Mobile Traffic Classification and Multi-Cell Base Station Control for Energy-Efficient 5G Networks

TIANZHANG CAI

Master's Programme, Machine Learning, 120 credits
Date: March 6, 2023

Supervisor: Özlem Tugfe Demir
Examiner: Cicek Cavdar
School of Electrical Engineering and Computer Science

Abstract

The global energy consumption of mobile networks is rapidly increasing due to the exponential growth of mobile network traffic. The advent of next-generation cellular technologies such as fifth-generation (5G) and beyond promises higher network throughput and lower latency but also demands higher power consumption for its denser base station (BS) deployment and more energy-intensive processors. It is one of the key design pillars of next-generation mobile networks to improve network energy efficiency. In this thesis, we aim to address this problem by applying machine learning to analyze mobile traffic and control the operation of 5G BSs with the goal of reducing network energy consumption while dynamically meeting the network traffic demand. To obtain some preliminary insights into the temporal and spatial patterns of mobile network traffic, we first analyze a real-world network flow dataset collected by a Swedish mobile operator. We find that mobile traffic exhibits a strong periodicity in time and a distinct locality in space. In order to develop a simulation environment to train and evaluate the artificial intelligence (AI)-based BS control algorithm, we apply a clustering algorithm to categorize the network traffic with different latency requirements into five traffic scenarios, e.g. urban, rural, and office. Other necessary network models are also implemented such as energy consumption, massive MIMO (multiple-input multiple-output) channel, user association, etc. in the simulation environment endeavoring to mimic the real-world 5G network.

The main contribution of this thesis is the development of a multi-agent reinforcement learning algorithm to jointly control the operations of 5G BSs such as multi-level sleeping, antenna switching, and user association. The algorithm is designed to minimize the total energy consumption of a multi-cell 5G network while preserving its overall quality of service (QoS). The trained algorithm has shown its ability to adaptively save energy by switching on/off the sleep modes and antennas of BSs according to the varying traffic intensity. Moreover, the multi-agent BS control policy produced by the algorithm has also demonstrated collaborative behaviours such as user offloading that reduces inter-cell interference which degrades the QoS. The evaluation results show that in comparison to the always-on configuration, the proposed algorithm can reduce the total energy consumption of a network by about 50%, double that of today's symbol-level sleeping strategy (25%), while with negligible degradation of QoS. The algorithm is also shown to be robust to variations in the volume and the QoS requirement of the network traffic after being evaluated in different traffic scenarios.

Sammanfattning

Mobilnätens globala energiförbrukning ökar snabbt på grund av den exponentiella tillväxten av mobilnätstrafik. Tillkomsten av nästa generations cellulära teknologier som 5G och framåt lovar högre nätverksgenomströmning och lägre latens, men kräver också högre strömförbrukning för dess tätare basstation (BS)-utbyggnad och mer energikrävande processorer. Det är en av de viktigaste designpelarna i nästa generations mobilnät för att förbättra nätverkets energieffektivitet. I det här examensarbetet syftar vi till att ta itu med detta problem genom att tillämpa maskininlärning för att analysera mobiltrafik och styra driften av 5G BS:er med målet att minska nätverkets energiförbrukning samtidigt som efterfrågan på nätverkstrafik dynamiskt tillgodoses. För att få några preliminära insikter om de tidsmässiga och rumsliga mönstren för mobilnätstrafik, analyserar vi först en verklig nätverksflödesdatauppsättning som samlats in av en svensk mobiloperatör. Vi finner att mobiltrafik uppvisar en stark periodicitet i tid och en distinkt lokalitet i rymden. För att utveckla en simuleringsmiljö för att träna och utvärdera den AI-baserade BS-kontrollalgoritmen, tillämpar vi en klustringsalgoritm för att kategorisera nätverkstrafiken med olika latenskrav i fem trafikscenarier, t.ex. stad, landsbygd och kontor. Andra nödvändiga nätverksmodeller implementeras också, såsom energiförbrukning, massiv MIMO-kanal, användarassociation, etc. i simuleringsmiljön som strävar efter att efterlikna det verkliga 5G-nätverket.

Det huvudsakliga bidraget från denna avhandling är utvecklingen av en multi-agent förstärkningsinlärningsalgoritm för att gemensamt styra driften av 5G BS:er som sömn på flera nivåer, antennväxling och användarförening. Algoritmen är utformad för att minimera den totala energiförbrukningen för ett flercells 5G-nätverk samtidigt som dess övergripande servicekvalitet (QoS) bevaras. Den tränade algoritmen har visat sin förmåga att adaptivt spara energi genom att slå på/stänga av vilolägen och antenner för BS:er enligt den varierande trafikintensiteten. Dessutom har multi-agent BS-kontrollpolicyn som produceras av algoritmen också visat samverkansbeteenden såsom användaravlastning som minskar inter-cellinterferens som försämrar QoS. Utvärderingsresultaten visar att den föreslagna algoritmen kan minska den totala energiförbrukningen för ett nätverk med cirka 50%, dubbelt så stor som dagens sovstrategi på symbolnivå (25%), jämfört med konfigurationen som alltid är på. försämring av QoS. Algoritmen har också visat sig vara robust mot variationer i volymen och QoS-kravet för nätverkstrafiken efter att ha utvärderats i olika trafikscenarier.

Acknowledgments

I would like to express my gratitude to my examiner Çiçek Çavdar and my supervisor Özlem Tuğfe Demir for providing me with invaluable guidance, support, and encouragement throughout my research. Their expertise, insightful feedback, and unwavering patience were instrumental in shaping this thesis. I am also grateful to the faculty members of KTH Wireless for their academic instruction and intellectual stimulation.

I would like to thank my family and friends for their continuous love and encouragement. Their unwavering support has been a constant source of motivation for me.

I would also like to extend my appreciation to Martina Lindman and Anders Björk from Tele2 for providing me with the mobile network flow dataset extensively used in this thesis project.

This research would not have been possible without the contributions of these individuals and organizations.

Stockholm, March 2023
Tianzhang Cai

Contents

List of Figures

List of Tables

List of acronyms and abbreviations

4G Fourth-Generation

5G Fifth-Generation

ASM Advanced Sleep Mode

BS Base Station

DTX Discontinuous Transmission

GHG Green House Gas

LTE Long-Term Evolution

ML Machine Learning

NR New Radio

OFDM Orthogonal Frequency Division Multiplexing

QoE Quality of Experience

QoS Quality of Service

RB Resource Block

RF Radio Frequency

RL Reinforcement Learning

SCS Sub-Carrier Spacing

SINR Signal-to-Interference-Plus-Noise Ratio

SM Sleep Mode

SS Synchronization Signal

Chapter 1

Introduction

1.1 Background

Recent years have seen exponential growth in global mobile network traffic, estimated to double every two years, reaching 115 EB (115×10^{18} bytes) per month by the end of 2022, as shown in Figure 1.1 [1]. The explosion of traffic has led to a rapid increase in the energy consumption of global cellular networks, reaching around 293 TWh per year in 2021, accounting for about 1.16% of the global electricity demand [6, 7]. In response to the challenge of global warming and in line with the United Nations Sustainable Development Goals (SDGs), a majority of GSMA (Global System for Mobile Communications) members, accounting for 63% of the global mobile industry by revenue, have commited to science-based targets (SBTs), aiming for net-zero greenhouse gas emissions by 2050 to limit global warming to well-below 2°C above pre-industrial levels [8, 9].

Mobile big data analytics and artificial intelligence (AI) are emerging as powerful forces transforming business and society, and the potential of these technologies to unlock life-changing benefits is only beginning to be seen. When grounded in ethical principles that protect privacy, these solutions can truly change the world for the better. PwC estimates that, over the next 5 years, 150 million people could be positively impacted by mobile big data and AI solutions, equating to 3% of the world population [10].

1.2 Problem

With the proliferating cellular connectivity, it is becoming increasingly important to attain a comprehensive understanding of the temporal and geographical patterns of the cellular traffic. Internet service providers (ISPs) can exploit the traffic patterns in time and space and deploy dedicated network management or pricing policies in order to improve quality of service (QoS) and reduce capital or operational expenses, including energy consumption. Pattern recognition of the cellular traffic can also inform the government of land usage, human activities, and so on. Analyses of real-world datasets have revealed that cellular traffic possesses a distinctive temporal periodicity and geographical locality. Machine learning (ML), given the well-organized data collection provided by ISPs and the distinguishable data patterns, thereafter, can be effectively applied to classify and predict cellular traffic patterns. The first part of this thesis aims to use ML

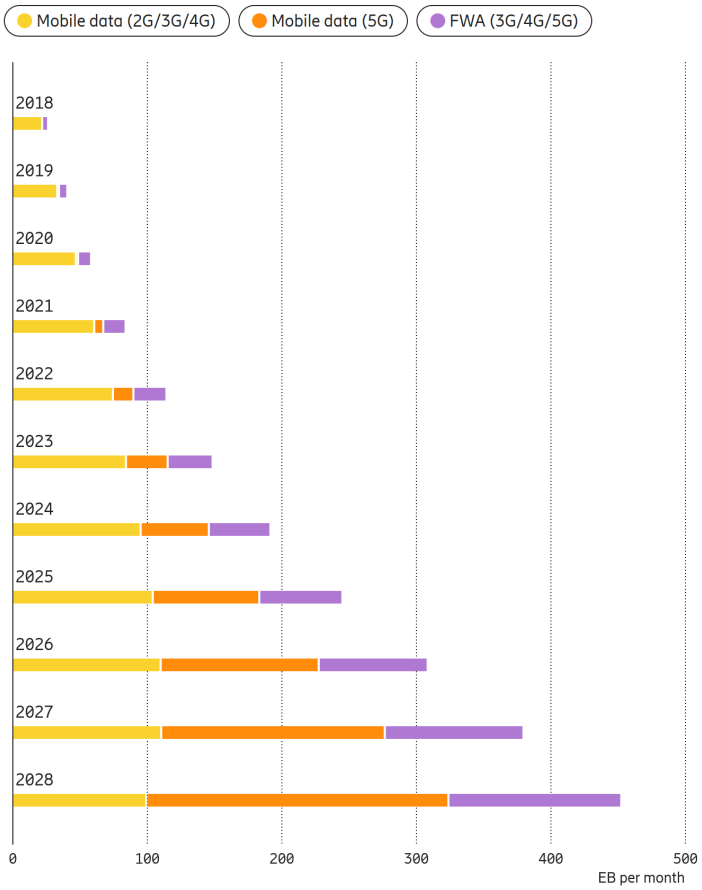


Figure 1.1: Global mobile network data traffic (FWA represents Fixed Wireless Access) [1].

to classify the cellular traffic patterns, to investigate their temporal and geographical distribution for each class of cells, and to extract a traffic profile in order to generate synthetic mobile traffic in the network simulation.

The advent of fifth-generation (5G) cellular networks is set to bring about a range of enhancements, including lower latency, higher data rates, and wider connectivity, compared to previous generations. The deployment of a denser network of base stations (BSs) is expected to accommodate these improvements. However, this densification will result in a considerable increase in energy consumption, as BSs are the most energy-intensive components of a wireless network and are responsible for approximately 80% of a network's energy consumption [11]. Furthermore, the higher bit-rate demands of 5G will require the use of more power-hungry processors and radio frequency hardware at BSs, leading to an estimated two to three times higher energy consumption compared to the fourth-generation (4G) BSs [12]. This increase in energy consumption is not sustainable in the long term, making the deployment of green networks essential. A green network is one that takes sustainability into account, ensuring that the network equipment and architecture are energy efficient across varying network conditions. While 5G presents challenges in terms of energy consumption, it also offers opportunities to implement new methods for energy conservation.

The traffic of a mobile network is highly dynamic and the latency a user can tolerant is usually very tight. Furthermore, the control of such mobile network systems as 5G and beyond possesses a high complexity. A multi-cell network, in addition, requires the collaboration of the BSs to save energy jointly, which may lead to a different multi-agent policy than the optimal one for each agent considered separately. Therefore, the required BS control policy must be dynamic, adaptive, and cooperative in nature. Nowadays, ML and its sub-fields like deep learning and reinforcement learning (RL) are seen as universal and powerful methods to solve a wide range of complicated problems. The performance of these methods largely depends on the quantity and quality of the data used to train the ML models. The availability of data processing devices and techniques in mobile networks like deep packet inspection (DPI) has enabled the application of ML in mobile network management and design, including radio resource allocation, user association, and sleep mode management. These ML-based mechanisms allow wireless networks to be predictive and proactive with regard to a mercurial mobile traffic environment so as to improve energy efficiency, as well as the QoS.

1.3 Purpose

The purpose of this thesis is to reduce the energy consumption of mobile networks by leveraging the flexibility introduced by 5G technology, such as extended sleep periods and adjustable resource block (RB) sizes, and by designing an autonomous advanced sleep mode (ASM) management system. This will be achieved through the following subgoals:

1. Classifying the cells based on their network traffic in different delay categories to investigate the temporal and geographic patterns of cellular traffic in different scenarios.

2. Implementing a multi-cell 5G network simulation environment with realistic modeling of massive MIMO channels, power consumption, etc., for traffic analysis and RL for BS control.

3. Designing an RL algorithm for multi-agent BS control of antenna switching, ASM management, and user association in the simulated network, in order to cooperatively save network power consumption without compromising QoS.

The thesis will assess the potential energy savings of these design approaches and compare them to two benchmarks: a baseline control policy without any BS sleeping and a simple policy that BSs automatically switch to the shallowest sleep mode.

This work aims to contribute to the pursuit of SDGs by reducing the carbon footprint of the information and communication technology (ICT) sector. BS operation accounts for around 80% of the total energy consumption of a mobile network. Improving the energy efficiency of BSs is therefore a key way to improve the sustainability of mobile networks. The energy-saving potential of BS control is significant, as mobile traffic varies greatly in time and space, and a BS can deactivate some of its components during periods of low traffic to reduce its power consumption without degrading performance.

This thesis prioritizes ethical considerations, crediting and citing any ideas or work used by other researchers in the references section. Any ethical issues arising from the implementation of the work are also avoided.

1.4 Research Questions

1. How to make reliable cellular traffic categorization that is aware of network services with different delay requirements, based on the collected dataset of mobile network traffic flows?

2. How to realistically model the 5G network including massive MIMO channel, power consumption, network service, BS sleeping, user equipments, packet drop, etc.?

3. How to generate synthetic mobile traffic based on the cellular traffic categorization to evaluate the performance of the energy-saving BS control policies?

4. How multi-agent RL (MARL) can be utilized in adaptively controlling the number of antennas, user association, and ASMs of multiple BSs that act cooperatively in order to optimize energy efficiency without compromising network QoS, according to the network traffic? Which MARL algorithm is best suitable for this task?

5. How to quantitatively measure the performance of different BS control policies and the advantage of the MARL-based policy over benchmark policies?

1.5 Research Methodology

In this research, we apply a sequence of procedures like data cleaning, temporal aggregation, and discrete Fourier transform (DFT) to produce weekly profiles that characterize the cellular traffic. The data was collected by a Swedish mobile operator, containing sampled records with detailed network flow information, including time duration, traffic volume, application name, etc. Due to ethical considerations and to ensure trust, accountability, mutual respect,

and fairness, sensitive information such as the locations of most BSs are not exposed and all users were anonymized. Machine learning approaches are applied to classify the cells and the temporal and spatial patterns of each category of cellular traffic are analyzed.

In order to tackle the problem of energy-efficient BS operation in multi-cell 5G networks, we develop AI algorithms to achieve dynamic and cooperative multi-agent BS control. We first develop a 5G network simulation environment with synthetic traffic following the patterns of previously obtained cellular traffic categories. Then a MARL agent is trained in this environment to produce a desirable BS control policy. The algorithms were evaluated through in the simulation and compared to multiple baselines, including a vanilla system without any energy saving mechanisms, and a system with a simple policy that automatically puts idle BSs to the shallowest sleep mode.

1.6 Delimitations

The focus of this thesis is on reducing the total energy consumption of multiple BSs in a multi-cell network.

The scope of this thesis is limited to the development and evaluation of a MARL algorithm for controlled BS sleeping to save energy. Specifically, the thesis proposes a novel approach to address the problem of balancing energy consumption and QoS in wireless networks using a MARL approach. The thesis presents a mathematical model for representing the power consumption of a BS in both operational and idle modes, and proposes an AI-based algorithm to solve the optimization problem. The thesis uses simulations to evaluate the performance of the proposed algorithm and its variations in terms of energy saving and QoS improvement.

However, the thesis has several limitations. Firstly, the proposed model for power consumption may not completely reflect the behavior of real-world BSs, and further work is needed to develop a more accurate model. Secondly, the proposed algorithm may not be scalable to larger networks, and further research is needed to investigate the algorithm's performance under different network topologies and traffic conditions. Thirdly, the thesis assumes that user QoS requirements are fixed and does not consider the dynamic nature of user demands in practice. Future work could explore dynamic user demands and adapt the algorithm to meet these demands.

In addition, the thesis proposes several directions for future work, including improving the network model and algorithm, refining the treatment of different QoS requirements, comparing the proposed approach with other RL algorithms, exploring the end-to-end optimization of the network, developing risk-aware RL algorithms, and dynamic functional split optimization based on user delay requirements.

Finally, the thesis does not address other important issues related to wireless communication systems, such as security, risk management, or network planning. Further work is needed to integrate the proposed algorithm into practical wireless communication systems and to evaluate its performance under different real-world scenarios.

1.7 Structure of the Thesis

The remainder of this thesis is organized as follows:

- Chapter 2 conducts data analysis of a cellular traffic dataset and ML-based classification based on the weekly traffic profiles of cells in different service categories. The temporal and geographic distributions of the traffic of each class of cells are then investigated to provide insights of the mobile network traffic.
- Chapter 3 presents the application of a multi-agent RL (MARL) algorithm, MAPPO, in the energy-efficient management of a multi-cell 5G network. The model of a simplified 5G network is given as the simulation environment for the MARL agent training. Some preliminary knowledge of RL and MARL is then provided to the readers, and the MAPPO algorithm and its implementation for this multi-BS control problem are explained. Finally, the analysis and results of the proposed MARL method, including training curves, analysis of the trained policy, the test performance in simulation, and comparisons with other benchmark policies.
- Chapter 4 concludes the research studies conducted in this thesis and points out its limitations and potential extensions in future work.

Chapter 2

Cellular Traffic Analysis and Categorization

In the first part of this thesis, we aim to analyze real-world mobile network traffic and categorize its temporal patterns using unsupervised machine learning. Different network traffic scenarios are consequently developed representing typical traffic patterns in different areas such as rural areas, urban areas, office areas, and so on. The different delay limits of various network services are considered in the analysis.

2.1 Dataset and Visualization

In this section, we describe the dataset we used, introduce service QoS specifications, and perform statistical analysis and spectral filtering for preprocessing.

2.1.1 Dataset description

The cellular traffic dataset is collected by one of the largest Swedish mobile operators and contains sampled and anonymized mobile network flows in a period of two months – from November 25th 2021 to January 25th 2022. In the dataset, network flows were sampled (by a rate of about 1%) and inspected using the Deep Packet Inspection (DPI) technique, which can record flow information, including its belonging site ID, start time, duration, identified application, downlink and uplink data size, etc. Some of the example flow records are shown in Table 2.1. There might be multiple cells, which represent the sectors of a BS, belonging to a single BS, which is reported as a "site" in the dataset. In this paper, we do the clustering based on the sites (BSs).

The "app name" and "app category" correspond to the application and the service category that it belongs, respectively. The DPI engine was able to identify the traffic flows in a time granularity of 2 seconds and differentiate between about 300 applications, such as HTTPS, Facebook, WhatsApp, Youtube, etc. This allows us to classify the sites based on their traffic patterns in different categories according to the delay requirement of the provided services. During preprocessing, we ignored several application categories including "Network Infrastructure", "Software Update", "Unidentified", and "Unknown", accounting to less than 1% of the total traffic.

site ID	start timestamp	duration (ms)	app name	app category	downlink bytes	uplink bytes
28705xxx	2021-11-xx xx:58:52	1 586	HTTPS	Web	4 168	2 156
28163xxx	2021-11-xx xx:59:14	705	WeChat	Real-Time Communication	10 523	896
80590xxx	2021-11-xx xx:59:21	52	Apple Game Center	Gaming	6 537	685
70945xxx	2021-11-xx xx:59:31	2 566	Facebook	Social Networking	16 339	10 815
31077xxx	2021-11-xx xx:59:52	4 437	SVTplay	Multimedia Streaming	1 558 264	18 798

Table 2.1: Sample flows from the dataset.

Delay category	Packet delay budget	Service category	Example applications
Delay stringent	50 ms	Real-time gaming	Steam, League Of Legends, Minecraft
		Real-time communication	Apple FaceTime, Skype, WeChat, WhatsApp
Delay sensitive	150 ms	Multimedia streaming	Spotify, Netflix, YouTube, SVTplay, HBO
		Social networking	Facebook, Instagram, Snapchat, TikTok
Delay tolerant	300 ms	Web applications	Google, Amazon, Paypal, Uber, HTTPS
		Business	Microsoft Office 365, Salesforce, LDAP
		Mail	Gmail, Microsoft Exchange, IMAP, POP3
		Database	Microsoft SQL Server, MySQL, PostgreSQL
		File hosting	Apple iCloud, Dropbox, Microsoft OneDrive
		Peer to peer	Bitcoin, BitTorrent, DHT, Xunlei
		Tunnel and remote access	IPSec, SSH, Telnet, OpenVPN, TLS

Table 2.2: Delay categorization in line with the 3GPP QoS specifications.

After excluding the traffic in these categories, the dataset consists of 1.0×10^8 mobile network flows involving 9861 sites and a total traffic volume of 321.2 TB.

2.1.2 Service QoS specifications

The 3rd Generation Partnership Project (3GPP) is a consortium with a number of standards organizations that develop protocols for mobile telecommunications. According to a 3GPP specification document TS 23.501 [13], there is a certain packet delay budget for each category of network service to specify its QoS requirement, and based on that, we can divide the delay budgets into three categories: *delay stringent*, *delay sensitive*, and *delay tolerant*. These three categories of services that we consider include almost all mobile network traffic generated in daily life, and we specify their delay budgets and delay categories in Table 2.2.

2.1.3 Statistical analysis

After examining the network flows in our dataset, 21.10 TB (6.57%) of the total traffic is delay stringent, 159.25 TB (49.58%) is delay sensitive, and 140.84 TB (43.85%) is delay tolerant. The statistics of the network traffic of BSs in each delay category is demonstrated by box plots in Figure 2.1. We can see that delay-stringent traffic is much less than dalay-sensitive traffic or delay-tolerant traffic. The traffic distribution is skewed to the higher end and all outliers are high-traffic ones as well. The data sizes of flows, as shown in the histogram in Figure 2.2, is a unimodal distribution close to log-normal. We plot the histogram of total flow counts of sites in Figure 2.3. We can see that the distribution of the flow counts is bimodal - we consider it as a mixture of a "low-traffic" unimodal distribution and a "high-traffic" unimodal distribution.

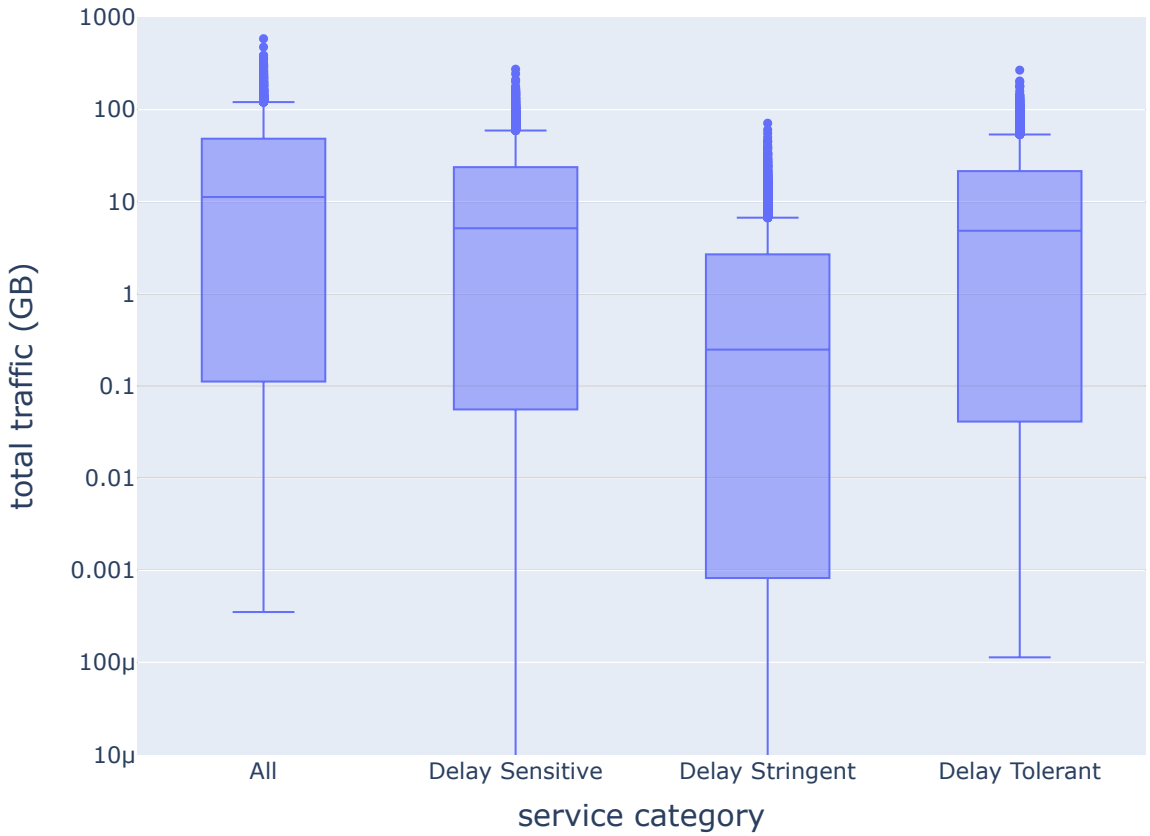


Figure 2.1: Box plot of the total traffic per each cite over two months in each delay category. The vertical axis is in log scale.

2.1.4 Geographical distribution

There are 287 sites in our dataset with available geographical information as well as other information, including site name, coverage radius, etc. We can see that the spatial and temporal patterns of cellular traffic are correlated from Figure 2.5 and 2.6. Figure 2.5 shows the geographical distributions of cellular traffic in a medium-sized city in Sweden on weekdays and weekends. We can see traffic is more concentrated in city centers on weekdays, especially in the city, and it becomes more diverse on weekends. Figure 2.6 shows the geographical distributions of the traffic at four different times of the day.

2.1.5 Data preprocessing

Missing values

For flow records with missing values necessary in our traffic analysis, for example site ID, timestamp, or uplink/downlink bytes, we remove them from the dataset before the subsequent preprocessing steps.

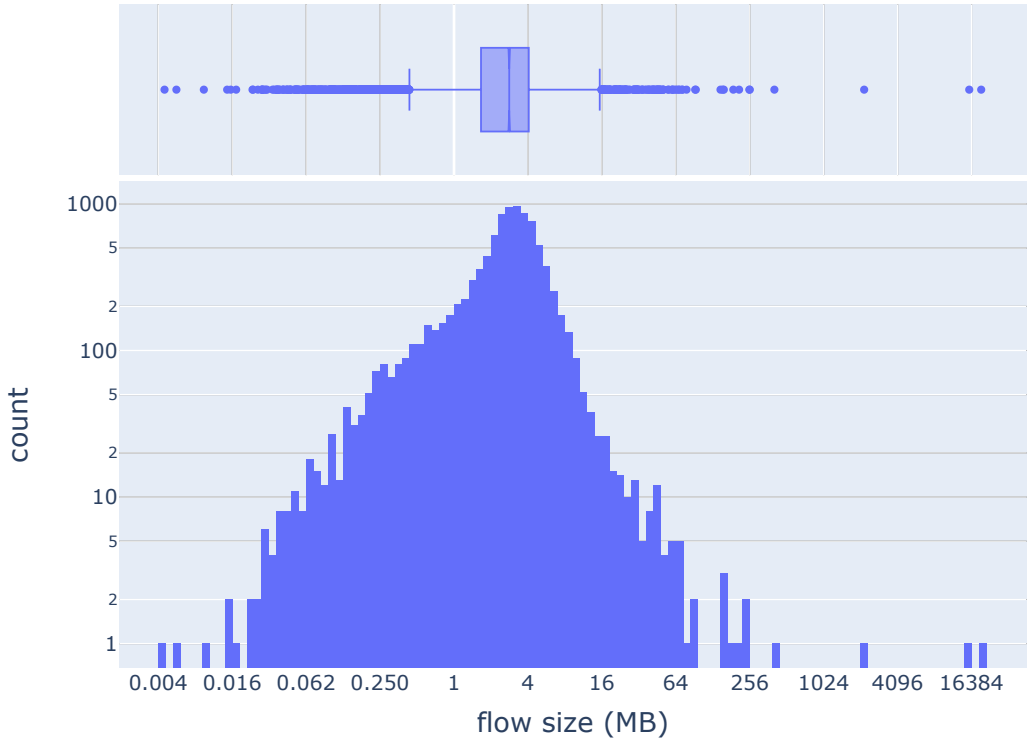


Figure 2.2: Histogram of data sizes of network flows.

