PhD-FSTM-2022-065
The Faculty of Science, Technology and Medicine

DISSERTATION

Defence held on 13/05/2022 in Luxembourg

to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG

EN INFORMATIQUE

by

Yaxiong YUAN
Born on 01 December 1992 in Hunan (CHINA)

MACHINE LEARNING-BASED EFFICIENT RESOURCE SCHEDULING FOR FUTURE WIRELESS COMMUNICATION NETWORKS

Dissertation defence committee:

Dr Symeon CHATZINOTAS, dissertation supervisor
Professor, Université du Luxembourg

Dr Osvaldo SIMEONE
Professor, King's College

Dr Björn OTTERSTEN, Chairman
Professor, Université du Luxembourg

Dr Arumugam NALLANATHAN
Professor, Queen Mary University of London

Dr Lei LEI, Vice Chairman
Associate Professor, Xi'an Jiaotong University

"Learning without thought is deceptive;
thought without learning is perilous."

Confucius
Philosopher

To my family and friends

Abstract

The next-generation mobile communication system, e.g., 6G communication system, is envisioned to support unprecedented performance requirements such as exponentially increasing data requests, heterogeneous service demands, and massive connectivity. When these challenging tasks meet the scarcity of wireless resources, efficient resource management becomes crucial. Conventionally, optimization algorithms, either optimal or suboptimal, are the main approaches for solving resource allocation problems. However, the efficiency of these iterative optimization algorithms can significantly degrade when the problems become large or difficult, e.g., non-convex or combinatorial optimization problems. Over the past few years, machine learning (ML), as an emerging approach in the toolbox, is widely investigated to accelerate the decision-making process. Since applying ML-based approaches to solve complex resource management problems is in its early-stage study, many open issues and challenges need to be solved towards the maturity and practical applications. The motivation and objective of this dissertation lie at investigating and providing answers to the following research questions: 1) How to overcome the shortcomings of extensively adopted end-to-end learning in addressing resource management problems, and which types of features are suited to be learned if supervised learning is applied? 2) What are the limitations and benefits when widely-used deep reinforcement learning (DRL) approaches are used to address constrained and combinatorial optimization problems in wireless networks, and are there tailored solutions to overcome the inherent drawbacks? 3) How to enable ML-based approaches to timely adapt to dynamic and complex wireless environments? 4) How to enlarge the performance gains when the paradigm shifts from centralized learning to distributed learning? The main contributions are organized by the following four research works.

Firstly, from a supervised-learning perspective, we address common issues, e.g., unsatisfactory prediction performance and resultant infeasible solutions, when end-to-end learning approaches are applied to resource scheduling problems. Based on the analysis of optimal results, we design suited-to-learn features for a class of resource scheduling problems, and develop combined learning-and-optimization approaches to enable time-efficient and energy-efficient resource scheduling in multi-antenna systems. The original optimization problems are mixed-integer programming problems with high-dimensional decision vectors. The optimal solution requires exponential complexity due to the inherent difficulties of the problems. Towards an efficient and competitive solution, we apply fully-connected deep neural network (DNN) and convolutional neural network (CNN) to learn the designed features. The predicted information can effectively reduce the large search space and accelerate the optimization process. Compared to the conventional optimization and pure ML algorithms, the proposed method achieves a good trade-off between quality and complexity.

Secondly, we address typical issues when DRL is adopted to deal with combinatorial and non-convex scheduling problems. The original problem is to provide energy-saving solutions via resource scheduling

in energy-constrained networks. An optimal algorithm and a golden section search suboptimal approach are developed to serve as offline benchmarks. For online operations, we propose an actor-critic-based deep stochastic online scheduling (AC-DSOS) algorithm. Compared to supervised learning, DRL is suitable for dynamic environments and capable of making decisions based on the current state without an offline training phase. However, for the specific constrained scheduling problem, conventional DRL may not be able to handle two major issues of exponentially-increased action space and infeasible actions. The proposed AC-DSOS is developed to overcome these drawbacks. In simulations, AC-DSOS is able to provide feasible solutions and save around more energy compared to the conventional DRL algorithms. Compared to the offline benchmarks, AC-DSOS reduces the computational time from second-level to millisecond-level.

Thirdly, the dissertation pays attention to the performance of the ML-based approaches in highly dynamic and complex environments. Most of the ML models are trained by the collected data or the observed environments. They may not be able to timely respond to the large variations of environments, such as dramatically fluctuating channel states or bursty data demands. In this work, we develop ML-based approaches in a time-varying satellite-terrestrial network and address two practical issues. The first is how to efficiently schedule resources to serve the massive number of connected users, such that more data and users can be delivered/served. The second is how to make the algorithmic solution more resilient in adapting to the time-varying wireless environments. We propose an enhanced meta-critic learning (EMCL) algorithm, combining a DRL model with a meta-learning technique, where the meta-learning can acquire meta-knowledge from different tasks and fast adapt to the new task. The results demonstrate EMCL's effectiveness and fast-response capabilities in over-loaded systems and in adapting to dynamic environments compare to previous actor-critic and meta-learning methods.

Fourthly, the dissertation focuses on reducing the energy consumption for federated learning (FL), in mobile edge computing. The power supply and computation capabilities are typically limited in edge devices, thus energy becomes a critical issue in FL. We propose a joint sparsification and resource optimization scheme (JSRO) to jointly reduce computational and transmission energy. In the first part of JSRO, we introduce sparsity and adopt sparse or binary neural networks (SNN or BNN) as the learning model to complete the local training tasks at the devices. Compared to fully-connected DNN, the computational operations can be significantly reduced, and thus requires less energy consumption and fewer transmitted data to the central node. In the second part, we develop an efficient scheduling scheme to minimize the overall transmission energy by optimizing wireless resources and learning parameters. We develop an enhanced FL algorithm in JSRO, i.e., non-smoothness and constraints - stochastic gradient descent, to handle the non-smoothness and constraints of SNN and BNN, and provide guarantees for convergence.

Finally, we conclude the thesis with the main findings and insights on future research directions.

Acknowledgements

The journey of my Ph.D. study in Interdisciplinary Centre for Security, Reliability and Trust (SnT) at University of Luxembourg has been an unforgettable, challenging, and wonderful experience of my life. Foremost, I would like to express my sincere gratitude to my supervisor Prof. Symeon Chatzinotas and my co-supervisor Prof. Lei Lei, for their instruction, motivation, patience, and immense knowledge. During these years, they guided me all the time on research direction, writing skills, expertise, and attitudes towards academics and life. Without their help, this dissertation would not have been possible. I would like to extend my special gratitude to Prof. Lei Lei for spending time to assist me to polish the papers over and over again. Additionally, I would like to thank the external member of my CET committee, Prof. Osvaldo Simeone, for providing me with valuable suggestions which have helped me to significantly enhance the quality of the thesis. Further, I would like to thank Dr. Thang X. Vu for his constructive comments which are helpful in refining the initial ideas.

I would like to thank my research collaborators sincerely: Dr. Yang Yang, Dr. Eva Lagunas, Dr. Thang X. Vu, Prof. Ruud JG van Sloun, Prof. Zheng Chang, Prof. Lei Lei, Prof. Sumei Sun, Prof. Symeon Chatzinotas, and Prof. Björn Ottersten for helping me improve my technical skills and the qualities of research papers. Also, I would like to thank my colleagues in SnT for encouraging me and creating a comfortable work environment. Many thanks to Anyue, Lin, Hieu, Puneeth, Liz, Arsham, Ehsan, Robin, Ahmed, Gabriel, Norshahida, Himani, Ashok, Sumit, Phuc, Deyi, Ruizhi, Tong, Linlong, and Yuan for their enjoyable discussions which bring me all the happy moments in SnT. Sincere gratitude to my friends from China and Luxembourg, Tianzi, Nan, Yiming, Anyue, Jiale, Bowen, Yu, and Jing for being next to me during my difficult time.

Last but not least, I could not have accomplished my Ph.D. study without the support of my devoted parents. Thanks to the University of Luxembourg community for giving me an inspiring environment and lots of opportunities to grow. The generous financial help from the Fonds National de la Recherche (FNR-Luxembourg National Research Fund) via University of Luxembourg is gratefully acknowledged.

List of Abbreviations

5G	The fifth-generation mobile technology.
B5G	Beyond the fifth-generation mobile technology.
6G	The sixth-generation mobile technology.
eMBB	Enhanced mobile broadband.
URLLC	Ultra-reliable low-latency communications.
mMTC	Massive machine-type communications.
UAV	Unmanned aerial vehicle.
IoT	Internet-of-thing.
BS	Base station.
AI	Artificial intelligence.
ML	Machine learning.
NLP	Natural language processing.
NN	Neural network.
DNN	Deep neural network.
CNN	Convolutional neural network.
RNN	Recurrent neural network.
SNN	Sparse neural network.
BNN	Binary neural network.
SVM	Support vector machine.
NB	Naive Bayes.

WSN	Wireless sensor network.
E2E	End-to-end.
GD	Gradient descent.
SGD	Stochastic gradient descent.
LSTM	long short term memory.
ESN	Echo state network.
PCA	Principle component analysis.
RL	Reinforcement learning.
AC	Actor-critic.
DRL	Deep reinforcement learning.
MDP	Markov decision process.
DQN	Deep Q-network.
TD	Temporal difference.
DDPG	Deep deterministic policy gradient.
A3C	Asynchronous advantage actor-critic.
PPO	Proximal policy optimization.
LP	Linear programming.
CQP	Convex quadratic programming.
QIP	Quadratic integer programming.
ILP	Integer linear programming.
MIP	Mixed integer programming.
B&B	Branch and bound.
MISO	Multi-input single-output.
MIMO	Multi-input multi-output.
SIMO	Single-input multi-output.
V2V	Vehicle-to-vehicle.
3D	3-dimensional.
QoS	Quality of service.
LoS	line of sight.
MEC	Mobile edge computing.

LEO	Low earth orbit.
GEO	Geostationary equatorial orbit.
FL	Federated learning.
ADMM	Alternating direction method of multipliers.
MBS	Macro base station.
TDMA	Time division multiple access.
SDMA	Space division multiple access.
MMSE	Minimum mean square error.
CSI	Channel state information.
FSMC	First state Markov channel.
GSS	Golden section search.
TST	Terrestrial-satellite terminal.
VSAT	Very small aperture terminal.
MT	Mobile terminal.
TN-NTN	Terrestrial-non-terrestrial network.
GD	Ground device.

