BASE STATION POWER OPTIMIZATION FOR GREEN NETWORKS USING REINFORCEMENT LEARNING

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

SEMİH AKTAŞ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

AUGUST 2019

Approval of the thesis:

BASE STATION POWER OPTIMIZATION FOR GREEN NETWORKS USING REINFORCEMENT LEARNING

submitted by **SEMİH AKTAŞ** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Halit Oğuztüzün
Head of Department, **Computer Engineering** _____

Assist. Prof. Dr. Hande Alemdar
Supervisor, **Computer Engineering, METU** _____

Examining Committee Members:

Prof. Dr. Nail Akar
Electrical and Electronics Engineering, Bilkent University _____

Assist. Prof. Dr. Hande Alemdar
Computer Engineering, METU _____

Assoc. Prof. Dr. Ertan Onur
Computer Engineering, METU _____

Date:

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Semih Aktaş

Signature :

ABSTRACT

BASE STATION POWER OPTIMIZATION FOR GREEN NETWORKS USING REINFORCEMENT LEARNING

Aktaş, Semih
M.S., Department of Computer Engineering
Supervisor: Assist. Prof. Dr. Hande Alemdar

August 2019, 52 pages

The next generation mobile networks have to provide high data rates, extremely low latency, and support high connection density. To meet these requirements, the number of base stations will have to increase and this increase will lead to an energy consumption issue. Therefore "green" approaches to the network operation will gain importance. Reducing the energy consumption of base stations is essential for going green and also it helps service providers to reduce operational expenses. However, achieving energy savings without degrading the quality of service is a huge challenge. In order to address this issue, we propose a machine learning based intelligent solution that also incorporates a network simulator. We develop a reinforcement based learning model by using deep deterministic policy gradient algorithm. Our model update frequently the policy of network switches in a way that, packet be forwarded to base stations with an optimized power level. The policies taken by the network controller are evaluated with a network simulator to ensure the energy consumption reduction and quality of service balance. The reinforcement learning model allows us to constantly learn and adapt to the changing situations in the dynamic network en-

vironment, hence having a more robust and realistic intelligent network management policy set. Our results demonstrate that energy efficiency can be enhanced by 32% and 67% in dense and sparse scenarios, respectively.

Keywords: Green networking, Reinforcement learning, Deep deterministic policy gradient

ÖZ

ÇEVRECİ YENİ NESİL AĞLARDA PEKİŞTİRMELİ ÖĞRENME KULLANARAK GÜÇ OPTİMİZASYONU YAPILMASI

Aktaş, Semih
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Dr. Öğr. Üyesi. Hande Alemdar

Ağustos 2019 , 52 sayfa

Yeni nesil mobil ağlar; yüksek veri hızı, çok düşük gecikme süresi ve yüksek bağlantı yoğunluğunu gereksinimlerini sağlamak zorundadır. Bu gereksinimleri karşılamak için, baz istasyonu sayısının artması gerekmektedir ve bu artış enerji tüketimi sorununu ön plana çıkartacaktır. Bu nedenle yeni nesil mobil ağlarda çevreci yaklaşımlar önem kazanacaktır. Hizmet kalitesini düşürmeden enerji tasarrufu sağlayarak çevreci ağ modelleri geliştirmek çözülmesi zor bir problemdir. Bu problemi çözmek için ağ simülasyonu üzerinde eğitilmiş makine öğrenimi modeli öneriyoruz. Modelimizi, pekiştirmeli öğrenme algoritmalarından biri olan deep deterministic policy gradient algoritmasını kullanarak geliştirdik. Model, zaman içerisinde baz istasyonlarının aktarım güçlerini güncelleyerek paketlerin optimum güç seviyesinde iletilmesini sağlar. Pekiştirmeli öğrenme modeli, dinamik ağ ortamındaki değişen durumları sürekli olarak öğrenmemize ve yeni durumlara adapte olmamıza olanak sağlar. Böylece daha sağlam ve gerçekçi bir akıllı ağ yönetimi politikası belirlenir. Sonuçlarımız, yoğun senaryoda %32 ve seyrek senaryoda %67 oranında enerji verimliliğinin arttırılabileceğini göstermektedir.

Anahtar Kelimeler: Çevreci ağlar, Pekiştirmeli öğrenme, Deep deterministic policy gradient

To my family and close friends

ACKNOWLEDGMENTS

First of all, I would like to mention my thesis advisor Assist. Prof. Dr. Hande Alemdar. Thanks to her wonderful personality, I would like to thank her for all the technical support she has given me and for the personal support she has given me when I have motivationally been challenged. Without her support, patience and guidance, I could not have composed this thesis.

Proceeding a thesis requires teamwork and I would like to thank Assoc. Prof. Dr. Ertan Onur for taking part in this team. He has patiently helped me and has improved our work with his comments. I would like to express my respect for both of Assist. Prof. Dr. Hande Alemdar and Assoc. Prof. Dr. Ertan Onur.

I also appreciate the committee members, Prof. Dr. Nail Akar and Assoc. Prof. Dr. Ertan Onur, for their concerns and valuable feedbacks.

This thesis is supported by Turkcell Teknoloji under BTK Graduate Scholarship Program. I would like to thank Salih Ergüt who is my advisor at Turkcell for his suggestions and helping with industry knowledge. I also appreciate my colleagues at Turkcell for their valuable support and understanding. I would also like to thank the authorities who initiated this program under BTK.

I would also like to thank the members of Wireless Systems, Networks and Cybersecurity Lab for the cozy and cheerful working environment.

I would like to thank my friends who always close to me, listening to me and giving me advice, Nihal, Emre, Görkem, Murat, Yusuf, Ali, Erdem and all others. Without their support and motivation, I could probably not bring this to the end.

Lastly yet most importantly, I would like to express my deepest gratitude to my parents for supporting me. I have a too big family to mention each of them individually, my best brother Kubilay and his wife Deniz, my cousins Ferda, Furkan, Yagmur and others... I want to tell my niece, Nehir, that I will always be there for you.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ALGORITHMS

ALGORITHMS

LIST OF ABBREVIATIONS

γ	Discount Factor
π	Policy
4G	Fourth Generation
5G	Fifth Generation
A	Action
AC-RL	Actor-critic Reinforcement Learning
BS	Base Station
CNRC	Consecutive Negative Reward Check
CTI	Communication Time Interval
CoMP	Coordinated Multipoint
DDPG	Deep Deterministic Policy Gradient
DP	Dynamic Programming
DPG	Deterministic Policy Gradient
DQN	Deep Q Network
DRL	Deep Reinforcement Learning
EE	Energy Efficiency
EPC	Evolved Packet Core
H-CRAN	Heterogeneous Cloud Radio Access Network
HCN	Heterogeneous Cellular Network
HetNets	Heterogeneous Networks
LTE	Long Term Evolution
MC	Monte Carlo
MEC	Multi-access Edge Cloud
ML	Machine Learning

MME	Mobility Management Entity
Mbit	Megabit
NFV	Network Function Virtualization
OFDMA	Orthogonal Frequency-Division Multiple Access
OPEX	Operational Expense
OS	Overall Statuses
P-DDPG	Parallel Deep Deterministic Policy Gradient
PCRF	Policy and Charging Rules Function
PGW	Packet Data Network Gateway
QoS	Quality of Service
R	Reward
RAN	Radio Access Network
RL	Reinforcement Learning
S	State
SDN	Software Defined Network
SGW	Serving Gateway
SINR	Signal to Interference plus Noise Ratio
TD	Temporal Difference
TTI	Transmission Time Interval
UE	User Equipment
V	Value Function
eNB	Evolved NodeB
gNB	Next Generation NodeB

CHAPTER 1

INTRODUCTION

Next generation mobile networks have to meet requirements such as high data rates, extremely low latency, and connection density. Due to the rapid growth of telecommunications technology, energy consumption is also growing at a very fast rate [4]. Mobile service providers are among the top energy consumers [5]. The increase in the energy consumption of mobile networks negatively affects the environment and causes higher operational expenses (OPEX) for mobile service providers. Therefore, "green network" approaches become more popular to reduce energy consumption as well as the cost [4, 6, 7, 8].

In a 5G network, the number of base stations (BS) will increase significantly to meet 5G requirements. Consequently, the energy consumption problem will be more prominent. Luckily, with the development of software defined networks (SDN), it is possible to dynamically configure cells to reduce power consumption when the traffic load is low. This dynamic configuration technique is known as the sleeping strategy or ON-OFF switching. With the sleeping strategies, network operators can avoid unnecessary energy consumption in situations where users are idle or network traffic is low. The sleeping strategy is considered as an approach for energy saving [9]. Therefore, advanced sleeping strategies need to be implemented for future green networks in order to achieve better efficiency without harming the network performance. Machine learning (ML) can be a remedy in that issue.

Mobile network function virtualization (NFV) can be applied over the core and the radio access network (RAN) [10]. This means that we can virtualize these modules and provide on-demand network functions in both of them. Because around 70%-80% network is consumed in RAN [11], network operators expand their investigations in

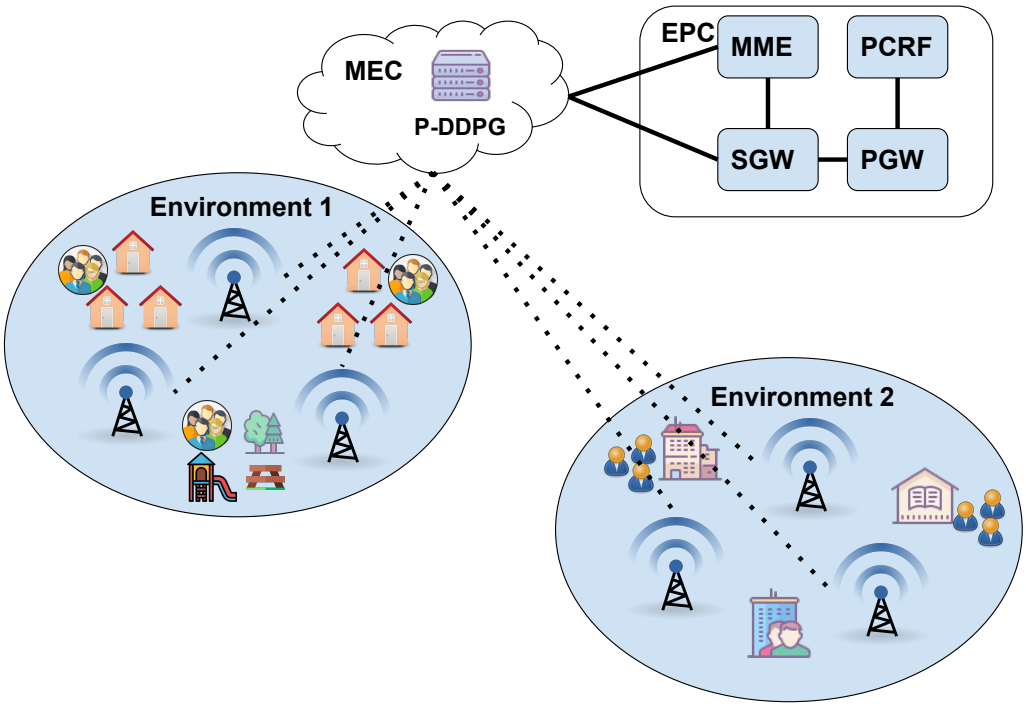


Figure 1.1: P-DDPG network architecture.

virtualizing RAN in the future networks. In this work, we propose consolidating RAN functions using NFV inside multi-access edge clouds (MECs) to reduce the amount of energy consumption in the access network. Our proposed network architecture is shown in Fig. 1.1.

ML provides a way of automatically learning about the environment by using historical data when it is challenging to construct and solve analytic models. Hence, ML is suitable for modelling stochastic environments such as the wireless networks [3]. With the development of SDN, it is now possible the exercise the powerful capabilities of machine learning for network management in terms of intelligent decision making. That is why, recently, the usage of machine learning in the network management has become an active research area [12, 13, 14]. However, there are only a handful of studies that addresses the dynamic nature of the wireless networks.

In order to address the issue, we propose a reinforcement based approach to employ an advanced adaptive sleeping strategy by gathering constant feedback from the network and continuing to learn under changing dynamic conditions such user requests, packet arrival time. We develop our solution basis on Deep Deterministic Policy Gradient

(DDPG) algorithm [3] which is a type of reinforcement learning (RL) algorithm. We extend DDPG algorithm to work with multiple environments as parallel and we called it Parallel DDPG (P-DDPG). Our model reduces power consumption of a group of base stations while maintaining users' quality of service (QoS). In RL, the model continuously learns by taking some actions following a strategy and observing the outcomes of these actions to adjust its strategy over time. By taking many actions, the model learns to differentiate the good actions from the bad ones. This makes its policies evolve over time. In order to find novel potential good strategies, the model sometimes explores new horizons rather than sticking to the exploitation of what has been learnt all the time. This mechanism allows the learner to adapt to the changing conditions as well. This scheme is often considered similar to how a child learns by exploring her environment. Like a child, the RL model makes more mistakes at the beginning of the learning and when it becomes more mature the decisions made are more robust and correct. These initial phases of the learning can be problematic if we deploy the model in the real network environment. To address this issue, we use a system-level network simulator to create a dynamic network environment and we observe the outcomes our actions in this simulator. This allows us to learn a more robust model that captures realistic network dynamics rather than static assumptions about the network while preventing the real users suffering from bad decisions. After the model is mature enough, the learnt policy can be deployed in the real network controller safely. To the best of our knowledge, this is the first study that employs such a realistic scheme.

