SAKARYA UNIVERSITY JOURNAL OF COMPUTER AND INFORMATION SCIENCES
VOL. 4, NO. 2, AUGUST 2021

Research Article

Base Station Power Optimization for Green Networks Using Reinforcement Learning

Semih Aktaş[1], Hande Alemdar[2]
[1] Department of Computer Engineering, Middle East Technical University, Turkey; aktas.semih@metu.edu.tr;
[2] Corresponding Author; Department of Computer Engineering, Middle East Technical University, Turkey;
alemdar@metu.edu.tr; +90 312 210 55 91

Received 4 May 2021; Revised 5 July 2021; Accepted 2 August 2021; Published online 30 August 2021

Abstract

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 environment, 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, LTE

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 [1, 2]. Mobile service providers are among the top energy consumers [3]. The increase in the energy consumption of mobile networks negatively affects the environment and causes higher operational expenses for mobile service providers. Therefore, "green network" approaches become more popular to reduce energy consumption as well as the cost [4, 5, 6].

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 networks (SN), 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. The sleeping strategy is considered as an approach for energy saving [7, 8]. 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) [9]. 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, network operators expand their investigations in 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 Figure 1.

To Cite This Article: S. Aktas, H. Alemdar, "Base Station Power Optimization for Green Networks Using Reinforcement Learning," Sakarya University Journal of Computer and Information Sciences, vol. 4, no. 2, pp. 244–265, 2021. doi: 10.35377/saucis.04.02.932709

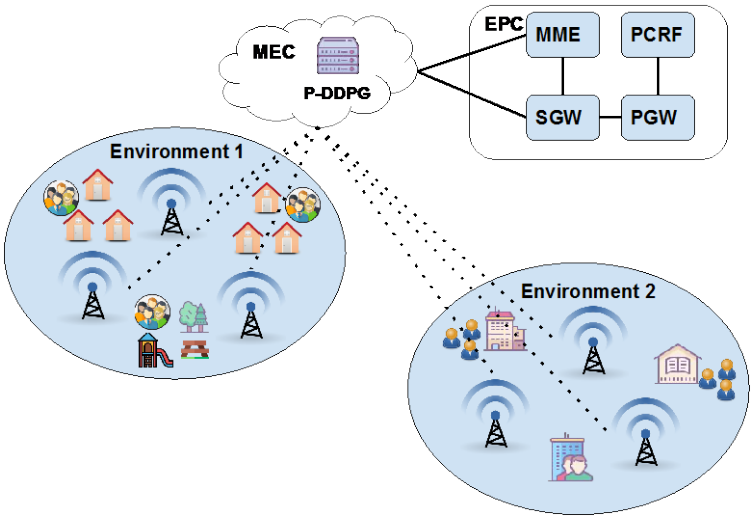


Figure 1 P-DDPG Network Architecture

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 [10]. With the development of SN, 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 [11, 12, 13]. 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 [10] 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 in this paper are:

- We developed P-DDPG algorithm, which enables DDPG to work for parallel environments (Section 2.2). 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 2.3).

- Simulation parameters are given for future reproduce. Simulation results and detailed parameter analysis are presented (Section 3).

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 sparser scenarios while preserving user QoS parameters such as throughput and Signal to Interference Ratio (SINR).

2. Modelling P-DDPG for Energy Efficient Base Station Control

In this section, the evolution of DDPG algorithm, our extended DDPG model which is P-DDPG and the details of our RL scheme are described.

2.1 Background of Reinforcement Model

Reinforcement learning is a type of machine learning which is focused on goal-directed learning from interaction [14]. RL is an efficient method for sequential decision-making problems, making them ideal for network management [15]. 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.

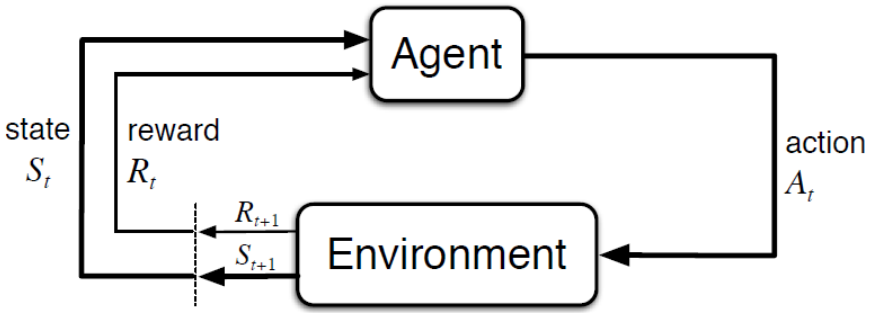


Figure 2 Overview of Reinforcement Learning Algorithms [12]

The general workflow of RL algorithms is summarized in Figure 2. 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.

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 [16]. It is formulated as follows:

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

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 pair. 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, which is also called as the "Deep Q Network" (DQN) [17]. Deep deterministic policy gradient (DDPG) algorithm is developed for continuous control with deep reinforcement learning [10]. DDPG is a combination of the deterministic policy gradient (DPG) algorithm [18] with the DQN. In this work, we employ the DDPG

approach since our space is continuous. In the next section, we provide the details of our approach and our extension to the original DDPG approach.

2.2 P-DDPG Model

As we mentioned previously, the proposed algorithm is implemented in a MEC in order to virtualize RAN functions. In this section, we explain the P-DDPG algorithm in detail.

Traditionally, DDPG algorithm works with a single environment. As can be seen in Figure 2, 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).