Notations

$\log(x)$	Natural logarithm function of x.
e^x	Exponential function of x.
$\mathbb{E}[\cdot]$	Expected value.
$\lceil \cdot \rceil$	Operation to round a value up to an integer.
$\boldsymbol{a}^{\mathbf{T}}$, $\boldsymbol{A}^{\mathbf{T}}$	Transpose of vector $\boldsymbol{a}$, transpose of matrix $\boldsymbol{A}$.
$\boldsymbol{a}^{\mathbf{H}}$, $\boldsymbol{A}^{\mathbf{H}}$	Conjugate transpose of vector $\boldsymbol{a}$, Conjugate transpose of matrix $\boldsymbol{A}$.
$diag(\boldsymbol{a})$	Diagonal matrix of vector $\boldsymbol{a}$.
$\lvert x \rvert$, $\lvert \mathcal{X} \rvert$	The absolute value of x, cardinality of set $\mathcal{X}$.
$\lVert \boldsymbol{a} \rVert$	The norm of vector $\boldsymbol{a}$.
$\max$	Maximum operation.
$\min$	Minimum operation.
argmax	An operation that finds the argument that gives the maximum value.
argmin	An operation that finds the argument that gives the minimum value.
$x \in \mathcal{X}$	Element x belongs to set $\mathcal{X}$.
$\mathcal{X} \setminus \mathcal{Y}$	Set $\mathcal{X}$ excluding elements in $\mathcal{Y}$.
$\mathcal{X} \cup \mathcal{Y}$	Union of set $\mathcal{X}$ and set $\mathcal{Y}$.
$\mathbb{R}, \mathbb{R}^{n \times m}$	Set of real values, set of $n \times m$ matrices with real-valued entries.
$\mathbb{C}, \mathbb{C}^{n \times m}$	Set of complex values, set of $n \times m$ matrices with complex-valued entries.
$\mathcal{N}(a, b)$	Gaussian distribution with mean a and variance b.
∇f	Gradient of function f.

Preface

This Ph.D. thesis has been carried out from October 2018 to March 2022 at Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, Luxembourg, under the supervision of Prof. Symeon Chatzinotas at SnT, University of Luxembourg, Luxembourg. Additionally, the Ph.D. thesis was co-supervised by Prof. Lei Lei, firstly as a Research Scientist at SnT, University of Luxembourg, Luxembourg and later as an Associate Professor at Xi'an Jiaotong University, China. The time-to-time evaluation of the Ph.D. thesis was duly performed by the CET members constituting the supervisors, co-supervisors, and Prof. Osvaldo Simeone from King's College, United Kingdom.

Contents

This Ph.D. thesis entitled *Machine Learning-based Efficient Resource Scheduling for Future Wireless Communication Networks* is divided into six chapters. In Chapter 1, the background, research problems, motivations, and contributions of the thesis are described. Chapter 2 investigates a supervised learning-assisted resource allocation scheme to reduce the computational time and guarantee the feasibility of the solutions. To solve the combinatorial and non-convex scheduling problems in an online manner, a deep reinforcement learning-based user scheduling algorithm is proposed in Chapter 3. Chapter 4 focuses on applying the meta-learning technique to enable the machine learning model to fast adapt to dynamic environments. Shifting from centralized learning to distributed learning, Chapter 5 develops a joint energy-saving scheme for wireless federated edge learning systems. Finally, the conclusions and future works are summarized in Chapter 6.

Support of the Thesis

This Ph.D. thesis has been fully supported by the Luxembourg National Research Fund under bilateral project LARGOS (12173206). The effort of collaborators and dissemination costs were supported by other projects, namely, the FNR CORE ProCAST (C17/IS/11691338), ROSETTA (C17/IS/11632107), and 5G-Sky (C19/IS/13713801), and the European Research Council under Project AGNOSTIC (742648). Additionally, the time-to-time support from SIGCOM is also gratefully acknowledged.

Contents

1 Introduction **1**
- 1.1 Problem Descriptions and Motivations . . . 3
  - 1.1.1 Supervised Learning-assisted Resource Optimization in Multi-Antenna Systems 3
  - 1.1.2 Enhanced DRL for Efficient Scheduling in Energy-Constrained Networks . . . 4
  - 1.1.3 Dynamic-Adaptive ML Solutions for Resource Management in Time-Varying Wireless Networks . . . 4
  - 1.1.4 Joint Energy-Saving Solutions for Federated Learning in Distributed Wireless Communication Networks . . . 5
- 1.2 Methodologies . . . 6
  - 1.2.1 Machine Learning . . . 6
  - 1.2.2 Optimization Theory . . . 8
- 1.3 Contributions and Organization . . . 10
- 1.4 Contributions Beyond Dissertation . . . 12

2 Centralized Learning: Supervised Learning-Assisted Resource Allocation **15**
- 2.1 Introduction . . . 15
  - 2.1.1 Contributions . . . 16
- 2.2 System Model . . . 16
- 2.3 Problem Formulation . . . 17
  - 2.3.1 Time Minimization Joint Scheduling and Power Allocation Problem . . . 18
  - 2.3.2 Energy Minimization by Group-Timeslots Allocation . . . 19
- 2.4 Scheduling Algorithm Design based on Deep Learning . . . 20
  - 2.4.1 Features to be Learned . . . 20
  - 2.4.2 Adopted Learning Models . . . 21
  - 2.4.3 Algorithm Summary . . . 23
- 2.5 Performance Evaluation . . . 23
- 2.6 Conclusion . . . 27

3 Centralized Learning: Reinforcement Learning-Based User Scheduling **29**
- 3.1 Introduction . . . 29
- 3.2 System Model and Problem Formulation . . . 31
  - 3.2.1 System Model . . . 31
  - 3.2.2 UAV's Energy Model . . . 33
- 3.3 Problem Formulation . . . 35

3.4 Heuristic Approach . . . 36
3.4.1 User-Timeslot Scheduling . . . 36
3.4.2 Hovering Time Allocation . . . 37
3.4.3 Algorithm Summary . . . 38
3.5 Actor-Critic-Based DRL Algorithm . . . 40
3.5.1 Overview of Actor-Cirtic-Based DRL (AC-DRL) . . . 40
3.5.2 Problem Reformulation . . . 41
3.5.3 The AC-DSOS Algorithm . . . 42
3.6 Numerical Results . . . 46
3.6.1 Parameter Settings . . . 47
3.6.2 Results and Analysis . . . 47
3.7 Conclusion . . . 52

4 Centralized Learning: Dynamic-Adaptive Scheduling with Meta Learning 55
4.1 Introduction . . . 55
4.1.1 Related Works: State-of-the-art and Limitations . . . 55
4.1.2 Motivations and Contributions . . . 56
4.2 System Model and Problem Formulation . . . 57
4.2.1 LEO-Terrestrial Network . . . 57
4.2.2 Channel Modeling . . . 59
4.2.3 Optimization Problem . . . 60
4.3 Characterization on Solution Development . . . 62
4.3.1 The Proposed Optimal and Sub-optimal Solutions . . . 62
4.3.2 Conventional Online-Learning Solutions and Limitations . . . 63
4.4 The Proposed EMCL Algorithm . . . 65
4.4.1 EMCL Framework . . . 65
4.4.2 Tailored Designs in EMCL . . . 67
4.4.3 Complexity Analysis for EMCL . . . 70
4.5 Numerical Results . . . 70
4.5.1 Capability in Dealing with Dynamic Environments . . . 71
4.5.2 Trade-offs between Computational Time and Optimality . . . 74
4.6 Conclusion . . . 75

5 Distributed Learning: Energy-Efficient Wireless Federated Edge Learning 77
5.1 Introduction . . . 77
5.1.1 Related Work . . . 77
5.1.2 Contributions . . . 79
5.2 Preliminary . . . 79
5.2.1 FL Framework in Wireless Edge . . . 79
5.2.2 Energy Models . . . 81
5.2.3 Time Consumption . . . 82
5.3 Sparsification in Local Learning Models . . . 83
5.4 Resource Optimization in FL . . . 85
5.5 The Proposed Joint Sparsification and Resource Optimization Scheme . . . 86
5.5.1 NSC-SGD for Local Training in JSRO . . . 87
5.5.2 Convergence Analysis . . . 88
5.6 Simulation Results . . . 90
5.6.1 Performance on Energy Conservation . . . 91

5.6.2 Performance on Learning Accuracy 93
5.7 Conclusion 94

6 Conclusions and Future Works **95**
6.1 Conclusions 95
6.2 Future Works 96
6.2.1 Feasibility Problem in ML 96
6.2.2 Embedding ML in Optimization Algorithm 97
6.2.3 Multi-Agent Learning Structure 97
6.2.4 Interpretability and Theoretical Supports of ML Models 98
6.2.5 AI application areas in B5G/6G scenarios 98
6.3 Proof of Lemma 1 117
6.4 Proof of Lemma 2 117
6.5 Proof of Lemma 3 118
6.6 Proof of Theorem 1 121

List of Figures

1.1 Illustrations of FC-DNN, CNN, and RNN 7
1.2 Illustrations of Value-based RL, Policy-based RL, and Actor-Critic 8

2.1 The illustration of FC-DNN structure 21
2.2 The illustration of CNN structure 22
2.3 Average gaps between DNN and optimum ($K = 10$, $L = 5$) 25
2.4 Transmission time comparison between the DNN-based algorithm and the optimal algorithm ($K = 10$, $L = 5$, $\alpha = 2.8$, training set size$= 800$) 26
2.5 Energy comparison between the learning-based algorithms and the optimal algorithm ($K = 5, 10, 15$, $L = 5$ $\alpha = 2.8$, training set size$= 800$) 26
2.6 Optimality and re-selection time with different α ($K = 10$, $L = 5$, training set size$= 800$) 27

3.1 An illustrative UAV-aided network. 32
3.2 An illustration of the frame-timeslot structure. 32
3.3 Energy curves for three possible cases. 38
3.4 The actor-critic framework of AC-DSOS. 43
3.5 Total energy vs. K ($T_{max} = 160$). 48
3.6 Average computational time vs. K ($T_{max} = 160$). 48
3.7 Total energy vs. T_{max} ($K = 7$). 49
3.8 Communication and hovering energy vs. T_{max} ($K = 7$). 49
3.9 Feasibility vs. learning episode. 50
3.10 Energy vs. learning episode. 50
3.11 Rewards vs. learning episode. 51
3.12 Rewards in AC-DSOS with fixed and reduced action space. 51
3.13 Energy comparison in a scenario with dynamic user arrival and departure. 52

4.1 An illustrative LEO-terrestrial communication system 58
4.2 Evolution of loss over time-varying demands. 64
4.3 The proposed EMCL framework. 66
4.4 Performance in adapting to dynamic scenario 1: user entry and leave. 72
4.5 Performance in adapting to dynamic scenario 2: bursty demands. 72
4.6 Performance in adapting to dynamic scenario 3: unforeseen channel variations. 73
4.7 Recovery time vs. number of users 73
4.8 Objective value vs. number of updates 74

4.9 Computational time vs. number of users 74

5.1 Illustration of wireless federated edge learning. 80
5.2 Timeline of the FL process. 83
5.3 The process of JSRO. 87
5.4 Distribution of weights of different neural networks. 91
5.5 Total energy vs. SINR. 91
5.6 Energy parts. 92
5.7 Total energy vs. Local iterations. 92
5.8 Accuracy of different neural networks with JSRO. 93
5.9 Accuracy of different federated optimizers with BNN. 93

List of Tables

2.1 DNN settings 24
2.2 CNN settings 24
2.3 CPU time in computation 25

3.1 Summary of symbols and notations 34
3.2 Parameters in AC-DSOS 47

4.1 Parameter settings 71

5.1 Values of Θ_k, Ψ_k, Φ_k, A_k and C_k 84
5.2 Parameter settings 90

Chapter 1

Introduction

Future wireless communication systems, e.g., 6G, are envisioned to support ever-advanced services and applications with the requirement of more disruptive and stringent technologies [1]. In particular, the services range from enhanced mobile broadband (eMBB), ultra-reliable low-latency communications (URLLC) to massive machine-type communications (mMTC) [2]. These unprecedented requirements can be progressively realized by adopting new radio technologies and advanced networking schemes, for example, unmanned aerial vehicle (UAV)-assisted heterogeneous networks, integrated satellite-terrestrial networks, and decentralized Internet-of-thing (IoT) frameworks.

