2019

Enhancing the performance of energy harvesting wireless communications using optimization and machine learning

Ala'eddin A. Masadeh
Iowa State University

Enhancing the performance of energy harvesting wireless communications using optimization and machine learning

by

Ala'eddin Masadeh

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Co-majors: Electrical Engineering; Computer Engineering

Program of Study Committee:
Ahmed E. Kamal, Co-major Professor
Zhengdao Wang, Co-major Professor
Daji Qiao
Lu Ruan
Sang W. Kim

The student author, whose presentation of the scholarship herein was approved by the program of study committee, is solely responsible for the content of this dissertation. The Graduate College will ensure this dissertation is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University

Ames, Iowa

2019

DEDICATION

I would like to dedicate this thesis to my family, especially to my extraordinary parents. All of this was, is and always will be for them.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

AC	Actor-Critic
AF	Amplify and Forward
AI	Artificial Intelligence
CR	Cognitive radio
CRN	Cognitive Radio Network
CSI	Channel State Information
DF	Decode and Forward
DP	Dynamic Programming
EH	Energy Harvesting
EnHANTs	Energy-Harvesting Active Networked Tags
ESAC	Estimator-Selector-Actor-Critic
FCC	Federal Communications Commission
i.i.d	Independent and Identically Distributed
MDP	Markov Decision Process
MLAC	Model Learning Actor-Critic
RMAC	Reference Model Actor-Critic
MPI	Modified Policy Iteration
PAMDP	Parameterized Action Space Markov Decision Processes
PI	Policy Iteration Algorithm
QoS	Quality of Service
RF	Radio Frequency
RL	Reinforcement Learning
SAC	Selector-Actor-Critic
SARSA	State-Action-Reward-State-Action

SINR	Signal-to-Interference-plus-Noise Ratio
TAC	Tuner-Actor-Critic
TD	Temporal-Difference
VI	Value Iteration Algorithm

LIST OF SYMBOLS

$B_{\max}$	The maximum capacity of a battery
B_i	Battery level at time slot i
$B_{\mathrm{x},i}$	Battery level of node x at time slot i
P_i^{Tx}	Transmission power during time slot i
$P_{\mathrm{x},i}^{Tx}$	Transmission power by node x during time slot i
E_i	Harvested energy during time slot i
$E_{\mathrm{x},i}$	Harvested energy by node x during the ith time slot
$d_{(\mathrm{x},\mathrm{y})}$	Distance between node x and node y
H_i	The channel signal gain at time slot i
$H_{(\mathrm{x},\mathrm{y}),i}$	Channel signal gain between the nodes x and y during time slot i
$\tilde{H}_{(\mathrm{x},\mathrm{y}),i}$	The fading coefficient between x and y during time slot i
α_{pl}	Path loss exponent
$y_{\mathrm{x},i}$	The received signal at node x during the time slot i
$x_{\mathrm{x},i}$	The transmitted signal from node x during the time slot i
$n_{\mathrm{x},i}$	The additive Gaussian noise at node x during time slot i
σ_n^2	The noise variance
$\sigma_{n,i}^2$	The noise variance during time slot i
σ	The standard deviation of the normal distribution
μ	The mean of the normal distribution
$\Gamma_{\mathrm{x},i}$	The SINR at node x during time slot i
Γ	The predefined SINR QoS threshold
$\log(\cdot)$	Logarithm function
$\lvert\cdot\rvert$	Absolute value of a function or cardinality of a set
N_0	Noise power density

B	Bandwidth
i	Time step number
T	Final time step of an episode
$\mathcal{S}$	Set of possible states
$\mathcal{A}$	Set of possible actions
S_i	The state at time slot i
A_i	The taken action at time slot i
s	A state
a	An action
a_g	The greedy action at state s, i.e., $a_g = \underset{b}{\operatorname{argmax}}\, Q(s,b), \quad \forall b$ at s
a_l	The l^{th} action
s_j	The j^{th} state
$\mathcal{A}(s)$	The set of actions available at state s
N_s	The total number of all possible states
N_a	The total number of available actions
$\pi(s)$	Action taken in state s under deterministic policy π
$\pi(a\|s,\boldsymbol{\theta})$	Probability selecting action a at state s given parameter vector $\boldsymbol{\theta}$
$\boldsymbol{\theta}$	Parameter vector of target policy
$\boldsymbol{w}$	Vector of weights used to approximate value function
π^*	The optimal policy
$J(\boldsymbol{\theta})$	Performance measure for policy $\pi(a\|s,\boldsymbol{\theta})$
$d^{\pi}(s)$	The steady-state distribution of the underlying MDP using policy π
$E[Z]$	Expectation of a random variable Z
$\Pr(Z=z)$	Probability that a random variable Z takes a value of z
$p(s'\|s,a)$	Probability of transition from state s to state s' under action a
$r(s,a,s')$	The expected reward for state-action-next-state
R_i	Reward at time i
$v_{\pi}(s)$	State-value of s under policy π

$v_{\pi^*}(s)$	State-value of s under the optimal policy
$q_\pi(s,a)$	Action-value of state-action pair (s,a) under policy π
$q_{\pi^*}(s,a)$	Action-value of state-action pair (s,a) under the optimal policy
V	Estimate of state-value function v_π
Q	Estimate of action-value function q_π
$\boldsymbol{\phi}(s)$	vector of features at state s
$\boldsymbol{\phi}(s,a)$	vector of features for (s,a) state-action pair
u	Random disturbance
γ	Discount factor
α, β	Step-size parameters
ζ	The action-value function convergence error in the convergence-based exploration algorithm
τ	The exploration time threshold in the convergence-based exploration algorithm
ϵ	Probability of random action in ϵ-greedy policy
$\mathcal{B}$	The set of battery levels
b_j	The j^{th} battery level
N_B	Number of possible battery levels
$\mathcal{E}_n$	The set of energy levels that can be harvested
e_j	The j^{th} level of energy that can be harvested
N_e	Number of energy levels that can be harvested
$p_{\mathcal{E}_n}(e'\|e)$	The transition probability from harvested energy level e to level e'
$\mathcal{H}$	The set of channel gains
h_j	The j^{th} channel gain
N_H	Number of possible channel gains
$p_{\mathcal{H}}(h'\|h)$	The transition probability from channel gain h to channel gain h'
$\mathcal{P}^{Tx}$	The set of transmission power levels
p_j^{Tx}	The j^{th} power level that can be transmitted
$N_{\mathcal{P}^{Tx}}$	Number of possible transmission power levels

ACKNOWLEDGEMENTS

"After all this time"... "Always".

Firstly, I thank Allah for allowing me to reach this milestone. This journey would have been meaningless without him. He has been my source of strength all throughout. I thank Allah for the extraordinary parents he has given me. I am nothing without them. Secondly, I thank my wonderful major advisers Professor Ahmed E. Kamal and Professor Zhengdao Wang for their support, patience, and guidance in almost every step throughout my thesis. And finally, I thank my friends I have made along the way. They have left wonderful memories for me.

ABSTRACT

The motivation behind this thesis is to provide efficient solutions for energy harvesting communications. Firstly, an energy harvesting underlay cognitive radio relaying network is investigated. In this context, the secondary network is an energy harvesting network. Closed-form expressions are derived for transmission power of secondary source and relay that maximizes the secondary network throughput. Secondly, a practical scenario in terms of information availability about the environment is investigated. We consider a communications system with a source capable of harvesting solar energy. Two cases are considered based on the knowledge availability about the underlying processes. When this knowledge is available, an algorithm using this knowledge is designed to maximize the expected throughput, while reducing the complexity of traditional methods. For the second case, when the knowledge about the underlying processes is unavailable, reinforcement learning is used. Thirdly, a number of learning architectures for reinforcement learning are introduced. They are called selector-actor-critic, tuner-actor-critic, and estimator-selector-actor-critic. The goal of the selector-actor-critic architecture is to increase the speed of learning a suboptimal policy by approximating the most promising action at the current state. The tuner-actor-critic aims at improving the learning process by providing the actor with a more accurate estimation about the value function. Estimator-selector-actor-critic is introduced to support intelligent agents. This architecture mimics rational humans in the way of analyzing available information, and making decisions. Then, a harvesting communications system working in an unknown environment is evaluated when it is supported by the proposed architectures. Fourthly, a realistic energy harvesting communications system is investigated. The state and action spaces of the underlying Markov decision process are continuous. Actor-critic is used to optimize the system performance. The critic uses a neural network to approximate the action-value function. The actor uses policy gradient to optimize the policy's parameters to maximize the throughput.

CHAPTER 1. INTRODUCTION

1.1 Background

Energy harvesting (EH) technology has been considered as an efficient solution that provides sustainable wireless communication systems. EH communications have been introduced to develop communication devices that are able to recharge their batteries using natural sources, and then use this energy for data transmission [1]. The lifetimes of EH devices are determined by their hardware lifetimes, since they are able to recharge their batteries. In addition, the possibility of deploying these devices in hard-to-reach places [2].

To implement efficient EH communications networks, it is important to consider two objectives, which are prolonging the network's lifetime and maximizing its throughput. This can be achieved by optimizing and managing the networks' resources especially in unknown environments [3].

Managing the use of the harvested energy for data transmission is the main challenge facing EH communications [3]. This is because of changing the amount of energy that can be harvested over time [1]. To overcome this challenge and to improve the performance of such systems, it is important to design power allocation policies that consider time-variant EH and the channel fading processes.

The issue of designing power allocation policies for data transmission based on the available information at the EH nodes has been investigated. This available knowledge at EH nodes can be classified into three groups. The first one is the non-causal knowledge. In this case, it is assumed that the EH nodes have perfect non-causal knowledge about the EH and the channel fading processes. This assumption insures an optimal allocation policy. The second group is the statistical knowledge, where the EH and the channel fading processes are stationary random

processes. The last group is the causal knowledge, which is the most realistic one. This means that at every time slot, EH nodes have only current and past information about harvested energy and channel gains [3].

Considering the most realistic scenario introduces a challenge in designing power allocation policies for EH networks. This challenge is to balance energy saving and energy consumption in unknown environments [3]. One of the promising methods to design allocation policies in such environments is reinforcement learning (RL). RL is known as a learning technique to know what to do in an unknown environment in order to maximize a numerical return [4].

1.1.1 Energy Harvesting Wireless Devices

Recently, EH has been considered as one of the promising solutions for sustainable wireless communications. EH technology converts the ambient energy into usable electric energy [5]. Current EH techniques are able to provide limited levels of energy, e.g., an outdoor solar panel can get the benefit of 10 mW/cm^2 solar energy flux with harvesting efficiency taking values between 5% and 30%, depending on the used material [6].

EH wireless devices are characterized by a number of properties, which make them attractive to a number of applications. One of these properties is their ability to provide communications in hard-to-reach or inaccessible areas, where replenishing batteries or recharging them is difficult. In addition, these devices have lifetimes that are limited by the lifetimes of their hardware [2]. Last but not least, They provide green or environment friendly devices that preserve the environment by reducing the pollution [7].

For EH wireless nodes, there are five types of energy that can be converted to electrical energy to power this type of nodes. These types are vibration, ambient light, radio frequency (RF) waves, temperature difference, and human body energy [8].

1.1.2 Cognitive Radio

Demands on wireless services have been increasing, which yields to what is known as spectrum congestion. Spectrum congestion can be defined as the spectrum scarcity, which makes it difficult to accommodate a number of wireless devices at the same time [9].

A number of recent studies have shown that the actual utilization of the licensed spectrum band varies from 15% to 85%, which motivated the Federal Communications Commission (FCC) to propose opportunistic utilization of the licensed band to overcome the spectrum shortage problem [9].

Cognitive radio (CR) can be defined as the technology that enables secondary users to utilize the primary spectrum (i.e. the licensed spectrum) opportunistically. The importance of this technology comes from the possibility of improving the spectrum utilization in a way that allows the next generation mobile devices to utilize the available spectrum efficiently [10].

There are three operating modes for cognitive radio networks (CRN), which are underlay, overlay or interweave [11; 12]. In a CR underlay setup, both primary and secondary users access the spectrum simultaneously. In order to protect the primary Quality-of-Service (QoS), the interference from all secondary nodes should be kept under a certain tolerance limit [13]. The overlay mode is based on cooperation between secondary and primary users. For instance, the secondary user relays the primary user's signal in order to get an opportunity to transmit its data using the primary spectrum [14]. In the interweave mode, the secondary users are allowed to utilize the primary spectrum holes opportunistically when they are unoccupied by the licensed users. In this mode, there is a crucial process that should be taken before utilizing the primary spectrum by the secondary users, which is the spectrum sensing [14]. It is reported by a number of works that imperfect sensing may result in degrading the network performance [15; 16].

1.1.3 Cooperative Relaying

Cooperative communication is characterized by a number of properties that make it an efficient technology in wireless communications. Some of the main properties of this technology can be summarized as its ability to improve the capacity, reduce the transmission powers, extend the coverage area, and combat fading of wireless channels [17; 18].

Using this technology, relay(s) are used to forward data from the source to the destination [17]. For this technology, there is a number of protocols to implement cooperative communications, which include adaptive and fixed relaying [19]. Among the available relaying

protocols, there are two well-known protocols, which are amplify and forward (AF) and decode and forward (DF). In AF, a relay amplifies received signals from the source, and then, transmits them to the destination. On the other hand, in DF relaying, a relay is used to decode the received signal firstly, and then, it re-encodes and transmits the signal to the destination [17].

1.2 Related Works

Before going into the details of our proposed models, some recent related works are reviewed in this chapter.

1.2.1 Energy Harvesting Cognitive Radio with Cooperative Communications

Energy Harvesting Cooperative Communications. Recently, EH has been considered as one of the promising solutions for sustainable wireless communications. Energy harvesting technology converts the ambient energy into usable electric energy [5]. Current EH techniques are able to provide limited levels of energy, e.g., an outdoor solar panel can get the benefit of 10 mW/cm^2 solar energy flux with harvesting efficiency taking values between 5% and 30%, depending on the used material [6]. At the same time, cooperative communication is one of the advanced technologies in wireless communications [20; 19], where the wireless nodes assist each other in delivering their data in order to achieve more reliable communication [21]. Combining EH with cooperative relaying can further achieve energy efficient reliable communication, in [22] the authors examined EH with cooperative communication system and energy transfer property. They assume that both source and relay can harvest energy from ambient environment. In addition, the relay can harvest energy RF signals from the source. The framework objective is to maximize the end-to-end throughput by optimizing transmission powers and energy transfer.

Energy Harvesting Cognitive Radio Networks. Combining EH with CR aims to allocate the spectrum efficiently in a green manner [23; 24; 25; 26]. In a CR underlay setup, both primary and secondary users access the spectrum simultaneously. In order to protect the primary Quality-of-Service (QoS), the interference from all secondary nodes should be kept

under a certain tolerance limit [27]. For instance, the authors in [13] study the EH underlay CRN with cooperation between the secondary and the primary users, where the secondary user, who shares the spectrum owned by the primary, is equipped with EH capability and has finite capacity battery for energy storing. In return, the secondary user transfers portion of its energy to the primary user.

Energy Harvesting Cognitive Radio Networks with Cooperative Relaying. Moreover, cooperative communication can be used in EH-CR networks in order to enhance the system performance, where the performance of the EH-CR combined with cooperative relay outperforms the performance of the direct link communication (i.e., without relays) specially if the distance between the communication terminals is relatively large, the available energy is limited, and the interference constraint is very low.

Combining these three technologies has been investigated [28; 29]. For instance, the authors in [28] studied a cooperative EH with a cognitive overlay system, in which the secondary user utilizes portion of the primary time for its transmission data. In return, acting as a relay for the primary user, the secondary user can help in primary transmission. In their proposed model, two RF EH techniques were used at the relay side. In [29], the authors investigated the problem of observing the secondary users sequentially to be used as cooperative relay to the primary transmitter. They derived an optimal stopping rule that selects a relay from a set of candidates to help in relay transmission, which maximizes the observation efficiency.

1.2.2 Energy Harvesting Communications with Non-Causal Knowledge

EH communications systems with non-causal knowledge have been widely discussed [30; 31; 32; 33; 34; 35]. Management approaches in this case are called offline approaches, where the amounts of harvested energy and their arrival times are known at the beginning of the communication session [36]. Despite the difficulty of considering this assumption in reality, it is used to find the upper bound performance [30].

In [30], the problem of maximizing the throughput of EH single hop communication system with non-causal knowledge is investigated. The authors prove that this problem can be modeled

as the minimization of the time required for transmitting a fixed amount of data. The problem of identifying the offline transmission policy for EH communications system with cooperative relay is discussed in [31]. The goal is to maximize the amount of data received by the destination within a given time interval. In the proposed model, both the transmitter and the relay are EH nodes. The model is investigated under two scenarios, which are half-duplex and full-duplex relaying for communications. In [35], the problem of finding the offline transmission power allocation for EH communications system with multiple half-duplex relays is studied. The problem is formulated as a convex optimization problem to find the optimal power allocation for the goal of maximizing the amount of delivered data by a deadline.

1.2.3 Energy Harvesting Communications with Statistical Knowledge

For modeling more realistic EH communications systems, it is assumed that the EH, and data generation processes are discrete Markov processes with full statistical knowledge [37; 38; 39]. A Markov decision process (MDP) is characterized by its ability to provide a suitable mathematical framework for modeling decision-making problems when the system has processes that follow the Markov property [38]. Due to its efficiency, MDP has been adopted to deal with a number of problems considering EH [37; 38; 39; 40]. In [38], a point-to-point wireless communication system is studied, where the transmitter is able to harvest energy and store it in a rechargeable battery. The goal is to maximize the expected total transmitted data during the activation time of the transmitter. The problem is formulated as a discounted MDP problem. The state space consists of the battery state, the size of the packet to be transmitted, the current channel state, and the amount of energy needed for transmitting this packet successfully. At the beginning of each time slot, the transmitter makes a binary decision, whether to drop or to transmit the packet based on the current conditions. In this work, policy iteration (PI) is employed to solve the problem. In [40], an CRN with a secondary user that is capable of harvesting RF energy is investigated. In this model, the secondary user cannot execute EH and data transmission simultaneously, since it has only one interface. As a result, at the beginning of each time slot, the secondary user has to select either harvesting or transmitting. The mode management problem is formulated as an MDP. The primary channel is modeled

as a three-state Markov chain, these states are occupied, idle with bad quality, and idle with good quality. The state space of the modeled MDP is a combination of the primary channel states and the secondary user energy levels. The action space consists of two actions, which are to harvest or to transmit. Value iteration (VI) is used to find the optimal policy, and the performance is compared with the greedy policy.

While traditional methods such as VI are able to find the optimal policy for MDP problems [41; 42], the complexity to find optimal solution grows as the number of states and actions increases. The complexity of finding the optimal solution using VI is $\mathcal{O}(|\mathcal{A}|.|\mathcal{S}|^2)$, where $\mathcal{A}$ is the set of actions, $\mathcal{S}$ is the set of states for the problem [43]. This has encouraged finding alternative methods for solving MDP problems, especially in the case of having large numbers of states and actions [37; 40; 44; 45]. The authors in [37] consider a network of objects equipped with energy-harvesting active networked tags (EnHANTs). The goal is to design an optimal transmission strategy for the EnHANTs to adapt to changes in the amounts of harvested energy and the identification request. The problem is formulated as an MDP. Modified policy iteration (MPI) method [46] is employed to solve the problem and to overcome the complixity of exhaustive search. In [45], the idea of using a mobile energy gateway is investigated, which has the capability of receiving energy from a fixed charging facility, as well as transferring energy to other users. The goal is to maximize the utility by taking the optimal action of energy charging/transferring. The problem is formulated as an MDP. The authors prove that there is a threshold structure of the optimal policy with respect to the system states, which helps in obtaining the optimal policy especially for MDPs with large numbers of states. The goal of determining these thresholds is to select immediate optimal actions based on the current state instead of using the traditional methods such as VI.

1.2.4 Energy Harvesting Communications in Unknown Environments

In the previous two frameworks, a priori knowledge, either deterministic or statistical, about the EH process is required. However, this knowledge might be unavailable, in which case reinforcement learning (RL) can be used to improve the performance of such systems [36]. RL enables an autonomous agent to optimize its policy in an unknown environment [4; 47].

In [3; 36], the problem of optimizing EH communications systems is investigated using RL. In this context, at any time, the EH nodes have only current local knowledge of the EH process. The authors aim to find a power allocation policy that maximizes the throughput. In these two works, the RL algorithm, which is known as the state-action-reward-state-action (SARSA), is used to evaluate the taken actions. On the other hand, the ϵ-greedy exploration algorithm is used to balance between exploring and exploiting available actions.

In [38], a point-to-point communications system is investigated. The transmitter is capable of harvesting energy and storing it in a rechargeable battery. The energy and data arrivals are formulated as Markov processes. In this work, the authors use Q-learning to find the optimal transmission policy when the system does not have a priori information about the Markov processes governing the system. They use the ϵ-greedy exploration algorithm to balance between exploration and exploitation.

1.2.5 Exploration and Exploitation in Reinforcement Learning

RL enables an autonomous agent to optimize its policy to maximize its total expected reward. Here, the agent is placed in an unknown environment, and learns by trial and error [48].

A deterministic policy π can be defined as a function specifying the action $\pi(s)$ that is taken by the agent when it is in state s. At any time, there is at least one action for each state that has highest estimated value, this action is called a greedy action. If the greedy action is selected, this mode is called exploitation, where the agent exploits its current knowledge to determine the current best action (i.e. greedy action), and then use it. On the other hand, when a nongreedy action is selected, this mode is called exploration. Exploration enables the agent to improve its estimates about the nongreedy actions, and discover which of them has higher value than the greedy action [4].

Exploitation should be used when the available time is limited, in which exploring new actions may degrade the performance. If this short time is used for exploration, it may result in exploring unfavorable actions, hence wasting the opportunity of exploiting a current greedy action. Meanwhile, when the available time is abundant and there is ample time to explore

all available actions, then exploration may result in discovering better actions, and exploit them in the remaining time. Balancing between exploration and exploitation is a crucial factor that affects the cumulative rewards. This introduces the importance of balancing between exploration and exploitation, which is one of the main challenges that face RL, and is known as the exploration-exploitation dilemma [4; 49].

Boltzmann and ϵ-greedy exploration algorithms are considered as the most popular exploration algorithms [50], where they are intensively used in the literature [49; 51; 52; 53; 54; 55; 47]. These methods depend on random action selection to learn about new actions [50]. In ϵ-greedy, the agent selects a new action from uniformly distributed actions with probability ϵ, while the greedy action is selected with probability $1 - \epsilon$ [53]. On the other hand, Boltzmann or softmax exploration uses Boltzmann distribution to assign selection probability to actions [49].

Developing new and efficient exploration algorithms is an active area of research [56], where the goal is to enhance RL algorithms, and make them applicable for real applications. Due to these reasons, this topic has been widely investigated [56; 57; 58]. In [56], a counting-based exploration algorithm is designed. The count for state-action visitation $count(s, a)$ is added to the Boltzmann equation to control the exploration. Each time an action a at state s is selected, the $count(s, a)$ increases by one. The main idea of adding this parameter is for balancing between exploration and exploitation and improving the performance.

In [57], the authors proposed the idea of adapting existing exploration algorithms, and combining them with each other. This work studies structured parameterized action space Markov decision processes (PAMDP), where there is a set of discrete actions, each action is parameterized with continuous parameters. For the goal of addressing the exploration of both discrete actions and the continuous parameters, it is proposed to use Boltzmann equation for discrete exploration, and Gaussian exploration to explore the continuous parameters.

1.2.6 Model-free and Model-based Reinforcement Learning

RL methods are categorized into two classes, which are model-free learning and model-based learning. Model-free learning updates the value function after interacting with an environment

without learning the underlying model. On the other hand, model-based learning estimates the dynamics of an environment firstly, and then, dynamic programming (DP) is used to optimize the policy [59; 60].

Each learning class has its own advantages, and suffers from a number of weaknesses. Model-based learning is characterized by its efficiency in learning [61], but at the same time, it struggles in complex problems [62]. On the other hand, model-free learning has strong convergence guarantees under certain situations [4], but the value functions change slowly over time [63], especially, when the learning rate is small.

Merging methods from the two learning classes aims at implementing intelligent agents, and to overcome the mutual weaknesses of these two classes. Combining methods from both learning classes has been discussed in many works [64; 65; 66; 67; 68; 69; 70]. In [64], a method called model-guided exploration was presented. This method integrated a learned model into an off-policy learning such as the Q-learning. The learned model is used to generate good trajectories using trajectory optimization, and then, these trajectories are mixed with on-policy experience to learn the agent. The improvement of this method is quite small even when the learned model is the true model. This returns to the reason of using two completely different policies for learning. This also can be explained by the need to learn bad actions too, so that the agent can distinguish between bad and good actions.

To overcome the weaknesses of the model-guided exploration algorithm, another algorithm called imagination rollout was designed [64]. It was proposed for applications that need large amounts of experience, or when undesirable actions are expensive. In this approach, synthetic samples are generated from the learned model that are called the imagination rollouts. These rollouts, the on-policy samples, and the optimized trajectories from the learned model are used with various mixing coefficients to evaluate each experiment. In each experiment, additional on-policy synthetic rollouts are generated from each visited state, and the model is refitted.

In [66], an algorithm called approximate model-assisted neural fitted Q-iteration was proposed. Using this algorithm, virtual trajectories are generated from a learned model to be used for updating the Q function. This work mainly aimed at reducing the amount of real trajectories required to learn a good policy.

Actor-critic (AC) is a model-free RL method, where the actor learns the policy that is modeled by a parameterized distribution, and the critic learns the value function. The actor optimizes the policy's parameters to maximize the return, while the critic evaluates the performance of the policy. In [65], the framework of human-machine nonverbal communication was discussed. The goal was to enable machines to understand people intention from their behavior. AC with model-based learning were used. The learned dynamics of the underlying model are used to control over the temporal difference error (TD), and the actor uses the TD to optimize the policy for exploring different actions.

In [67], two learning algorithms were designed. The first one is called model learning actor-critic (MLAC), while the second one is called reference model actor-critic (RMAC). The MLAC is an algorithm that combines AC with model-based learning. In this algorithm, the gradient of the approximated state-value function $\hat{V}(s)$ with respect to the state s, and the gradient of the approximated model $s' = f(s, a)$ with respect to the action a are calculated. The actor is updated by calculating the gradient of $\hat{V}(s)$ with respect to a using the chain rule and the previously mentioned two gradients. However, using RMAC, two functions are learned. The first function is the underlying model. The second one is the reference model $s' = R(s)$, which maps state s to the desired next state s' with the highest possible value. Then, using the inverse of the approximated underlying model, the desired action can be found. The integrated paradigm of the reference model and the approximated underlying model serves as an actor, which maps states to actions.

1.2.7 Off-policy Reinforcement Learning

One of the promising methods for developing data efficient RL is off-policy RL. It does not learn from the policy being followed like on-policy methods, it utilizes and learns from data generated from past interactions with the environment [71]. Off-policy learning has been investigated in many works [71; 72; 73; 74].

Q-learning is considered as the most well known off-policy RL method [72]. It enables agents to act optimally in Markovian environments. This method evaluates an action at a state using its current value, the received immediate reward resulting from this action, and the value of the

expected best action at the next state. This expected best action at the next state is selected independently of the currently executed policy, which is the reason for classifying this method as an off-policy learning method.

Combining off-policy learning with AC was studied in [73]. As mentioned, it is the first AC algorithm that can be applied off-policy, where a target policy is learned while following and getting data from another behavior policy. A stream of data (a sequence of states, rewards, and actions) is generated according to a fixed behavior policy. Using this algorithm, the critic learns off-policy estimates of the value function. Then, these estimates are used by the actor for updating the weights of the target policy, and so on.