Filtration of low-traffic sites

In order to make sure that the number of DPI-sampled flows of a site is sufficiently large to reveal a temporal pattern with statistical significance, we filter out the sites with a low flow count. As we have considered the sites as a mixture of a "low-traffic" group and a "high-traffic" group, we want to find a percentile of the sites where we can divide the bimodal distribution of flow counts into two unimodal distributions corresponding to these two groups. Therefore we plot the site flow counts versus the percentiles from 2% to 98% as the blue line in Figure 2.4. We take the derivative of this curve as the red line in the same figure, in order to find the percentile corresponding to the bottom of the valley between the two unimodal distributions in Figure 2.3. Since there are the least sites in the bins at the bottom of the valley, the flow counts per percentile should change the fastest, which means the bottom of the valley in the histogram corresponds to the local maximum of the red curve. As shown in Figure 2.4, this point locates at the 40th percentile and corresponds to a count of 650 flows, on average 10.5 sampled flows per day. We consider sites with a flow count below this threshold as low-traffic sites and filter them out. After that, we plot the histogram of flow counts again for the remaining sites in Fig. 2.7. We can see that both the flow counts and the total traffic volumes now follow a unimodal distribution similar to log-normal (the horizontal axis is in the logarithmic scale). As a result, 40% of the sites are filtered out as low-traffic and 5902 sites remain in the dataset. The remaining total traffic, however, has 99.6% of the original volume, and the total number of flows is also 99.6% of its original amount.

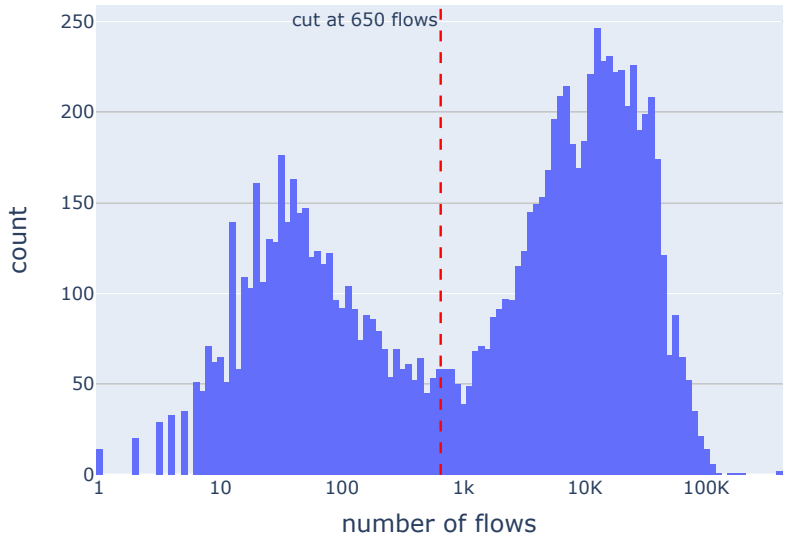


Figure 2.3: Histograms of total flow counts of sites in logarithmic scale. The red dash line divides the sites into "low-traffic" and "high-traffic" as explained in Figure 2.4.

Temporal traffic profiling

To analyze the temporal patterns of the cellular traffic with different QoS requirements, for each site and each delay category, we aggregate the uplink and downlink traffic of network flows into time slots of half an hour. Formally speaking, for a site c with sampled flows f_i^c, $i = 1, \ldots, N_c$, where N_c is the number of flows in site c, and $v(f), t_0(f), t_1(f), q(f)$ represent the total traffic data, the start time, the end time, and the delay category of flow f, we define the traffic profile of c in delay category k as

$$x(c,k) = \left(x_1, \ldots, x_{T/D}\right), \tag{2.1}$$

with

$$\begin{aligned} x_i = & \sum_{j=1, q\left(f_j^c\right)=k}^{N_c} \frac{v\left(f_j^c\right)}{t_1\left(f_j^c\right) - t_0\left(f_j^c\right)} \\ & \times \left| \left[t_0\left(f_j^c\right) - T_0, t_1\left(f_j^c\right) - T_0\right] \cap \left[(i-1)D, iD\right] \right|, \end{aligned} \tag{2.2}$$

where D is the length of the time slot, i.e. 30 minutes, T is the overall time span of flows in the dataset, i.e. 62 days or 1488 hours, T_0 is the start time of the first day when traffic was sampled in the dataset. For a flow f_j^c, we assume a uniform data rate throughout its duration, calculated as $\frac{v\left(f_j^c\right)}{t_1\left(f_j^c\right) - t_0\left(f_j^c\right)}$. The term $|[\ldots] \cap [\ldots]|$ gives the time duration this flow occurred in the i-th time slot.

After that, in order to reduce the dimensionality of the traffic profiles as well as their variance (by increasing the sample size of aggregated flows in each time slot), we further take the average of the traffic in time slots in monthly (31 days), weekly, and daily periods. For a period of H

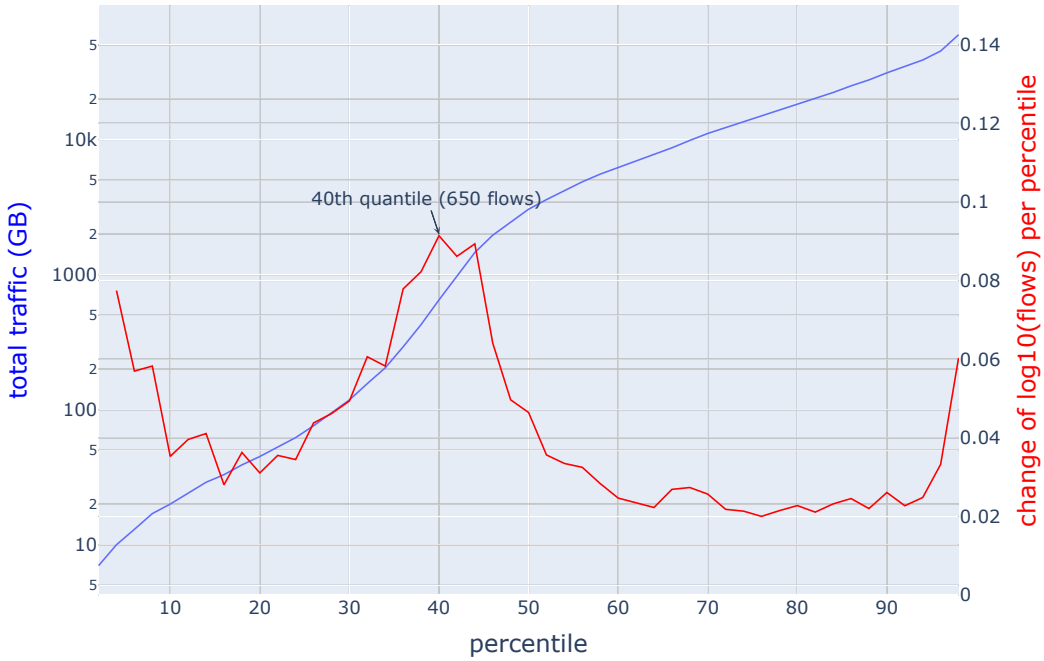


Figure 2.4: The blue line represents percentiles of total flow counts of sites and the red line represents its "derivative" or rate of change. We can see that the peak of the red line locates at the local minimum of the histogram in Figure 2.3. We consider this point as the dividing point between the mixture of two unimodal distributions of flow counts.

days, the traffic profile of c in delay category k is

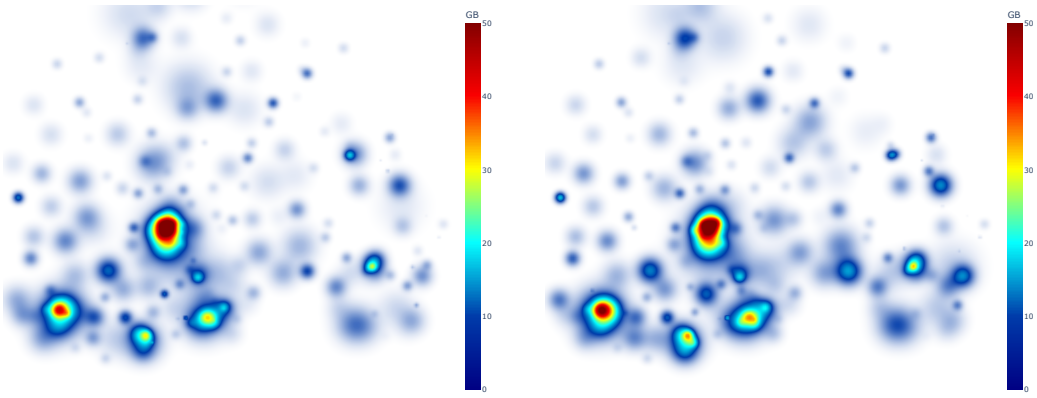
$$x^H(c,k) = \left(\bar{x}_1^H, \ldots, \bar{x}_{H/D}^H\right), \tag{2.3}$$

with

$$\bar{x}_i^H = \frac{\sum_{j=1, j+d^H \equiv i-1 \,(\mathrm{mod}\, H/D)}^{T/D} x(c,k)_j}{\sum_{j=1, j+d^H \equiv i-1 \,(\mathrm{mod}\, H/D)}^{T/D} 1}, \tag{2.4}$$

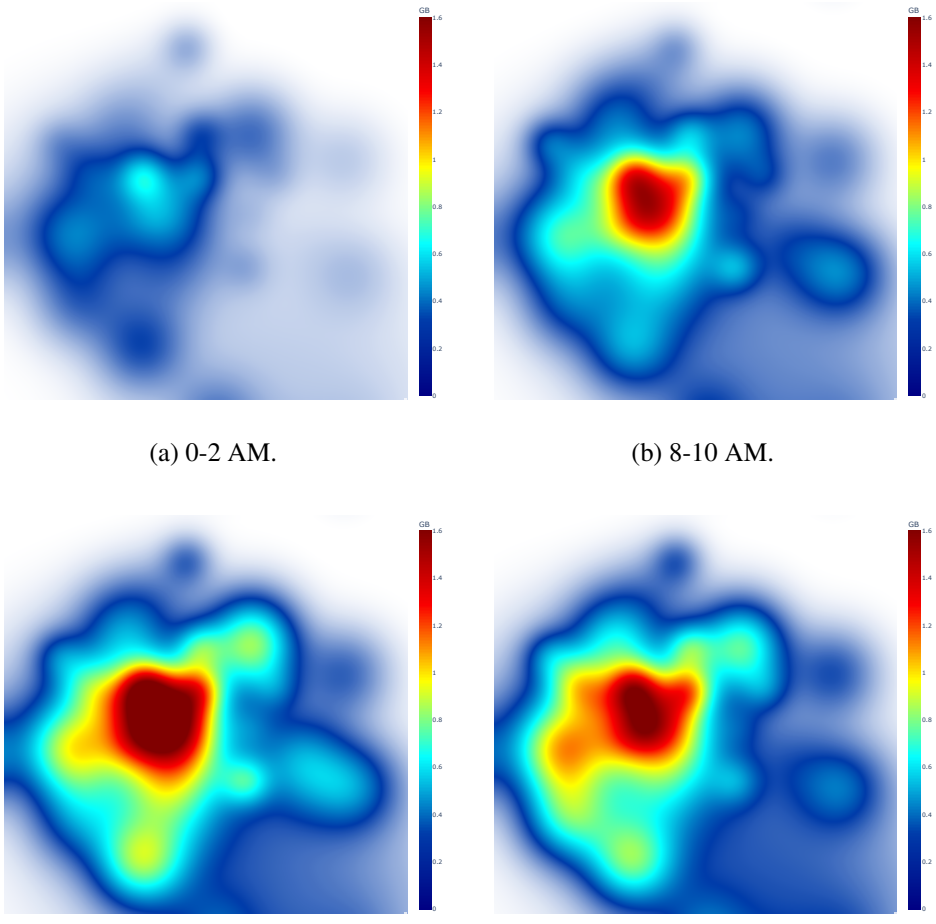
where d^H is an offset constant to permute the traffic profile cyclically such that $x^H(c,k)$ starts on day 1st of a month for $H = 31$ days and starts on Monday for $H = 7$ days.

Figure 2.8 shows the average traffic patterns of all sites in these three time scales. We can see that the monthly network traffic has less variance but more irregularity in comparison to those in weekly and daily periods. Although the traffic pattern in the weekly period seems quite repetitive every day, there are still varieties, the most distinctive among which is the difference between traffic patterns on weekdays and weekends. Therefore, to build a representative profile of the network traffic of a site, we choose to use its weekly traffic profiles in the aforementioned three delay categories. The resulting traffic profile of a site can be illustrated in Figure 2.9. For each delay category, its traffic profile is a $7 \times 48 = 336$ dimensional array. Consequently, the dataset is transformed into a $5902 \times 3 \times 336$ dimensional array.



(a) Weekday. (b) Weekend.

Figure 2.5: Geographical distribution of cellular traffic on weekdays and weekend in the region. The color scale indicates the total traffic of a day averaged per day.



(a) 0-2 AM. (b) 8-10 AM.

(c) 14-16 PM. (d) 18-20 PM.

Figure 2.6: Geographical distribution of cellular traffic in the city at different times of a day. The color scale indicates total traffic in two hours averaged per day.

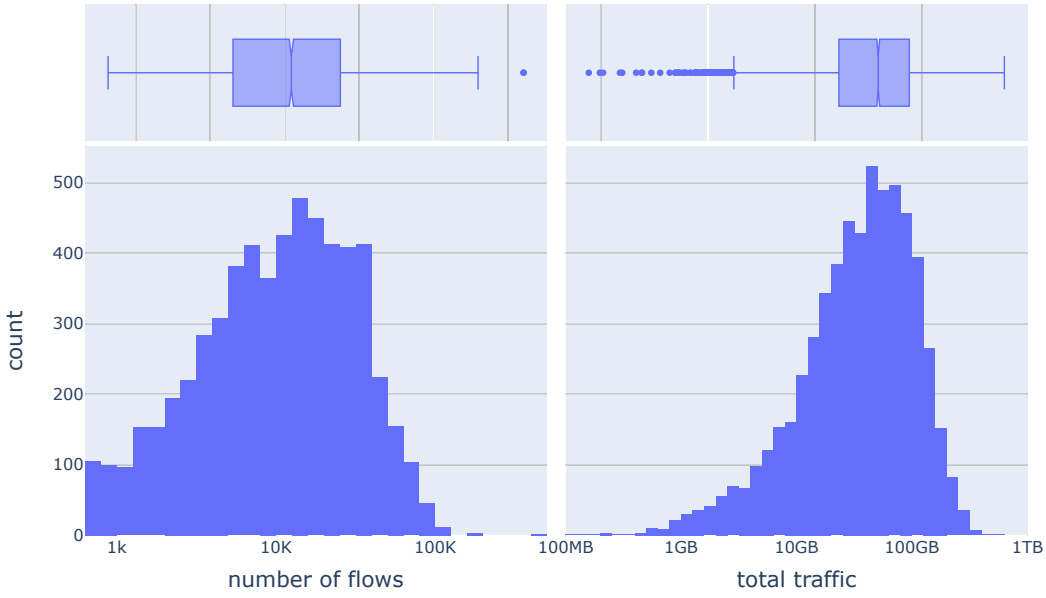


Figure 2.7: Flow count and traffic volume distributions of sites in logarithmic scale after filtration of low-traffic sites.

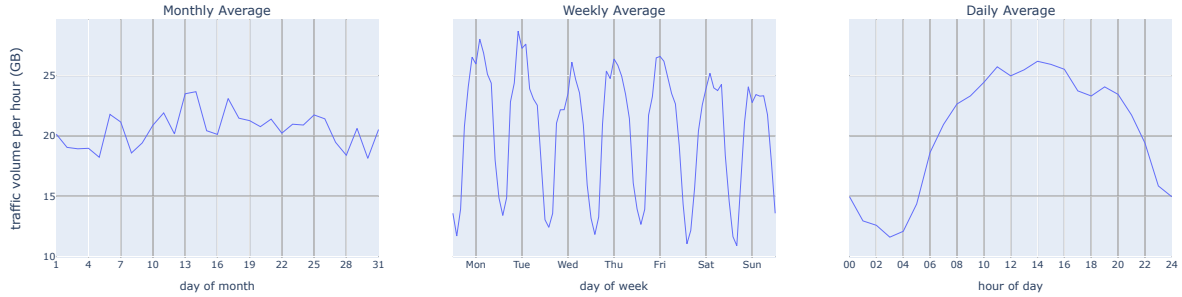


Figure 2.8: Patterns of the average cellular traffic in different time scales.

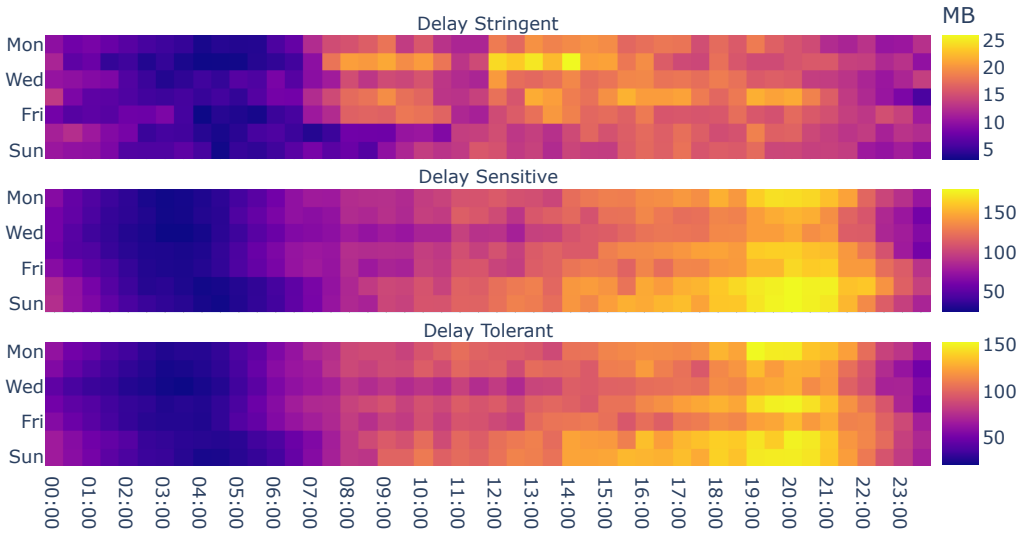


Figure 2.9: The traffic profile of average traffic of all sites.

2.1.6 Spectral filtering

Since cellular traffic depends on human activity, its pattern demonstrates strong periodicity, as shown in Figure 2.8(b). Therefore, for each traffic profile, we apply *fast Fourier transform* (FFT) to obtain the a frequency spectrum. Given a weekly traffic profile

$$x = (x_1, x_2, \ldots, x_N),$$

where $N = 336$ as we determined in the last section, we apply standard scaling (to make the profile zero-mean and unit-variance) followed by FFT to transform it into the frequency spectrum by calculating

$$X_k = \sum_{t=1}^{N} \frac{x_t - \bar{x}}{\sigma_x} \exp\left(-\frac{2\pi t k}{N} i\right), \quad k = 0, 1, \ldots, N-1, \tag{2.5}$$

where $\bar{x} = \frac{1}{N}\sum_{t=1}^{N} x_t$, $\sigma_x = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(x_t - \bar{x})^2}$, and i is the imaginary unit. k stands for frequency, i.e. the number of cycles in the period N, and X_k is the complex Fourier coefficient that estimates the magnitude and phase of the sinusoidal component of the profile with frequency k. Note that the standard scaling before the FFT makes the Fourier coefficients independent of the magnitude of the traffic volume.

In order to denoise the traffic profiles and reduce their dimensionality, we only select a few significant frequencies in the spectrum and drop the rest. We use *power* to measure the significance of each frequency in each profile, which is defined as

$$P_k = |X_k|^2, \tag{2.6}$$

for frequency $k = 0, 1, \ldots, N-1$. For each site and each delay category, we perform this transformation to the weekly traffic profile and obtain the Fourier coefficients and the corresponding power spectrum. Taking the average of the powers of the same frequency of the weekly traffic over all sites, we obtain the power spectrum as shown in Figure 2.10. We can observe that daily (1), half-daily (2), and weekly (1/7) are the 3 most distinctive periodicities of the weekly traffic.

We want to select M frequencies with the highest powers in the spectrum to filter the frequency spectrum of the weekly cellular traffic. To measure how good the spectral filtering is in the sense of maintaining the original traffic pattern, we subtract the weekly traffic profile reconstructed from the selected frequencies using *inverse fast Fourier transformation* (IFFT) from the original (standardized) traffic series, to obtain a residual series, as shown in Figure 2.12 (a).

If the filtered traffic series captures most of the original series' seasonal component, then the residual series should show little seasonality. This can be verified by a low auto-correlation of the residual series, as shown in Figure 2.13. In our experiment, we set $M = 17$, because it is the smallest value of M to produce a low auto-correlation of the residual series. The selected frequencies are labeled in orange in Figure 2.12 (b).

For the cellular traffic of every site in each delay category, we produce three variants of time series as its vectorization:

1. Standardized: the standardized time series of its weekly traffic. Its dimensionality is 336.

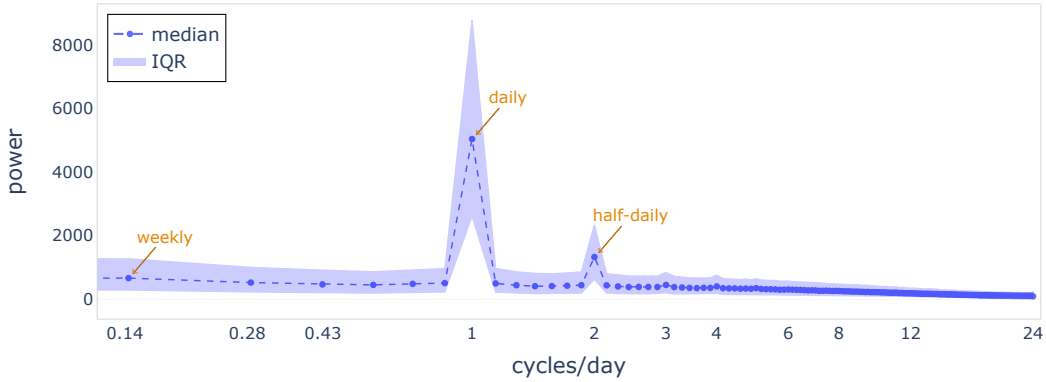


Figure 2.10: Power spectrum of average weekly cellular traffic. Orange labels indicate high-power frequencies, i.e. daily (1 cycle/day), half-daily (2 cycles/day), and weekly (1/7 cycles/day).

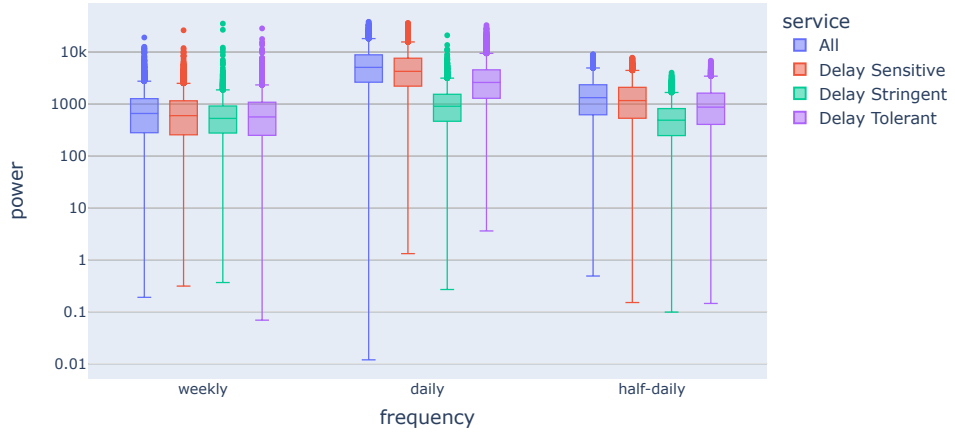
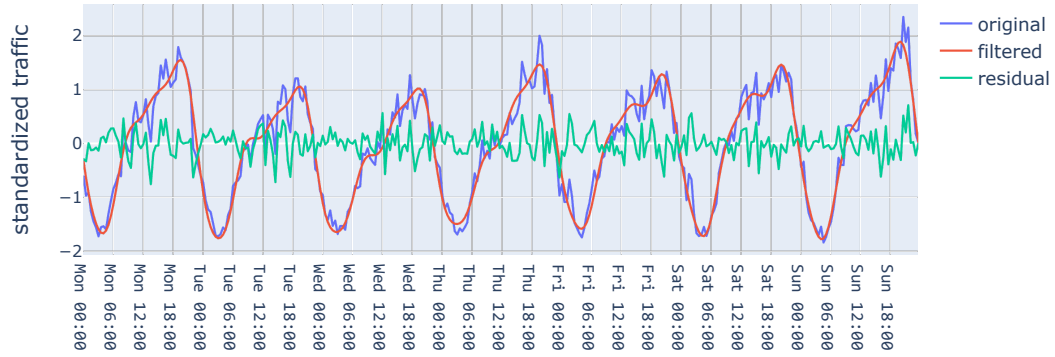


Figure 2.11: Box plot of spectral power distribution for each service category and for weekly, daily, and half-daily frequencies.

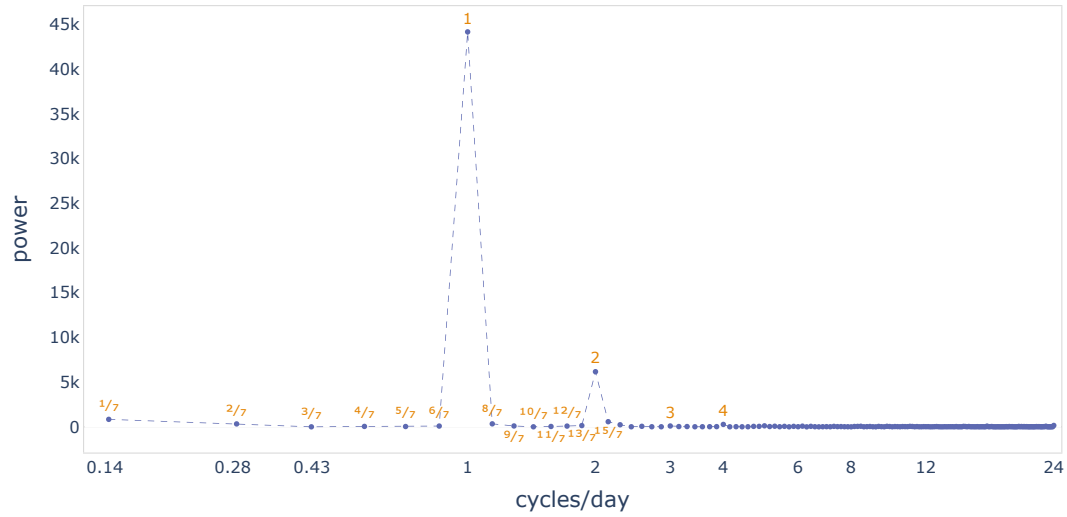
2. Filtered (frequency): the Fourier coefficients of the selected frequencies after FFT of the standardized series. Since the Fourier coefficients are complex numbers, we split their real and imaginary parts and combine them into a single vector. Thus its dimensionality is $2M = 34$.

3. Filtered (time): the time series reconstructed from the filtered frequency spectrum by IFFT. Its dimensionality is 336, the same as the original.

Finally, for each variant of vectorization, we concatenate the time series of delay-stringent, delay-sensitive, and delay-tolerant traffic together as a single vector to describe the full traffic pattern of a site. As a result, the vectorizations in the time domain have a dimensionality of 1008 and the vectorization in the filtered frequency domain has a dimensionality of 102.



(a) The standardized original traffic series, the filtered traffic series, and the residual series (difference between the formal two series) in the time domain.



(b) The power spectrum of the normalized traffic in the frequency domain with orange labels marking the selected frequencies in spectral filtering.

Figure 2.12: Spectral filtering of a typical weekly cellular traffic pattern in the time domain and the frequency domain.

2.2 Cell Classification

In order to classify the base stations by their traffic patterns, we apply clustering on the vectorizations of their weekly traffic obtained by the procedures we elaborated above. Clustering is an unsupervised machine learning method to divide a set of points into groups with the objective of maximizing the similarity of points within a group (or intra-cluster similarity) and minimizing the the similarity of points between groups (or inter-cluster similarity).

2.2.1 Hierarchical clustering

We choose to use the method of hierarchical clustering [14], because it has good generality (no assumptions in the distance measure) and provides a family of solutions with different number of clusters (so that we do not need to predetermine the number of clusters). The pseudo-code

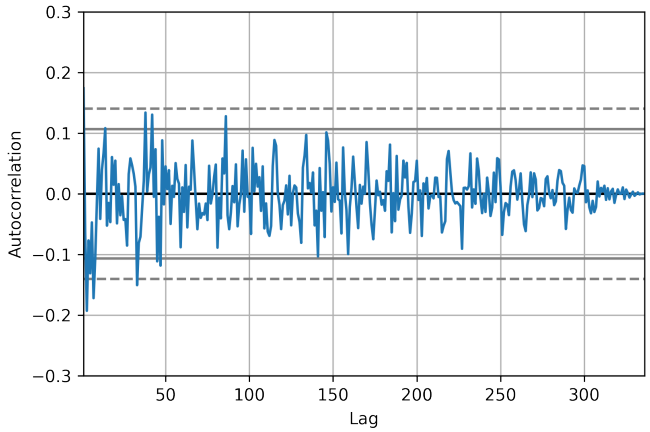


Figure 2.13: Auto-correlation plot of the residual series. Low auto-correlation values indicate that the seasonal components have been removed. The horizontal lines in the plot correspond to 95% (solid) and 99% (dashed) confidence bands.