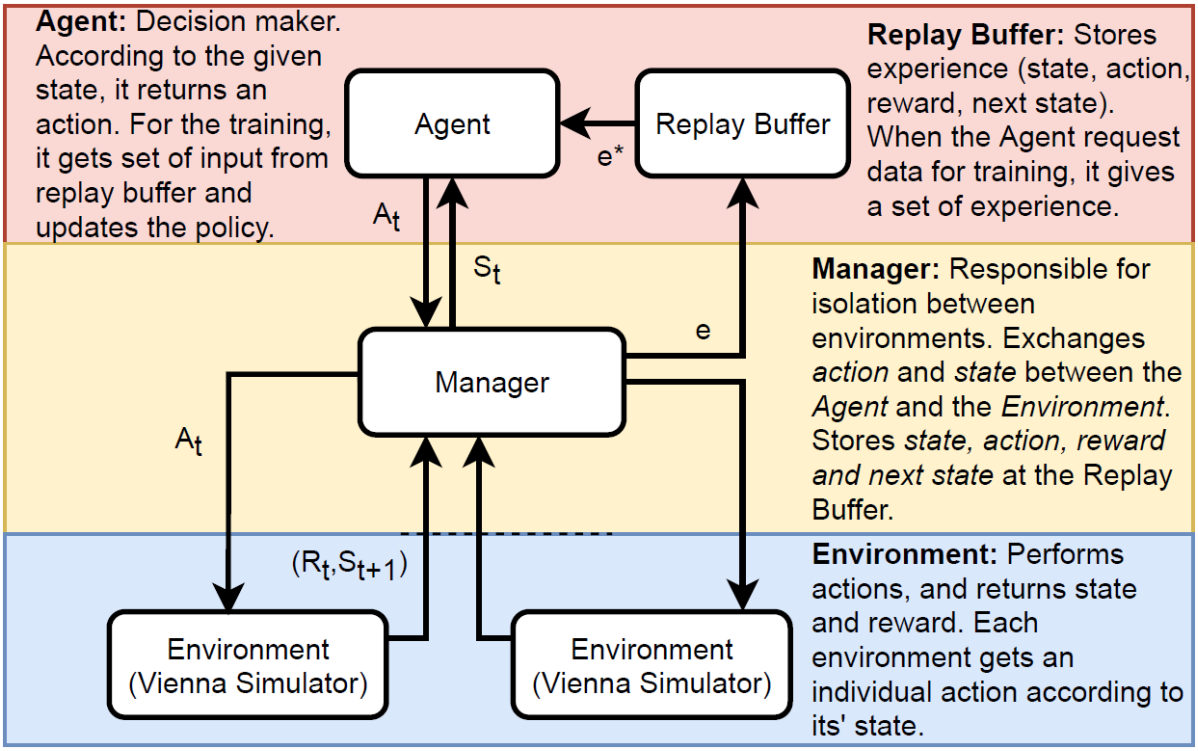


Figure 3 Structure of P-DDPG Algorithm with Parallel Environment

Figure 3 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. 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 (S_t), action (A_t), reward (R_t) and next state (action result, S_{t+1}) in replay buffer [17]. Each replay buffer entry is called as experience (e) and the agent uses replay buffer entities for batch training. It trains deep neural networks by using samples from replay buffer. The DDPG algorithm [10] psudocode is followed for policy updating and state-action exchange.

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 [19], 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.

2.3 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 behavior 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 words, each RL cycle period that is depicted in Figure 2 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 (MB) that transmitted by the user at specific CTI is represented as $\Psi^k(u)$. User active TTI count for CTI^k notated as $\varphi^k(u)$.

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:

$$S^k = \left(OS_i^k, S1_j^k, S2_j^k, S3_j^k, S4_j^k\right), \tag{2}$$

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

$$OS_1^k = \frac{\sum_{u \in U} SINR^k(u) \times \phi^k(u)}{\sum_{u \in U} \phi^k(u)} \tag{3}$$

In Equation 3, users' wideband SINR average value is calculated when users are actively using communication channels. Overall network throughput is formulated as

$$OS_2^k = \frac{\sum_{u \in U} \Psi^k(u)}{t} \tag{4}$$

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

$$OS_3^k = \frac{\sum_{u \in U} \Psi^k(u)}{\sum_{u \in U} \phi^k(u)} \tag{5}$$

Users who are close to base stations can mislead Equation 4 but averaging with active TTI count will normalize throughput considering idle users. Average throughput per user is formulated as

$$OS_4^k = \frac{\sum_{u \in U} \Psi^k(u)}{|U| \times t} \tag{6}$$

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 in Equation 3 shows the possibility of transmitting data while users' throughput sum in Equation 4 shows the actual usage. Equation 5 considers idle users. Users' throughput sum in Equation 4 and users' throughput average in Equation 6 change when the number of users change. Therefore, each OS value has a unique usage and give information about network. OS values are used as QoS parameters.

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

$$S1_j^k = \frac{P^k(BS_j)}{P_{\max}}, \tag{7}$$

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

$$S2_j^k = \left|U_j^k\right|, \tag{8}$$

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

$$S3_j^k = \frac{\sum_{u \in BS_j} SINR^k(u) \times \phi^k(u)}{\sum_{u \in BS_j} \phi^k(u)}, \tag{9}$$

where $u \in BS_j$ refers to users that assigned to BS_j. Equation 9 gives information about users' wideband SINR average value according to their active TTI count. Users' average throughput according to their active TTI count is formulated as

$$S4_j^k = \frac{\sum_{u \in BS_j} \Psi^k(u)}{\sum_{u \in BS_j} \phi^k(u)}, \tag{10}$$

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 $a^k = (\Delta P_1, \Delta P_2, \ldots, \Delta P_N)$ 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:

$$P^{k+1}(BS_i) = P^k(BS_i) + \left[a_i^k \times P_{max}\right] \tag{11}$$

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 as follows

$$R_{OS_i}^k = \begin{cases} 1, & if \;\; th_i^u < OS_i \\ -\dfrac{th_i^u - OS_i}{th_i^u - th_i^l}, & if \;\; th_i^u \geq OS_i > th_i^l \\ -1, & otherwise \end{cases} \tag{12}$$

where th^u_i and th^l_i refers to upper and lower threshold values for OS_i. When OS_i is below th^u_i 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

$$R_{EC}^{k} = 1 - \frac{\sum_{bs \in BS} S_1^k(bs)}{|BS|} + \epsilon, \tag{13}$$

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

$$R^k = min\left(R_{\mathrm{OS}_1}^k, R_{\mathrm{OS}_2}^k, R_{\mathrm{OS}_3}^k, R_{\mathrm{OS}_4}^k, R_{EC}^k\right) \tag{14}$$

At CTI^k, if any of OS_i is below $th^u{}_i$ 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

$$EE = \frac{\sum_{u \in U} \Psi(u)}{\sum_{bs \in BS} P(bs) \times t}, \tag{15}$$

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

Experience: Each replay buffer entity is called an *experience*. The experience is a combination of *state, action, reward,* and *next state* (action result). It is denoted as $e^k = (S^k, A^k, R^k, S^{k+1})$. These entities are used to train deep neural networks.

3. Experiments

In this section, 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 3.1 – 3.2. 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 3.3 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 3.3 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 user equipments (UEs) and 100 user equipments. We train our algorithm on these scenarios independently. Section 3.4 provides the details 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 3.5 gives the modelling details and experiment results.

3.1 P-DDPG Model and Simulation Environment

The P-DDPG model is developed by using TensorFlow 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 [10]. Actor and critic networks are also referenced from there. Table 1 shows parameters of P-DDPG algorithm. In order to improve the learning stability, target networks are updated with the learning parameter of τ. This means that target network values slowly track the learned network with respect to that parameter.

Table 1 P-DDPG Parameters

Parameter	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

3.2 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 [20]. In this simulation, we focused on a downlink scenario where users' packet requests are varying during the runtime. The details of the SINR calculation and environment pathloss are stated in [21]. The author notes that in their formulation, the constant K at 6910 KM^{-1} corresponds to the COST Walfisch-Ikegami model for an urban environment. The pathloss simulation parameters and traffic models are summarized in Table 2.