Using off-policy data to estimate the policy gradient accurately for a target policy was investigated in [71]. A method called behavior policy gradient [75] is used to generate a behavior policy with low variance estimates from a target policy. Trajectories generated from the behavior policy are used to estimate the direction of the policy gradient of the target policy.

1.3 Thesis Scope and Contributions

The thesis aims at providing energy efficient planning for wireless communications and developing algorithms to achieve this goal. The main contributions are summarized as follows:

1. In Chapter 3, the problem of optimizing the transmission power for underlay CR networking with cooperative relaying and EH is investigated. The contributions can be summarized as follows:

    - Formulating an offline optimization problem aims at maximizing the sum rate of EH source and relay in the secondary network while satisfying the primary QoS.
    - The optimization problem is a non-convex problem. The problem is transformed to an equivalent convex form using change of variables.
    - The projected subgradient method is used to find the power allocated to the secondary network.
    - Finally, comparisons are made between the proposed system and other conventional scenarios.

2. Chapter 4 investigates the performance of a point-to-point EH communications system in uncertain environments. Two scenarios are considered. The first one assumes that the EH node has only statistical knowledge about the harvested energy and channel gains. In the second scenario, the EH node works in an unknown environment. The contributions of this chapter can be summarized as follows

    - The decision making problem for the considered EH communications is formulated as an optimization problem.
    - Due to the unavailability of non-causal knowledge about the channel gain and energy arrival processes, the problem is reformulated as a discrete state and action MDP.
    - An algorithm was designed to find a suboptimal power allocation policy when the statistical knowledge about the environment is available. This algorithm is called "Look-ahead Algorithm for EH Communications".
    - RL is used when the EH nodes have only causal knowledge about the environment.
    - A new exploration algorithm for RL called "convergence-based algorithm" is introduced.
    - The introduced algorithms were evaluated by comparing their performance with other algorithms.

3. In Chapter 5, a number of proposed architectures for RL are presented. These architectures are called tuner-actor-critic (TAC), selector-actor-critic (SAC), and estimator-selector-actor-critic (ESAC). The main contributions are:

    - A number of elements are added to the conventional actor-critic (AC) model to implement the introduced architectures, where the AC architecture consists of an actor and a critic. The actor optimizes a stochastic parameterized policy to maximize a performance measure, while the critic estimates a value function and criticises the policy optimized by the actor.

- In SAC, a selector determines the action with the best value at the current state. Then, this action is used by the actor to increase the speed of learning a suboptimal policy.
- In TAC, the newly added components to the conventional AC are a model learner and a tuner. The model learner approximates the dynamics of the underlying model. The tuner uses the approximated value function by the critic and the learned underlying model to improve the learning process.
- ESAC combines the advantages of TAC, and SAC to provide an architecture for intelligent agents. This architecture mimics rational humans in the way of analyzing the available information to make decisions.
- A point-to-point EH communications system working in an uncertain and unknown environment is investigated. The agent-environment interaction is modeled by a discrete state and action MDP. The system performance is evaluated when the considered system is supported by AC, SAC, TAC, and ESAC.

4. Finally, Chapter 6 investigates the performance of an EH point-to-point communications system working in unknown and uncertain environments, when the state and action spaces are continuous. The main contributions are:
   - The agent-environment interaction is modeled by a continuous state and action MDP.
   - A neural network is implemented to approximate the action-value function.
   - A stochastic parameterized policy is used and modeled by a normal distribution with parameterized standard deviation and mean.
   - Policy gradient is used to optimize the parameterized mean and standard deviation to maximize the system throughput.
   - The performance of the considered EH communications system supported by AC are evaluated.

1.4 Thesis Organization

The rest of this thesis is organized as follows:

Chapter 2 includes a brief background on main concepts of MDP and RL.

Chapter 3 investigates the problem of power allocation for CR networking with cooperative relaying and EH.

Chapter 4 presents two algorithms supporting EH communications systems working in uncertain environments. The first algorithm is used when the statistical knowledge about the underlying model is known, while the second one supports systems working in unknown environments.

Chapter 5 presents learning architectures for RL. These architectures are TAC, SAC, and ESAC. This chapter also investigates the problem of optimizing the transmission power for EH communications system in unknown and uncertain environments, using the proposed RL architectures, when the state and action spaces are discrete.

Chapter 6 investigates the problem of learning a suboptimal power allocation policy for EH communications system in unknown and uncertain environments, when agent-environment interaction is modeled by an MDP with continuous state and action spaces.

Chapter 7 concludes this work and outline its main contributions. Then, some potential open problems and possible future works are presented.

CHAPTER 2. MARKOV DECISION PROCESS AND REINFORCEMENT LEARNING

Before presenting a detailed discussion on the main contributions of the thesis, this chapter provides a brief background on main concepts of Markov decision process (MDP) and reinforcement learning (RL). It is not a comprehensive tutorial, but includes essential concepts needed to understand the main contributions presented in the thesis. MDP is characterized by its ability to provide a framework for decision making problems, where outcomes are partly random and partly under control. When the complete and perfect model of an MDP is unavailable, RL is a good choice that can be used to solve such problems [4].

2.1 Markov Decision Process

MDP is defined as a stochastic control process, each time, the environment is at a state $S_t \in \mathcal{S}$, and the agent (decision maker) responses by an action $A_t \in \mathcal{A}$. At the next time slot, the environment responses by moving into a new random state S_{t+1}, and gives the agent a corresponding reward $R_{t+1} \in \mathcal{R}$. Figure 2.1 shows the interaction between the agent and the environment [59].

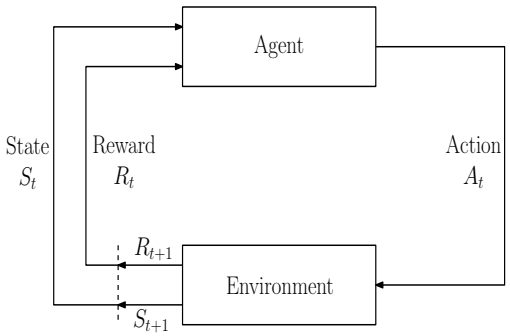

Figure 2.1: The agent-environment interaction in MDP.

In finite MDPs, the sets of states $\mathcal{S}$, actions $\mathcal{A}$, and rewards $\mathcal{R}$ have finite numbers of elements. R_t and S_t are random variables (RVs) with discrete probability distributions that depend on the previous state and action only. The mathematical model of finite MDPs can be defined using the following elements [59]:

1. A set of states $\mathcal{S}$.
2. A set of actions $\mathcal{A}$.
3. The state-transition probabilities, where $p(s'|s,a)$ is the probability of reaching state $s' \in \mathcal{S}$ given that action $a \in \mathcal{A}$ is taken at state $s \in \mathcal{S}$.
4. The expected reward for state-action-next-state $r(s,a,s')$.
5. The discount factor $\gamma \in [0,1]$ that determines the weight of the immediate reward compared to future rewards.

Figure 2.2 shows a finite MDP with two states, $\mathcal{S} = \{\text{Tired}, \text{Active}\}$. At each state, the decision maker can decide whether to sleep or work. The action set is $\mathcal{A} = \{\text{Sleep}, \text{Work}\}$. Working in the tired state leaves the agent at the tired state with probability 1 and results in a reward of 0.7. Sleeping in the tired state moves the agent to the active state with probability 1 and 0 reward. On the other hand, if the agent decides to sleep in the active state, it will stay active with probability 1 and get a 0 reward. Deciding to work in the active state results in a reward of 5, leaves the agent in the active state with probability 0.8, and moves it to the tired state with probability 0.2.

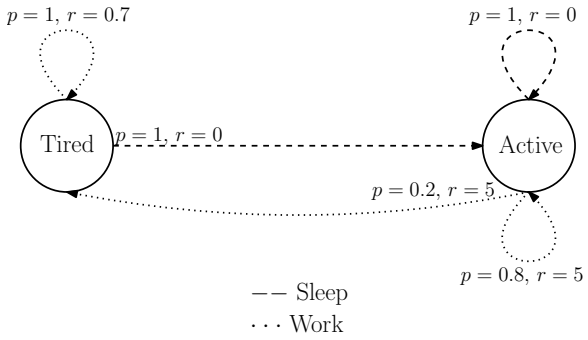

Figure 2.2: Markov decision process.

2.2 Problem Formulation of Decision Making in Uncertain Environments

According to [76], a general problem formulation of decision under stochastic uncertainty is described by the system's state evolution function, a random disturbance depending on the context, a policy π used to control actions' selection, and an objective function.

Given an action a selected at state s, the state evolution formula is given by

$$s' = f(s, a, u) \tag{2.1}$$

where s' is the next state, $u \in \mathcal{U}$ is a random disturbance, $\mathcal{U}$ is the disturbance set, and f is the function that describes the mechanism of evolving the states.

The random disturbance u is characterized by a probability distribution $\Pr(u|s, a)$ that may depend on the current state-action pair. The policy π controls selecting actions at the available states. A deterministic policy π maps states into a particular action at each state, $\pi(\cdot) : s \rightarrow a, \forall s$. A parameterized stochastic policy $\pi(a|s, \boldsymbol{\theta})$ selects each action a with a certain probability given a state s and a policy's parameter vector $\boldsymbol{\theta}$ [59]. $\pi(a|s, \boldsymbol{\theta})$ is given by

$$\pi(a|s, \boldsymbol{\theta}) = \Pr(a|s, \boldsymbol{\theta}) \tag{2.2}$$

where $\Pr(\cdot)$ is a probability distribution describing the probabilities of selecting actions.

Due to the presence of the disturbance u, the objective function J is formulated as the expected cumulative cost over a period of time. An optimal policy π^* is the policy that minimizes J; that is,

$$J^* = \min_{\pi \in \Pi} J \tag{2.3}$$

where Π is the set of all admissible policies.

2.3 Value Functions

To evaluate different policies, value functions (state-value function $v_\pi(s)$ and action-value function $q_\pi(s, a)$) can be used. The state-value function is the expected cumulative reward starting from state s and then following policy π thereafter.

$$v_\pi(s) = \mathbb{E}_\pi \left[\sum_{i=0}^{\infty} \gamma^i R_{t+i+1} \,\middle|\, S_t = s \right] \tag{2.4}$$

The action-value function of a state-action pair (s, a) is defined as the expected cumulative reward starting from state s with action a, and then following policy π successively [4].

$$q_\pi(s,a) = \mathbb{E}_\pi \left[\sum_{i=0}^{\infty} \gamma^i R_{t+i+1} \,\middle|\, S_t = s, A_t = a \right] \tag{2.5}$$

The optimal policy π^* has expected cumulative reward that is better than or equal to any other policy π in all states (i.e., $v_{\pi^*}(s) \geq v_\pi(s), \quad \forall s \in \mathcal{S}$) [38], where

$$v_{\pi^*}(s) = \max_\pi v_\pi(s), \qquad \forall s \in \mathcal{S} \tag{2.6}$$

The optimal action-value function is given by

$$q_{\pi^*}(s,a) = \max_\pi q_\pi(s,a), \qquad \forall s \in \mathcal{S}, \forall a \in \mathcal{A} \tag{2.7}$$

From (2.6), (2.7)

$$v_{\pi^*}(s) = \max_a q_{\pi^*}(s,a), \qquad \forall s \in \mathcal{S} \tag{2.8}$$

2.4 Reinforcement Learning

Machine Learning types are mainly divided into three categories based on their purposes. These categories are supervised learning, unsupervised learning, and RL [77]. In supervised learning, the algorithms are provided by a set of features with their correct labels. Once the mapping of provided features to their labels is learned, it is used to map unseen features to their labels. On the other hand, unsupervised learning aims to extract meaningful representations for data without using labels. In RL, there is an agent interacts with an environment. This interaction results a feedback signal, which is used to modify the agent's interaction and improve its performance [78; 79].

The main point that characterises RL from other types of learning is that it uses training information for evaluating the taken action. On the other hand, the other types use these information for instructing by giving correct actions [4]. RL mainly aims at enabling an autonomous agent to optimize its policy to maximize its total expected reward. Here, the agent is placed in an unknown environment, and learns by trial and error [48].

Exploration and exploitation are fundamental concepts in RL, which are used to optimizes the agent's policy. At any time, there is at least one action for each state with the highest

estimated value, this action is called a greedy action. If the greedy action is selected, this mode is called exploitation, where the agent exploits its current knowledge to determine the current best action (i.e. greedy action). On the other hand, when a nongreedy action is selected, this mode is called exploration. Using exploration, the agent can improve its estimates about nongreedy actions, and discover actions with higher values than the greedy action [4]. Balancing between exploration and exploitation is an important factor affecting the cumulative rewards. Balancing is one of the main challenges facing RL, and it is known as the exploration-exploitation dilemma [4; 49].

Methods used for learning an optimal policy in RL can be categorized into two main classes, which are value-based RL methods, and policy gradient RL methods. Value-based RL is defined as methods that learn the values of actions, and then, select actions according to the estimated actions' values (i.e., policies are extracted from the estimated actions' values). Value-based methods use a sequence of policy evaluation and policy improvement cycles. Policy evaluation is used to estimate a value function for the agent's current policy, while policy improvement is used to improve the policy based on the new estimated value function [59]. Temporal difference (TD) learning is one of the well known value-based reinforcement learning methods. The idea of the TD methods is to estimate a value function from the difference between temporally successive estimates. State-action-reward-state-action (SARSA) is an example of TD prediction methods, which estimates values of state-action pairs according to

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r(s,a,s') + \gamma \, Q(s',a') - Q(s,a) \right) \tag{2.9}$$

where α is the learning rate, $Q(s,a)$ is the estimate of the current state-action pair (s,a), $Q(s',a')$ is the estimate of the next state-action pair (s',a'), $r(s,a,s')$ is the reward resulting from taking action a at state s, and $\delta = (r(s,a,s') + \gamma \, Q(s',a') - Q(s,a))$ is the TD error, which is the difference between the target $(r(s,a,s') + \gamma \, Q(s',a'))$ and the current prediction $Q(s,a)$.

Policy gradient RL is defined as methods learning a parameterized policy, which is able to select actions without consulting a value function. Using this type of learning, value functions may be used to learn the policy's parameters, but they are not needed for actions selection [59]. Policy gradient methods are characterized by a number advantages, which are summarized as

follows. The first one is the ability to learn mixed strategies, which are balanced stochastically. The second one is their convergence properties, which are better than those of value-based methods. They are able to converge to at least a local optimal policy. The third advantage is their capability of learning in problems with continuous action spaces [80]. One of the well known policy gradient learning algorithms is the REINFORCE [81]. This algorithm uses a differentiable policy $\pi(a|s, \boldsymbol{\theta})$ parameterised by a vector of adjustable weights $\boldsymbol{\theta}$. Stochastic gradient ascent is used to optimize $\boldsymbol{\theta}$ to maximize a policy performance measure.

Actor-critic (AC) are learning algorithms combining value-based and policy gradient RL methods. AC algorithms mainly consist of an actor and a critic. The actor estimates a value function, while the critic optimizes the policy's parameters.

CHAPTER 3. COGNITIVE RADIO NETWORKING WITH ENERGY HARVESTING AND COOPERATIVE RELAYING

In this chapter, an underlay cognitive radio energy harvesting (CR-EH) system assisted by a decode and forward (DF) relay is considered, where the secondary users can access the primary user frequency band by exploiting the allowance of its signal-to-interference-plus-noise ratio (SINR) constraints. Moreover, both the secondary source and relay are considered as buffer-aided nodes that can buffer infinite data and store finite energy. The main goal of this work is to derive the optimal power policy that maximizes the number of bits received by the secondary destination. Finally, the performance of the proposed scheme is analyzed to verify our findings, and compared with other schemes to check its validity and efficiency.

The remainder of this chapter is organized as follows. Section 3.1 describes the proposed EH-CR system. The problem formulation is given in Section 3.2. Then, the proposed solution is discussed in Section 3.3. Numerical simulation results are presented in Section 3.4. Finally, this chapter is concluded in Section 3.5.

3.1 System Model

We consider a cooperative CRN with EH consisting of primary and secondary networks, as illustrated in Figure 3.1. The primary network consists of primary source and destination nodes, denoted by PS and PD, respectively. The secondary network is a two-hop relay network consisting of a source, a relay, and a destination node denoted by SS, SR and SD, respectively. SS and SD are far away from each other, i.e., they are not in the coverage communication range of each other. Each of the SS and SR is equipped with infinite data queue to buffer received data and finite battery to store harvested energy. It is assumed that there are always data

packets in the data queue of the SS to be delivered to the SD, which causes depletion of the energy of the SS. Data packets are sent from SS to SR and then are transmitted from the SR to the SD, which causes depletion of the SR's energy. The SR is a full-duplex node, which can transmit and receive data at the same time. In this model, energy consumption is considered only due to data transmission, and it does not take into account any other energy consumption, such as processing, circuitry, etc.

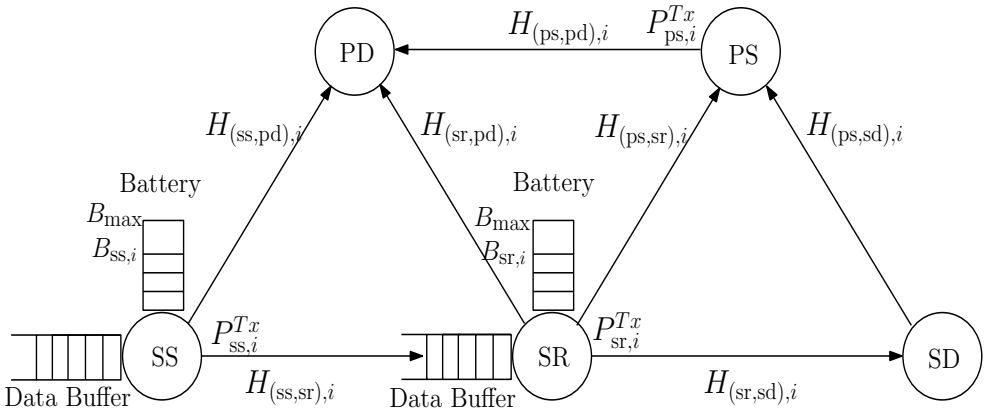

Figure 3.1: Cooperative underlay CR system with EH.

We consider a time slotted system with T equal length time slots. In the secondary network, a DF relaying protocol is used, where the SR can decode the SS signal before broadcasting it to the SD. It is assumed that updating the data queue and energy storage of the SR is delayed by one time slot with respect to the SS. Because of this delay, the efficiency of data transfer from the SS to SD is affected slightly. We assume that T is large so that the slight loss of inefficiency can be neglected.

Let the maximum capacity of the batteries be $B_{\max}$. $B_{\mathrm{ss},i}$ and $B_{\mathrm{sr},i}$ represent the battery levels of both SS and SR, respectively, at time slot i. Each time slot is divided into two equal sub-slots for both transmitting and harvesting. Figure 3.2 illustrates the slotted system model for both SS and SR. This model is designed such that the EH nodes first transmit their signals and then harvest energy, where the transmit power in the current slot depends only on the previous battery level. The batteries are assumed to be ideal, which means that there is no loss during retrieving or storing energy.

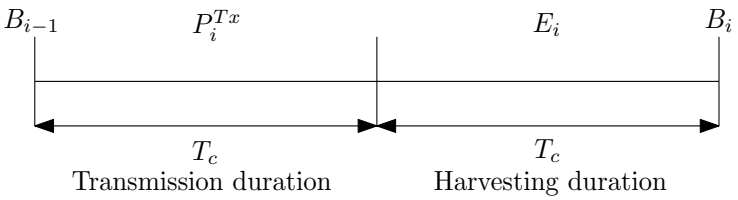

Figure 3.2: Slotted system model for EH nodes.

The channels are Rayleigh fading channels. It is also assumed that we have full channel state information (CSI), and the channels are stationary within every time slot, i.e., $H_{(\mathrm{x,y}),i}$ is constant during the ith time slot, where the channel between the nodes x and y at time slot i is given by

$$H_{(\mathrm{x,y}),i} = \sqrt{d_{(\mathrm{x,y})}^{\alpha_{pl}}}\tilde{H}_{(\mathrm{x,y}),i} \tag{3.1}$$

where $d_{(\mathrm{x,y})}$ is the distance between the two nodes, α_{pl} is a path loss constant, and $\tilde{H}_{(\mathrm{x,y}),i}$ is the fading coefficient between x and y. Let us similarly define $H_{(\mathrm{ss,sr}),i}$, $H_{(\mathrm{sr,sd}),i}$, $H_{(\mathrm{ss,pd}),i}$, $H_{(\mathrm{sr,pd}),i}$, $H_{(\mathrm{ps,pd}),i}$, $H_{(\mathrm{ps,sd}),i}$ and $H_{(\mathrm{ps,sr}),i}$ as the channel coefficients during the ith time slot between SS and SR, SR and SD, SS and PD, SR and PD, PS and PD, PS and SD, and PS and SR, respectively. The harvested energy at SS and SR during the ith time slot are denoted by $E_{\mathrm{ss},i}$ and $E_{\mathrm{sr},i}$, respectively. Two transmission cases are studied, case 1) SS keeps transmitting its data for all of the T slots, case 2) SS completes its transmission within M slots where $M < T$, after that, it keeps harvesting for $T - M$ slots without sending data to the SR. This gives the SR extra time $T - M$ to try sending all the data in its buffer. The latter case allows SR to receive data from the SS but not transmit them to the SD due to channel conditions, primary interference, or lack of harvested energy at the SR.

3.2 Problem Formulation

It is assumed that the PS transmits to the PD during all time slots. So, the received signal and the SINR at the PD side during the ith slot are, respectively, given as

$$\begin{aligned} y_{\mathrm{pd},i} = &\sqrt{P_{\mathrm{ps},i}^{Tx}}\, H_{(\mathrm{ps,pd}),i}\, x_{\mathrm{ps},i} + \sqrt{P_{\mathrm{ss},i}^{Tx}}\, H_{(\mathrm{ss,pd}),i}\, x_{\mathrm{ss},i} \\ &+ \sqrt{P_{\mathrm{sr},i}^{Tx}}\, H_{(\mathrm{sr,pd}),i}\, x_{\mathrm{sr},i} + n_{\mathrm{pd},i}, \quad i = 1, \ldots, T \end{aligned} \tag{3.2}$$

$$\Gamma_{\mathrm{pd},i} = \frac{P^{Tx}_{\mathrm{ps},i}|H_{(\mathrm{ps,pd}),i}|^2}{\sigma_n^2 + P^{Tx}_{\mathrm{ss},i}|H_{(\mathrm{ss,pd}),i}|^2 + P^{Tx}_{\mathrm{sr},i}|H_{(\mathrm{sr,pd}),i}|^2}, \qquad i = 1, \ldots, T \tag{3.3}$$

where $P^{Tx}_{\mathrm{ps},i}$ and $x_{\mathrm{ps},i}$ are the peak transmit power, and the transmitted signal by PS, respectively. $n_{\mathrm{pd},i}$ is additive Gaussian noise with zero-mean and noise variance σ_n^2. The received signals at the SR and the SD during ith slot are, respectively, given as

$$y_{\mathrm{sr},i} = \sqrt{P^{Tx}_{\mathrm{ss},i}}H_{(\mathrm{ss,sr}),i}x_{\mathrm{ss},i} + \sqrt{P^{Tx}_{\mathrm{ps},i}}H_{(\mathrm{ps,sr}),i}x_{\mathrm{ps},i} + n_{\mathrm{sr},i}, \quad i = 1, \ldots, M \tag{3.4}$$

$$y_{\mathrm{sd},i} = \sqrt{P^{Tx}_{\mathrm{sr},i}}H_{(\mathrm{sr,sd}),i}x_{\mathrm{sr},i} + \sqrt{P^{Tx}_{\mathrm{ps},i}}H_{(\mathrm{ps,sd}),i}x_{\mathrm{ps},i} + n_{\mathrm{sd},i}, \quad i = 1, \ldots, T \tag{3.5}$$

where $P^{Tx}_{\mathrm{ss},i}$ and $P^{Tx}_{\mathrm{sr},i}$ are the peak power transmitted by SS and SR, respectively. $x_{\mathrm{ss},i}$ and $x_{\mathrm{sr},i}$ are the transmitted signals by SS and SR during the ith time slot, respectively. We assume that $n_{\mathrm{sr},i}$ and $n_{\mathrm{sd},i}$ are Gaussian, independent, zero mean, and they both also have variance σ_n^2. The SINR at the SR and SD are given, respectively, by

$$\Gamma_{\mathrm{sr},i} = \frac{P^{Tx}_{\mathrm{ss},i}|H_{(\mathrm{ss,sr}),i}|^2}{\sigma_n^2 + P^{Tx}_{\mathrm{ps},i}|H_{(\mathrm{ps,sr}),i}|^2}, \quad i = 1, \ldots, M \tag{3.6}$$

$$\Gamma_{\mathrm{sd},i} = \frac{P^{Tx}_{\mathrm{sr},i}|H_{(\mathrm{sr,sd}),i}|^2}{\sigma_n^2 + P^{Tx}_{\mathrm{ps},i}|H_{(\mathrm{ps,sd}),i}|^2}, \quad i = 1, \ldots, T \tag{3.7}$$

where the SR can remove the self interference by eliminating its own signal.

The objective is to optimize transmit powers for both SS and SR in order to maximize the sum rate between the SR and SD during T time slots, while satisfying the required QoS of the primary users, in addition to the data and energy causality constraints. The sum rate from SR to SD is given by

$$\max_{\{P^{Tx}_{\mathrm{ss},i}, P^{Tx}_{\mathrm{sr},i}, B_{\mathrm{ss},i}, B_{\mathrm{sr},i}\}} \sum_{i=1}^{T} \log(1 + \Gamma_{\mathrm{sd},i}) \tag{3.8}$$

Thus, the energy causality constraints at SS and SR (i.e., the SS and SR cannot use more energy than their battery levels in the previous time slot), respectively, are given by

$$P^{Tx}_{\mathrm{ss},i}\, T_c \leq B_{\mathrm{ss},i-1}, \quad i = 1, \ldots, T \tag{3.9}$$

$$P^{Tx}_{\mathrm{sr},i}\, T_c \leq B_{\mathrm{sr},i-1}, \quad i = 1, \ldots, T \tag{3.10}$$

where T_c is the transmission duration. Since SS will keep silent (i.e. $P^{Tx}_{\mathrm{ss},i} = 0$) after time slot M, the constraint in (3.9) can be written for all time slots.