Our main contributions are:

- We developed P-DDPG algorithm, which enables DDPG to work for parallel environments (Chapter 3). This enabled us to run multiple environments to accelerate learning.
- We developed a machine learning model that reduces energy consumption while maintaining QoS parameters of users on a realistic simulation environment by using the P-DDPG algorithm (Section 3.2).
- Simulation parameters are given for future reproduce. Simulation results and detailed parameter analysis are presented (Chapter 4).

Our results show that it is possible to achieve up to 32% increase in the energy efficiency in a dense scenario and up to 67% increase in sparse scenarios while preserving user QoS parameters such as throughput and SINR.

1.1 Outline

The rest of this thesis is organized as follows. Chapter 2 gives the background of reinforcement learning and reviews the related work on sleeping strategies and machine learning usage on the network. In Chapter 3, the background information of reinforcement learning algorithm and the evaluation of the DDPG algorithm are presented. The motivation of our algorithm and the detailed schema of P-DDPG are presented. This chapter continues with reinforcement learning definitions which are used for network energy efficiency. In Chapter 4, the analysis of parameters and experiment results are presented. Finally, the suggested energy-efficient model is discussed and future works are suggested in Chapter 5.

CHAPTER 2

BACKGROUND AND RELATED WORK

In this chapter, details of reinforcement learning (RL), definitions of RL, and key algorithms of RL are explained in Section 2.1. Section 2.2 reviews the energy efficient methods and algorithms. We examine them under two groups namely analytic models and machine learning models in Section 2.2.1 and Section 2.2.2, respectively.

2.1 Reinforcement Learning and Definitions

Machine learning (ML) is defined as a computer program which performance on a task improves with experience [15]. Machine learning is a way of generating autonomous systems by using historical data as an experience. ML methods are divided into three categories; supervised learning, unsupervised and reinforcement learning [15].

ML algorithms in which training data consist of input vectors and target vector is known as *supervised learning*. In supervised learning, training data is a collection of (x, y) where x refers to input vector and y refers to labels, and the goal is predicting the output (y^*) in response to x^* [16]. Data is labelled in supervised learning.

Unsupervised learning algorithms are used in cases where output results are not included in the training data. Data is unlabelled in unsupervised learning. The goal of unsupervised learning may be finding the similarities in the data for clustering problems, or dimension reduction for computation and visualization, or to determine the distribution of data within the input space [15].

Reinforcement learning (RL) is third paradigms in ML where it concerns the solving

problem with actions that maximize the reward [1]. The basis of RL is trial-and-error, therefore, RL learners take *actions* to learn the *environment*. Each action and state is rewarded or punished as a feedback signal that the environment gives. The learner's goal is to optimize actions by considering possible future *rewards* to maximize the prize. Trial-and-error and considering future reward are the two most important characteristic features of reinforcement learning [1]. Reinforcement learning systems learn from the outcome of their own actions without using human experts [17]. RL has widely usage area on control problems such as resources management, traffic signal control, games, chemistry, autonomous vehicles and robotics [18, 19, 20, 21, 22]. The problems based on control and decision making problems are tried to solve by using reinforcement learning. In particular, the success achieved in the Go game proves that reinforcement learning can be successful in complex and long-term decision-making problems [23]. In addition to that, RL is widely used in robotics. In robotics, environments perform stochastic behaviour, therefore, instead of explicitly detailing the solution to a problem, RL algorithms are used to solve the problem with the trial-and-error approach [24]. With RL proving itself in different areas, control problems on the wireless network are tried to be solved by using RL. It has a wide usage area in the wireless network such as resource management, scheduling, power control and power management, network slicing, edge caching [18, 25, 26].

2.1.1 Elements of Reinforcement Learning

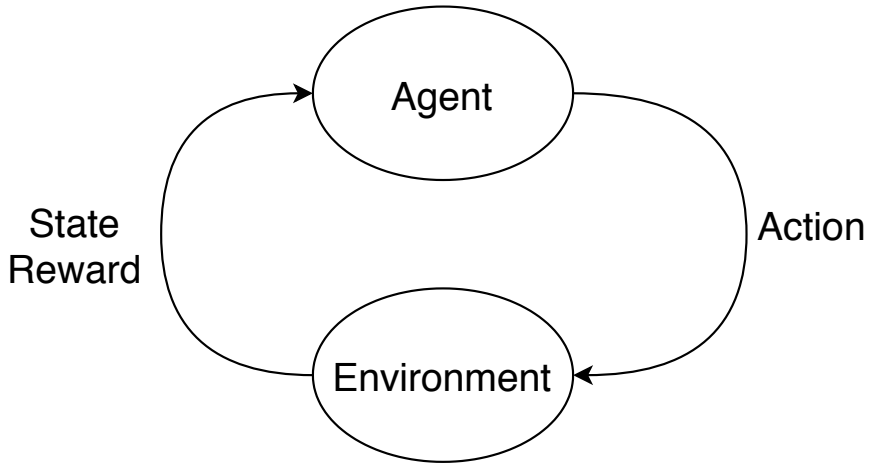


Figure 2.1: Overview of reinforcement learning.

Reinforcement learning components are illustrated in Fig. 2.1. Main components of RL algorithms are *agent* and *environment*.

Agent: The agent is a decision-maker. The agent takes actions and interacts with the environment. The agent is the learning mechanism that optimizes its policy by using the environment's feedback signal (reward).

Environment: The environment is the simulation world. The environment performs the agent's actions and calculates the resulting state. The environment yields a reward to state and reports the state and the reward to the agent.

State (S): The situation of the agent in the environment. The state is calculated from the environment according to the agent actions and the environment's dynamics.

Action (A): The set of actions that the agent can perform on the environment.

Reward (R): A reward is a feedback signal generated from the environment according to the state. Since the agent's actions affect the state, a reward is given according to the agent's action and the resulting state. A reward can be a prize (positive) or punishment (negative).

Policy (π): A policy is the strategy of the agent. The agent chooses actions according to its policy. The policy determines the next action based on the current state.

Value Function (V): A reward signal indicates the instant status of the state. A value function specifies what is good in the long run [1]. The value function indicates the long-term desirability state.

Model: The model is the behaviour of the environment. If the environment knows all resultant states' probabilities without actually performing actions, these models are called *model-based*, as opposed to *model-free* methods that are explicitly trial-and-error learners [1].

Episode: An episode is a sequence of state, action and reward. The endpoint of the episode is called a terminal state, followed by a new episode which begins independently of previous. In other words, the episode is the path from initial to a terminal state.

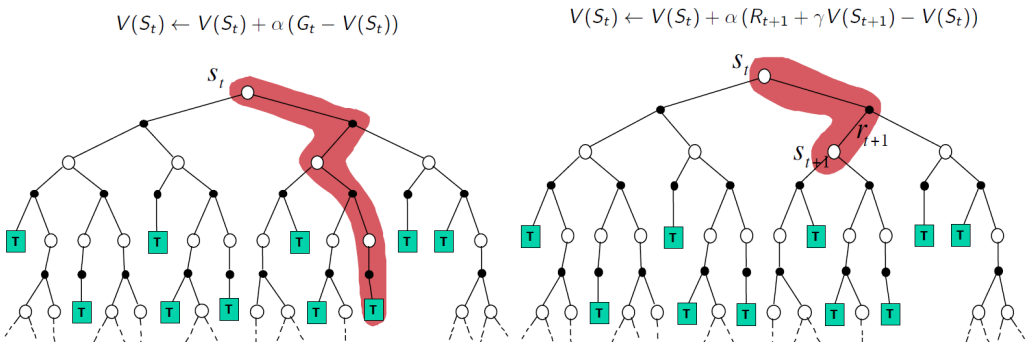
Discount Factor (γ): The discount factor is a value between 0 and 1. The discount factor determines the importance of a future reward. $\gamma = 0$ means that the algorithm only care the immediate reward.

2.1.2 Temporal Difference

Monte Carlo (MC) ideas and dynamic programming (DP) are combined in Temporal difference (TD) which is the basis of reinforcement learning [1]. The main difference with Monte Carlo methods is that MC methods must wait until the end of the episode to increment $V(S_t)$, whereas TD methods update $V(S_t)$ at each time step [1]. The difference is illustrated in Fig 2.2 [27]. The TD update equation is that [28]:

$$V(S_t) = V(S_t) + \alpha\big[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\big], \tag{2.1}$$

where R_{t+1} refers to the reward of action (A_t) that taken at time t, α is a constant step-size, and γ is discount factor of future (expected) reward.



(a) Monte Carlo (b) Temporal Difference

Figure 2.2: Comparison of Monte Carlo and Temporal Difference algorithms. G_t refers to the actual return following time t [1].

2.1.3 Q-Learning

Q-learning is introduced by Watkins in 1989 [29]. It is one of the early breakthroughs in RL. *Q-learning* is a an off-policy TD control algorithm [1]. The Q-learning formu-

lation is that:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)], \tag{2.2}$$

In Q-learning, the aim is that learning the optimal action-value function. The important point of Q-learning is that actions are selected independent of the policy with the probability of ϵ, therefore, Q-learning is examined under off-policy algorithms. This action selection strategy is called ϵ-greedy.

Algorithm 2.1: Q-Learning [1]

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\epsilon > 0$
Initialize $Q(s, a)$, for all $s \in S^+$, $a \in A(s)$, except that $Q(terminal, \cdot) = 0$
for *each episode* **do**
    Initialize S
    while S *is* ***not*** *terminal* **do**
        Choose A from S using policy derived from Q (ϵ-greedy)
        Take action A, observe R, S'
        $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$
        $S \leftarrow S'$
    end
end

Q-learning is constructing a Q-table for each state-action pair. This limits the state space and action space, since constructing Q-table for continuous values are unfeasible.

2.1.4 Deep Q Network

Mnih et al. introduce a deep learning based model for reinforcement learning which is called "Deep Q Network" (DQN) in 2013 [2]. The significant performance improvement is achieved with DQN models. DQN has the capability of processing huge state space, unlike Q-learning. DQN combines neural network approach with reinforcement learning and it is the creator of deep reinforcement learning (DRL). DQN uses neural network based function approximator to estimate the Q-value function for actions. The algorithm is model-free that solves the task directly using samples from the

environment, and it is off-policy: it learns with the greedy strategy [2]. The algorithm which is also known as deep Q-learning is:

Algorithm 2.2: Deep Q-learning [2]

Initialize replay memory D to capacity N

Initialize action-value function Q with random weights

for *episode* $= 1, M$ **do**

  Initialize sequence s_1

  for $t = 1$, T **do**

    With probability ϵ select a random action a_t

    otherwise select $a_t = \max_a Q^*(s_t, a; \theta)$

    Execute action a_t and observe r_t and new state s_{t+1}

    Store transition (s_t, a_t, r_t, s_{t+1}) in D

    Sample random minibatch of transitions (s_j, a_j, r_j, s_{j+1}) from D

    Set

$$y_j = \begin{cases} r_j & \text{for terminal } s_{j+1} \\ r_j + \gamma \max_{a'} Q(s_{j+1}, a'; \theta) & \text{for non-terminal } s_{j+1} \end{cases} \tag{2.3}$$

    Perform a gradient descent step on $(y_j - Q(s_j, a_j; \theta))^2$

  end

end

2.1.5 Deep Deterministic Policy Gradient

Even if Deep Q Network (DQN) has the capability of solving problems with high-dimensional state spaces, it is limited with low-dimensional action spaces [3]. DQN cannot be used for high dimensional action spaces tasks such as physical control tasks. It has the curse of dimensionality in term of high-dimensional action spaces.

Lillicrap et al. provide modifications to the deterministic policy gradient algorithm (DPG) and combine DPG algorithm with DQN. They call their algorithm as Deep DPG (DDPG) [3]. DDPG algorithm uses the actor-critic method. DDPG uses neural network function approximators to learn in large state and large action spaces. Like

DQN, DDPG uses replay buffer to update its policy with batch normalization. DDPG is *model-free*, *off-policy*, *actor-critic* algorithm.

2.1.5.1 Model-free

Model-based methods rely on planning as their primary component, while model-free methods primarily rely on learning [1]. Model-free and model-based are almost the opposite definitions. Methods for solving reinforcement learning problems that use models and planning are called *model-based* methods, as opposed to *model-free* methods that are explicitly trial-and-error learners.

2.1.5.2 Off-policy

Algorithms which strictly obey the state-action decision that generated from the policy are considered as *on-policy*, while those ignoring it are known as *off-policy*. Unlike on-policy learning, off-policy learning are able to learn about an optimal policy while executing an exploratory policy [30].

2.1.5.3 Actor-critic

In Actor-only methods, the gradient of the performance is directly estimated by simulation, and the parameters are updated in a direction of improvement. The gradient is estimated independently of past estimates. Actor-only methods try to optimize the policy without using a value function [31]. These are also known as *policy-based* methods. Critic-only methods are known as *value-based* methods. Critic-only methods aim at learning an approximate solution to the Bellman equation. These methods do not try to optimize directly over a policy space. Actor-critic methods are hybrid methods that combine the strong points of actor-only and critic-only methods [1]. In actor-critic learning, the actor decides actions against states. Critic evaluates the actor's action against the state and tells the actor to how it should adjust its actions. In this case, the actor is a policy and the critic is a value function.

