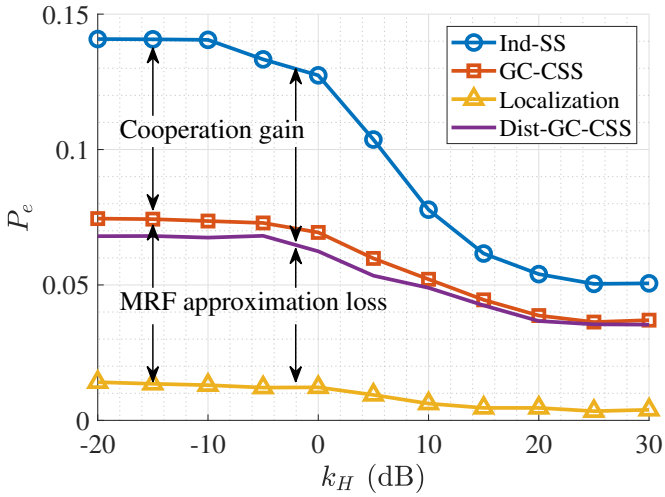


Figure 5.9: P_e for Ind-SS, GC-CSS, localization and Dist-GC-CSS algorithms

Results are shown in Fig. 5.9. It can be seen that, compared with Ind-SS, GC-CSS achieves considerable performance gain, especially under poor channel conditions. However, GC-CSS is inferior to the localization algorithm. It is because the MRF model does not exploit location information, but only pairwise neighboring relationship. Although this approximation causes performance loss, it avoids SU localization (which can be difficult and expensive), and ensures the efficiency and flexibility of solving the CSS problem. In addition, we can further improve the performance by refining the MRF model, as shown by Dist-GC-CSS. Specifically, if an SU is able roughly measure its distance to neighbors (e.g., from received signal strength), we can reduce MRF approximation loss. Another potential refining strategy is to use

[5] We solve this optimization problem via exhaustive search with 2-meter resolution.

higher-order MRFs (by defining multiple-hop potential functions), as they demonstrate better performance than pairwise MRFs in computer visions [120].

5.7.4 Maximization v.s. Marginalization

In this subsection, the performance of GC-CSS, DD-CSS and DD1-CSS is investigated. For DD-CSS, we group SUs as 5 clusters (each cluster has 40 SUs in average). We also consider Ind-CSS and BP-CSS algorithm for comparison.

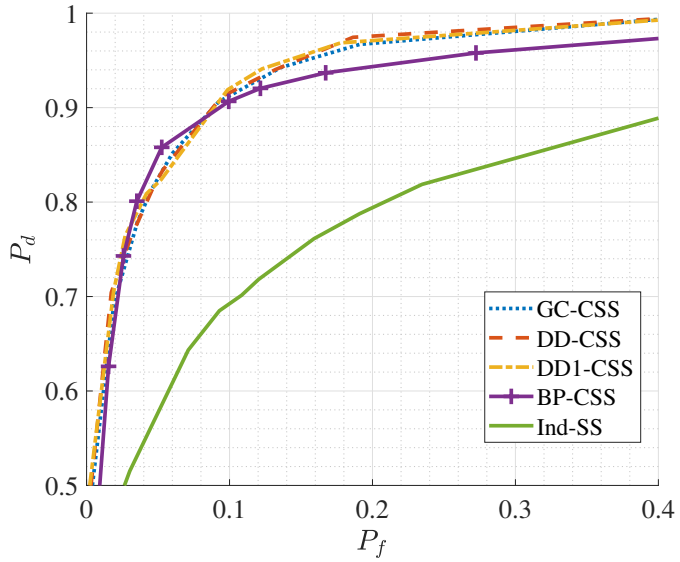


Figure 5.10: ROC for sensing algorithms

Fig. 5.10 shows algorithms' (P_d, P_f) pairs (i.e., a receiver operating curve (ROC)) measured by varying $\mathcal{T}$ within $(0, 1)$ for BP-CSS, and varying γ within $(0, 3)$ for the rest of algorithms. It can be seen that, compared with Ind-SS, all CSS algorithms considerably improve sensing performance. In addition, since both GC-CSS and DD1-CSS have theoretical guarantee of solving the MAP-MRF problem, they should achieve the same performance, which can be confirmed from Fig. 5.10. Furthermore, we observe that DD-CSS perform as well as GC-CSS and DD1-CSS, which implies that, although lack of theoretical guarantee, DD-CSS obtains a MAP solution very probably.

Interestingly, BP-CSS achieves higher detection probability P_d than our proposed algorithms (i.e., GC-CSS, DD-CSS and DD1-CSS) when $P_f \in [0.03, 0.08]$, but shows inferior performance under other choices of P_f. Nevertheless, their performance dif-

ferences are insignificant. Hence, we may conclude that, although CSS based on marginalization behaviors slightly differently from CSS based on MAP-MRF, both of them work well in terms of sensing performance. However, the advantage of the MAP-MRF methodology is the computational efficiency and flexibility, as shown in the next subsection.