Battery overflow constraints for both SS and SR (i.e., the update rules for the available energy in their batteries at the end of the current time slot, which are functions of the previous battery levels, in addition to transmit and harvested energy in the current time slot), respectively, are given by

$$B_{\mathrm{ss},i} = \min\{B_{\mathrm{ss},i-1} + E_{\mathrm{ss},i} - P_{\mathrm{ss},i}^{Tx}\, T_c, B_{\max}\}, \quad i = 1, \ldots, T \tag{3.11}$$

$$B_{\mathrm{sr},i} = \min\{B_{\mathrm{sr},i-1} + E_{\mathrm{sr},i} - P_{\mathrm{sr},i}^{Tx}\, T_c, B_{\max}\}, \quad i = 1, \ldots, T \tag{3.12}$$

Constraints (3.11) and (3.12) can be rewritten as follow

$$B_{\mathrm{ss},i} \leq B_{\mathrm{ss},i-1} + E_{\mathrm{ss},i} - P_{\mathrm{ss},i}^{Tx}\, T_c, \qquad i = 1, \ldots, T \tag{3.13}$$

$$B_{\mathrm{ss},i} \leq B_{\max}, \qquad i = 1, \ldots, T \tag{3.14}$$

$$B_{\mathrm{sr},i} \leq B_{\mathrm{sr},i-1} + E_{\mathrm{sr},i} - P_{\mathrm{sr},i}^{Tx}\, T_c, \qquad i = 1, \ldots, T \tag{3.15}$$

$$B_{\mathrm{sr},i} \leq B_{max}, \qquad i = 1, \ldots, T \tag{3.16}$$

$P_{\mathrm{ss},i}^{Tx}$, $P_{\mathrm{sr},i}^{Tx}$, $B_{\mathrm{ss},i}$ and $B_{\mathrm{sr},i}$ should satisfy the following constraints

$$P_{\mathrm{ss},i}^{Tx} \geq 0, \qquad i = 1, \ldots, M \tag{3.17}$$

$$P_{\mathrm{ss},i}^{Tx} = 0, \qquad i = M+1, \ldots, T \tag{3.18}$$

$$P_{\mathrm{sr},i}^{Tx} \geq 0, \qquad i = 1, \ldots, T \tag{3.19}$$

$$B_{\mathrm{ss},i}, B_{\mathrm{sr},i} \geq 0, \qquad i = 1, \ldots, T \tag{3.20}$$

Without loss of generality, and for simplicity, it is assumed that T_c is normalized, hence, it is omitted from the following equations. The following constraint is to ensure the data causality (i.e., the SR will not transmit the data to the SD before receiving it)

$$\sum_{k=1}^{i} \log(1 + \Gamma_{\mathrm{sd},k}) \leq \sum_{k=1}^{i} \log(1 + \Gamma_{\mathrm{sr},k}), \quad i = 1, \ldots, M \tag{3.21}$$

The data queue of the SR increases by $\log(1+\Gamma_{\mathrm{sr},k})$ bit/Hz in the following time slot, when the SS transmits with $P_{\mathrm{ss},i}^{Tx}$ to the SR during the ith slot. The same observation can be made at SD when the SR transmits with $P_{\mathrm{sr},i}^{Tx}$.

To grantee a QoS to the primary network, the following constraint should be satisfied

$$\Gamma_{\mathrm{pd},i} \geq \Gamma, \qquad i = 1, \ldots, T \tag{3.22}$$

where Γ is the predefined SINR QoS threshold at the PD. With simple manipulations, constraint (3.22) can be rewritten as

$$P_{\mathrm{ss},i}^{Tx}|H_{(\mathrm{ss,pd}),i}|^2 + P_{\mathrm{sr},i}^{Tx}|H_{(\mathrm{sr,pd}),i}|^2 \leq I_{th,i}, \quad i = 1, \ldots, T \tag{3.23}$$

where $I_{th,i}$ is given by

$$I_{th,i} = \frac{P_{\mathrm{ps},i}^{Tx}|H_{(\mathrm{ps,pd}),i}|^2}{\Gamma} - \sigma_n^2, \quad i = 1, \ldots, T \tag{3.24}$$

Finally, the following constraint requires that the received data at SD during $T - M$ slots (during the time where the SS is not transmitting) is limited by the data in the SR buffer

$$\sum_{i=1}^{T} \log(1 + \Gamma_{sd,i}) \leq \sum_{i=1}^{M} \log(1 + \Gamma_{sr,i}) \tag{3.25}$$

The end-to-end rate optimization problem that maximizes the sum rate between SR and SD can now be formulated as

$$\begin{aligned} &\max_{\{P_{\mathrm{ss},i}^{Tx}, P_{\mathrm{sr},i}^{Tx}, B_{\mathrm{ss},i}, B_{\mathrm{sr},i}\}} \quad \sum_{i=1}^{T} \log(1 + \Gamma_{sd,i}) \\ &\text{subject to} \quad (3.9)\text{–}(3.10), (3.13)\text{–}(3.21), (3.23), (3.17)\text{–}(3.25) \end{aligned} \tag{3.26}$$

3.3 Proposed Solution

The formulated optimization problem given in (3.26) is a non convex problem because of constraints (3.21) and (3.26). In the sequel, we will transform it to an equivalent convex form. Change of variables can be used as follows [22]. Let $C_{\mathrm{sr},i} = \log(1 + \Gamma_{\mathrm{sr},i})$, $C_{\mathrm{sd},i} = \log(1 + \Gamma_{\mathrm{sd},i})$.

For simplicity, let us define the following

$$\varsigma_i = \frac{\sigma_n^2 + P_{\mathrm{ps},i}^{Tx}|H_{(\mathrm{ps,sr}),i}|^2}{|H_{(\mathrm{ss,sr}),i}|^2}, \quad i = 1, \ldots, M \tag{3.27}$$

$$\vartheta_i = \frac{\sigma_n^2 + P_{\mathrm{ps},i}^{Tx}|H_{(\mathrm{ps,sd}),i}|^2}{|H_{(\mathrm{sr,sd}),i}|^2}, \quad i = 1, \ldots, T \tag{3.28}$$

From (3.6), (3.7), (3.27), and (3.28), $P_{\mathrm{ss},i}^{Tx}$ and $P_{\mathrm{sr},i}^{Tx}$ can be written, respectively, as follows

$$P_{\mathrm{ss},i}^{Tx} = \varsigma_i \Gamma_{\mathrm{sr},i} \tag{3.29}$$

$$P_{\mathrm{sr},i}^{Tx} = \vartheta_i \Gamma_{\mathrm{sd},i} \tag{3.30}$$

Therefore, the formulated optimization problem after transformation can be written as

$$\max_{\{C_{\mathrm{sd},i},C_{\mathrm{sr},i},B_{\mathrm{ss},i},B_{\mathrm{sr},i}\}} \quad \sum_{i=1}^{T} C_{\mathrm{sd},i}$$

$$\sum_{k=1}^{i} C_{\mathrm{sd},k} \leq \sum_{k=1}^{i} C_{\mathrm{sr},k}, \qquad i = 1,\ldots,M$$

$$\varsigma_i(2^{C_{\mathrm{sr},i}} - 1) \leq B_{\mathrm{ss},i-1}, \qquad i = 1,\ldots,T$$

$$\vartheta_i(2^{C_{\mathrm{sd},i}} - 1) \leq B_{\mathrm{sr},i-1}, \qquad i = 1,\ldots,T$$

$$B_{\mathrm{ss},i} \leq B_{\mathrm{ss},i-1} + E_{\mathrm{ss},i} - \varsigma_i(2^{C_{\mathrm{sr},i}} - 1), \qquad i = 1,\ldots,T$$

$$B_{\mathrm{ss},i} \leq B_{max}, \qquad i = 1,\ldots,T$$

$$B_{\mathrm{sr},i} \leq B_{\mathrm{sr},i-1} + E_{\mathrm{sr},i} - \vartheta_i(2^{C_{\mathrm{sd},i}} - 1), \qquad i = 1,\ldots,T$$

$$B_{\mathrm{sr},i} \leq B_{\max}, \qquad i = 1,\ldots,T$$

$$\sum_{i=1}^{T} C_{\mathrm{sd},i} \leq \sum_{i=1}^{M} C_{\mathrm{sr},i}$$

$$\varsigma_i(2^{C_{\mathrm{sr},i}} - 1)|H_{(\mathrm{ss,pd}),i}|^2 + \vartheta_i(2^{C_{\mathrm{sd},i}} - 1)|H_{(\mathrm{sr,pd}),i}|^2 \leq I_{th,i}, \qquad i = 1,\ldots,T$$

$$2^{C_{\mathrm{sr},i}} - 1 \geq 0, \qquad i = 1,\ldots,M$$

$$2^{C_{\mathrm{sr},i}} - 1 = 0, \qquad i = M+1,\ldots,T$$

$$2^{C_{\mathrm{sd},i}} - 1 \geq 0, \qquad i = 1,\ldots,T$$

$$B_{\mathrm{ss},i}, B_{\mathrm{sr},i} \geq 0, \qquad i = 1,\ldots,T \tag{3.31}$$

Now to transform the problem to a convex one, the last three constraints can be rewritten, respectively, as

$$-C_{\mathrm{sr},i} \leq 0, \quad i = 1,\ldots,M \tag{3.32}$$

$$C_{\mathrm{sr},i} = 0, \quad i = M+1,\ldots,T \tag{3.33}$$

$$-C_{\mathrm{sd},i} \leq 0, \quad i = 1,\ldots,T \tag{3.34}$$

Hence, the optimization problem becomes a convex problem, where the objective function is concave and the constraints are convex functions [82]. The Lagrangian of (3.31) is given in (3.35).

$$\begin{aligned}\mathcal{L} = &-\sum_{i=1}^{T} C_{\mathrm{sd},i} + \sum_{i=1}^{M} \mu_i \Big[\sum_{k=1}^{i}(C_{\mathrm{sd},k} - C_{\mathrm{sr},k})\Big] + \sum_{i=1}^{T} \theta_i \Big[\varsigma_i(2^{C_{\mathrm{sr},i}} - 1) - B_{\mathrm{ss},i-1}\Big] \\ &+ \sum_{i=1}^{T} \omega_i \Big[\vartheta_i(2^{C_{\mathrm{sd},i}} - 1) - B_{\mathrm{sr},i-1}\Big] + \sum_{i=1}^{T} \eta_i \Big[B_{\mathrm{ss},i} - B_{\mathrm{ss},i-1} - E_{\mathrm{ss},i} + \varsigma_i(2^{C_{\mathrm{sr},i}} - 1)\Big] \\ &+ \sum_{i=1}^{T} \lambda_i \Big[B_{\mathrm{sr},i} - B_{\mathrm{sr},i-1} - E_{\mathrm{sr},i} + \vartheta_i(2^{C_{\mathrm{sd},i}} - 1)\Big] + \sum_{i=1}^{T} \kappa_i \Big[B_{\mathrm{ss},i} - B_{max}\Big] \\ &+ \sum_{i=1}^{T} \phi_i \Big[B_{\mathrm{sr},i} - B_{\max}\Big] - \sum_{i=1}^{M} \sigma_i C_{\mathrm{sr},i} + \sum_{i=M+1}^{T} \nu_i C_{\mathrm{sr},i} + \xi\Big[\sum_{k=1}^{T} C_{\mathrm{sd},k} - \sum_{k=1}^{T} C_{\mathrm{sr},k}\Big] \\ &+ \sum_{i=1}^{T} \psi_i \Big[\varsigma_i(2^{C_{r,i}} - 1)|H_{(\mathrm{ss,pd}),i}|^2 + \vartheta_i(2^{C_{\mathrm{sd},i}} - 1)|H_{(\mathrm{sr,pd}),i}|^2 - I_{th,i}\Big] \\ &- \sum_{i=1}^{T} \rho_i C_{\mathrm{sd},i} - \sum_{i=1}^{T} \varphi_i B_{\mathrm{ss},i} - \sum_{i=1}^{T} \varrho_i B_{\mathrm{sr},i}\end{aligned} \tag{3.35}$$

The Karush-Kuhn-Tucker (KKT) conditions are given as follows

$$-\sum_{i=k}^{M} \mu_i - \sigma_k - \xi + \varsigma_k \ln(2) 2^{C_{\mathrm{sr},k}} \Big\{\theta_k + \eta_k + \psi_k |H_{(\mathrm{ss,pd}),k}|^2\Big\} = 0, \qquad k = 1, \ldots, M \tag{3.36}$$

$$\nu_k + \varsigma_k \ln(2) 2^{C_{\mathrm{sr},k}} \Big\{\theta_k + \eta_k + \psi_k |H_{(\mathrm{ss,pd}),k}|^2\Big\} = 0, \qquad k = M+1, \ldots, T \tag{3.37}$$

$$-1 + \sum_{i=k}^{M} \mu_i - \rho_k + \xi + \vartheta_k (\ln(2)) 2^{C_{\mathrm{sd},k}} \Big\{\omega_k + \lambda_k + \psi_k |H_{(\mathrm{sr,pd}),k}|^2\Big\} = 0, \quad k = 1, \ldots, M \tag{3.38}$$

$$-1 - \rho_k + \xi + \vartheta_k (\ln(2)) 2^{C_{\mathrm{sd},k}} \Big\{\omega_k + \lambda_k + \psi_k |H_{(\mathrm{sr,pd}),k}|^2\Big\} = 0, \qquad k = M+1, \ldots, T \tag{3.39}$$

Using (3.36) - (3.39), the closed form expressions can be obtained as

$$C^*_{\mathrm{sr},k} = \begin{cases} (c_{\mathrm{sr},1})^+, & k = 1, \ldots, M \\ \\ (c_{\mathrm{sr},2})^+, & k = M+1, \ldots, T \end{cases} \tag{3.40}$$

$$C^*_{\mathrm{sd},k} = \begin{cases} (c_{\mathrm{sd},1})^+, & k = 1, \ldots, M \\ \\ (c_{\mathrm{sd},2})^+, & k = M+1, \ldots, T \end{cases} \tag{3.41}$$

where

$$c_{\mathrm{sr},1} = \log_2\left(\frac{\sum_{i=k}^{T}\mu_i + \sigma_k + \xi}{\varsigma_k(\ln(2))\Big\{\theta_k + \eta_k + \psi_k|H_{(\mathrm{ss,pd}),k}|^2\Big\}}\right) \tag{3.42}$$

$$c_{\mathrm{sr},2} = \log_2\left(\frac{-\nu_k}{\varsigma_k(\ln(2))\Big\{\theta_k + \eta_k + \psi_k|H_{(\mathrm{ss,pd}),k}|^2\Big\}}\right) \tag{3.43}$$

$$c_{\mathrm{sd},1} = \log_2\left(\frac{1 - \sum_{i=k}^{T}\mu_i + \rho_k - \xi}{\vartheta_k(\ln(2))\Big\{\omega_k + \lambda_k + \psi_k|H_{(\mathrm{sr,pd}),k}|^2\Big\}}\right) \tag{3.44}$$

$$c_{\mathrm{sd},2} = \log_2\left(\frac{1 - \sum_{i=k}^{T}\mu_i + \rho_k - \xi}{\vartheta_k(\ln(2))\Big\{\omega_k + \lambda_k + \psi_k|H_{(\mathrm{sr,pd}),k}|^2\Big\}}\right) \tag{3.45}$$

and $(x)^+ = \max(x, 0)$.

The closed form expressions for the optimal power levels can be obtained from (3.40), (3.41) and expressed as

$$P_{\mathrm{ss},k}^{Tx*} = \varsigma_i(2^{C^*_{\mathrm{sr},i}} - 1) \tag{3.46}$$

$$P_{\mathrm{sr},k}^{Tx*} = \vartheta_i(2^{C^*_{\mathrm{sd},i}} - 1) \tag{3.47}$$

Note that the optimal powers are functions of the Lagrangian multipliers. To find the optimal Lagrangian multipliers, the projected subgradient method is employed [83]. In this method, any initial values of the Lagrangian multipliers can be used as a start points to evaluate the rates given in (3.40), (3.41). After that, projections of the primal variables on the constraints are computed, so that all values of $C_{\mathrm{sr},k}$ and $C_{\mathrm{sd},k}$ satisfy the feasibility of the solution, where the updated values of the optimal rates/powers and the Lagrangian multipliers are repeated until convergence. The Lagrangian multipliers at the next iteration $(n+1)$ are given as follows

$$\mu_k^{n+1} = \mu_k^n - \delta_{\mu,k}^n\Big[\sum_{i=1}^{k}(C_{\mathrm{sd},i} - C_{\mathrm{sr},i})\Big], \qquad k = 1, \ldots, M \tag{3.48}$$

$$\theta_k^{n+1} = \theta_k^n - \delta_{\theta,k}^n\big[\varsigma_k(2^{C_{\mathrm{sr},k}} - 1) - B_{\mathrm{ss},k-1}\big], \qquad k = 1, \ldots, T \tag{3.49}$$

$$\omega_k^{n+1} = \omega_k^n - \delta_{\omega,k}^n\big[\vartheta_k(2^{C_{\mathrm{sd},k}} - 1) - B_{\mathrm{sr},k-1}\big], \qquad k = 1, \ldots, T \tag{3.50}$$

$$\eta_k^{n+1} = \eta_k^n - \delta_{\eta,k}^n\big[B_{\mathrm{ss},k} - B_{\mathrm{ss},k-1} - E_{\mathrm{ss},k} + \varsigma_k(2^{C_{\mathrm{sr},k}} - 1)\big], \qquad k = 1, \ldots, T \tag{3.51}$$

$$\lambda_k^{n+1} = \lambda_k^n - \delta_{\lambda,k}^n\big[B_{\mathrm{sr},k} - B_{\mathrm{sr},k-1} - E_{\mathrm{sr},k} + \vartheta_k(2^{C_{\mathrm{sd},k}} - 1)\big], \qquad k = 1, \ldots, T \tag{3.52}$$

$$B_{\mathrm{ss},k}^{n+1} = B_{\mathrm{ss},k}^{n} - \delta_{\mathrm{ss},k}^{n}\big[\eta_k - \eta_{k+1} - \theta_{k+1} + \kappa_k - \varphi_k\big], \qquad k = 1,\ldots,T \tag{3.53}$$

$$B_{\mathrm{sr},k}^{n+1} = B_{\mathrm{sr},k}^{n} - \delta_{\mathrm{sr},k}^{n}\big[\lambda_k - \lambda_{k+1} - \omega_{k+1} + \phi_k - \varrho_k\big], \qquad k = 1,\ldots,T \tag{3.54}$$

$$\kappa_k^{n+1} = \kappa_k^{n} - \delta_{\kappa,k}^{n}\big[B_{\mathrm{ss},k} - B_{max}\big], \qquad k = 1,\ldots,T \tag{3.55}$$

$$\phi_k^{n+1} = \phi_k^{n} - \delta_{\phi,k}^{n}\big[B_{\mathrm{sr},k} - B_{max}\big], \qquad k = 1,\ldots,T \tag{3.56}$$

$$\sigma_k^{n+1} = \sigma_k^{n} - \delta_{\sigma,k}^{n}\big[-C_{\mathrm{sr},k}\big], \qquad k = 1,\ldots,M \tag{3.57}$$

$$\nu_k^{n+1} = \nu_k^{n} - \delta_{\nu,k}^{n}\big[C_{\mathrm{sr},k}\big], \qquad k = T+1,\ldots,T \tag{3.58}$$

$$\rho_k^{n+1} = \rho_k^{n} - \delta_{\rho,k}^{n}\big[-C_{\mathrm{sd},k}\big], \qquad k = 1,\ldots,T \tag{3.59}$$

$$\psi_k^{n+1} = \psi_k^{n} - \delta_{\psi,k}^{n}\big[-I_{th,k} + \varsigma_k(2^{C_{\mathrm{sr},k}} - 1)|H_{(\mathrm{ss,pd}),k}|^2 + \vartheta_k(2^{C_{\mathrm{sd},k}} - 1)|H_{(\mathrm{sr,pd}),k}|^2\big], k = 1,\ldots,T \tag{3.60}$$

$$\xi^{n+1} = \xi^{n} - \delta_{\xi}^{n}\Big[\sum_{i=1}^{T} C_{\mathrm{sd},i} - \sum_{i=1}^{M} C_{\mathrm{sr},i}\Big] \tag{3.61}$$

$$\varphi_k^{n+1} = \varphi_k^{n} - \delta_{\varphi,k}^{n}\big[-B_{\mathrm{ss},k}\big], \qquad k = 1,\ldots,T \tag{3.62}$$

$$\varrho_k^{n+1} = \varrho_k^{n} - \delta_{\varrho,k}^{n}\big[-B_{\mathrm{sr},k}\big], \qquad k = 1,\ldots,T \tag{3.63}$$

The step-sizes are updated according to the nonsummable diminishing step length policy [84]. We have also used constant step-size, which seems to work well also.

3.4 Simulation Results

In this section, simulation results are provided to demonstrate the performance of the proposed system model given in Figure 3.1. In all simulation results, it is assumed that all the time slots are of length 1 second (i.e., $T_c = 1$ sec). The available bandwidth $B = 1$ MHz and the noise spectral density is $N_0 = 10^{-16}$ W/Hz, so that the noise variance $\sigma_n^2 = BN_0$. It is also considered that both of the SS and SR are equipped with solar panels of area of 100 cm^2 with 20% efficiency. Both SS and SR equipped with finite batteries of size $B_{max} = 5$ J. The initial battery levels of both SS and SR are zeros. The harvested energy levels are given by normal distribution with a mean equal to 0.2 and standard deviation equal to 0.033, where the harvested energy values are restricted to be between 0.1 and 0.3 Joule. Normal distribution is used as a distribution of the average harvested energy levels according to central limit theorem,

where the sum of a large number of independent and identically distributed (i.i.d) variables is approximated by normal distribution [85]. The simulation is performed with $M = 5$ time slots and $T = 7$ time slots, unless otherwise stated. All channels are considered to be i.i.d Rayleigh fading channels, and the path loss exponent is equal to 4.

The distances between the communication nodes are assumed as $d_{(\mathrm{ss,sr})} = d_{(\mathrm{sr,sd})} = d_{(\mathrm{ps,pd})} = 50$ meters, corresponding to the distances between SS and SR, SR and SD, and PS and PD, respectively, and $d_{(\mathrm{ss,pd})} = d_{(\mathrm{sr,pd})} = d_{(\mathrm{ps,sd})} = d_{(\mathrm{ps,sr})} = 100$ meters, corresponding to the distances between SS and PD, SR and PD, PS and SD, and PS and SR, respectively.

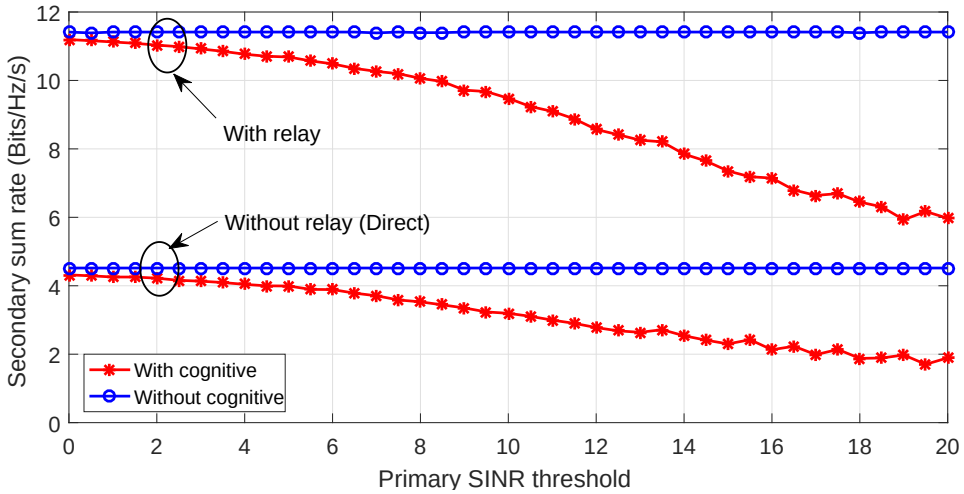

Figure 3.3: Secondary sum rate of direct and relayed transmission systems with and without CR versus primary SINR.

Figure 3.3 plots the maximum achievable secondary sum rate as a function of the pre-defined primary SINR threshold with and without cooperative relay. Thanks to relaying, it can be seen that using a relay increases the overall sum rate with considerable gap. This figure also shows that as the primary SINR threshold increases, the secondary sum rate decreases for both cases. This is due to the fact that by increasing the primary SINR threshold, the allowable transmit power of the secondary nodes should be reduced in order to respect this threshold. This figure also compares the proposed scheme without the interference constraint (i.e., no primary users in the system) as an upper bound (i.e., secondary nodes have flexibility to transmit with more powers).

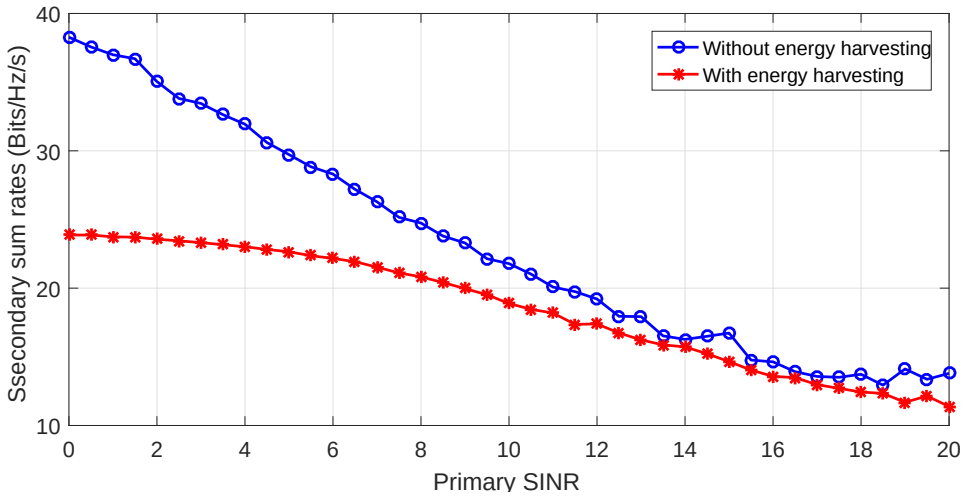

Figure 3.4: Sum rate from the SR to the SD with and without EH versus different values for SINR at the primary network.

Figure 3.4 compares the performance of the proposed system with two cases, when the secondary network uses EH technology, and when it uses non-harvested traditional batteries. One can see that, the performance of the traditional batteries outperforms the energy harvested with a considerable gap in the region where the primary SINR threshold is relatively small. However, this gap becomes smaller and smaller when the SINR threshold is relatively large. This can be justified by the fact that with high values of SINR at the primary network, both models are restricted to transmit with low power to satisfy the primary QoS. In addition to the saving energy advantage, using EH technology becomes more interesting at the high values of the primary SINR threshold.

Finally, Figure 3.5 shows the importance of having some extra slots $T - M$ that allow the relay to empty its data buffer. This can help the SD to avoid missing data that could not receive within the M time slots. This figure plots the sum rate between the SR and the SD versus the number of extra time slots $T - M$ for different values of primary SINR threshold. It can be shown that the sum rate can be increased up to a certain level, where adding more extra slots will not contribute to the rate, since the SR broadcast all the data in its buffer. The optimal number of $T - M$ time slots can be deduced from the figure for different values of the primary interference threshold SINR$= \{0, 10, 20\}$, where the optimal $T - M$ time slots increases as the SINR increases.

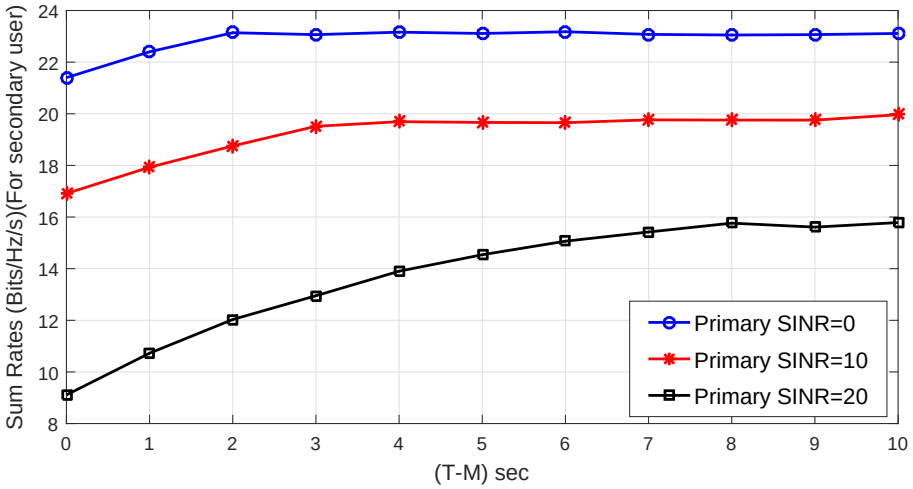

Figure 3.5: The effect of increasing the extra time on sum rate the SR to the SD versus number of extra time slots.