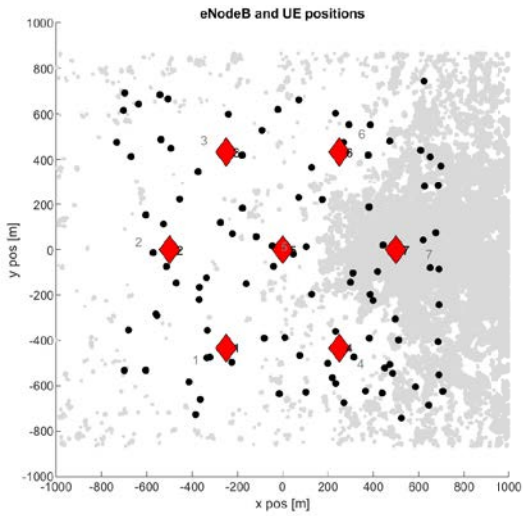


Figure 4 Example Distribution of Environment from Vienna Simulator

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.

In Figure 4, diamonds show base stations. Base stations are configured as omni-directional. Points refer to UEs. UEs are distributed randomly. Thanks to the random distribution, our model will try to solve generalized problem.

Table 2 Simulation Parameters

Parameter	Value	Reference
Frequency	2.14 GHz	[22]
K	6910 KM^{-1}	[21]
γ	4	[21]
N_0	$10^{-15.82}$	[21]
Subcarrier Frequency	15 kHz	[22]
Macro BSs Max Power	40 W	[22]
RRH Antenna Gain	Omni-directional	[23]
Path Loss Model	128.1 + 37.6 $\log_{10}^{R}$, R in km	[22]
Noise Power Spectral Density	-174 dBm/Hz	[22]
Receiver Noise Figure	9 dB	[22]
Feedback	CQI	[23]
Maximum TX power of BS	40	
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 × 2000 m	
Active UEs	50, 100	

3.3 Experiment 1: Parameter Selection and P-DDPG Benefits

The environment measures the network status at each transmission time interval (TTI). Because of the realistic behaviors 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 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.

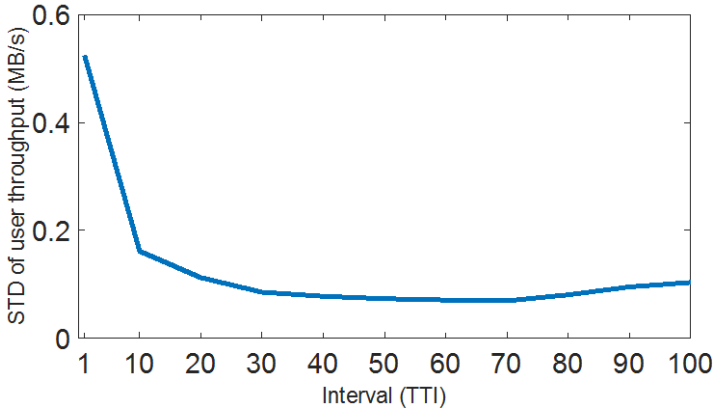


Figure 5 Standard Deviation of Average User Throughput at Different CTI Values

Figure 5 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.

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

Figure 6. Each simulation runs until recording 200 states, therefore, the simulation time on a TTI basis is calculated as CTI × 200. When the simulation time increases, then the run time of environment is increases.

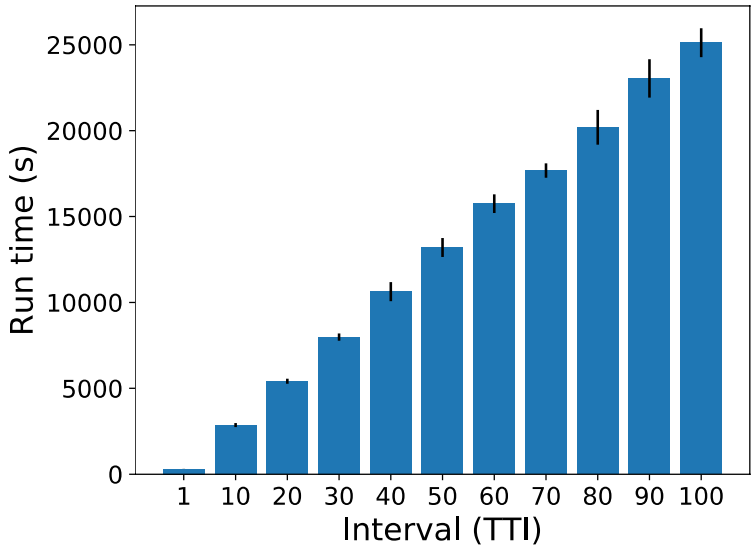


Figure 6 Analysis of Different Communication Time Interval (CTI) Run Time

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.

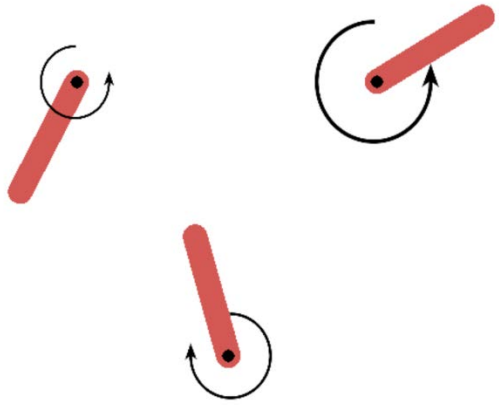


Figure 7 Snapshots of Pendulum Environment

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 [24]. Figure 7 shows the snapshot of pendulum problem. The arrows illustrate the magnitude of the action and the direction of the action. 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 Figure 6, we added a delay to the pendulum environment to observe the effect of the environment run time on learning time.