5.7.5 Computation complexity

In this subsection, we investigate the computation complexity of GC-CSS, DD-CSS, DD1-CSS and BP-CSS under different secondary network densities. Specifically, we increase expected number of SUs (within the disc) from 10 to 200. To compare computation complexity, we measure algorithms' CPU time. All algorithms are executed serially on a single computer[6]. Since the implementation does not exploit the parallelizability of BP-CSS, DD-CSS and DD1-CSS, for ensuring fair comparisons, we divide measured CPU time by algorithms' "potential parallelizability". Specifically, we define a so-called Time Per Unit (TPU) metric as

$$\text{TPU} = \mathbb{E}\left[\frac{\text{CPU time}}{\#\text{ processing units}}\right],$$

where the number of processing units equals 1 for GC-CSS, equals 5 for DD-CSS, equals the number of SUs for BP-CSS and DD1-CSS.

It can be seen from Fig. 5.11 that, when network size/density is small, GC-CSS has the highest TPU. When network size/density increases, the TPU of BP-CSS increases much faster than the rest of algorithms, and bypasses that of GC-CSS when the expected number of SUs increases to 40. It is because BP-CSS's computational complexity increases exponentially with network density, while min-cut algorithms (embedded in GC-CSS, DD-CSS and DD1-CSS) can solve MAP-MRF problems with polynomial time complexity versus both network size and density (see Theorem 5.1). Also observing that, due to parallelization, DD-CSS enjoys smaller TPU than GC-CSS. It suggests that, rather than directly solving the MAP-MRF problem as GC-CSS, DD-CSS provides computational gain by decomposing the problem and iteratively solving 5 subproblems. Furthermore, comparing DD-CSS with DD1-CSS, we

[6] Algorithms are implemented with Matlab R2017a on a computer with Intel i7-3770 cores and 16 GB RAM. The min-cut problem (5.19) (embedded in GC-CSS, DD-CSS and DD1-CSS) is solved with the Boykov-Kolmogorov algorithm [113].

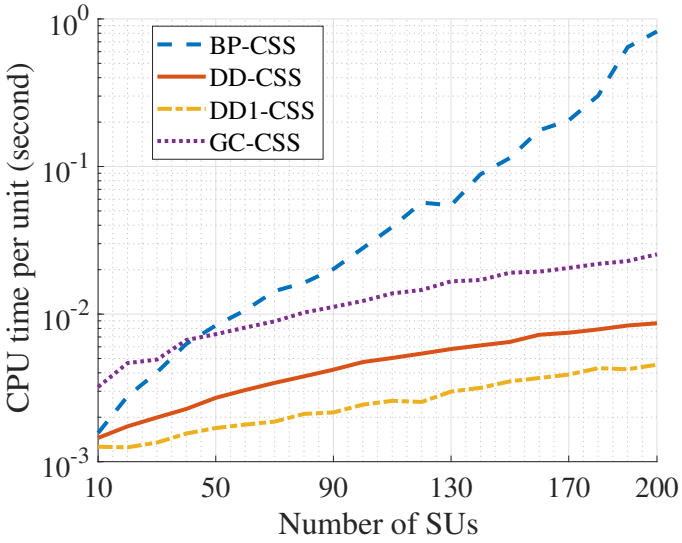


Figure 5.11: CPU time per unit under different number of SUs

see that this computational saving holds, if we further decompose the problem until one subproblem per SU.

5.8 Summary

In this chapter, we studied CSS under heterogeneous spectrum availability. Exploiting MRF a prior, we proposed a CSS methodology, named MAP-MRF, that fuses sensing observations via solving the MAP estimation. Given the MAP-MRF methodology, we developed three CSS algorithms that are, respectively, designed for centralized, cluster-based and distributed secondary networks. Compared with existing methods, our developed algorithms achieve comparable performance, but with less computational complexity and communication overhead.

Chapter 6

Conclusions and Future Research

6.1 Conclusions

This thesis exploits machine learning approaches for intelligent spectrum and energy management in cognitive radio and energy-harvesting wireless systems. Three contributions are made.

Chapter 3 studies the optimal sensing, probing and power control problem for an energy-harvesting cognitive node operating in fading channels. The problem is modeled as a two-stage continuous state MDP, and then simplified via an after-state value function. A reinforcement learning algorithm is proposed, which enables the cognitive node to learn the optimal policy without needing the statistical distributions of the wireless channel and the energy-harvesting process.