2.1.5.4 DDPG Algorithm

Algorithm 2.3: DDPG algorithm [3]

Randomly initialize critic network $Q(s, a|\theta^Q)$ with weights θ^Q
Randomly initialize actor network $Q(s|\theta^\mu)$ with weights θ^μ
Initialize critic target network Q' with weights $\theta^{Q'} \leftarrow \theta^Q$
Initialize actor target network μ' with weights $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer $\mathcal{R}$
for *episode* $= 1$, M **do**
    Initialize a random process $\mathcal{N}$ for action exploration
    Receive initial observation state s_1
    for $t = 1$, T **do**
        Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
        Execute action a_t and observe reward r_t and new state s_{t+1}
        Store transition (s_t, a_t, r_t, s_{t+1}) in $\mathcal{R}$
        Sample a random minibatch of n transitions (s_i, a_i, r_i, s_{i+1}) from $\mathcal{R}$
        set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
        Update critic by minimizing the loss:

$$L = \frac{1}{n}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2 \tag{2.4}$$

        Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{n}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i} \tag{2.5}$$

        Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'} \tag{2.6}$$

$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'} \tag{2.7}$$

    end
end

τ is the soft target update parameter with $\tau << 1$. Instead of directly copying the actor weights (θ^{μ}) and the critic weights (θ^{Q}) to the target networks ($\theta^{\mu'}$ and $\theta^{Q'}$, respectively), the weights of these target networks are updated by having them slowly track the learned networks [3].

2.2 Related Work

Given the importance of the subject, several studies have been carried out to increase network energy efficiency using different approaches. In general, these studies try to adjust a viable sleeping strategy to achieve energy efficiency by using analytic models or machine learning models.

2.2.1 Energy Efficient Algorithms with Analytic Models

In the first group, researchers try to find an optimum sleeping strategy by modelling the network analytically. In [32], the authors aim to quantify the trade-off between energy consumption and throughput in a heterogeneous cellular network (HCN) where small cell BSs have four distinct power-saving modes. These are *On*, *Standby*, *Sleep* and *Off* with power consumption ratios given as 100, 50, 15 and 0 per cent, respectively. They used a static traffic model which treats all users as stationary with known positions. Instead of using discrete power levels, our proposed model can use continuous power levels. It increases complexity but thanks to the machine learning, the proposed model has the capability of handling continuous power level adaptation. Feng et al. list new challenges of a design of BSs sleeping strategy in 5G networks [9]. They provide a comprehensive review of recent advances on ON-OFF switching mechanisms in different application scenarios. They list various ON-OFF switching problems and known solutions. Their claim is that ON-OFF scheduling is generally an NP-hard problem and solving with standard techniques is unfeasible. Moreover, they point the application of machine learning techniques on the network as future research. Cai et al. propose to dynamically change the operating states of the BSs (as on and off) to reduce the power consumption of the heterogeneous networks (HetNets) [33]. They consider location and user density based operation scheme to optimize

power consumption. These studies describe the network by using mathematical models. After modelling, they try to solve an optimization problem to find an optimum sleeping mode for each BS. Mathematical models try to find threshold values such as the number of user that are assigned to each BS, or throughput threshold. Threshold values are used for sleeping decisions. When we consider the dynamic nature of the system, these models have to make very restricting assumptions for a network to perform well. Unlike machine learning used models, modelling the stochasticity is a challenging problem as well as solving it.

2.2.2 Energy Efficient Algorithms with Machine Learning Models

On the other side, with the advent of SDN, machine learning approaches are available for a network. Historical data and online learning concepts are key-enablers for using machine learning in the network management. Reinforcement learning is well suited to the online and continuous nature of the network management problem. In [34], Lu et al. develop a RL model for cellular networks with coordinated multipoint (CoMP) communication. The model aims to find an optimum solution for ON-OFF switching. They use the Q-learning algorithm while modelling solution. Their main focus is macro base stations and they use only ON-OFF as an action. Therefore, they cannot use the intermediate power values of the base stations. Sharma et al. use an actor-critic reinforcement learning (AC-RL) approach for ON-OFF switching in HetNets [35]. They emphasis transfer learning benefits and they point the relation between energy efficiency and delay importance. RL is also used for energy efficient resource allocation in 5G heterogeneous cloud radio access networks (H-CRAN) by Al Qerm and Shihada [36]. They build an online Q-learning model for resource allocation. Their action and state space are relatively larger than previous studies, however, because of the Q-learning, the curse of dimensionality problem is there for them. Ghadimi et al. try to use RL for transmit power adaptation [37]. Their action space is $\{0, \pm 1, \pm 3\}$. Limited action space and state space are the known phenomenon of Q-learning. There are studies for developing RL models to optimize energy efficiency in small cells such as WiFi routers, 4G home eNBs and 5G home gNBs [38]. Researchers point the problem of small cell energy consumption in next generation networks in their study. They try to optimize energy consumption by

considering QoS. They transform continuous decision variables into discrete ones to reduce complexity and to fit their models which are based on regret learning based RL and fictitious play based RL. In contrast to their work, we focused macro cells and continuous actions are supported in our proposed model. Unlike previous works, we used deep deterministic policy gradient based reinforcement learning. Our proposed model is constructed to support continuous state space and continuous action space. This is a novel approach that finds the optimum power consumption in the network by using deep deterministic policy gradient based reinforcement learning that trained in a realistic environment.

CHAPTER 3

MODELLING P-DDPG FOR ENERGY EFFICIENT BASE STATION CONTROL

In this chapter, our extended DDPG model, which is P-DDPG, is presented in Section 3.1. Section 3.1 also reviews the evolution of DDPG algorithm, and motivation of extending DDPG. Reinforcement learning definitions for energy efficient model are presented in Section 3.2.

3.1 P-DDPG Model

Reinforcement learning is a type of machine learning which is focused on goal-directed learning from interactions [1]. RL is an efficient method for sequential decision-making problems, making them ideal for network management [39]. In RL, the *learner agent* takes *actions*, and each action receives a *reward* as a feedback signal. The reward is positive if the outcome of the action is good in terms of the goal achievement and it is negative otherwise. Through this reward collection mechanism, the agent learns a *policy*, that is, the action sequences required to solve a problem. RL is widely used in dynamic environments where a *state* can be rewarded as positive or negative without analytically modelling the environment but making observations about the outcome instead.

The general workflow of RL algorithms are summarized in Fig. 3.1. The agent takes an action at time t, (A_t), according to the observation in the same time step, (S_t). The environment performs the action and returns the observation (S_{t+1}) and the feedback signal (R_{t+1}). The feedback signals are used to update the policy, i.e., action decision model.

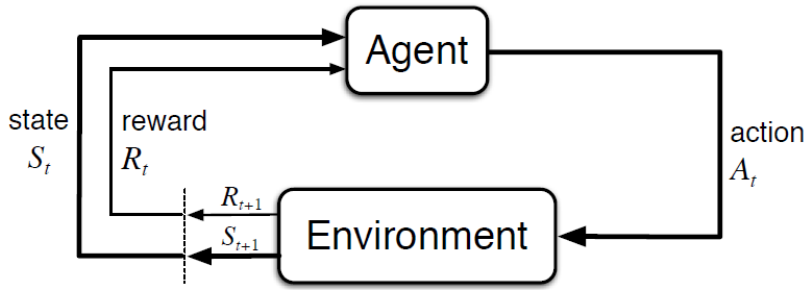


Figure 3.1: Overview of reinforcement learning algorithms [1].

RL has been applied in different ways over time. Q-learning is a one type of RL algorithm. It is based on Q-tables, where rows represent the states and columns represent the actions. All of the action decisions are made by looking at the Q-table, which contains the whole *policy*. Q-learning considers the possible future reward when rewarding the instant status [29]. It is formulated as follows:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)], \tag{3.1}$$

where α represents learning rate and γ represents the discount factor. Each cell in the Q-table is created by considering the maximum expected future reward. Q-learning suffers from the curse of dimensionality because of the need to create a table for each state-action pairs. Although it is a convenient way to use when a problem has discrete state space or discrete action space, Q-table cannot be created for continuous state spaces or continuous action spaces.

The significant performance improvement in RL comes with deep reinforcement learning (DRL), which is also called as the "Deep Q Network" (DQN). Mnih et al. introduce a deep learning based model for reinforcement learning [2]. They demonstrate that DQN model has the ability to master complicated control policies for Atari 2600 computer games, using only raw pixels as input. DQN models can process huge state spaces unlike original Q-learning. In DQN, the experience replay memory is used to speed-up the training of deep networks for RL.

Deep deterministic policy gradient (DDPG) algorithm is developed for continuous control with deep reinforcement learning [3]. DDPG uses actor-critic learning. In actor-critic learning, the actor decides actions against states. Critic evaluates the actor's action against the state and tells the actor to how it should adjust its actions. In

this case, the actor is a policy and the critic is a value function. DDPG is a combination of the deterministic policy gradient (DPG) algorithm [40] with the DQN. DQN still suffers when state space is continuous and there is high-dimensional action space. Therefore, DQN cannot solve control problems with continuous state space and continuous action space, whereas DDPG can solve them. In this work, we also employ the DDPG approach since our space is continuous. DDPG is *model-free*, *off-policy*, *actor-critic* algorithm which is explained in Section 2.1.5.

Traditionally, DDPG algorithm works with a single environment. As can be seen in Fig. 3.1, the operating time is restricted by the Agent's response time or the runtime of the environment. In our case, environment's runtime is considerably slower than the Agent's response time. Therefore, we extend DDPG algorithm to work with multiple environments as parallel and we call it Parallel DDPG (P-DDPG).

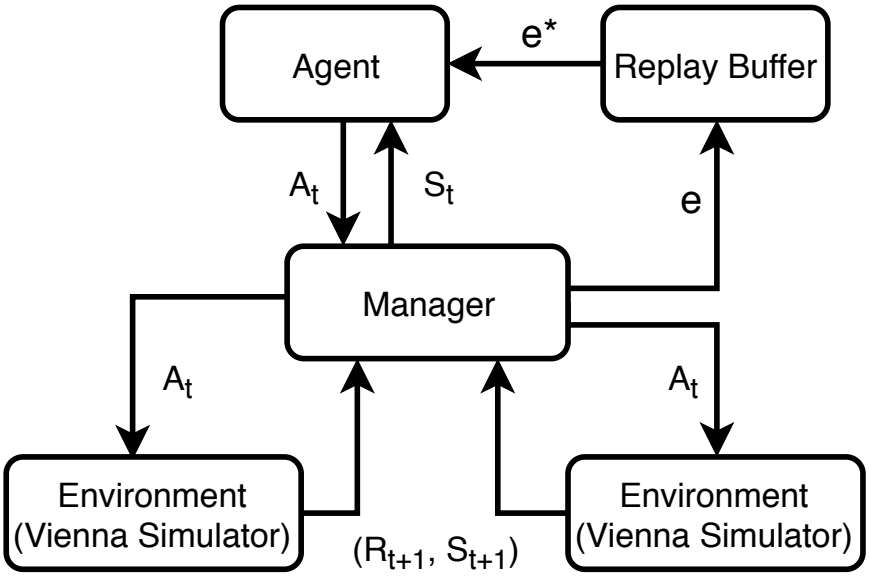


Figure 3.2: Structure of P-DDPG algorithm with parallel environment. e refers to experience [2]. e^* refers to set of experience for batch processing.

Fig. 3.2 shows that the structure of P-DDPG algorithm which is implemented in MEC in order to virtualize RAN functions including BSs transmit power adaptation. Implementing P-DDPG in MEC can provide higher perspective over the network with respect to BSs, while actions and decisions will be made quicker in comparison to the core, due to the location of MEC in the network.

In P-DDPG, the manager is responsible for environment virtualization and memorizing operations. It virtualizes each environment and creates a private channel between the agent and each environment. Through the private channels, each environment works independently from each other. The manager is responsible for taking an environment state and sending them to the agent. The agent decides an action according to the state and sends an action back to the manager. The manager sends back the action to a related environment. Meanwhile, manager stores state, action, reward and next state (action result) in *replay buffer* [2]. Each replay buffer entry is called as *experience* and the agent uses replay buffer entities for batch training. It trains deep neural networks by using samples from replay buffer.

Environment paralleling is a novel approach for DDPG algorithms, and it reduces the convergence time of DDPG algorithm when environment operation time significantly greater than agent's response time.

Each environment configuration is called an episode. Episode length refers to a number of state-action exchange which also defined as CTI. In a single episode, when a certain number of consecutive rewards are negative, we stop the environment's episode and start a new episode. We called this method as a consecutive negative reward check (CNRC). CNRC method is used because when actions cannot improve environment states, and negative rewards continue, eventually negative state-action pairs dominate replay buffer. By using CNRC, the number of positive and negative state-action pairs are balanced in replay buffer. It increases the convergence time of the model.