In complex and heterogeneous networks, efficient resource management can improve the efficiency of resource utilization and guarantee service quality but face several challenges. Firstly, due to the expansion of the network scale and the diversification of the services, the amount of data requests and the number of mobile users are explosively growing. As a result, this may sharpen the scarcity of network resources, e.g., spectrum, energy, and computing resources, since massively connected users may need to compete for the limited resources [3]. Secondly, the previous communication schemes may not be able to directly apply to the state-of-the-art system to satisfy the stringent requirements. To solve this issue, a large number of works have proposed optimization-based algorithms to obtain the optimal or suboptimal solutions [4]. However, due to the inherent complexity of network structures and the combinatorial natures of the resource scheduling problems, the established optimization problems can be non-convex, discrete, high-dimensional, or with the proven NP-hardness. For example, many user grouping/scheduling, computation offloading, and power allocation problems are NP-hard in the scenarios of IoT [5], mobile edge computing [6], and network slicing [7]. In these cases, conventional gradient- or search-based optimization may be impracticable to provide optimal/near-optimal solutions timely due to high computational complexity. Although the development of heuristic algorithms could be an alternative to reduce the complexity, the solution quality might have to be compromised. Thirdly, wireless communication networks are inherently dynamic, for example, time-varying network topologies, fluctuated channel states, and bursty users' demands. Under such dynamic environments, the optimized solution needs to be updated in a real-time manner, which might not be always possible for conventional optimization algorithms.

Machine learning (ML), as a key technique of artificial intelligence (AI), has been widely adopted in many areas to enable automated and intelligent decision-making processes. For example, image detection, semantic recognition in natural language processing (NLP), intelligent driving, and AI-powered interactive experiences in games [8]. In recent years, ML-based techniques have been used in wireless networks to promote intelligent communication schemes and resource management. In [9, 10, 11, 12], the authors applied supervised learning algorithms, such as neural networks (NNs), support vector ma-

chine (SVM), and Bayesian learning, to solve the practical problem in wireless systems, e.g., signal detection in coded MIMO systems, resource allocation in secure IoT network, and intrusion detection in wireless sensor networks (WSNs). In [13, 14, 15, 16], reinforcement learning (RL) methods have been investigated to deal with the decision problems, e.g., DQN-based beamforming policy for intelligent reflecting surface (IRS)-aided systems and actor-critic (AC)-based UAV flying control. Compared to the conventional optimization methods with precise objectives and constraints, the advantages of ML-based algorithms lie in the following aspects:

- ML is a data/experience-driven method possessing the capability of finding intrinsic correlations and extracting useful information in complex environments without explicit programming. Thus, it has the ability to efficiently predict an optimized result which might be costly in the optimization process.
- As the well-trained ML model has the potential to output/predict the outcomes in real-time, it can adapt to the dynamic environments if the statistical properties of environmental changes are captured by the learning model [17].
- ML can be also embedded into the traditional schemes. For example, to reduce the operational complexity while guaranteeing high solution quality, we can utilize ML to learn some features which can accelerate the process of optimization or help the network realize self-optimize [18, 19].

Despite ML, as a promising technology, having achieved significant progress in wireless communication systems, it has bottlenecks in the design and implementation in practice. In [20], the authors summarized the limitations of ML such as difficulty in obtaining good training data and lack of interpretability. Besides the general limitations in [20], we conclude the main challenges for the design of ML algorithms, which stem from the inherent difficulties of the problems, the specificity of data structures and distributions, and the dynamics of the environments.

Firstly, there is no one-size-fits-all ML tool for the problems in wireless scenarios. For example, supervised learning is able to train the model based on a large amount of experienced data. When the considered problem is difficult or impossible to obtain the optimal solution, e.g., some NP-hard problems, we cannot collect sufficient labeled training data such that the supervised learning is impracticable. As another example, in distributed and privacy-sensitive scenarios, the federated learning (FL) model is more suitable than centralized learning. Generally, it is non-trivial but crucial to customize an ML scheme that is suitable for the characteristics of the problem.

Secondly, the optimal policy may not be easy to learn with end-to-end (E2E) learning, where E2E learning means using an ML model to directly output solutions. When there exists excess noise in the original data or the internal relationships between the inputs and outputs are too complicated, the learning algorithms are incapable of grasping the accurate mapping to fit the model. Besides, the curse of dimensionality will also make E2E learning infeasible. The high-dimensional data leads to the intricacy of data compiling in implementation and the difficulty of policy searching in a huge decision space.

Thirdly, the solutions provided by ML algorithms cannot guarantee practical constraints. In communication systems, resource limitations and network quality requirements are the common constraints on the design of communication schemes. However, the ML inference model may violate these constraints and return infeasible solutions.

Fourthly, a learning model may encounter the issue of re-training in the wireless environments with unpredictable and sudden variations. ML can learn optimal policies under ideal environments, i.e., the variation follows a certain statistical distribution. Nevertheless, for the unpredictable changes, e.g., bursty demands and random user arrival/departure, the performance of the previously-learned model may degrade. In this case, the re-training process is inevitable, which could be time-consuming and inefficient.

In this thesis, we exploit ML techniques to solve resource scheduling problems, such as user grouping, timeslot scheduling, power allocation, and learning parameter selection, in several challenging and promising future communication scenarios for efficient resource utilization, low algorithm complexity, and intelligentization. On top of that, this thesis focuses on addressing the practical issues in the phases of ML algorithm design and implementation, and providing answers to the following questions:

- Which types of features are suited to be learned if supervised learning is applied to resource management?
- What are the limitations and benefits when widely-used deep reinforcement learning (DRL) approaches are used to address constrained and combinatorial optimization problems in wireless networks, and any tailored solutions to overcome the inherent drawbacks?
- How to make ML models quickly adapt to new wireless environments when the system configurations are changing?
- How to enlarge the performance gains when the paradigm shifts from centralized learning to distributed learning?

In the following sections, we describe the research problems and the motivations. Then, we introduce the methodologies, i.e., ML and optimization, involved in analyzing the problems. Finally, the main contributions, publications, and organization of the thesis are presented.

1.1 Problem Descriptions and Motivations

In this section, we present details and motivations of the selected four research problems in future wireless communication systems.

1.1.1 Supervised Learning-assisted Resource Optimization in Multi-Antenna Systems

End-to-end (E2E) supervised learning, i.e., applying an ML model to directly output solutions, is an effective ML tool to learn the mapping between inputs and labeled outputs but not the wisest option in certain complicated cases. This is because E2E learning may suffer from performance degradation caused by unknown learnability of original problems, lack of optimality/feasibility guarantee, and the high dimension of the decision variables. To enhance the learning performance and solution quality, we can design some features that are suitable to learn and helpful for obtaining the optimal solutions. To this end, we investigate the supervised learning-assisted resource allocation in a multi-antenna system, i.e., multi-input single-output (MISO).

Multi-antenna techniques, e.g., MISO, single-input multi-output (SIMO), and multi-input multi-output (MIMO), are designed to improve spectrum efficiency, network capacity, and link reliability by using spatial multiplexing and diversity technique. In [21], the authors proposed an ultra-reliable multi-user MIMO detector for 5G/B5G enabled IoT networks, where the system is operating in interfering environments in the time-frequency domain. In [22], a three-dimensional (3D) cluster-based model was designed for vehicle-to-vehicle (V2V) massive MIMO systems with wideband channels. In multi-antenna systems, designing efficient user scheduling and resource allocation schemes benefits for improving resource utilization and energy efficiency. In addition, multiple users can be simultaneously served with independent information over the same spectrum resources with precoding which is considered as an effective way to mitigate the interference [23]. To improve the spectral efficiency, the authors in [24] proposed an algorithm via joint user scheduling, power allocation, and precoding for massive

MIMO Systems. In [25], a joint user scheduling and precoding scheme was designed to maximize energy efficiency in MISO systems. To satisfy the predefined quality of service (QoS) requirements in B5G, the authors in [26] developed joint bandwidth allocation and precoding algorithms for two problems in multi-antenna scenarios, i.e., user fairness maximization and transmit power minimization, respectively. However, the optimization-based algorithms have some limitations on determining resource management strategies. Specifically, most of the user scheduling and resource allocation problems are combinatorial and non-convex. Due to the high complexity, the conventional optimal/suboptimal iterative algorithms may be time-consuming to converge or achieve satisfactory performance. To tackle this issue, in [27, 28, 29], the authors developed ML-based resource allocation to learn the relationship between input parameters and the optimal decisions and predict solutions efficiently. Nevertheless, most ML-related researches adopt E2E learning, which motivates us to find proper features that facilitate ML-based algorithms providing feasible, near-optimal, and efficient resource allocation schemes.

1.1.2 Enhanced DRL for Efficient Scheduling in Energy-Constrained Networks

RL/DRL algorithms are popular and efficient approaches in wireless areas to solve the decision-making and control problems. However, when conventional RL/DRL is used to address combinatorial and constrained optimization problems, the learning performance and solution quality may decrease due to the issues of huge action space and infeasibility. We consider a DRL application for a user scheduling problem in unmanned aerial vehicle (UAV)-aided networks, where the combinatorial nature and constraints exist in the problem formulation.

With significant growing demands of high data transmission rate, flexible connectivity, and wide range of service coverage in future wireless networks, the applications of UAVs have been fast growing in the last few decades owing to the properties of mobility, low cost, and line-of-sight (LoS) communication. In UAV-aided communication systems, the network performance can be optimized by UAV's trajectory planning, hovering/flying control, and resource scheduling. In [30, 31], the authors investigated resource allocation and trajectory optimization for multi-UAV-assisted mobile edge computing systems by taking into account the propulsion energy. In [32], a joint UAV-to-community offloading and user clustering scheme was designed to maximize the system throughput with the constraints of transmission rate, tasks' atomicity, and UAVs' speed. In addition, RL-based UAV control and communication schemes are emerging due to the facts that 1) the flying control and resource allocation problems are suitable for transforming into decision processes over the time sequences; 2) RL does not require the prepared training data thus avoiding the data collection phase as in supervised learning; 3) RL is apt to make online decisions and explore optimal strategy in dynamic environments. In [33, 34, 35], RL/DRL are adopted to enable UAVs to maximize network benefit via intelligently updating their flying directions, placement strategy, and resource allocation. However, the problems of joint UAV flying control and user scheduling are difficult because of the mixed and intertwined variables, combinatorial decision space, and constraints, which could degrade the learning performance. Thus, the focus of this work is to design a tailored DRL-based user scheduling scheme that overcomes the inherent hardness in UAV-assisted networks.