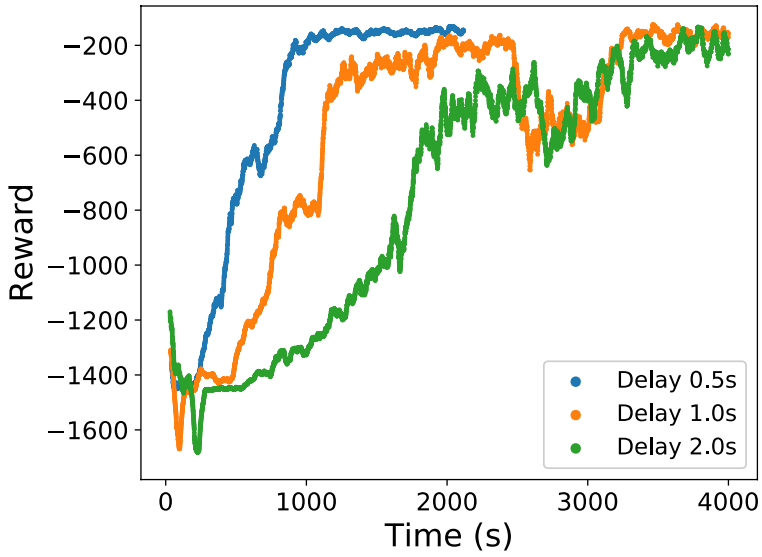
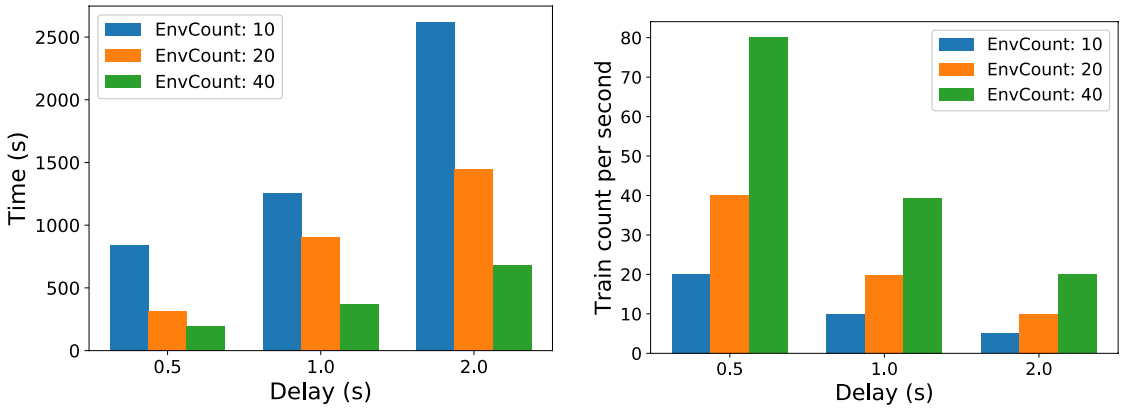


Figure 8 The Effect of Run Time of the Environment on Learning Time

Figure 8 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.



(a) Training Time (b) Training Count per Second

Figure 9 Running Multiple Environments Effect

Running multiple environments is examined in Figure 9. We 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. Figure 9 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. Hence, training count per second is important to improve algorithm. Figure 9 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 (Figure 5), therefore, state-action exchange has 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 (Figure 6). Training is directly related with environment run time (Figure 8). 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 (Figure 9).

3.4 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 to 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.

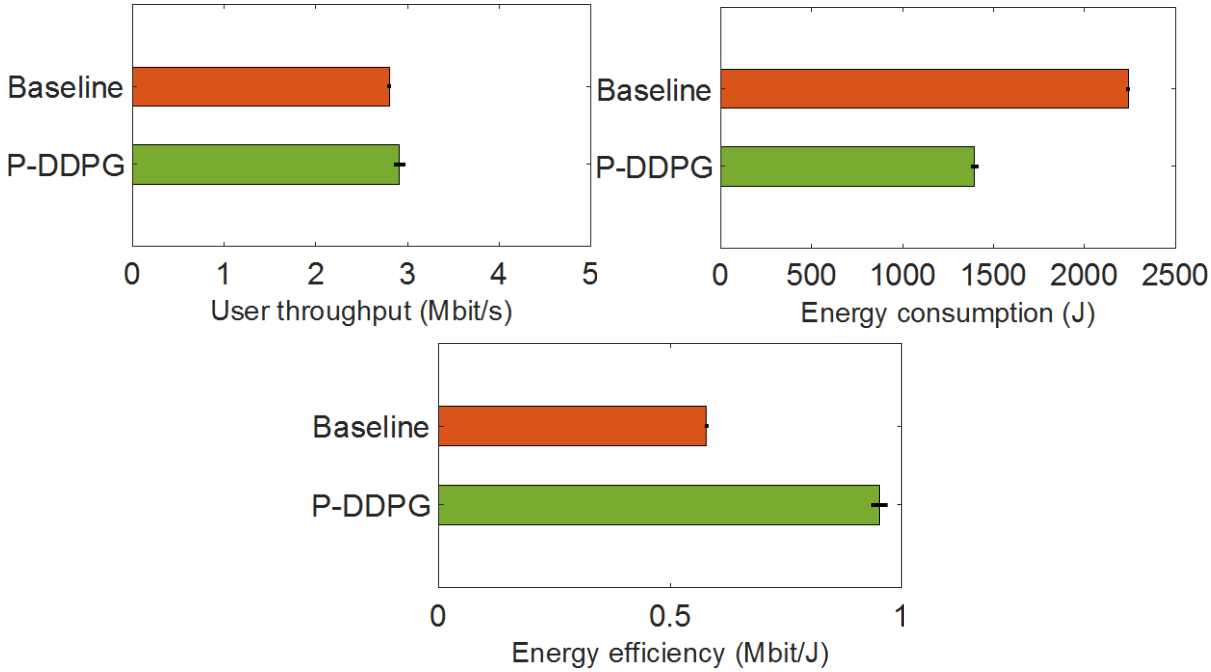


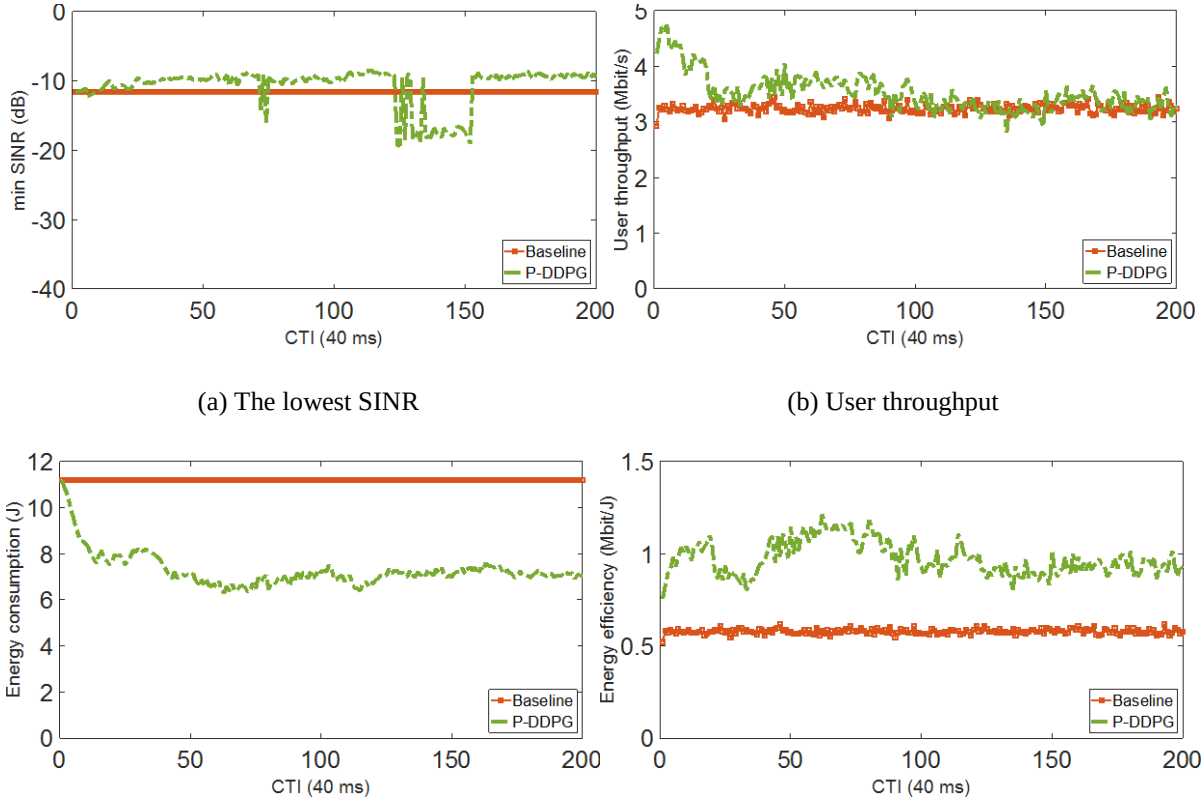
Figure 10 P-DDPG Algorithm Effects on The Environment with 50 UEs