3.5 Chapter Summary

In this chpter, cooperative underlay CRN with EH was investigated, where the secondary users can access the primary frequency band by exploiting the allowance of the SINR constraint. Each of the secondary source and relay is equipped with infinite data buffer to carry data packets to be delivered and finite battery to store the harvested energy. We formulated an optimization problem aiming at maximizing the sum of the achievable rate over multiple time slots. After solving the problem using a projected subgradient method, the performance of the proposed scheme was investigated. Finally, we discussed the effect adding extra time for transmitting data from relay to destination that could not receive in allotted time.

CHAPTER 4. LOOK-AHEAD AND LEARNING APPROACHES FOR ENERGY HARVESTING COMMUNICATIONS SYSTEMS

This chapter investigates the performance of a communications system with an energy harvesting (EH) node. This system consists of a transmitter and a receiver. The transmitter is equipped with an infinite buffer to store data, and EH capability to harvest renewable energy and store it in a finite battery. The goal is to maximize the expected cumulative throughput of such systems while prolonging their lifetime. The problem of finding an optimal power allocation policy is formulated as a Markov decision process (MDP) with discrete state and action spaces. Two cases are considered based on the availability of the statistical knowledge. When this knowledge is available, an algorithm is designed to maximize the expected throughput, while reducing the complexity of traditional methods (e.g., value iteration, and policy iteration). The proposed algorithm uses instant knowledge about the channel, harvested energy, and current battery level to find a policy that maximizes the throughput. For the second scenario, when the underlying statistical knowledge of the underlying processes is unavailable, reinforcement learning (RL) is used. Two different exploration algorithms, convergence-based and the epsilon-greedy algorithms, are used. The first one uses the action-value function convergence error and the exploration time threshold to balance between exploration and exploitation, while the second algorithm tries to achieve balancing through the exploration probability epsilon. Simulations and comparisons with conventional algorithms show the effectiveness of the look-ahead algorithm when the statistical knowledge is available, and the effectiveness of RL in optimizing the system performance when this knowledge is unavailable.

The remainder of this chapter is organized as follows. Section 4.1 describes the proposed communications system model. Then, the problem is formulated in subsection 4.2.1. The

problem is reformulated as an MDP in subsection 4.2.2. Section 4.3 presents the look-ahead algorithm, which is used when the dynamics of the underlying model are available. Section 4.4 discusses two exploration algorithms for RL to optimize the system performance when the knowledge about the underlying model is unavailable. Numerical simulation results are presented in Section 4.6. Finally, the paper is concluded in Section 4.7.

4.1 System Model

In this section, a point-to-point communication system that consists of a source (S) and a destination (D) is considered. As illustrated in Figure 6.1, each of S and D is equipped with an infinite buffer to store data. S has the capability of harvesting solar energy and storing it in a finite battery. We consider a time slotted system with equal length time slots, where each slot has a duration of T_c. For this system, the energy can be harvested and stored as an integer multiple of a fundamental unit. Let the maximum capacity of the battery be $B_{\max}$. B_i represents the battery level of S at the beginning of time slot i, where $B_i \in \mathcal{B} \triangleq \{b_1, b_2, ..., b_{N_b}\}$, N_b is the number of elements in $\mathcal{B}$, and $b_{N_b} = B_{\max}$.

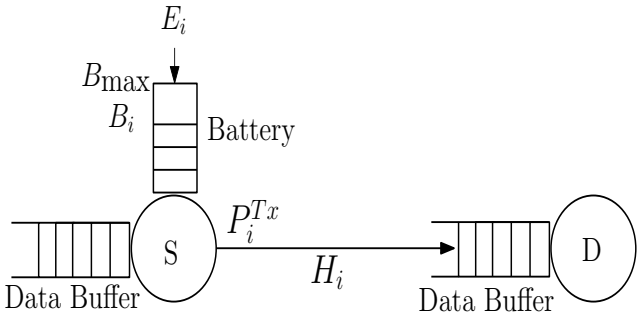

Figure 4.1: Point-to-point communication system with an energy harvesting source.

The energy harvesting and channel gain processes are modeled as two independent Markov chains. Based on the current technologies, the amount of energy to be harvested E_i can be computed precisely [86]. During time slot i, the source harvests E_i units from solar sources, where $E_i \in \mathcal{E}_n \triangleq \{e_1, e_2, ..., e_{N_E}\}$, and N_E represents the number of elements in $\mathcal{E}_n$. $p_{\mathcal{E}_n}(e'|e)$ is the transition probability of harvested energy going from state e to state e' during one step transition. Let H_i be the channel state during time slot i, where $H_i \in \mathcal{H} \triangleq \{h_1, h_2, ..., h_{N_H}\}$,

and N_H is the number of elements in $\mathcal{H}$. $p_{\mathcal{H}}(h'|h)$ is the transition probability for the channel going from state h to state h' during one time slot.

Let P_i^{Tx} be the transmission power during time slot i, and T_c is the transmission duration, which is constant during all time slots and equals 1 second. Since the source has causal knowledge about its environment, P_i^{Tx} is a function of E_i, B_i, and H_i. $P_i^{Tx} \in \mathcal{P}^{Tx} \triangleq \{p_1^{Tx}, p_2^{Tx}, ..., p_{N_p}^{Tx}\}$, where N_p is the number of elements in $\mathcal{P}^{Tx}$. In the proposed scheme, selecting P_i^{Tx} fulfills the Markov property, so the problem of optimizing the transmission power can be modeled as an MDP [4]. In this model, energy consumption is considered only due to data transmission, and it does not take into account any other energy consumption, such as processing, circuitry, etc.

In this context, the harvested energy is managed using harvest-store-use scheme. Using this scheme, harvested energy is stored partially or totally in a battery before using it. This scheme is characterized by its suitability for systems equipped with energy storage devices. It enables these systems to improve their performance by storing the harvested energy and using it when the channel gains are relatively good [87; 88].

4.2 Problem Formulation

4.2.1 Throughput Maximization Problem

In this section, we formulate the problem of maximizing the throughput by optimizing the transmission power over an infinite horizon. Two scenarios are taken into account. The first one considers the existence of statistical knowledge about the EH and channel gain processes, while the other considers the case of having only causal knowledge about these processes.

Due to lack of information about the harvestable energy and the channel gains in the future, the goal is to maximize the expected discounted return, where the discounted return following time t, G_t, is given by

$$G_t = \sum_{i=t}^{T-1} \gamma^{i-t} R_{i+1} \tag{4.1}$$

where t is the starting time for collecting a sequence of rewards, T is a final time step of an episode, and $\gamma \in (0, 1)$ is the discount factor, which is used to weight the value (i.e., the

importance) of the received data over time. It is a measure for the importance of transmitting data at the current time compared to transmitting the same data in the future, when it might not be important to the destination. R_{i+1} is the reward (i.e., the amount of received data) at time $i+1$ resulting from transmission using P_i^{Tx}

$$R_{i+1} = T_c \log_2 \left(1 + \frac{P_i^{Tx} |H_i|^2}{\sigma_n^2}\right) \tag{4.2}$$

where σ_n^2 is the noise variance.

The energy causality constraints at the source, which is to ensure that the source cannot use more energy than its current battery level, and is given by

$$T_c \, P_i^{Tx} \leq B_i, \quad i = t, \ldots, T-1 \tag{4.3}$$

Battery overflow constraint for the source, which is a rule for updating the energy level in the source's battery. It is a function of the battery level, transmission energy, harvested energy during time slot i, which is given by

$$B_{i+1} = \min\{B_i + E_i, B_{\max}\} - T_c \, P_i^{Tx}, \quad i = t, \ldots, T-1 \tag{4.4}$$

Finally, P_i^{Tx}, and B_i should satisfy the following constraints

$$P_i^{Tx} \geq 0, \qquad i = t, \ldots, T-1 \tag{4.5}$$

$$B_i \geq 0, \qquad i = t, \ldots, T-1 \tag{4.6}$$

The optimization problem that maximizes the expected discounted return over an infinite horizon can now be formulated as

$$\max_{\{P_i^{Tx}\}} \lim_{T \to \infty} \mathbb{E}[G_t]$$

such that for $i = t, \ldots, T-1$,

$$\begin{aligned} P_i^{Tx} \, T_c &\leq B_i, \\ B_{i+1} &= \min\{B_i + E_i, B_{\max}\} - T_c \, P_i^{Tx}, \\ P_i^{Tx} &\geq 0, \\ B_i &\geq 0. \end{aligned} \tag{4.7}$$

4.2.2 MDP Reformulation

Since the exact values of the harvested energy levels and channel gains are unknown in the future, this problem cannot be solved using convex optimization techniques although the problem is convex.

MDP is characterized by its ability to provide a framework for decision making problems, where outcomes are partly random and partly under control. The mathematical model of an MDP is defined by the following principles: (a) A set of states $\mathcal{S}$. (b) A set of actions $\mathcal{A}$. (c) The transition probability model $p(s'|s,a)$, which is the probability of reaching state $s' \in \mathcal{S}$ given that action $a \in \mathcal{A}$ is taken at state $s \in \mathcal{S}$. (d) The immediate reward, $r(s,a,s')$, yielded by taking action a at state s and then transiting to state s' [4].

The problem in (4.7) is reformulated as an MDP [76], where each state s is defined by three elements, which are the battery level, channel gain, and amount of harvested energy (i.e. $s=(b,h,e)$). The action a is defined as the selected transmission power p^{Tx}. Each state s has a subset of actions $\mathcal{P}_s^{Tx}$ such that $\mathcal{P}_s^{Tx} \in \mathcal{P}^{Tx}$. Battery levels evolve according to

$$b' = \min\{b+e, B_{\max}\} - T_c\, p^{Tx} \tag{4.8}$$

The transition probability $p(s'|s,p^{Tx})$ is given by

$$p(s'|s,p^{Tx}) = \begin{cases} p_{\mathcal{E}_n}(e'|e)\cdot p_{\mathcal{H}}(h'|h), & \text{if (4.8) is satisfied} \\ 0, & \text{otherwise} \end{cases} \tag{4.9}$$

where the channel gain and EH processes are independent.

The immediate reward, which is the amount of received data resulting from taking action p^{Tx} at state s is given by

$$r(s,p^{Tx}) = T_c \log_2\left(1+\frac{p^{Tx}\,|h|^2}{\sigma_n^2}\right) \tag{4.10}$$

In the proposed system, the immediate reward is a function of the current state s and the selected action p^{Tx} only, and it is independent of the next state s'. It is important to note the difference between (4.2) and (6.3). (4.2) is the resulting data rate in terms of the state and action at time i, while (6.3) represents the resulting data rate in terms of the state and action spaces of the underlying MDP.

A deterministic policy π maps states into the transmission power taken at each state, $\pi(\cdot) : s \rightarrow p^{Tx}, \forall s$. The objective function is to maximize the expected cumulative throughput in (4.1) by finding an optimal policy π^*.

To evaluate different policies, value functions (state-value function $v_\pi(s)$ and action-value function $q_\pi(s, a)$) can be used. The optimal policy π^* has action-value function that is better than or equal to any other policy π for all states (i.e., $q_{\pi^*}(s, p^{Tx}) \geq q_\pi(s, p^{Tx}), \quad \forall s \in \mathcal{S}$) [38].

In this work, two problems are studied. The first one is when the source has causal knowledge about the states (i.e., knowledge about past and current states), the available actions at the current state, the immediate reward given an action, and the transition probabilities between states. The second problem is the same as the first but when the transition probabilities are unavailable.

Two approaches are used to deal with the considered problems. The first one is the look-ahead algorithm for EH communications, which is proposed to solve the first problem. This algorithm utilizes the available statistical knowledge to maximize the objective function. It is designed to avoid the complexity of the available methods used for solving such problems, such as VI. The second approach is RL, where the transition probabilities between states are unavailable. Two exploration algorithms for RL are used to evaluate and improve the performance of the proposed system.

4.3 Look-ahead Policy for EH Communications (known underlying model)

This proposed algorithm is a two-step look-ahead algorithm used when the statistical knowledge about the underlying model is available. This algorithm is broken down into a number of stages, as follows: Firstly, the overflow energy is computed. Then, the throughput using different transmission power levels are computed and compared. Finally, selecting a transmission power level based on the comparison from the previous step.

4.3.1 Two-step look-ahead throughput

The two-step look-ahead throughput is derived from the Bellman equation [4], and it is given by

$$\begin{aligned} R_{\text{l}}(s, p^{Tx}) =& r(s, p^{Tx}) + \gamma \sum_{s'} p(s'|s, p^{Tx'}) r(s', p^{Tx'}) \\ =& T_c \log_2 \left(1 + \frac{p^{Tx} |h|^2}{\sigma_n^2}\right) + \gamma \sum_{s'} p(s'|s, p^{Tx'})\, T_c \log_2 \left(1 + \frac{p^{Tx'} |h'|^2}{\sigma_n^2}\right) \end{aligned} \tag{4.11}$$

In this equation, the state value function v in the Bellman equation is replaced by the immediate reward r for one step only. This equation consists of two parts, the first one is the resulting throughput from using p^{Tx} in the current state s, while the second part is the expected throughput resulting from using $p^{Tx'}$ in the next slot s'.

4.3.2 Look-ahead throughput algorithm

The overflow energy is defined as the amount of energy that could be lost due to reaching the battery's maximum capacity. This results from harvesting, not utilizing the available energy, and using a limited size battery. Overflow situations should be avoided since they are not optimal, where a higher throughput can always be achieved if the overflow energy is utilized.

Given b, e, and $B_{\max}$, the overflow energy is written as

$$e_{\text{ovf}} = \max\{b + \min\{e, B_{\max}\} - B_{\max}, 0\} \tag{4.12}$$

In each time slot, the goal is to use at least this amount of energy regardless of the channel state. This is because this energy will be lost if it is not utilized.

The proposed algorithm depends on computing the two-step look-ahead throughput for different energy levels at each state. These energy levels are integer multiples of a fundamental energy unit. The first step is to find the set of all possible energy levels, Λ, that can be used at each state s. $\Lambda = \{\lambda_1, ..., \lambda_{N_\Lambda}\}$, where $\lambda_1 = e_{\text{ovf}}$, $\lambda_{N_\Lambda} = b$, and N_Λ is the total number of energy levels in Λ. Λ_M is a set of M random energy levels selected from Λ, $1 \leq M \leq N_\Lambda$. If the battery's maximum capacity is relatively small, Λ_M can contain all elements of Λ, i.e, $\Lambda_M = \Lambda$.

The two-step look-ahead throughput for each energy level in Λ_M is computed according to

$$R_{\rm l}(s,(\lambda_m/T_c)) = r(s,(\lambda_m/T_c)) + \gamma \sum_{s'} p(s'|s,(b'/T_c)) r(s',(b'/T_c)),\ m = 1,...,M \tag{4.13}$$

where b' is the battery level at the next state s', which depends on the used energy at state s (i.e., λ_m).

Based on the different values of $R_{\rm l}(s,(\lambda_m/T_c))$ at state s, the energy level is selected according to

$$\lambda_{\max} \leftarrow \arg\max_{\lambda_m} [R_{\rm l}(s,(\lambda_m/T_c))], \quad m = 1,...,M \tag{4.14}$$

where the selected action (i.e., the transmission power) at state s is $\lambda_{\max}/T_c$. Algorithm 4.1 summarizes the proposed algorithm.

Algorithm 4.1 Look-ahead Throughput Algorithm

for each $s \in \mathcal{S}$ **do**
  Compute the expected overflow energy $e_{\rm ovf}$.
  Find the set of all available energy levels at s, (i.e., Λ).
  Sample M random energy levels from Λ, and assign them to Λ_M.
  for each $m \in M$ **do**
    Compute $R_{\rm l}(s,(\lambda_m/T_c))$ using (4.13).
  end for
  $\lambda_{\max} \leftarrow \arg\max_{\lambda_m}[R_{\rm l}(s,(\lambda_m/T_c))]$, m=1,...,M.
  $p^{Tx} \leftarrow \lambda_{\max}/T_c$.
  $s \leftarrow s'$.
end for

4.4 RL for EH Communications (unknown underlying model)

This section provides a solution for the second scenario, where RL is used to handle the challenge of knowledge unavailability about the channel gain and EH processes. SARSA learning algorithm is used to evaluate different actions. The performance of the proposed model is investigated using two different exploration algorithms, which are the convergence-based algorithm, and the ϵ-greedy algorithm.

4.4.1 RL prediction methods

In this work, SARSA and Q-learning learning are used to predict the action-value function for different state-action pairs. SARSA is an on-policy updating strategy, which attempts

to evaluate the policy that is used to make decisions. On the other hand, Q-learning is an off-policy method, where the action-value function is estimated for the policy that is unrelated to the policy used for evaluation [4].

Updating in SARSA works as follows. Starting from time slot i, let the agent be at state s, and the selected action according to the current policy π is a. Based on the selected action, it moves to the next state s' and receives a reward $r(s, a, s')$. Using a policy derived from the $Q(s, a)$ (e.g., ϵ-greedy algorithm), an action a' is selected to the next state s'. At this point, the estimate of the action-value function, $Q(s, a)$, is updated using the gained experience. The updating equation in SARSA is given by [4]

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r(s,a,s') + \gamma\, Q(s',a') - Q(s,a)] \tag{4.15}$$

Using Q-learning, actions are assigned as follows. At the current state, actions are selected according to a policy derived from $Q(s, a)$ (e.g., ϵ-greedy algorithm), while the greedy action is assigned to the next state s'. The updating equation in Q-learning is given by [4]

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r(s,a,s') + \gamma \max_b Q(s',b) - Q(s,a)] \tag{4.16}$$

where $0 < \alpha < 1$ refers to the learning rate. This factor determines the amount of contribution of the newly acquired information for updating the action-value function. If $\alpha = 0$, then the agent will not learn any thing from the acquired information. On the other hand, if $\alpha = 1$, the agent will only consider the newly acquired information [89].

4.4.2 RL exploration algorithms

This part discusses two exploration algorithms for RL to deal with the case of knowledge unavailability about the underlaying model. The exploration algorithms play an essential role in RL. Their role appears in finding a balance between exploration and exploitation to maximize the cumulative rewards. The exploitation mode can be defined as using the current available knowledge to select the best policy to be used. On the other hand, exploration is known as investigating new policies in the hope of getting policy that is better than the current best one [4].

4.4.2.1 The ϵ-greedy algorithm

This algorithm [53] uses the exploration probability ϵ to find a balancing point between exploration and exploitation modes. This parameter changes the mode based on its value at each time slot.

In this algorithm, the current best action is selected with probability $1 - \epsilon$. On the other hand, a random non-greedy action is selected with probability ϵ. The ϵ can be either fixed [4], or with adaptive value during the learning time [36]. In the case of adaptive ϵ-greedy, ϵ takes values that changes with time. For example, in [36], ϵ is set to $e^{-0.1i}$, where i the time slot number. In this case, at the beginning of the session, the exploration probability ϵ has large values to increase the probability of exploration. As time increases, the probability of exploration decreases and the exploitation probability increases. This is to increase the opportunity of exploitation at the end of the session, where most of the policies have been explored and it is preferred to exploit the best known policy.

4.4.2.2 The convergence-based algorithm

This part presents our exploration algorithm. It uses two parameters to balance between exploration and exploitation. The first parameter is the action-value function convergence error ζ. The same action at a state is exploited for a number of iterations until the estimated value of this state-action pair converges to a value with an error less than or equal to ζ. The second parameter is the exploration time threshold τ. This parameter controls the exploration process, where the agent can explore different actions for a τ from the total available time T, after that, the agent is forced to exploit the best available policy π_{best} during the remaining time [90; 91].

In this algorithm, the first step is to assign random feasible actions to all available states. Then, for each visited state, the same action is selected for a time until its estimated value converges to a value determined by ζ. Once the estimated value of a state-action pair converges to a value with an error less than or equal to ζ, a new random action is assigned from uniformly distributed unexplored actions to that state. This mechanism continues for all states, and stops in two cases: The first one occurs if all available actions for a states s are evaluated before

reaching τ. At this time, the action with the best value $\pi_{\text{best}}(s)$ will be exploited in the future. The second case occurs when the available time reaches τ. Then, the agent suspends exploration, and starts exploiting the best available policy π_{best} regardless of exploring all available actions or not.

Using the SARSA with the convergence-based algorithm, an action for next state s' is selected according to the current policy π

$$p^{Tx'} \leftarrow \pi(s') \tag{4.17}$$

and for the case of integrating the Q-learning and convergence-based algorithms, an action is assigned to next state s' according to

$$p^{Tx'} \leftarrow \arg\max_{a} Q(s', a) \tag{4.18}$$

One of the main advantages of the convergence-based algorithm is that once an action at a state has been evaluated, and its action-value function has converged to an unfavorable value, this action will not be exploited in the future. This is an important property that contributes to discarding actions that may reduce the cumulative reward in the future. One more characteristic is that it assigns dynamic evaluation time for different actions at different states. This evaluation time depends on the required time by the estimated action-value function to converge for each state-action pair. Algorithm 4.2 summarizes the proposed algorithm.

Algorithm 4.2 Convergence-based Algorithm for estimating π^*

Initialize $Q^0(s, p^{Tx})$, $\forall s \in \mathcal{S}$, $\forall p^{Tx} \in \mathcal{P}_s^{Tx}$, arbitrarily
Initialize the action-value convergence error ζ, the exploration time threshold τ, and the learning rate α
Initialize $Q_{\text{best}}(s) = -\infty,\ \forall s \in S$
Initialize the policy π and the current best policy π_{best} by random actions $\forall s \in \mathcal{S}$
for each $s \in \mathcal{S}$ **do**
    $\pi_{\text{best}}(s), \pi(s) \leftarrow \varrho$
    $\mathcal{P}_s^{Tx} \leftarrow \mathcal{P}_s^{Tx} - \varrho$
end for
for each step i of episode **do**
    Observe current state S
    Select action P^{Tx} to state S according to the policy π (i.e., $P^{Tx} \leftarrow \pi(S)$)
    Observe the immediate reward $r(S, P^{Tx})$, and next state S'
    Predict $Q(S, P^{Tx})$ using a prediction method (e.g., SARSA or Q-learning)
    if $|Q^i(S, P^{Tx}) - Q^{i-1}(S, P^{Tx})| \leq \zeta$ AND $i < \tau$ **then**
        if $Q^i(S, P^{Tx}) \geq Q^{\text{best}}(S)$ **then**
            $Q_{\text{best}}(S) \leftarrow Q^i(S, P^{Tx})$
            $\pi_{\text{best}}(S) \leftarrow P^{Tx}$
        end if
        if $\mathcal{P}_S^{Tx} \neq \phi$ **then**
            Update π by selecting a new random action $\varrho \in \mathcal{P}_S^{Tx}$ to state S
            $\pi(S) \leftarrow \varrho$
            $\mathcal{P}_S^{Tx} \leftarrow \mathcal{P}_S^{Tx} - \varrho$
        else
            $\pi(S) \leftarrow \pi_{\text{best}}(S)$
        end if
    else if $i \geq \tau$ **then**
        $\pi \leftarrow \pi_{\text{best}}$
    end if
    $S \leftarrow S'$
end for

4.5 Complexity

Algorithm 4.1 aims at reducing the complexity of solving the formulated MDP problem, while approaching the optimal performance. Using the proposed algorithm, there is no need to go through all possible policies and select the optimal one, which is difficult especially when the system has a large number of actions/states combinations. For the case of using VI to get the optimal solution, the complexity is $\mathcal{O}(|\mathcal{A}| \cdot |\mathcal{S}|^2)$, where $\mathcal{A}$ is the set of actions, and $\mathcal{S}$ is the set of states for the problem [43]. On the other hand, the proposed algorithm has a complexity of $\mathcal{O}(|\mathcal{S}| \cdot |M|)$, where M is the number of sampled energy levels that are evaluated at each state.

For Algorithm 4.2, it aims at providing an efficient exploration algorithm for RL to improve the learning performance. This algorithm tries to estimate the values of different state-action pairs accurately, and then, exploit the best resulting policy. Let T is the final time step of an episode. The complexity of Algorithm 4.2 is $\mathcal{O}(|T|)$ when SARSA is used as a prediction method, and $\mathcal{O}(|\mathcal{A}| \cdot |T|)$ when Q-learning is used.

4.6 Simulation Results

In this section, the proposed algorithms are evaluated. Then, the effects of their parameters are investigated. To evaluate the proposed algorithms, three additional approaches are considered:

- Value iteration (VI) [42].
- Hasty Policy: At each time slot, all available energy is allocated for data transmission, regardless of previous experience. The goal is to avoid energy overflow situations [36].
- Random Policy: In this case, a set of feasible random transmission power levels is considered, where all levels are uniformly distributed across their range [36].

Two types of scenarios were considered in the simulation, simple scenarios that consider small numbers of states and actions, and scenarios with large numbers of states and actions. For simple scenarios such as in [38; 92], where the optimal policy can be found easily, VI was

used to evaluate the performance of the proposed algorithms. On the other hand, the proposed algorithms are compared with the hasty and random approaches only in the case of considering large number of states.

In the numerical analysis, it is assumed that each time slot is 1 second in duration. The available bandwidth BW is 1 MHz, and the noise spectral density is $N_0 = 4 \times 10^{-21}$ W/Hz.

It is also assumed that the S is equipped with solar panels with an area of 25 cm^2 and 10% harvesting efficiency. An outdoor solar panel can get the benefit of 100 mW/cm^2 solar irradiance under standard testing conditions, and harvesting efficiency between 5% and 30%, based on the used material in the panel [6]. It is assumed that the fundamental energy unit that can be harvested, stored, and transmitted is 0.05 J.

The used parameters were set as follows. The discount factor γ is set to 0.9, and the learning rate α is selected to be 0.1. Adaptive ϵ-greedy exploration algorithm is used [36]. For this algorithm, the exploration probability is set to $\epsilon = e^{-0.001i}$, where i is the time slot number in an episode. For the convergence-based exploration algorithm, ζ is set to 4, and the τ is set to be 0.8 of the total available time in an episode (i.e., $\tau = 0.8\,T$). For the throughput comparison step in the look-ahead algorithm, all possible energy levels at each state are considered, i.e., $\Lambda_M = \Lambda$.

It is also assumed that the set of harvested energy levels is $\mathcal{E}_n = \{0, 0.05, 0.1, 0.15, 0.2, 0.25\}$ J with transition probability matrix P_e

$$P_e = \begin{bmatrix} 0.4011 & 0.3673 & 0.1027 & 0.0899 & 0.0279 & 0.0111 \\ 0.4072 & 0.3441 & 0.1002 & 0.0973 & 0.0305 & 0.0207 \\ 0.3966 & 0.3239 & 0.1165 & 0.0860 & 0.0400 & 0.0370 \\ 0.3796 & 0.3272 & 0.1158 & 0.0782 & 0.0514 & 0.0478 \\ 0.3612 & 0.3451 & 0.1055 & 0.0837 & 0.0501 & 0.0544 \\ 0.3711 & 0.3341 & 0.1107 & 0.0801 & 0.0502 & 0.0538 \end{bmatrix}$$

The set of channel gains consists of 11 states that were selected between 0 and -20 dB with random transition probabilities between the states.

The used battery has a maximum capacity of 12 units. All results are averaged over 500 runs. The starting state is selected randomly, where all the states have equal probability to

be the starting state. The convergence-based and ϵ-greedy algorithms starts learning from the same policy, which is the Hasty policy. All mentioned parameters were used in all experiments unless otherwise stated.

4.6.1 Comparison with the upper bound

In this part, we evaluated the proposed algorithms by comparing them with the optimal performance. The VI was used to find the optimal policy to get the upper bound performance. The VI along with the look-ahead algorithms need a priori statistical knowledge about the channel gain and EH processes. On the other hand, this knowledge is unavailable for the learning approaches.