1.1.3 Dynamic-Adaptive ML Solutions for Resource Management in Time-Varying Wireless Networks

A common problem with ML algorithms is that when they encounter a dynamic environment, previously well-trained models may become invalid. This can lead to time-consuming retraining and data re-collecting. Thus, it is necessary to develop a technique to improve the sustainability and adaptability

of an ML model. For this purpose, we investigate a DRL-based resource management problem in a highly dynamic LEO-terrestrial scenario.

In the Beyond 5G (B5G)/6G era, terrestrial communications systems, which heavily rely on cellular wireless systems and fiber optic cables, will not be enough to fulfill the requirement of nearly ubiquitous and instantaneous connectivity for large numbers of devices globally. Instead, satellite communication can take advantage of its wide coverage, large capacity, and high transmission rate to solve the limitations in terrestrial networks, e.g., "the last mile" problem in certain areas where the fiber is difficult to deploy or the discrete requests frequently occur, e.g., processing credit card payments [36]. Satellites can be classified according to the altitude of their orbits, e.g., geostationary equatorial orbit (GEO) satellite and low earth orbit (LEO) satellite. LEO satellites are smaller and much closer to earth compared to GEO. Therefore, the economic and energy costs to launch LEO are cheaper. Besides, LEO satellites are flexible and well suited for connecting mobile devices and integrating them into terrestrial networks. These advantages render LEO communication becomes a promising technique in future satellite-terrestrial integration [37]. The authors in [38, 39] have proposed the resource management problems over LEO satellites, e.g., users association, resource allocation, and routing problems, with optimization-based solutions. In [40, 41, 42], RL/DRL-based algorithms were developed to allow online operations and suit time-varying network topology. In LEO-terrestrial systems, the communication environment is remarkably dynamic, e.g., fluctuated channel fading, users' random arrival/departure, and bursty data demands. This dynamic nature also exists in the majority of future communication systems. Unfortunately, for ML methods, certain unpredictable configuration changes will invalidate the well-trained model as the probability distribution of the input data in the current environment deviates from the previous observations [43]. Hence, it is significant to develop a learning model that can fast adapt to new environments.

1.1.4 Joint Energy-Saving Solutions for Federated Learning in Distributed Wireless Communication Networks

The above research problems focus on centralized learning methods. That is, the data processing, training, and decision-making are performed in a single centralized agent. Considering protecting the privacy of user data, reducing the computational pressure on the central node, and improving training efficiency, distributed learning methods can be adopted. However, in distributed scenarios, the computation ability and energy storage are limited at edge nodes. To explore how to improve the performance gain, e.g., on energy conservation and learning accuracy, for distributed learning, we study federated learning (FL) schemes in a distributed wireless communication network.

Distributed networking is a network architecture where the computing, communication, networking, and storage capabilities are spread out across multiple nodes, thus facilitating the reduction of transmission delay, privacy protection, and parallel processing. Besides, it supports data transmission between two nodes through wireless links and merges the distributed structures to B5G/6G scenarios, e.g., massive MIMO, network slicing, and AI-based mIoT networks [44, 45]. There exist some studies on the application of distributed learning in wireless communication. In [46], the authors investigated distributed learning algorithms for resource allocation in wireless IoT networks In [47], a resource-efficient distributed learning method was proposed in mobile edge computing (MEC) to coordinate intensive resources of mobile clients in computation, bandwidth, energy, and data. In this work, we consider one of the distributed learning algorithms, i.e., federated learning (FL), in wireless networks, where the devices learn a shared prediction model collaboratively while maintaining the training data locally rather than uploading it on a central server. One of the critical issues in FL design in wireless systems is the energy restriction at the local devices. In [48, 49, 50], the authors focused on minimizing the system energy consumption, including computing and communication, via optimizing the network and learning parameters

with a fixed ML model. Therefore, we explore whether the energy consumption can be further reduced by choosing a low-energy/lightweight ML model while guaranteeing learning performance.

1.2 Methodologies

In this section, we discuss the techniques, tools, and methods adopted in problem analysis. In general, the methodologies in this work can be categorized into two classes. One is ML-related models and technologies, and the other is optimization theory and algorithms. Additionally, the two methods can be combined and bring about greater benefits. For instance, ML can be embedded into the steps of optimization algorithms to accelerate the process of searching the optimal solutions.

1.2.1 Machine Learning

ML tasks are typically classified into three types: supervised learning, unsupervised learning, and reinforcement learning. Besides, many new techniques are investigated to address the shortcomings of the conventional ML methods, e.g., transfer learning for task migration, meta-learning for fast adaptation to the new environment, and federated learning for the non-centralized data structure. In this subsection, we introduce the three main types of ML and the representative algorithms.

Supervised Learning

Given a training set composed of multiple input-output pairs, the goal of supervised learning is to learn a hypothesis (also called a mapping function) so that, for any input, the hypothesis is a good "predictor" for the corresponding output. When the outputs (or learning targets) are continuous values, the task is a regression problem. When the outputs (or learning targets) are discrete values, the task is a classification problem. Several commonly used supervised learning techniques are:

- Linear/Logistic regression: The hypothesis is represented as a linear function or logistic function with regard to the input variables, which is easy to implement, interpret and efficient to train but based on the assumption of linearity for the input-output relationships [51].
- Naive Bayes (NB): With the assumption that hat the value of a particular feature (or the component of the input vector) is independent of the value of any other feature[1], NB constructs the prediction model based on the prior probability and Bayes rule. NB can easily predict the class of a test dataset but it is not suitable for the cases that violate the NB assumption [51].
- Support vector machine (SVM): SVM is a classifier to provide a $(p-1)$-dimensional hyperplane to separate data points that are p-dimensional vectors. Different SVM algorithms use different types of kernels to construct non-linear decision boundaries using linear classifiers, where the kernels are the mathematical functions to transform the input data into the required form. Although SVM works well in high dimensional spaces with a clear margin of separation, the performance deteriorates in the case of large datasets due to the higher required training time [52].
- Neural network (NN): NN is a series of algorithms to build the learning model that mimics the way the human brain operates to recognize underlying relationships from a dataset [53]. NN is flexible to adjust the number of nodes and layers and deft at fitting the non-linear model. The after-trained NN can produce the predicted result pretty fast. The above advantages make the NN very popular in a wide range of areas.

[1] This is also called NB assumption.

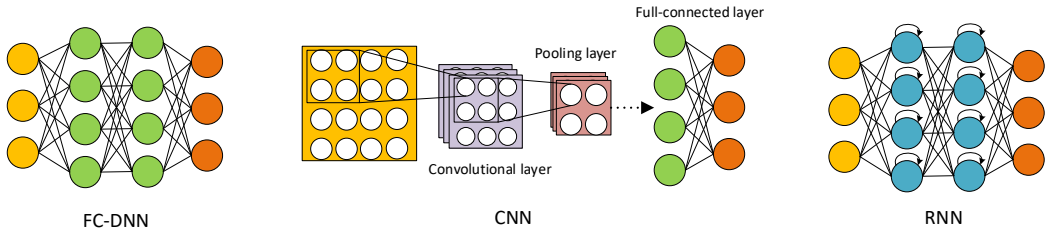


Figure 1.1: Illustrations of FC-DNN, CNN, and RNN

There exist different NN models and each suits a specific class of tasks. Three typical NNs are fully-connected deep NN (FC-DNN), convolutional NN (CNN), and recurrent NN (RNN), as illustrated in Fig. 1.1. In FC-DNN, the inputs pass forward through the network layer by layer, and the model is updated iteratively by backpropagation with stochastic gradient descent (SGD)-based methods, e.g., Adam. FC-DNN can be used for general regression and classification such as pattern recognition and function fitting [53]. CNN is a variation of the multi-layer structure, containing one or more convolutional layers and pooling layers. The convolutional layers use a convolutional kernel to create feature maps that record a region of an image and forward the features for nonlinear processing. CNN has high accuracy in image recognition problems with automatical feature detection and weight sharing [54]. For RNN, the outputs of each processing node do not pass the information in a unidirection but feed the result back to the model. By unfolding the RNN architecture, the output of a node is dependent on both the current input and former inputs, which endows RNN with the capability of remembering. RNN and its variants like echo state networks (ESNs) and long short term memory (LSTM) devote to the prediction based on time-related data, e.g., speech/text recognition [55].

Unsupervised Learning

Unsupervised learning aims to discover inherent patterns and information from unlabeled data [56]. Some examples of unsupervised learning algorithms include:

- K-means clustering: The aim is to group similar data points together and discover underlying patterns. The data points are allocated to the corresponding clusters by reducing the in-cluster sum of squares. K-means clustering is relatively simple to implement with a large scale of datasets and guarantees convergence, but the performance largely depends on the selected K-value and initial points [56].
- Principal component analysis (PCA): By analyzing the principal features, PCA can realize dimensionality reduction, information compression, and data de-noising. PCA is commonly used in the data pre-processing phase to reduce the dimension and size of the original data thus improving the algorithm performance and avoiding overfitting [56].

Reinforcement Learning

RL is to train a learning agent to find an optimal policy, i.e., a sequence of decisions/actions, for a Markov decision process (MDP) without knowing the transition probabilities of the environments [57]. Three elements of RL are state, action, and reward. The goal of the RL agent is to optimize a long-term reward by interacting with the environment based on a trial and error process. Different from supervised learning, RL updates the policy based on exploiting the rewards feedback which is obtained from the

exploration instead of the pre-collected labeled data. Thus, RL does not require data preparation in advance. The RL algorithms can be divided into three classes, as shown in Fig. 1.2.

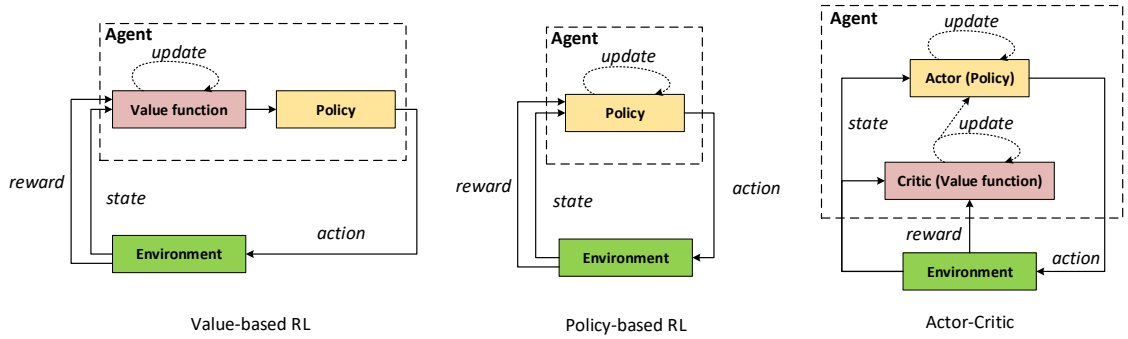


Figure 1.2: Illustrations of Value-based RL, Policy-based RL, and Actor-Critic

- Value-based RL: In a value-based approach, we define a value function related to the state or action, e.g., Q-value or V-value. The value function is iteratively updated until the agent finds the optimal value function. During this iteration, the policy (or a sequence of the actions), which is selected based on the value function, is implicitly updated. The typical value-based RL algorithms are Q-learning and state-action-reward-state-action (SARSA) algorithm. To overcome the issue of huge/continuous state space, DNN can be integrated into RL to approximate the value function, such as deep Q-network (DQN) and Double DQN. The value-based methods allow using the idea of temporal difference (TD) learning to improve the sample efficiency [57].