Chapter 4 considers the selective transmission problem for energy-harvesting wireless nodes, whose the optimal control problem is modeled as an MDP. The optimal policy is constructed by an after-state value function, which is further shown to be a differentiable and non-decreasing function. In order to solve the after-state value function efficiently, a learning algorithm is proposed, which approximates the function with a monotone neural network, and learns the associated parameters by iterative least-square regression.

Chapter 5 focuses on CSS in presence of spectrum heterogeneity. To exploit SU spatial correlation for sensing decisions, a CSS framework based on MAP-MRF is proposed. By using it, three CSS algorithms are proposed, which are designed for centralized, cluster-based and distributed secondary networks. These proposed algorithms have superior computation efficiency and less communication overhead.

6.2 Future Research

6.2.1 Optimal sensing-probing policy without primary user model

In Chapter 3, the sensing and probing decisions are made by considering a belief value about channel availability. With sensing and probing outcomes, the belief value is periodically updated, via exploiting the primary user's activity model. When the model is not known a prior, this method cannot be directly applied. One intuitive solution is to first estimate the model from historic sensing outcomes, and then, to learn the optimal sensing-probing policy with the methods from Chapter 3. However, this approach may induce high memory overhead for storing sensing and probing outcomes. In addition, we may need to periodically update the model and the policy to adapt to environmental changes. Therefore, learning algorithms incorporating memory can be designed to generate (estimated) best actions based on past information.

6.2.2 Multi-link selective transmission for energy-harvesting sensors

Chapter 4 studies single-link selective transmission by exploiting CSI. This problem can be generalized to multiple receiver links. In this case, the sensor needs to decide the best next hop by sequentially probing the potential receivers. This leads to two basic research questions: 1) What is the best probing order?; and 2) how to avoid the increase in delay and energy consumption? the sensor may need to decide when to stop probing, and whether or not to transmit the packet, given current probed CSI.

6.2.3 Learn MRF model from data

In Chapter 5, the hyperparameter of underlying MRF is heuristically chosen by comparing different values with numerical evaluation. However, this method is not suitable for real-time applications, since it requires repeatedly solving the MAP-MRF problem, and evaluating performance with SUs' true spectrum status. Therefore, it is beneficial to learn and gradually refine the hyperparameter with incrementally augmented sensing database. In addition, since it is not always possible to obtain SUs' true status, the learning process should be able to handle sparsely labeled data.

Bibliography

[1] A. Osseiran, F. Boccardi, V. Braun, K. Kusume, P. Marsch, M. Maternia, O. Queseth, M. Schellmann, H. Schotten, H. Taoka, H. Tullberg, M. A. Uusitalo, B. Timus, and M. Fallgren, "Scenarios for 5G mobile and wireless communications: the vision of the METIS project," *IEEE Commun. Mag.*, vol. 52, no. 5, pp. 26–35, May 2014.

[2] "United states frequency allocations chart," www.ntia.doc.gov/files/ntia/publications/january_2016_spectrum_wall_chart.pdf, 2016, accessed: 2018-6-12.

[3] RCRWirelessNews, "FCC 600 MHz incentive auction closes at nearly $19.8B," www.rcrwireless.com/20170331/policy/fcc-600-mhz-incentive-auction-closes-at-nearly-19-8b-tag2, 2017, accessed: 2018-6-12.

[4] RCRWirelessNews, "5G spectrum allocation in united states," www.everythingrf.com/community/5g-spectrum-allocation-in-united-states, 2018, accessed: 2018-6-12.

[5] "Spectrum for 4G and 5G," www.qualcomm.com/media/documents/files/spectrum-for-4g-and-5g.pdf, 2017, accessed: 2018-6-12.

[6] B. Hanna, P. Jacques, and M. Christophe, *Green communications*. New York, NY, USA: Springer, 2012.

[7] A. S. Andrae and T. Edler, "On global electricity usage of communication technology: Trends to 2030," *Challenges*, vol. 6, no. 1, pp. 117–157, June 2015.

[8] Federal Communications Commission, "Spectrum Policy Task Force," Tech. Rep. ET Docket no. 02-135, Nov. 2002.

[9] J. Mitola, III, "An integrated agent architecture for software defined radio," Ph.D. dissertation, Royal Institute of Technology (KTH), May 2000.

[10] L. Deng and E. Williams, "Measures and trends in energy use of semiconductor manufacturing," in *Proc. of the IEEE International Symposium on Electronics and the Environment*, San Francisco, CA, USA, May 2008, pp. 1–6.

[11] S. Chalasani and J. M. Conrad, "A survey of energy harvesting sources for embedded systems," in *Proc. of the IEEE SoutheastCon 2008*, Huntsville, AL, USA, Apr. 2008, pp. 442–447.

[12] A. R. El-Sayed, K. Tai, M. Biglarbegian, and S. Mahmud, "A survey on recent energy harvesting mechanisms," in *Proc. of the IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, Vancouver, Canada, May 2016, pp. 1–5.