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.

Figure 10 - 12 show P-DDPG effects on the environment with 50 UEs. Figure 10 is obtained by evaluating 40 simulation outputs. As we can see in Figure 10, by applying the P-DDPG model, the amount of energy consumption is reduced, while the overall UEs throughput is preserved, and in some cases, they even enhanced slightly. In sparse scenarios, we achieve up to 67% increase in energy efficiency by using the P-DDPG model in the dynamic environment.

In Figure 11, a single environment's lifetime is presented to observe the realistic environment behaviors 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. Figure 11 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 Figure 11. Because of the user behaviors, there are fluctuations but the P-DDPG model successfully maintains the average throughput of users.



(a) The lowest SINR

(b) User throughput

(c) Energy consumption

(d) Energy efficiency

Figure 11 P-DDPG Algorithm Single Environment's Lifetime Effects on The Environment with 50 UEs

The P-DDPG model maintains QoS parameters while reducing energy consumption. In Figure 11, 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 Figure 11. The P-DDPG model always keeps energy efficiency higher than baseline. Due to users' behaviors, 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 Figure 12. The P-DDPG model plays with BS's power and find an optimum energy level for each of them.

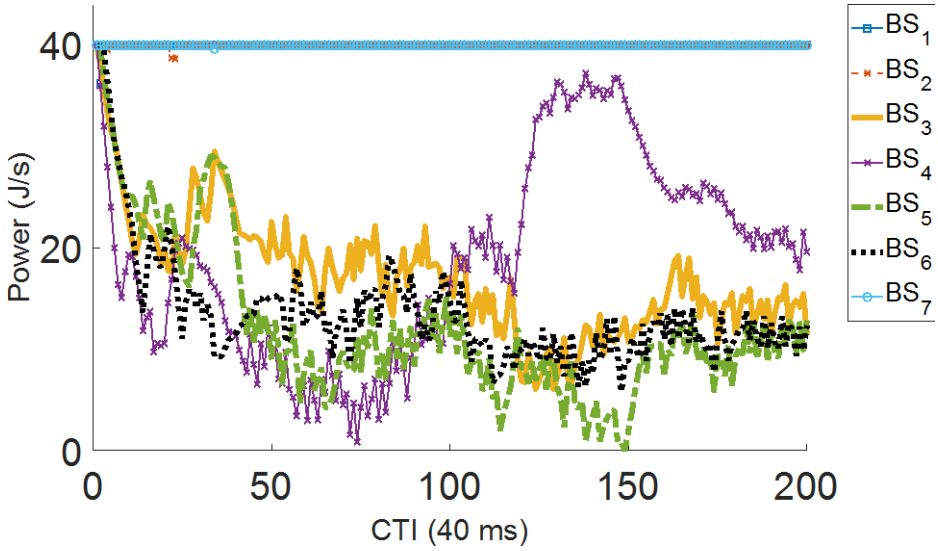


Figure 12 BS Power Changes in a Single Environment with 50 UEs

Another test scenario is constructed with 100 UEs. We illustrate the dense scenario results in Figure 13 and Figure 14. 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 Figure 13. 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.

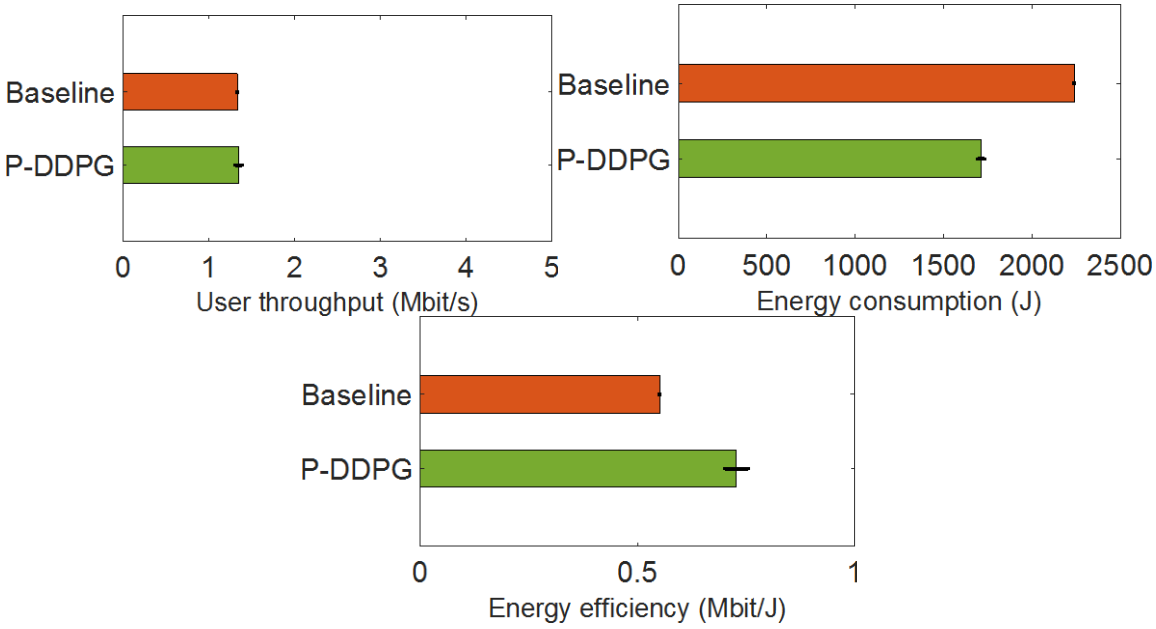
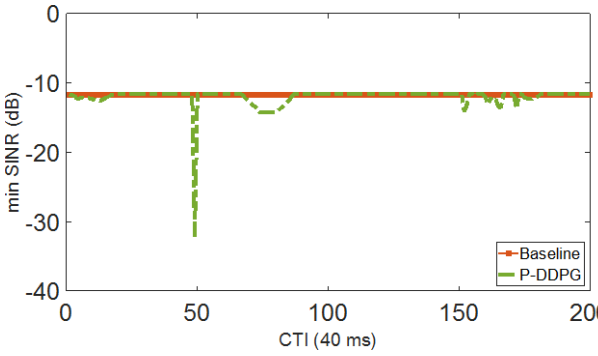