The use of mathematically modelled environments based on strong assumptions in model training is a common but unrealistic method. It is not possible to use models that trained in hypothetical environments in real systems. Modelling with realistic environment is a challenging problem. We use Vienna Simulator [41], known as advanced realistic network simulator, as an environment. We make some changes to the simulator so that Vienna Simulator can work with P-DDPG. We develop connectors for information exchange between Vienna Simulator and P-DDPG. Moreover, we develop our custom indicators which are defined as overall statuses (OSs). With these modifications, Vienna Simulator has capability of working with P-DDPG. Thanks to

Vienna Simulator, our model is trained in a realistic environment that accommodates many real-life factors such as noise, interference, shadowing, fading and etc.

3.2 Reinforcement Learning Definitions

In our model, user states are measured at each transmission time interval (TTI) which is 1 millisecond in 4G mobile networks. The user behaviour and requirements change during time. Sometimes the users are idle, sometimes they actively use the communication channels, therefore each TTI status does not represent the status of the network. Also, after changing base stations' transmission powers, we need to wait a while before observing action result. Therefore, the environment and agent state exchange is performed at certain time intervals, and we call this interval as the communication time interval (CTI). In other word, each RL cycle period that is depicted in Fig. 3.1 is one CTI.

The user denoted by u are assigned to the base station that denoted by bs. U and BS refers to the set of users and set of base stations with respectively. User status for specific CTI is represented as different notations. For CTI at k, user wideband SINR is called as $SINR^k(u)$. The amount of data (Mbit) that transmitted by the user at specific CTI is represented as $\Psi^k(u)$. User active TTI count for CTI^k notated as $\phi^k(u)$.

State: State is a representation of the instant status. Environment creates state vector by using status of network and BSs. Environment state at CTI^k is defined as:

$$S^k = (OS_i^k, S1_j^k, S2_j^k, S3_j^k, S4_j^k), \tag{3.2}$$

where $i = 1, 2, .., 4$ and $j = 1, 2, .., N$. N is the number of base stations. OS represents the overall status of network. Users' wideband SINR and transmission values are used while formulating overall statuses of network and base station.

Overall wideband SINR equation is:

$$OS_1^k = \frac{\sum_{u \in U} SINR^k(u) \times \phi^k(u)}{\sum_{u \in U} \phi^k(u)}. \tag{3.3}$$

In (3.3), users wideband SINR average value is calculated when users actively using communication channels.

Overall network throughput is formulated as:

$$OS_2^k = \frac{\sum_{u \in U} \Psi^k(u)}{t}. \tag{3.4}$$

The total amount of data that served to users is divided by CTI length (t).

Users' average throughput according to their active TTI count is formulated as:

$$OS_3^k = \frac{\sum_{u \in U} \Psi^k(u)}{\sum_{u \in U} \phi^k(u)}. \tag{3.5}$$

Users who are close to base stations can mislead (3.4) but averaging with active TTI count will normalize throughput considering idle users.
Average throughput per user is formulated as:

$$OS_4^k = \frac{\sum_{u \in U} \Psi^k(u)}{|U| \times t}. \tag{3.6}$$

Overall statuses (OS_i, i=1,2,..,4) give general information about the network configuration which consists a group of base station and users. Overall SINR value (3.3) shows the possibility of transmitting data while users' throughput sum (3.4) shows the actual usage. (3.5) considers idle users. Users' throughput sum (3.4) and users' throughput average (3.6) change when the number of user change. Therefore, each OS value has a unique usage and give information about network. OS values are used as a QoS parameters.

Our model also considers base stations' statuses when deciding their power level. BSs' powers scaled to $[0, 1]$ according to:

$$S1_j^k = P^k(BS_j) \; / \; P_{\text{max}}, \tag{3.7}$$

where $P^k(b)$ refers to the base station b's power at CTI^k and P_{max} refers to maximum power of macro BS.
The number of users assigned to each base station at CTI^k is also used as a status of base station. The equation is:

$$S2_j^k = |U_j^k|, \tag{3.8}$$

where U_j^k refers to the set of user's which are assigned to base station j at CTI k.
The status of BS with respect to the user average wideband SINR is:

$$S3_j^k = \frac{\sum_{u \in BS_j} SINR^k(u) \times \phi^k(u)}{\sum_{u \in BS_j} \phi^k(u)}, \tag{3.9}$$

where $u \in BS_j$ refers to users that assigned to BS_j. (3.9) gives information about users, which are assigned to the same base station, wideband SINR average value according to their active TTI count.

User's average throughput according to users' active TTI count is formulated as:

$$S4_j^k = \frac{\sum_{u \in BS_j} \Psi^k(u)}{\sum_{u \in BS_j} \phi^k(u)}, \tag{3.10}$$

where $u \in BS_j$ refers to users that assigned to BS_j.

The state vector is a combination of the status of network and BSs, therefore, an increase in the number of base stations causes the state vector to grow.

Action: The action at CTI^k is represented as $a^k = (\Delta P_1, \Delta P_2, ..., \Delta P_N)$ where ΔP_i refers to the power change for base station i and N refers to the number of base stations. Actions are scaled to [-1, 1]. The action dimension is related with the number of base stations. The formula for new transmit powers of base stations is that:

$$P^{k+1}(BS_i) = P^k(BS_i) + [a_i^k \times P_{max}], \tag{3.11}$$

where P_{max} refers to maximum power of macro BS.

Reward: Rewards are decided according to the long term goal that the model should satisfy. Our long term goal is maintaining QoS while reducing energy consumption. Hence, the proposed model decides rewards by considering the overall status of network and energy consumption. Each OS creates its own reward according to pre-defined threshold values. Equation is:

$$R_{\text{OS}_i}^k = \begin{cases} 1, & \text{if } th_i^u < \text{OS}_i, \\ -\frac{th_i^u - \text{OS}_i}{th_i^u - th_i^l}, & \text{if } th_i^u \geq \text{OS}_i > th_i^l, \\ -1, & \text{otherwise}, \end{cases} \tag{3.12}$$

where th_i^u and th_i^l are refers to upper and lower threshold values for OS_i. When OS_i is below th_i^u then negative reward appears. To calculate threshold values, the network is observed without taking any action and we call these observations as a

baseline. The long term goal is maintaining the overall status of the network same as the baseline. Therefore, we set threshold values according to the baseline's average QoS parameters. Upper and lower threshold values are used to scale negative rewards to [0, -1].

Energy consumption is also important when we consider a reward. When the proposed model satisfies QoS, then according to energy consumption, it gains a positive reward. The environment uses an average of scaled powers divided by the number of the base station and uses the gain as a positive reward:

$$R_{EC}^{k} = 1 - \frac{\sum_{bs \in BS} S_{1}^{k}(bs)}{|BS|} + \epsilon, \tag{3.13}$$

where R_{EC} stands for energy consumption reward and ϵ is a small positive value. The ϵ value ensures that even if all BSs are configured as maximum power, R_{EC} will always positive. Reducing the current transmit power of the base stations increases R_{EC}.

Environment sends only one reward to agent, and this reward is calculated as:

$$R^{k} = min(R_{\text{OS}_1}^{k}, R_{\text{OS}_2}^{k}, R_{\text{OS}_3}^{k}, R_{\text{OS}_4}^{k}, R_{EC}^{k}) \tag{3.14}$$

At CTI^{k}, if any of OS_i is below th_i^u then environment sends negative reward to agent. Otherwise, since R_{EC}^{k} is always positive, environment sends positive reward to agent. Environment sends positive reward if and only if user QoS parameters, which are OS1-4, are higher than threshold values.

We use energy efficiency (EE) as a measure metric. The equation of energy efficiency is that:

$$EE = \frac{\sum_{u \in U} \Psi(u)}{\sum_{bs \in BS} P(bs) \times t} \tag{3.15}$$

where $P(bs)$ refers to base station power (J/s) and t refers to time (s). The amount of data (Mbit), that is transmitted to users, is divided into the sum of power consumption (J) of base stations.

CHAPTER 4

EXPERIMENTS

In this chapter, the environment setup and details of experiments are explained. In order to reproduce the experiments, we provide simulation parameters and P-DDPG parameters in Sections (4.1) - (4.2). In the sections that follow the simulation model and parameters, our experiments take place. While constructing our experiments, we aim to prove that P-DDPG model can be used for energy efficiency, therefore, we ask the following questions:

- What is the motivation of developing P-DDPG algorithm? As we claim that, Vienna simulator is a slow environment, therefore, we modify DDPG algorithm to work with multiple environments to reduce the training time. Section 4.3 contains experiments of the run time of the simulation and CTI selection. This section also includes the benefits of running multiple environments with P-DDPG algorithm. To sum up, Section 4.3 contains experiments related to parameter selection and benefits of P-DDPG algorithm.

- Are we really energy efficient? We are trying to develop a model that can increase energy efficiency while maintaining QoS. To test our algorithm, we construct two scenarios with 50 users equipment (UEs) and 100 users equipment. We train our algorithm on these scenarios independently. Section 4.4 provides detail of the experiments and their results.

- Final question is that what happens if perturbation occurs in the network? Can the model still be trained? The base stations could shut down suddenly because of different reasons such as internal errors, firmware update, hardware change. We try to prove that after restarting the system, P-DDPG model can continue to

manage the network in an energy efficient way. Section 4.5 gives the modelling details and experiment results.

4.1 P-DDPG Model and Simulation Environment

The P-DDPG model is developed by using TensorFlow [42] with Python language. With socket programming, we developed a manager class to create a private channel for each environment. Thanks to the manager class, P-DDPG algorithm becomes capable of supporting distributed environment. P-DDPG algorithm is constructed according to [3]. Actor and critic networks are also referenced from there. Table 4.1 shows parameters of P-DDPG algorithm.

Table 4.1: P-DDPG parameters and their values.

Parameters	Value
Actor Learning Rate	10^{-4}
Critic Learning Rate	10^{-3}
γ	0.99
τ	10^{-3}
Replay Buffer size	1000000
Mini Batch Size	64
CNRC	40

In order to evaluate our work, we employed an OFDMA based system level simulator (Vienna-LTE) which implement the downlink and the uplink channel of LTE networks by considering real-life parameters [41]. In this simulator, all the characteristic parameters of LTE such as noise, interference, shadowing, fading, antenna size, BSs height, number of transceivers, angles of antennas, handover, channel models, traffic models and etc. are applied which are vital to evaluate the validity of the proposed algorithm in a real-life scenario. We also developed and enhanced the power allocation model of the simulator in this thesis.

Table 4.2: Simulation parameters and their values.

Parameters	Value	Ref
Frequency	2.14 GHz	[43]
K	6910 KM^{-1}	[44]
γ	4	[44]
N_0	$10^{-15.82}$	[44]
Subcarrier Frequency	15 kHz	[43]
Macro BSs Max Power	40 W	[43]
RRH Antenna Gain	Omni-directional	[45]
Path Loss Model	$128.1 + 37.6log_{10}^{R}$, R in km	[43]
Noise Power Spectral Density	-174 dBm/Hz	[43]
Receiver Noise Figure	9 dB	[43]
Feedback	CQI	[45]
Feedback Delay	3 TTI	
Scheduler	Round Robin Traffic	
Traffic Model	Video Stream	
UE speed	Stationary	
Number of Macro BS	7	
TTI	1 ms	
Communication Time Interval (CTI)	40 TTI	
Simulation Length	200 CTI	
Simulation Area	2000 m $\times$ 2000 m	
Active UEs	50,100	

4.2 Parameters of the Simulation Environment

The proposed power management algorithm is implemented for omnidirectional 7 BS with hexagonal geometry scenario to show its capabilities with different UE counts. To provide a fair evaluation we applied realistic traffic load such as video streaming which is modelled based on real-life LTE networks [46]. Simulation parameters and traffic models are summarized in Table 4.2.

Vienna LTE system level simulator consists of different modules including antennas, channel models, network generation, schedulers, traffic models and etc. The proposed power management module is implemented in the network generator module to applied the expected modifications in the lowest level to make the proposed model applicable in real-life cellular networks.

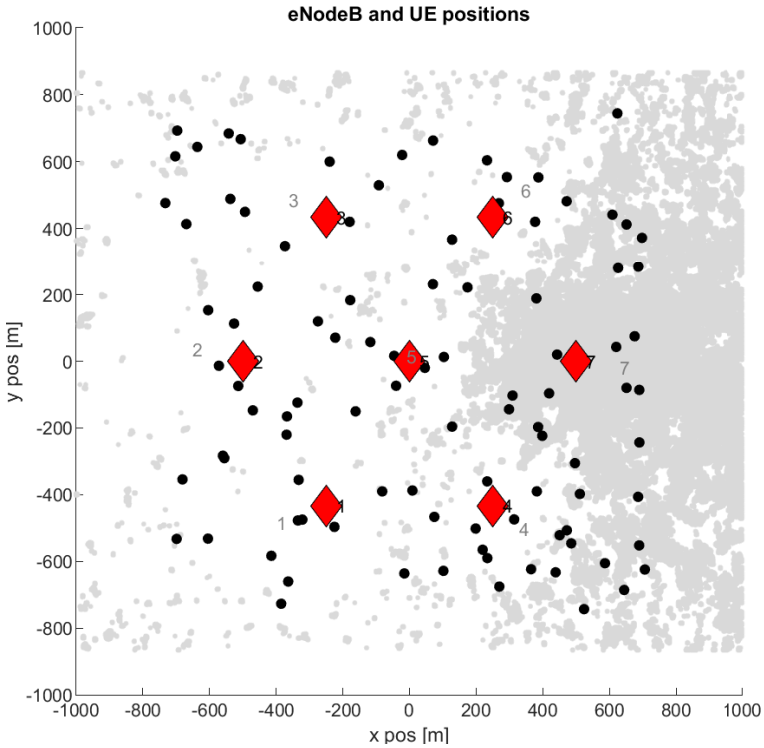


Figure 4.1: Example distribution of environment from Vienna Simulator. Diamonds refer to base stations, where points refer to UEs.