Algorithm 1 Hierarchical clustering algorithm

Input: Data vectors $\{x_n\}_{n=1}^N$, cluster-wise distance $\mathrm{DIST}\,(\mathcal{G}, \mathcal{G}')$
$\mathcal{A} \leftarrow \varnothing$
for $n \leftarrow 1, \ldots, N$ **do**
    $\mathcal{A} \leftarrow \mathcal{A} \cup \{\{x_n\}\}$
end for
$\mathcal{T} \leftarrow \mathcal{A}$
while $|\mathcal{A}| > 1$ **do**
    $\mathcal{G}_1^\star, \mathcal{G}_2^\star \leftarrow \underset{\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{A}}{\arg\min}\, \mathrm{DIST}\,(\mathcal{G}_1, \mathcal{G}_2)$
    $\mathcal{A} \leftarrow (\mathcal{A} \setminus \{\mathcal{G}_1^\star\}) \setminus \{\mathcal{G}_2^\star\}$
    $\mathcal{A} \leftarrow \mathcal{A} \cup \{\mathcal{G}_1^\star \cup \mathcal{G}_2^\star\}$
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{\mathcal{G}_1^\star \cup \mathcal{G}_2^\star\}$
end while
Output: Tree $\mathcal{T}$.

of the hierarchical clustering algorithm is given in Algorithm 1.

To calculate the distance between a pair of clusters, we apply Ward's method, which aims to minimize the intra-cluster variance [15].

$$\begin{aligned}\mathrm{DIST}\,(\mathcal{G}_1, \mathcal{G}_2) = &\sum_{\mathbf{x} \in \mathcal{G}_1 \cup \mathcal{G}_2} \|\mathbf{x} - \overline{\mathbf{x}_{\mathcal{G}_1 \cup \mathcal{G}_2}}\|_2^2 \\ &- \sum_{\mathbf{x} \in \mathcal{G}_1} \|\mathbf{x} - \overline{\mathbf{x}_{\mathcal{G}_1}}\|_2^2 - \sum_{\mathbf{x} \in \mathcal{G}_2} \|\mathbf{x} - \overline{\mathbf{x}_{\mathcal{G}_2}}\|_2^2 \\ = &\frac{|\mathcal{G}_1|\,|\mathcal{G}_2|}{|\mathcal{G}_1| + |\mathcal{G}_2|} \|\overline{\mathbf{x}_{\mathcal{G}_1}} - \overline{\mathbf{x}_{\mathcal{G}_2}}\|_2^2\,,\end{aligned}$$

where $\overline{\mathbf{x}_{\mathcal{G}}} = \frac{1}{|\mathcal{G}|} \sum_{\mathbf{x} \in \mathcal{G}} \mathbf{x}$, for some set of vectors $\mathcal{G}$.

We can see in Algorithm 1 that hierarchical clustering yields a tree of successive agglomeration (known as a dendrogram) of clusters, starting from treating each point as a single cluster and

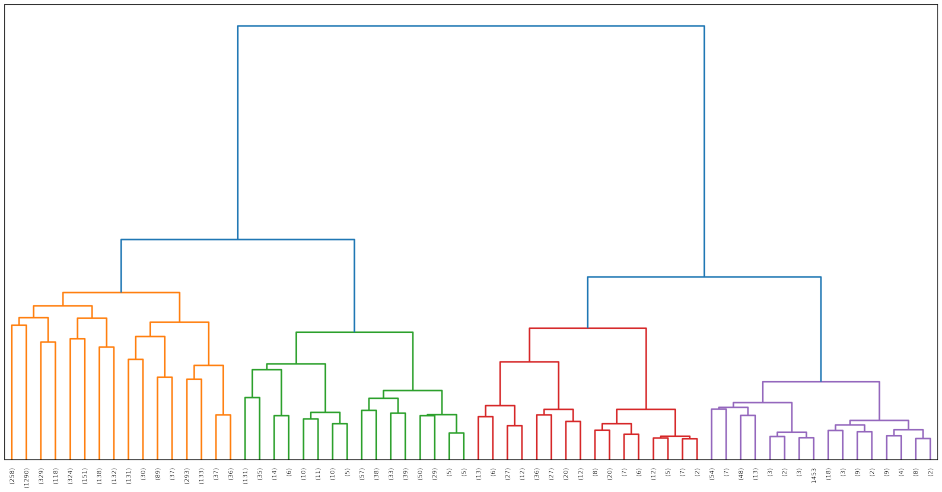


Figure 2.14: Ward linkage dendrogram of the traffic patterns of sites. Only the top 5 levels of the tree is shown for the sake of visual clarity. The numbers in parentheses at the bottom of the figure indicate the number of sites each truncated branch contains. The sub-trees' colors distinguish between different clusters obtained from this dendrogram for a certain cut-off distance threshold.

Figure 2.15: Number of clusters per cut-off distance threshold from the Ward linkage hierarchy. Both x-axis and y-axis are in log scales. At distance 200 all sites are included in the same cluster.

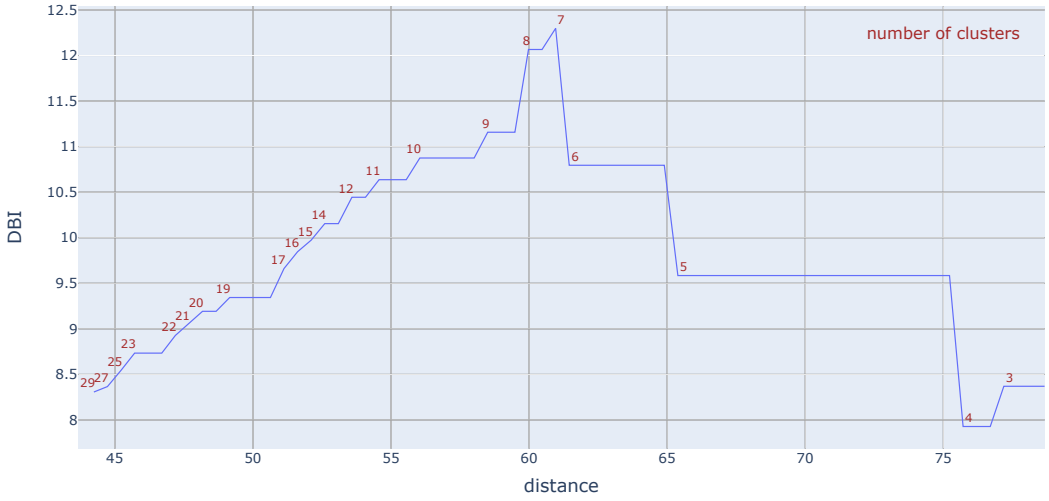


Figure 2.16: DBI score per cut-off distance from the Ward linkage hierarchy. A cluster number more than 30 or less than 3 is not considered because it will be difficult to find an interpretation of the clustering result.

ending at including all points in the same cluster, as shown in Figure 2.14. Different numbers of clusters can be obtained by setting different distance thresholds to cut off the agglomeration. The higher the threshold, the less the total number of clusters. For the vectorization with spectral filtering, the number of clusters in relation to the cut-off distance threshold can be seen in Figure 2.15.

2.2.2 Clustering performance metric

We use Davies-Bouldin index (DBI) to measure the quality of clustering, formulated as

$$\frac{1}{C}\sum_{i=1}^{C}\max_{1\leq j\leq C, j\neq i}\frac{S_i+S_j}{\|A_i-A_j\|_2}, \tag{2.7}$$

where $S_i = \frac{1}{N_i}\sum_{k=1}^{N_i}\|X_{i,k}-A_i\|_2$, $X_{i,k}$ is the vectorization of the cellular traffic of the k-th site in cluster i, $A_i = \frac{1}{N_i}\sum_{k=1}^{N_i}X_{i,k}$ is the centroid of each cluster, C is the number of clusters and N_i is the number of sites in the i-th cluster. Intuitively speaking, the DBI measures the ratio of intra-cluster variance to inter-cluster variance. Therefore, the lower the DBI value, the better quality the clustering has.

We compute Ward's hierarchical clustering using each of the three variants of site traffic vectorization we mentioned in Section 2.1.6. We have found that for Ward's method, using the traffic vectors after spectral filtering as input yields much lower (better) DBI than the original standardized vectors. Not only that, after many experiments, by using different numbers of selected frequencies and different cut-off distances, it turns out that after spectral filtering, the traffic vectorization in either the frequency domain or the time domain (reconstructed by IFFT) gives exactly the same clustering results. The underlying reason remains to be investigated. If this fact could be established, we can compute Ward's clustering more efficiently in the frequency domain, because the vectorization has a much lower dimensionality than that in the

time domain.

2.2.3 Selection of the optimal number of clusters

To find the optimal number of clusters, we compute the Ward's hierarchical clustering again using the vectorization in the filtered frequency domain, and we use different cut-off distances to produce different clustering results. After that, we calculate the DBI of each group of clusters, obtaining a relation between the cut-off distance and the DBI, as shown in Figure 2.16. We can see that the DBI increases as the cut-off distance increases until there are 7 clusters. After that it decreases and reaches its minimum when the number of clusters is 4. Therefore, we choose to use 4 clusters in the following clustering analysis.

2.2.4 Benchmarking

In order to comprehensively compare the clustering performance, in addition to Ward's hierarchical clustering, we use K-means and Gaussian mixture models (GMM), both commonly used clustering algorithms, to produce 4 clusters for each variant of site vectorization. The DBI values of all these clustering results are shown in Table 2.3. We can see that Ward's method produces the best clustering with the cellular traffic vectors after spectral filtering, in either the time domain or the frequency domain.

	Filtered (time)	Filtered (frequency)	Standardized
GMM	22.978381	21.959638	9.779725
K-means	9.870880	9.727646	9.430667
Ward	**7.928102**	**7.928102**	12.923762

Table 2.3: DBI scores obtained by different clustering algorithms applied on different variants of vectorization. The best scores are highlighted in bold.

2.3 Results and Analysis

2.3.1 Statistics

Including the low-traffic sites as cluster 0, we divide the sites into 5 clusters. Each cluster's size ratio and traffic ratio are shown in the bar charts in Figure 2.17. For each cluster and each delay category, we investigate the traffic distribution of sites by calculating its statistics, such as min/max/median sum rate and the hours in a week when the lowest/highest traffic density takes place, as summarized in Table 2.4. We also visualize these distributions by box plots in Figure 2.18.

2.3.2 Temporal patterns

For each cluster and each service category, we take the average of the traffic profiles of all sites in this cluster to obtain a representative profile. These representative profiles are shown

category	service	mean	std	median	max	peak time	min	vale time
(1) rural	All	10.41	4.41	10.77	18.84	Sat 20:00	2.83	Wed 03:00
	Delay Sensitive	5.15	2.30	5.17	9.81	Sun 20:00	1.35	Wed 03:00
	Delay Stringent	0.68	0.26	0.72	1.26	Tue 13:00	0.18	Tue 04:00
	Delay Tolerant	4.58	1.92	4.76	8.31	Sat 20:00	1.21	Wed 03:00
(2) residential	All	23.72	10.32	24.68	42.92	Mon 19:00	6.02	Wed 03:00
	Delay Sensitive	12.00	5.48	12.33	22.76	Sat 20:00	3.07	Wed 03:00
	Delay Stringent	1.61	0.72	1.72	3.58	Tue 14:00	0.24	Fri 04:00
	Delay Tolerant	10.11	4.39	10.41	20.10	Mon 19:00	2.66	Wed 03:00
(3) urban	All	6.02	3.60	6.24	12.95	Mon 15:00	0.67	Thu 02:00
	Delay Sensitive	3.30	1.99	3.35	7.85	Thu 16:00	0.36	Tue 03:00
	Delay Stringent	0.32	0.32	0.28	2.73	Tue 08:00	0.01	Wed 03:00
	Delay Tolerant	2.40	1.43	2.47	5.44	Tue 16:00	0.22	Thu 02:00
(4) office	All	5.23	2.90	5.09	12.00	Mon 11:00	0.97	Sun 04:00
	Delay Sensitive	2.64	1.43	2.63	6.14	Mon 11:00	0.54	Fri 03:00
	Delay Stringent	0.31	0.28	0.21	1.41	Tue 12:00	0.01	Fri 03:00
	Delay Tolerant	2.28	1.25	2.19	5.38	Mon 11:00	0.40	Sun 04:00

Table 2.4: Statistics of the classified cellular traffic. The numeric values in this table are the average sum rates (MB/s) of BSs in each cluster and each service category. "Peak time" and "vale time" are the hour in a week where the network traffic is the busiest and the least busy respectively.

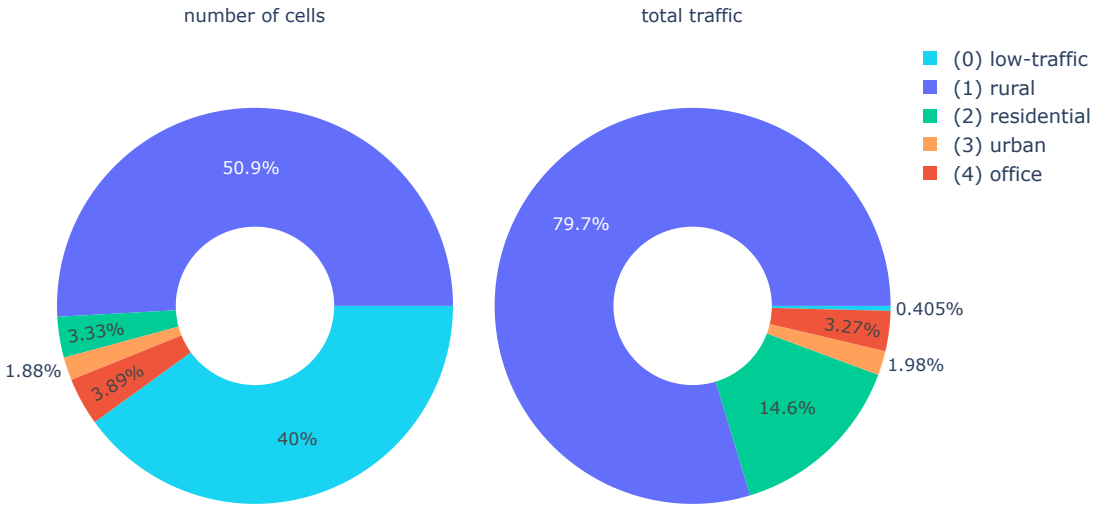


Figure 2.17: Distributions of the number of sites and the total traffic volume in each site category.

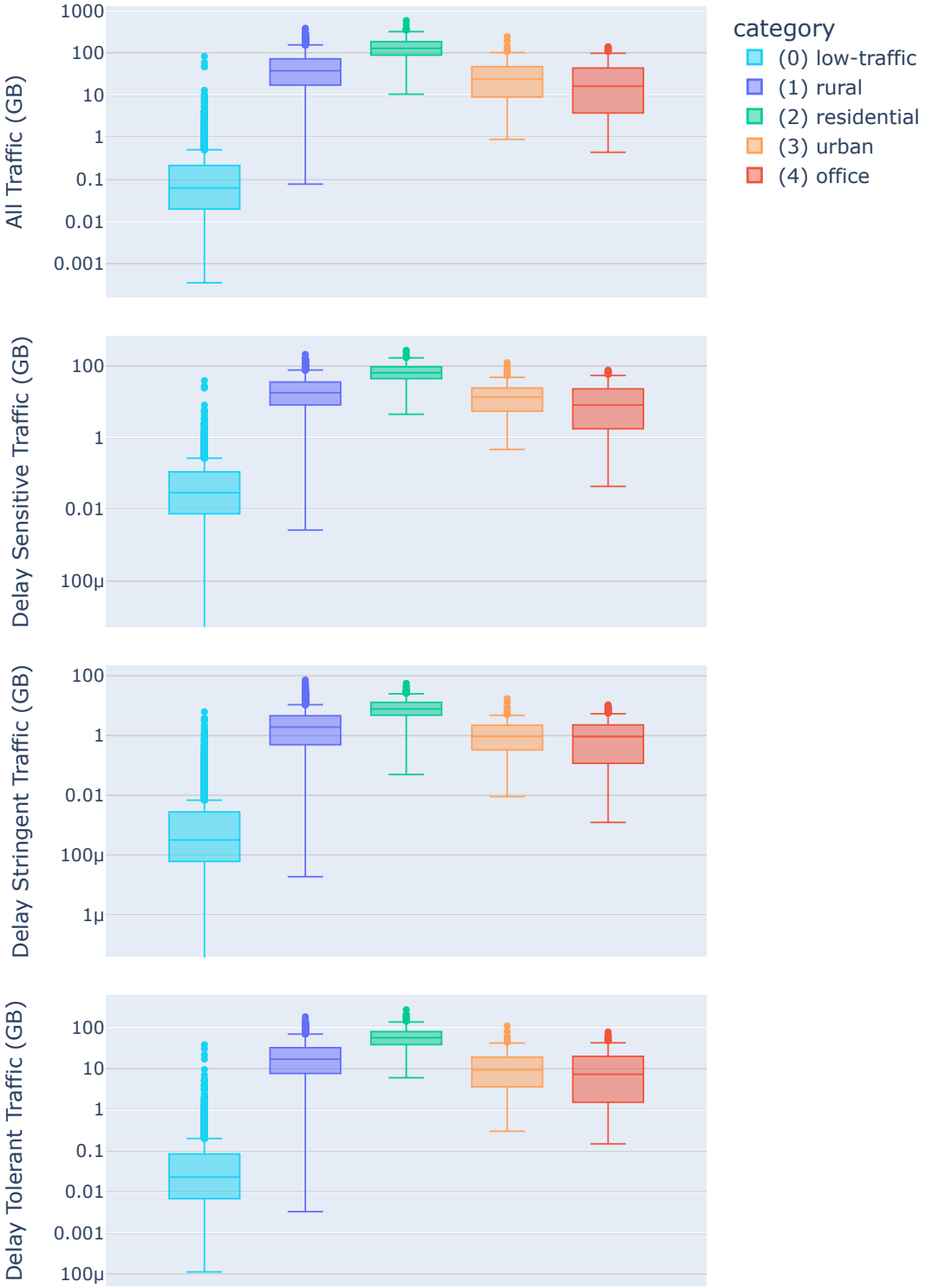


Figure 2.18: Box plots of total site traffic for different site categories for each service category.

as heatmaps in Figure 2.19 (only for total traffic). In order to compare the traffic patterns of different clusters in each service category, we apply standard scaling to these weekly profiles (so that they have the same scale) and plot them in Figure 2.20.

Finally, we think the comparison between traffic on weekdays and weekends is particularly interesting. Therefore we calculate the average traffic volume per day per site on weekends and on weekdays and take their ratio. Thus we obtain the distribution of weekend-weekday traffic volume ratio of sites in each cluster. We use box plots to visualize these distributions in Figure 2.21. We find that if we rank the clusters by weekend-weekday traffic ratio, the order would be $rural > residential > urban > office$.

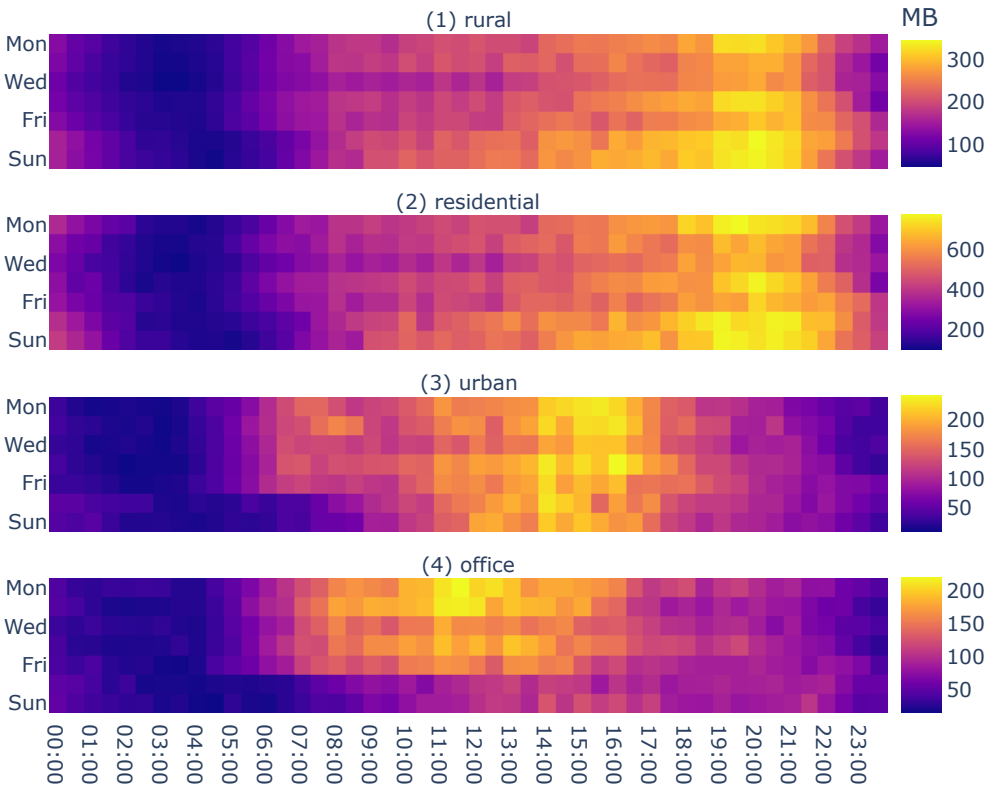


Figure 2.19: Average weekly traffic profile in each site category.

2.3.3 Spectral patterns

We plot the power spectrum of the average traffic profile of sites in each category in Figure 2.22. To visualize the clustering in the frequency domain, we scatter different categories of clusters in the phase-amplitude 2D space as in Figure 2.23. For a traffic profile of some site and some delay category, after we obtain the Fourier series by FFT, the phase and amplitude for a certain frequency from the corresponding complex Fourier coefficient.

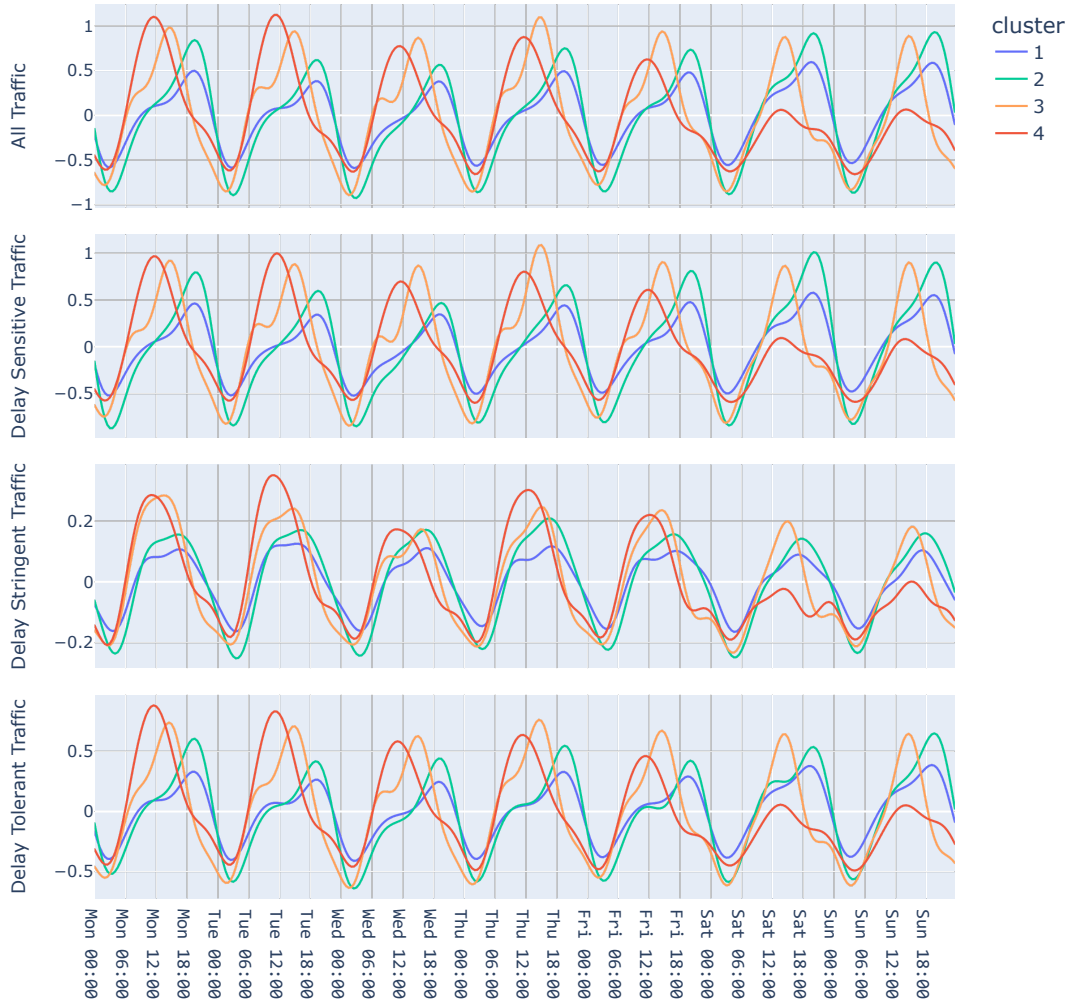


Figure 2.20: Average weekly traffic patterns for each site cluster.

2.3.4 Geographical distribution

For privacy protection, only a small fraction of sites have geographic information available. For these sites, we also know their radius of cell coverage. After assigning these sites different categories as we obtained in the clustering results, we plot them on an interactive map, as in Figure 2.24, using the same color scheme as in previous figures.

2.3.5 Interpretations of the site categorization

Combining the information about traffic statistics, weekly patterns, and geographical distribution about the result of site classification, we can attempt to interpret these categories:

(0) Low-traffic areas. Amounting to about 40% of all sites, the low-traffic sites only produced 0.4% of the total traffic.

(1) Rural areas. Almost all sites in the rural areas on the map are in the blue cluster (the few exceptions are in the residential (green) category). Therefore we interpret the blue cluster

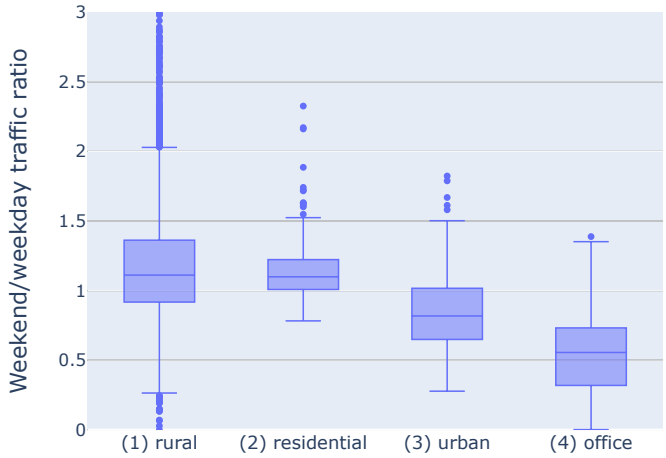


Figure 2.21: Weekend-weekday traffic volume ratio for each site category.

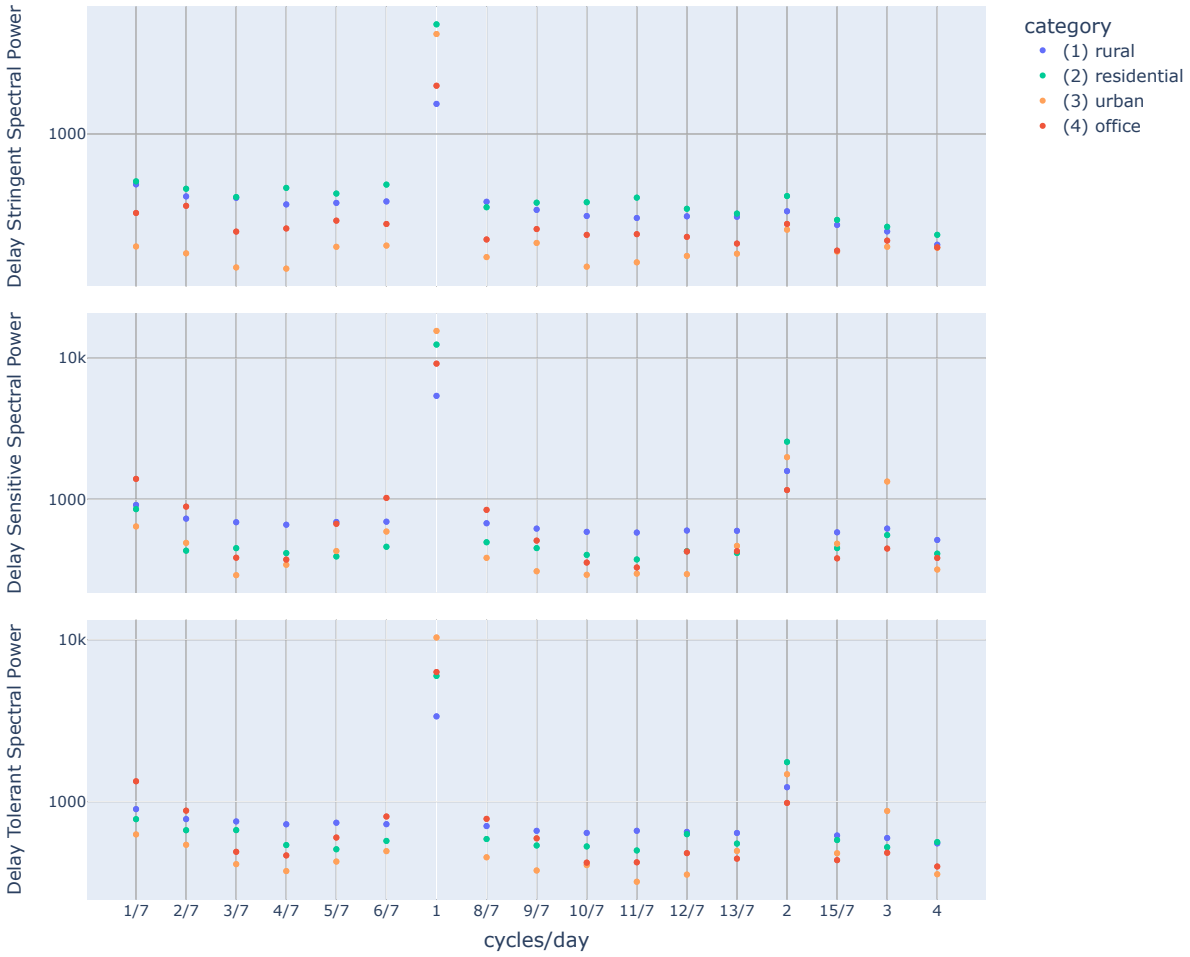


Figure 2.22: Spectral power of selected frequencies for each site category.