In this scenario, the battery maximum capacity $B_{\max}$ is set to 2 units. The set of harvested energy is $\mathcal{E}_n = \{0, 0.05\}$ J with transition probability matrix P_e

$$P_e = \begin{bmatrix} 0.5050 & 0.4950 \\ 0.5215 & 0.4785 \end{bmatrix}$$

The set of channel gains $\mathcal{H}_n = \{0, -10, -20\}$ dB with transition probability matrix P_h

$$P_h = \begin{bmatrix} 0.3946 & 0.3991 & 0.2064 \\ 0.4145 & 0.3470 & 0.2385 \\ 0.5524 & 0.3637 & 0.0838 \end{bmatrix}$$

Figure 4.2 shows the discounted return G_t (i.e., the cumulative discounted received data starting from time t). The cumulative discounted received data is defined as the amount of valuable data received within a given time frame. The discounted returns of the optimal policy and look-ahead algorithm take a near-constant pattern all the time. This is due to use one policy all the time, and the discount factor which bounds the discounted return to a value. For the learning approaches, in the beginning of the session, their discounted returns increase with experience, where these approaches start from hasty policy. As the time increases, they start taking a near-constant pattern, which results from learning a policy that cannot be improved any more, and the discount factor that bounds the discounted return to a value.

As shown, the upper-bound on the discounted return can be achieved by exploiting the optimal policy all the time. This figure also shows that the look-ahead algorithm outperforms

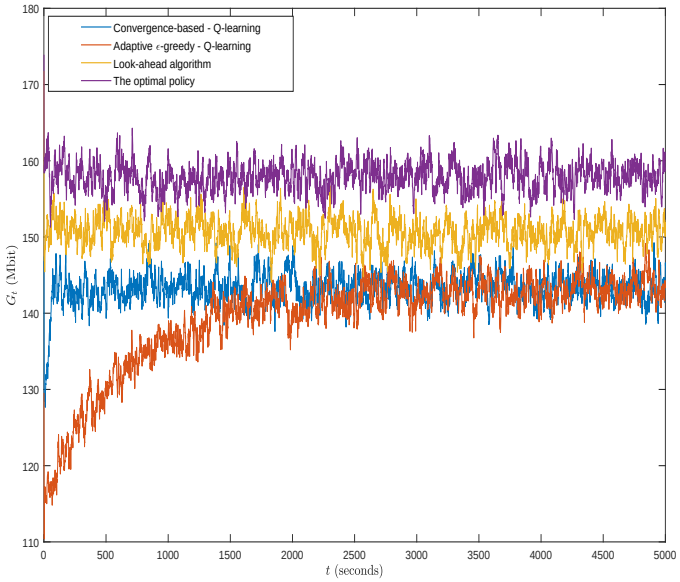

Figure 4.2: The discounted return G_t versus time t for different approaches.

the remaining approaches, which is due to exploiting the statistical knowledge that is available to this approach. It can also be noticed that the convergence-based algorithm outperforms the adaptive ϵ-greedy, where the convergence-based algorithm finds a suboptimal policy faster than the adaptive ϵ-greedy. The superiority of the convergence-based algorithm is attributed to its approach in evaluating different actions. The convergence-based algorithm evaluates the available actions based on their convergent values. This gives the source relatively accurate indications about the values of different state-action pairs, and enables it to determine and select a suboptimal policy in a relatively high-precision pattern.

On the other hand, the ϵ-greedy algorithm evaluates actions based on the instant values of state-action pairs, especially in the beginning of the learning process, when the exploration probability is relatively high and the values of different state-action pairs are unable to converge. Unfortunately, these instant values might not be the actual or near actual values of these pairs, which may slow down finding a suboptimal policy with actual high discounted return.

4.6.2 Comparison in large scenario

This part considers the case of large number of states. The goal is to examine the validity of the proposed algorithms in the case of large scenarios, where the number of states used in this

part is 858 states. Figure 4.3 shows the performance of the proposed approaches compared with the hasty and random algorithms, where finding the optimal policy is difficult. The discounted returns of the look-ahead, hasty, and random algorithms take a near-constant pattern all the time. This is because of using one policy all the time, and the discount factor that bounds the discounted return to a value.

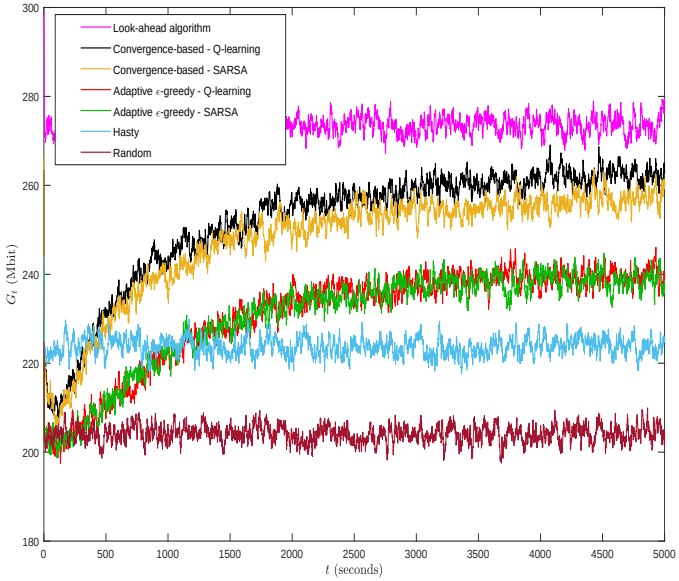

Figure 4.3: The discounted return G_t versus time t for different approaches.

Figure 4.3 shows that the look-ahead algorithm outperforms the other algorithms. This is due to having the statistical knowledge about the channel gain and EH processes, which enables the source to exploit a suboptimal policy from the beginning. It can also be noticed that the convergence-based algorithm outperforms the ϵ-greedy algorithm in terms of the speed of finding a suboptimal policy, and the quality of learned policies by each algorithm. This superiority is due to the used approach by each algorithm for evaluating different actions as explained in the previous subsection. For the hasty and random approaches, they do not exploit the available causal knowledge in exploring and exploiting different policies, which explains the relatively poor performance of these two approaches.

Using the convergence-based algorithm, it is clear that the Q-learning outperforms the SARSA insignificantly. Q-learning learns $Q(s, a)$ by approximating the optimal action-value function q^* directly. Fortunately, approximating the optimal action-value function has improved

the performance by finding a suboptimal policy in a shorter time compared to SARSA, even if this improvement is relatively small. On the other hand, SARSA is more conservative, it improves its performance using the estimate of the action-value function under the current policy. Although, SARSA uses a safer path, but this slows down finding a suboptimal policy and exploiting it early. Regarding to the adaptive ϵ-greedy, it can be seen that both Q-learning and SARSA have approximately the same performance, where approximating the optimal policy by Q-learning has not improved the performance.

4.6.3 RL algorithms - harvested energy levels with equal probabilities

This part considers another large scenario. It aims at investigating the considered RL exploration algorithms when the EH process is a process with independent and identically distributed random variables. The set of harvested energy levels is $\mathcal{E}_n = \{0, 0.05, 0.1, 0.15, 0.2, 0.25\}$ J, each with equal probability.

The considered exploration algorithms are compared using Q-learning. For the convergence-based algorithm, $\zeta = 4$ and τ has values that are changing between $0.2\,T$ and $0.6\,T$. The adaptive ϵ-greedy algorithm uses exploration probability changing between $\epsilon = \exp(0.1i)$ and $\epsilon = \exp(0.0001i)$.

Figure 4.4 shows the superiority of the convergence-based algorithm over hasty and ϵ-greedy algorithms. It can also be noticed that the best performance of the adaptive ϵ-greedy approximates the performance of the hasty policy, while the hasty outperforms the adaptive ϵ-greedy for the remaining values of ϵ. The superiority of the convergence-based algorithm and the poor performance of the adaptive ϵ-greedy algorithm are explained in the previous subsections.

4.6.4 Effect of the τ in Convergence-based algorithm

This experiment investigated the effect of the exploration time threshold τ on the performance of the convergence-based algorithm. In this experiment, ζ is set to 4.

Figure 4.5 shows the discounted return versus time. When $\tau = 0$, the performance takes a near constant pattern from the beginning, which is due to exploiting one policy all the time

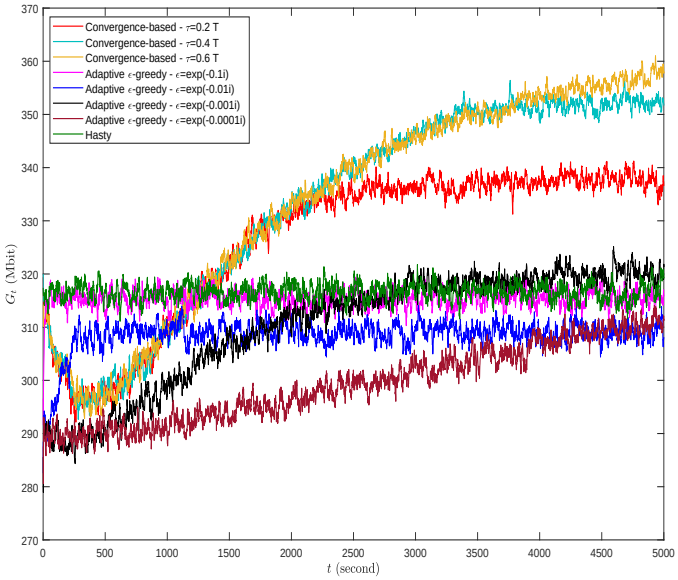

Figure 4.4: The discounted return G_t versus time t for different RL exploration algorithms.

(i.e., there is no exploration). For the remaining values of τ, the discounted returns increase with experience. Then, they take near-constant shapes, which is due to the discount factor effect, and the inability to improve the policy any more.

Figure 4.5 also shows that the discounted returns increase as τ increases up to a value, then saturation occurs. As the value of this threshold increases, the opportunity of exploring more policies increases, which also increases the opportunity of finding a good policy that increases the discounted return. After a certain time, the effect of increasing τ on the performance diminishes, which is due to assigning values for τ that are bigger than the required time for exploring all available actions. In this case, the source will be forced to exploit the best learned policy once it has evaluated all available actions regardless the value of τ.

4.6.5 Effect of the ζ in Convergence-based algorithm

In this experiment, the effect of ζ on the performance of the convergence-based algorithm was studied. The value of τ is set to be $0.8\,T$.

Figure 4.6 shows the influence of the experience on the discounted return at different values of ζ. As shown, for $\zeta = 0$, the performance has a near constant shape from the beginning, since there is no exploration. In this experiment it is difficult to achieve convergence with zero

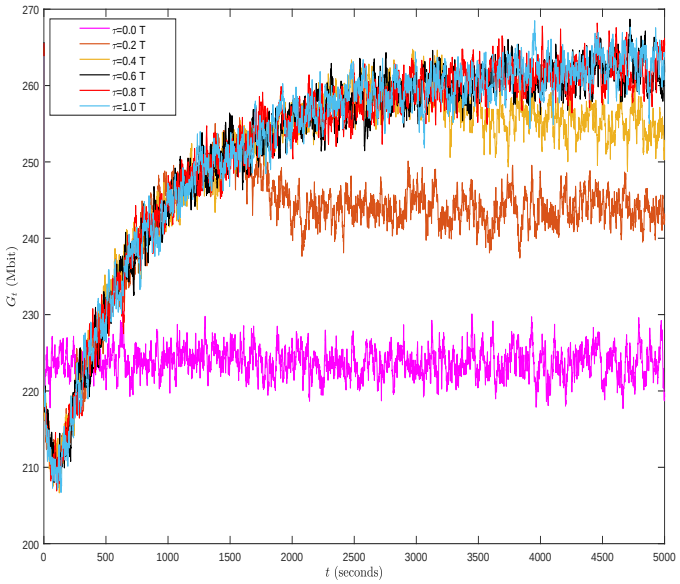

Figure 4.5: The discounted return G_t versus time t for different values of the τ.

error, which prevents exploration. For the remaining values of ζ, the performance is improved with experience. Then, the discounted returns take near-constant patterns, since the source is unable to improve the policy any more, and the discount factor which bounds the return. This figure also shows that the best performance is achieved when ζ has a value of 4.

It can also be noticed that the discounted return increases as the convergence error increases up to a certain value, and then starts decreasing. This is due to the fact that increasing the convergence error increases the opportunity of exploration, which improves the performance up to a certain value of ζ. After that, the performance starts to degrade, which is due to inaccurate evaluation of various actions.

4.6.6 Effect of the ϵ in ϵ-greedy algorithm

This part discusses the effect of ϵ on the performance of the ϵ-greedy algorithm. Figure 4.7 investigates the performance of the adaptive ϵ-greedy algorithm using different scenarios ($\epsilon = 0$, $\epsilon = e^{-0.1i}$, $\epsilon = e^{-0.01i}$, $\epsilon = e^{-0.001i}$, $\epsilon = e^{-0.0001i}$, and $\epsilon = 1$), where i is the time slot number in the episode. This figure shows that the discounted return when $\epsilon = 0$ remains constant approximately from the beginning, since there is no exploration. For the remaining values of ϵ, the performance is enhanced with experience, then, the discounted returns maintain

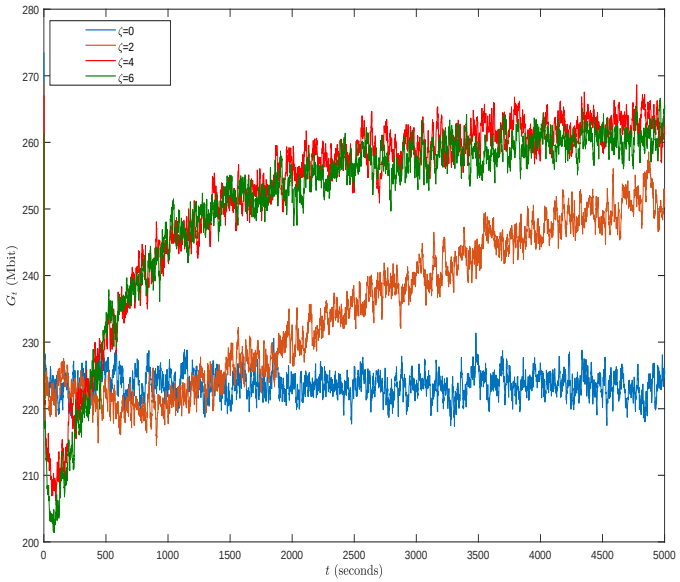

Figure 4.6: The discounted return G_t versus time t for different values of the ζ.

near-constant shapes due to the inability of improving the learned policy, and bounding the return by the discount factor. It can be noticed that the $\epsilon = e^{-0.001i}$ scenario outperforms the other scenarios.

This figure shows that slowing the decay of ϵ improves the performance up to a value, and then starts to degrade the performance. Decelerating decay of the ϵ means increasing the exploration probability at the beginning, which gives the source more opportunity to explore more policies and find a good policy. Increasing the exploration probability improves the performance up to a value, but then it starts to degrade the performance, which is due to slowing down exploiting the best resulting policy from the exploration.

4.7 Chapter Summary

In this chapter, two different scenarios for a realistic energy harvesting communication system were investigated. The first one assumes the availability of statistical knowledge about the channel gain and EH processes. On the other hand, the system in the second scenario does not have that knowledge. The source is equipped with an infinite data buffer to carry data packets and finite battery to store the harvested energy. We formulated the problem of maximizing the cumulative discounted received data as an MDP. For the first scenario, a

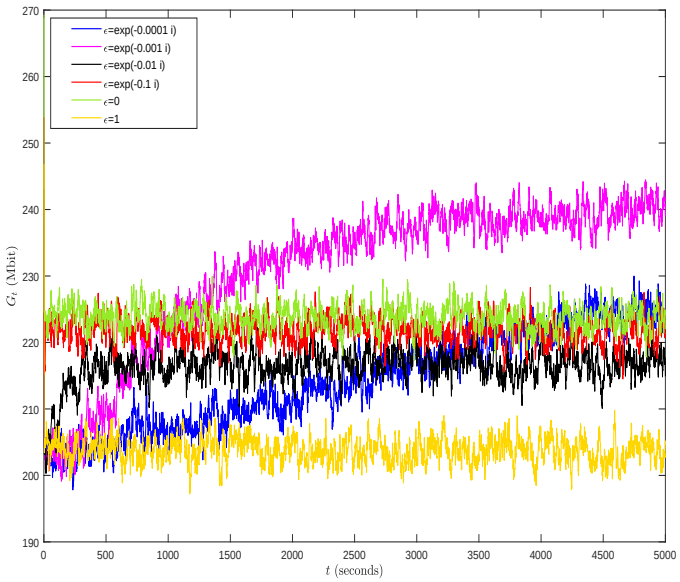

Figure 4.7: The discounted return G_t versus time t for different values of the ϵ.

look-ahead algorithm was designed to solve the problem efficiently by exploiting the availability of the transition probabilities between states. To optimize the performance of the system in the second scenario, RL was used. The results showed the effectiveness of the look-ahead algorithm when the statistical knowledge is available, even when the number of states and action is large. This work also showed the effectiveness of RL for optimizing the system performance in the case of unavailability of that knowledge. Two different exploration algorithms for RL were used, which are the convergence-based and ϵ-greedy algorithms. It was noticed that the convergence-based algorithm outperforms the other one. Finally, we discussed the effects of the parameters of each algorithm on the system performance. As a future work, function approximation and neural networks can be used along with RL to consider the case of having continuous state and action spaces.

CHAPTER 5. REINFORCEMENT LEARNING ARCHITECTURES

The first part of this chapter presents a number of introduced reinforcement learning (RL) architectures. These architectures aim at improving the RL field, and they are called selector-actor-critic (SAC), tuner-actor-critic (TAC), and estimator-selector-actor-critic (ESAC). These architectures are improved models of the well known architecture in RL called actor-critic (AC). In AC, an actor optimizes the used policy, while a critic is used to estimate a value function and evaluate the optimized policy by the actor. SAC is an architecture equipped with an actor, a critic, and a selector. The selector determines the most promising action at the current state based on the last estimate from the critic. Then, the actor uses the most promising action to optimize the policy. TAC consists of a tuner, a model-learner, an actor, and a critic. After receiving the approximated value of the current state-action pair from the critic and the learned model from the model-learner, the tuner uses the Bellman equation to tune the value of the current state-action pair. Then, this tuned value is used by the actor to optimize the policy. ESAC is proposed to implement intelligent agents that are able to estimate the values of all actions at the next state, and optimize its policy before experiencing an action. ESAC architecture is implemented based on two ideas, which are lookahead and intuition. Lookahead appears in collecting information about the available actions at the next state and estimating their values, while the intuition appears in maximizing the probability of selecting the most promising action. A number of elements are added to AC model to provide agents with these two capabilities. These elements are an underlying model learner, an estimator, and a selector. The model learner is used to approximate the dynamics of the underlying model. The estimator uses the approximated value function from the critic, the learned underlying model, and the Bellman equation to estimate the values of all actions at the next state. The selector is used to determine the most promising action at the next state, which will be used by the

actor to optimize the used policy. Lookahead capability is implemented using the interaction between the model-learner, the critic, and the estimator. While the actor and the selector are used to support the agent by the intuition capability. In the second part of this chapter, an energy harvesting (EH) point-to-point communications system working in an uncertain and unknown environment is investigated. The agent-environment interaction is modeled by a Markov decision process (MDP) with discrete state and action spaces. The considered EH communications system is supported by AC, SAC, TAC, and ESAC. The performance of the EH system is evaluated using the considered RL architectures.

The remainder of this chapter is organized as follows. Section 5.1 describes different RL architectures. The conventional AC learning is discussed in Subsection 5.1.1. The effect of adding a selector to the AC architecture is described in Subsection 5.1.2. Subsection 5.1.3 discusses the TAC architecture. Subsection 5.1.4 describes the main contribution in this chapter, which is the ESAC model. A brief discussion highlights main features of each investigated architecture is presented in Subsection 5.1.5. Section 5.2 describes and evaluates an EH communications system supported by the considered RL architectures. Finally, the paper is concluded in Section 5.3.

5.1 The Estimator-Selector-Actor-Critic Architecture

This section mainly aims at describing the architectures of AC, SAC, TAC, and ESAC. This section discusses the role of each component in each architecture. It also presents main features provided by each architecture.

5.1.1 Actor-Critic

This part discusses a well-known architecture in RL, which is called actor-critic or AC learning. AC refers to algorithms where the actor generates stochastic actions, and the critic estimates the value function and criticises the policy formatted by the actor [93]. Figure 5.1 [94] shows the interaction between the actor and the critic in the AC architecture.

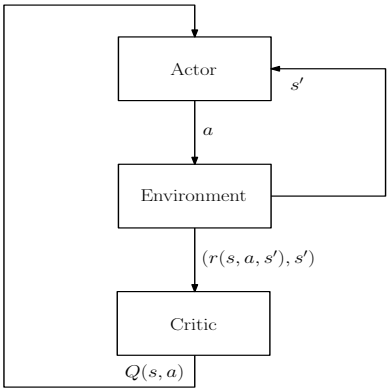

Figure 5.1: Actor-critic architecture.

In this context, the critic approximates the action-value function $q^{\pi}(s,a) \approx Q(s,a)$, and evaluates the currently optimized policy using SARSA, which is given by

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r(s,a,s') + \gamma Q(s',a') - Q(s,a)] \tag{5.1}$$

where α is the learning rate used to update Q, and γ is the discount factor.

The actor uses policy gradient to optimize a parameterized stochastic policy $\pi(a|s,\boldsymbol{\theta})$. Using policy gradient, the policy objective function $J(\boldsymbol{\theta})$ takes one of three forms, which are

- The value of the start state in episodic environments

$$J_1(\boldsymbol{\theta}) = V^{\pi}(s_1) \tag{5.2}$$

- The average value in continuing environments

$$J_{\text{avV}}(\boldsymbol{\theta}) = \sum_s d^{\pi}(s) V^{\pi}(s) \tag{5.3}$$

- The average reward per time-step in continuing environments also

$$J_{\text{avR}}(\boldsymbol{\theta}) = \sum_s d^{\pi}(s) \sum_a \pi(a|s,\boldsymbol{\theta}) r(s,a) \tag{5.4}$$

where $d^{\pi}(s)$ is the steady-state distribution of the underlying MDP using policy π, and $r(s,a,s')$ is the reward resulting from taking action a at state s and then transiting to state s'. The goal is to maximize $J(\boldsymbol{\theta})$ [59; 95]. The updating rule for $\boldsymbol{\theta}$ is given by

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \beta \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \tag{5.5}$$

where $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ is the gradient of $J(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$, and β is the step-size used to update the gradient of the policy.

One of the main challenges in this optimization problem is to ensure improvement during changing $\boldsymbol{\theta}$. This is because changing $\boldsymbol{\theta}$ changes two functions at the same time, which are the policy and the states' distribution. The other challenge is that the effect of $\boldsymbol{\theta}$ on the states' distribution is unknown, which makes it difficult to find the gradient of $J(\boldsymbol{\theta})$. Fortunately, policy gradient theorem provides an expression for the gradient of $J(\boldsymbol{\theta})$ that does not involve the derivative of the states' distribution with respect to $\boldsymbol{\theta}$ [59]. According to policy gradient theorem, for any differentiable policy and for any of the policy objective functions, the policy gradient is [95]

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx E_{\pi}[\nabla_{\boldsymbol{\theta}} \ln(\pi(a|s,\boldsymbol{\theta}))\, Q(s,a)] \tag{5.6}$$

Due to the difficulty of finding the gradient of $J(\boldsymbol{\theta})$, a stochastic estimate $\widehat{\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})}$ is used to approximate $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ [59; 96]. The new updating rule for $\boldsymbol{\theta}$ is given by

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \beta \widehat{\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})} \tag{5.7}$$

where

$$\widehat{\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})} = \nabla_{\boldsymbol{\theta}} \ln(\pi(a|s,\boldsymbol{\theta}))\, Q(s,a) \tag{5.8}$$

5.1.2 Selector-Actor-Critic

On-policy learning is defined as methods used to evaluate or improve the same policy used to make decisions. On the other hand, off-policy approaches try to improve or evaluate a policy different from the one that is used to generate data [59]. This section discusses an off-policy policy gradient, where the policy being followed is optimized using the most promising action at state s. The idea is to approximate the most promising action (i.e., the optimal action) at state s by the greedy action a_g. To the best of our knowledge, it is the first work using the most promising action a_g to optimize stochastic parameterized policies using policy gradient methods. The goal is to optimize the policy in the direction that maximizes the probability of selecting a_g, and increase the speed of learning a suboptimal $\boldsymbol{\theta}$. To achieve this goal, a

selector has been added to the conventional AC. This selector determines a_g at the current state greedily according to

$$a_g = \underset{b}{\operatorname{argmax}}\, Q(s, b), \quad \forall b \text{ at } s \tag{5.9}$$

where b indicates each possible action at state s.

After determining a_g by the selector, it is used by the actor to optimize the policy. The action a selected by the policy being followed in (6.8) is replaced by a_g. The new updating rule of $\boldsymbol{\theta}$ is given by

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \beta \nabla_{\boldsymbol{\theta}} \ln(\pi(a_g|s, \boldsymbol{\theta}))\, [Q(s, a_g)] \tag{5.10}$$

After selecting an action by the actor and interacting with the environment, the critic updates the action-value function according to

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r(s, a, s') + \gamma Q(s', a') - Q(s, a)] \tag{5.11}$$

Figure 5.2 shows the interaction between the components in the SAC model.

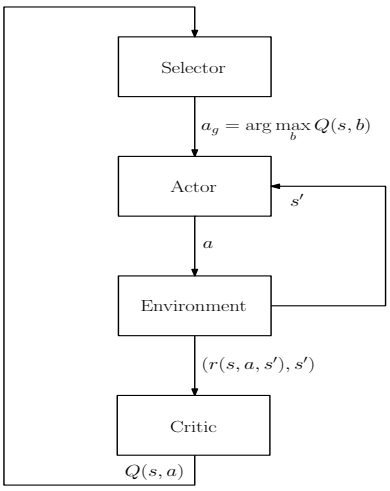

Figure 5.2: Selector-actor-critic architecture.

5.1.3 Tuner-Actor-Critic

Approximating the underlaying model, and using it with AC learning was discussed in [65]. The main idea in [65] is to use the learned model to control over the TD learning, and use TD to update the policy for exploring different actions. This section presents our modified

architecture, which is called tuner-actor-critic or TAC. TAC mainly aims at improving the learning process through integrating a tuner and a model-learner with AC. The main differences between TAC and the proposed model in [65] are concluded as follows.

- In [65], the critic approximates the state-value function to evaluate the system performance, while the critic in TAC approximates the action-value function.
- In [65], the policy uses a preference function for selecting actions, which indicates the preference of taking an action at a state. The preference function of the current state-action pair is updated by adding its old value to the current TD error learned by the critic. On the other hand, the actor in TAC uses stochastic parameterized policies to select actions, and uses policy gradient to optimize these policies.
- TAC uses the approximated underlying model, the approximated action-value function learned by the critic, and the Bellman equation to tune the value of the current state-action pair. In contrast, [65] uses the approximated underlying model to find the expected TD error for the current state. The value of the current state is updated by adding its previous value to the expected TD error.

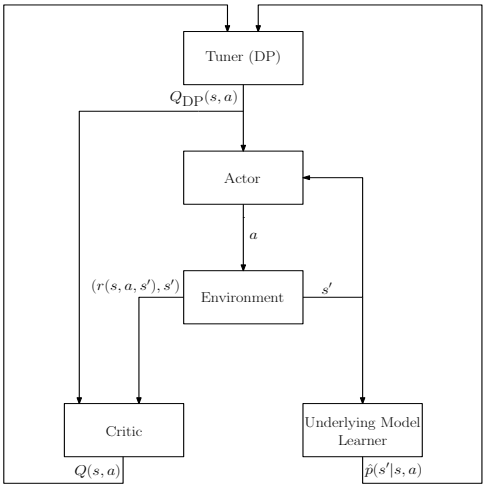

Figure 5.3: Tuner-actor-critic architecture.