[13] M. L. Ku, W. Li, Y. Chen, and K. J. R. Liu, "Advances in energy harvesting communications: Past, present, and future challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1384–1412, Secondquarter 2016.

[14] K. W. Choi, P. A. Rosyady, L. Ginting, A. A. Aziz, D. Setiawan, and D. I. Kim, "Theory and experiment for wireless-powered sensor networks: How to keep sensors alive," *IEEE Trans. Wireless Commun.*, vol. 17, no. 1, pp. 430–444, Jan. 2018.

[15] C. R. Stevenson, G. Chouinard, Z. Lei, W. Hu, S. J. Shellhammer, and W. Caldwell, "IEEE 802.22: The first cognitive radio wireless regional area network standard," *IEEE Commun. Mag.*, vol. 47, no. 1, pp. 130–138, Jan. 2009.

[16] Y. C. Liang, Y. Zeng, E. C. Y. Peh, and A. T. Hoang, "Sensing-throughput tradeoff for cognitive radio networks," *IEEE Trans. Wireless Commun.*, vol. 7, no. 4, pp. 1326–1337, Apr. 2008.

[17] E. Axell, G. Leus, E. G. Larsson, and H. V. Poor, "Spectrum sensing for cognitive radio: State-of-the-art and recent advances," *IEEE Signal Process. Mag.*, vol. 29, no. 3, pp. 101–116, May 2012.

[18] T. Shu and M. Krunz, "Throughput-efficient sequential channel sensing and probing in cognitive radio networks under sensing errors," in *Proc. of the International Conference on Mobile Computing and Networking (MobiCom)*, Beijing, China, Sept. 2009, pp. 37–48.

[19] H. Jiang, L. Lai, R. Fan, and H. V. Poor, "Optimal selection of channel sensing order in cognitive radio," *IEEE Trans. Wireless Commun.*, vol. 8, no. 1, pp. 297–307, Jan. 2009.

[20] S. S. Tan, J. Zeidler, and B. Rao, "Opportunistic channel-aware spectrum access for cognitive radio networks with interleaved transmission and sensing," *IEEE Trans. Wireless Commun.*, vol. 12, no. 5, pp. 2376–2388, May 2013.

[21] S. Geirhofer, L. Tong, and B. M. Sadler, "Dynamic spectrum access in WLAN channels: Empirical model and its stochastic analysis," in *Proc. of the International Workshop on Technology and Policy for Accessing Spectrum (TAPAS)*, Boston, Massachusetts, USA, Aug. 2006, p. 14.

[22] Q. Zhao, L. Tong, A. Swami, and Y. Chen, "Decentralized cognitive MAC for opportunistic spectrum access in ad hoc networks: A POMDP framework," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 3, pp. 589–600, Apr. 2007.

[23] I. F. Akyildiz, B. F. Lo, and R. Balakrishnan, "Cooperative spectrum sensing in cognitive radio networks: A survey," *Physical Communication*, vol. 4, no. 1, pp. 40–62, Mar. 2011.

[24] G. Ganesan and Y. Li, "Cooperative spectrum sensing in cognitive radio, part ii: multiuser networks," *IEEE Trans. Wireless Commun.*, vol. 6, no. 6, pp. 2214–2222, June 2007.

[25] J. Ma, G. Zhao, and Y. Li, "Soft combination and detection for cooperative spectrum sensing in cognitive radio networks," *IEEE Trans. Wireless Commun.*, vol. 7, no. 11, pp. 4502–4507, Dec. 2008.

[26] S. Chaudhari, J. Lunden, V. Koivunen, and H. V. Poor, "Cooperative sensing with imperfect reporting channels: Hard decisions or soft decisions?" *IEEE Trans. Signal Process.*, vol. 60, no. 1, pp. 18–28, Oct. 2012.

[27] K. B. Letaief and W. Zhang, "Cooperative communications for cognitive radio networks," *Proc. IEEE*, vol. 97, no. 5, pp. 878–893, May 2009.

[28] Q. Zhang, J. Jia, and J. Zhang, "Cooperative relay to improve diversity in cognitive radio networks," *IEEE Commun. Mag.*, vol. 47, no. 2, pp. 111–117, Feb. 2009.

[29] X. Huang, T. Han, and N. Ansari, "On green-energy-powered cognitive radio networks," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 2, pp. 827–842, Jan. 2015.

[30] X. Gong, S. A. Vorobyov, and C. Tellambura, "Joint bandwidth and power allocation with admission control in wireless multi-user networks with and without relaying," *IEEE Trans. Signal Process.*, vol. 59, no. 4, pp. 1801–1813, Apr. 2011.

[31] O. Ozel, K. Tutuncuoglu, J. Yang, S. Ulukus, and A. Yener, "Transmission with energy harvesting nodes in fading wireless channels: Optimal policies," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 8, pp. 1732–1743, Sept. 2011.