- Policy-based RL: Policy-based RL directly learns optimal policy instead of the value function. Thanks to the policy gradient theorem [58], we can calculate the gradient of the objective reward function with regard to the parameters of the policy. Therefore, the policy can be updated by the gradient ascent method. The policy-based approaches tend to be more stable and have better convergence performance than value-based RL due to the inaccuracy of the value function approximation [57].

- Actor-Critic (AC): AC combines the advantages of value-based and policy-based methods by dividing the agent into two parts, i.e., an actor and a critic. The actor employs the policy gradient theorem to directly update the policy under the guidance of the critic. The critic is responsible for evaluating the actions via the value function with a good critic, which can be sample efficient via TD updates at every step. In terms of implementation, the approximation tools, such as DNN, can be adopted to parameterize the actor and critic. The popular AC methods include deep deterministic policy gradient (DDPG) [58], asynchronous advantage actor-critic (A3C) [59], and proximal policy optimization (PPO) [60].

1.2.2 Optimization Theory

Optimization theory establishes a set of systematic approaches to find an optimal or a suitable solution from a feasible set. In practical applications, it enables companies or industries to solve complex practical problems and make better use of available resources and data. Different from the ML approaches, which are based on data or experiences, the optimization-based method requires an explicit model construction. An optimization model encompasses 1) an objective, e.g., minimizing a loss/cost function or maximizing

a utility function; 2) one or several decision variables that directly affect the objective and can be adjusted; 3) multiple constraints that are used to confine the feasible range of the decision variables due to the practical or numeric limitations. We note that the constraints are optional. An optimization problem without constraints is called an unconstrained problem, otherwise, a constrained problem.

The optimization problem can be categorized into different types based on the properties of the objective functions (e.g., convex or non-convex), types of variables (e.g., continuous or discrete), and so on. The following lists the types of optimization problems involved in the problem investigation in the thesis:

- Linear programming (LP): The objective function and constraints are represented by linear combinations of the variables. Simplex and interior-point methods are the widely-used approaches to solve the LP problems [61].

- Convex quadratic programming (CQP): CQP is a type of non-linear programming with convex quadratic objective function and linear constraints. For CQP, a variety of methods can be used such as interior-point methods, augmented Lagrangian and conjugate gradient [62].

- Bilinear programming: Bilinear programming deals with a class of problems whose objectives or constraint functions are bilinear. Due to the non-convexity of the bilinear terms, the bilinear programming is non-convex. To solve the problem, McCormick envelopes can be employed to relax and linearize the bilinear terms [63].

- General convex problem: A convex problem is to minimize a convex function over convex sets, including LP and CQP. If the convex problem is unconstrained, it can be easily solved with the gradient descent method or Newton's method, which are mathematically proven to converge quickly [64]. For a constrained convex optimization problem, the KKT conditions are the equivalent conditions for the global optimum. The KKT conditions usually can not be solved directly, except in the few special cases where a closed-form solution can be derived analytically, such as CQP. In general, many optimization algorithms can be interpreted as the processes for numerically solving the KKT conditions. Besides, the constrained convex optimization problem can also be solved by logarithmic barrier approaches that replace constraints with a penalizing term in the objective function [64].

- Integer programming combinatorial optimization: In integer programming, some or all variables are restricted to be integers, so the feasible set is non-convex. Integer programming is proven to be NP-hard and it has variants, e.g., integer linear problem (ILP), binary optimization, and mixed-integer programming (MIP). One type of exact algorithm is the cutting plane method and the other is the branch and bound (B&B) [65]. In addition, some heuristic algorithms, e.g., tabu search for ILP, can be used to provide efficient solutions.

  The goal of combinatorial optimization is to find an optimal object from a finite set of objects, where the set of feasible solutions is discrete and the variables are integers. The typical combinatorial optimization includes the traveling salesman problem, assignment problem, and knapsack problem, which can be formulated into an integer programming problem and solved by dynamic programming or the same way we deal with the integer programming problem. However, the searching process for combinatorial optimization is more difficult as the feasible set possesses combinatorial nature and increases exponentially with the scale of the problems.

- Optimization with cumulative objectives: In machine learning algorithms, the parameters are updated in the direction of minimizing an objective function that is the cumulative sum of all the loss

functions of the training samples. Due to the huge size of the sample data, conventional gradient descent (GD) is not efficient, as it needs to calculate the cumulative gradient at each iteration. To improve the learning speed, stochastic gradient descent (SGD) is adopted to approximate the gradient with parts of the sample at each iteration. The commonly-used SGD-based method includes Adam, RMSProp, and AdaBound [66].

The optimization-based approaches are interpretable as they are based on the appropriate mathematical formulation and analyzed by the optimization theory. However, in the B5G/6G era, communication networks, structures, and services are complex and diversified, leading to the challenges, such as time costs, from optimization modeling to optimization algorithm development. In this work, we mainly seek data/experiences-based ML methods to solve problems that are difficult to handle with traditional optimization. The optimization-based algorithms are also considered to benchmark the ML-based or ML-assisted schemes.

1.3 Contributions and Organization

In this thesis, we investigate ML applications for resource scheduling problems in four different wireless communication scenarios. Concretely, we formulate optimization problems, e.g., power control, user scheduling, and spectrum allocation, to improve resource utilization, network quality, and users' experience. Since the traditional optimization-based and heuristic method are incapable of realizing a good trade-off between computational complexity and solution quality, ML-based algorithms are adopted to make fast and intelligent decisions. To design time-, energy-, or resource-efficient ML-based schemes, we select suitable learning models to handle the practical limitations in the implementation. The adopted ML methods range from centralized learning to distributed learning, and from supervised learning to reinforcement learning. Simulations are conducted to verify the effectiveness of the proposed ML-based algorithms for all the cases. The organization and the main contributions are presented below.

The thesis is organized into six chapters. In chapter 1, a general background of the research topic is provided. In detail, the description and significance of the selected research problems are explained first. Then, we introduce an overview of the adopted methodologies. In addition, the major contributions of this thesis are discussed.

In chapter 2, to design the resource scheduling schemes in MISO systems, we first formulate a time minimization problem and an energy minimization problem via user grouping with power control and timeslot-group allocation, respectively. The formulated problems are non-linear MIP and ILP, respectively, which are proven to be NP-complete. To solve the problems, we propose optimization-based approaches by decomposition method and B&B search. However, the exponentially increasing computing complexity leads to difficulties in real-time applications. To significantly reduce the computational time, we develop an ML-based approach to enable fast execution in decision making while guaranteeing learning performance. Different from the E2E learning-based methods, we focus on extracting the learnable features from the original problems and learning the mapping between the inputs and features. This firstly improves the learning performance, e.g., learning accuracy and feasibility issues. Besides, the optimization model can exploit the well-learned features to reduce the complexity without much loss of optimality. Numerical results show that the learning-assisted algorithm is effective in approximating optimal scheduling solutions and reducing computational time compare to the conventional optimization and search algorithms. The outputs of the chapter are published in:

[**C1**] **Y. Yuan**, T. X. Vu, L. Lei, S. Chatzinotas, B. Ottersten, "Joint User Grouping and Power Allocation for MISO Systems: Learning to Schedule," in Proc. *European Signal Processing Conference (EUSIPCO)*, pp. 1-5, 2019.

[**C2**] L. Lei, **Y. Yuan**, T. X. Vu, S. Chatzinotas and B. Ottersten, "Learning-Based Resource Allocation: Efficient Content Delivery Enabled by Convolutional Neural Network," in Proc. *IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pp. 1-5, 2019.

In chapter 3, we consider applying a DRL algorithm to solve the decision problem in UAV-aided networks without the offline training process. Specifically, we formulate an energy minimization problem for multi-antenna UAV communication systems. The formulated problem is combinatorial and non-convex with bilinear constraints and two decision variables, i.e., user-timeslot allocation and UAV's hovering time assignment, are coupled in the optimization tasks. By analyzing the interplay among communication energy, hovering time, and hovering energy, two offline benchmarks are proposed. One is a relax-and-approximate method combining McCormick envelopes relaxation and B&B search to approach the optimum. The other is a golden section search-based heuristic algorithm with lower complexity than the optimal solution. To meet the requirement of online operations, we develop an AC-based deep stochastic online scheduling algorithm for UAV energy savings, where the original problem is transformed into an MDP. Unlike conventional DRL, we design a set of tailored approaches, e.g., stochastic policy quantification, action space reduction, and feasibility-guaranteed reward function design, to overcome DRL's limitations in guaranteeing feasibility and controlling exponentially-increased action space Simulations demonstrate that the proposed AC-based method enables a feasible, fast-converging, and dynamically adaptive solution and achieves promising energy-saving performance compared with other AC-DRL and heuristic algorithms. The publications corresponding to this chapter are listed below:

[**J1**] **Y. Yuan**, L. Lei, T. X. Vu, S. Chatzinotas, S. Sun and B. Ottersten, "Energy Minimization in UAV-Aided Networks: Actor-Critic Learning for Constrained Scheduling Optimization," in *IEEE Transactions on Vehicular Technology*, vol. 70, no. 5, pp. 5028-5042, May 2021.

[**C3**] **Y. Yuan**, L. Lei, T. X. Vu, S. Chatzinotas and B. Ottersten, "Actor-Critic Deep Reinforcement Learning for Energy Minimization in UAV-Aided Networks," in Proc. *European Conference on Networks and Communications (EuCNC)*, pp. 348-352, 2020.

In chapter 4, considering the well-learned ML model could be invalidated when environmental parameters have changed significantly, we investigate a meta-learning-based resource allocation scheme in LEO-terrestrial networks to save the re-training time and let the ML model fast adapt to the new environment. In order to serve more users and deliver a higher volume of requested data in over-loaded LEO systems with dense user distribution, two optimization-based resource scheduling schemes are proposed, i.e., optimal B&B algorithm and suboptimal alternating direction method of multipliers (ADMM)-based heuristic algorithm. Due to the combinatorial nature and the high complexity of the offline solutions, we train an AC learning model to make online decisions with the identical objective as the original problem. Considering unpredictable dynamic environments, the meta-learning technique is applied to enhance the adaptation ability of the AC model, i.e., meta-critic. In the proposed meta-critic-based algorithm, we design a tailored hybrid neural network for the critic to extract the features from the current and historical samples such that it has a good generalization ability to evaluate any new task when the environment changes. In addition, we integrate Wolpertinger policy to allow the actor to make efficient decisions in combinatorial action space. The numerical results verify the effectiveness and fast-response capabilities of the proposed method in adapting to dynamic environments, i.e., bursty user demands, dramatically fluctuated channel states, and user departure/arrival. Following are the submitted and accepted publications as part of the study in this chapter:

[**J2**] **Y. Yuan**, L. Lei, T. X. Vu, S. Chatzinotas, S. Sun, B. Ottersten, "Meta-Critic Learning for Dynamic-Adaptive Resource Scheduling in LEO-Terrestrial Systems," submitted to TWC, major revision, under the second round review.