As shown in Figure 5.3, the newly added components to AC architecture are the tuner and the model learner. Starting from the values received from the critic and the model learner, the

tuner uses the Bellman equation to tune the value of the state-action pair received from the critic. The tuner tunes the value of (s, a) state-action pair according to

$$Q_{\mathrm{DP}}(s,a) = \sum_{s'} \hat{p}(s'|s,a)\{r(s,a,s') + \gamma V(s')\} \tag{5.12}$$

where $\hat{p}(s'|s,a)$ is the approximated probability for transiting from the current state s to next possible state s' given action a is taken, and $V(s') = \sum_{a'} \pi(a'|s', \boldsymbol{\theta}) Q(s', a')$ is the approximated value of s'.

The critic replaces the value of the current state-action pair, (s, a), by the value computed by the tuner

$$Q(s,a) = Q_{\mathrm{DP}}(s,a) \tag{5.13}$$

The actor updates $\boldsymbol{\theta}$ using

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \beta \nabla_{\boldsymbol{\theta}} \ln(\pi(a|s,\boldsymbol{\theta}))\,[Q_{\mathrm{DP}}(s,a)] \tag{5.14}$$

After selecting an action and interacting with the environment, the critic evaluates the current policy using

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r(s,a,s') + \gamma Q(s',a') - Q(s,a)] \tag{5.15}$$

5.1.4 Estimator-Selector-Actor-Critic

This architecture aims at providing an intelligent agent. It enables agents to lookahead in unknown environments by estimating the values of the available actions at the next state, before optimizing the policy and taking an action. This enables agents to maximize the probability of selecting the most promising action, and minimize the probability of selecting dangerous actions before experiencing an action. This is the main contribution of this chapter, and the main property that distinguishes ESAC from AC, TAC, and SAC, which optimize the policy after experiencing actions. ESAC mainly consists of a model learner, estimator and selector, an actor, and a critic. Figure 5.4 shows the interaction between these components.

The tuner in TAC is renamed as estimator in ESAC. The reason for renaming this part is explained as follows. In TAC, this part is just used to tune the value of the current state-action

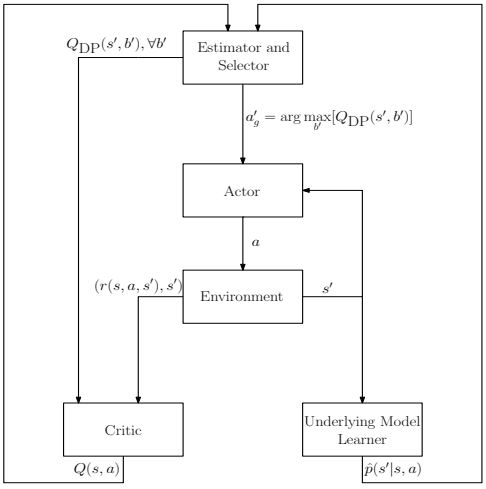

Figure 5.4: Estimator-selector-actor-critic architecture.

pair approximated by the critic. However, ESAC uses this part to estimate the values of all actions at the next state.

Using Bellman equation, and the last updates from the critic and the model learner, the estimator estimates the values of all the available actions at the next state s' according to

$$Q_{\mathrm{DP}}(s',b') = \sum_{s''} \hat{p}(s''|s',b')\{r(s',b',s'') + \gamma V(s'')\}, \quad \forall b' \quad \text{at} \quad s' \tag{5.16}$$

where s'' refers to next possibly reachable states from state s' given action b', and $\hat{p}(s''|s',b')$ is the approximated probability for transiting from s' to s'' given action b' is taken. Then, the selector determines the most promising action a'_g at s' to be used by the actor. The most promising action at s' is given by

$$a'_g = \underset{b'}{\mathrm{argmax}}[Q_{\mathrm{DP}}(s',b')], \qquad \forall b' \quad \text{at} \quad s' \tag{5.17}$$

The critic updates the values of actions at s' according to the last update from the estimator using

$$Q(s',b') = Q_{\mathrm{DP}}(s',b'), \qquad \forall b' \quad \text{at} \quad s' \tag{5.18}$$

The actor updates $\boldsymbol{\theta}$ according to

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \beta \nabla_{\boldsymbol{\theta}} \ln(\pi(a'_g|s',\boldsymbol{\theta}))\,[Q_{\mathrm{DP}}(s',a'_g)] \tag{5.19}$$

After selecting an action and interacting with the environment, the critic updates the action-value function according to

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r(s,a,s') + \gamma Q(s',a') - Q(s,a)] \tag{5.20}$$

5.1.5 Discussions

Section 5.1 discusses the investigated RL architectures. The first architecture is called actor-critic (AC). It mainly consists of an actor and a critic. The critic approximates a value function, and evaluates the selected actions by the actor. The actor uses a stochastic parameterized policy to select actions, and policy gradient to optimize the policy.

The second architecture is called selector-actor-critic (SAC). The newly added component is the selector. In AC architecture, the actor uses the currently selected action at the current state to optimize the policy's parameters. However, the selector in SAC determines the most promising action at the current state, which is used by the actor to optimize the policy's parameters.

The third scheme is called tuner-actor-critic (TAC). It has two more elements added to AC, which are a model learner and a tuner. The model learner approximates the dynamics of the underlying environment, while the tuner tunes the value of the current state-action pair using the Bellman equation, the learned model, and the learned value function by the critic. The actor uses the tuned value of the current state-action pair to optimize the policy's parameters.

The last model is estimator-selector-actor-critic (ESAC). The new components added to AC are a model learner, an estimator, and a selector. Before selecting an action, the estimator estimates the values of available actions at the next state using the Bellman equation, the learned model, and the learned value function. Then, the selector determines the most promising action at the next state, which is used by the actor to optimize the policy. This model mimics rational humans in the way of analyzing the available knowledge before taking an action. It aims at maximizing the cumulative rewards, and minimizing the probability of selecting bad and dangerous actions.

5.2 Energy Harvesting Communications Supported by Reinforcement Learning Architectures

5.2.1 System Model and Problem Formulation

The system model and the formulated problem are discussed in Section 4.1, and Section 4.2, respectively. In this chapter, policy gradient is used to optimize a parameterized stochastic policy $\pi(a|s,\boldsymbol{\theta})$. When policy gradient is used, the policy is evaluated using one of the functions defined in subsection 5.1.1 instead of the action-value or the state-value functions. The task is a continuing task, and the average value of the states is used to evaluate the used policy. The average value of the states is given by

$$J(\boldsymbol{\theta}) = \sum_{s} d^{\pi}(s) \sum_{p^{Tx}} \pi(p^{Tx}|s,\boldsymbol{\theta})\, q_{\pi}(s,p^{Tx}) \tag{5.21}$$

5.2.2 EH Communications System Supported by RL Architectures

In this part, the concepts of AC, TAC, SAC, and ESAC are applied to the considered EH communications system to improve its performance, and to evaluate the considered RL architectures. This part summarizes the equations used by each component in each architecture.

EH communications system supported by AC

The actor-critic cycle is described as follows:

A) The critic approximates the action-value function and evaluates the current policy using SARSA

$$Q(s,p^{Tx}) \leftarrow Q(s,p^{Tx}) + \alpha[r(s,p^{Tx}) + \gamma Q(s',p^{Tx'}) - Q(s,p^{Tx})] \tag{5.22}$$

where the immediate reward is a function of (s,p^{Tx}) pair only in the considered model, and it does not depend on the next state s'.

B) The actor optimizes the policy according to

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \beta \nabla_{\boldsymbol{\theta}} \ln(\pi(p^{Tx}|s,\boldsymbol{\theta}))\, Q(s,p^{Tx}) \tag{5.23}$$

EH communications system supported by SAC

The selector-actor-critic cycle is described as follows:

A) The critic approximates the action-value function and evaluates the current policy using SARSA

$$Q(s, p^{Tx}) \leftarrow Q(s, p^{Tx}) + \alpha[r(s, p^{Tx}) + \gamma Q(s', p^{Tx'}) - Q(s, p^{Tx})] \tag{5.24}$$

B) The selector determines the most promising transmission power level at the current state

$$p_g^{Tx} = \underset{c}{\operatorname{argmax}}\, Q(s, c), \quad \forall c \text{ at } s \tag{5.25}$$

where c indicates each possible transmission power level at state s.

C) The actor optimizes the policy according to

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \beta \nabla_{\boldsymbol{\theta}} \ln(\pi(p_g^{Tx}|s, \boldsymbol{\theta}))\, Q(s, p_g^{Tx}) \tag{5.26}$$

EH communications system supported by TAC

The tuner-actor-critic cycle is described as follows:

A) The critic approximates the action-value function and evaluates the current policy using SARSA

$$Q(s, p^{Tx}) \leftarrow Q(s, p^{Tx}) + \alpha[r(s, p^{Tx}) + \gamma Q(s', p^{Tx'}) - Q(s, p^{Tx})] \tag{5.27}$$

B) Using the learned action-value function Q from the critic and the approximated underlying model $\hat{p}(s'|s, p^{Tx})$, the tuner tunes the value of the current state-action pair using

$$Q_{\mathrm{DP}}(s, p^{Tx}) = \sum_{s'} \hat{p}(s'|s, p^{Tx}) \{r(s, p^{Tx}) + \gamma V(s')\} \tag{5.28}$$

where $V(s') = \sum_{p^{Tx'}} \pi(p^{Tx'}|s', \boldsymbol{\theta}) Q(s', p^{Tx'})$ is the approximated value of s'.

C) The actor optimizes the policy according to

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \beta \nabla_{\boldsymbol{\theta}} \ln(\pi(p^{Tx}|s, \boldsymbol{\theta}))\, [Q_{\mathrm{DP}}(s, p^{Tx})] \tag{5.29}$$

D) The critic replaces the value of the current state-action pair, (s, p^{Tx}), by the value computed by the tuner

$$Q(s, p^{Tx}) = Q_{\mathrm{DP}}(s, p^{Tx}) \tag{5.30}$$

EH communications system supported by ESAC

The estimator-selector-actor-critic cycle is described as follows:

A) The critic approximates the action-value function and evaluates the current policy using SARSA

$$Q(s, p^{Tx}) \leftarrow Q(s, p^{Tx}) + \alpha[r(s, p^{Tx}) + \gamma Q(s', p^{Tx'}) - Q(s, p^{Tx})] \tag{5.31}$$

B) Using the learned action-value function Q from the critic and the approximated underlying model $\hat{p}(s'|s, p^{Tx})$, the estimator estimates the values of all actions at the next state

$$Q_{\mathrm{DP}}(s', c') = \sum_{s''} \hat{p}(s''|s', c')\{r(s', c') + \gamma V(s'')\}, \quad \forall c' \text{ at } s' \tag{5.32}$$

where c' indicates each possible transmission power level at state s', and s'' refers to next possibly reachable states from state s' given action c'.

C) The selector determines the most promising transmission power level at next state s'

$$p_g^{Tx'} = \underset{c'}{\mathrm{argmax}}[Q_{\mathrm{DP}}(s', c')], \quad \forall c' \text{ at } s' \tag{5.33}$$

D) The actor optimizes the policy according to

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \beta \nabla_{\boldsymbol{\theta}} \ln(\pi(p_g^{Tx'}|s', \boldsymbol{\theta}))\,[Q_{\mathrm{DP}}(s', p_g^{Tx'})] \tag{5.34}$$

E) The critic updates the values of actions at s' according to the last update from the estimator using

$$Q(s', c') = Q_{\mathrm{DP}}(s', c'), \qquad \forall c' \text{ at } s' \tag{5.35}$$

5.2.3 Exponential Softmax Policy

In this work, the exponential softmax distribution [59] is used as a stochastic policy for selecting actions at each state. The policy is given by

$$\pi(p^{Tx}|s, \boldsymbol{\theta}^s_{p^{Tx}}) = \frac{\exp(h(s, p^{Tx}, \boldsymbol{\theta}^s_{p^{Tx}}))}{\sum_c \exp(h(s, c, \boldsymbol{\theta}^s_c))} \tag{5.36}$$

where $\exp(\cdot)$ is the base of the natural logarithm, $h(s, p^{Tx}, \boldsymbol{\theta}^s_{p^{Tx}}) \in \mathbb{R}$ is the parameterized preference for (s, p^{Tx}) state-action pair, and $\boldsymbol{\theta}^s_{p^{Tx}}$ is the policy's parameter vector related to

action p^{Tx} at state s. For discrete and small action spaces, the parameterized preferences can be allocated for each state-action pair [59].

The parameterized preference of (s, p^{Tx}) state-action pair is a function of a vector of features $\boldsymbol{\phi}(s, p^{Tx})$ and $\boldsymbol{\theta}^s_{p^{Tx}}$, which is used to determine the preference of action p^{Tx} at state s. The action with the highest preference at a state will be selected with the highest probability, and so on [59]. These preferences can take different forms. One of these forms is that when the preference is a linear function of weighted features, which is given by

$$h(s, p^{Tx}, \boldsymbol{\theta}^s_{p^{Tx}}) = (\boldsymbol{\theta}^s_{p^{Tx}})^\top \boldsymbol{\phi}(s, p^{Tx}) \tag{5.37}$$

The $\nabla_{\boldsymbol{\theta}^s_{p^{Tx}}} \ln(\pi(p^{Tx}|s, \boldsymbol{\theta}^s_{p^{Tx}}))$ is given by

$$\begin{aligned} \nabla_{\boldsymbol{\theta}^s_{p^{Tx}}} \ln(\pi(p^{Tx}|s, \boldsymbol{\theta}^s_{p^{Tx}})) &= \frac{\nabla \pi(p^{Tx}|s, \boldsymbol{\theta}^s_{p^{Tx}})}{\pi(p^{Tx}|s, \boldsymbol{\theta}^s_{p^{Tx}})} \\ &= \boldsymbol{\phi}(s, p^{Tx}) - \pi(p^{Tx}|s, \boldsymbol{\theta}^s_{p^{Tx}})\, \boldsymbol{\phi}(s, p^{Tx}) \end{aligned} \tag{5.38}$$

Feature functions $\boldsymbol{\phi}(s, p^{Tx})$ of (s, p^{Tx}) state-action pair is used for representing the states and actions in an environment. Feature functions should correspond to aspects of state and action spaces, where the generalization can be implemented properly [59]. This work uses binary feature functions. Feature function for a state-action pair is set to one if the action satisfies the energy feasibility condition at state s, otherwise, it is set to zero.

5.2.4 Experimental Results

This part discusses the validity of supporting EH communications systems with the investigated RL architectures, when these systems are working in uncertain and unknown environments.

Experimental Set-up. Two scenarios were considered in the simulation; a simple scenario with small number of states, and a scenario with large number of states. The simple scenario is used to evaluate and compare the investigated RL architectures with the optimal performance. For the large scenario, the considered architectures are only compared with each other, where it is difficult to find the optimal solution.

In all scenarios, the discount factor γ is set to 0.9. The learning rate α used by the critic is set to 0.1. The step-size learning parameter β used in policy gradient is set to 0.1. All the simulations started with an initial policy selecting available actions with equal probabilities. The approximated transition model was initialized with zero transition probabilities.

In the simulated environments, the simple scenario is modeled by an MDP with 18 states. Three actions are available with different immediate rewards and random transition probabilities. The second scenario is modeled using 858 states, and the available actions are 13. All the results were averaged over 500 runs. The starting state is selected randomly, where all the states have equal probabilities to be the starting state. All mentioned parameters were used in all experiments unless otherwise stated.

Comparison with the upper bound. In this scenario, the battery maximum capacity $B_{\max}$ is set to 2 units. The set of harvested energy is $\mathcal{E}_n = \{0, 0.05\}$ J with transition probability matrix P_e

$$P_e = \begin{bmatrix} 0.5050 & 0.4950 \\ 0.5215 & 0.4785 \end{bmatrix}$$

The set of channel gains $\mathcal{H}_n = \{0, -10, -20\}$ dB with transition probability matrix P_h

$$P_h = \begin{bmatrix} 0.3946 & 0.3991 & 0.2064 \\ 0.4145 & 0.3470 & 0.2385 \\ 0.5524 & 0.3637 & 0.0838 \end{bmatrix}$$

In this experiment, the total rewards G_0 and the discounted return G_t of each architecture were evaluated. The optimal performance uses the optimal policy from the beginning. It requires a priori statistical knowledge about the environment, which is unavailable to the remaining RL architectures. Value iteration (VI) was used to find the optimal policy to find the upper-bound [42].

Figure 5.5 shows the resulting discounted return G_t versus t for the considered architectures. As expected, the discounted return of the optimal policy takes a near-constant pattern all the time. This is due to using one policy all the time, and the discount factor γ which bounds the discounted return to a certain value. The discounted returns of the RL architectures increase

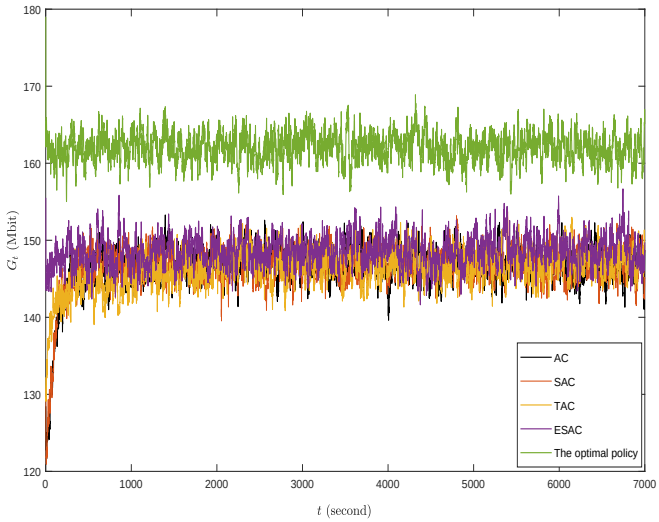

Figure 5.5: The cumulative discounted return G_t versus t.

with experience significantly, in the beginning. As the time increases, they start taking a near-constant pattern, which results from learning a policy that they could not improve any more, and γ that bounds the discounted return to a particular value. As shown, ESAC has found a suboptimal policy before AC, TAC, and SAC. Explanations for these results are summarized as follows. AC, SAC, and TAC are risky architectures, and they do not have accurate estimations about actions in the beginning. They need to experience different actions to get accurate estimations about their values and optimize their policies. This means experiencing different actions including low-value actions that result in relatively low discounted returns. AC experiences an action at the current state, and then, it optimizes the policy based on the estimated value of the current state-action pair. TAC just tunes the estimated value of the experienced action using the learned underlying model, then, this tuned value is used by the actor to optimize the policy. It is clear that both AC and TAC do not exploit the available information about the remaining actions at the current state to optimize the policy. SAC experiences an action at the current state, and then, based on its estimated value and the estimated values of other actions, it optimizes the policy. The superiority of ESAC in finding a suboptimal policy before the remaining approaches without taking a risky path is due to its capability to utilize information from other states, and use this information to estimate the most promising action at next state before optimizing the policy and experiencing an action.

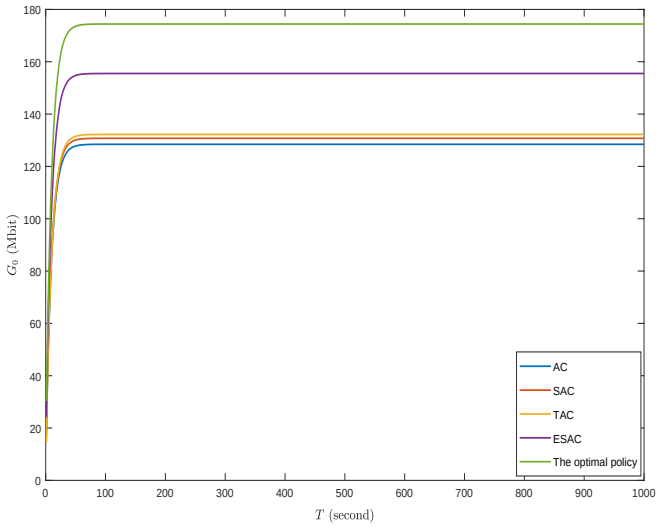

Figure 5.6: The total rewards G_0 versus T.

Figure 5.6 shows the resulting total rewards G_0 versus T using different architectures. As expected, the total rewards of all approaches increase significantly as T increases, and then, they take a near-constant performance. This is due to the discount factor γ, which diminishes the effect of the future rewards with time. As shown, ESAC outperforms AC, TAC, and SAC. This is because ESAC takes a safer path for optimizing its policy, which enables it from finding a suboptimal policy from the beginning where γ^i is relatively high, and the total rewards can be increased significantly. On the other hand, AC, SAC, and TAC take a risky path from the beginning, which prevents them from finding a suboptimal policy in an early time when γ^i is relatively high.

Comparison in large scenario. This part considers the case of a large number of states. The goal is to evaluate the considered architectures when the number of states is large. It is also assumed that the set of harvested energy levels is $\mathcal{E}_n = \{0, 0.05, 0.1, 0.15, 0.2, 0.25\}$ J, each with equal probability. The set of channel gains consists of 11 states that were uniformly selected between 0 and -20 dB with random transition probabilities between the states. The used battery has a maximum capacity of 12 units. All results are averaged over 500 runs.

Figure 5.7 shows the discounted return G_t versus t for ESAC, SAC, TAC, and AC. It shows that ESAC is able to find a suboptimal policy before the remaining architectures. On the

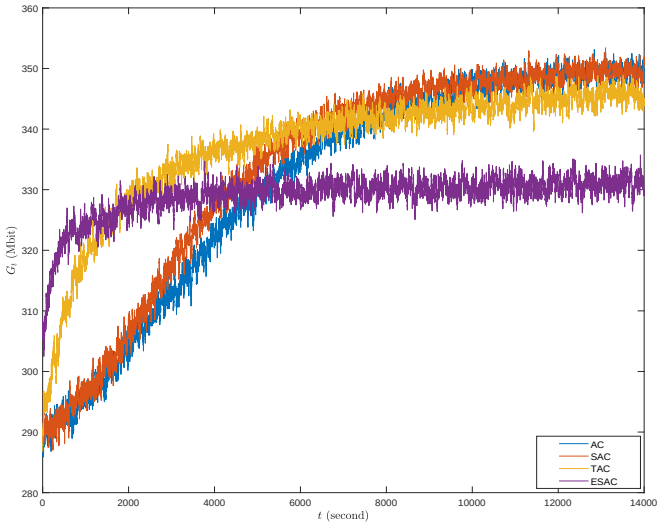

Figure 5.7: The cumulative discounted return G_t versus t when $\gamma = 0.9$.

other hand, AC, SAC, and TAC are able to find better policies compared to ESAC, which is attributed to their risky behavior, which enables them to find better policies compared to ESAC, at the end. This figure also shows that the best learned policies are the ones learned by AC and SAC.

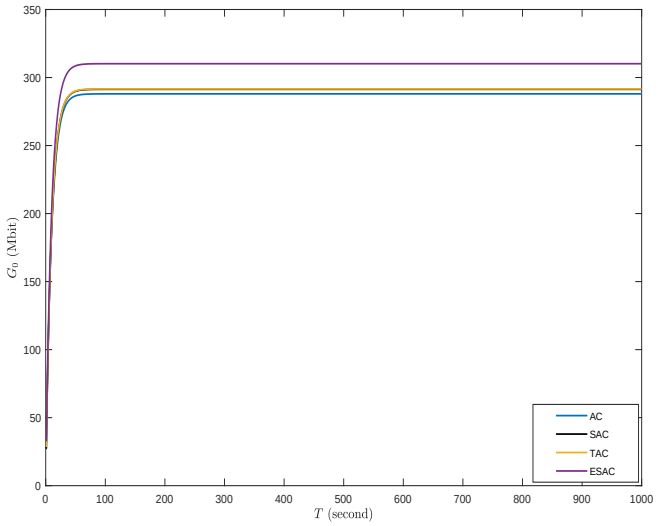

Figure 5.8: The total rewards G_0 versus T when $\gamma = 0.9$.

Figure 5.8 shows the total rewards G_0 versus T for ESAC, SAC, TAC, and AC. It also shows the superiority of ESAC in terms of G_0 compared to its competitors even in the case of large number of states. This is due to the ESAC's tendency to take a safe path from the beginning,

which enables it from collecting relatively high rewards when γ^i is relatively high. Regarding to AC, SAC, and TAC, they prefer taking risky paths to find a good policy. However, this prevents these architectures from collecting relatively high rewards in the beginning, when γ^i is relatively high.

Zero discount factor. In this part, we investigate the performance of the considered architectures when the discount factor $\gamma = 0$. In this case, the agent should care about which action will yield the largest expected immediate reward. The remaining parameters are the same parameters used in the previous part.

Figure 5.9 shows the return G_t versus t for ESAC, SAC, TAC, and AC when $\gamma = 0$. Due to the large fluctuation in Figure 5.9, it is reploted using a moving average filter as shown in Figure 5.10. Figure 5.10 shows the same performance trends and relative performance in terms of G_t when $\gamma = 0.9$. Since there is no accumulation of the rewards, and only the first reward R_{t+1} is included, G_t values in Figure 5.10 are much less than those in Figure 5.7.

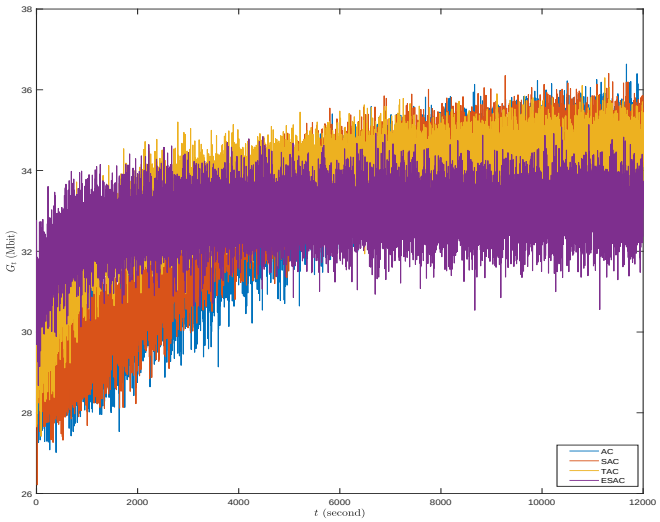

Figure 5.9: The return G_t versus t when $\gamma = 0.0$.

For the case of $\gamma = 0$, G_0 is the only reward resulting in the first time slot (i.e., R_1), which means that G_0 will have a fixed value regardless the value of T.

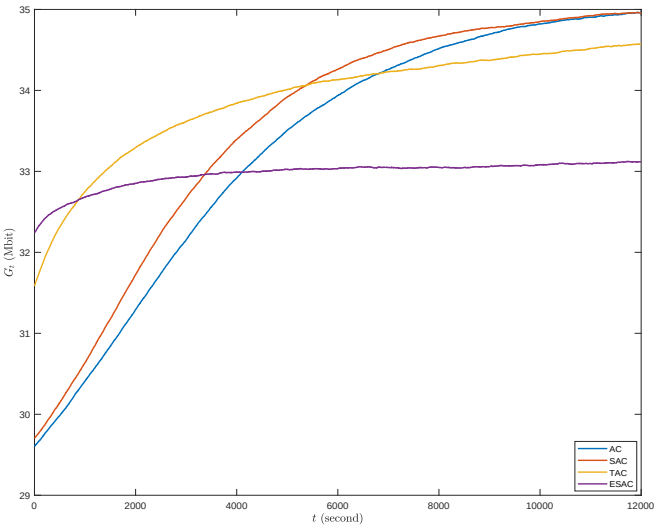

Figure 5.10: The fitted return G_t versus t when $\gamma = 0.0$.