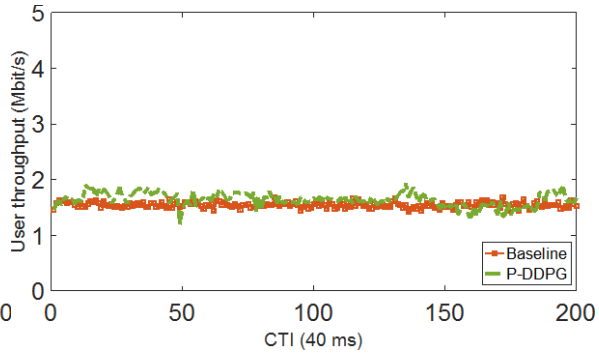
Figure 13 P-DDPG Algorithm Effects on The Environment with 100 UEs

Figure 14 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 as shown in Figure 12. The P-DDPG model acts according to the scenario

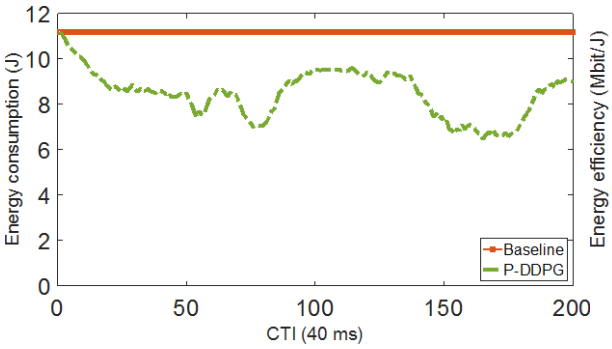
requirements and keeps QoS above the threshold, and reduces energy consumption. By reducing BS's powers, it increases energy efficiency as shown in Figure 14.



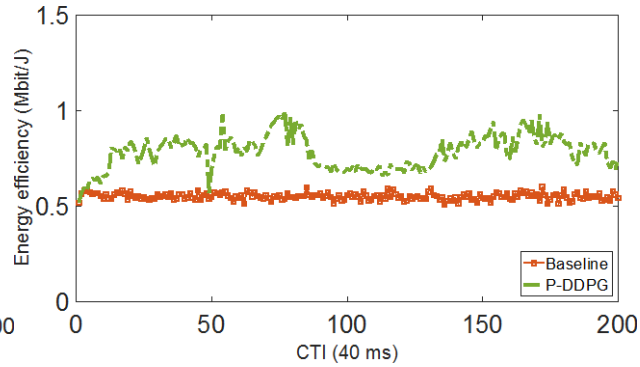
(a) The lowest SINR



(b) User throughput



(c) Energy consumption



(d) Energy efficiency

Figure 14 P-DDPG Algorithm Single Environment's Lifetime Effects on The Environment with 100 UEs

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 behavior, and it can find an optimum energy consumption level for both cases.

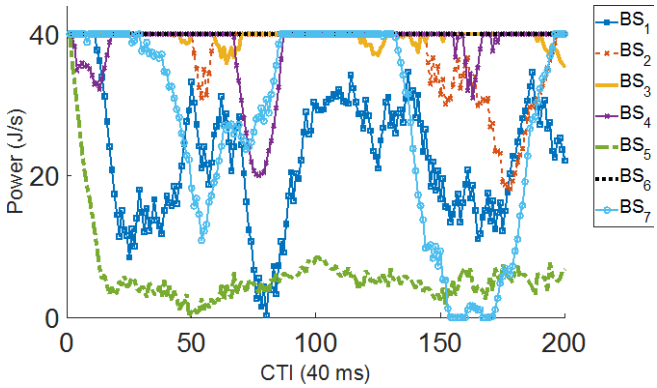


Figure 15 BS Power Changes in a Single Environment with 100 UEs

3.5 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. To test this scenario, we set up a simulation with 50 users and 7 base stations. We set simulation length to 200 CTI. At 25th CTI, we close 3 out of 7 base stations. In Figure 16, 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.

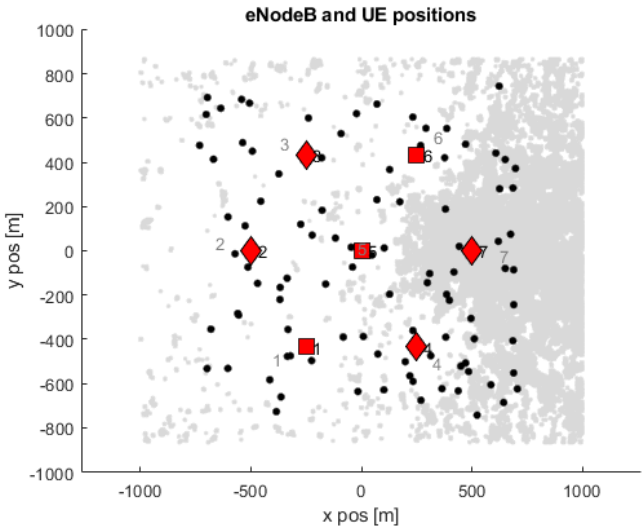


Figure 16 Perturbation Scenario Illustration

The reward function is updated for this scenario. During 25 ≤ CTI ≤ 50, the model is neither rewarded nor punished. We set R = 0 during perturbation. For the perturbation scenario, the reward function is formulated as $R^k = min(R^k_{OS_2}, R^k_{EC})$, where R_{OS_2} refers to overall status of users' throughput reward (Equations 4 – 12) and R_{EC} refers to energy consumption reward (Equation 13).