[**J3**] L. Lei, **Y. Yuan**, T. X. Vu, S. Chatzinotas, M. Minardi and J. F. M. Montoya, "Dynamic-Adaptive AI Solutions for Network Slicing Management in Satellite-Integrated B5G Systems," in *IEEE Network*, vol. 35, no. 6, pp. 91-97, Nov. 2021.

In chapter 5, we study the ML application in distributed wireless systems, where the goal is to design an energy-efficient FL scheme. In this work, the energy conservation of FL is realized from two aspects. Firstly, we analyze the energy requirement of different learning models and verify that sparse or binary learning models, i.e., sparse neural network (SNN) and binary neural network (BNN), are able to reduce the energy consumption by simplifying the computation operations and curtailing the amount of data in communication. However, the conventional FL algorithms fail to handle the non-smoothness and constraints in the loss functions of SNN and BNN. To tackle this issue, we propose a federated proximal stochastic gradient descent algorithm that incorporates the advantages of the momentum acceleration, proximal terms, and convexity of the designed local sub-problem. Secondly, we formulate an optimization problem to further minimize the total energy consumption, where the wireless resources and learning parameters are optimized. In summary, we design a comprehensive FL scheme that integrates optimization, learning model selection, and the improved FL algorithms to reduce energy from multiple aspects and improve the learning performance. Numerical results show that the proposed FL scheme brings significant performance gain on energy-saving and outperforms other FL algorithms on learning accuracy and convergence for a general learning model. The accepted and to be submitted publications corresponding to this chapter are listed below:

[**C4**] Y. Yang, **Y. Yuan**, A. Chatzimichailidis, R. J. van Sloun, L. Lei, S. Chatzinotas, "ProxSGD: Training Structured Neural Networks under Regularization and Constraints," in Proc. *International Conference on Learning Representations (ICLR)*, Sep. 25, 2019.

[**J4**] **Y. Yuan**, L. Lei, Y. Yang, S. Chatzinotas, S. Sun, B. Ottersten, "Energy-Efficient Federated Learning in Wireless Systems with Sparse and Binary Neural Networks," to be submitted to IEEE Journal on Selected Areas in Communications (Multi-Tier Computing for Next Generation Wireless Networks)

In chapter 6, the main conclusions of the thesis are summarized and the future works are discussed.

1.4 Contributions Beyond Dissertation

During my Ph.D. period, I have contributed to a number of publications, which are not part of this thesis. The details are listed as follows:

[**J5**] L. Lei, E. Lagunas, **Y. Yuan**, M. Kibria, S. Chatzinotas, B. Ottersten, "Beam Illumination Pattern Design in Satellite Networks: Learning and Optimization for Efficient Beam Hopping," in *IEEE Access*, vol. 8, pp. 136655-136667, 2020.

[**J6**] **Y. Yuan**, L. Lei, T. X. Vu, S. Chatzinotas, S. Sun, B. Ottersten, "Actor-Critic Learning-Based Energy Optimization for UAV Access and Backhaul Networks," in *EURASIP Journal on Wireless Communications and Networking*, vol. 1, pp. 1-27, Dec. 2021.

[**C5**] L. Lei, E. Lagunas, **Y. Yuan**, M. G. Kibria, S. Chatzinotas and B. Ottersten, "Deep Learning for Beam Hopping in Multibeam Satellite Systems," in *IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, pp. 1-5, 2020.

[**C6**] **Y. Yuan**, L. Lei, T. X. Vu, S. Chatzinotas, B. Ottersten, "Resource Scheduling and Optimization for Over-Loaded Integrated LEO Satellite-Terrestrial Networks," accepted by *IEEE International Conference on Communications (ICC) 2022*.

Chapter 2

Centralized Learning: Supervised Learning-Assisted Resource Allocation

2.1 Introduction

For the upcoming 5G/B5G communication systems, developing intelligent transmission schemes and advanced resource scheduling algorithms are considered a key enabler to achieve the strict performance requirements 5G [67]. In a content delivery network, the system can usually be fully-loaded, e.g., in presence of severe interference among a large number of terminals with excessive data traffic requested [1], [2]. In this scenario, how to use limited available resources to efficiently serve the terminals at the cell edge is challenging. As one of the solutions, popular contents can be cached at the edge, such that most of the edge terminals can be served directly from the local cache. Otherwise, more resources, e.g., energy and time, are consumed if the terminals have to get the service from the macro base station (MBS) remotely [3].

On this track, a number of studies in the literature develop sophisticated scheduling algorithms to enable efficient content-delivery schemes. A related topic is transmission scheduling without power control. In [68], the authors investigated a minimum-length scheduling problem for generic wireless networks. The problem's NP-hardness, optimality conditions, and a set of scheduling algorithms have been studied. The authors in [69] developed a column-generation based scheduling algorithm to complete all the data transmission within a transmission deadline. For joint transmission scheduling with power control, the authors in [70] proposed centralized and distributed scheduling approaches to improve energy efficiency. In [71], the authors studied joint optimization of link scheduling and power control for energy minimization. Compared to the scheduling problem without power control, the joint optimization for user scheduling and power control is more challenging. This is because the two components, i.e., finding the optimal user groups and the optimal power allocation policy, are mutually coupled in decision making, which typically leads to a non-linear optimization [71] and thus results in difficulties to deliver optimal solutions [72].

In general, the scheduling optimization problems are hard to solve [68, 69]. It may not be practical to apply a sophisticated algorithm to real-time operations. Due to the problem's inherent hardness, the high computational complexity and long processing time limit the algorithm's applicability in practice. The execution time for performing an algorithm in real-time systems is typically stringent, e.g., during a scheduling period, one should return the optimized results within a short interval, e.g., seconds or milliseconds [67]. However, executing optimal algorithms may require considerable computational capabil-

ities and time [73]. An optimizer has to balance the algorithm's computation efficiency and the solution quality. In this chapter, from a machine-learning perspective, we investigate an alternative trade-off solution for joint scheduling and power control. As an emerging research area, integrating machine learning to resource optimization has received considerable research attention [29, 27, 74, 75, 76, 28, 73]. In [27, 74], the authors proposed a set of learning-based optimization approaches to improve the scheduling performance and reduce the computational complexity without considering power control. In [76, 28], the authors investigated machine learning-based approaches, e.g., training a fully-connected deep neural network (FC-DNN) or a logistic regression model, to address the resource scheduling problems in caching, multi-antenna, and non-orthogonal multiple access systems, respectively. As shown in [75], considerable efforts have been devoted to applying reinforcement learning to resource management in complex wireless networks. Recently, the authors in [28] used convolutional neural network (CNN) as a heuristic method to provide a fast solution for transmitting power control.

2.1.1 Contributions

In this chapter, we study two content-delivery problems, i.e., minimum-time and minimum-energy, in multiple-input single-output (MISO) systems, and solve the two problems under a unified learning-based framework. The following are the main contributions:

- We investigate a time minimization problem and an energy minimization problem via joint user grouping with power control and timeslot-group allocation, respectively. The formulated problems are non-linear mixed-integer programming (MIP) and linear integer programming (ILP).
- For the first problem, we provide an approach to decouple these two parts and obtain optimal solutions for performance benchmarking. For the second problem, B&B method can be used to provide the optimal solutions. The optimization approach, however, imposes an exponential computing complexity, which may limit its capability in real-time applications.
- To significantly reduce the computational time, we develop a learning-based algorithm to enable fast execution in decision making while providing competitive performance. Firstly by our analysis, we identify the features to be learned in resource allocation. Secondly, two deep-learning models, CNN and FC-DNN, are trained to learn the mapping between the channel gains and the optimal discrete decisions.
- Unlike [28], the output from the CNN or FC-DNN cannot directly provide a complete and feasible solution for solving the formulated problem. We then rely on the deep-learning model to tackle the combinatorial part in optimization which is most difficult and computational-heavy in resource allocation. We combine the predictions from CNN or FC-DNN with the optimal B&B algorithm to enable a near-optimal, feasible, and fast solution.
- The numerical results demonstrate the promising capabilities of the proposed learning approach in approximating optimal scheduling solutions and reducing computational time, compared to the conventional iterative algorithm B&B. In addition, for large-scale instances, the designed CNN achieves better performance than FC-DNN, in terms of prediction accuracy and computational time.

2.2 System Model

We consider a downlink MISO system including an L-antenna base station (BS) and K single-antenna users denoted as $\mathcal{K} = \{1, \ldots, k, \ldots, K\}$. All the users share a common communication channel in

transmission. The k-th user requests a content of length Q_k bits. We assume the wireless channel follows the block Rayleigh fading. In order to mitigate co-channel interference and improve the transmission efficiency, the dynamic user-group scheduling and precoding strategy are adopted. Let g denote a user group and $\mathcal{K}_g$ denotes the users included in group g. When group g is scheduled, the BS will deliver the requested contents to all the users in $\mathcal{K}_g$ until all the transmissions for $\mathcal{K}_g$ are completed. In total, there are $G = 2^K - 1$ possible candidate groups by enumerating all of the user groups, and the union of all the candidates is denoted by $\mathcal{G} = \{1, ..., g, ..., G\}$. For instance, when $K = 3$, we can list all the candidate groups $\{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}$.

The channel vector of user k is denoted by $\mathbf{h}_k \in \mathbb{C}^{L\times 1}$, where L is the number of antennas. We assume $\mathbf{h}_k$ follows circular-symmetric complex Gaussian distribution $\mathbf{h}_k \sim \mathcal{CN}(\mathbf{0}, \sigma_{h_k}^2 \mathbf{I}_L)$, where $\sigma_{h_k}^2$ is the parameter related to the path loss between the BS and user k. In order to mitigate inter-user interference, the BS precodes the data before serving the users. Denote x_k^g as the modulated signal and $\mathbf{w}_k^g \in \mathbb{C}^{L\times 1}$ as the precoding vector for user k in group g. The received signal at user $k \in \mathcal{K}_g$ is given as

$$y_k^g = \mathbf{h}_k^H \mathbf{w}_k^g x_k^g + \sum_{i\in\mathcal{K}_g\setminus\{k\}} \mathbf{h}_k^H \mathbf{w}_i^g x_i^g + n_k, \tag{2.1}$$

where the first and second terms in (2.1) represent the desired signal and the inter-user interference respectively. n_k is Gaussian noise with zero mean and variance σ^2. The signal-to-interference-plus-noise ratio (SINR) for user $k \in \mathcal{K}_g$ is given by $\mathrm{SINR}_k^g = \frac{|\mathbf{h}_k^H \mathbf{w}_k^g|^2}{\sum_{i\in\mathcal{K}_g\setminus\{k\}} |\mathbf{h}_k^H \mathbf{w}_i^g|^2 + \sigma^2}$. The achievable data rate can be expressed as

$$R_k^g = B \log_2 \left(1 + \mathrm{SINR}_k^g\right), k \in \mathcal{K}_g. \tag{2.2}$$

With respect to the delivery time, once the scheduling combination $\mathcal{C}$ is determined, the selected groups will be scheduled sequentially. Denote t_g as the transmission duration when group g is scheduled. The transmission for a scheduled group g will last until all the users' requests in $\mathcal{K}_g$ are satisfied. t_g is given as $t_g = \max_{k\in\mathcal{K}_g} \frac{Q_k}{R_k^g}$. The total delivery time is defined as $T_{tot} = \sum_{g\in\mathcal{C}} t_g$.