No discounting. This part studies the performance of the considered architectures when there is no discounting on the rewards (i.e., the discount factor $\gamma = 1$), where the return does not converge to a certain value. In this case, the agent should care about maximizing the expected sum of all future rewards. The remaining parameters are the same parameters used in the previous part.

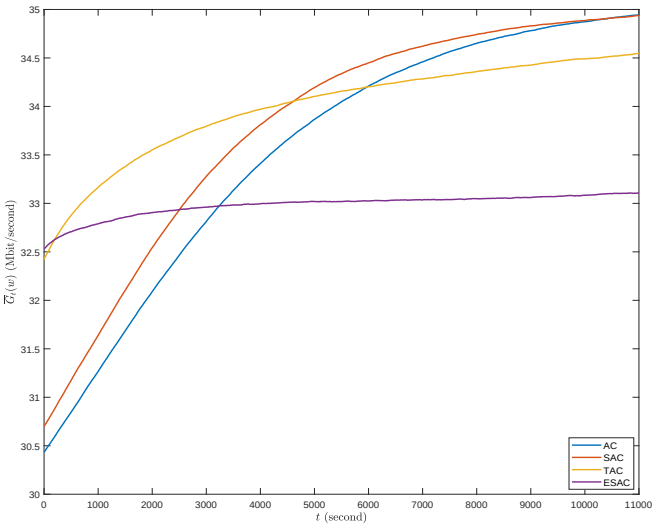

Figure 5.11: The average return $\overline{G}_t(w)$ versus t when $\gamma = 1.0$.

Figure 5.11 shows the average return, $\overline{G}_t(w) \triangleq \frac{1}{w}\sum_{i=0}^{w-1}\gamma^i R_{i+t+1}$, versus t for the investigated architectures. Due to the difficulty of finding the return G_t for this case, $\overline{G}_t(w)$ is

calculated and averaged over $w = 4000$ time steps for each time t. This figure also shows the superiority of AC and SAC compared to TAC and ESAC in terms of the policies that can be learned by each architecture.

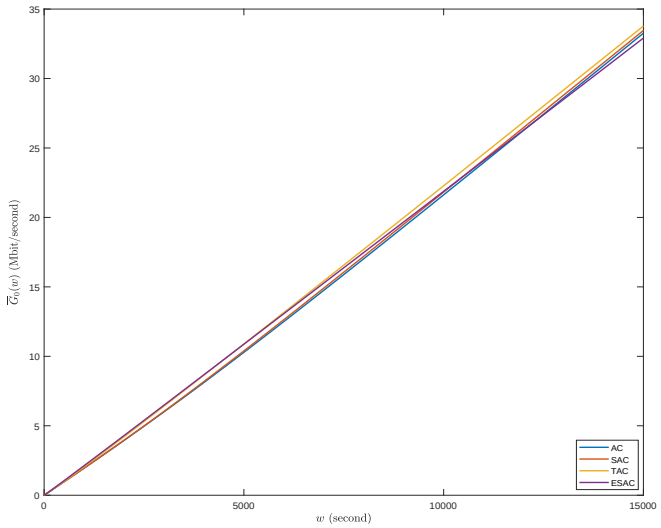

Figure 5.12: The average reward $\overline{G}_0(w)$ versus w when $\gamma = 1.0$.

Figure 5.12 shows the average reward, $\overline{G}_0(w) \triangleq \frac{1}{w} \sum_{i=0}^{w-1} \gamma^i R_{i+1}$, versus w for ESAC, TAC, SAC, and AC. It also shows the superiority of TAC in terms of the average reward compared with the remaining architectures, and the ability of SAC and AC to learn better policies compared to ESAC as time increases.

5.3 Chapter Summary

In this chapter, new architectures for RL were introduced. These architectures are called SAC, TAC, and ESAC. They aim at improving RL by adding components to the conventional AC. The main idea of SAC is to improve the learning process, and accelerate optimizing the policy's vector $\boldsymbol{\theta}$. The main difference between AC and SAC is in the updating equation of $\boldsymbol{\theta}$. AC uses on policy policy gradient, where selected action by the current policy is used to update $\boldsymbol{\theta}$, while SAC uses off-policy policy gradient, it updates $\boldsymbol{\theta}$ using the most promising action (i.e., the approximately optimal action). TAC aims at improving the learned value function by adding a model-learner and a tuner. The tuner tunes the approximated value of the current state-action pair using the learned underlying model and the Bellman equation. The goal of

developing ESAC is to provide an RL architecture for intelligent agents that mimics human in the way of making decisions. It aims at maximizing the probability of selecting promising actions, and minimizing the probability of selecting dangerous and expensive actions before experiencing actions. It takes a safe path in optimizing its policy. The ESAC architecture uses two ideas, which are the lookahead and intuition, to implement such agents with high level of understanding and analyzing available information before making decisions. The lookahead appears in collecting information from the model learner and the critic to estimate the values of all available actions at next state. The intuition is seen in optimizing the policy to maximize the probability of selecting the most promising action. To maintain exploration during learning, ESAC does not select the most promising action each iteration, it just maximizes its probability of being selected. AC, SAC, TAC, and ESAC were evaluated using an EH communications system working in uncertain and unknown environment, when this system is supported by these architectures. Simulation results show that ESAC outperforms other potential competitors in terms of total rewards collected through learning. This is due to the conservative behavior of ESAC, which prefers to take a safer path from the beginning compared to the remaining architectures. AC, SAC, and TAC experience an action, and then, they optimize their policies based on the value of the experienced action. Due to their behavior in taking risky paths, they are able to explore more policies, which explains their ability to find better policies at the end compared with ESAC. This dangerous behavior might be unwanted in some applications, where experiencing dangerous actions to evaluate them is expensive such as learning robots and drones.

CHAPTER 6. AN ACTOR-CRITIC REINFORCEMENT LEARNING APPROACH FOR ENERGY HARVESTING COMMUNICATIONS SYSTEMS

In this chapter, an energy harvesting (EH) point-to-point communications system working in an uncertain and unknown environment is investigated. This system consists of a source and destination. The source is an EH node that is able to harvest solar energy and store it in a finite capacity battery. The EH and channel gain processes are Markov processes. The underlying agent-environment interaction is modeled by a Markov decision process (MDP) with continuous state and action spaces. The goal is to optimize the transmission power allocation policy to maximize the expected cumulative throughput. Due to lack of knowledge about the underlying processes, actor-critic reinforcement learning (RL) is used. The critic uses a neural network of three layers to approximate the action-value function and evaluate the policy approximated by the actor. The actor uses a stochastic parameterized policy modeled by a normal distribution with parameterized mean and standard deviation. Policy gradient is used to optimize the policy's parameters to select actions maximizing the expected cumulative throughput.

The remainder of this chapter is organized as follows. Section 6.1 describes the proposed communications system model. Then, the problem is formulated in Section 6.2. Section 6.3 presents the architecture of actor-critic learning used in this work. Numerical simulation results are presented in Section 6.4. Finally, this chapter is concluded in Section 6.5.

6.1 System Model

In this section, a point-to-point communication system that consists of a source (S) and a destination (D) is investigated. As illustrated in Figure 6.1, each of S and D is equipped with

an infinite buffer to store data. S is an energy harvesting node that is able to harvest solar energy and store it in a finite capacity battery. A time slotted system with equal length time slots is assumed, where each slot has a duration of T_c. The maximum capacity of the battery is $B_{\max}$ J. B_i is the battery level of S at the beginning of time slot i, where $B_i \in \mathcal{B} \triangleq [0, B_{\max}]$.

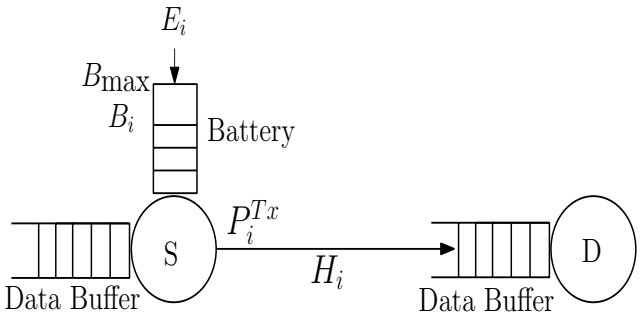

Figure 6.1: Point-to-point communication system with an energy harvesting source.

The EH and channel gain between slots are governed by Markov processes. During time slot i, S is harvesting E_i J from solar sources, where $E_i \in \mathcal{E}_n \triangleq [E_{\min}, E_{\max}]$. $f_{\mathcal{E}_n}(e'|e)$ is the transition probability density function for transiting from energy level e to energy level e'. Let H_i be the channel gain from S to D during time slot i, where $H_i \in \mathcal{H} \triangleq [H_{\min}, H_{\max}]$. $f_{\mathcal{H}}(h'|h)$ is the transition probability density function for transiting from channel gain h to channel gain h'.

Let P_i^{Tx} be the transmission power during time slot i. $P_i^{Tx} \in \mathcal{P}^{Tx} \triangleq [0, p_{\max}^{Tx}]$. In this model, energy consumption is considered only due to data transmission, and it does not take into account any other energy consumption, such as processing, circuitry, etc. P_i^{Tx} is the decision variable that will be determined in order to maximize the amount of data transmitted from S to D. In this work, harvest-store-use scheme is used to manage the harvested energy [87; 88].

6.2 Problem Formulation

Since the exact values of the harvested energy levels and channel gains are unknown in the future, the optimization problem 4.7 presented in subsection 4.2.1 cannot be solved using convex optimization techniques. This section presents an MDP reformulation of this optimization

problem, when the state and action spaces are continuous. The mathematical model of an MDP with continuous state and action spaces is defined by the following principles:

- A continuous set of states $\mathcal{S}$.
- A continuous set of actions $\mathcal{A}$.
- $f(s'|s,a)$ is the transition probability density function defining the transition from state s to state s' given action a is taken at state s.
- The immediate reward, $r(s,a,s')$, is attained by taking action a at state s and then transiting to state s'.
- The discount factor γ.

An MDP reformulation for (4.7) is described as follows. Each state s is defined by the battery level b, channel gain h, and amount of harvested energy e, where $s = (b,h,e)$. The action a is the transmission power p^{Tx}. Each state s has a subset of actions $\mathcal{P}_s^{Tx}$ such that $\mathcal{P}_s^{Tx} \subseteq \mathcal{P}^{Tx}$. Battery levels evolve according to

$$b' = \min\{b+e, B_{\max}\} - T_c\, p^{Tx} \tag{6.1}$$

The transition probability density function $f(s'|s,p^{Tx})$ is given by

$$f(s'|s,p^{Tx}) = \begin{cases} f_{\mathcal{E}_n}(e'|e)\cdot f_{\mathcal{H}}(h'|h), & \text{if (6.1) is satisfied} \\ 0, & \text{otherwise} \end{cases} \tag{6.2}$$

where the channel gain and EH processes are independent.

The immediate reward, which is the amount of received data resulting from taking action p^{Tx} at state s is given by

$$r(s,p^{Tx}) = T_c \log_2\left(1 + \frac{p^{Tx}\,|h|^2}{\sigma_n^2}\right) \tag{6.3}$$

In this context, the immediate reward is a function of the current state s and the selected action p^{Tx} only, and it is independent of the next state s'.

A stochastic parameterized policy $\pi(p^{Tx}|s,\boldsymbol{\theta})$ maps states to actions stochastically, where the goal is to find a suboptimal policy's parameter vector that maximizes a performance measure $J(\boldsymbol{\theta})$.

6.3 Actor-Critic

6.3.1 Actor

In this context, the actor uses policy gradient to optimize a parameterized stochastic policy $\pi(p^{Tx}|s,\boldsymbol{\theta})$. Policy gradient aims at maximizing the average value of the states

$$J(\boldsymbol{\theta}) = \int_{\mathcal{S}} d^{\pi}(s) \int_{\mathcal{P}^{Tx}} \pi(p^{Tx}|s,\boldsymbol{\theta})\, q_{\pi}(s,p^{Tx}) dp^{Tx}\, ds \tag{6.4}$$

where transmitting data from S to D is a continuing task, $\pi(p^{Tx}|s,\boldsymbol{\theta})$ is the used stochastic parameterized policy, $d^{\pi}(s)$ is the steady-state distribution of the underlying MDP using the policy $\pi(p^{Tx}|s,\boldsymbol{\theta})$, and $q_{\pi}(s,p^{Tx})$ is the action-value of state-action pair (s,p^{Tx}) under policy $\pi(p^{Tx}|s,\boldsymbol{\theta})$. Tasks can be classified into to episodic tasks and continuing tasks. In episodic tasks, the agent-environment interaction breaks into subsequences called episodes, such as plays in a game. On the other hand, continuing tasks refer to tasks where the agent-environment interaction does not break into episodes and continues without limit [59]. In continuing tasks, average value of the states or the average reward per time-step are used to evaluate stochastic parameterized policies when policy gradient is used [59; 95]. The policy's parameter vector $\boldsymbol{\theta}$ is updated according to

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \beta \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \tag{6.5}$$

where $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ is the gradient of $J(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$, and β is the learning rate.

One of the main challenges in finding $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ is to ensure improvement during changing $\boldsymbol{\theta}$, since changing $\boldsymbol{\theta}$ will change the policy and the states' distribution at the same time. The other challenge is that the effect of $\boldsymbol{\theta}$ on the states' distribution is unknown. Policy gradient theorem provides an expression for $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ that does not involve the derivative of the states' distribution with respect to $\boldsymbol{\theta}$ [59]. According to this theorem, for any differentiable policy, $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ is approximated by [95]

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx E_{\pi}[\nabla_{\boldsymbol{\theta}} \ln(\pi(p^{Tx}|s,\boldsymbol{\theta}))\, Q(s,p^{Tx},\boldsymbol{w})] \tag{6.6}$$

where $Q(s,p^{Tx},\boldsymbol{w})$ is the approximated action-value function by the critic.

Due to the difficulty of finding $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$, the stochastic estimate $\widehat{\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})}$ that approximates $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ is used [59; 95], and $\boldsymbol{\theta}$ is updated according to

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \beta \widehat{\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})} \tag{6.7}$$

where

$$\widehat{\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})} = \nabla_{\boldsymbol{\theta}} \ln(\pi(p^{Tx}|s, \boldsymbol{\theta}))\, Q(s, p^{Tx}, \boldsymbol{w}) \tag{6.8}$$

In this work, the parameterized policy $\pi(p^{Tx}|s, \boldsymbol{\theta})$ is modeled by a normal distribution with parameterized mean $\mu(s, \boldsymbol{\theta}_\mu)$ and standard deviation $\sigma(s, \boldsymbol{\theta}_\sigma)$. $\pi(p^{Tx}|s, \boldsymbol{\theta})$ is given by

$$\pi(p^{Tx}|s, \boldsymbol{\theta}) = \frac{1}{\sqrt{2 * \pi * (\sigma(s, \boldsymbol{\theta}_\sigma))^2}} \exp\left(- \frac{(T_c\, p^{Tx} - \mu(s, \boldsymbol{\theta}_\mu))^2}{2 * (\sigma(s, \boldsymbol{\theta}_\sigma))^2} \right) \tag{6.9}$$

where $\boldsymbol{\theta} = [\boldsymbol{\theta}_\mu, \boldsymbol{\theta}_\sigma]^\top$, and π is the number $\pi \approx 3.14159$.

The mean $\mu(s, \boldsymbol{\theta}_\mu)$ should be within the range of minimum and maximum values at each state. Due to this constraint, the hyperbolic tangent function, which restricts the output between -1 and 1, is used to model the parameterized mean

$$\mu(s, \boldsymbol{\theta}_\mu) = [\mu_{\max}(s) - \mu_{\min}(s)] \left(\frac{1 + \tanh(\boldsymbol{\theta}_\mu^\intercal\, \boldsymbol{\phi}(s))}{2} \right) + \mu_{\min}(s) \tag{6.10}$$

where $\mu_{\max}(s) = b$ is the maximum value that can be assigned to the mean at state s, which is the current battery level, $\mu_{\min}(s) = 0$ is the minimum value that can be assigned to the mean at state s, and $\boldsymbol{\phi}(s)$ is a vector of features at state s. $\boldsymbol{\phi}(s)$ is a vector of two binary feature functions. The first feature is related to energy availability condition. It is set to one if the available energy is more than zero; otherwise it is set to zero. The second feature function is related to energy overflow condition. If the current energy level is the maximum capacity of the battery, it is set to one, otherwise, it is set to zero.

The standard deviation should be positive, so, it is modeled by an exponential with a linear exponent [59]

$$\sigma(s, \boldsymbol{\theta}_\sigma) = \exp(\boldsymbol{\theta}_\sigma^\intercal\, \boldsymbol{\phi}(s)) \tag{6.11}$$

$\boldsymbol{\theta}$ is updated according to

$$\boldsymbol{\theta}_\mu \leftarrow \boldsymbol{\theta}_\mu + \beta[\nabla_{\boldsymbol{\theta}_\mu} \ln(\pi(p^{Tx}|s, \boldsymbol{\theta}))\, Q(s, p^{Tx}, \boldsymbol{w})] \tag{6.12}$$

$$\boldsymbol{\theta}_\sigma \leftarrow \boldsymbol{\theta}_\sigma + \beta[\nabla_{\boldsymbol{\theta}_\sigma} \ln(\pi(p^{Tx}|s, \boldsymbol{\theta}))\, Q(s, p^{Tx}, \boldsymbol{w})] \tag{6.13}$$

$\boldsymbol{\theta}$ can be rewritten as

$$\boldsymbol{\theta}_\mu \leftarrow \boldsymbol{\theta}_\mu + \beta \left[\left(\frac{(T_c\, p^{Tx} - \mu(s, \boldsymbol{\theta}_\mu))}{\sigma(s, \boldsymbol{\theta}_\sigma)^2} \nabla_{\boldsymbol{\theta}_\mu} \mu(s, \boldsymbol{\theta}_\mu) \right) \cdot Q(s, p^{Tx}, \boldsymbol{w}) \right] \tag{6.14}$$

$$\boldsymbol{\theta}_\sigma \leftarrow \boldsymbol{\theta}_\sigma + \beta \left[\left(\left(\frac{(T_c\, p^{Tx} - \mu(s, \boldsymbol{\theta}_\mu))^2}{\sigma(s, \boldsymbol{\theta}_\sigma)^3} - \sigma(s, \boldsymbol{\theta}_\sigma)^{-1} \right) \nabla_{\boldsymbol{\theta}_\sigma} \sigma(s, \boldsymbol{\theta}_\sigma) \right) \cdot Q(s, p^{Tx}, \boldsymbol{w}) \right] \tag{6.15}$$

where

$$\nabla_{\boldsymbol{\theta}_\mu} \mu(s, \boldsymbol{\theta}_\mu) = \left(\frac{\mu_{\max}(s) - \mu_{\min}(s)}{2} \right) \left[1 - \tanh^2(\boldsymbol{\theta}_\mu^\intercal\, \boldsymbol{\phi}(s)) \right] \boldsymbol{\phi}(s) \tag{6.16}$$

and

$$\nabla_{\boldsymbol{\theta}_\sigma} \sigma(s, \boldsymbol{\theta}_\sigma) = \exp(\boldsymbol{\theta}_\sigma^\intercal\, \boldsymbol{\phi}(s))\, \boldsymbol{\phi}(s) \tag{6.17}$$

6.3.2 Critic

The critic part of RL agent is used to approximate the action-value function and evaluate the policy optimized by the actor. A neural network of three layers is used to approximate the action-value function, which is given by $Q(s, p^{Tx}, \boldsymbol{w})$, where $\boldsymbol{w}$ is a weight vector used by the neural network. Backpropagation is used to minimize the squared TD error, $r(s, p^{Tx}) + \gamma\, Q(s', p^{Tx'}, \boldsymbol{w}) - Q(s, p^{Tx}, \boldsymbol{w})$.

6.4 Simulation Results

In this section, the proposed algorithm is evaluated. In the numerical analysis, it is assumed that each time slot is 1 second in duration. The available bandwidth BW is 1 MHz, and the noise spectral density is $N_0 = 4 \times 10^{-21}$ W/Hz.

It is also assumed that the S is equipped with a solar panel with an area of 25 cm^2 and 10% harvesting efficiency. An outdoor solar panel can get the benefit of 100 mW/cm^2 solar irradiance under standard conditions, and harvesting efficiency between 5% and 30%, depending on the material used in manufacturing the panel [6].

The used parameters are as follows. The discount factor γ is set to 0.9, and the learning rate β is selected to be 0.0002. The used battery has a maximum capacity of 3 J. All results are averaged over 300 runs. The starting state is selected randomly using a uniform distribution. The EH and channel gain processes are Markov processes. $E_i \in \mathcal{E}_n \triangleq [0, 0.25]$ J is a continuous

random variable with a normal distribution $f_{\mathcal{E}_n}$ with standard deviation 0.1 and mean equals to E_{i-1}. $H_i \in \mathcal{H} \triangleq [0.1, 1]$ is a continuous random variable with a normal distribution $f_{\mathcal{H}}$ with standard deviation 0.1 and mean equals to H_{i-1}.

6.4.1 The cumulative discounted return for actor-critic versus hasty

In this experiment, the critic is implemented using a 3 layer neural network. The first and second layers have 10 and 5 neurons respectively, with ReLU activation function. The third layer has one linear neuron. Using Hasty policy, all available energy is allocated for data transmission each time, regardless of previous experience, i.e., using a greedy approach. The goal is to avoid energy overflow situations [36].

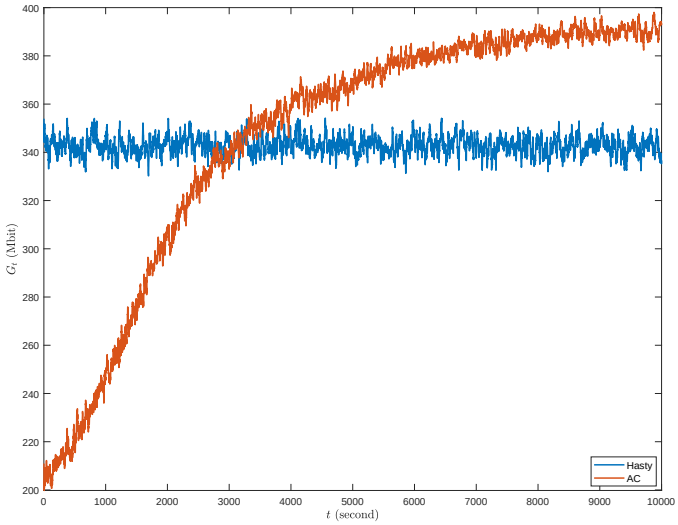

Figure 6.2: The cumulative discounted return G_t versus t.

Figure 6.2 shows the discounted return G_t (i.e., the cumulative discounted received data starting from time t). The cumulative discounted received data refers to the amount of valuable data received within a given period of time. The discounted return G_t of the hasty algorithm takes a near-constant shape all the time, since this algorithm uses one policy all the time, and the discount factor γ which restricts the discounted return to a certain value. For the actor-critic, it starts from a random policy. At the beginning of the session, its discounted return increases significantly with experience. As time increases, it starts taking a near-constant

pattern, which is due to learning a suboptimal policy that can not be further improved, and the discount factor γ that restricts the discounted return to a particular value.

6.4.2 The effect of the approximated value function on the cumulative discounted return

In this part, different architectures of the neural network used by the critic are considered. The goal is to investigate the effect of the approximated action-value function on the cumulative discounted return. The considered architectures of the neural network are, three layer neural network with 3 and 2 neurons at the first and second layers, respectively, three layer network with 5 and 3 neurons at the first and second layers, respectively, and three layer network with 10 and 5 neurons at the first and second layers, respectively. The neurons in the first and second layers are neurons with ReLU activation function, while the neuron at the third layer is a neuron with a linear activation function.

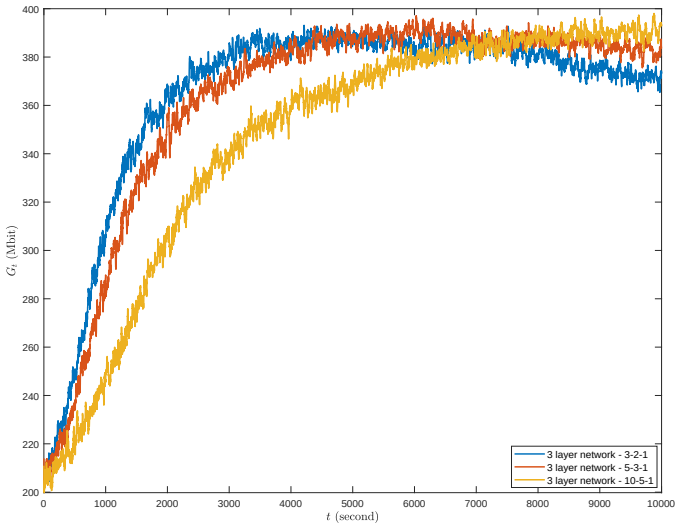

Figure 6.3: The cumulative discounted return G_t versus t.

Figure 6.3 shows the performance of the considered architectures, which depends on their accuracy in approximating the action-value function. As the accuracy of the estimated action-value function increases, the actor will be able to optimize its policy more precisely in a direction that maximizes the cumulative discounted return. The best performance is

achieved by a three layer neural network with 10 and 5 neurons at the first and second layers, respectively.

6.5 Chapter Summary

In this chapter, a point-to-point EH communications system is investigated. This system consists of a source and a destination. The source is capable of harvesting solar energy and storing it in a finite capacity battery. The EH and channel gain processes are Markov processes with continuous spaces. The agent-environment interaction is modeled by an MDP with continuous state and action spaces. AC was used to optimize the performance of the considered system. The critic used a neural network of three layers to approximate the action-value function and evaluate the policy optimized by actor. The actor used a parameterized stochastic policy to map states to actions stochastically. The policy is modeled by a normal distribution with parameterized mean and standard deviation. The mean is modeled by the hyperbolic tangent function to restrict the mean by available actions at each state. The standard deviation is modeled by an exponential function with a linear exponent to guarantee positive values for the standard deviation. Policy gradient was used to optimize the policy's parameters to maximize the system throughput. The system performance was compared to hasty algorithm, where the results showed the ability of AC learning to improve the performance of EH communications systems with experience, when these systems work in unknown and uncertain environments, and the state and action spaces are continuous. Then, the performance of different architectures of the neural network used by the critic was evaluated.

CHAPTER 7. CONCLUSIONS AND FUTURE WORK

7.1 Conclusions and Contributions

This thesis focuses on analyzing and proposing models and algorithms for EH wireless communications systems, which can provide efficient communications systems with long lifetime and high quality of service (QoS). A brief summary of the main contributions of this thesis is presented.

- Chapter 3. Cooperative underlay cognitive radio network with energy harvesting was investigated. In this underlay cognitive radio scheme, secondary users are allowed to access the spectrum, respecting a certain primary interference threshold. The secondary nodes employ decode-and-forward relaying in order to maximize the total received data by optimizing their transmit powers. In the secondary network, both the secondary source and relay are able to harvest energy from renewable sources and store it in finite capacity batteries. An optimization problem was formulated to maximize the sum of the achievable rate over multiple time slots. The formulated problem is a non convex problem, and change of variable was used to transform it to an equivalent convex form. Projected subgradient method was used to find the power allocated to the secondary network. Numerical simulations were conducted to study the performance of the proposed system. Comparisons were made between the proposed system and other conventional scenarios, and it is observed that when the required signal-to-interference-plus-noise ratio (SINR) at the primary receiver is high, the proposed harvesting-based scheme and conventional-based scheme perform similarly.