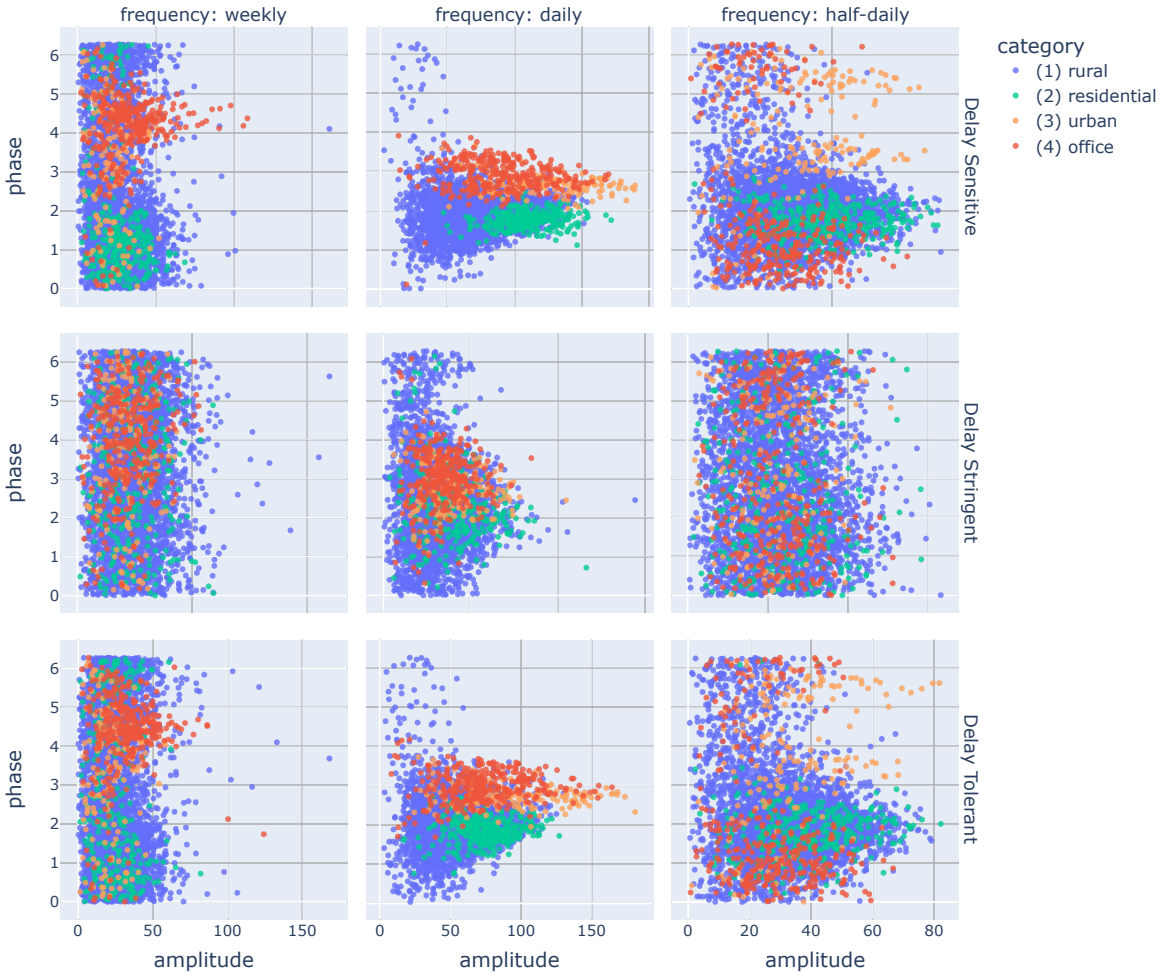


Figure 2.23: Phase and amplitude distribution in the frequency domain for each major frequency (daily, half-daily, weekly) and delay category.

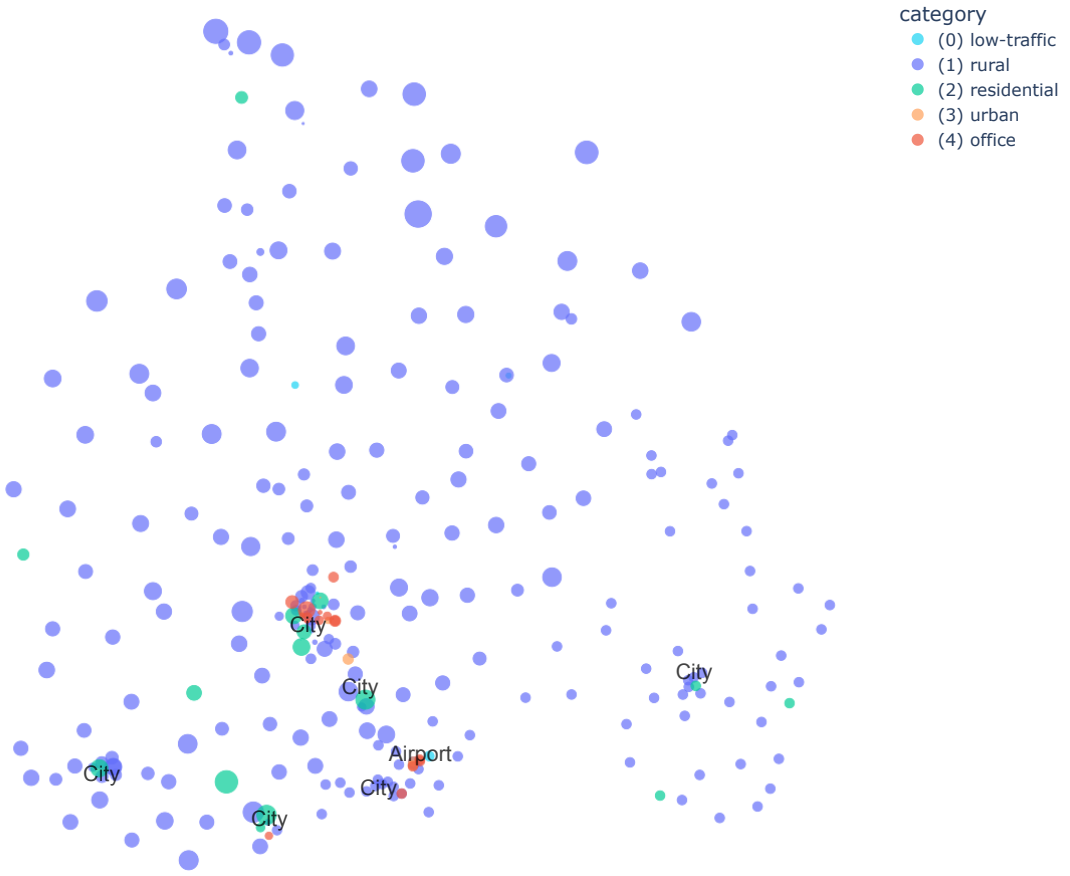


Figure 2.24: Geographical distribution of clustered sites. This map contains virtually all the sites with geographical information available.

as rural sites. It is the largest class in our categorization, including a bit more than 50% of the sites. Its average traffic per site is the second highest, only next to sites in the residential areas. Its daily traffic pattern seems to have a single peak between 19pm and 21pm. For most of the rural sites, the traffic on weekends is busier than on weekdays.

(2) Residential areas. After looking at the map, the sites in the green cluster tend to locate in residential areas. Its average traffic per site is exceptionally high, more than double that of the rural category. Its daily traffic pattern is quite similar to the rural, with the busiest hours between 19pm and 21pm. For about 75% of the residential sites, the traffic on weekends is busier than on weekdays.

(3) Urban areas. Sites in the orange cluster tend to locate at shopping centers, bus or railway stations, etc. For example a part of the airport and a shopping mall are in the orange cluster. Daily traffic of these sites tend to have a peak in the morning at around 8am and a peak in the evening between 15pm and 17pm. About 75% of the sites have a busier traffic on weekdays than on weekends.

(4) Office areas. The distribution of sites in the red cluster concentrates in the city centers. It has the lowest average traffic per site. Its daily traffic peaks at around 11am. For almost all sites in this category, the traffic on weekdays is distinctively busier than that on weekends. Therefore the red sites seem to be in areas where people work or study. For example the sites in the university are red, and sites near government offices are red as well.

Chapter 3

Multi-Agent RL for Energy-Efficient BS control

In this part of the thesis, we are going to develop an AI algorithm for dynamic BS energy saving. More specifically, a multi-agent reinforcement learning will be trained with the goal of minimizing the total energy consumption of multiple BSs in a network while preserving the overall quality of service by making decisions on the multi-level sleeping, antenna switching, and user association of these BSs. Before we introduce the learning algorithm, we will first go into the network system model as the simulation environment of the RL task, including massive MIMO channel, power consumption, user arrival, advanced sleep modes, and service mechanism. The modelling of user arrival links this part of the thesis with the first part by mimicking the temporal patterns of different traffic scenarios we discovered by clustering. The user association and inter-cell interference in the network necessitate the collaboration between individual BSs. To this end, the problem will be modelled as a decentralized partially observable Markov decision process (DEC-POMDP), and a multi-agent PPO (MAPPO) algorithm is proposed to learn a collaborative BS control policy. One major challenge of reinforcement learning in a DEC-POMDP is its non-stationary nature. The centralized training and distributed execution mechanism has been shown to be capable of mitigating the non-stationary problem and stabilizing the training. The trained MAPPO agent demonstrates an ability to significantly improve the network energy efficiency, adaptively switch the BSs into different depths of sleeping, reduce the inter-cell interference, and maintain a good quality of service.

3.1 Network System Model

In this section, we introduce the system modelling of the 5G network simulation. Seven base stations are deployed in an open space area and massive MIMO channel with zero-force precoding is considered in network service. Network traffic is simulated as a non-homogeneous Poisson process of user arrival. Each user has a certain size of traffic demand and a delay budget as described in Section 2.1.2. The arrival rate of the users is defined so as to mimic the traffic pattern in each of the delay category in one of the traffic scenarios we discovered by the end of Chapter 2. The available actions for the multi-agent BS control will be given in this section and the effects of different actions will be briefly described to give some intuition of an energy-efficient BS control policy.

3.1.1 Environment Setup

We choose to use a square area of 1km $\times$ 1km as our simulation environment. 7 base stations b_c ($c = 0, 1, \ldots, 6$) are placed in this area – one (b_0) locates in the center and the other six surround it as a hexagon with a side length of $d_{\text{BS}} = 400$ m, as shown in Figure 3.1. The simulation will proceed by time-stepping with a step length of $dt = 1$ ms. In each timestep, users may arrive at any random location in the area. Each user equipment (UE) demands the network service in one of the categories in Table 2.2. If a UE u_k can establish connection with a BS, during the period of service, its demand size, denoted by x_k, will decrease with data rate r_k. If u_k is not being served by any BS, its data rate $r_k = 0$. Let the initial demand size of u_k be $x_{\text{max},k}$. We denote the delay, i.e. the elapsed time since the arrival of u_k, by τ_k, and then the demand size of a UE over time follows the equation

$$x_k(t) = \max\left(0, x_{\text{max},k} - \sum_{i=1}^{\tau_k/dt} r_k(i \cdot dt)\, dt\right).$$

A UE will stay in the environment until its demand has been finished, i.e. $x_k = 0$, or its delay has reached the budget. Since each service category z has a different delay budget $\tau_{\text{max}}^{(z)}$, the maximum time u_k can stay in the environment is $\tau_{\text{max},k} = \tau_{\text{max}}^{(z_k)}$. Thus each UE requires a minimum data rate $r_{\text{req},k} = \frac{x_{\text{max},k}}{\tau_{\text{max},k}}$. At the time when u_k quits the network, we denote its remaining demand as χ_k and when $\chi_k > 0$, it will be "dropped ", which degrades the QoS of u_k. The network performance in our simulation is mainly based on the *drop ratio* $\frac{\chi_k}{x_{\text{max},k}}$.

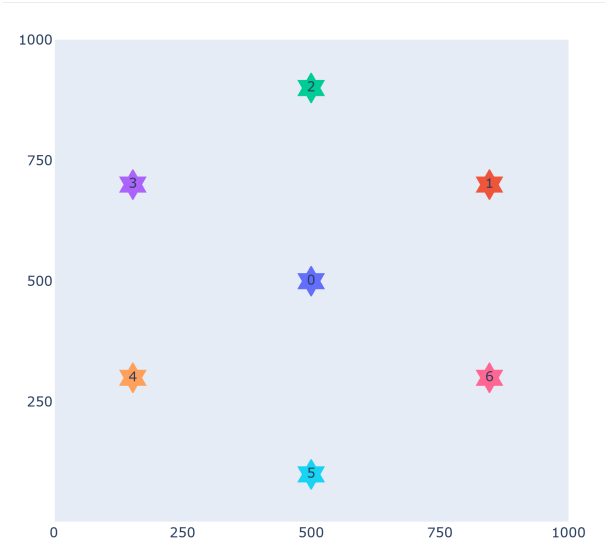


Figure 3.1: Base station layout in the simulation environment.

3.1.2 Traffic Model

Using cell classification, we want to create different cellular traffic scenarios for our simulation. Since cells in the blue category generated a very low amount of traffic and their traffic profiles are not as interesting as the rest three categories, we want to focus on these three in our simulation. We call the green category scenario A, the orange category scenario B, and the red one scenario

C. For each traffic scenario and each service category a, we aggregate the network flows into time slots of 20 minutes (a slightly finer granularity than in previous clustering) and take the weekly average, and then calculate the temporal-spatial traffic density $\kappa_{a,k}$ in each time slot k in Mb/s/km^2. The traffic density throughout a week in each traffic scenario shown in Figure 3.2 for delay stringent services, Figure 3.3 for delay sensitive services, and Figure 3.4 for delay tolerant services.

scenario	service category	mean	std	min	max	peak time
A	Delay Stringent	1.474373	0.961889	0.030689	4.490108	Mon, 08:00
	Delay Sensitive	23.147750	7.549798	7.950001	37.506251	Sat, 18:40
	Delay Tolerant	45.372236	11.748287	21.799172	64.098010	Sat, 14:20
	Total	69.994360	19.544497	31.707593	98.828759	Sat, 14:40
B	Delay Stringent	1.465876	0.978151	0.063916	3.815741	Mon, 08:20
	Delay Sensitive	39.983041	14.343149	13.831339	62.412639	Sat, 18:40
	Delay Tolerant	84.328688	28.138885	32.017521	127.785424	Mon, 15:00
	Total	125.777605	42.154542	48.428296	180.999283	Fri, 13:40
C	Delay Stringent	1.207334	0.909703	0.007066	4.088568	Tue, 08:00
	Delay Sensitive	35.837636	15.430952	10.215564	70.564844	Fri, 14:00
	Delay Tolerant	70.689616	40.308315	18.243969	158.088487	Tue, 13:20
	Total	107.734586	55.874644	31.578940	222.152462	Mon, 11:00

Table 3.1: Statistics of weekly cellular traffic in each scenario and service category.

We assume each UE only requests a single network flow and we model the traffic flows as a Poisson process with an average arrival rate of $\lambda_{a,t}$, equivalently the probability of a new UE to be generated in the area in a timestep. Its value follows the equation

$$\lambda_{a,t} = \frac{\kappa_{a,t} A}{x_{\max}} dt.$$

Although in reality the demand size varies, this variability can be compensated by the arrival rate λ, so it does not hurt to simplify the problem formulation by assuming a fixed demand size. Since there are 3 service categories, there are a maximum of 3 new UEs to arrive in a timestep.

At the start of an episode of simulation, a traffic scenario will be chosen and random UEs will be generated according to its traffic profile. Since the timestep is tiny in comparison to the time scale of the traffic profile, we reduce the time scale of the traffic (only affecting traffic patterns; the arrival rates remain the same) by a factor of 1/600 in the simulation.

Parameter	Value
Number of cells: C	7
Inter-site distance: d_{BS}	400 m
Simulation area: A	1 km^2
BS antenna height: h_{BS}	30 m
UE average height: h_{UE}	1.5 m
Simulation timestep: dt	1 ms
Time scale factor	1/600
Episode time length	1008 s
Delay budgets: $\tau_{\text{max}}^{(1/2/3)}$	50/150/300 ms
File size: x_{max}	3 Mbits

Table 3.2: Parameters for network and traffic models.

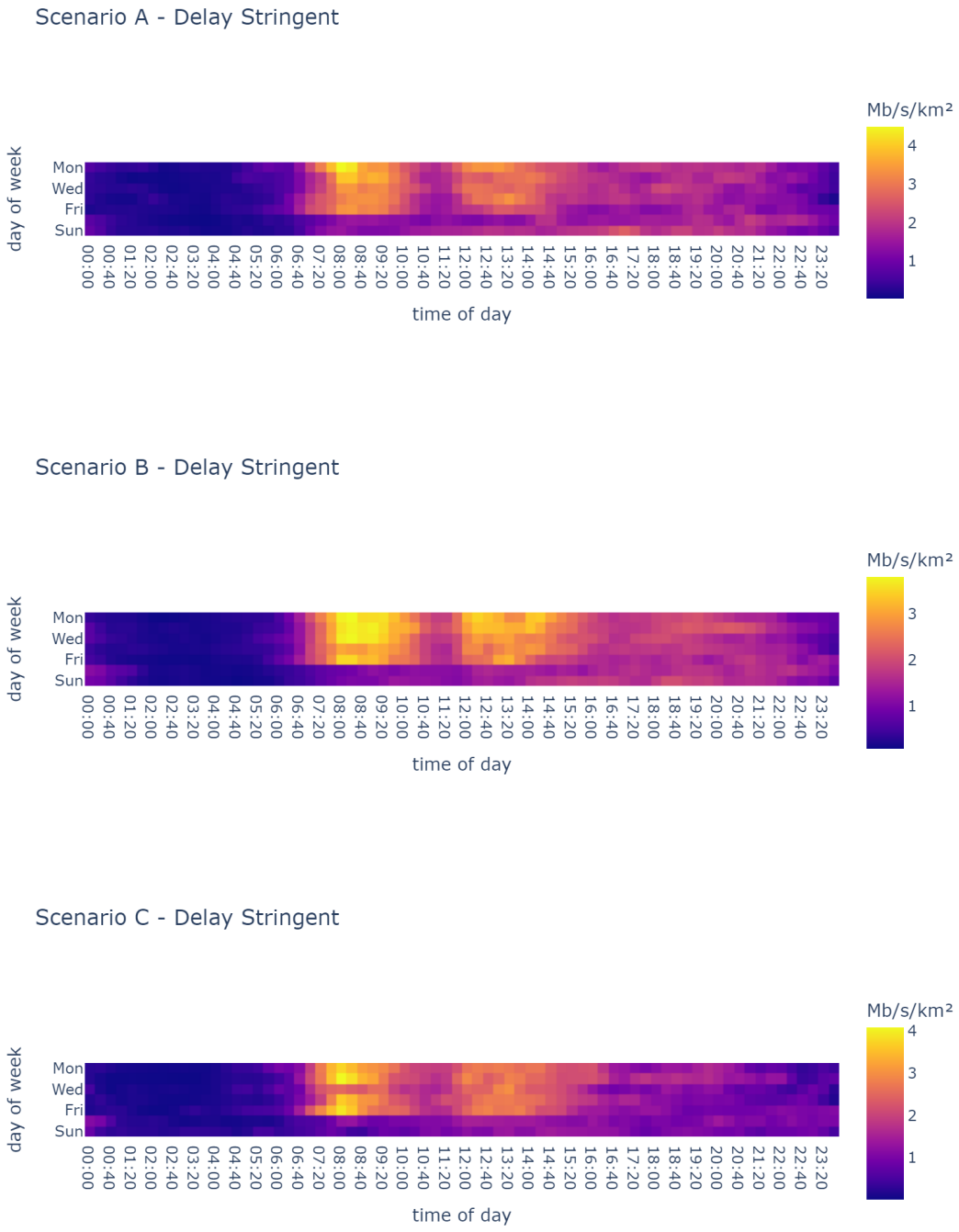


Figure 3.2: Weekly average traffic density of delay stringent services in different traffic scenarios.

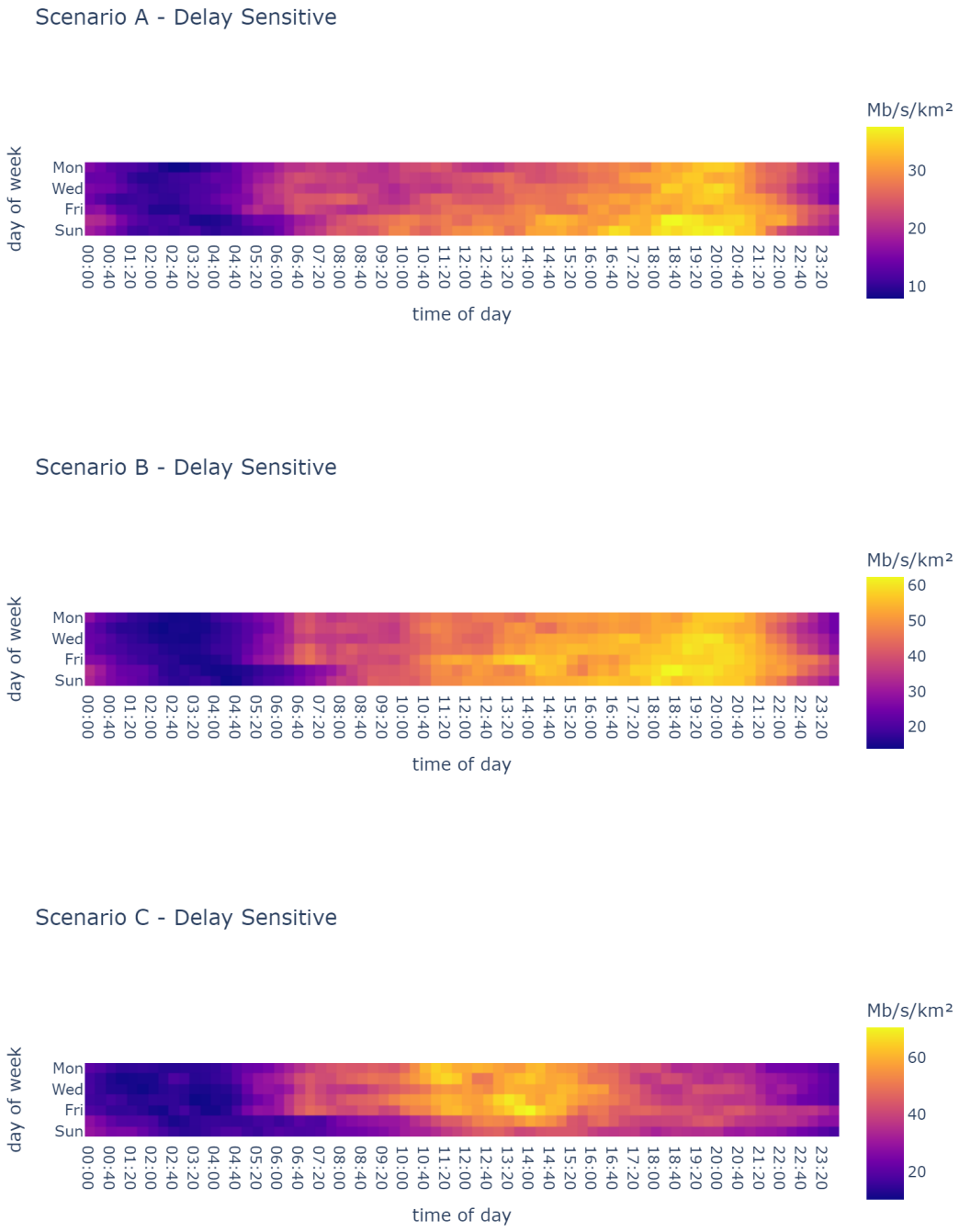


Figure 3.3: Weekly average traffic density of delay sensitive services in different traffic scenarios.

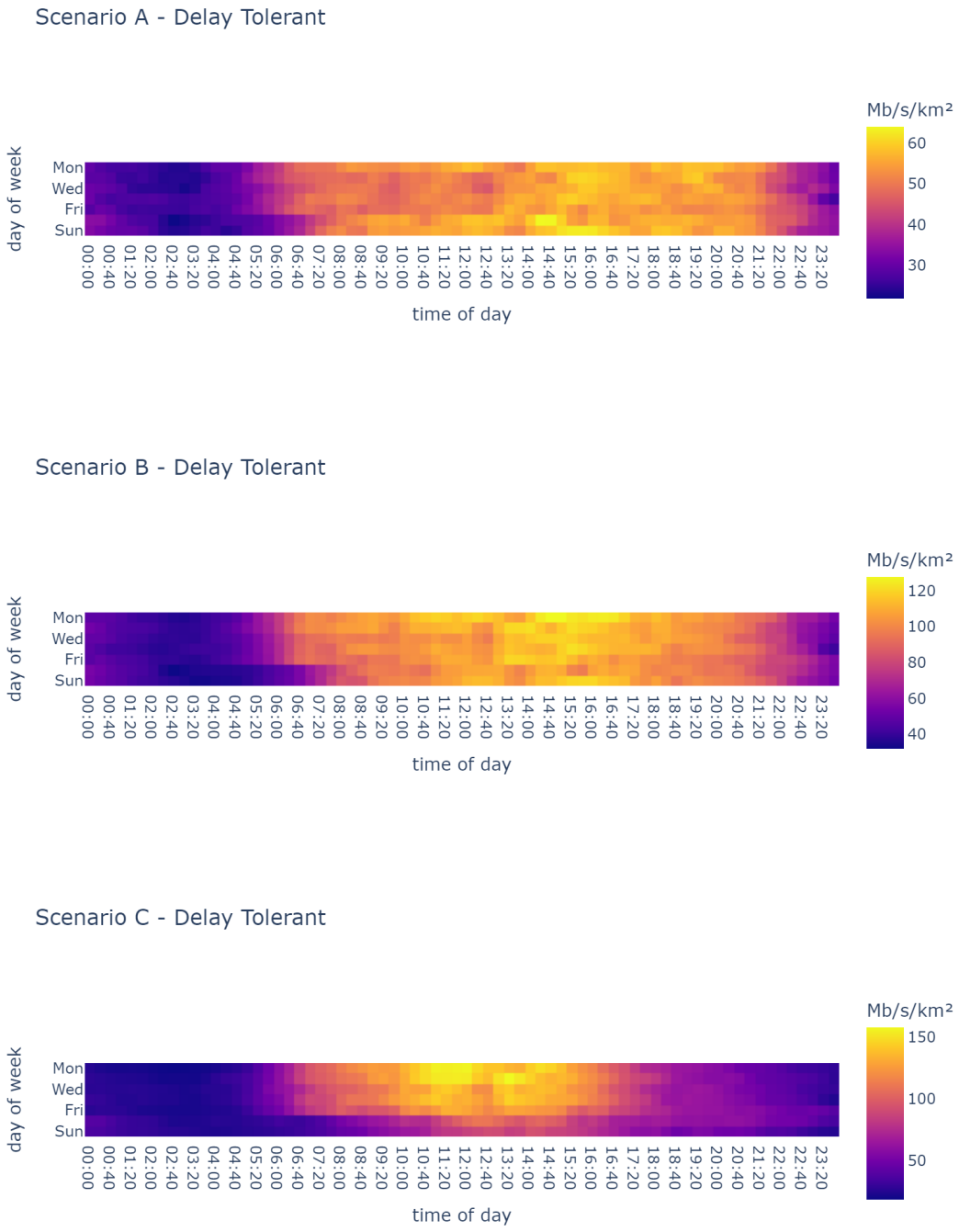


Figure 3.4: Weekly average traffic density of delay tolerant services in different traffic scenarios.