2.3 Problem Formulation

We consider a joint user scheduling and power allocation problem in order to minimize the total delivery time. The zero-forcing (ZF) precoding is adopted for each group due to its low computational complexity. For each scheduled group g, denote $\boldsymbol{H}_g$ as the channel coefficients from the BS's antennas to the users in group g, which is a $|\mathcal{K}_g| \times L$ matrix. Under the ZF design, the beamforming vector for user k in group g is of the form $\mathbf{w}_k^g = \sqrt{p_k^g}\tilde{\mathbf{w}}_k^g$, where p_k^g is the power allocation for user k in group g and $\tilde{\mathbf{w}}_k^g$ is the k-th column of the ZF precoding matrix $\mathbf{H}_g^H(\mathbf{H}_g\mathbf{H}_g^H)^{-1}$. It is worth noting that under the ZF design, the inter-user interference is fully mitigated, i.e., $|\mathbf{h}_k^H \mathbf{w}_j^g| = \delta_{kj}$. Then the achievable rate for user k in group g is given as

$$R_k^g = B \log_2 \left(1 + p_k^g/\sigma^2\right), k \in \mathcal{K}_g. \tag{2.3}$$

The transmit power of user k is $p_k^g \beta_k^g$, where $\beta_k^g = \| \tilde{\mathbf{w}}_k^g \|^2, \forall k \in \mathcal{K}_g$.

2.3.1 Time Minimization Joint Scheduling and Power Allocation Problem

The joint user scheduling and power allocation is formulated in $\mathcal{P}_0$. We introduce two sets of variables, $z_g \in \{0,1\},\ \forall g \in \mathcal{G}$ and $p_k^g,\ \forall k \in \mathcal{K}, \forall g \in \mathcal{G}$. The binary variable z_g is used to indicate if group g is scheduled ($z_g = 1$) or not ($z_g = 0$). The continuous variable $p_k^g > 0$ represents the power allocation among the users in group g.

$$\mathcal{P}_0 : \min_{\substack{z_g \in \{0,1\} \\ p_k^g > 0}} \sum_{g \in \mathcal{G}} z_g \max_{k \in \mathcal{K}_g} \frac{Q_k}{R_k^g} \tag{2.3a}$$

$$s.t.\ R_k^g \geq \eta_k, \forall k \in \mathcal{K}_g, \forall g \in \mathcal{G}, \tag{2.3b}$$

$$\sum_{k \in \mathcal{K}_g} \beta_k^g p_k^g \leq P_{tot}, \forall g \in \mathcal{G}, \tag{2.3c}$$

$$\sum_{g \in \mathcal{G}} a_{kg} z_g = 1, \forall k \in \mathcal{K}. \tag{2.3d}$$

The objective in $\mathcal{P}_0$ is to minimize the total data transmission length, where the duration of scheduling group g depends on $\max_{k \in \mathcal{K}_g} \frac{Q_k}{R_k^g}$. Constraint (2.3b) is to guarantee users' minimum rate η_k in their transmission period. The second constraint (2.3c) restricts that the power consumption in each group cannot exceed power budget P_{tot}. In constraints (2.3d), we consider that each user is scheduled once to reduce the implementation complexity and the signaling overhead in practice, where binary parameters $a_{kg} = 1$ indicates group g contains user k, otherwise 0.

In general, $\mathcal{P}_0$ is a mixed integer non-linear programming problem. To enable the optimal solution, we observe that the problem can be decomposed to two steps. In the first step, we enumerate all the groups and optimize the power allocation within each group such that the QoS constraints (2.3b) and (2.3c) are satisfied for each group and users. In the second step, we select a combination of groups, which leads to the minimum transmission time, and covers all the users. In addition, each user appears only once in the combination.

Firstly, for each group g, the intra-group power allocation problem can be formulated in $\mathcal{P}_1(g)$:

$$\mathcal{P}_1(g) : \min_{p_k^g > 0} \max_{k \in \mathcal{K}_g} \frac{Q_k}{R_k^g} \tag{2.4}$$

$$s.t.\ R_k^g \geq \eta_k,\ \forall k \in \mathcal{K}_g, \tag{2.4a}$$

$$\sum_{k \in \mathcal{K}_g} \beta_k^g p_k^g \leq P_{tot}. \tag{2.4b}$$

By introducing a variable T_g, problem $\mathcal{P}_1(g)$ is equivalent to the following problem:

$$\mathcal{P}_1'(g) : \min_{p_k^g > 0, T_g \geq 0} T_g \tag{2.5}$$

$$s.t.\ T_g \geq \frac{Q_k}{\log_2(1 + p_k^g/\sigma^2)},\ \forall k \in \mathcal{K}_g, \tag{2.5a}$$

$$\log_2(1 + p_k^g/\sigma^2) \geq \eta_k,\ \forall k \in \mathcal{K}_g, \tag{2.5b}$$

$$\sum_{k \in \mathcal{K}_g} \beta_k^g p_k^g \leq P_{tot}. \tag{2.5c}$$

The constraint (2.5a) is equivalent to, $\log_2(1 + p_k^g/\sigma^2) - \frac{Q_k}{T_g} \geq 0, \forall k \in \mathcal{K}_g$, which is a convex con-

straint. Thus $\mathcal{P}_1'(g)$ is a convex problem, and can be solved by standard tools [77]. The next optimization task is to select a group combination leading to the minimum transmission time and covering all the users once. The problem can be formulated by an ILP problem in $\mathcal{P}_2$.

$$\mathcal{P}_2: \min_{z_g \in \{0,1\}} \sum_{g \in \mathcal{G}} z_g t_g \tag{2.6}$$

$$s.t. \sum_{g \in \mathcal{G}} a_{kg} z_g = 1, \ \forall k \in \mathcal{K}. \tag{2.6a}$$

After solving $\mathcal{P}_2$, the solution z_g can be used for generating the optimal group combination $\mathcal{C}_{opt}$ by deleting the unscheduled groups ($z_g = 0$) from $\mathcal{G}$. The optimal method is summarized as Alg. 1.

Algorithm 1 Optimal Algorithm For Time Minimization Problem

Inputs: $Q_1, \ldots, Q_K$, $\mathbf{h}_1, \ldots, \mathbf{h}_K$, $\eta_1, ..., \eta_K$, and P_{tot}

1: **for** each $g \in \mathcal{G}$ **do**
2: Solve $\mathcal{P}_1'(g)$ and obtain $p_k^g, k \in \mathcal{K}_g$
3: Compute t_g based on $\max_{k \in \mathcal{K}_g} \frac{Q_k}{R_k^g}$
4: **end for**
5: Solve $\mathcal{P}_2$ and obtain $z_g, g \in \mathcal{G}$
6: Use the groups with $z_g = 1$ to form a combination $\mathcal{C}_{opt}$

Outputs: $\mathcal{C}_{opt}$ and power allocation $p_k^g, k \in \mathcal{K}_g, g \in \mathcal{C}_{opt}$

By our characterizations, the optimal solution of $\mathcal{P}_0$ can be obtained by solving a convex problem for each group along with solving an ILP problem $\mathcal{P}_2$. Conventionally, the B&B algorithm is a straightforward way to obtain a global optimum for ILP, where a linear relaxation problem is solved at each node of the branch-and-bound tree. The whole problem is inherently hard and with high complexity since exponential number of groups need to be optimized by $\mathcal{P}_1'(g)$. Moreover, the subproblem $\mathcal{P}_2$ is equivalent to an exact cover problem which is known as NP-complete [78]. Alg. 1 can be used as an offline optimization method to provide optimal solutions, and for performance benchmarking.

2.3.2 Energy Minimization by Group-Timeslots Allocation

To minimize energy consumption for the system, an optimization problem $\mathcal{P}_3$ is formulated by deciding the number of timeslots assigned to each group with fixed transmit power p_k and limited time resources. We introduce integer variables $y_g \in \{0, 1, ..., T\}$ to indicate the number of used time slots for group g.

$$\mathcal{P}_3: \min_{y_g \in \mathbb{Z}} \sum_{g \in \mathcal{G}} \Phi y_g p_g \tag{2.7}$$

$$s.t. \sum_{g \in \mathcal{G}} \Phi y_g R_k^g \geq Q_k, \ \ \forall k \in \mathcal{K}, \tag{2.7a}$$

$$\sum_{g \in \mathcal{G}} y_g \leq T, \tag{2.7b}$$

$$y_g \in \{0, 1, ..., T\}, \ \ \forall g \in \mathcal{G}. \tag{2.7c}$$

Different from $\mathcal{P}_0$, the allocated time for each user is discretized to the time slots. We denote the duration of each slot as a constant Φ such that the objective function represents the energy consumption for all the groups. The constraints (2.7a) mean that the transmitted data should satisfy the users' data demands. Considering the time limitations, in constraint (2.7b), all the content delivery tasks need to be completed within T time slots. The constraints (2.7c) confine all the variables to the finite integer values.

$\mathcal{P}_3$ is an ILP problem. Although the $\mathcal{P}_3$ is simpler than $\mathcal{P}_0$ and can be solved directly by B&B, the NP-hardness still exists. For both $\mathcal{P}_0$ and $\mathcal{P}_3$, since the computational complexity of the optimal algorithm increases exponentially with the input size, the required computational time dramatically increases. To deal with this issue in real-time applications, next, we propose a learning-based approach.

2.4 Scheduling Algorithm Design based on Deep Learning

In order to overcome the high complexity in obtaining the optimum, we design a supervised deep learning-based approach to provide high-quality and computational-efficient solutions. The optimization decisions in $\mathcal{P}_0$ and $\mathcal{P}_3$ consist of two parts. One is to select the best groups from an exponential number of candidates. For example, the binary variable $z_g = 0$ or the integer variable $y_g = 0$ mean the group g is not selected. The other is to determine the power or time slots allocation to these selected groups. We remark that the major difficulties are from the first part which is computationally heavy. Once the scheduled groups have been decided, the remaining problem is relatively easy to solve. By analyzing the optimal decisions, we observe that there exists a pattern between the spatial features of channel coefficients and the decisions of optimal groups. For example, two users located distantly or with significant differences in channel coefficients, are more likely to be grouped in a time slot at the optimum. Thus, we treat the grouping information, i.e., the most promising group combinations with a high probability to be scheduled, as the feature to be learned and predicted by the deep-learning models. The optimal algorithm in Section 2.3 is used to generate training sets. The after-trained learning model is then applied to predict the scheduled groups.