[32] M. L. Ku, Y. Chen, and K. J. R. Liu, "Data-driven stochastic models and policies for energy harvesting sensor communications," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 8, pp. 1505–1520, Aug. 2015.

[33] F. Zhang and V. K. N. Lau, "Closed-form delay-optimal power control for energy harvesting wireless system with finite energy storage," *IEEE Trans. Signal Process.*, vol. 62, no. 21, pp. 5706–5715, Nov. 2014.

[34] Z. Wang, V. Aggarwal, and X. Wang, "Power allocation for energy harvesting transmitter with causal information," *IEEE Trans. Commun.*, vol. 62, no. 11, pp. 4080–4093, Nov. 2014.

[35] K. Tutuncuoglu and A. Yener, "Energy harvesting networks with energy cooperation: Procrastinating policies," *IEEE Trans. Commun.*, vol. 63, no. 11, pp. 4525–4538, Nov. 2015.

[36] P. Blasco, D. Gunduz, and M. Dohler, "A learning theoretic approach to energy harvesting communication system optimization," *IEEE Trans. Wireless Commun.*, vol. 12, no. 4, pp. 1872–1882, Apr. 2013.

[37] Z. Wang, X. Wang, and V. Aggarwal, "Transmission with energy harvesting nodes in frequency-selective fading channels," *IEEE Trans. Wireless Commun.*, vol. 15, no. 3, pp. 1642–1656, Mar. 2016.

[38] K. T. Phan, C. Tellambura, S. A. Vorobyov, and R. Fan, "Joint medium access control, routing and energy distribution in multi-hop wireless networks," *IEEE Trans. Wireless Commun.*, vol. 7, no. 12, pp. 5244–5249, Dec. 2008.

[39] Z. Shen, H. Jiang, and Z. Yan, "Fast data collection in linear duty-cycled wireless sensor networks," *IEEE Trans. Veh. Technol.*, vol. 63, no. 4, pp. 1951–1957, May 2014.

[40] J. Carle and D. Simplot-Ryl, "Energy-efficient area monitoring for sensor networks," *IEEE Computer*, vol. 37, no. 2, pp. 40–46, Feb. 2004.

[41] J. Zhang, W. Li, Z. Yin, S. Liu, and X. Guo, "Forest fire detection system based on wireless sensor network," in *Proc. of the IEEE Conference on Industrial Electronics and Applications*, Xi'an, China, May 2009, pp. 520–523.

[42] B. BenGouissem and S. Dridi, "Data centric communication using the wireless control area networks," in *Proc. of the IEEE International Conference on Industrial Technology*, Mumbai, India, Dec. 2006, pp. 1654–1658.

[43] W. Shen, T. Zhang, F. Barac, and M. Gidlund, "PriorityMAC: A priority-enhanced MAC protocol for critical traffic in industrial wireless sensor and actuator networks," *IEEE Trans. Ind. Informat.*, vol. 10, no. 1, pp. 824–835, Feb. 2014.

[44] L. Krishnamachari, D. Estrin, and S. Wicker, "The impact of data aggregation in wireless sensor networks," in *Proc. of the IEEE International Conference on Distributed Computing Systems Workshops*, July 2002, pp. 575–578.

[45] X. Lu, P. Wang, D. Niyato, D. I. Kim, and Z. Han, "Wireless networks with RF energy harvesting: A contemporary survey," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 2, pp. 757–789, Nov. 2015.

[46] X. Zhou, R. Zhang, and C. K. Ho, "Wireless information and power transfer: Architecture design and rate-energy tradeoff," *IEEE Trans. Commun.*, vol. 61, no. 11, pp. 4754–4767, Oct. 2013.

[47] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[48] T. Mitchell, *Machine Learning*, ser. McGraw-Hill International Editions, New York, NY, USA, 1997.

[49] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York, NY, USA: John Wiley & Sons, 1994.

[50] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. Cambridge, MA, USA: MIT press, 1998.

[51] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning*. Cambridge, MA, USA: MIT press, 2016.

[52] M. Sezgin and B. Sankur, "Survey over image thresholding techniques and quantitative performance evaluation," *Journal of Electronic Imaging*, vol. 13, no. 1, pp. 146–166, Jan. 2004.

[53] S. Atapattu, C. Tellambura, and H. Jiang, "Energy detection based cooperative spectrum sensing in cognitive radio networks," *IEEE Trans. Wireless Commun.*, vol. 10, no. 4, pp. 1232–1241, Apr. 2011.

[54] S. Atapattu, C. Tellambura, and H. Jiang, *Energy Detection for Spectrum Sensing in Cognitive Radio*, ser. SpringerBriefs in Computer Science. New York, NY, USA: Springer, 2014.