In Fig. 4.1, diamonds shows base stations. Base stations are configured as omnidirectional. Points refer to UEs. UEs are distributed randomly. Thanks to the random distribution, our model will try to solve generalized problem.

4.3 Experiment I: Parameter Selection and P-DDPG Benefits

The environment measures the network status at each transmission time interval (TTI). Because of the realistic behaviours of users, the measurements of each TTI does not directly represent the network status. Since users are sometimes idle, sometimes actively use the network, the traffic load and average users' throughput is fluctuates. Therefore, the agent cannot take decisions according to each TTI measurement. The communication time interval (CTI) is defined as a state-action exchange interval between environment and agent. To find optimum CTI, we run the simulation with different CTI. Each simulation records 200 states and we measure the standard deviation of these states in term of users' throughput. Figure 4.2 shows the effect of CTI on the standard deviation of average users' throughput. When the communication time interval increases, in term of number of TTI, the fluctuation of average users' throughput decreases. This figure proves that each transmission time interval does not represent the network status and we need to consider some time interval to state-action exchange.

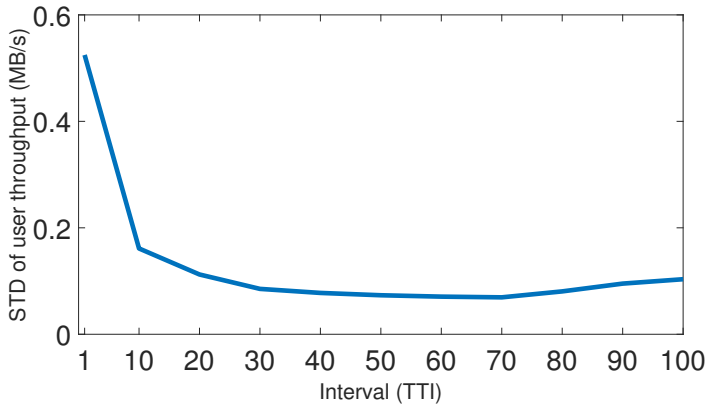


Figure 4.2: The standard deviation of the average users' throughput at different CTI values.

The second important point of CTI selection is run time. Vienna simulator is a complex, realistic LTE system level simulator and it consists of different modules. Therefore, simulation of the real-life network in Vienna Simulator is costly and it is time required task. Run times of different CTIs are shown in Fig. 4.3. Each simulation runs until recording 200 states, therefore, the simulation time on a TTI basis is calculated

as CTI $\times$ 200. When the simulation time increases, then the run time of environment is increases.

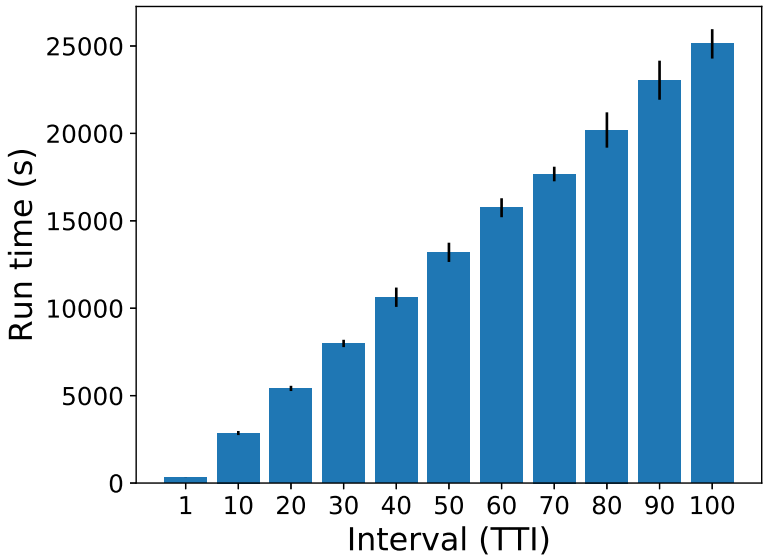


Figure 4.3: Analysis of different communication time intervals' (CTI) run time.

DDPG algorithm can train itself after each state-action exchange. The bottleneck of training can be analyzed under two categories, one of them is agent response time and another one is environment response time. In our case, the environment's run time is significantly smaller than the agent response time, so the number of training per unit time is limited by the environment's run time. We propose P-DDPG model to increase training per unit time. The P-DDPG model can run with multiple environments in parallel.

Effect of environment response time and P-DDPG algorithm are tested on *Pendulum-v0* which is one of the well-known OpenAI Gym environment. OpenAI Gym is a toolkit for developing and comparing reinforcement learning algorithms [47]. Fig. 4.4 shows the snapshot of pendulum problem. The problem is that trying to keep a pendulum standing up by taking actions. The action is a value between -2.0 and 2.0, representing the amount of left or right force on the pendulum.

Since the Vienna Simulator is a time-consuming environment as shown in Fig. 4.3, we added a delay to the pendulum environment to observe the effect of the environment run time on learning time.

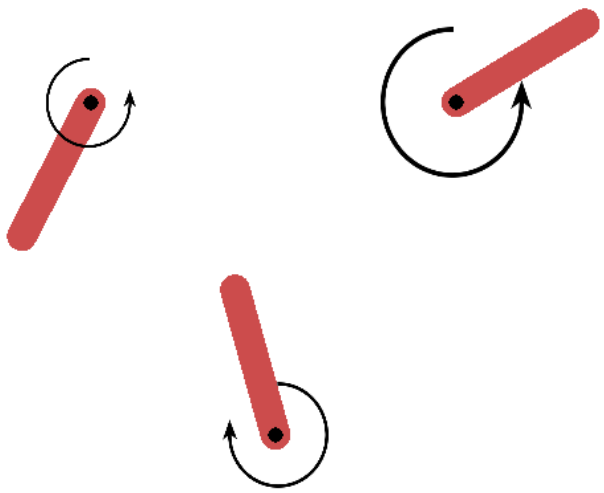


Figure 4.4: Snapshots of pendulum environment. The arrows illustrates the magnitude of the action and the direction of the action.

Fig. 4.5 shows environment run time effect on training time. We simulate pendulum problem on environments which have 0.5, 1 and 2 seconds delay. In these results, we run 10 multiple environments as a parallel and these results shows the moving average of last 100 episode rewards. According to convergence time of each simulation, when the environment run time increases, the learning time of the algorithm is also increased. The fastest trained model is achieved with the minimum delayed environment.

Running multiple environments is examined in Fig. 4.6. We try to examine the effect of environment run time and the effect of the environment count on learning. The environment counts are 10, 20 and 40 where delays are 0.5, 1 and 2 seconds. We recorded run times of simulations when the average reward value of the last 100 simulations exceeded -250. Fig 4.6a shows that when the environment count increases, algorithm learning time decrease. For each delay value, increase in the environment count positively affects the learning time. Fastest learning time is achieved with lowest delay and highest environment count. Reinforcement learning models need trial and error. The algorithm needs to be trained as much as possible to complete learning.

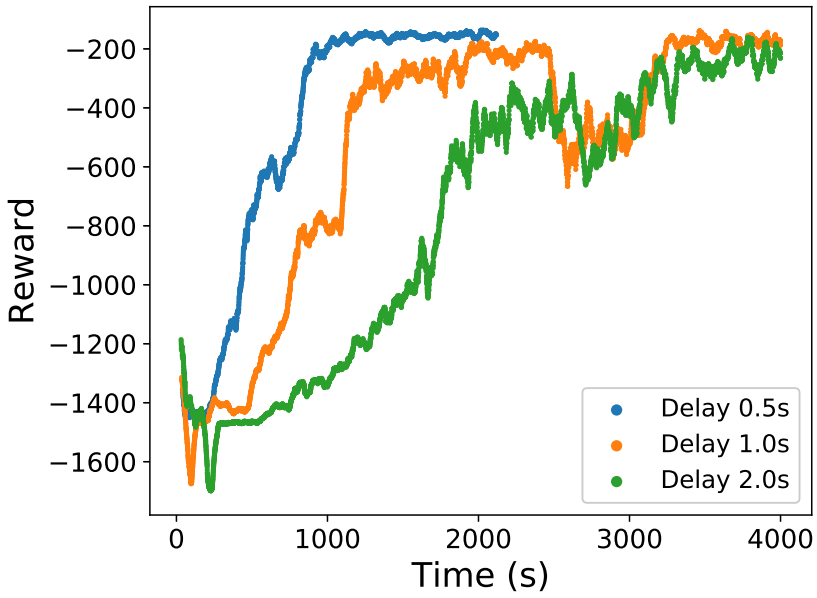
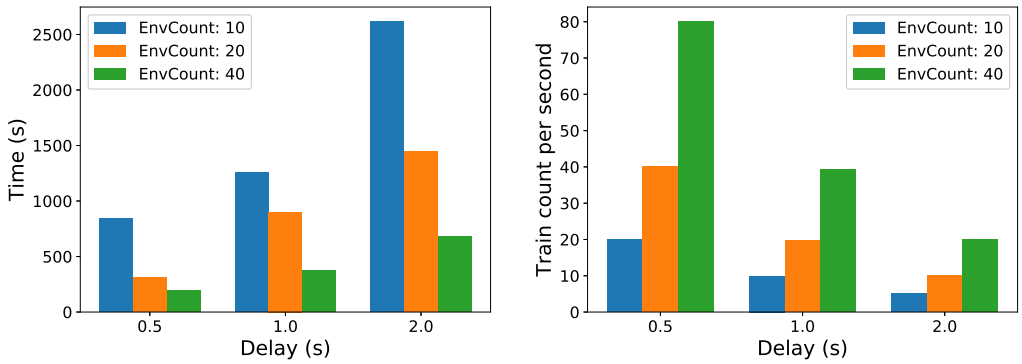


Figure 4.5: The effect of the run time of the environment on learning time.

Hence, training count per second is important to improve algorithm. Fig. 4.6b shows training count per second at different configurations. The delay and environment count effects on training per second are observable in this figure. There is an inverse ratio between delay and training count per second. Increase in the delay causes a decrease in the training count per second. Conversely, there is a direct correlation between the environment count and training count per second. When the environment count increases, then the training count per second increases.

To sum up, each TTI measurement does not represent the network status (Fig. 4.2), therefore, state-action exchange have to be done at certain intervals which we called communication time interval (CTI). Vienna Simulator is a slow environment which takes time to simulate the network (Fig. 4.3). Training is directly related with environment run time (Fig. 4.5). Therefore, while selecting communication time interval we have to consider run time and we need to choose long enough CTI that describes the network in term of low fluctuation. We empirically choose CTI length as 40 TTI with considering these reasons. Our motivation of developing P-DDPG is the slow environment. Since the environment slow, we need to run multiple environments to increase training count per second to decrease the learning time (Fig. 4.6).



(a) Training time.

(b) Training count per second.

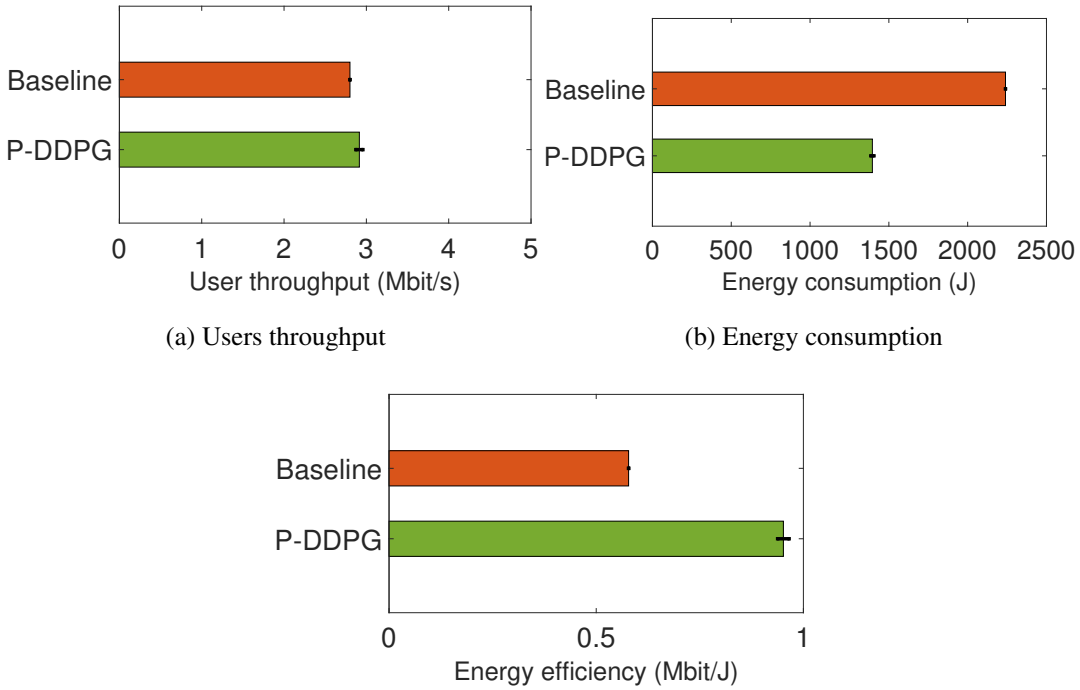
Figure 4.6: Running multiple environment effect on training time and training count per second.