3.1.3 Channel Model

In this multi-cell network, each BS b_c has M_c active antennas and is serving K_c UEs. Thus there are a total number of $K = \sum_{c=1}^{C} K_c$ UEs in this network. Let c_k denote the index of the BS serving UE u_k for $k = 1, \ldots, K$. We assume that the communication bandwidth is B Hz for all the BSs. According to the Shannon channel capacity theorem, the maximum achievable data

rate of user k is

$$r_k = B \log_2 (1 + \text{SINR}_k), \quad \text{bit/s} \tag{3.1}$$

where SINR is the *signal-to-interference-plus-noise ratio.* Assuming perfect channel state information is available to UEs, the pilot sequences are perfectly orthogonal from cell to cell, and zero-forcing precoding is used at each BS, the SINR in a massive MIMO system is given by [16]

$$\text{SINR}_k = \frac{\mathcal{S}_k}{\mathcal{I}_k + \mathcal{N}} = \frac{(M_{c_k} - K_{c_k})\beta_{c_k,k} p_{c_k,k}}{\sum\limits_{c \neq c_k} \beta_{c,k} p_c + \mathcal{N}} \tag{3.2}$$

where $\mathcal{S}_k, \mathcal{I}_k, \mathcal{N}$ are the signal power, the interference power, and the noise power, respectively. $p_{c,k}$ is the transmit power allocated to UE u_k by BS b_c ($p_{c,k} = 0$ for $c \neq c_k$) and $p_c = \sum\limits_{c_k = c} p_{c,k}$ is the total output power of BS b_c. Assuming that the average output power per antenna is a constant value p_{avg}, then

$$\sum_{c_k = c} p_{c,k} = p_c = M_c p_{\text{avg}}, \quad c = 1, \ldots, C.$$

Let M_{max} be the maximum number of antennas each BS possesses, then the total output power of each BS c follows the constraint

$$p_c \leq M_{\text{max}} p_{\text{avg}}.$$

$\beta_{c,k} \geq 0$ is the channel gain from BS b_c to user k is modeled as

$$10 \log_{10} \beta_{c,k} = PL(d_{c,k}) + \chi,$$

where PL models the path loss, $d_{c,k}$ is the distance between BS b_c and UE u_k, and χ is a random variable modeling shadow fading that follows a Gaussian distribution $\mathcal{N}(0, \sigma_{\text{SF}})$.
According to the UMi (micro urban) model with NLOS in the 3GPP TR 38.901 report,

$$PL(d_{c,k}) = 35.3 \log_{10} d_{c,k} + 22.4 + 21.3 \log_{10} f_c - 0.3 (h_{UE} - 1.5)$$

and $\sigma_{\text{SF}} = 7.82$.
The noise power is given by

$$\mathcal{N} = B 10^{(N_0 + N_B)/10},$$

where N_0 is the noise power spectral density, and N_B is a constant called "noise figure " depending on the type of BS.

3.1.4 Power Consumption Model

We can model the total power consumption of an active BS b_c as follows [16]:

$$P_c^{\text{active}}(K_c, M_c) = M_c P_{\text{PA}}(p_c) + P_{\text{BB}}(K_c, M_c) + P_{\text{oth}}, \tag{3.3}$$

where $P_{\text{PA}}(p_c)$ gives the power consumption of a PA when its average output power is p_c, $P_{\text{BB}}(K_c, M_c)$ is the baseband signal processing power when the BS serves K_c number of users simultaneously with M_c number of antennas. P_{oth} includes the load-independent power for

site cooling, control signal, DC-DC conversion loss, etc. We assume the BS adopts envelope tracking PA (ET-PA) since it is more energy-efficient, whose PC is given by [17]

$$P_{\mathrm{PA}}(p) = \frac{p}{(1+\epsilon)\eta} + \frac{\epsilon P_{\mathrm{max,PA}}}{(1+\epsilon)\eta},$$

where ϵ is a parameter depending on the PA type. According to [18], the baseband PC can be calculated as

$$P_{\mathrm{BB}}(K_c, M_c) = A_{\mathrm{cd}} \sum_{c_k=c} R_k + \sum_{i=0}^{3} C_{0,i} K_c^i + M_c \sum_{i=0}^{2} C_{1,i} K_c^i,$$

where $C_{0,0} = P_{\mathrm{syn}}, C_{0,1} = 0, C_{0,2} = 0, C_{0,3} = \frac{B}{3T_c L_{\mathrm{BS}}}, C_{1,0} = P_{\mathrm{BS}}, C_{1,1} = \frac{B}{L_{\mathrm{BS}}}\left(2 + \frac{1}{T_c}\right), C_{1,2} = \frac{3B}{L_{\mathrm{BS}}}$, and $\sum_{c_k=c} R_k$ is sum rate of all UEs being served by b_c (as given by equation 3.1).

When a BS b_c is actively transmitting network data, we assume its average antenna power p_c is a constant value p_{avg}, regardless of the number of UEs being served, unless there are no UE requests at all and thus no transmission. In this case the BS is in the idle mode and its p_c becomes zero. Accordingly the PC is

$$\begin{aligned} P_c^{\mathrm{idle}}(M_c) &= M_c P_{\mathrm{PA}}(0) + P_{\mathrm{BB}}(0, M_c) + P_{\mathrm{oth}} \\ &= M_c \left(\frac{\epsilon P_{\mathrm{max,PA}}}{(1+\epsilon)\eta} + P_{\mathrm{BS}} \right) + P_{\mathrm{syn}} + P_{\mathrm{oth}}. \end{aligned}$$

We model the PC of a sleeping BS as a discounted value of P^{idle}:

$$P_c^{\mathrm{sleep}}(M_c) = \delta_{s_c} P_c^{\mathrm{idle}}(M_c),$$

where s_c is the *sleep level* of b_c and δ_s is the discount parameter in sleep level s. The concept of different sleep levels of a BS will be elaborated further in the next section.

If we also allow $s = 0$ to represent the active mode and let $\delta_0 = 1$, then the total PC of BS b_c can be expressed in a unified formula

$$P_c(K_c, M_c) = \delta_{s_c} \left(M_c P_{\mathrm{PA}}(p_c) + P_{\mathrm{BB}}(K_c, M_c) + P_{\mathrm{oth}} \right),$$

where

$$p_c = \begin{cases} p_{\mathrm{avg}} & K_c > 0, \\ 0 & K_c = 0. \end{cases}$$

Parameter	Value
Bandwidth: B	2×10^7 Hz
Noise power spectral density: N_0	-204 dB ·J
Noise figure: N_B	7 dB
Signal power threshold: p_{thr}	-87 dBm
Average transmit power per antenna: p_{avg}	0.1 W
Minimum number of antennas: M_{min}	16
Maximum number of antennas: M_{max}	64
Maximum PA output power: $P_{\text{max,PA}}$	3 W
Maximum PA efficiency at $P_{\text{max,PA}}$: η	80%
ET-PA parameter: ϵ	0.0082
Local oscillator power: P_{syn}	1 W
BS circuit power: P_{BS}	1 W
Other power: P_{oth}	18 W
Power for data coding and decoding: A_{cd}	1 W/(Gbit/s)
Channel coherence time: T_c	5000 symbols
BS computational efficiency: L_{BS}	12.8 Gflops/W

Table 3.3: Parameters for channel and power models.

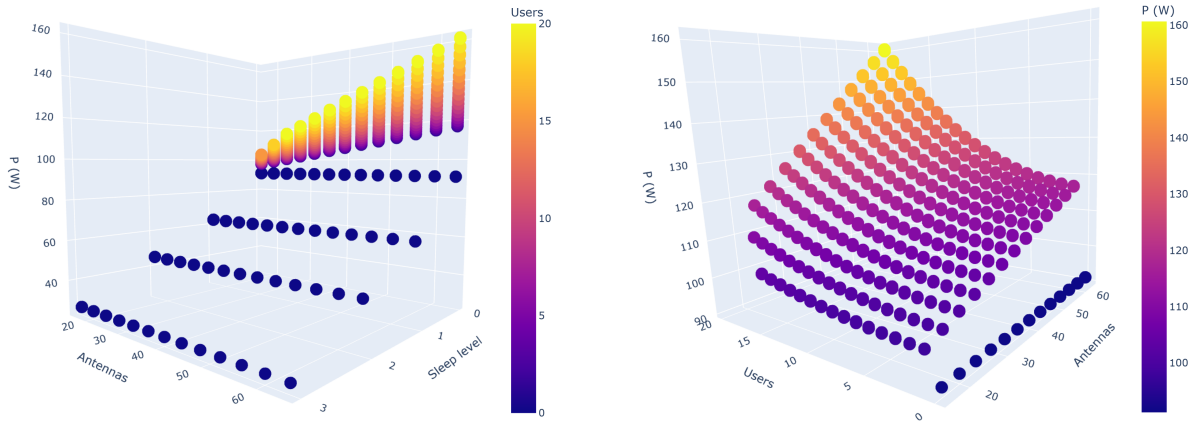


Figure 3.5: Power consumption P versus antenna number, UE number, and sleep level.

3.1.5 Advanced Sleep Modes

Even in the idle mode, the hardware of the BS remains fully active and ready to start serving user demands instantly. However partial deactivation of the BS hardware, called BS sleeping, can result in further PC reduction, with the cost of an activation delay when network requests arrive, possibly to the detriment of the QoS experienced by UEs [19]. Since different hardware components of the BS have different PC and activation delays, a multi-level sleeping mechanism called *advanced sleep modes (ASM)* can be designed where the BS goes to deeper sleep levels by incrementally deactivating components with longer activation delays, resulting in lower total

PC but longer overall wake-up latency. As in [19], there can be 4 sleep modes (SM 1-4) with an activation delay from 71 μs to 1s and Table 3.4 provided by [20] gives which parts of BS hardware can be deactivated in each sleep mode.

In addition to user requests, a BS also need wake up periodically to broadcast controlling signals in order to be detectable by UEs. Previous cellular technologies like LTE use a multitude of cell-specific signals for channel estimation and synchronization like *Cell-specific Reference Signals* (CRS), *Primary and Secondary Synchronization Signals* (PSS and SSS respectively). These cell-specific signals have to be "always on " (regardless of user requests) and the tight intervals between them prevent BSs in an LTE system from going to sleep levels deeper than SM1. This multi-level sleeping mechanism with activation for data or signal transmissions can be illustrated in Figure 3.6.

Components	**Subcomponents**	**SM$_1$** (71μs)	**SM$_2$** (1ms)	**SM$_3$** (10ms)	**SM$_4$** (1s)
PA		X	X	X	X
Digital Baseband	Predistorsion			X	X
	BB filtering	X	X	X	X
	Up-Down sampling	X	X	X	X
	FFT/IFFT	X	X	X	X
	MIMO precoding	X	X	X	X
	Synchronization	X	X	X	X
	Channel estimation	X	X	X	X
	Equalizer computation	X	X	X	X
	Equalization	X	X	X	X
	OFDM Mod/Demod	X	X	X	X
	Mapping/Demapping	X	X	X	X
	Channel coding	X	X	X	X
Digital Control	Control				
	Backhaul				
	Network				
Analog Front End (Rx)	LNA	X	X	X	X
	LNA 2	X	X	X	X
	Frequency synthesis		X	X	X
	Mixer	X	X	X	X
	VGA	X	X	X	X
	CLK generation				X
	ADC			X	X
Analog Front End (Tx)	Modulator	X	X	X	X
	Buffer	X	X	X	X
	Frequency synthesis		X	X	X
	Feedback VCO		X	X	X
	Feedback Mixer	X	X	X	X
	CLK generation				X
	DAC		X	X	X
	Feedback DAC			X	X
Power Supply	AC/DC				
	DC/DC				
	Cooling				

Table 3.4: Deactivated sub-components in each sleep mode.

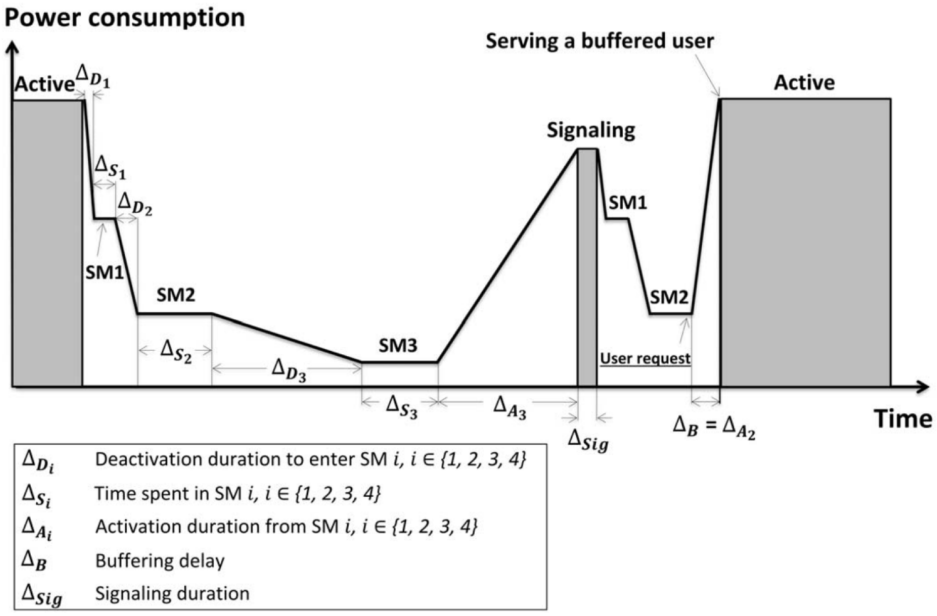


Figure 3.6: ASM mechanism [2, 3].

However, the usage of SSB still imposes a constraint on the maximum duration of the sleeping (160ms), thus rendering SM4 with a 1 second activation latency infeasible. In our experiment, we also reckon that the activation delay of SM1 (71 μs) is so short that it can be switched to greedily – whenever there is no traffic demand, with a negligible impact on the QoS. We choose to combine the remaining two SMs with another level of sleeping investigated in [21], for which the activation latency is 100ms. Including the active mode as sleep level 0, there are a total of 4 levels of sleeping in our experiment, each having a different PC discount and activation latency. The values of these parameters are adopted from [21] [2] and shown in Table 3.5.

Sleep level s	0 (Active)	1	2	3
Activation latency Δ_s	0 ms	1 ms	10 ms	100 ms
PC discount δ_s	1	0.69	0.5	0.29

Table 3.5: Operation modes

3.1.6 Base Station Actions

The set of actions a BS can take in the simulation is defined as

$$\mathcal{A} = \mathcal{M} \times \mathcal{S} \times \mathcal{U}, \tag{3.4}$$

where

$$\mathcal{M} = \{-4, 0, 4\}$$

stands for the options of antenna switching: -4 for switching off 4 antennas, 0 for no switching, and 4 for switching on 4 antennas;

$$\mathcal{S} = \{0, 1, 2, 3\}$$

represents actions for switching to the corresponding sleep level $s \in \mathcal{S}$;

$$\mathcal{C} = \{-1, 0, 1\} \tag{3.5}$$

determines user offloading: (1) Accept service requests without offloading (when the BS is active, this means immediate connection, and when it is sleeping, the user will be put into the service queue of this BS); (0) Offload new users to other BSs with $U = 1$; (-1) Detach all associated users and offload new users. When a UE is associated to a BS, it is either being served or waiting in the queue of this BS. For actions in $\mathcal{U}$, -1 is particularly useful when a BS goes to sleep, so that the detached UEs can request new connections with other BSs. However, the choices remain available that a sleeping BS wants to keep its current UEs in its queue until it wakes up again by taking action 0 or even put newly arrived UEs into its queue by taking action 1.

The action interval of a BS agent is set to 20ms, which is also the default periodicity of the SS block transmission [2]. This period, however, is too long for SM1 and SM2 sleeping. Therefore, we add a automatic wake-up mechanism to both SM1 and SM2, where the BS wakes up by itself as long as there is traffic demand during sleeping at these two levels. Since the the activation delay of SM1 is really short, we also want it to go back to SM1 once it has it has finished its service, thus allowing it to automatically switch between the active mode when there is traffic load and SM1 when there is no load. For SM2, however, it would have already taken 10ms to wake up, so we let it keep awake before its next action.

3.1.7 Signal Coverage

In order for a UE u_k to establish connection with BS b_c, the maximal signal power it could receive should be beyond a threshold p_{thr}, which means

$$\begin{aligned}
(M_c - K_c)\,\beta_{c,k} p_c = (M_c - K_c)\,\frac{10^{-\varphi}}{d_{c,k}^{\alpha}} p_c &\geq p_{\text{thr}} \\
d_{c,k} &\leq \left(\frac{(M_c - K_c)\,M_c p_{\text{avg}}}{10^{\varphi} p_{\text{thr}}} \right)^{1/\alpha} \\
&\propto \left((M_c - K_c)\,M_c \right)^{1/\alpha}.
\end{aligned}$$

We assume the signal coverage of b_c is a circular area and this upper bound of distance is its radius. b_c is allowed to acquire information of all UEs in this circle, such as data rate, demand size, service time limit, as a part of its observation in the environment.

3.1.8 User Association

For every UE u_k newly arrived in the network environment, it sends a service request to the BS b_c containing itself in the signal coverage with the strongest *average* signal power it is estimated to receive: $(M_c - K_c)\,\beta_{c,k} \frac{M_c}{K_c} p_{\text{avg}}$. If the BS is in the mode $U = 1$ as defined in Eq. 3.5, then

the network connection will be established except that it is sleeping or its users are already full ($K_c = M_c - 1$), in which case this UE will be put into its service queue q_c. UEs in q_c will be automatically connected once b_c has waked up and its UEs are not full. Otherwise, if b_c decides to offload new UEs ($U = -1, 0$ in Eq. 3.5), the UE has to continue to send another service request to the BS which also contains itself in the signal coverage and provides the next strongest signal coverage. The UE will repeat this process until it is attached to a BS, i.e. connected if the BS is active or put into the queue if the BS is sleeping, or it has no more BSs to send requests to. In the latter case the UE will remain idle and start to send a new round of service requests in the next time step. Once attached to a BS, a UE will keep belonging to the same BS until this BS decides to detach all its UEs ($U = -1$) or the UE leaves the network environment with its traffic demand either finished or dropped.

3.1.9 Power Allocation

After the connection is established, the BS reallocates its output power using a relatively simple method. As a first-order approximation, other conditions given the same, the data rate of a UE u_k satisfies $r_k = O\left(\log_2 p_{c_k,k}\right)$, as can be seen in Eq. (3.1) and (3.2). Therefore, suppose UE u_k requires a minimum data rate of $r_{\min,k} = \frac{x_k}{\tau_{\max,k} - \tau_k}$, we introduce a weight vector $w_p = (w_{p,1}, \ldots, w_{p,K_c})$ to allocate the power as

$$p_{c,k} = \frac{w_{p,k}}{\sum_j w_{p,j}} p_c.$$

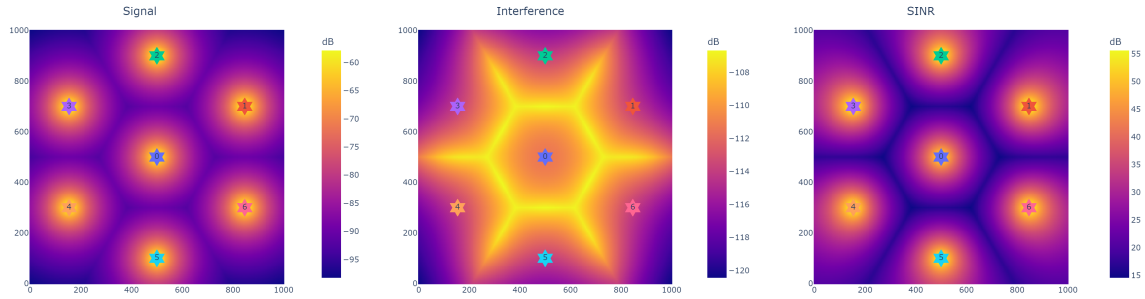
where

$$w_{p,k} = 2^{r_{\min,k}}.$$

3.1.10 Effects of Actions on Signal Quality

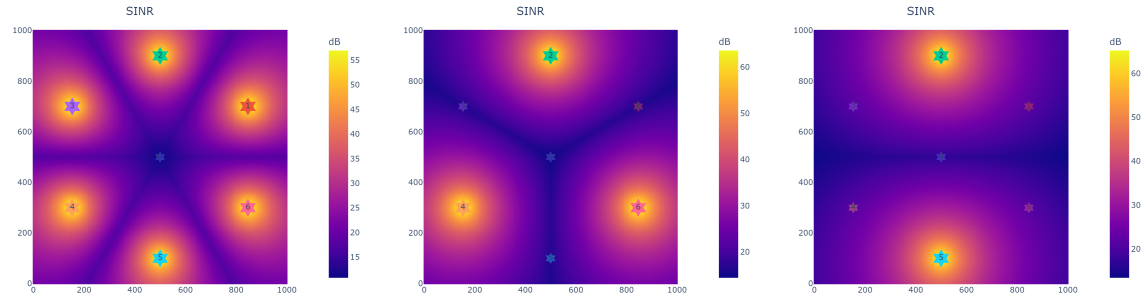
To visualize the effect of BS actions (switching antennas, sleeping, accept/reject new connections) on the signal quality, we create a grid of points in the simulation area and for each point, we use a UE as a test probe to measure the signal power, interference, and SINR at this location, following the service mechanism we mentioned above to decide which BS to connect with.

In the Figure , with a slight abuse of notations, we denote "BS 0,1,2,3 have 64 antennas turned on " by $M_{0123} = 64$, "BS 0,1 are in sleep level 0, BS 2,3 are in sleep level 1 " by $s_{01} = 0, s_{23} = 1$, and "BS 0,1,2 accepts new connections with BS 3 rejects " by $a_{012} = 1, a_3 = 0$.



(a) (b) (c)

$M_{0123456} = 64, s_{0123456} = 0, a_{0123456} = 1$ (the default BS configurations unless otherwise specified).



(d) $s_0 = 1$ (e) $s_{0135} = 1$ (f) $s_{01346} = 1$

Max SINR slightly increases due to the sleeping of BS 0, but min SINR decreases.

Max SINR increases a lot and interestingly min SINR also increases.

Max SINR is slightly higher than (2) but min SINR becomes slightly lower.

Figure 3.7: Visualizations of signal quality in the environment in different network configurations.

With only BS 2, 4, 6 actively operating, both the maximum and the minimum SINR over the whole area are increased in comparison to all 7 BSs turned on. This is due to the strong interference between neighbor BSs when they are close to each other. As the ISD (inter-site distance) gets larger, the maximum SINR in the area will certainly increase due to the decrease of the interference, but the minimum SINR will increase first for the same reason, as can be seen by comparing (c), (d), (e) in Figure 3.7. However, comparing (e) with (f), we see that the decay of signal becomes faster than the decay of interference as two BSs get further apart.

The multi-cell network is an *environment* (as demonstrated in Figure 3.8) where the base stations are *agents* that can take actions which will change the *state* of the environment. In our simulation, this proceeds in discrete time steps $t = 0, 1, 2, \ldots$. Some states, e.g. low power consumption with satisfactory QoS, are more favorable than others, so we want the agents to take actions so as to maximize the "favorableness " of the state. The method of *reinforcement learning (RL)* introduces *reward* as a numerical feedback given to an agent as a consequence of its last action. The agent then must possess some learning capacity to decide on the optimal action which is expected to maximize its reward, as well as some perception capacity to *observe* the current state of the environment to inform its decision. Although the reward system is not included in the simulation environment, we can integrate them together as a *learning* environment. Consequently the paradigm of reinforcement learning can be summarized as a continuous interaction of " observation $\rightarrow$ action $\rightarrow$ reward $\rightarrow$ observation " between the agent(s) and the environment, as illustrated in Figure 3.9.

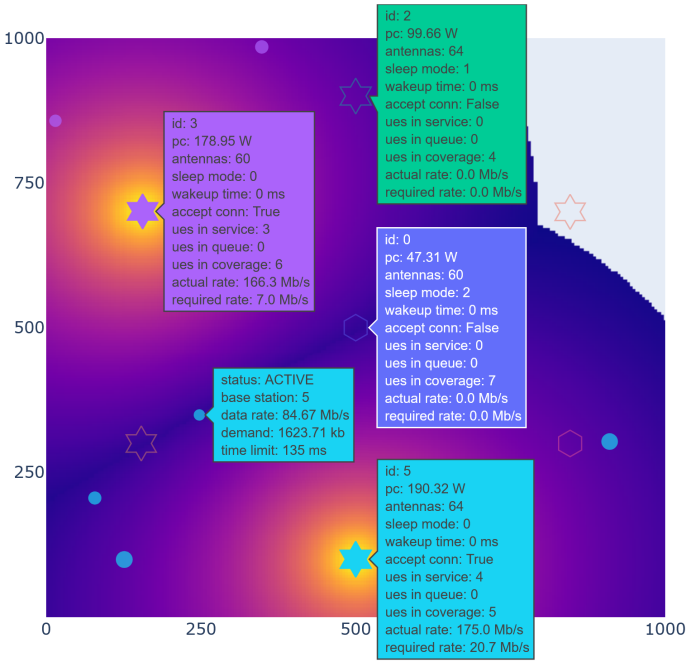


Figure 3.8: A snapshot of the multi-cell network environment.

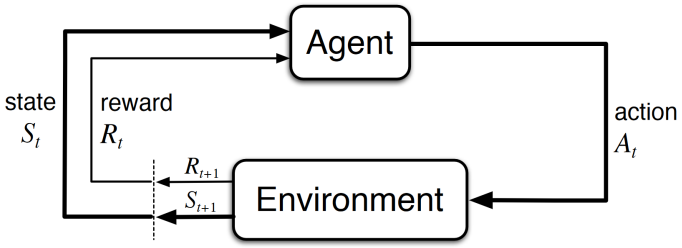


Figure 3.9: The agent-environment interaction in RL [4] .

3.2 Preliminaries on Reinforcement Learning

3.2.1 Space

The *state space* S of the environment is the set of all possible environment states.
The *action space* A of an agent is the set of actions it can take in any environment state.

3.2.2 Transition

An agent action $a \in A$ can change the state of the environment from s to s', which is called a *transition* (s, a, s').

3.2.3 Reward

After an agent takes an action a and the environment goes through a transition (s, a, s'), the reward system will give a reward $r = R(s, a, s')$ to the agent, where $R : S \times A \times S \to \mathbb{R}$ is

called the *reward model* .

3.2.4 Trajectory

The interaction between the environment and an agent for a duration of T can be expressed in a sequence of transitions and rewards is called a *trajectory* $(s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_{T-1}, a_{T-1}, r_{T-1}, s_T)$, where $r_t = R(s_t, a_t, s_{t+1})$. The trajectory space is thus $\Gamma_T = \Pi_{t=0}^{T} S \times R \times A$.

3.2.5 Dynamics and the Markov Property

The *dynamics* or *transition model* of an environment predicts the next state given the history trajectory, which can be described by a probability distribution $\mathbb{P}[s_{t+1} \mid s_0, a_0, \ldots, s_t, a_t]$ at any time t.

If a state s_t satisfies

$$\mathbb{P}[s_{t+1} \mid s_0, a_0, \ldots, s_t, a_t] = \mathbb{P}[s_{t+1} \mid s_t, a_t],$$

which means that given the current state and actions, the next state is *conditionally independent* of all previous actions and states, then s_t is said to have the *Markov property*, or simply *Markov*. If all states of an environment are Markov, then its dynamics is said to be a *Markov process*.

The dynamics of an RL environment is typically assumed to be a Markov process and denoted as $P(s' \mid s, a) = \mathbb{P}[s_{t+1} = s' \mid s_t = s, a_t = a]$.

3.2.6 Policy

The *policy* of an agent defines how an agent will act in a environment state. A *deterministic policy* is a direct mapping from state to action. A *stochastic policy* is a conditional probability distribution in the action space given a state. At any timestep t, a deterministic policy gives $a_t = \pi(s_t)$, while a stochastic policy gives $\mathbb{P}[a_t = a \mid s_t = s] = \pi(a \mid s)$.

3.2.7 Return and Value Functions

The *return* G_t is the sum of *discounted* rewards in all future steps after t:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k},$$

where $\gamma \in [0, 1]$ is called the *discount*. Generally γ needs to be less than 1 in order to ensure the convergence of the infinite series, as well as prioritizing immediate rewards over delayed rewards. Meanwhile a γ close to 1 encourages the agents to learn a "far-sighted" policy, i.e. to be considerate of a sufficiently long-term future.

Assuming the Markov property, the *value* of a state s is defined as the expected return

$$V_\pi(s) = \mathbb{E}_\pi[G_t \mid s_t = s].$$

The *value* of an action a in state s is then defined as

$$Q_\pi(s,a) = \mathbb{E}_\pi\left[G_t \mid s_t = s, a_t = a\right].$$

Note that the return, thus the value of a state or an action, is dependent on the policy.

3.2.8 Markov Decision Process

If the environment dynamics is a Markov process, then tuple (S, A, P, R, γ) defines a *Markov decision process (MDP),*

Under a certain policy π, an MDP allows us to predict the distribution of the next state $\mathbb{P}_\pi\left[s_{t+1} \mid s_t\right]$ given only the current state, and from s_{t+1} we can infer about s_{t+2}. Iterating this process, we can infer $\mathbb{P}_\pi\left[s_{t+k} \mid s_t\right]$ any time in the future, and consequently estimate the state value $V_\pi(s_t) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k}\right]$.

3.2.9 Optimal Value Functions

For a state s, its maximum value under all possible policies is called its *optimal value*

$$V^*(s) = \max_\pi V_\pi(s).$$

Similarly the optimal value of an action is

$$Q^*(s,s) = \max_\pi Q_\pi(s,a).$$

And they have the relation

$$V^*(s) = \max_{a\in\boldsymbol{A}} Q^*(s,a).$$

An MDP problem is said to be *solved* if an *optimal policy* π^* is found under which all states have their optimal value, i.e.