[55] T. Cui and C. Tellambura, "Joint data detection and channel estimation for OFDM systems," *IEEE Trans. Commun.*, vol. 54, no. 4, pp. 670–679, 2006.

[56] G. Wang, F. Gao, Y.-C. Wu, and C. Tellambura, "Joint CFO and channel estimation for OFDM-based two-way relay networks," *IEEE Trans. Wireless Commun.*, vol. 10, no. 2, pp. 456–465, 2011.

[57] G. Wang, F. Gao, W. Chen, and C. Tellambura, "Channel estimation and training design for two-way relay networks in time-selective fading environments," *IEEE Trans. Wireless Commun.*, vol. 10, no. 8, pp. 2681–2691, 2011.

[58] S. Park and D. Hong, "Optimal spectrum access for energy harvesting cognitive radio networks," *IEEE Trans. Wireless Commun.*, vol. 12, no. 12, pp. 6166–6179, Dec. 2013.

[59] W. Chung, S. Park, S. Lim, and D. Hong, "Spectrum sensing optimization for energy-harvesting cognitive radio systems," *IEEE Trans. Wireless Commun.*, vol. 13, no. 5, pp. 2601–2613, May 2014.

[60] D. T. Hoang, D. Niyato, P. Wang, and D. I. Kim, "Performance optimization for cooperative multiuser cognitive radio networks with RF energy harvesting capability," *IEEE Trans. Wireless Commun.*, vol. 14, no. 7, pp. 3614–3629, July 2015.

[61] Pratibha, K. H. Li, and K. C. Teh, "Dynamic cooperative sensing-access policy for energy-harvesting cognitive radio systems," *IEEE Trans. Veh. Technol.*, vol. 65, no. 12, pp. 10 137–10 141, Dec. 2016.

[62] A. Celik, A. Alsharoa, and A. E. Kamal, "Hybrid energy harvesting-based cooperative spectrum sensing and access in heterogeneous cognitive radio networks," *IEEE Trans. Cogn. Commun. Netw*, vol. 3, no. 1, pp. 37–48, Mar. 2017.

[63] D. Zhang, Z. Chen, M. K. Awad, N. Zhang, H. Zhou, and X. S. Shen, "Utility-optimal resource management and allocation algorithm for energy harvesting cognitive radio sensor networks," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3552–3565, Dec. 2016.

[64] C. Xu, M. Zheng, W. Liang, H. Yu, and Y. C. Liang, "End-to-end throughput maximization for underlay multi-hop cognitive radio networks with rf energy harvesting," *IEEE Trans. Wireless Commun.*, vol. 16, no. 6, pp. 3561–3572, June 2017.

[65] A. Sultan, "Sensing and transmit energy optimization for an energy harvesting cognitive radio," *IEEE Wireless Commun. Lett.*, vol. 1, no. 5, pp. 500–503, Oct. 2012.

[66] Z. Li, B. Liu, J. Si, and F. Zhou, "Optimal spectrum sensing interval in energy-harvesting cognitive radio networks," *IEEE Trans. Cogn. Commun. Netw*, vol. 3, no. 2, pp. 190–200, June 2017.

[67] S. Yin, Z. Qu, and S. Li, "Achievable throughput optimization in energy harvesting cognitive radio systems," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 3, pp. 407–422, Mar. 2015.

[68] Pratibha, K. H. Li, and K. C. Teh, "Optimal spectrum access and energy supply for cognitive radio systems with opportunistic RF energy harvesting," *IEEE Trans. Veh. Technol.*, vol. 66, no. 8, pp. 7114–7122, Aug. 2017.

[69] J. J. Pradha, S. S. Kalamkar, and A. Banerjee, "Energy harvesting cognitive radio with channel-aware sensing strategy," *IEEE Commun. Lett.*, vol. 18, no. 7, pp. 1171–1174, July 2014.

[70] D. Zhang, Z. Chen, J. Ren, N. Zhang, M. K. Awad, H. Zhou, and X. S. Shen, "Energy-harvesting-aided spectrum sensing and data transmission in heterogeneous cognitive radio sensor network," *IEEE Trans. Veh. Technol.*, vol. 66, no. 1, pp. 831–843, Jan. 2017.

[71] A. Goldsmith, S. A. Jafar, I. Maric, and S. Srinivasa, "Breaking spectrum gridlock with cognitive radios: An information theoretic perspective," *Proc. IEEE*, vol. 97, no. 5, pp. 894–914, May 2009.

[72] Y. Chen, Q. Zhao, and A. Swami, "Distributed spectrum sensing and access in cognitive radio networks with energy constraint," *IEEE Trans. Signal Process.*, vol. 57, no. 2, pp. 783–797, Feb. 2009.