4.4 Experiment II: Energy Efficiency under Stationary Scenario

Our problem is reducing the energy consumption of base stations while maintaining UE's QoS parameters. We try to solve this problem with our P-DDPG model. QoS parameters, that are defined in Equation (2) - (5) are basis on UEs' wideband SINR and UE's throughput. We first calculated the baseline values without running our model on the network. We run simulation 40 times for each scenario and we observe the network without taking any action. UEs and small cells are randomly distributed. The QoS distributions obtained in these observations are called baselines. We calculated acceptable QoS values (threshold values) for the network based on baseline values. These threshold values are used while training our model. We have compared baseline values with the last 40 results that pass CNRC.

We observed the proposed model effects in different environments. We prepare two different test scenarios to show the effect of the environmentalist energy efficient model. One of them is composed of 50 users and the other 100 users. In our observations, QoS parameters such as average throughput of users, the lowest SINR value received by users, energy consumption and energy efficiency are analyzed. The average throughput of users gives information about network usage. Users on the edge or users which are far from base stations generally get the lowest SINR value.

In that case, even if we maintain the average throughput of users, some users cannot reach the network because of the poor SINR. Therefore, we compare the lowest SINR value received by users to observe the effect of the proposed model on poor users. Energy consumption and energy efficiency are the main targets that we need to improve. The realistic natural environment results of Vienna Simulator are called the Baseline, while the results of the trained model are called P-DDPG.



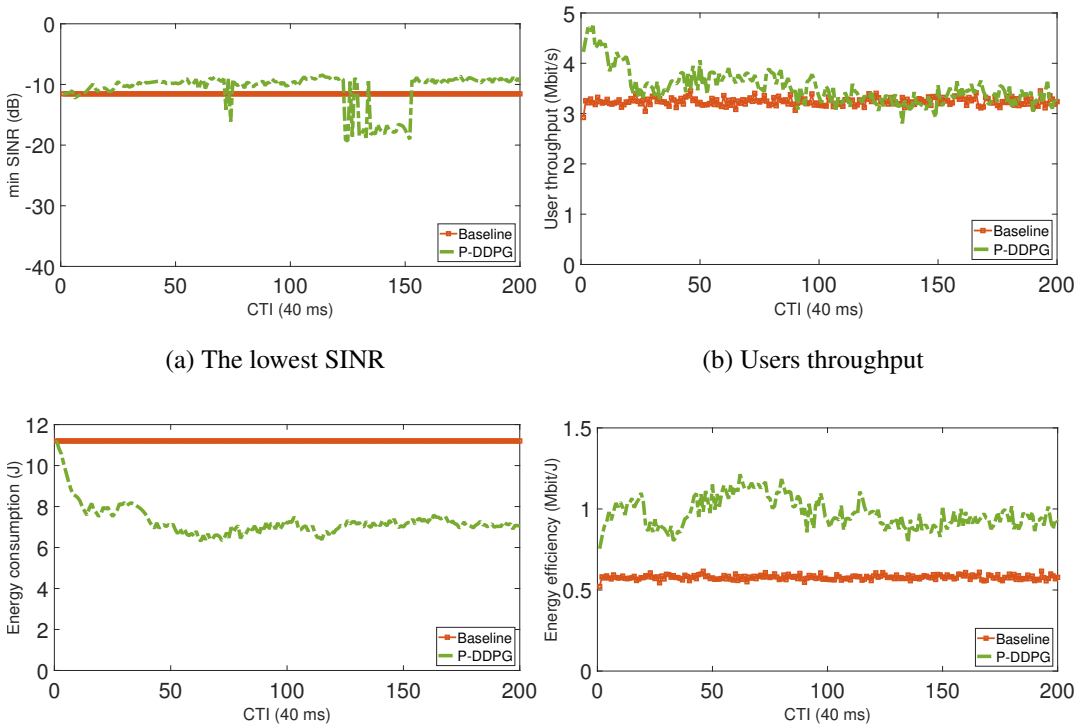
(a) Users throughput

(b) Energy consumption

(c) Energy efficiency

Figure 4.7: P-DDPG algorithm effects on the environment is compared with the baseline. In this scenario, there is 50 UEs. Last 40 simulation results are averaged.

Fig. (4.7) - (4.9) show P-DDPG effects on the environment with 50 UEs. Fig. 4.7 is obtained by evaluating 40 simulation outputs. As we can see in Fig. 4.7b, by applying the P-DDPG model, the amount of energy consumption is reduced, while the overall UEs' throughput (Fig. 4.7a) are preserved, and in some cases, they even enhanced slightly. In sparse scenarios, we achieve up to 67% increase in energy efficiency (Fig. 4.7c) by using the P-DDPG model in the dynamic environment.



(a) The lowest SINR

(b) Users throughput

(c) Energy consumption

(d) Energy efficiency

Figure 4.8: P-DDPG algorithm effects on the environment are compared with the baseline. In this scenario, there is 50 UEs. These figures show a single environment's lifetime.

In Fig. 4.8, a single environment's lifetime is presented to observe the realistic environment behaviours and the P-DDPG model effects. In this work, in order to provide a real-life condition, we simulate a dynamic network where UEs have various behavior (such as the amount of received or transmit data); therefore, QoS requirements will be varied in each TTI. Even though it is challenging to maintain the QoS parameters in this variability, our model is trained to maintain QoS while increasing energy efficiency. Fig. 4.8a shows the lowest SINR value received by users. Although there are some deviations because of the handover, the lowest SINR value is preserved. The average throughput of users is observable in Fig. 4.8b. Because of the user behaviours, there are fluctuations but the P-DDPG model successfully maintains the average throughput of users. The P-DDPG model maintains QoS parameters while reducing energy consumption. In Fig. 4.8c, the effect of applying P-DDPG on energy consumption is presented. As it is shown, the proposed model by taking varied

decisions over time can enhance energy consumption in the network for about the optimum level for the environment. The effect of fluctuations in user throughput and energy consumption on energy efficiency is also seen in Fig. 4.8d. The P-DDPG model always keeps energy efficiency higher than baseline. Due to users behaviours, throughput is changed naturally. This explains fluctuations in energy efficiency. During a single episode, model actions which are increasing or reducing BS's powers are shown in Fig. 4.9. The P-DDPG model plays with BS's power and find an optimum energy level for each of them.

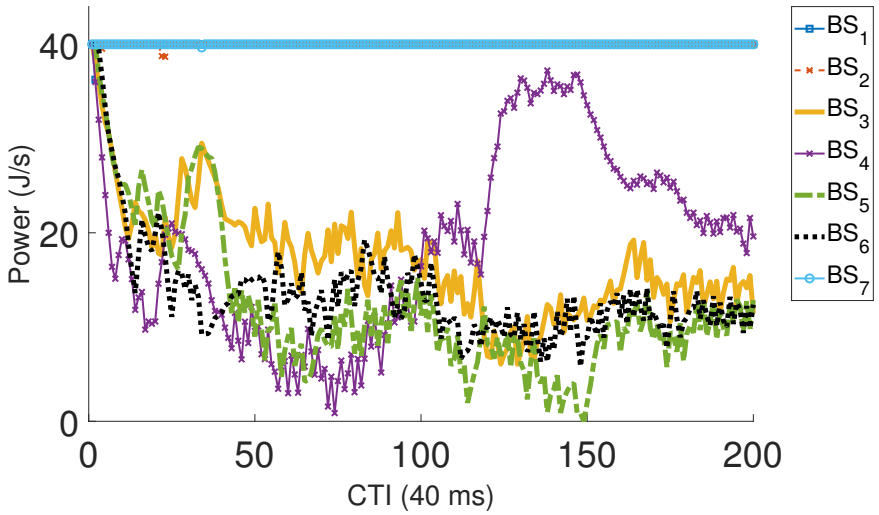
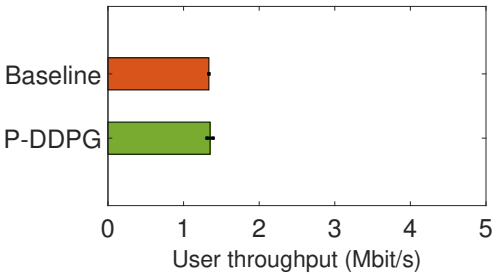
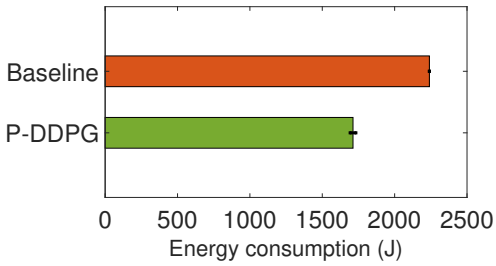


Figure 4.9: BS's power changes in a single environment's lifetime is presented. In this scenario, there is 50 UEs.

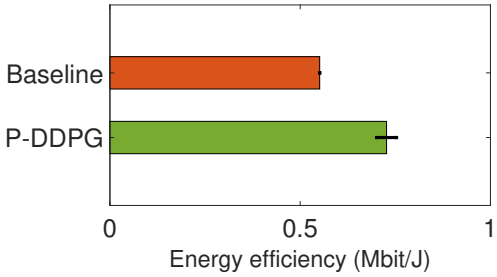
Another test scenario is constructed with 100 UEs. We illustrate the dense scenario results in Fig. 4.10 and Fig. 4.11. Like the sparse scenarios, the P-DDPG model can increase energy efficiency while maintaining QoS parameters. In the dense scenarios, we achieve up to 32% increase in energy efficiency. When the user density increases, users are more affected by power reduction of base stations in term of users' throughput. Therefore, the P-DDPG model maintains the power level of BSs at high and consume more power with respect to the sparse scenarios to protect the QoS. Hence, the increase in energy efficiency is less than the sparse scenarios. 40 simulations' average results are presented in Fig. 4.10. As we can see in this figure, the P-DDPG model efficiently maintained the QoS parameters during the simulation. As it is shown, during multiple simulations, average power consumption is reduced and energy efficiency is increased when we use the P-DDPG model.



(a) Users throughput



(b) Energy consumption

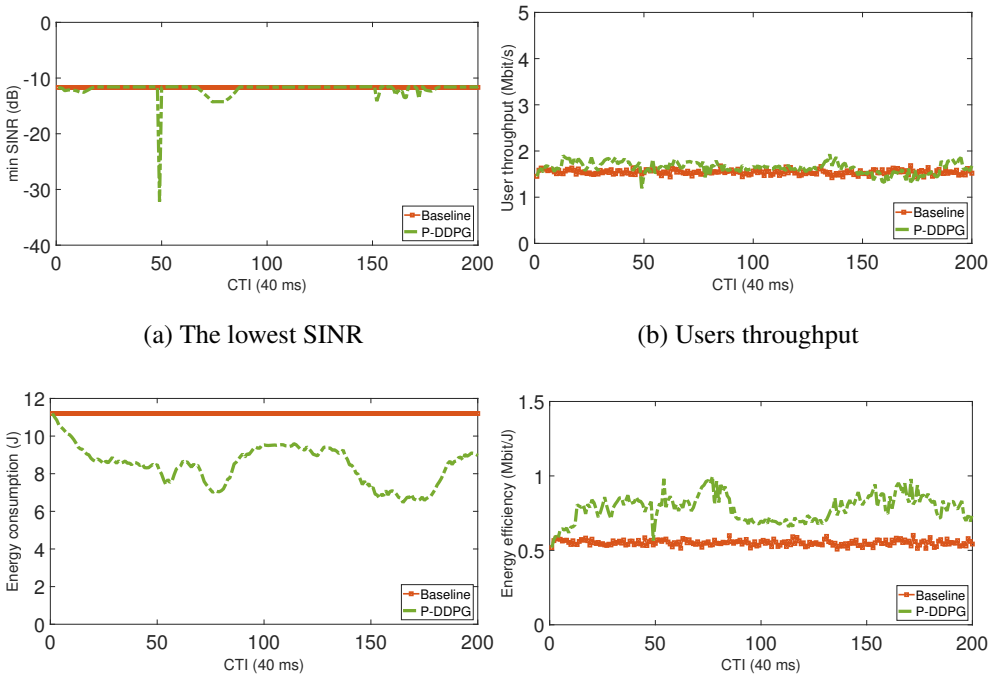


(c) Energy efficiency

Figure 4.10: P-DDPG algorithm effects on the environment is compared with the baseline. In this scenario, there is 100 UEs. Last 40 simulation results are averaged.

Fig. 4.11a and Fig. 4.11b shows the lowest SINR value received by users and throughput changes during one episode in the dense scenario. It appears in these figures that the P-DDPG model maintains SINR and UEs' throughput. There is a sharp reduction in the lowest SINR at 49th CTI, which may be caused handover. Except for 49th CTI, the model maintains the lowest SINR value, similar to the base. BS's power changes in the dense scenario is shown in Fig. 4.9. The P-DDPG model acts according to the scenario requirements and keeps QoS above the threshold, and reduces energy consumption. By reducing BS's powers, it increases energy efficiency as shown in Fig. 4.11d.