- Chapter 4. A practical scenario in terms of information availability about the energy harvesting and channel gain processes was investigated. The considered system is a

point-to-point energy harvesting communications system consisting of a source and a destination, and works in uncertain environment. The source is equipped with an infinite buffer to store data, and energy harvesting capability to harvest solar energy and store it in a finite capacity battery. The goal is to maximize the expected cumulative throughput of the considered system while prolonging its lifetime. The agent-environment interaction was modeled by a Markov decision process with discrete state and action spaces. Two cases were considered, which are statistical knowledge availability about the underlying processes, and when the system works in an unknown environment. In the first case, we designed an algorithm utilizing the statistical knowledge availability to improve the system performance, while reducing the complexity of traditional methods (e.g., value iteration, and policy iteration). The proposed algorithm uses instant knowledge about the channel, harvested energy, and current battery level to find a suboptimal policy. In the second case, when the statistical knowledge of the underlying processes is unavailable, reinforcement learning was used. Two different reinforcement learning exploration algorithms, one that is introduced in this thesis called convergence-based and the epsilon-greedy algorithms, were used. The first one uses the action-value function convergence error and the exploration time threshold to balance between exploration and exploitation, while the second algorithm tries to achieve balancing through the exploration probability epsilon. Simulations and comparisons with conventional algorithms show the effectiveness of the look-ahead algorithm when the statistical knowledge is available, and the effectiveness of reinforcement learning in optimizing the system performance when this knowledge is unavailable.

- Chapter 5. Learning architectures for reinforcement learning were introduced. They are called selector-actor-critic, tuner-actor-critic, and estimator-selector-actor-critic. These architectures aim at developing reinforcement learning field and providing intelligent agents. A number of elements were added to the conventional actor-critic model to implement these architectures. Actor-critic mainly consists of an actor and a critic. The actor is used to optimize a stochastic parameterized policy to select actions that maximize

a performance measure. The critic estimates the action-value function and criticises the optimized policy by the actor. In the selector-actor-critic architecture, a selector was added to actor-critic model, which is used to determine the most promising action at the current state. Then, this action is used by the actor to optimize its policy. The goal is to increase the speed and the efficiency of learning a suboptimal policy. The newly added components to the tuner-actor-critic architecture are a model learner and a tuner. The model learner is used to approximate the dynamics of the underlaying model. The tuner uses the approximated action-value function from the critic, the learned underlying model, and the Bellman equation to tune the value of the current state-action pair. The goal of the tuner-actor-critic is to improve the learning process by providing the actor by a tuned value of the current state-action pair. Estimator-selector-actor-critic is an architecture introduced to implement intelligent agents. This architecture mimics rational humans in the way of analyzing available information, and making decisions. It is implemented based on two ideas, which are lookahead and intuition. Lookahead appears in collecting information about available actions at next state and estimating their values, while the intuition appears in maximizing the probability of selecting the most promising action at next state. A number of elements were added to the actor-critic architecture to provide agents with these two capabilities. These elements are an underlying model learner, an estimator, and a selector. The estimator uses the approximated action-value function, the learned underlying model, and the Bellman equation to estimate the values of all actions at next state. The selector determines the most promising action at next state, which will be used by the actor to optimize the policy. Lookahead capability is implemented using the interaction between the model learner, the critic, and the estimator. While the actor and the selector are used to support the agent by the intuition capability. The introduced architectures were evaluated using an energy harvesting communications system working in an uncertain and unknown environment, where a discrete state and action Markov decision process was used to model the agent-environment interaction. Simulation results showed the superiority of the estimator-selector-actor-critic over the remaining architectures in terms of the total rewards collected by each one. On the

other hand, the results showed the ability of the tuner-actor-critic, selector-actor-critic, and actor-critic to find better policies at the end of learning session compared to the estimator-selector-actor-critic.

- Chapter 6. More realistic point-to-point energy harvesting communications system was investigated, where the energy harvesting and channel gain processes are random processes with continuous random variables. The source is an energy harvesting node that is capable of harvesting solar energy and store it in a finite capacity battery. The agent-environment interaction was modeled by a continuous state and action Markov decision process, where the numbers of states and actions are infinite. Actor-critic was used to optimize the system performance. In this work, a neural network of three layers is implemented and used by the critic to approximate the action-value function and criticise the policy optimized by the actor. The actor uses a stochastic parameterized policy to select actions stochastically at different states. The policy is modeled by a normal distribution with parameterized mean and standard deviation. The standard deviation is modeled by an exponential function to bound the standard deviation by positive values. The mean is modeled by the hyperbolic tangent function to bound the mean within the available actions at each state. Policy gradient was used to optimize the policy's parameters to maximize the system throughput. The system performance was compared with hasty policy, where the results showed the ability of actor-critic reinforcement learning to improve its performance and learn a suboptimal policy with experience. Then, different architectures of the neural network used by the critic were evaluated.

7.2 Future Work

In this part of the thesis, two directions for future and ongoing work are presented. These directions will deal with new research field that can improve our work.

Reinforcement learning with continuous state and action spaces. This direction aims at extending our reinforcement learning architectures proposed in Chapter 5 to be

compatible with continuous state and action MDPs. In this direction, we have solved some challenges related to optimizing the parameters of the used stochastic parameterized policy. We have used a normal distribution of a parameterized mean and standard deviation that are needed to be optimized. The first challenge is related to optimizing the mean. The mean is dependent of the states, where it is constrained by a minimum and a maximum value at each state. This problem has been solved by modeling the mean by the hyperbolic tangent function that restricts the mean by the minimum and maximum value of available actions at each state. The second challenge is to have a parameterized standard deviation with positive values all time. This challenge has been solved by modeling the standard deviation by a parameterized exponential function which guarantees positive values all time. Solving the previous challenges has enabled us to implement actor-critic dealing with continuous state and action MDPs. Our future work aims at finding efficient solutions to the remaining challenges to implement selector-actor-critic, a tuner-actor-critic, and an estimator-selector-actor-critic that are able to solve reinforcement learning problems with continuous state and action spaces. The challenges are summarized as follows. The first one is to approximate the dynamics of the underlying continuous state and action MDPs. Solving this challenge will enable us to implement a tuner-actor-critic. The second challenge is to implement an efficient and precise method to extract the greedy action at a state, when the action-value function is approximated by neural networks. Solving this challenge will be used to implement a selector-actor-critic. Implementing estimator-selector-actor-critic needs integrating solutions of the previously mentioned challenges with an actor-critic.

The survival of our communications networks is not restricted by the ability of communications systems to withstand electronic and physical threats, it also needs to manage limited resources. Another promising direction related these proposed architectures is to provide intelligent communications systems that are able to adapt their characteristics in uncertain and unknown environments. Intelligent techniques and systems will play a major role in managing available resources in a way that guarantees good communications services in any environment. An example on using these proposed architectures in wireless communications is to implement intelligent cognitive radio networks. In these networks, secondary users can

be supported by our proposed architectures to optimize their transmission power and their utilization to primary spectra in a way that increases the throughput of both secondary and primary networks.

Multi-agent reinforcement learning. The second direction is to extend our work to multi-agent reinforcement learning. This considers cases when a system has more than one agent interacting with each other to do a particular task. Multi-agent reinforcement learning manages the operation of multiple agents working in unknown and uncertain environments. Some of the main challenges in multi-agent environments can be summarized as follows. Firstly, an agent's action is dependent on the other agents' actions. Secondly, all agents should converge to an optimal joint policy that provides good overall performance. One way to find an optimal joint policy is to implement joint states of all agents, especially if these agents are heterogeneous. Our future work aims at developing multi-agent reinforcement learning architectures capable of learning joint optimal policies with minimum number of overheads required to implement joint states. One proposed solution to implement such architectures is to approximate the joint behavior of all agents, which will minimize the exchanged overheads required to implement joint states. The other advantage of approximating joint states is to minimize the time required to make decisions, especially for applications that need immediate actions.

There is a number of promising applications of the proposed architectures in communications networks, when these networks consist of multiple agents interacting with each other in unknown environments. In this direction, we aim at providing intelligent protocols for these networks. One of the promising applications is multi-user cognitive radio networks, when the secondary network has more than one secondary user wanting to utilize a primary spectrum. Using our proposed architecture, the behavior of all users in both primary and secondary networks can be approximated, and an optimal joint policy can be learned, even if the behaviors of all users are unknown in both networks.

BIBLIOGRAPHY

[1] 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 Journal on Selected Areas in Communications*, vol. 33, no. 3, pp. 360–381, Mar. 2015.

[2] O. Ozel, K. Tutuncuoglu, S. Ulukus, and A. Yener, "Fundamental limits of energy harvesting communications," *IEEE Communications Magazine*, vol. 53, no. 4, pp. 126–132, 2015.

[3] A. Ortiz, H. Al-Shatri, X. Li, T. Weber, and A. Klein, "Reinforcement learning for energy harvesting decode-and-forward two-hop communications," *IEEE Transactions on Green Communications and Networking*, vol. 1, no.3, pp. 309–319, Sep. 2017.

[4] R. Sutton and A. Barto, *Reinforcement learning: An introduction*. Cambridge, MA, MIT Press, 1998.

[5] M. Gregori and J. Gómez-Vilardebò, "Online learning algorithms for wireless energy harvesting nodes," in *Proc. of IEEE International Conference on Communications (ICC), Kuala Lumpur, Malaysia*, pp. 1–6, May 2016.

[6] R. Vullers, R. Schaijk, H. Visser, J. Penders, and C. Hoof, "Energy harvesting for autonomous wireless sensor networks," *IEEE Solid-State Circuits Magazine*, vol. 2, no. 2, pp. 29–38, Spring 2010.

[7] Z. Hasan, H. Boostanimehr, and V. Bhargava, "Green cellular networks: A survey, some research issues and challenges," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 4, pp. 524–540, Fourth 2011.

[8] P. Jain, "Recent trends in energy harvesting for green wireless sensor networks," in *Signal Processing and Communication (ICSC), 2015 International Conference on*, pp. 40–45, IEEE, 2015.

[9] L. Qian, C. Sorrells, X. Li, and D. Kataria, "Detection of spectrum congestion in cognitive radio ad hoc networks," in *Advanced Networks and Telecommuncations Systems (ANTS), 2012 IEEE International Conference on*, pp. 44–48, IEEE, 2012.

[10] Z. Tabakovic, S. Grgic, and M. Grgic, "Fuzzy logic power control in cognitive radio," in *Systems, Signals and Image Processing, 2009. IWSSIP 2009. 16th international conference on*, pp. 1–5, IEEE, 2009.

[11] A. Alsharoa, H. Ghazzai, E. Yaacoub, and M.-S. Alouini, "Bandwidth and power allocation for two-way relaying in overlay cognitive radio systems," in *Global Communications Conference (GLOBECOM), 2014 IEEE*, pp. 911–916, IEEE, 2014.

[12] B. B. Jalaeian, R. Zhu, and M. Motani, "An optimal scheduling framework for concurrent transmissions in wireless cognitive radio networks," *Telecommunication Systems*, vol. 60, no. 1, pp. 169–177, 2015.

[13] K. Pathak and A. Banerjee, "On energy cooperation in energy harvesting underlay cognitive radio network," in *Proc. of Twenty Second National Conference on Communication (NCC), Guwahati, India*, pp. 1–6, Mar. 2016.

[14] M. El Tanab and W. Hamouda, "Resource allocation for underlay cognitive radio networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 2, pp. 1249–1276, 2017.

[15] O. Altrad, S. Muhaidat, A. Al-Dweik, A. Shami, and P. D. Yoo, "Opportunistic spectrum access in cognitive radio networks under imperfect spectrum sensing," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 2, pp. 920–925, 2014.

[16] A. Ali and W. Hamouda, "Low power wideband sensing for one-bit quantized cognitive radio systems," *IEEE Wireless Communications Letters*, vol. 5, no. 1, pp. 16–19, 2016.

[17] Y. Liu, G. Pan, H. Zhang, and M. Song, "Hybrid decode-forward & amplify-forward relaying with non-orthogonal multiple access," *IEEE Access*, vol. 4, pp. 4912–4921, 2016.

[18] M. Hasna and M.-S. Alouini, "Optimal power allocation for relayed transmissions over Rayleigh-fading channels," *IEEE Transactions on Wireless Communications*, vol. 3, pp. 1999–2004, Nov. 2004.

[19] J. N. Laneman, D. N. Tse, and G. W. Wornell, "Cooperative diversity in wireless networks: Efficient protocols and outage behavior," *IEEE Transactions on Information Theory*, vol. 50, pp. 3062–3080, Dec. 2004.

[20] A. Sendonaris, E. Erkip, and B. Aazhang, "User cooperation diversity. part I. system description," *IEEE Transactions on Communications*, vol. 51, pp. 1927–1938, Nov. 2003.

[21] D. Hu and S. Mao, "Cooperative relay in cognitive radio networks: Decode-and-forward or amplify-and-forward?," in *Proc. of IEEE Global Telecommunications Conference (GLOBECOM), Miami, Florida, USA*, pp. 1–5, December 2010.

[22] B. Gurakan, O. Ozel, J. Yang, and S. Ulukus, "Energy cooperation in energy harvesting communications," *IEEE Transactions on Communications*, vol. 61, pp. 4884–4898, Dec. 2013.

[23] X. Huang, T. Han, and N. Ansari, "On green-energy-powered cognitive radio networks," vol. 17, no. 2, 2015.

[24] X. Lu, P. Wang, D. Niyato, D. I. Kim, and Z. Han, "Wireless networks with RF energy harvesting: A contemporary survey," vol. 17, no. 2, 2015.

[25] R. Duan, M. Elmusrati, and R. Virrankoski, "Stable transmission for a cognitive-shared channel with rechargeable transmitters," in *Proc. of IEEE International Conference on Communications (ICC)*, (Ottawa, Ontario, Canada), pp. 4632–4636, June 2012.

[26] S. Park, H. Kim, and D. Hong, "Cognitive radio networks with energy harvesting," *IEEE Transactions on Wireless Communications*, vol. 12, pp. 1386–1397, Mar. 2013.

[27] A. Alsharoa, H. Ghazzai, and M. S. Alouini, "Near-optimal power allocation with PSO algorithm for MIMO cognitive networks using multiple AF two-way relays," in *Proc. of IEEE International Conference on Communications (ICC), Sydney, Australia*, pp. 1580–1584, June 2014.

[28] T. Kalluri and V. A. Bohara, "Regenerative relaying in energy harvesting cognitive radio networks," in *Proc. of European Conference on Networks and Communications (EuCNC), Athens, Greece*, pp. 84–88, June 2016.

[29] T. Jing, S. Zhu, H. Li, X. Xing, X. Cheng, Y. Huo, R. Bie, and T. Znati, "Cooperative relay selection in cognitive radio networks," *IEEE Transactions on Vehicular Technology*, vol. 64, pp. 1872–1881, May 2015.

[30] K. Tutuncuoglu and A. Yener, "Optimum transmission policies for battery limited energy harvesting nodes," *IEEE Transactions on Wireless Communications*, vol. 11, no. 3, pp. 1180–1189, 2012.

[31] D. Gündüz and B. Devillers, "Two-hop communication with energy harvesting," in *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2011 4th IEEE International Workshop on*, pp. 201–204, IEEE, 2011.

[32] K. Tutuncuoglu and A. Yener, "Optimal power policy for energy harvesting transmitters with inefficient energy storage," in *Information Sciences and Systems (CISS), 2012 46th Annual Conference on*, pp. 1–6, IEEE, 2012.

[33] O. Orhan, D. Gunduz, and E. Erkip, "Throughput maximization for an energy harvesting communication system with processing cost," in *Information Theory Workshop (ITW), 2012 IEEE*, pp. 84–88, IEEE, 2012.

[34] O. Ozel, K. Tutuncuoglu, J. Yang, S. Ulukus, and A. Yener, "Transmission with energy harvesting nodes in fading wireless channels: Optimal policies," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 8, pp. 1732–1743, 2011.

[35] O. Orhan and E. Erkip, "Energy harvesting two-hop communication networks," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 12, pp. 2658–2670, 2015.

[36] A. Ortiz, H. Al-Shatri, X. Li, T. Weber, and A. Klein, "Reinforcement learning for energy harvesting point-to-point communications," in *Proc. of the IEEE International Conference on Communications (ICC 2016), Kuala Lumpur, Malaysia*, pp. 1–6, May 2016.

[37] Z. Wang, A. Tajer, and X. Wang, "Communication of energy harvesting tags," *IEEE Transactions on Communications*, vol. 60, no. 4, pp. 1159–1166, Apr. 2012.

[38] P. Blasco, D. Gunduz, and M. Dohler, "A learning theoretic approach to energy harvesting communication system optimization," *IEEE Transactions on Wireless Communications*, vol. 12, no. 4, pp. 1872–1882, Apr. 2013.

[39] H. Li, N. Jaggi, and B. Sikdar, "Relay scheduling for cooperative communications in sensor networks with energy harvesting," *IEEE Transactions on Wireless Communications*, vol. 10, no. 9, pp. 2918–2928, 2011.

[40] J. Zhao, Y. Wei, M. Song, and X. Wang, "Dynamic mode management in cognitive radio networks with rf energy harvesting," in *Proc. of the 11th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM 2015), Shanghai, China*, pp. 1–6, Sep. 2015.

[41] W. Li, M.-L. Ku, Y. Chen, and K. R. Liu, "On outage probability for two-way relay networks with stochastic energy harvesting," *IEEE Transactions on Communications*, vol. 64, no. 5, pp. 1901–1915, May 2016.

[42] T. Wang, C. Jiang, and Y. Ren, "Access points selection in super wifi network powered by solar energy harvesting," in *Proc. of the IEEE Wireless Communications and Networking Conference (WCNC 2016), Doha, Qatar*, pp. 1–5, Apr. 2016.

[43] M. L. Littman, T. L. Dean, and L. P. Kaelbling, "On the complexity of solving markov decision problems," in *Proc. of the Eleventh conference on Uncertainty in artificial intelligence, Montreal, Quebec, Canada*, pp. 394–402, Aug. 1995.

[44] V. S. Rao, R. V. Prasad, and I. G. Niemegeers, "Optimal task scheduling policy in energy harvesting wireless sensor networks," in *Proc. of the IEEE Wireless Communications and Networking Conference (WCNC 2015), New Orleans, LA, USA*, pp. 1030–1035, Mar. 2015.

[45] Y. Zhang, D. Niyato, P. Wang, and D. I. Kim, "Optimal energy management policy of mobile energy gateway," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 5, pp. 3685–3699, May 2016.

[46] M. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. Wiley-Interscience, 2005.

[47] C. Szepesvári, "Algorithms for reinforcement learning," *Synthesis lectures on artificial intelligence and machine learning*, vol. 4, no. 1, pp. 1–103, 2010.

[48] T. Mannucci, E.-J. van Kampen, C. de Visser, and Q. Chu, "Safe exploration algorithms for reinforcement learning controllers," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, no.99, pp. 1–13, 2017.

[49] A. D. Tijsma, M. M. Drugan, and M. A. Wiering, "Comparing exploration strategies for q-learning in random stochastic mazes," in *Proc. of the IEEE Symposium Series on Computational Intelligence (SSCI), Athens, Greece*, pp. 1–8, Dec. 2016.

[50] J. van Ast and R. Babuska, "Dynamic exploration in q (λ)-learning," in *Proc. of the IEEE International Joint Conference on Neural Network Proceedings, Vancouver, BC, Canada*, pp. 41–46, July 2006.

[51] S. Mahadevan, "Average reward reinforcement learning: Foundations, algorithms, and empirical results," *Machine learning*, vol. 22, no. 1, pp. 159–195, 1996.

[52] M. Emre, G. Gür, S. Bayhan, and F. Alagöz, "Cooperativeq: Energy-efficient channel access based on cooperative reinforcement learning," in *Proc. of the IEEE International Conference on Communication Workshop (ICCW), London, UK*, pp. 2799–2805, June 2015.

[53] Z. Xia and D. Zhao, "Online reinforcement learning by bayesian inference," in *Proc. of the International Joint Conference on Neural Networks (IJCNN), Killarney, Ireland*, pp. 1–6, July 2015.

[54] Z. Wang, Z. Shi, Y. Li, and J. Tu, "The optimization of path planning for multi-robot system using boltzmann policy based q-learning algorithm," in *Proc. of the IEEE International Conference on Robotics and Biomimetics (ROBIO), Shenzhen, China*, pp. 1199–1204, Dec. 2013.

[55] A. Ortiz, H. Al-Shatri, X. Li, T. Weber, and A. Klein, "Reinforcement learning for energy harvesting point-to-point communications," in *Proc. of the IEEE International Conference on Communications (ICC), Kuala Lumpur, Malaysia*, pp. 1–6, May 2016.

[56] Z.-X. Xu, X.-L. Chen, L. Cao, and C.-X. Li, "A study of count-based exploration and bonus for reinforcement learning," in *Proc. of the IEEE 2nd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA), Chengdu, China*, pp. 425–429, Apr. 2017.

[57] M. Khamassi, G. Velentzas, T. Tsitsimis, and C. Tzafestas, "Active exploration and parameterized reinforcement learning applied to a simulated human-robot interaction task," in *Proc. of the First IEEE International Conference on Robotic Computing (IRC), Taichung, Taiwan*, pp. 28–35, Apr. 2017.

[58] T.-H. Teng and A.-H. Tan, "Knowledge-based exploration for reinforcement learning in self-organizing neural networks," in *Proc. of the IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, Macau, China*, pp. 332–339, Dec. 2012.

[59] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. Cambridge, MA, USA: MIT Press, 2018.

[60] M. A. Wiering, M. Withagen, and M. M. Drugan, "Model-based multi-objective reinforcement learning," in *Proc. of the IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pp. 1–6, IEEE, Orlando, FL, USA, Dec. 2014.

[61] M. P. Deisenroth, G. Neumann, J. Peters, *et al.*, "A survey on policy search for robotics," *Foundations and Trends in Robotics*, vol. 2, no. 1–2, pp. 1–142, Aug. 2013.

[62] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7559–7566, Brisbane, QLD, Australia, May 2018.

[63] Q. J. Huys, A. Cruickshank, and P. Seriès, "Reward-based learning, model-based and model-free," in *Encyclopedia of Computational Neuroscience*, pp. 2634–2641, Mar. 2015.

[64] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep q-learning with model-based acceleration," in *Proc. of the International Conference on Machine Learning*, pp. 2829–2838, New York, NY, USA, June 2016.

[65] M. A. T. Ho, Y. Yamada, and Y. Umetani, "An hmm-based temporal difference learning with model-updating capability for visual tracking of human communicational behaviors," in *Proc. of the IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 170–175, Washington, DC, USA, May 2002.

[66] T. Lampe and M. Riedmiller, "Approximate model-assisted neural fitted q-iteration," in *Proc. of the International Joint Conference on Neural Networks (IJCNN)*, pp. 2698–2704, Beijing, China, July 2014.

[67] I. Grondman, M. Vaandrager, L. Busoniu, R. Babuska, and E. Schuitema, "Efficient model learning methods for actor–critic control," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 3, pp. 591–602, June 2012.

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

[69] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, May 1996.

[70] R. S. Sutton, "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming," in *Machine Learning Proceedings*, pp. 216–224, Austin, Texas, USA, June 1990.

[71] J. P. Hanna and P. Stone, "Towards a data efficient off-policy policy gradient," in *Proc. of the AAAI Spring Symposium on Data Efficient Reinforcement Learning*, pp. 320–323, Palo Alto, CA, Mar. 2018.

[72] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, May 1992.

[73] T. Degris, M. White, and R. S. Sutton, "Off-policy actor-critic," *arXiv preprint arXiv:1205.4839*, 2012.

[74] H. R. Maei, C. Szepesvári, S. Bhatnagar, and R. S. Sutton, "Toward off-policy learning control with function approximation," in *Proc. of the International Conference on Machine Learning (ICML)*, pp. 719–726, Haifa, Israel, June 2010.

[75] J. P. Hanna, P. S. Thomas, P. Stone, and S. Niekum, "Data-efficient policy evaluation through behavior policy search," *arXiv preprint arXiv:1706.03469*, 2017.

[76] D. P. Bertsekas, D. P. Bertsekas, D. P. Bertsekas, and D. P. Bertsekas, *Dynamic programming and optimal control*, vol. 1. Athena scientific Belmont, MA, 1995.

[77] Y. Li, "Deep reinforcement learning: An overview," *arXiv preprint arXiv:1701.07274*, 2017.

[78] A. Carrio, C. Sampedro, A. Rodriguez-Ramos, and P. Campoy, "A review of deep learning methods and applications for unmanned aerial vehicles," *Journal of Sensors*, vol. 2017, 2017.

[79] V. Heidrich-Meisner, M. Lauer, C. Igel, and M. A. Riedmiller, "Reinforcement learning in a nutshell.," in *ESANN*, pp. 277–288, 2007.

[80] D. Silver, "Reinforcement learning and simulation-based search in computer go," 2009.

[81] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[82] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[83] R. M. Freund, "Subgradient optimization, generalized programming, and nonconvex duality," *Massachusetts Institute of Technology*, 2004.

[84] S. Boyd and A. Mutapcic, "Subgradient methods," *Lecture notes of EE364b, Stanford University, Winter Quarter*, vol. 2007, 2006.

[85] D. Dehay, J. Leskow, and A. Napolitano, "Central limit theorem in the functional approach," *IEEE Transactions on Signal Processing*, vol. 61, pp. 4025–4037, Aug. 2013.

[86] C. Bergonzini, B. Lee, J. R. Piorno, and T. S. Rosing, "Management of solar harvested energy in actuation-based and event-triggered systems," in *Proc. of the Energy Harvesting Workshop*, Virginia, USA, Jan. 2009.

[87] A. Alsharoa, H. Ghazzai, A. E. Kamal, and A. Kadri, "Optimization of a power splitting protocol for two-way multiple energy harvesting relay system," *IEEE Transactions on Green Communications and Networking*, vol. 1, no. 4, pp. 444–457, Dec. 2017.

[88] T. Li, P. Fan, Z. Chen, and K. B. Letaief, "Optimum transmission policies for energy harvesting sensor networks powered by a mobile control center," *IEEE Transactions on Wireless Communications*, vol. 15, no. 9, pp. 6132–6145, Sep. 2016.

[89] Y. Xu, W. Zhang, W. Liu, and F. Ferrese, "Multiagent-based reinforcement learning for optimal reactive power dispatch," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1742–1751, Dec. 2012.

[90] A. Masadeh, Z. Wang, and A. E. Kamal, "Convergence-based exploration algorithm for reinforcement learning," in *Electrical and Computer Engineering Technical Reports and White Papers at Iowa State University Digital Repository*, pp. 1–11, 2018.

[91] A. Masadeh, Z. Wang, and A. E. Kamal, "Reinforcement learning exploration algorithms for energy harvesting communications systems," in *Proc. of the IEEE International Conference on Communications (ICC)*, pp. 1–6, Kansas City, MO, USA, May 2018.

[92] W. T. Lin, I. W. Lai, and C. H. Lee, "Distributed energy cooperation for energy harvesting nodes using reinforcement learning," in *Proc. of the IEEE Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pp. 1584–1588, Hong Kong, China, Aug. 2015.

[93] Y. Wei, F. R. Yu, M. Song, and Z. Han, "User scheduling and resource allocation in hetnets with hybrid energy supply: An actor-critic reinforcement learning approach," *IEEE Transactions on Wireless Communications*, vol. 17, no. 1, pp. 680–692, 2018.

[94] X. Xu, D. Hu, and X. Lu, "Kernel-based least squares policy iteration for reinforcement learning," *IEEE Transactions on Neural Networks*, vol. 18, no. 4, pp. 973–992, July 2007.

[95] D. Silver, "Lecture 7: Policy gradient." http://cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/pg.pdf, University College London, London, UK, 2015.

[96] S. Bhatnagar, R. S. Sutton, M. Ghavamzadeh, and M. Lee, "Natural actor–critic algorithms," *Automatica*, vol. 45, no. 11, pp. 2471–2482, Nov. 2009.