[73] S. Geirhofer, L. Tong, and B. M. Sadler, "A measurement-based model for dynamic spectrum access in WLAN channels," in *Proc. of the IEEE Military Communications Conference (MILCOM)*, Washington, DC, USA, Oct. 2006, pp. 1–7.

[74] S. Ulukus, A. Yener, E. Erkip, O. Simeone, M. Zorzi, P. Grover, and K. Huang, "Energy harvesting wireless communications: A review of recent advances," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 3, pp. 360–381, Mar. 2015.

[75] K. Prabuchandran, S. K. Meena, and S. Bhatnagar, "Q-learning based energy management policies for a single sensor node with finite buffer," *IEEE Commun. Lett.*, vol. 2, no. 1, pp. 82–85, Feb. 2013.

[76] H. Kushner and G. G. Yin, *Stochastic Approximation and Recursive Algorithms and Applications*. New York, NY, USA: Springer Science & Business Media, 2003.

[77] W. B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. New York, NY, USA: John Wiley & Sons, 2007.

[78] E. García-Martín, "Energy efficiency in machine learning: A position paper," in *Proc. of the Annual Workshop of the Swedish Artificial Intelligence Society SAIS*, vol. 137. Linköping University Electronic Press, 2017, pp. 68–72.

[79] A. Agarwal, S. Rajput, and A. S. Pandya, "Power management system for embedded RTOS: An object oriented approach," in *Proc. of the Canadian Conference on Electrical and Computer Engineering*, 2006, pp. 2305–2309.

[80] J. Seguro and T. Lambert, "Modern estimation of the parameters of the Weibull wind speed distribution for wind energy analysis," *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 85, no. 1, pp. 75 – 84, Mar. 2000.

[81] D. P. Bertsekas, *Abstract Dynamic Programming*. Belmont, MA, USA: Athena Scientific, 2013.

[82] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-dynamic Programming*. Belmont, MA, USA: Athena Scientific, 1996.

[83] F. K. Shaikh, S. Zeadally, and E. Exposito, "Enabling technologies for green internet of things," *IEEE Syst. J*, vol. 11, no. 2, pp. 983–994, June 2017.

[84] S. Sudevalayam and P. Kulkarni, "Energy harvesting sensor nodes: Survey and implications," *IEEE Commun. Surveys Tuts.*, vol. 13, no. 3, pp. 443–461, Thirdquarter 2011.

[85] R. Arroyo-Valles, A. G. Marques, and J. Cid-Sueiro, "Optimal selective transmission under energy constraints in sensor networks," *IEEE Trans. Mobile Comput.*, vol. 8, no. 11, pp. 1524–1538, Nov. 2009.

[86] R. Arroyo-Valles, A. G. Marques, and J. Cid-Sueiro, "Optimal selective forwarding for energy saving in wireless sensor networks," *IEEE Trans. Wireless Commun.*, vol. 10, no. 1, pp. 164–175, Jan. 2011.

[87] J. Lei, R. Yates, and L. Greenstein, "A generic model for optimizing single-hop transmission policy of replenishable sensors," *IEEE Trans. Wireless Commun.*, vol. 8, no. 2, pp. 547–551, Feb. 2009.

[88] N. Michelusi, K. Stamatiou, and M. Zorzi, "On optimal transmission policies for energy harvesting devices," in *Proc. of the Information Theory and Applications Workshop*, San Diego, CA, USA, Feb. 2012, pp. 249–254.

[89] N. Michelusi, K. Stamatiou, and M. Zorzi, "Transmission policies for energy harvesting sensors with time-correlated energy supply," *IEEE Trans. Commun.*, vol. 61, no. 7, pp. 2988–3001, July 2013.

[90] J. Fernandez-Bes, J. Cid-Sueiro, and A. G. Marques, "An MDP model for censoring in harvesting sensors: Optimal and approximated solutions," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 8, pp. 1717–1729, Aug. 2015.

[91] H. Arslan and G. E. Bottomley, "Channel estimation in narrowband wireless communication systems," *Wireless Communications and Mobile Computing*, vol. 1, no. 2, pp. 201–219, Mar. 2001.

[92] H. Daniels and M. Velikova, "Monotone and partially monotone neural networks," *IEEE Trans. Neural Netw.*, vol. 21, no. 6, pp. 906–917, June 2010.

[93] S. Weber, J. G. Andrews, and N. Jindal, "The effect of fading, channel inversion, and threshold scheduling on ad hoc networks," *IEEE Trans. Inf. Theory*, vol. 53, no. 11, pp. 4127–4149, Oct. 2007.

[94] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, vol. 2, no. 4, pp. 303–314, Feb. 1989.

[95] M. Riedmiller, "Neural fitted Q iteration–first experiences with a data efficient neural reinforcement learning method," in *Proc. of the European Conference on Machine Learning (ECML)*, Porto, Portugal, Oct. 2005, pp. 317–328.