In both sparse and dense cases, the P-DDPG model is trained and it decreases energy consumption. The P-DDPG model has the capability of handling natural user behaviour, and it can find an optimum energy consumption level for both cases.



(a) The lowest SINR

(b) Users throughput

(c) Energy consumption

(d) Energy efficiency

Figure 4.11: P-DDPG algorithm effects on the environment are compared with the baseline. In this scenario, there is 100 UEs. A single environment's lifetime is presented.

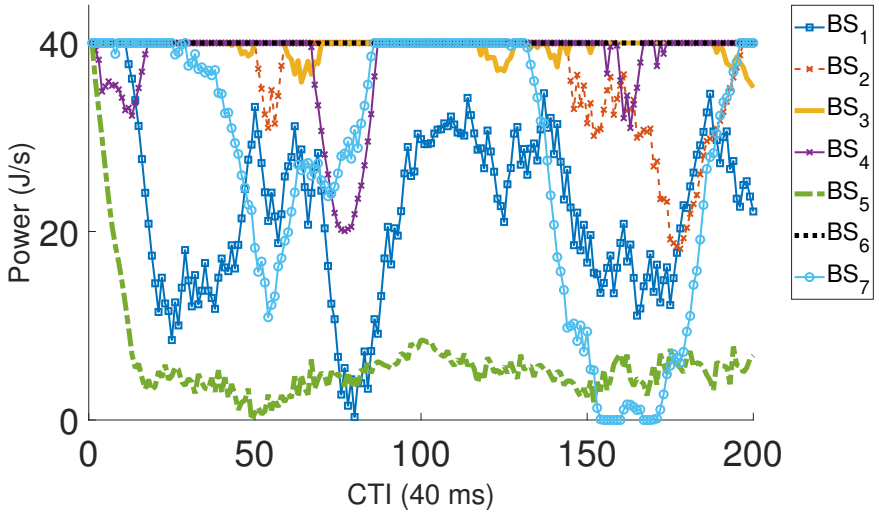


Figure 4.12: BS's power changes in a single environment's lifetime is presented. In this scenario, there is 100 UEs.

4.5 Experiment III: Energy Efficiency under Perturbation in the Network

The P-DDPG model stationary scenarios' energy efficiency results are presented in the previous section. In this section, we test the P-DDPG model when the perturbation occurs in the network. The base stations could shut down suddenly because of different reasons such as internal errors, firmware update, hardware change. Therefore, the proposed models should have a tolerance to sudden base stations turn off. After restarting the system, the proposed models should be able to continue its management. To test this scenario, we set up a simulation with 50 users and 7 base station. We set simulation length to 200 CTI. At 25th CTI, we close 3 out of 7 base station. In Fig. 4.13, the squares show the closed base stations and the diamonds show the rest. At 50th CTI, we restart the system and set the closed base station powers to the maximum level. During that time, model actions are neglected and we keep base station powers constant.

The reward function is updated for this scenario. During $25 \leq CTI \leq 50$, the model is neither rewarded nor punished. We set $R = 0$ during perturbation. For the perturbation scenario, the reward function is formulated like: $R^k = min(R^k_{\text{OS}_2}, R^k_{EC})$, where R_{OS_2} refers to overall status of users' throughput reward (Eq. (3.4) - (3.12)) and R_{EC} refers to energy consumption reward (Eq. (3.13)).

We illustrate the perturbation scenario results in Fig. 4.14 and Fig. 4.15. In perturbation scenario, P-DDPG model can increase energy efficiency while maintaining QoS parameters. We achieve up to 39% increase in energy efficiency.

In this experiment, the average throughput of users and the lowest SINR value received by users are compared. The average throughput of users gives information about network usage. Hence, the average throughput of users is important with respect to maintaining QoS. QoS is not only depended on average throughput. Users on the edge or users which are far from base stations generally get the lowest SINR value. In that case, even if we maintain the average throughput of users, some users cannot reach the network because of the poor SINR. Therefore, we also try to maintain the lowest SINR value received by users to consider poor users.

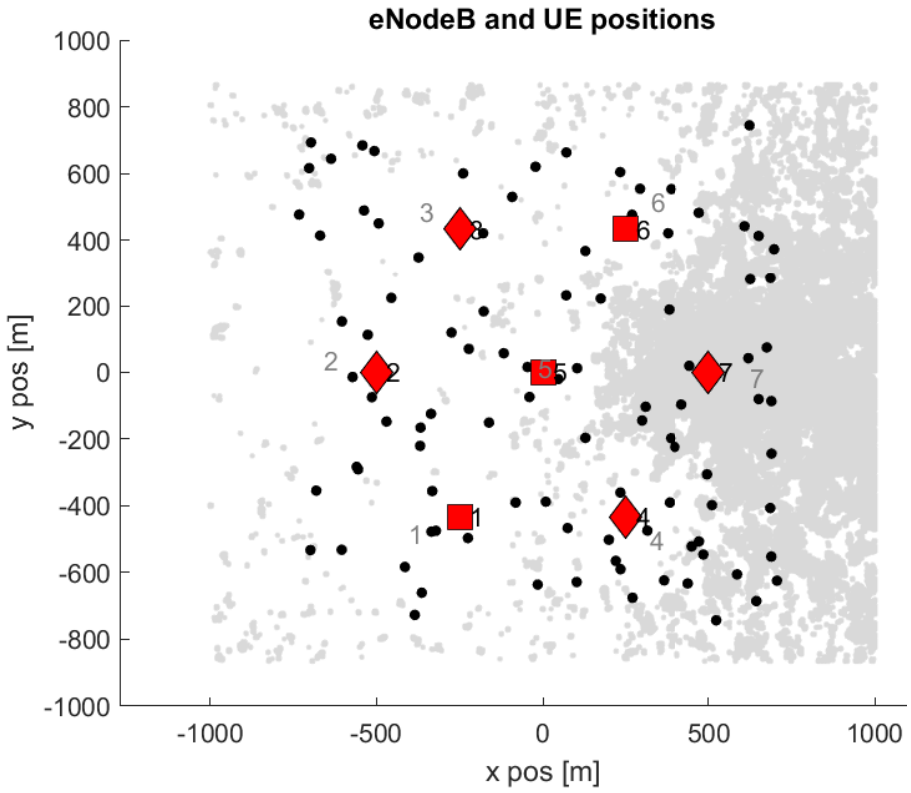
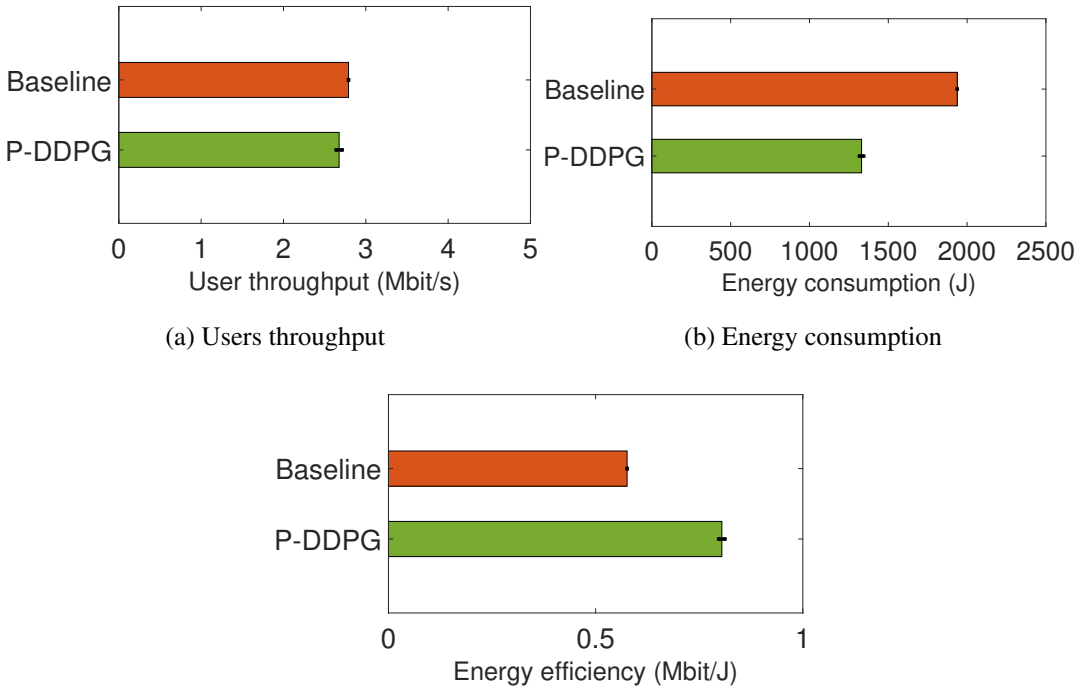


Figure 4.13: Perturbation scenario illustration. Squares are the base stations that closed at 25th CTI and restarted at 50th CTI. Diamonds are other base stations and dots show the users.

Fig. 4.14 is obtained by evaluating 40 simulations outputs. As shown in this figure, the P-DDPG model efficiently maintained the QoS parameters during the simulation like previous scenarios. Fig. 4.14a compares the average throughput of users. The average throughput of users almost preserved (Fig. 4.14a) while reducing energy consumption (Fig. 4.14b). There is a 4% decrease in the average throughput of users where energy consumption is reduced by 30%. As a result, energy efficiency is increased by 39% by using the P-DDPG model (Fig. 4.14c).

In Fig. 4.15, a single environment's lifetime is presented. The base station sudden shutdown effect is manifestly observable in these figures. There is a significant change between 25th CTI and 50th CTI. The shutting down 3 out of 7 base stations at 25th CTI sharply decrease average throughput of users (Fig. 4.15b).



(a) Users throughput

(b) Energy consumption

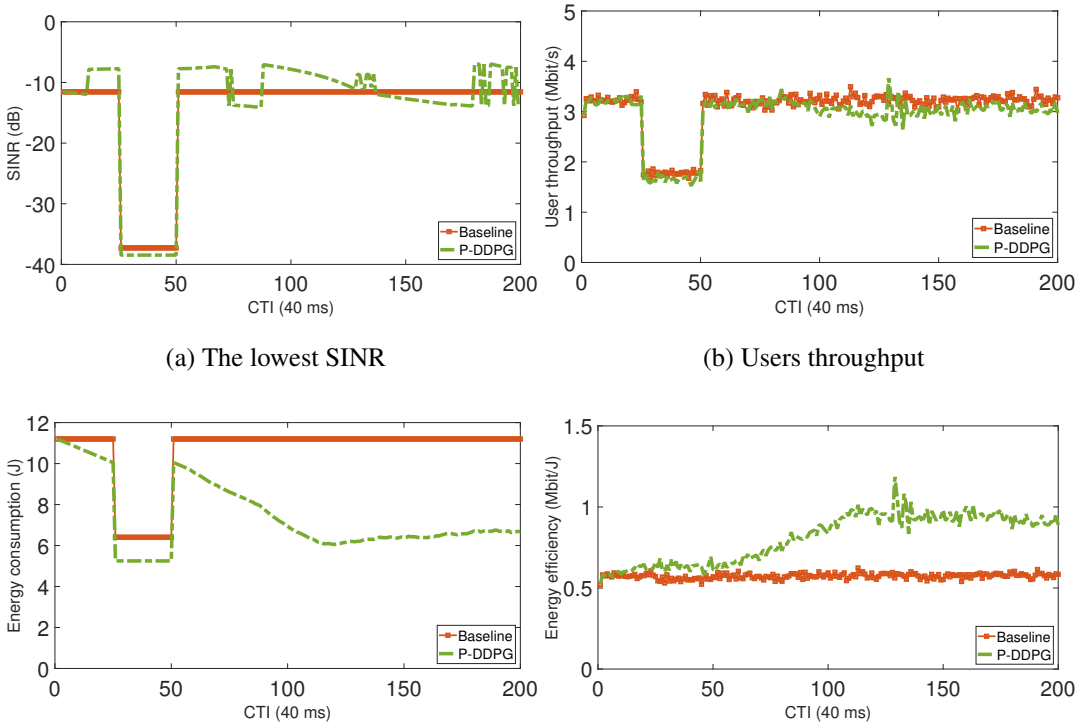
(c) Energy efficiency

Figure 4.14: P-DDPG algorithm effects on the environment is compared with the baseline. In this scenario, 3 out of 7 base station suddenly shut down and they are restarted later. Last 40 simulation results are averaged.

Fig. 4.15a shows the lowest SINR value that is received from users. There is a sudden decrease in the lowest SINR value because of the shutdown.

After restarting closed base stations at 50th CTI, the P-DDPG model successfully continue to the management of the network. The average throughput of users is almost equally maintained with baseline. By using the P-DDPG model, energy consumption is significantly reduced after 50th CTI. The model tries to find optimum energy consumption level by changing base stations transmit powers (Fig. 4.15c). During that time, it increases energy efficiency (Fig. 4.15d) and preserves the average throughput of users (Fig. 4.15b).

Effects of the sudden shutdown and the model actions on base stations' transmit powers are observable in Fig. 4.16. The base stations $[1, 5, 6]$ are turned off during $25 \leq CTI \leq 50$. After 50th CTI, the P-DDPG model plays with BSs' transmit powers to find the optimum energy consumption level for each of them.