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

3.2.10 Bellman Equations

We can derive the relations between the state-value function V and the action-value function Q:

$$V_\pi(s) = \sum_{a\in\boldsymbol{A}} \pi(a \mid s)\, Q_\pi(s,a),$$
$$Q_\pi(s,a) = R(s) + \gamma \sum_{s'\in S} P(s' \mid s,a)\, V_\pi(s').$$

Inserting these equations into each other, we derive what called the *Bellman equations* for V and Q

$$V_\pi(s) = \mathbb{E}_\pi\left[r_t + \gamma V_\pi(s_{t+1}) \mid s_t = s\right], \tag{3.6}$$
$$Q_\pi(s,a) = \mathbb{E}_\pi\left[r_t + \gamma Q_\pi(s_{t+1}, a_{t+1}) \mid s_t = s, a_t = a\right]. \tag{3.7}$$

The Bellman equations for V^* and Q^* are

$$V^*(s) = R(s) + \gamma \max_{a \in \boldsymbol{A}} \sum_{s' \in S} P(s' \mid s, a) V^*(s'), \tag{3.8}$$

$$Q^*(s,a) = R(s) + \gamma \sum_{s' \in S} P(s' \mid s, a) \max_{a \in \boldsymbol{A}} Q^*(s,a). \tag{3.9}$$

If the state space S is finite and the transition model P and the reward model R are known to the agent, then the optimal action-value function Q^* can be found by *dynamic programming* methods such as *value iteration* [4], which in turn can directly yield the optimal deterministic policy $\pi^*(s) = \max_{a \in A} Q^*(s,a)$. However, this is usually not the case, so *model-free learning* methods have been developed to solve an MDP without its model.

3.3 Learning Methods

In model-free reinforcement learning, the agent has no knowledge of the model of the MDP, i.e. P and R. It learns by trial and error in the environment, collecting trajectories $s_0, a_0, r_0, s_1, a_1, r_1, \ldots$ and updating its policy based on these experience samples. If for an RL algorithm, the policy it is improving (called target policy) can differ from the policy (called behavioral policy) that produced the actions in the trajectory samples used to update the target policy, it is called *off-policy*. Otherwise, if the target policy and the behavioral policy are identical, the algorithm is called *on-policy*. An advantage of an off-policy learning algorithm is that it can use experience sampled under older-version policies to update the current policy, thus increasing sample efficiency, Q learning as an example. However, on-policy algorithms like SARSA and PPO, usually learn faster and with more stability.

3.3.1 Deep Neural Networks as Function Estimator

To be able to numerically model and update the policy function as well as its corresponding value functions, various machine learning models can be employed as function estimators. Among them, *deep neural network (DNN)* is one of the most popular and successful methods due to its differentiability high expressiveness produced by an arbitrarily large set of internal parameters in which can fit data patterns from the simplest to the most complicated.

The simplest form of a DNN is called *multilayer perceptron (MLP)*, which consists of *layers* of affine transformations (called fully-connected or dense layers) or non-linear functions (called activation layers). Each dense layer is parametrized by a *weight* matrix W and a *bias* vector b, and transforms its input X to output $XW + b$. A variety of activation functions are available for use to add non-linearity to the DNN, like ReLU (the most commonly used), tanh, SELU, etc.

To model the policy function π by a DNN, called the *policy network*, we use the SoftMax activation function as its output layer

$$\text{SoftMax}(x) = \left(\frac{\exp(x_1)}{\sum_{i=1}^{D} \exp(x_i)}, \ldots, \frac{\exp(x_D)}{\sum_{i=1}^{D} \exp(x_i)} \right),$$

where $x \in \mathbb{R}^D$ is its input value. Note that the output p of SoftMax satisfies $\sum_{i=1}^{D} p_i = 1$, so it

can model a probability distribution over D actions. For a multi-dimensional action space like ours, several output layers may be used in a row, each yielding the distribution in one action dimension. Letting θ be all the parameters in the DNN, the parametrized policy can be denoted as π_θ.

Due to the infinite state space, the value function V, if needed in the learning algorithm, has to be estimated by a model $\hat{V}$. Again we use a DNN as the estimator, called the *value network*. Letting ϕ be its parameter set, the estimated value function can be denoted as $\hat{V}_\phi$.

In *deep learning (DL)*, the optimization of DNN parameters, θ as an example, is converted to the minimization of a real-value differentiable *loss function* $L(\theta)$. In a DL framework like PyTorch or TensorFlow, the computation of $L(\theta)$ is accompanied with the recording of differentiable operations, which allows automatic gradient computation via the *back-propagation* mechanism [22]. After that, an optimizer like SGD or Adam [23], also provided in the DL framework, is used to update θ in the opposite direction of the gradient $\nabla_\theta L$ to minimize $L(\theta)$, namely *gradient descent*. A hyperparameter of the optimizer called *learning rate* is used to control the rate of update.

3.3.2 Value Estimation

Under a certain policy π, we want to improve $\hat{V}$ so that it gives more accurate estimations of V. From the Bellman Equation 3.6, we have

$$\mathbb{E}_\pi \left[r_t + \gamma V_\pi (s_{t+1}) - V_\pi (s_t) \right] = 0.$$

We define $\delta_t = r_t + \gamma \hat{V}_\pi (s_{t+1}) - \hat{V}_\pi (s_t)$, which is called the *temporal difference (TD)* at time t. It can proved that by minimizing the absolute TD, $\hat{V}_\pi$ will converge to the true value function V_π [4]. Therefore we can define a loss function, e.g. $L_V(\phi) = \mathbb{E}_\pi [\delta_t^2]$ (mean-squared error), to improve the DNN value estimator $\hat{V}_\phi$ by gradient descent with respect to $L_V(\phi)$.

3.3.3 Policy Gradient

For a parameterized policy π_θ, to optimize it towards an optimal policy π^* is equivalent to updating its parameters θ so as to maximize the objective

$$J(\theta) = \mathbb{E}_{\pi_\theta} [G_0] = \mathbb{E}_{\pi_\theta} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right],$$

i.e. the expected return at the start of the MDP. Its gradient, namely *policy gradient*, is

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta} \left[\nabla_\theta \log \pi_\theta (a \mid s) \, Q_{\pi_\theta} (s, a) \right].$$

Practically, the expectation is estimated by an empirical average obtained from sampled trajectories and $Q_{\pi_\theta}(s, a)$ is estimated by the returns, which is unbiased but typically has a large variance, leading to unstable learning. It can be shown that by introducing an *advantage* function

$$A(s, a) = Q(s, a) - V(s)$$

and replacing Q with A in the policy gradient, the sample estimate for $\nabla_\theta J$, still unbiased, can have much lower variance [24].

Provided with the estimation $\widehat{\nabla_\theta J}(\theta)$, the loss function $L_\pi(\theta)$ is defined such that $\nabla_\theta L_\pi = -\widehat{\nabla_\theta J}$. Consequently $J(\theta)$ can be maximized via gradient descent with respect to $L_\pi(\theta)$.

3.3.4 Generalized Advantage Estimation

As the value functions are unknown to the agent, the advantage must be estimated as well. One commonly used method called *generalized advantage estimation (GAE)* is defined as [24]

$$\hat{A}_t = \sum_{k=0}^{T-t} (\gamma\lambda)^k \delta_{t+k},$$

where δ_{t+k} is the temporal difference and the parameter λ controls the trade-off between bias and variance – if $\lambda = 0$, $\hat{A}_t$ is reduced to TD, which introduces bias to the estimation of $\nabla_\theta J$ unless $\hat{V}$ is perfectly accurate; while if $\lambda = 1$, the estimation of $\nabla_\theta J$ effectively becomes Monte Carlo and is unbiased regardless the choice of $\hat{V}$, but the sum of a long chain without decaying typically results in high variance. Empirically the best value of λ is lower than γ, somewhere in the range of $[0.9, 0.99]$, assuming a reasonable accuracy of the estimator $\hat{V}$ [24].

3.3.5 Actor-Critic Architecture

Consequently we have two DNN models in our RL system – π_θ, called the *actor*, and $\hat{V}_\phi$, called the *critic*.

Many state-of-the-art RL algorithms adopt the *actor-critic* architecture. Its learning procedure can summarized as: at a certain time interval, the agent samples a trajectory τ, uses τ to compute the policy loss $L_\pi(\theta)$ and the value loss $L_V(\phi)$, and update θ and ϕ using gradient descent. This procedure is also illustrated as the diagram in Figure 3.10.

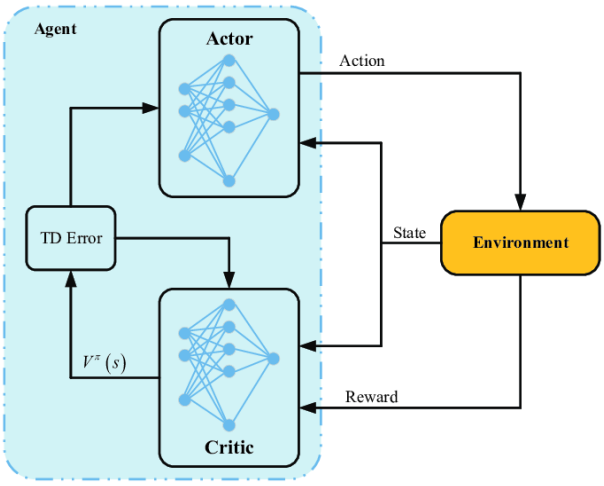


Figure 3.10: The actor-critic architecture (adopted from [5]).

3.3.6 Proximal Policy Optimization

When training the actor network using vanilla policy gradient, a large learning rate – the rate in which network parameters are updated in the direction of the policy gradient, is possible to result in a distinct change of the policy and considerably damage the policy performance. Thus the learning rate must be kept small, which on the other hand hurts its sample and training efficiency.

Proximal policy optimization (PPO) [25], motivated by *TRPO* [26], is a policy optimization method that aims to take the largest possible optimization step without pushing the new policy too far away from the old policy, by using a more sophisticated policy loss. The implementation of PPO commonly adopts the actor-critic architecture, so we also need to design a value loss. The policy loss and the value loss are defined as follows.

Policy Loss

In PPO, the objective $J(\theta)$ is estimated using importance sampling as

$$L^{\text{CPI}}(\theta) = \mathbb{E}_{\theta}\left[\hat{A}_t\right] = \mathbb{E}_{\theta_{\text{old}}}\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t\right],$$

which is called the "surrogate objective". CPI refers to conservative policy iteration [27] and $\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is called the probability ratio. Without constraint, maximization of L^{CPI} can cause extremely large update of the policy. Thus PPO seeks to constrain the policy update in simple and efficient ways. There are two variants of PPO: PPO-Penalty and PPO-Clip. PPO-Penalty penalizes the KL-divergence $KL(\pi_{\theta_{\text{old}}} \| \pi_\theta)$ to avoid large policy update, while PPO-Clip simply clips L_t^{CPI} as

$$\begin{aligned} L_t^{\text{CLIP}}(\theta) &= \min\left(\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t, \text{clip}\left(\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}, 1-\varepsilon, 1+\varepsilon\right)\hat{A}_t\right) \\ &= \min\left(\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t, \hat{A}_t + \varepsilon\left|\hat{A}_t\right|\right), \end{aligned}$$

where ε is a hyperparameter constraining how close the new policy should be to the old policy. We used PPO-Clip for our problem due to its simplicity and efficiency.

In addition to the clipped loss, an *entropy bonus* $c_e H(\pi_\theta \mid s_t) = c_e \mathbb{E}_{a\sim\pi_\theta}[-\log \pi_\theta(a \mid s_t)]$ is used to encourage a higher policy entropy or randomness, in order to ensure more exploration. The hyperparameter c_e is used to balance between exploration and exploitation.

Therefore the complete policy loss is defined as

$$L_\pi(\theta) = -\mathbb{E}_{\theta_{\text{old}}}\left[L_t^{\text{CLIP}}(\theta) + c_e H(\pi_\theta \mid s_t)\right].$$

Value Loss

As introduced in Section 3.3.2, we use TD to measure the value estimation error. Another loss function is required to reduce TD errors throughout the trajectory to a scalar value. Instead of mean-squared-error, to avoid large updates of the critic network, we use *Huber loss*

$$L_\epsilon^{\text{Hb}}(e) = \begin{cases} \frac{1}{2}e^2 & |e| \leq \epsilon, \\ \epsilon\left(|e| - \frac{1}{2}\epsilon\right) & |e| > \epsilon. \end{cases}$$

The value loss is then defined as

$$L_V(\phi) = \mathbb{E}_{\theta_{\text{old}}}\left[L_\epsilon^{\text{Hb}}\left(r_t + \gamma\hat{V}_\phi(s_{t+1}) - \hat{V}_\phi(s_t)\right)\right].$$

For $|e| \leq \epsilon$, L_ϵ^{Hb} is quadratic, otherwise linear. Hence the value loss will not grow too fast for a large TD error.

3.4 Multi-Agent PPO

Since there are multiple BS agents in the multi-cell network environment, and cooperative actions are required to achieve the desirable outcome of balancing between energy efficiency and service quality, the problem is formulated in a *multi-agent reinforcement learning (MARL)* framework

With some specific modifications, the PPO algorithm has demonstrated strong performance in cooperative multiagent games [28] like the Starcraft micromanagement challenge (SMAC) or Google Research Football [29].

3.4.1 DEC-POMDP

Let n be the number of agents in the network system and denote the agents as $b_1, b_2, \ldots, b_n$.

We define the *joint action space* $A = A_1 \times A_2 \times \ldots \times A_n$ as the set of joint actions $(a_1, a_2, \ldots, a_n)$. In our problem, all the BS agents have the same action space $\mathcal{A}$, as defined in Eq. 3.4, thus $A = \mathcal{A}^n$.

An action of a single agent can cause a state transition of the environment, but to simplify the problem, we assume every time all the agents act simultaneously, so each transition is produced by joint actions and is in the space $S \times A \times S$.

In reality, each BS agent can only observe a part of the whole environment, possibly with uncertainty. For example, a BS agent is only aware of UEs within its own cell coverage, and if it increases its output power to improve the QoS of its covered UEs, this may on the other hand creates more interference to those UEs outside of its coverage. Thus the best action based on the observation of a single agent is often not the optimal one in terms of the state of the whole environment.

Formally speaking, if the environment is in state s, an agent b_i only has access to its local observations o_i, which contains partial or uncertain information of s. The *observation space* O_i of agent b_i is the set of observations it can get from any state. We define the *joint observation space* $O = O_1 \times O_2 \times \ldots \times O_n$ as the set of joint observations $(o_1, o_2, \ldots, o_n)$.

If the environment is partially observable, then the observation space O and an observation model $Z(o \mid s) = \mathbb{P}[o \mid s]$ need be included in the MDP framework to define a *partially observable MDP (POMDP)*. The framework can be further extended to a *decentralized POMDP (DEC-POMDP)* when there are n agents acting in the environment without a centralized

controller,. To simplify the problem, we assume the observation model is deterministic, i.e. without uncertainty. Consequently Z becomes a direct mapping $S \rightarrow O$ such that the joint observations in state s is $(o_1, \ldots, o_n) = Z(s)$.

Since in our multi-cell network environment, we are only concerned with the global state of the network and we have homogeneous agents, we assume the same reward model R for all agents, only dependent on the new state after transition, i.e. $R_i(s, a_i, s') = R(s')$ for $i = 1, 2, \ldots, n$. Therefore after each joint actions, all agents share the same reward.

In summary, a DEC-POMDP is defined by a tuple $(n, S, O, A, P, Z, R, \gamma)$, where S is the state space, $O = O_1 \times \ldots \times O_n$ the joint observation space, $A = A_1 \times \ldots \times A_n$ the joint action space, $P(s' \mid s, a) = \mathbb{P}[s_{t+1} = s' \mid s_t = s, a_t = a]$ the transition model, $Z(o \mid s) = \Pi_{i=1}^{n} Z_i(o_i \mid s)$ the joint observation model, $R : S \times A \times S \rightarrow \mathbb{R}^n$ the reward model, n the number of agents, and γ the discount parameter.

Our RL problem is an instance of DEC-POMDP with two modifications: shared reward $R : S \rightarrow \mathbb{R}$ and deterministic observation model $Z : S \rightarrow O$.

3.4.2 Policy

We define the *joint policy* of all agents as

$$\pi(a \mid o) = \prod_{i=1}^{n} \pi_i(a_i \mid o_i), a \in A, o \in O.$$

3.4.3 Learning in DEC-POMDP

The trajectory is extended to include the observations

$$\tau = (s_0, o_0, a_0, r_0, \ldots, s_{T-1}, o_{T-1}, a_{T-1}, r_{T-1}, s_T),$$

where $o_t = (o_{t,1}, \ldots, o_{t,n})$ is the joint observations and $a_t = (a_{t,1}, \ldots, a_{t,n})$ is the joint actions. In MARL, for one agent, the rest of agents are part of the environment. To optimize its policy, the agent must develop some knowledge about the environment. However, as all agents are updating their policies, the environment dynamics for a single agent is also changing, which can detriment or even prohibit the convergence of training. This is called the *non-stationarity* problem of MARL.

Since the agents are homogeneous, we use a single policy network π_θ for all agents.This can reduce the *non-stationarity* of the environment since the policies of other agents as part of the environment have lower variance. Hence $a_{t,i} = \pi_\theta(o_{t,i})$. We also denote $a_t = \pi_\theta(o_t)$ for convenience. Assuming shared policy and shared reward, the DEC-POMDP is in effect reduced to a POMDP, whose action space is all agents' joint action space and observation space is their joint observation space.

The *centralized training and decentralized execution (CTDE)* approach can help tackle the non-stationarity problem. In an actor-critic architecture, the training is centralized and since the critic has access to the global state of the environment as well as local observations of all agents, the dynamics of the environment is stationary, which allows it to give stable evaluations of the states under the agents' policies, which in turn helps stabilize the policy gradients. After the

training, the critic is no longer needed and each agent can execute its learned policy in a fully decentralized manner.

3.4.4 Value Normalization

To stabilize value learning, we standardize the targets of the value function by using running estimates of the average and Standarddeviation of the value targets. This stabilizes the value targets, which, without normalization, can drastically change during training. Concretely, during value learning, the value network will regress to the normalized target values. When computing the GAE, we will use the running average to denormalize the output of the value network so that the value outputs are properly scaled. We find that using value normalization never hurts training and often significantly improves the final performance of MAPPO.

3.4.5 Input Representation to Value Function

For multi-agent CTDE RL algorithms, it is critical that the input to the value network accurately represents the state of the environment. Otherwise it becomes very difficult, if not impossible, for the policy to converge to an optimum. The implementation of the state representation commonly follows two approaches: concatenation of all local observation vectors, and a global state vector provided by the environment, both of which turn out to be necessary for the learning of the value network [28]. Therefore we represent the environment state as a concatenation of all local observations as well as some global information not observed by any agent. The detailed construction of the state representation is given in Section 3.5.2.

3.4.6 Sample Reuse and Mini-Batches

Due to the importance sampling method in the policy loss of PPO, the same trajectory sample, called a *batch* of training data, can be used for updating the policy network for multiple times, without introducing excessive off-policy bias. The batch can also be splitted into several mini-batches to further multiply the number of updates for the same amount of training data, thus increasing sample efficiency. However, It was found in [28] that too many updates per batch degrades the performance of MAPPO, possibly because of the non-stationary environment.

3.4.7 Algorithm

Algorithm 2 Multi-Agent Proximal Policy Optimization

Input: initial actor parameters θ, initial critic parameters ϕ, number of episodes N, trajectory length T, updates per epoch M, GAE parameter λ
for k = 0, 1, ..., N do

Record a trajectory $\tau =$ $(s_0, o_0, a_0, r_0, \ldots, s_{T-1}, o_{T-1}, a_{T-1}, r_{T-1}, s_T)$ under the policy π_θ
Compute TD errors $\delta_t = r_t + \gamma \hat{V}_\phi(s_{t+1}) - \hat{V}_\phi(s_t),\ 0 \le t < T$
Estimate advantages $\hat{A}_t = \sum_{i=0}^{T-t} (\gamma\lambda)^i \delta_t,\ 0 \le t < T$
$\pi_{\text{old}} \leftarrow \pi_\theta$
for j = 0, 1, ..., M do

$L_\pi(\theta) = -\frac{1}{T}\sum_{t=0}^{T-1}\left[\min\left(\frac{\pi_\theta(a_t|o_t)}{\pi_{\text{old}}(a_t|o_t)}\hat{A}_t, \hat{A}_t + \epsilon\left|\hat{A}_t\right|\right) - c_e H(\pi_\theta \mid s_t)\right]$
Update θ using gradient descent w.r.t. $L_\pi(\theta)$
$L_V(\phi) = \frac{1}{T}\sum_{t=0}^{T-1} L_\delta^{\text{Hb}}(\delta_t)$
Update ϕ using gradient descent w.r.t. $L_V(\phi)$

end for

end for

3.5 Implementation

3.5.1 Reward Design

Suppose during an action interval, the average power consumption of the network is p kW, and a total of n UEs have quit the environment, which means either their traffic demand has been finished or the service delay has reached their delay budget. For a UE u_k among them, let $r_{\text{avg},k}$ be its average data rate, i.e. $\frac{x_{\max,k}-\chi_k}{\tau_k}$, $r_{\text{req},k}$ be its required data rate of UE u_k, i.e. $\frac{x_{\max,k}}{\tau_{\max,k}}$, and ρ_k be their ratio $\frac{r_{\text{avg},k}}{r_{\text{req},k}}$. We design the reward function as

$$
\begin{aligned}
R &= w_{\text{qos}}\xi - w_{\text{pc}}p \\
\xi &= \frac{1}{n}\sum_{k=1}^{n}\xi_k = \frac{1}{n}\sum_{k=1}^{n}\begin{cases}\rho_k - 1 & \rho_k < 1, \\ \phi\left(1 - \frac{1}{\rho_k}\right) & \rho_k \ge 1.\end{cases}
\end{aligned}
$$

ξ_k is the reward for the quality of service experienced by UE u_k. When dropped, $r_{\text{avg},k} < r_{\text{req},k}$ and $\rho_k < 1$, so $\xi_k = \rho_k - 1$ is a negative value as penalty, whose *magnitude* in this case equals $\frac{\chi_k}{x_{\max,k}}$, *the ratio of dropped data*. When the request is finished, $r_{\text{avg},k} \ge r_{\text{req},k}$ and $\rho_k \ge 1$, so $\xi_k = \phi\left(1 - \frac{1}{\rho_k}\right)$ is a non-negative reward. Note that in this case $\frac{1}{\rho_k} = \frac{\tau_k}{\tau_{\max,k}}$, which can be interpreted as *the ratio of delay to its budget*. The relation between ξ_k and ρ_k for different values of the parameter ϕ is shown in Figure 3.11.

When the service completely failed the demand, $\xi_k = -1$; when it finished the total demand "just in time ", $\xi_k = 0$, thus no reward nor penalty; while if its data rate was much higher than required, there is an upper bound of reward, which is ϕ. Generally we want to set ϕ to a small value because in comparison to those demands failed to be finished in time and thus dropped, the data rates of those services finished within the delay budget are much less critical to the

QoS. Finally, w_{pc} and w_{qos} are parameters used to balance between the QoS reward and the PC penalty in the reward. We can also neglect the data rate of a service as long as it finishes before the delay budget by setting $\phi = 0$, but provided that other conditions (PC, drop rate, etc.) are the same, we may still want to encourage the BSs to finish the requests sooner.

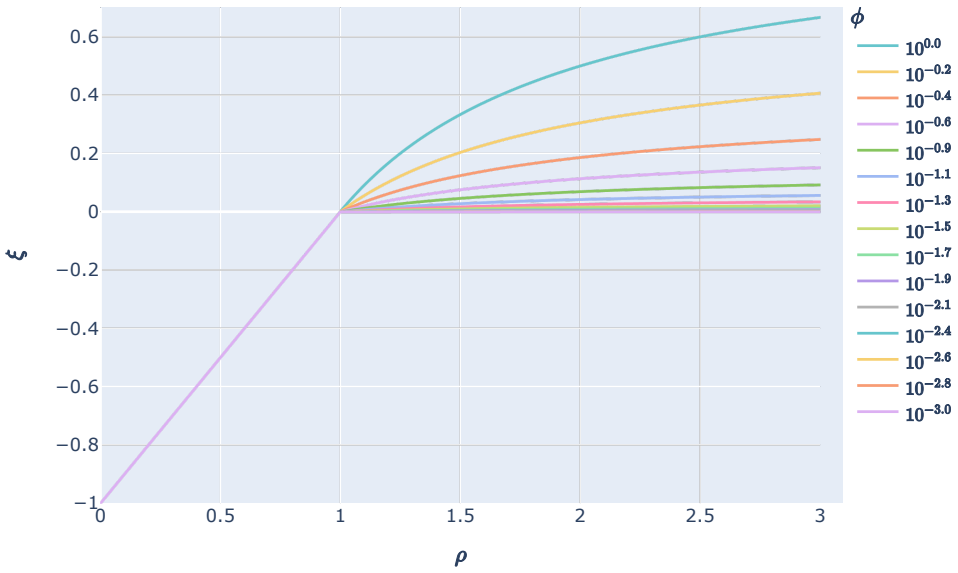


Figure 3.11: QoS reward ξ versus the data rate ratio ρ for different values of ϕ.

Parameter	Value
w_{qos}	4
w_{pc}	1
ϕ	0.005

Table 3.6: Parameters in the reward function.

According to the parameter values, the BS agents in the network are willing to spend 25W more PC in total to reduce average drop ratio by 1% or 1W more PC to decrease the average delay by 10% of the delay budget.

3.5.2 Observation and State Representations

For each BS agent b_c, its observation is represented by a vector obs $\in \mathbb{R}^{116}$:

obs[0] Power consumption.

obs[1] Number of antennas.

obs[2] Whether accepting new requests.

obs[3...6] Sleep level 0, 1, 2, 3 (binary encoding: 1 for in this level and 0 for not)

obs[7...10] Next sleep level 0, 1, 2, 3 (the same as **obs[3...6]** if not switching sleep level).

obs[11] Remaining delay to switch to the next sleep level.

obs[12...16] Average traffic demand rate in the signal coverage each minute, for the last 5 minutes.

Let U_0 = UEs in the cell coverage of b_c, U_1 = UEs being served by b_c, U_2 = UEs in the queue of b_c.
For a set U of UEs, define UEsInfo (U) as the vector (size of U, current sum rate of U, required sum rate of U, sum of transmit powers of U, number of UEs in U whose time limit is no more than a threshold (20ms)).
For i in [0,1,2]:

obs[17+5i...21+5i] UEsInfo (U_i).

For j in [0,1,2,3,4,5], let b'_j be the j-th neighbor BS of b_c, and U'_j be the set of UEs in the signal coverage of b'_j.

obs[32+14j] Power consumption of b'_j in W.

obs[33+14j] Number of antennas of b'_j.

obs[34+14j] Whether b'_j is accepting new requests.

obs[35+14j...38+14j] Sleep level of b'_j.

obs[39+14j] Distance between b_c and b'_j.

obs[40+j] Average traffic demand rate in the signal coverage of b'_j during the last minute.

obs[41+14j...45+14j] UEsInfo $\left(U'_j\right)$.

Combining all the agent observations, plus some global information, we assume it can completely describe the environment state. Thus network environment is described by a vector state $\in \mathbb{R}^{821}$:

state[0] Total power consumption in kW.
state[1] Number of finished UE requests in the last action interval.
state[2] Average of (average data rate / required data rate) for finished UE requests in the last action interval.
state[3] Number of dropped UE requests in the last action interval.
state[4] Average of (average data rate / required data rate) for dropped UE requests in the last action interval.
state[5] Required sum rate of all idle UEs.
state[6] Required sum rate of all queued UEs.
state[7] Required sum rate of all UEs being served.
state[8] Current sum rate of all UEs.
state[9+116i...124+116i] The observation of BS b_i, for $i = 0, 1, \ldots, 6$.

3.5.3 Neural Network Structure

The actor-critic architecture of the MAPPO algorithm is shown in Figure 3.12. The network at the left is the critic network and that at the right is the actor network. Both the actor and the critic are MLP networks. The dimensionality of each layer in the network is also shown by the numbers surrounded by the angle brackets. The input to the critic network is a vector s_t representing the current global environment state, while the input to the actor network is a batch of vectors o_t representing the local observations of all 7 BS agents. The output of the critic is a scalar value $\hat{V}_\phi(s_t)$ estimating the state value, and is used in the calculation of the TD error δ_t. The output of the actor is 3 batches of vectors, giving the joint action probability distributions $\pi_\theta(\cdot \mid o_t)$ of all agents.

Following the common practices adopted in PPO implementations, we apply normalization to the state vector, the observation vectors, the output of each hidden layer, and the estimated advantages, and we use ReLU activation with orthogonal initialization [28]. These techniques have been shown helpful to stabilize training and improve performance.