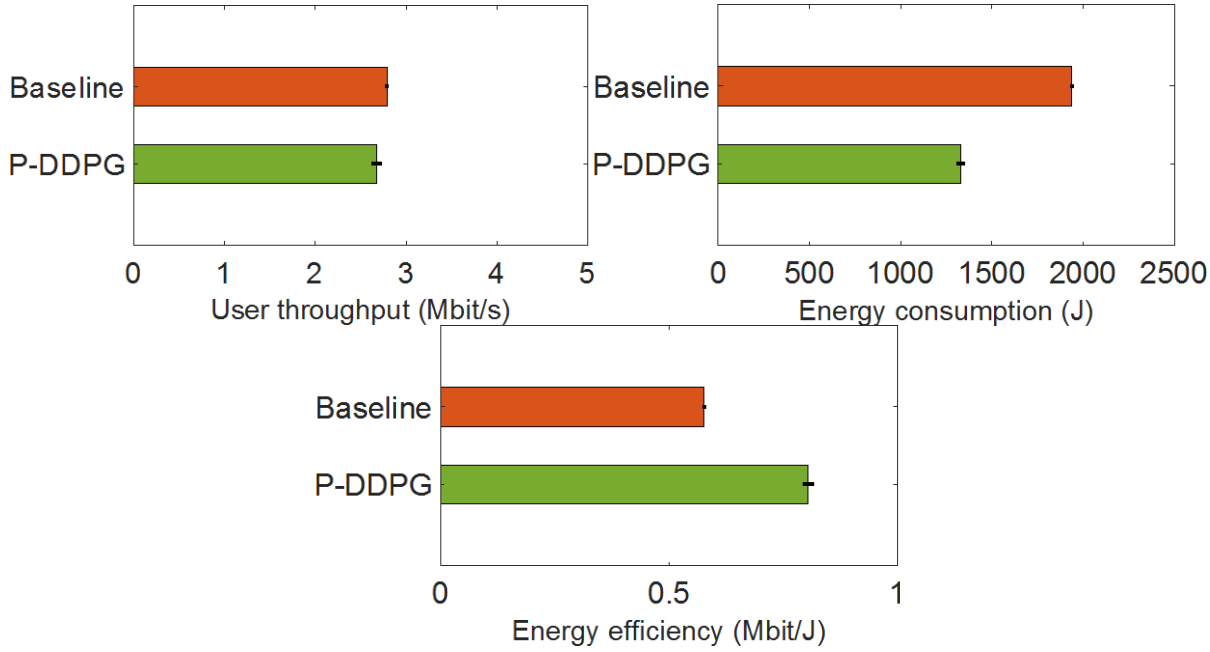
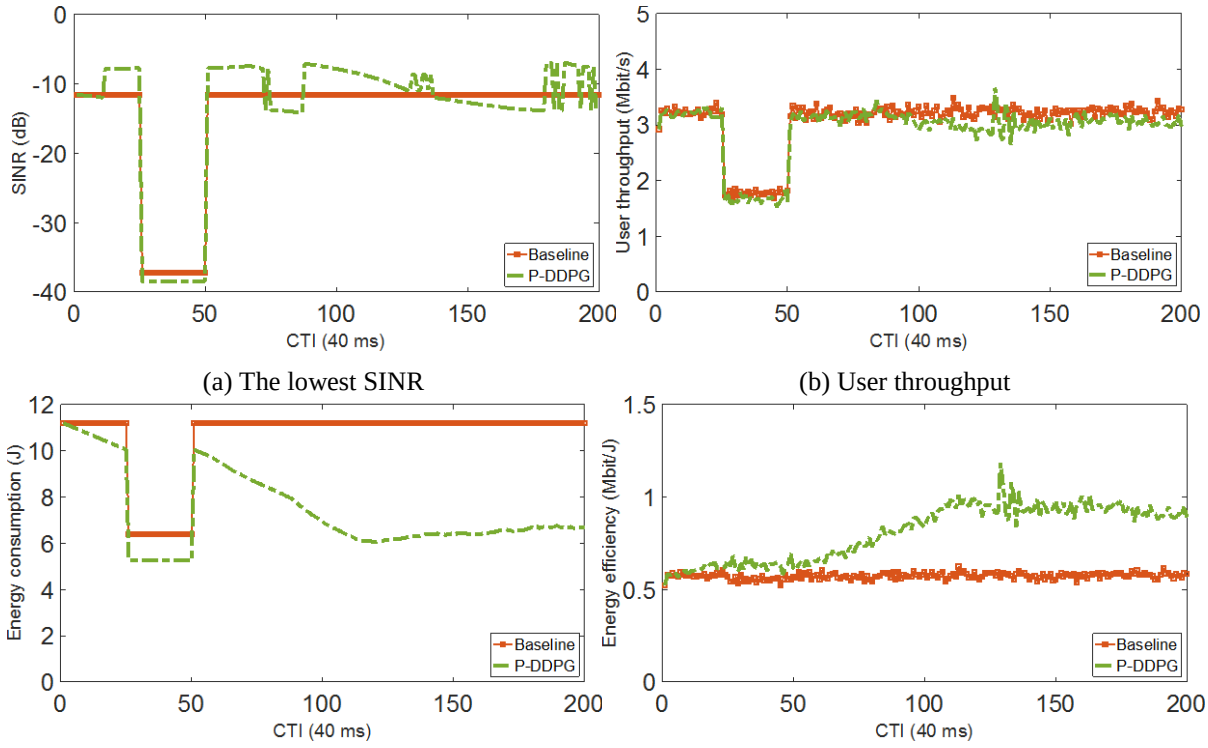


Figure 17 P-DDPG Algorithm Effects on The Environment in Perturbation Scenario

We illustrate the perturbation scenario results in Figure 17 and Figure 18. In perturbation scenario, P-DDPG model can increase energy efficiency while maintaining QoS parameters. We achieve up to 39% increase in energy efficiency.



(a) The lowest SINR

(b) User throughput

(c) Energy consumption

(d) Energy efficiency

Figure 18 P-DDPG Algorithm Single Environment's Lifetime Effects on The Environment in Perturbation Scenario

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 17 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. The average throughput of users almost while reducing energy consumption. 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.

In Figure 18, 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. Figure 18 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. During that time, it increases energy efficiency and preserves the average throughput of users.

Effects of the sudden shutdown and the model actions on base station transmit powers are observable in Figure 19. The base stations 1, 5, and 6 are turned off during $25 \leq CTI \leq 50$. After 50th CTI, the P-DDPG model plays with BS transmit powers to find the optimum energy consumption level for each of them.