(a) The lowest SINR

(b) Users throughput

(c) Energy consumption

(d) Energy efficiency

Figure 4.15: P-DDPG algorithm effects on the environment are compared with the baseline. In this scenario, 3 out of 7 base station suddenly shut down and they are restarted later. A single environment's lifetime is presented.

To sum up, in this experiment the perturbation scenario is tested. The base stations could shut down suddenly because of different reasons such as internal errors, firmware update, hardware change. Users' QoS parameters adversely affected by sudden shutdowns. The P-DDPG model has the capability of continuing its management after restarting the system. It maintains users' QoS parameters while increasing energy efficiency. In perturbation scenario, we achieve up to a 39% increase in energy efficiency.

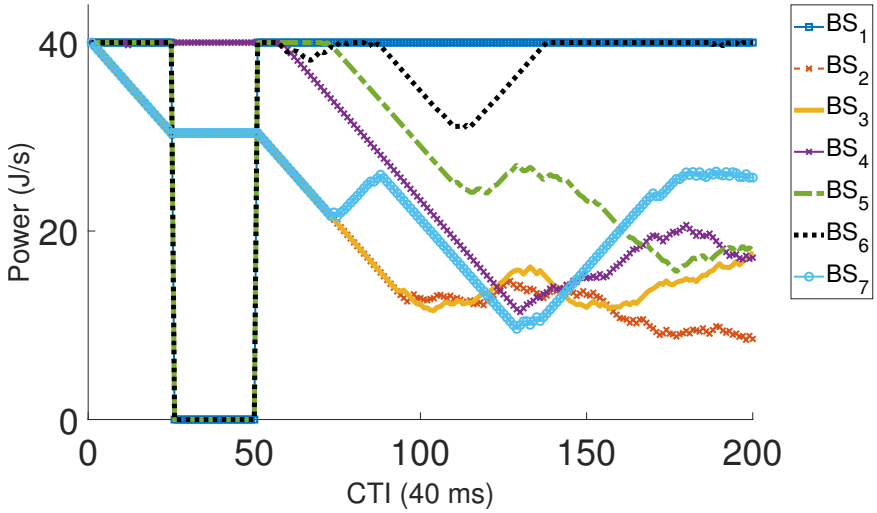


Figure 4.16: BS's power changes in a single environment's lifetime is presented. In this scenario, 3 out of 7 base station suddenly shut down and they are restarted later. A single environment's lifetime is presented.

CHAPTER 5

CONCLUSION AND FUTURE WORK

In the next-generation networks, due to the increase in the BS density, environmentalist approach will be essential. The increase in the energy consumption of mobile networks negatively affects the environment and causes higher operational expenses (OPEX) for mobile service providers. Therefore, energy-efficient algorithm are essential to reduce energy consumption and OPEX. The network is a highly dynamic environment that is hard to describe with mathematical models. Moreover, autonomous systems are required to manage network cleverly and optimally. Therefore, in the next-generation networks, machine learning will be more involved to solve the problems. These machine learning models should be trained in a simulation environment which is close to real life. Without using real-life simulators, machine learning models will not be able to adapt to the real-life system.

In this thesis, we present a dynamic machine learning based energy saving model which is trained with Vienna Simulator that is known as realistic network simulator. The disadvantage of using a realistic simulator is that because the environment simulation is slow, the learning time will be extended and it will take time for the machine learning model to adapt. We propose P-DDPG that is an extended version of DDPG which has the capability of work with multiple environments as a parallel. Thanks to the P-DDPG, we demonstrate that we can provide faster learning in slower environments. We analyze the relationship between environment run time and learning time. Our results show that an increase in the environment run time causes an increase in the learning time. In addition to that, we analyze running multiple environment effect on learning. Our experiments prove that the increase in the number of parallel environments reduces learning time.

Reinforcement learning can be a remedy for reducing energy consumption in the network. We adapt the P-DDPG, which basis on DDPG that belongs to RL family, to the network environment. We define state, action and reward for the network environment. The network status is defined in the state vector. Action vector is defined as a change in the transmit power of the group of the base station. According to QoS parameters, given the feedback signal is defined as a reward. Our experiments show that it is possible to achieve up to 32% increase in the energy efficiency in a dense scenario and up to 67% increase in sparse scenarios. P-DDPG successfully manages networks in term of reducing energy consumption and maintaining QoS.

As future work, the proposed approach can be tested in environments with mobile users. By taking advantage of transfer learning, it is possible to work in environments with mobile users as a continuation of this study. The proposed model can be tested on different network scenarios. These scenarios can be clustered with ML algorithms. A fully autonomous system running on the network can be obtainable by using P-DDPG models that are trained for each cluster.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. Cambridge, Massachusetts: The MIT Press, 2018.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," in *Neural Information Processing Systems (NIPS) Workshop on Deep Learning*, 2013.

[3] 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," in *4th International Conference on Learning Representations*, May 2016.

[4] A. Abrol and R. K. Jha, "Power Optimization in 5G Networks: A Step Towards GrEEn Communication," *IEEE Access*, vol. 4, pp. 1355–1374, Apr. 2016.

[5] E. C. Strinati and L. Herault, "Holistic approach for future energy efficient cellular networks," *Elektrotechnik und Informationstechnik*, vol. 127, pp. 314–320, Nov. 2010.

[6] S. Zhou, J. Gong, Z. Yang, Z. Niu, P. Yang, and D. Corporation, "Green mobile access network with dynamic base station energy saving," *ACM MobiCom*, vol. 9, pp. 10–12, Jan. 2009.

[7] M. Ismail, W. Zhuang, E. Serpedin, and K. Qaraqe, "A survey on green mobile networking: From the perspectives of network operators and mobile users," *IEEE Communications Surveys and Tutorials*, vol. 17, pp. 1535–1556, thirdquarter 2015.

[8] M. Aykut Yigitel, O. D. Incel, and C. Ersoy, "Dynamic BS Topology Management for Green Next Generation HetNets: An Urban Case Study," *IEEE Journal on Selected Areas in Communications*, vol. 34, pp. 3482–3498, Dec. 2016.

[9] M. Feng, S. Mao, and T. Jiang, "Base Station ON-OFF Switching in 5G Wire-

less Systems: Approaches and Challenges," *IEEE Wireless Communications*, vol. 24, pp. 46–54, Aug. 2017.

[10] A. Al-Quzweeni, A. Lawey, T. El-Gorashi, and J. M. H. Elmirghani, "A framework for energy efficient NFV in 5G networks," in *2016 18th International Conference on Transparent Optical Networks (ICTON)*, pp. 1–4, July 2016.

[11] S. Mollahasani and E. Onur, "Density-Aware, Energy- and Spectrum-Efficient Small Cell Scheduling," *IEEE Access*, vol. 7, pp. 65852–65869, May 2019.

[12] T. E. Bogale, X. Wang, and L. B. Le, "Machine Intelligence Techniques for Next-Generation Context-Aware Wireless Networks," *ITU Journal: ICT Discoveries, Special Issue*, pp. 1–11, Feb. 2018.

[13] C. Jiang, H. Zhang, Y. Ren, Z. Han, K. Chen, and L. Hanzo, "Machine Learning Paradigms for Next-Generation Wireless Networks," *IEEE Wireless Communications*, vol. 24, pp. 98–105, Apr. 2017.

[14] M. G. Kibria, K. Nguyen, G. P. Villardi, O. Zhao, K. Ishizu, and F. Kojima, "Big Data Analytics, Machine Learning and Artificial Intelligence in Next-Generation Wireless Networks," *IEEE access*, vol. 6, pp. 32328–32338, May 2018.

[15] T. M. Mitchell, *Machine Learning*. New York, NY, USA: McGraw-Hill, Inc., 1 ed., 1997.

[16] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[17] A. G. Barto, P. Thomas, and R. S. Sutton, "Some recent applications of reinforcement learning," in *Proceedings of the Eighteenth Yale Workshop on Adaptive and Learning Systems*, 2017.

[18] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, pp. 50–56, ACM, Nov 2016.

[19] I. Arel, C. Liu, T. Urbanik, and A. G. Kohls, "Reinforcement learning-based

multi-agent system for network traffic signal control," *IET Intelligent Transport Systems*, vol. 4, pp. 128–135, June 2010.

[20] Z. Zhou, X. Li, and R. Zare, "Optimizing chemical reactions with deep reinforcement learning," *ACS Central Science*, vol. 3, Dec 2017.

[21] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," *Electronic Imaging*, vol. 2017, no. 19, pp. 70–76, 2017.

[22] A. S. Polydoros and L. Nalpantidis, "Survey of model-based reinforcement learning: Applications on robotics," *Journal of Intelligent & Robotic Systems*, vol. 86, pp. 153–173, May 2017.

[23] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. R. Baker, M. Lai, A. Bolton, Y. Chen, T. P. Lillicrap, F. F. C. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of go without human knowledge," *Nature*, vol. 550, pp. 354–359, 2017.

[24] J. Kober, J. Andrew Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, pp. 1238–1274, 09 2013.

[25] I. Comşa, S. Zhang, M. E. Aydin, P. Kuonen, Y. Lu, R. Trestian, and G. Ghinea, "Towards 5g: A reinforcement learning-based scheduling solution for data traffic management," *IEEE Transactions on Network and Service Management*, vol. 15, pp. 1661–1675, Dec 2018.

[26] Z. Xiong, Y. Zhang, D. Niyato, R. Deng, P. Wang, and L. Wang, "Deep reinforcement learning for mobile 5g and beyond: Fundamentals, applications, and challenges," *IEEE Vehicular Technology Magazine*, vol. 14, pp. 44–52, June 2019.

[27] D. Silver, "Lecture 4: Model-Free Prediction." University College London Lecture, Dec 2018.

[28] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine Learning*, vol. 3, pp. 9–44, Aug 1988.

[29] C. J. C. H. Watkins, *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK, May 1989.

[30] T. Degris, M. White, and R. Sutton, "Off-policy actor-critic," *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*, vol. 1, May 2012.

[31] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," *Society for Industrial and Applied Mathematics*, vol. 42, April 2001.

[32] C. Liu, B. Natarajan, and H. Xia, "Small Cell Base Station Sleep Strategies for Energy Efficiency," *IEEE Transactions on Vehicular Technology*, vol. 65, pp. 1652–1661, Mar. 2016.

[33] S. Cai, Y. Che, L. Duan, J. Wang, S. Zhou, and R. Zhang, "Green 5G Heterogeneous Networks Through Dynamic Small-Cell Operation," *IEEE Journal on Selected Areas in Communications*, vol. 34, pp. 1103–1115, May 2016.

[34] H. Lu, B. Hu, Z. Ma, and S. Wen, "Reinforcement learning optimization for energy-efficient cellular networks with coordinated multipoint communications," *Mathematical Problems in Engineering*, vol. 2014, pp. 1–9, July 2014.

[35] S. Sharma, S. J. Darak, and A. Srivastava, "Energy saving in heterogeneous cellular network via transfer reinforcement learning based policy," in *9th International Conference on Communication Systems and Networks (COMSNETS)*, pp. 397–398, Jan. 2017.

[36] I. AlQerm and B. Shihada, "Enhanced machine learning scheme for energy efficient resource allocation in 5G heterogeneous cloud radio access networks," in *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pp. 1–7, Oct. 2017.

[37] E. Ghadimi, F. Davide Calabrese, G. Peters, and P. Soldati, "A reinforcement learning approach to power control and rate adaptation in cellular networks," in *2017 IEEE International Conference on Communications (ICC)*, pp. 1–7, May 2017.

[38] Y. Wang, X. Dai, J. M. Wang, and B. Bensaou, "A Reinforcement Learning Approach to Energy Efficiency and QoS in 5G Wireless Networks," *IEEE Journal on Selected Areas in Communications*, vol. 37, pp. 1413–1423, June 2019.

[39] L. Raju, R. Milton, S. Suresh, and S. Sankar, "Reinforcement learning in adaptive control of power system generation," *Procedia Computer Science*, vol. 46, pp. 202–209, Dec. 2015.

[40] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic Policy Gradient Algorithms," *31st International Conference on Machine Learning, ICML 2014*, vol. 1, June 2014.

[41] M. Rupp, S. Schwarz, and M. Taranetz, *The Vienna LTE-Advanced Simulators: Up and Downlink, Link and System Level Simulation*. Signals and Communication Technology, Springer Singapore, 1 ed., 2016.

[42] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[43] 3rd Generation Partnership Project (3GPP), "Evolved universal terrestrial radio access (EUTRA)," Tech. Rep. TR 36.942, radio frequency (RF) system scenarios, in 3rdGenerationPartnership Project (3GPP), 10 2014.

[44] H. P. Keeler, B. Blaszczyszyn, and M. K. Karray, "Sinr-based k-coverage probability in cellular networks with arbitrary shadowing," in *2013 IEEE International Symposium on Information Theory*, pp. 1167–1171, July 2013.

[45] B. Clerckx, G. Kim, and S. Kim, "Mu-mimo with channel statistics-based codebooks in spatially correlated channels," in *IEEE Global Telecommunications Conference*, pp. 1–5, Nov. 2008.

[46] G. Karagiannis, G. T. Pham, A. D. Nguyen, G. J. Heijenk, B. R. Haverkort, and F. Campfens, “Performance of LTE for Smart Grid Communications,” in *Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance*, pp. 225–239, Springer International Publishing, Jan. 2014.

[47] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, “Openai gym,” 2016.