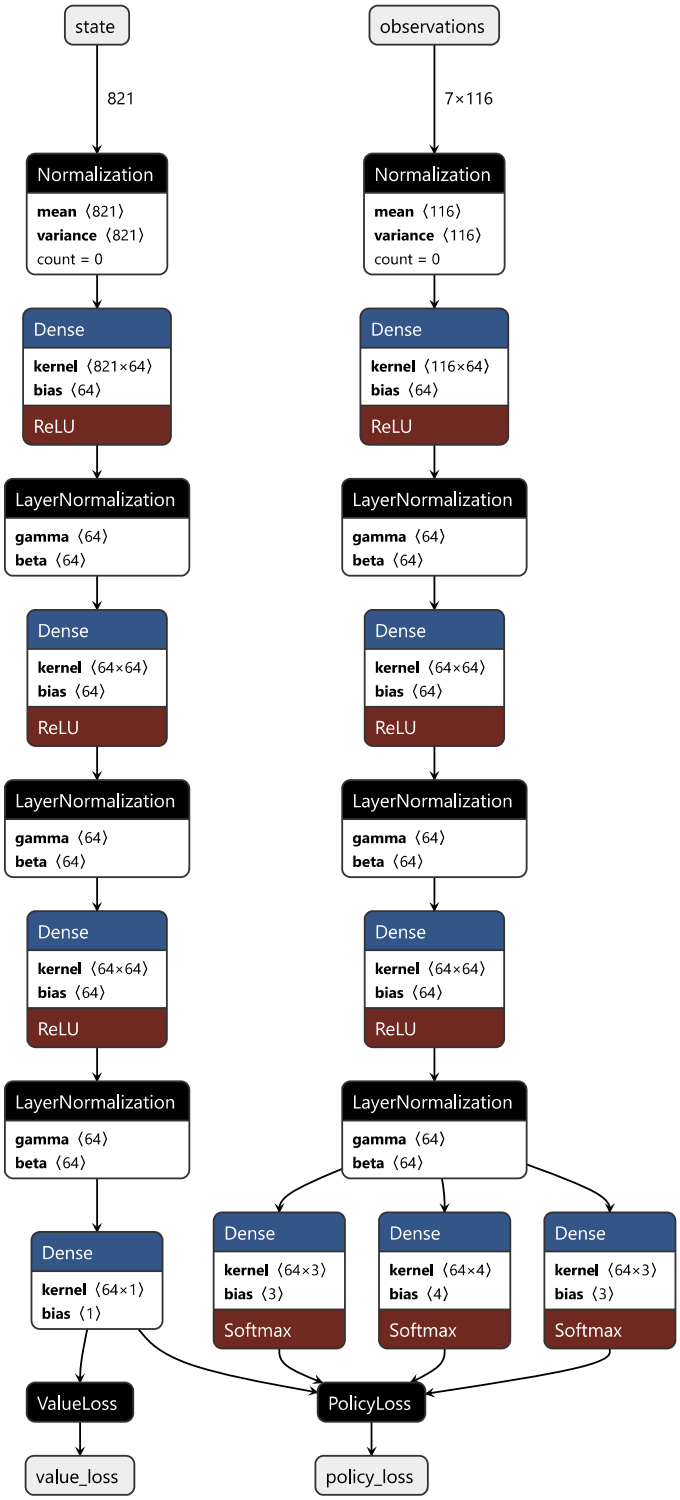


Figure 3.12: Network structure of MAPPO agent.

3.5.4 Training

The batch size T, i.e. the length of trajectories used for MAPPO learning, can have a flexible range of values, but another common practice in PPO implementations is to use a large batch size [28]. Therefore, we let the batch size equal to the episode length, so that the network learns

from trajectories of a full week length. Parallel computation of 42 threads is applied to speed up simulation and training, and after each trajectory has been gathered, it is used for training the network for 10 passes (epochs). Due to the reason given in Section 3.4.6, instead of splitting the trajectory into mini-batches, the training is done via full-batch gradient descent. The whole training ends after 100 episodes of simulation.

Since there are 3 traffic scenarios, we experiment with 4 kinds of simulation strategies during training – A, B, C, and RANDOM. For the first three, all episodes of simulation use the same traffic scenario, but for the last one, each episode randomly chooses a traffic scenario for simulation.

Parameter	Value
Agent timestep	20 ms
Batch size T	50400 steps = 1 episode
Training episodes N	100
Epochs per episode M	10
Discount factor γ	0.99
Actor learning rate η_π	6e-4
Critic learning rate η_v	5e-4
Number of mini-batches	1
Clip parameter ε	0.2
GAE parameter λ	0.95
Entropy coefficient c_e	0.01
Huber loss parameter ϵ	10

Table 3.7: Parameters of MAPPO learning.

3.6 Results and Analysis

The trained MAPPO policy is tested in the time-stepping-based network simulation environment. It shows its ability to adapt the number of antennas, the different sleep modes, and the user association of the base stations according to the varying network traffic density throughout a week to minimize the total energy consumption while preserving a good quality of service. A certain degree of cooperation has been shown between the BSs, reducing the inter-cell interference by putting part of the BSs to sleep. The performance of the multi-agent BS control policy, including energy efficiency and QoS, is compared with several baseline policies, including an always-on policy, a simple sleeping policy in which any BS with no associated users will automatically switch to the shallowest sleep mode (SM1), and an RL policy trained on individual agents without multi-agent cooperation.

3.6.1 Training Performance

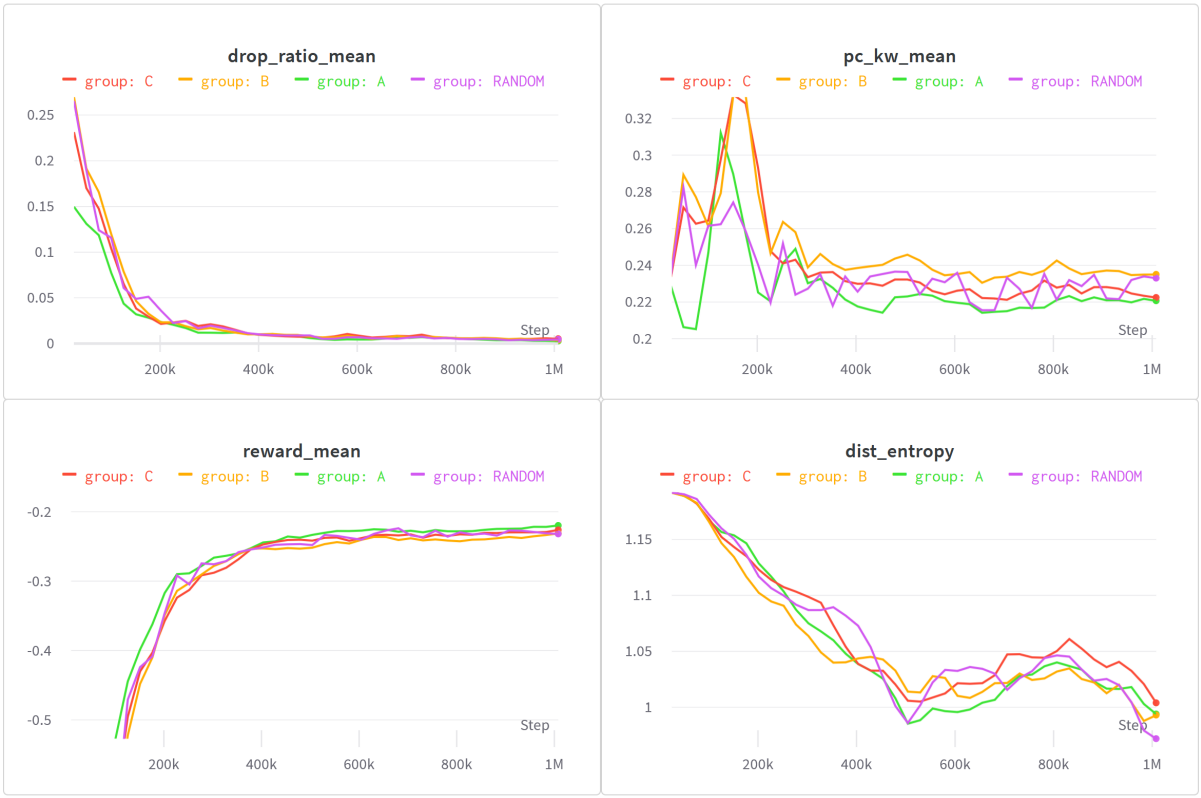
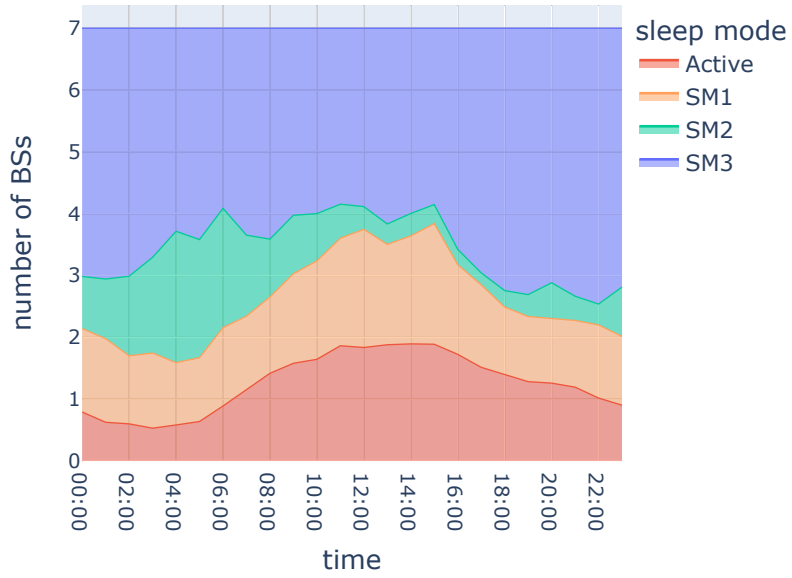


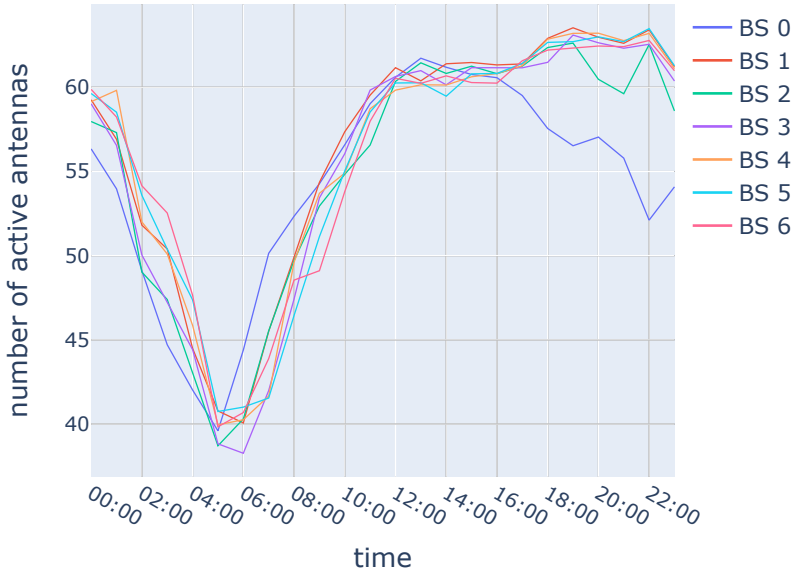
Figure 3.13: Training performance curves for different simulation strategies. The x-axes are the number of simulation steps during training. The y-axes are the average values of "packet drop ratio", "power consumption (kW)", "reward", and "entropy of policy distribution", respectively.

Four different simulation strategies were applied during training: A (rural), B (urban), C (office), and RANDOM. For strategies A, B, and C, the single corresponding traffic scenario was simulated in all episodes throughout training. For the RANDOM strategy, a random traffic scenario was chosen for each episode for simulation. We can see from the reward curves in Figure **??** that for all of these strategies, the MAPPO agent has succeeded in improving its reward until the reward has converged. Throughout the training, the drop ratio keeps decreasing and converges to a very low value. The power consumption increases at the beginning of the training to improve QoS or lower packet drop, but then decreases as the agent learns to save more energy while preserving a good QoS. It is worth emphasizing that the training performance of the RANDOM simulation strategy is as good as other strategies with a single traffic scenario. Therefore, a single policy trained in this RANDOM simulation strategy suffices to adapt to any of these traffic scenarios during its deployment.

3.6.2 Policy Visualization



(a) Number of BSs in different active/sleep modes.



(b) Number of antennas per BS.

Figure 3.14: Visualization of the daily decisions of the trained MAPPO policy.

We can see from Figure 3.14 that the MAPPO policy is able to adjust the sleep modes and the number of antennas of the BSs according to the temporally varying network traffic demand. During daytime, when there is a higher traffic demand, more BSs are active or in shallower sleep modes, and almost all of the antennas of each BS are turned on. During nighttime, when the traffic is less busy, the BSs sleep deeper with less antennas turned on.

3.6.3 Performance Metrics

To quantitatively measure the performance of the BS control policy, we collect relevant statistics during the simulation and calculate several performance metrics of the energy saving and the QoS.

We use *energy efficiency (EE)* as the metric to measure the energy saving performance of the BS control policy

$$\text{EE} = \frac{\text{total traffic volume}}{\text{total energy consumption}} \,[\text{kb/J}]\,.$$

Since the total traffic volume in our network simulation is approximately constant, the EE is in reverse proportion to the total energy consumption.

We use *drop ratio* as the metric of the QoS, defined as

$$\text{DR} = \frac{\text{total data size of dropped traffic}}{\text{total data size of requested traffic}},$$

Recall that if the requested traffic of a user is not served within its delay budget, the remaining traffic demand will be dropped.

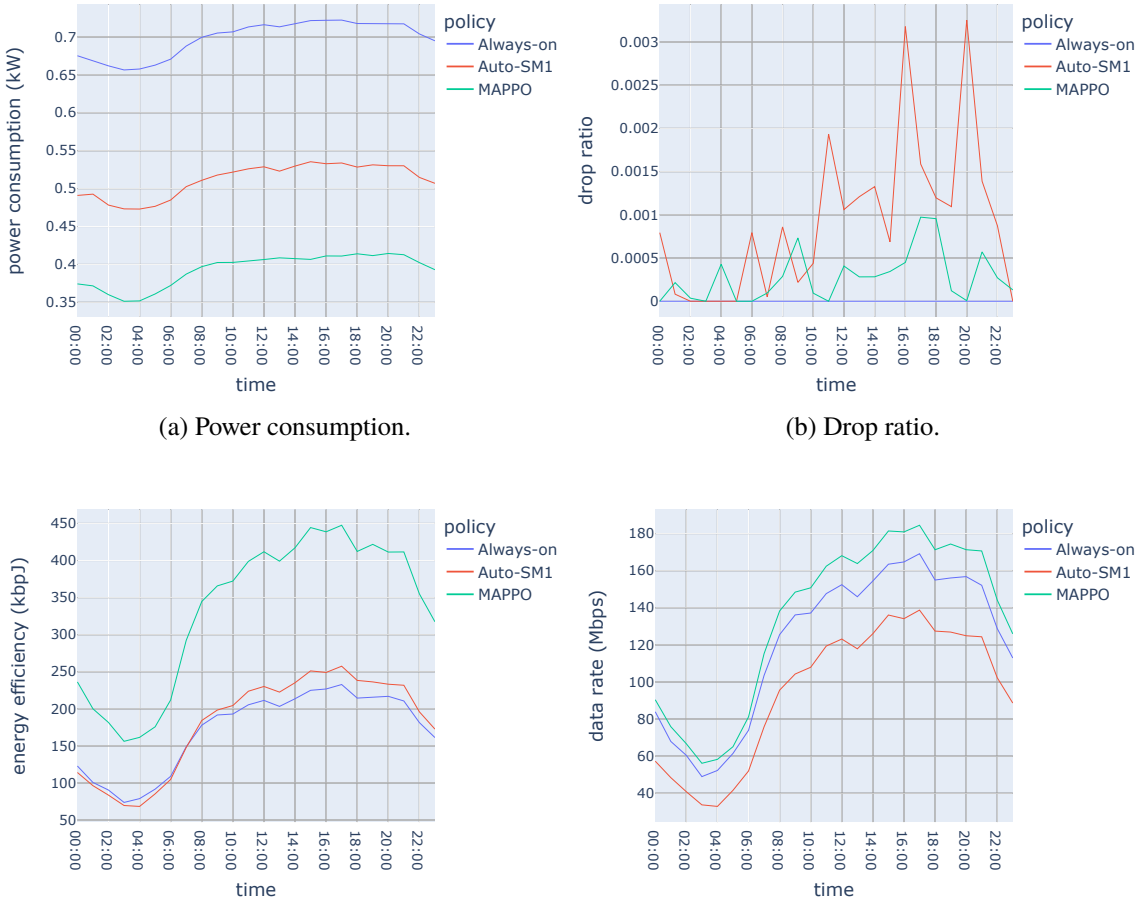
3.6.4 Overall Performance Statistics

The numeric values of the overall average performance metrics are also shown in Table 3.8.

policy	scenario	avg PC (W)	avg rate (Mb/s)	EE (Mb/J)	avg drop (%)	energy saving (%)
Always On	A	667.263407	84.364559	0.208653	0.000000	0.000000
	B	698.308698	82.946281	0.360139	0.000000	0.000000
	C	688.036811	82.992174	0.156469	0.000000	0.000000
Auto SM1	A	482.460464	49.728549	0.288529	0.023498	27.695651
	B	512.114219	48.986153	0.491005	0.045924	26.663634
	C	501.801536	49.075359	0.214539	0.059078	27.067632
MAPPO ($w_{qos} = 1.0$)	A	279.277140	51.035882	0.743922	0.192087	58.145893
	B	295.800081	47.639107	0.850045	0.220333	57.640499
	C	290.924319	47.317153	0.742144	0.131457	57.716751
MAPPO ($w_{qos} = 4.0$)	A	311.165711	43.827244	0.447362	0.261221	53.366885
	B	327.868221	40.571553	0.766904	0.124714	53.048240
	C	332.093293	40.618032	0.650142	0.179455	51.733209
MAPPO ($w_{qos} = 7.0$)	A	364.979143	40.974922	0.381361	0.023702	45.302089
	B	392.604126	41.428531	0.640431	0.013646	43.777855
	C	386.723861	41.254836	0.558299	0.035535	43.793144
MAPPO (ignore interference)	A	287.446027	38.961543	0.484227	0.529351	56.921656
	B	338.758773	40.055617	0.370466	0.242728	51.488679
	C	307.525363	39.185893	0.350073	0.157113	55.303937
MAPPO (no offloading)	A	368.197267	47.580268	0.378069	0.754334	44.819802
	B	423.042989	52.810545	0.296639	0.419425	39.418915
	C	409.894357	52.993895	0.526704	0.382794	40.425519

Table 3.8: Performance statistics of different policies. The "energy saving" in the last column uses "Always On" policy as the reference. Unless specified, the default value of w_{qos} for a MAPPO policy is 4.

3.6.5 Comparison with Baselines



(a) Power consumption.

(b) Drop ratio.

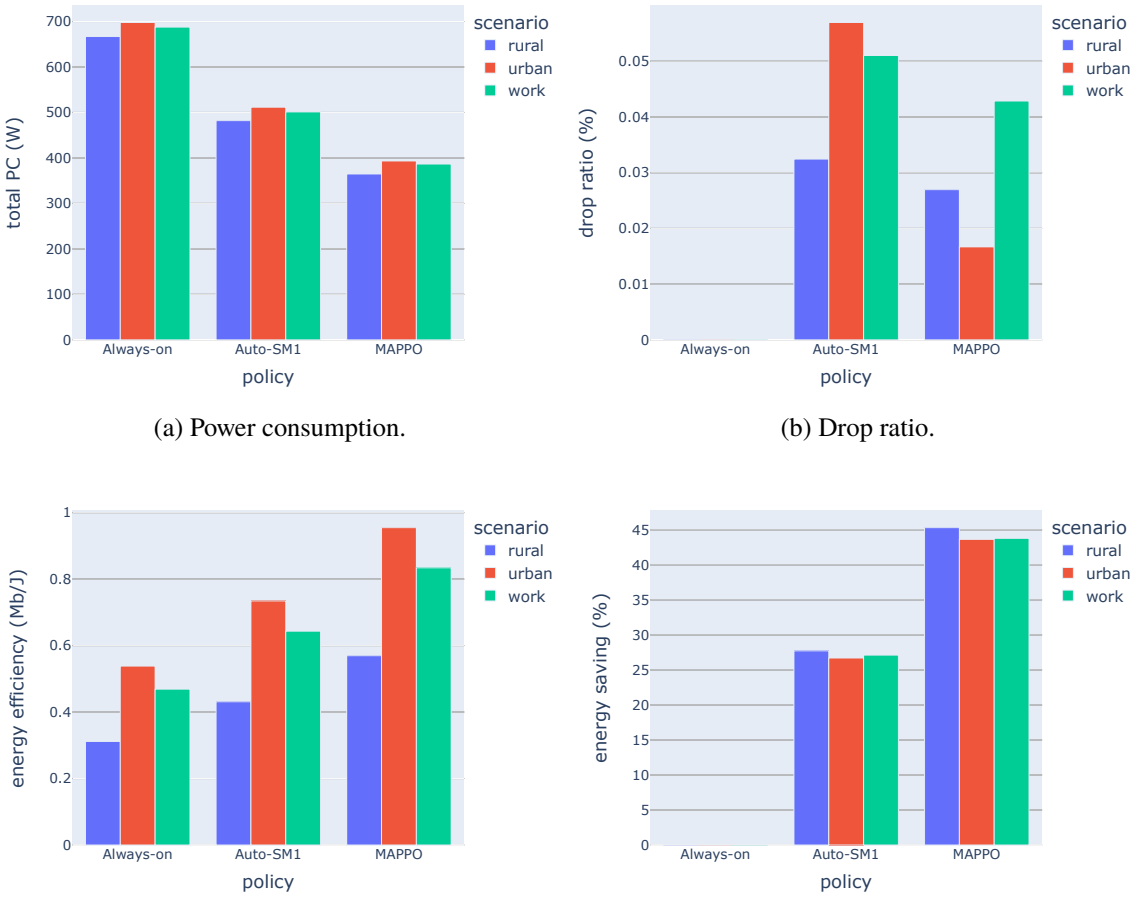
(c) Energy efficiency.

(d) Network throughput.

Figure 3.15: Comparisons of daily performance in the urban scenario between a trained MAPPO policy and baseline policies.

Two baseline policies are developed for performance benchmarking. The always-on policy simply keeps all the BSs always active with all antennas turned on; the auto-SM1 policy turns any BS with no associated users into SM1, and wakes up a sleeping BS as soon as a user enters its signal coverage. The temporal variations of the performance metrics are shown in Figure 3.15 and the overall average of these metrics are shown in Figure 3.16. For the MAPPO policy, we set the default value of its hyper-parameter w_{qos} as 4.0, because it produces the best trade-off between energy saving and quality of service.

We can see that the auto-SM1 policy can already save a significant amount of energy, but the MAPPO policy is even more energy efficient, introducing a negligible amount of packet drop.



(a) Power consumption.

(b) Drop ratio.

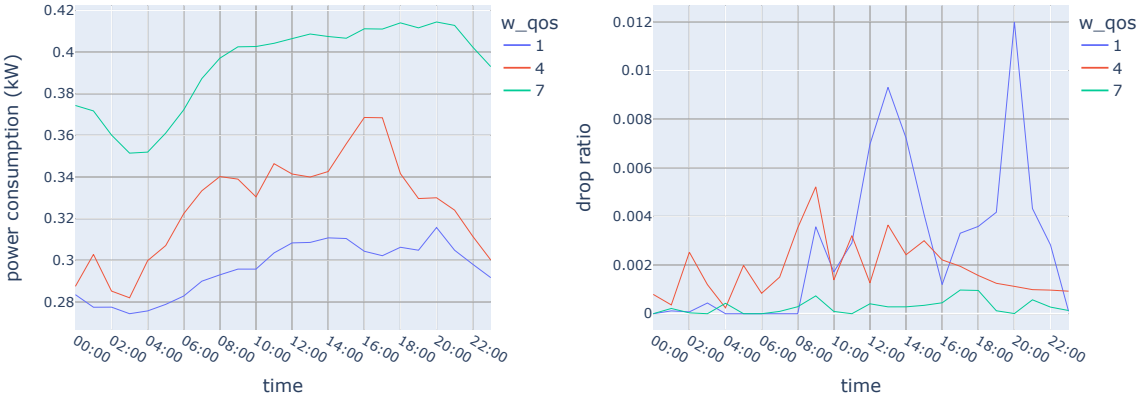
(c) Energy efficiency.

(d) Energy saving (w.r.t. "always-on" policy).

Figure 3.16: Comparisons of overall performance between a trained MAPPO policy and baseline policies.

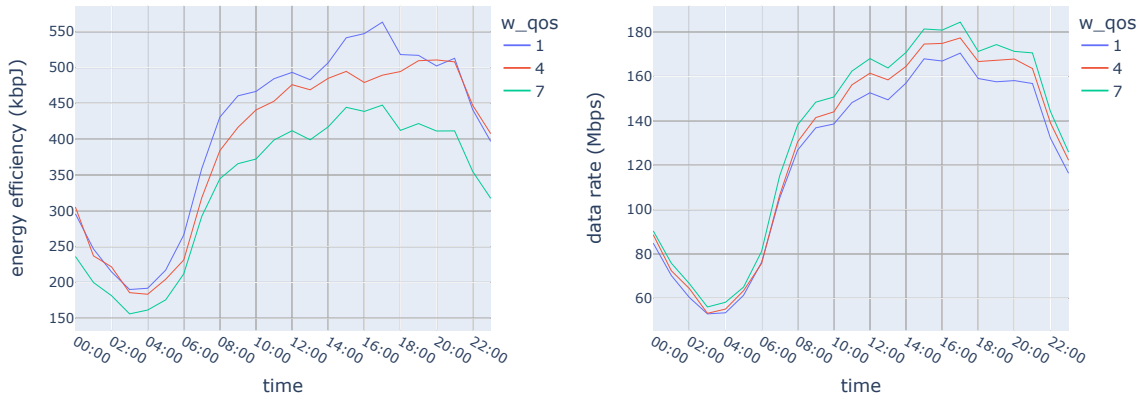
3.6.6 Effect of Parameter w_{qos}

Keeping other hyper-parameters the same, we vary the value of w_{qos} in the reward function of MAPPO to investigate its effect on the energy efficiency and the QoS. As shown in Figure 3.17 and 3.18, a higher w_{qos} enforces less packet drop and higher network throughput (sum rate) and QoS. This discourages BS sleeping, thus considerably increasing power consumption and reducing energy saving. Although a higher w_qos increases the throughput, the improvement is not significant, so the energy efficiency still decreases.



(a) Power consumption.

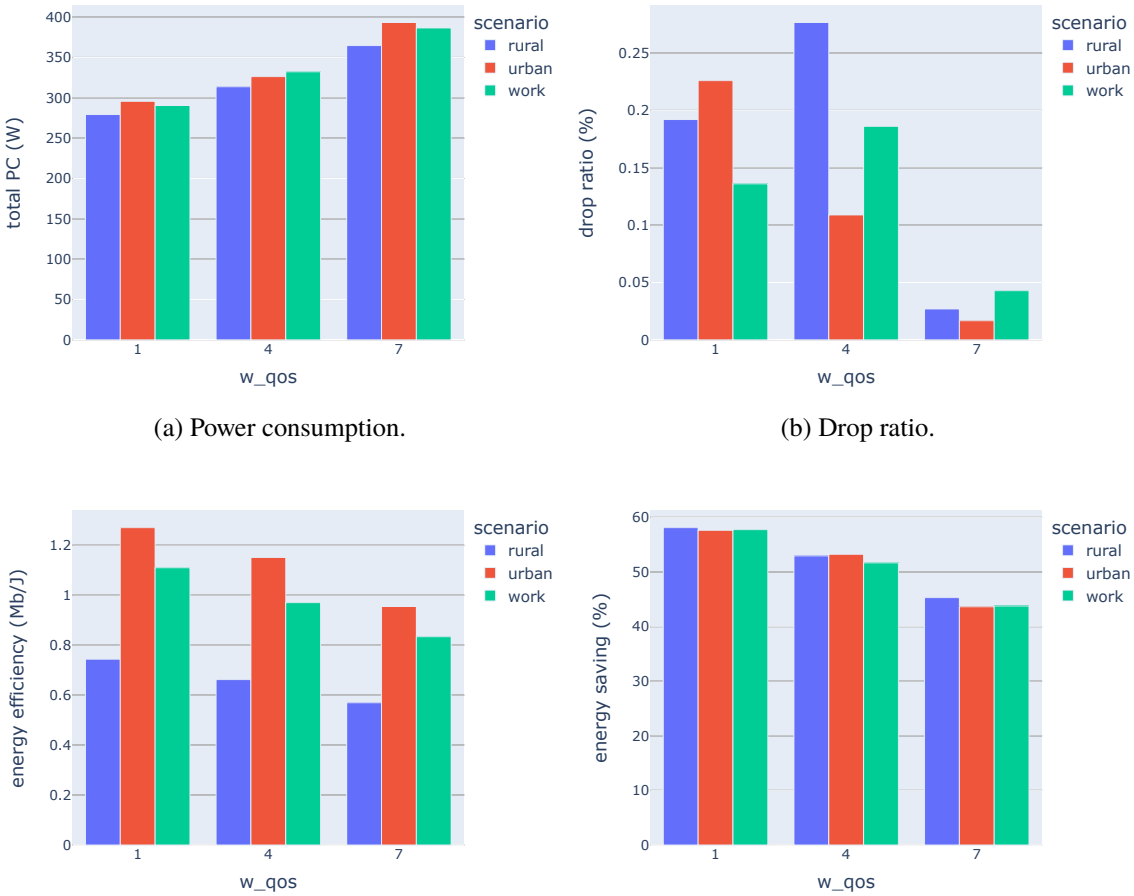
(b) Drop ratio.