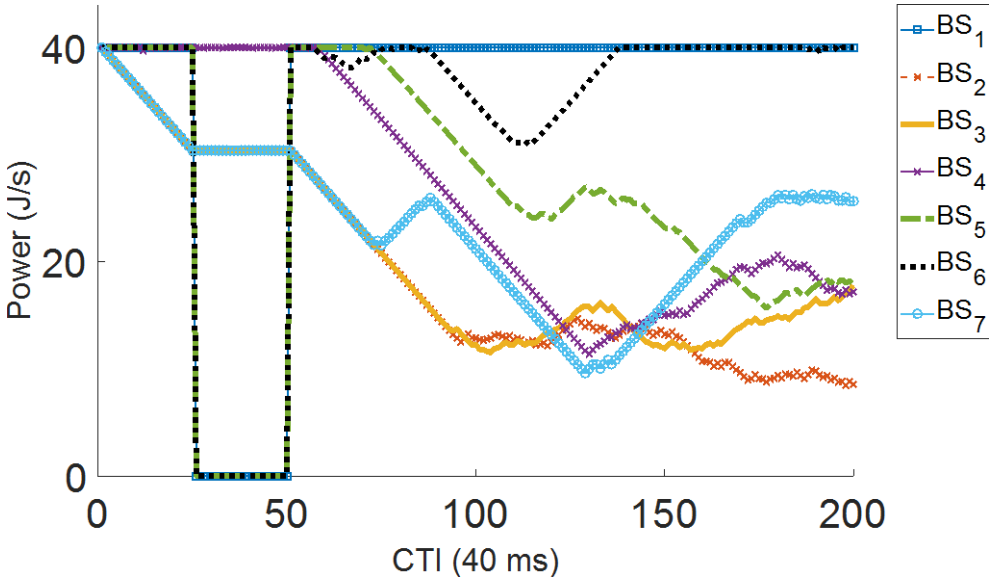


Figure 19 BS Power Changes in a Single Environment in The Perturbation Scenario

To conclude, 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.

4. 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 using analytic models or machine learning.

In the first group, researchers try to find an optimum sleeping strategy by modelling the network analytically. In [25], the authors aim to quantify the trade-off between energy consumption and throughput in a heterogeneous cellular network 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 [7]. 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) [26]. 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 users 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.

On the other side, with the advent of software defined networking, 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 [27], Lu et al. develop a RL model for cellular networks with coordinated multipoint 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 approach for ON-OFF switching in HetNets [28]. 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 by Al Qerm and Shihada [29]. 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 [30]. Their action space is {0, ±1, ±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 Wi-Fi routers, 4G home eNBs and 5G home gNBs [31]. 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.

Moreover, reinforcement learning is used for various control problems in the mobile network. In [32], Wang et al. use federated deep reinforcement learning for decentralized cooperative edge caching. They address to content replacement problem by modeling as a Markov decision process and they use a double deep Q-network (DDQN) to find a solution in the noncontinuous huge space. The time division duplexing control problem is also adressed by using reinforcement learning. In [33], authors work on duplexing framework that let network to dynamically adapt according to the traffic demands. At the same time, intercell interference is mitigated by using their suggested approach. They model the problem by using DDQN.

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.

5. Conclusion

In the next-generation networks, due to the increase in the BS density, environmentalist approach will be essential. Autonomous systems will be more involved in the next generation network. These autonomous systems should be trained in a simulation environment which is close to real life. In this paper, we present a dynamic machine learning based energy saving model. P-DDPG is an extended version of DDPG which has the capability of work with parallel environments. Unlike previous works, we train our model on Vienna Simulator, which is known as realistic and advanced network simulator. 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 sparser 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. A fully autonomous system running on the network can be obtainable by using P-DDPG models that are trained for each cluster.

Acknowledgments

This study is supported by Turkcell under BTK Graduate Scholarship Program.

List of Abbreviations

BS	Base Station
CNRC	Consecutive Negative Reward Check
CTI	Communication Time Interval
DDPG	Deep Deterministic Policy Gradient
DDQN	Double Deep Q Network
DQN	Deep Q Network
HetNet	Heterogeneous Networks
LTE	Long-Term Evolution
MEC	Multi-access Edge Cloud
ML	Machine Learning
NFV	Network Function Virtualization
OS	Overall Status
P-DDPG	Parallel Deep Deterministic Policy Gradient
QoS	Quality of service
RAN	Radio Access Network
RL	Reinforcement Learning
SINR	Signal to Interference Ratio
SN	Software networks
TTI	Transmission Time Interval
UE	User Equipment

References

[1] A. Usman, I. Ozturk, A. Hassan, S. M. Zafar and S. Ullah, "The effect of ICT on energy consumption and economic growth in South Asian economies: an empirical analysis," *Telematics and Informatics*, vol. 58, p. 101537, 2021.

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

[3] E. C. Strinati and L. Herault, "Holistic approach for future energy efficient cellular networks," *Elektrotechnik und Informationstechnik*, vol. 127, p. 314–320, 11 2010.

[4] 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, p. 10–12, 1 2009.

[5] 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, 2015.

[6] 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, p. 3482–3498, 12 2016.

[7] M. Feng, S. Mao and T. Jiang, "Base Station ON-OFF Switching in 5G Wireless Systems: Approaches and Challenges," *IEEE Wireless Communications*, vol. 24, p. 46–54, 8 2017.

[8] B. B. Post and H. van den Berg, "A self-organizing base station sleeping and user association strategy for dense cellular networks," *Wireless Networks*, vol. 27, no. 1, pp. 307-322, 2021.

[9] 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)*, 2016.

[10] 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*, 2016.

[11] 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, 4 2017.

[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*, p. 1–11, 2 2018.

[13] 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, p. 32328–32338, 5 2018.

[14] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine Learning*, vol. 3, p. 9–44, 8 1988.

[15] L. Raju, R. S. Milton, S. Suresh and S. Sankar, "Reinforcement learning in adaptive control of power system generation," *Procedia Computer Science*, vol. 46, p. 202–209, 12 2015.

[16] C. J. C. H. Watkins, "Learning from Delayed Rewards," Cambridge, 1989.

[17] 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.

[18] 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, 6 2014.

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

[20] 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*, Springer International Publishing, 2014, p. 225–239.

[21] 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*, 2013.

[22] 3rd Generation Partnership Project (3GPP), "Evolved universal terrestrial radio access (EUTRA)," in 3rdGenerationPartnership Project (3GPP), 2014.

[23] B. Clerckx, G. Kim and S. Kim, "MU-MIMO with Channel Statistics-Based Codebooks in Spatially Correlated Channels," in *IEEE Global Telecommunications Conference*, 2008.

[24] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang and W. Zaremba, *OpenAI Gym*, 2016.

[25] C. Liu, B. Natarajan and H. Xia, "Small Cell Base Station Sleep Strategies for Energy Efficiency," *IEEE Transactions on Vehicular Technology*, vol. 65, p. 1652–1661, 3 2016.

[26] 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, 5 2016.

[27] 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, 7 2014.

[28] 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)*, 2017.

[29] 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)*, 2017.

[30] E. Ghadimi, F. D. 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)*, 2017.

[31] 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, 6 2019.

[32] X. W. Wang, L. Chenyang, L. Xiuhua, T. Victor and Tarik, "Federated deep reinforcement learning for Internet of Things with decentralized cooperative edge caching," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9441-9455, 2020.

[33] N.-N. N. Dao and C. S. Wonjong, "Deep Reinforcement Learning-Based Hierarchical Time Division Duplexing Control for Dense Wireless and Mobile Networks," *IEEE Transactions on Wireless Communications*, 2021.