[96] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, Feb. 2015.

[97] S. Wang, H. Liu, P. H. Gomes, and B. Krishnamachari, "Deep reinforcement learning for dynamic multichannel access in wireless networks," *IEEE Trans. Cogn. Commun. Netw*, vol. 4, no. 2, pp. 257–265, June 2018.

[98] R. Munos and C. Szepesvári, "Finite-time bounds for fitted value iteration," *Journal of Machine Learning Research*, vol. 9, pp. 815–857, May 2008.

[99] A. Ghasemi and E. S. Sousa, "Spectrum sensing in cognitive radio networks: requirements, challenges and design trade-offs," *IEEE Commun. Mag.*, vol. 46, no. 4, pp. 32–39, Apr. 2008.

[100] S. M. Mishra, "Maximizing available spectrum for cognitive radios," Ph.D. dissertation, University of California, Berkeley, 2009.

[101] S. A. R. Zaidi, D. C. McLernon, and M. Ghogho, "Quantifying the primary's guard zone under cognitive user's routing and medium access," *IEEE Commun. Lett.*, vol. 16, no. 3, pp. 288–291, Mar. 2012.

[102] G. Ding, J. Wang, Q. Wu, F. Song, and Y. Chen, "Spectrum sensing in opportunity-heterogeneous cognitive sensor networks: How to cooperate?" *IEEE Sensors J.*, vol. 13, no. 11, pp. 4247–4255, May 2013.

[103] H. Li, "Cooperative spectrum sensing via belief propagation in spectrum-heterogeneous cognitive radio systems," in *Proc. of the IEEE Wireless Communications and Networking Conference (WCNC)*, Sydney, NSW, Australia, July 2010, pp. 1–6.

[104] F. Penna, R. Garello, and M. A. Spirito, "Distributed inference of channel occupation probabilities in cognitive networks via message passing," in *Proc. of the IEEE Symposium on New Frontiers in Dynamic Spectrum*, Singapore, 2010, pp. 1–11.

[105] Z. Zhang, Z. Han, H. Li, D. Yang, and C. Pei, "Belief propagation based cooperative compressed spectrum sensing in wideband cognitive radio networks," *IEEE Trans. Wireless Commun.*, vol. 10, no. 9, pp. 3020–3031, Sept. 2011.

[106] Y. Wang, H. Li, and L. Qian, "Belief propagation and quickest detection-based cooperative spectrum sensing in heterogeneous and dynamic environments," *IEEE Trans. Wireless Commun.*, vol. 16, no. 11, pp. 7446–7459, Nov. 2017.

[107] S. Z. Li, *Markov random field models in computer vision.* Berlin, Heidelberg: Springer-Verlag, 1995.

[108] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques.* Cambridge, MA, USA: MIT press, 2009.

[109] D. M. Greig, B. T. Porteous, and A. H. Seheult, "Exact maximum a posteriori estimation for binary images," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 51, no. 2, pp. 271–279, 1989.

[110] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms.* Cambridge, MA, USA: MIT press, 2009.

[111] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network," *Canadian Journal of Mathematics*, vol. 8, no. 3, pp. 399–404, 1956.

[112] E. A. Dinic, "Algorithm for solution of a problem of maximum flow in a network with power estimation," *Soviet Math Doklady*, vol. 11, pp. 1277–1280, 1970.

[113] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 9, pp. 1124–1137, July 2004.

[114] N. Komodakis, N. Paragios, and G. Tziritas, "MRF energy minimization and beyond via dual decomposition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 3, pp. 531–552, May 2011.

[115] N. Komodakis and J. C. Pesquet, "Playing with duality: An overview of recent primal-dual approaches for solving large-scale optimization problems," *IEEE Signal Process. Mag.*, vol. 32, no. 6, pp. 31–54, Nov. 2015.

[116] B. Korte and J. Vygen, *Combinatorial Optimization:Theory and Algorithms*, ser. Algorithms and Combinatorics. New York, NY, USA: Springer, 2006.

[117] A. P. Ruszczyński, *Nonlinear optimization*. Princeton, New Jersey, USA: Princeton university press, 2006.

[118] P. Kohli and P. H. Torr, "Dynamic graph cuts for efficient inference in Markov random fields," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 12, pp. 2079–2088, Nov. 2007.

[119] B. L. Mark and A. O. Nasif, "Estimation of interference-free transmit power for opportunistic spectrum access," in *Proc. of the IEEE Wireless Communications and Networking Conference (WCNC)*, Las Vegas, NV, USA, Mar. 2008, pp. 1679–1684.

[120] N. Komodakis and N. Paragios, "Beyond pairwise energies: Efficient optimization for higher-order MRFs," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPRW)*, Miami, FL, USA, June 2009, pp. 2985–2992.