(c) Energy efficiency.

(d) Network throughput.

Figure 3.17: Comparisons of daily performance in the urban scenario among MAPPO policies with different values of w_{qos} in the reward.



(a) Power consumption.

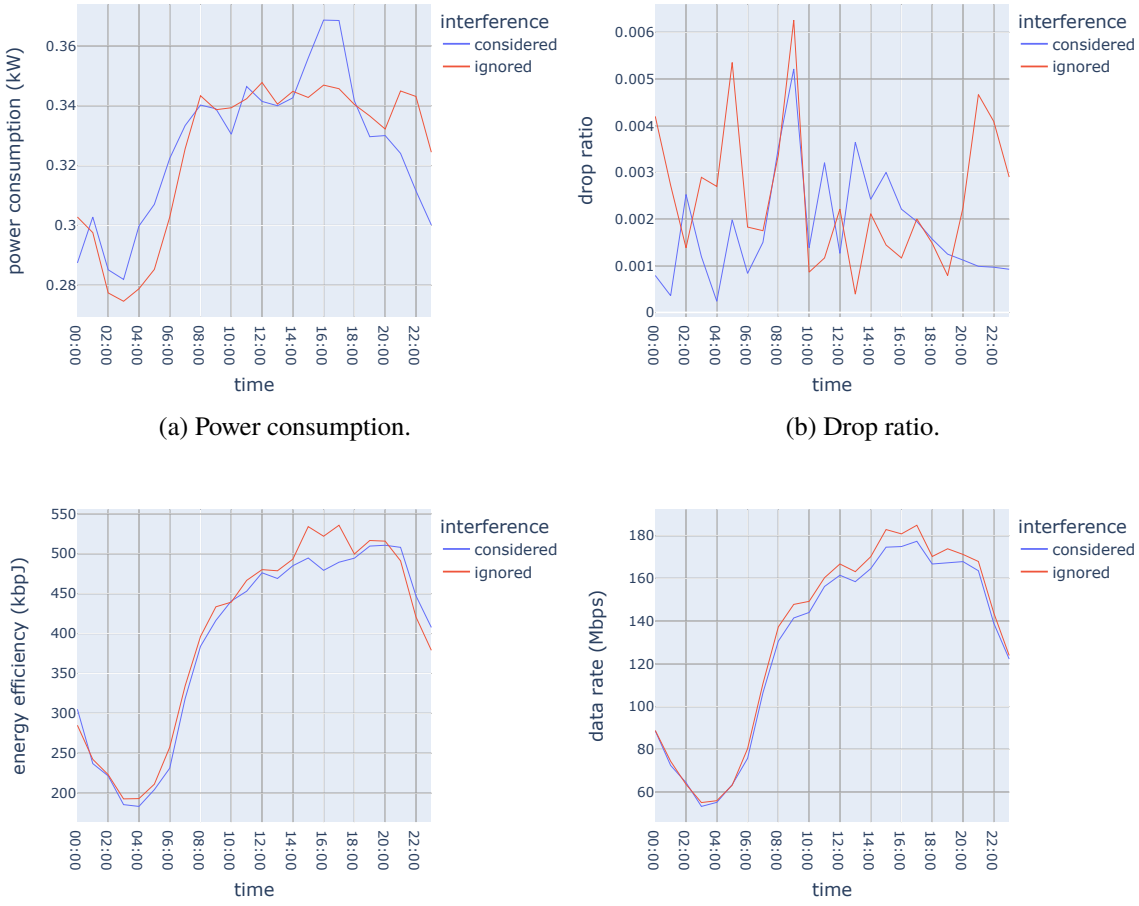
(b) Drop ratio.

(c) Energy efficiency.

(d) Energy saving (w.r.t. "always-on" policy).

Figure 3.18: Comparisons of overall performance among MAPPO policies with different values of w_{qos} in the reward.

3.6.7 Consideration of Interference



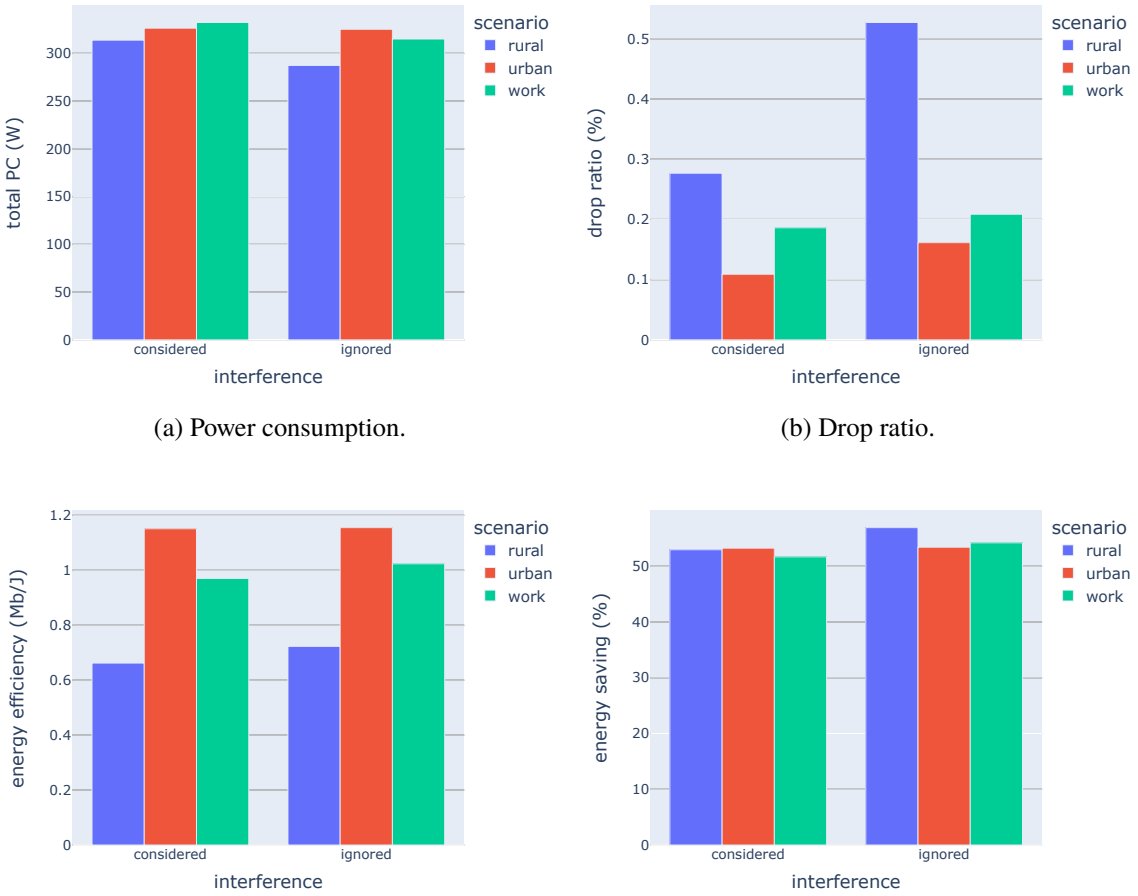
(a) Power consumption.

(b) Drop ratio.

(c) Energy efficiency.

(d) Network throughput.

Figure 3.19: Comparisons of daily performance in the urban scenario between two MAPPO policies trained with and without consideration of the inter-cell interference.



(a) Power consumption.

(b) Drop ratio.

(c) Energy efficiency.

(d) Energy saving (w.r.t. "always-on" policy).

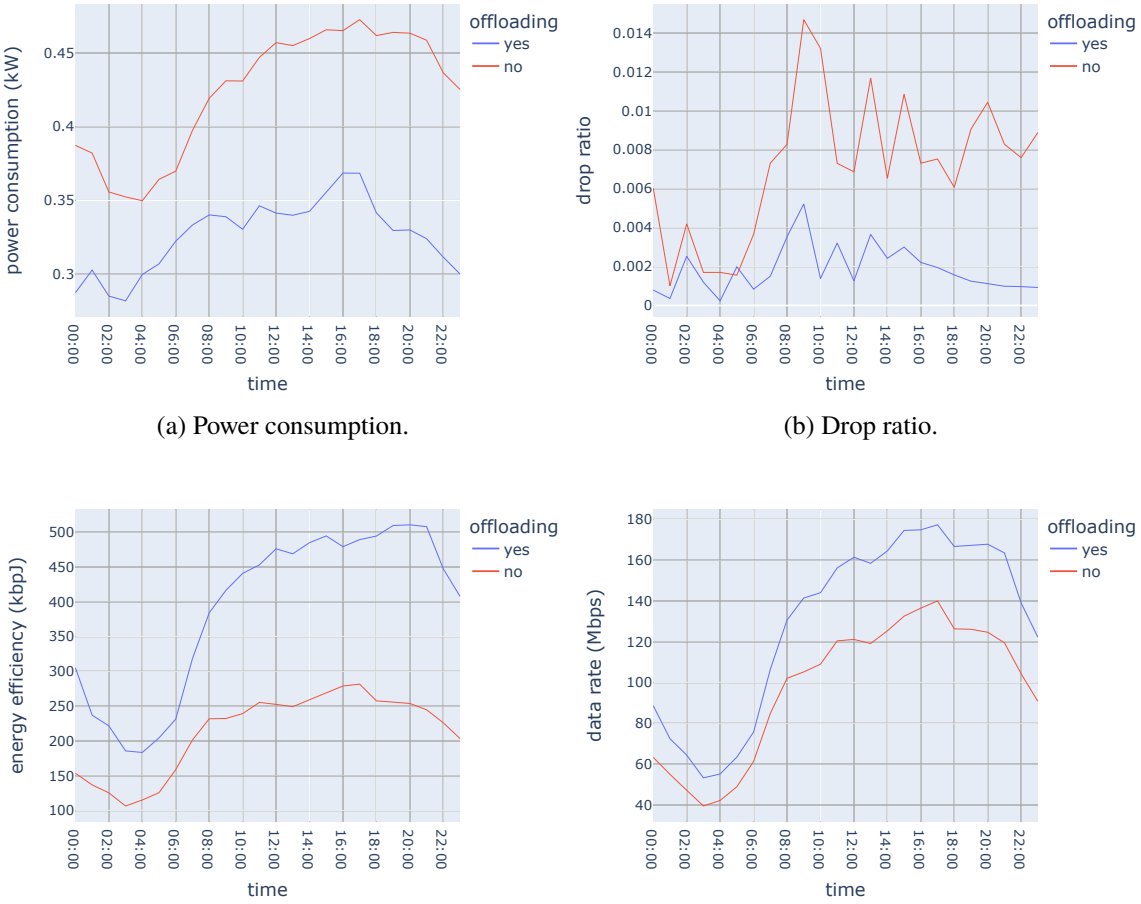
Figure 3.20: Comparisons of overall performance between two MAPPO policies trained with and without consideration of the inter-cell interference.

The zero-force precoding of the massive MIMO channel eliminates the interference between user equipments served by the same BS, but interference between those served by different BSs still exists. The multi-agent BS control policy should be aware of this inter-cell interference and try to procure a collaboration between BSs to reduce it, thus improving energy efficiency, by adaptively switching off some BSs. Here we compare the performance of two MAPPO policies, one trained with the knowledge of inter-cell interference, the other ignorant of it, in a simulation environment where this interference exists. The results are shown in Figure 3.19 and 3.20.

We can see that the ignorance of interference makes the agents over-optimistic about their data rates, thus leading to more sleeps. As a result, there are more packet drops, but the total power consumption goes a bit lower.

3.6.8 Effect of Offloading

Finally we consider the effect of user offloading in the BS control policy. When a user arrives in the network, as described in Subsection 3.1.8, it will first send a service request to the BS with the best estimated signal gain. In our service model the BS can decide whether to accept the request or to offload this UE to another BS. We want to investigate how this factor of offloading



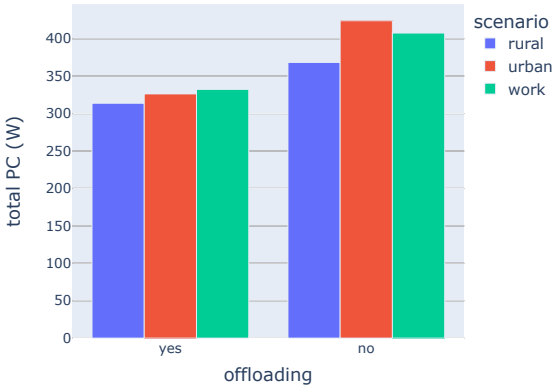
(a) Power consumption.

(b) Drop ratio.

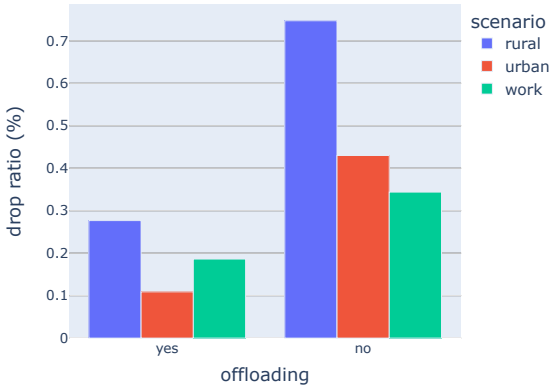
(c) Energy efficiency.

(d) Network throughput.

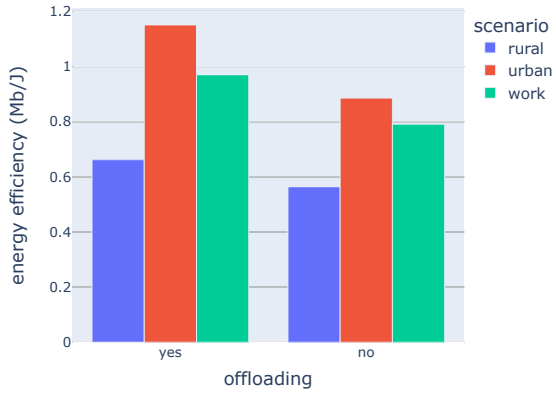
Figure 3.21: Comparisons of daily performance in the urban scenario between MAPPO policies enabling and disabling offloading.



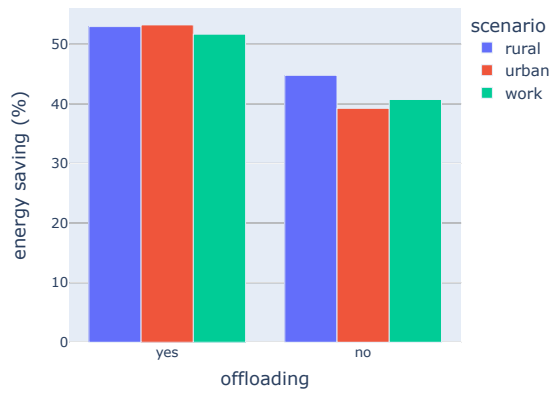
(a) Power consumption.



(b) Drop ratio.



(c) Energy efficiency.



(d) Energy saving (w.r.t. "always-on" policy).

Figure 3.22: Comparisons of overall performance between MAPPO policies enabling and disabling offloading.

improve the performance of the BS control policy, in comparison to that in a network without user offloading, where a BS will always accept the service request and attach the new UE to itself. As we can see in Figure 3.21 and 3.22, with offloading, sleeping BSs can sleep longer and deeper, letting active neighbor BSs serve UEs in the signal coverage. Otherwise, it has to wake up to serve these UEs, both increasing service delay (consequently packet drop) and power consumption.

3.6.9 Computational Overhead

The number of CPU operations for a single forward pass of each layer in the actor network has been calculated and shown in Table 3.9. Assuming 7 BS agents and a decision frequency of 50 Hz, the total computational cost of all agents in the network is calculated in Gflops (flops standing for *floating point operations per second*). Assuming further that the computational efficiency of a BS is 12.8 Gflops/W, as given in Table 3.3, we can estimate the power consumption of the computation of the multi-agent BS decisions. It turns out that the PC is very small, less than 0.01 W. Therefore, in comparison to the PC the algorithm is able to save in the multi-BS network, its own PC can be regarded as negligible.

Layer	Single pass ops	Total Gflops	PC (mW)
LayerNorm(116)	1740	0.000609	0.047578
Dense(116, 64, activation=relu)	104384	0.036534	2.854250
LayerNorm(64)	899	0.000315	0.024582
Dense(64, 64, activation=relu)	57792	0.020227	1.580250
LayerNorm(64)	899	0.000315	0.024582
Dense(64, 64, activation=relu)	57792	0.020227	1.580250
LayerNorm(64)	899	0.000315	0.024582
Dense(64, 3, activation=softmax)	2814	0.000985	0.076945
Dense(64, 4, activation=softmax)	3752	0.001313	0.102594
Dense(64, 3, activation=softmax)	2814	0.000985	0.076945
TOTAL	233785	0.081825	6.392559

Table 3.9: Computation and energy overhead of the multi-agent BS control.

Chapter 4

Conclusions and Future Work

4.1 Conclusions

This thesis consists of two parts: (1) the classification of cellular traffic with a consideration of the different packet delay budgets of network services and (2) cooperative base station control of antennas, advanced sleep modes, and user association in the 5G network using multi-agent reinforcement learning (MARL). We divided the delay requirements of different services into three categories: delay stringent, delay sensitive, and delay tolerant. In order to classify the mobile network cells, we vectorize the average weekly traffic for each delay category and used unsupervised machine learning to cluster the vectorized cells. FFT is used in the vectorization to extract different periodities, filter out the noise, and reduce the dimensionality of the vectorization. A outlier detection algorithm based on the K-nearest-neighbor method is used to remove the cells with peculiar traffic patterns and K-means is used to cluster the rest of the cells. Based on Davies-Bouldin (DBI) score, the best number of clusters turned out to be 4. By visualizing the average weekly traffic patterns in different service categories, analyzing the average traffic density, and observing the geographic distribution of different cell clusters, these four cell clusters can be approximately interpreted as low-traffic, rural or suburban, urban, and workplace. This clustering result can be useful for further network traffic analysis for applications like base station deployment, resource allocation, land use detection, etc. In addition, the cell classification provides different traffic scenarios simulated in the RL environment of the second part of work.

To investigate the potential of MARL in the application of cooperative 5G BS control for dynamic energy saving, we implemented a learning environment simulating a network of 7 BSs with randomly arriving users following a pre-defined traffic pattern, where each BS can control its antennas, sleeping, and user association. Three sleep modes with different wake-up delays and power-saving ratios are available to the BSs, owing to new features in the 5G technology. The BS agents were jointly rewarded for energy saving and packet delay reduction, and penalized for packet dropping. MAPPO with an actor-critic architecture is used to train the policy. In our investigation, the trained MAPPO policy is able to save about 50% of energy in comparison with the scenario that all BSs are always active with all their antennas. Another policy we call Auto-SM1 is also used as a benchmark, in which a BS sleeps in the shallowest sleep mode when it has no users to serve. The Auto-SM1 can save about 25% of energy without incurring any additional dropping. With regard to QoS, the MAPPO policy has an additional

0.2-0.5% of packet drop ratio. The ability to control user association allows a sleeping BS to offload arriving UEs in its cell to an active neighbor BS, thus avoiding waking up and additional service delay. In the simulation, this means 20-50% less packet drop and 5-10% more energy saving. The zero-forcing precoding in massive MIMO eliminates the intra-cell interference, but the inter-cell interference still has a critical affect on the QoS for the UEs. The control policy should also take interference into account, or else it will incur a dramatic increase in packet drop.

In addition to the technical contributions of this thesis, the work is also related to the United Nations' Sustainable Development Goals (SDGs), particularly SDG 7: Affordable and Clean Energy, and SDG 11: Sustainable Cities and Communities. SDG 7 aims to ensure access to affordable, reliable, sustainable, and modern energy for all, and our research addresses the challenge of reducing energy consumption in 5G networks. By proposing a multi-agent reinforcement learning-based algorithm for energy-efficient and cooperative base station control, we have achieved a significant reduction in network energy consumption while maintaining QoS for users. This has the potential to contribute to sustainable development by reducing the carbon footprint and making 5G networks more energy-efficient. Additionally, our work contributes to SDG 11, which focuses on building sustainable cities and communities. The classification of cellular traffic based on the delay requirements of network services can be useful for the development of smart cities that can provide better mobile network services to their residents with more efficient usage of network resources. Our work shows how technology, specifically AI and machine learning, can contribute to sustainable development goals by improving the efficiency of energy consumption and creating smarter and more sustainable cities.

4.2 Future work

We propose the following different directions in which future work could further advance the state of the art in multi-agent reinforcement learning for dynamic BS control to reduce energy consumption, and ultimately contribute to the development of more energy-efficient and high-performing next-generation wireless networks.

Improved Network Model and Algorithm

One potential direction for future work is to develop more sophisticated models of the network, which could lead to more accurate predictions and better performance in terms of energy saving and QoS. Additionally, more advanced power allocation algorithms could be developed to improve the efficiency of the base station sleeping strategy. This could involve combining reinforcement learning with other optimization techniques, such as convex optimization or game theory.

Fine-Tuned QoS and State Representation

Another area for future work is to further refine the treatment of different QoS requirements. This could involve developing more sophisticated metrics to evaluate the quality of service provided to different users or applications. Additionally, more detailed state representations of

user equipment could be explored, which could lead to more accurate modeling and prediction of user behavior.

More Comprehensive Performance Benchmarking

Another important direction for future work is to compare the proposed multi-agent reinforcement learning approach with other RL algorithms such as DQN, as well as with other MARL algorithms such as QMix or MADDPG. This could provide valuable insights into the strengths and weaknesses of the proposed approach and help to identify areas where further improvements could be made.

End-to-End Optimization

In addition to improving the model and algorithm, future work could also explore the end-to-end optimization of the network, including the joint design of radio, fronthaul, and cloud computing resources. This could involve developing reinforcement learning algorithms for such a design, and would require a holistic approach to network optimization that takes into account the entire network architecture and its various components.

Risk-Aware Reinforcement Learning and Dynamic Functional Split Optimization

Finally, future work could also explore risk-aware reinforcement learning and dynamic functional split optimization based on user delay requirements. This could involve developing new RL algorithms that take into account the potential risks associated with different actions, as well as developing new optimization techniques to dynamically adjust the functional split of the network based on changing user requirements.

References

[1] "Mobile data traffic forecast – Mobility Report," https://www.ericsson.com/en/reports-and-papers/mobility-report/dataforecasts/mobile-traffic-forecast, Nov. 2020.

[2] S. K. G. Peesapati, M. Olsson, M. Masoudi, S. Andersson, and C. Cavdar, "An Analytical Energy Performance Evaluation Methodology for 5G Base Stations," in *2021 17th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Oct. 2021. doi: 10.1109/WiMob52687.2021.9606296. ISSN 2160-4894 pp. 169–174.

[3] F. E. Salem, A. Gati, Z. Altman, and T. Chahed, "Advanced Sleep Modes and Their Impact on Flow-Level Performance of 5G Networks," in *2017 IEEE 86th Vehicular Technology Conference (VTC-Fall)*, Sep. 2017. doi: 10.1109/VTCFall.2017.8288125 pp. 1–7.

[4] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, Feb. 1998. ISBN 978-0-262-30384-2

[5] H. Giang, T. Hoan, P. Thanh, and I. Koo, "Hybrid NOMA/OMA-Based Dynamic Power Allocation Scheme Using Deep Reinforcement Learning in 5G Networks," *Applied Sciences*, vol. 10, p. 4236, Jun. 2020. doi: 10.3390/app10124236

[6] "Mobile Net Zero: State of the Industry on Climate Action 2022 report."

[7] "International - U.S. Energy Information Administration (EIA)," https://www.eia.gov/international/data/world/electricity/electricity-consumption?pd=2&p=0000002&u=0&f=A&v=mapbubble&a=-&i=none&vo=value&t=C&g=00000000000000000000000000000000000000000000000001&l=249-ruvvvvvfvtvnvv1vrvvvvfvvvvvvfvvvou20evvvvvvvvvvvnvvvs0008&s=315532800000&e=1609459200000&.

[8] "Mobile Industry position paper: Access to renewable energy."

[9] "Ambitious corporate climate action," https://sciencebasedtargets.org/.

[10] "GSMA | Mobile Net Zero: State of the Industry on Climate Action 2022 report - #BetterFuture," https://www.gsma.com/betterfuture/resources/mobile-net-zero-state-of-the-industry-on-climate-action-2022-report.

[11] P. Lähdekorpi, M. Hronec, P. Jolma, and J. Moilanen, "Energy efficiency of 5G mobile networks with base station sleep modes," in *2017 IEEE Conference on Standards for Communications and Networking (CSCN)*, Sep. 2017. doi: 10.1109/CSCN.2017.8088616 pp. 163–168.

[12] F. Xu, Y. Li, H. Wang, P. Zhang, and D. Jin, "Understanding Mobile Traffic Patterns of Large Scale Cellular Towers in Urban Environment," *IEEE/ACM Transactions on Networking*, vol. 25, no. 2, pp. 1147–1161, Apr. 2017. doi: 10.1109/TNET.2016.2623950

[13] "Specification # 23.501," https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3144.

[14] F. Corpet, "Multiple sequence alignment with hierarchical clustering," *Nucleic Acids Research*, vol. 16, no. 22, pp. 10 881–10 890, Nov. 1988. doi: 10.1093/nar/16.22.10881

[15] J. H. Ward, "Hierarchical Grouping to Optimize an Objective Function," *Journal of the American Statistical Association*, vol. 58, no. 301, pp. 236–244, Mar. 1963. doi: 10.1080/01621459.1963.10500845

[16] M. M. A. Hossain, C. Cavdar, E. Bjornson, and R. Jantti, "Energy-Efficient Load-Adaptive Massive MIMO," in *2015 IEEE Globecom Workshops (GC Wkshps)*, Dec. 2015. doi: 10.1109/GLOCOMW.2015.7414181 pp. 1–6.

[17] M. M. A. Hossain, K. Koufos, and R. Jantti, "Minimum-Energy Power and Rate Control for Fair Scheduling in the Cellular Downlink under Flow Level Delay Constraint," *IEEE Transactions on Wireless Communications*, vol. 12, no. 7, pp. 3253–3263, Jul. 2013. doi: 10.1109/TWC.2013.062413.120686

[18] E. Björnson, L. Sanguinetti, J. Hoydis, and M. Debbah, "Designing multi-user MIMO for energy efficiency: When is massive MIMO the answer?" in *2014 IEEE Wireless Communications and Networking Conference (WCNC)*, Apr. 2014. doi: 10.1109/WCNC.2014.6951974. ISSN 1558-2612 pp. 242–247.

[19] B. Debaillie, C. Desset, and F. Louagie, "A Flexible and Future-Proof Power Model for Cellular Base Stations," in *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, May 2015. doi: 10.1109/VTCSpring.2015.7145603. ISSN 1550-2252 pp. 1–7.

[20] F. Salem, "Management of advanced sleep modes for energy-efficient 5G networks," Ph.D. dissertation, Dec. 2019.

[21] S. Tombaz, P. Frenger, M. Olsson, and A. Nilsson, "Energy performance of 5G-NX radio access at country level," in *2016 IEEE 12th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Oct. 2016. doi: 10.1109/WiMOB.2016.7763183 pp. 1–6.

[22] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986. doi: 10.1038/323533a0

[23] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Jan. 2017.

[24] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-Dimensional Continuous Control Using Generalized Advantage Estimation," Oct. 2018.

[25] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," Aug. 2017.

[26] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust Region Policy Optimization," Apr. 2017.

[27] S. Kakade and J. Langford, "Approximately Optimal Approximate Reinforcement Learning," in *Proceedings of the Nineteenth International Conference on Machine Learning*, ser. ICML '02. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., Jul. 2002. ISBN 978-1-55860-873-3 pp. 267–274.

[28] C. Yu, A. Velu, E. Vinitsky, Y. Wang, A. Bayen, and Y. Wu, "The Surprising Effectiveness of PPO in Cooperative, Multi-Agent Games," Jul. 2021.

[29] K. Kurach, A. Raichuk, P. Stańczyk, M. Zając, O. Bachem, L. Espeholt, C. Riquelme, D. Vincent, M. Michalski, O. Bousquet, and S. Gelly, "Google Research Football: A Novel Reinforcement Learning Environment," Apr. 2020.

For DIVA

{
"Author1": {
"Last name": "Cai",
"First name": "Tianzhang",
"E-mail": "tcai@kth.se",
"organisation": {"L1": "School of Electrical Engineering and Computer Science ",
}
},
"Degree": {"Educational program": "Master's Programme, Machine Learning, 120 credits"},
"Title": {
"Main title": "Mobile Traffic Classification and Multi-Cell Base Station Control for Energy-Efficient 5G Networks",
"Language": "eng" },
"Alternative title": {
"Main title": "",
"Language": "swe"
},
"Supervisor1": {
"Last name": "Demir",
"First name": "Özlem Tugfe",
"Local User Id": "u100003",
"E-mail": "ozlemtd@kth.se",
"organisation": {"L1": "School of Electrical Engineering and Computer Science ",
"L2": "Computer Science" }
},
"Examiner1": {
"Last name": "Cavdar",
"First name": "Cicek",
"Local User Id": "u100004",
"E-mail": "cavdar@kth.se",
"organisation": {"L1": "School of Electrical Engineering and Computer Science ",
"L2": "Computer Science" }
},
"Other information": {
"Year": "2023", "Number of pages": "xiii,79"}
}