2.4.1 Features to be Learned

We denote the training data as $(\mathbf{x}, \mathbf{y})$. The input $\mathbf{x}$ refers to the channel vector $\mathbf{h}_k \in \mathbb{C}^{L\times 1}$, users' demands $Q_1, ..., Q_K$, users' rate requirements $\eta_1, ..., \eta_K$, and the maximum power P_{tot}. For the outputs, they can not be treated as the optimal solution $z_1, ..., z_G$ directly, since $\mathcal{P}_0$ and $\mathcal{P}_3$ have an exponential number of variables and have constraints to be satisfied. This introduces difficulties of achieving solution feasibility and good prediction performance. For instance, the learning model may determine to schedule several groups. These groups could violate constraints (2.3d). Instead, the output node is designed as the possibility of using a group combination. The reason is that due to the system limitations, e.g., number of antennas, the number of all the group combinations, denoted by N, can be much less than the number of groups G. Moreover, the feasibility issue of constraints (2.3d) can be resolved in each combination. By our design, we first list all of the candidate combinations as a union $\boldsymbol{\mathcal{C}} = \{\mathcal{C}_1, ..., \mathcal{C}_N\}$. Then the output of the learning model is organized in a N-dimension binary vector $\mathbf{y} = [y_1, ..., y_N]$. If $y_n = 1$, it means the combination $\mathcal{C}_n$ is selected and all the groups in combination $\mathcal{C}_n$ are scheduled. For example, when $K = 3$ and $L = 2$, the candidate groups are $\{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}$ and $G = 6$. Note that the group $\{1, 2, 3\}$ is excluded due to the practical limit of $|\mathcal{K}_g| \leq L$. The candidate combinations are $\{\{1\}, \{2\}, \{3\}\}, \{\{1, 2\}, \{3\}\}, \{\{1, 3\}, \{2\}\}, \{\{2, 3\}, \{1\}\}$ and $N = 4$. If the combination $\{\{1, 2\}, \{3\}\}$ is selected, vector $\mathbf{y}$ is $[0, 1, 0, 0]$ in the training set.

For preparing the training set, we generate the instances with different channel conditions, demands, rate requirements and power limits. After sufficient training, the learning model is able to predict the

grouping information $\mathbf{y}$. That is, given a new input $\hat{\mathbf{x}}$, the after-trained learning model will provide the estimated scheduling decision $\hat{\mathbf{y}}$. However, the original DNN output is not binary which cannot serve as an indicator for user scheduling. To solve the problem, we use the sigmoid function [79] in the output layer to limit the value between 0 and 1. Then, a rounding operation is adopted to convert fractional values to binary. More specifically, we use M as the mean of $\hat{\mathbf{y}}$. If any fractional value $\hat{y}_n > \alpha M$, we set $\hat{y}_n = 1$, otherwise zero, where $\alpha > 0$ is a control parameter. By design, smaller α increases the likelihood of "1" in $\hat{\mathbf{y}}$, which also contributes to the improvement of prediction accuracy.

2.4.2 Adopted Learning Models

To establish a predicting system, several machine-learning models might be adopted, such as logistic regression, support vector machine, and neural networks. In this chapter, we adopt FC-DNN and CNN to learn the relations between channel coefficients and the optimal group combinations. FC-DNN can be effective to capture the nonlinear input-output relations. CNN has been widely used to extract spatial features for image classification and has high computational efficiency with the parameter sharing [80].

FC-DNN Structure

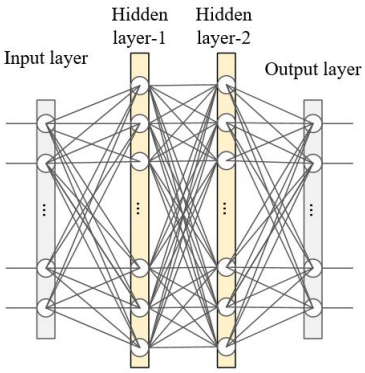


Figure 2.1: The illustration of FC-DNN structure

Fig. 2.1 demonstrate the structure of an FC-DNN with 2 hidden layers.

- Input layer. In FC-DNN, the input data should be expanded into a column vector. In $\mathcal{P}_0$, as we take the channel information, users' demands, users' rate requirements and the maximum power as the input data, which can be simply organized as:

$$\boldsymbol{x} = [h_{1,1}, \cdots, h_{l,k}, \cdots, h_{L,K}, Q_1, \cdots, Q_K, \eta_1, \cdots, \eta_K, P_{tot}]. \tag{2.8}$$

  Thus the number of the input nodes is $K(L+2)+1$.

- Hidden layer. The function of hidden layers is to identify features from the input data which can be used to correlate between a given input and the real output. The number of hidden layers and the number of neurons in each hidden layer reflect the depth of the neural network. It is necessary to design a suitable depth for the neural networks to prevent overfitting and excessive computational complexity.

- Output layer. By our design, the output should carry the estimated information for the optimal grouping decisions, which is referred to as the K-dimension binary vector $\hat{\boldsymbol{y}}$.

CNN Structure

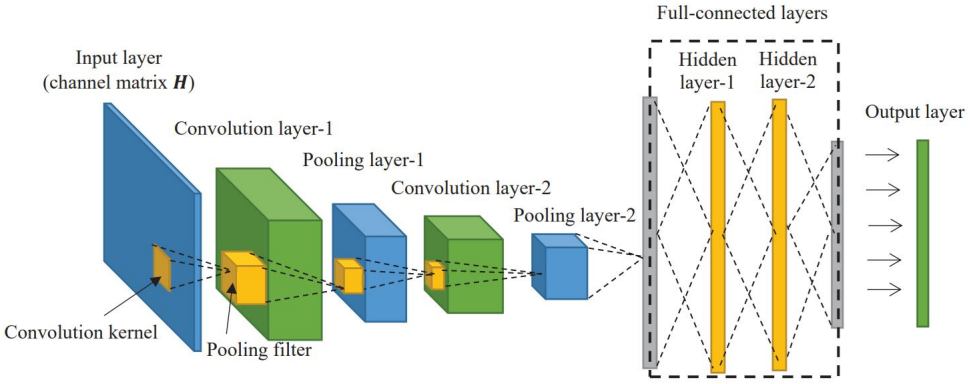


Figure 2.2: The illustration of CNN structure

Fig. 2.2 illustrates the structure of the adopted CNN which consists of the following five main components.

- Input layer. In CNN, the input data is reorganized as an image-like 3-dimension matrix. The first two dimensions represent the image's length and width while the last dimension refers to the depth. For $\mathcal{P}_3$, the input data refers to the matrix $\boldsymbol{X}$ below. To facilitate the training process, the channel coefficients can be further normalized and converted to dB [28].

$$\boldsymbol{X} = \begin{bmatrix} h_{1,1} & \cdots & h_{1,K} \\ \vdots & \cdots & \vdots \\ h_{L,1} & \cdots & h_{L,K} \\ Q_1 & \cdots & Q_K \\ \eta_1 & \cdots & \eta_K \\ P_{tot} & \cdots & 0 \end{bmatrix}. \tag{2.9}$$

  As the depth of the matrix $\boldsymbol{X}$ is 1, the size of the input is $(L+3) \times K \times 1$.

- Convolution layer. As shown in Fig. 2.2, each neuron in the convolution layer only connects a squared part of the previous layer. This squared core is called filter or convolution kernel. By our design, we use two convolution layers with a 3×3 and a 2×2 convolution kernel respectively. The processing depth is set to 3. It means that the kernel enables to transfer the node matrix into a 3-tier unit node by convolution. Then the kernel will move around to cover the entire image with a fixed step. The convolution layers try to analyze the data of each kernel for obtaining the features with a higher level of abstraction. More than that, since the weights are shared via the convolution kernel, the number of parameters can be significantly reduced in the neural network.

- Pooling layer. The Pooling layer is used to further decrease the size of the output matrix from the previous convolution layer. Similar to the convolution layer, the pooling layer also adopts a filter to convert a node matrix into a unit node. The pooling filter applies the maximizing or averaging operations instead of convoluting operations.

- Fully-connected layer. The convoluting and pooling can be regarded as a process of automatic feature extraction. After that, fully-connected (FC) layers are also needed to generate the final output. The structure of the FC layer is identical to FC-DNN which includes several hidden layers.

- Output layer. The output of CNN is the same as the output of FC-DNN.

2.4.3 Algorithm Summary

The neural network-based approach is summarized in Algorithm 2. It is worth noting that there are three cases with regard to $\hat{\mathbf{y}}$ after rounding. The ideal case is $\| \hat{\mathbf{y}} \| = 1$, which means only one element is 1. Then the scheduled groups are from this only combination. In the second case, vector $\hat{\mathbf{y}}$ may have multiple elements with value "1". Hence, we design a re-selection scheme to make final decisions. Firstly, we delete a considerably large amount of combinations from $\boldsymbol{\mathcal{C}}$ according to $\hat{\mathbf{y}}$. The remaining combinations compose a reduced candidate list $\boldsymbol{\mathcal{C}}^*$. For example, if $\hat{\mathbf{y}} = [0, 1, 1, 0]$, then the combination list $\boldsymbol{\mathcal{C}} = \{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4\}$ is reduced to $\boldsymbol{\mathcal{C}}^* = \{\mathcal{C}_2, \mathcal{C}_3\}$. After that, we recall Algorithm 1 with the restricted set $\boldsymbol{\mathcal{C}}^*$ and obtain the final scheduling combination $\mathcal{C}_{dnn}$ more efficiently. In the third case, DNN may return an all-zero vector $\hat{\mathbf{y}}$. For dealing with this case, we then adjust the control parameter α as follows to enable at least one element with the value "1".

Algorithm 2 Learning-Based Approach

Inputs: $\boldsymbol{\mathcal{C}}, Q_1, \ldots, Q_K, \mathbf{h}_1, \ldots, \mathbf{h}_K, \eta_1, ..., \eta_K$, and P_{tot}.

1: Given a new input $\hat{\mathbf{x}}$ to the well-trained FC-DNN or CNN.
2: Apply rounding operations with α then obtain $\hat{\mathbf{y}}$.
3: **if** $\| \hat{\mathbf{y}} \| = 0$ **then**
4: Reduce α until $\| \hat{\mathbf{y}} \| \geq 1$.
5: **end if**
6: **if** $\| \hat{\mathbf{y}} \| > 1$ **then**
7: Delete combinations from $\boldsymbol{\mathcal{C}}$ according to the zero elements in $\hat{\mathbf{y}}$.
8: Obtain the restricted set $\boldsymbol{\mathcal{C}}^*$ and obtain the reduced candidate groups $\mathcal{G}^*$.
9: **else**
10: Choose the only combination as $\mathcal{C}^*$ and obtain the reduced candidate groups $\mathcal{G}^*$.
11: **end if**
12: Based on $\mathcal{G}^*$, apply Alg. 1 for $\mathcal{P}_0$ or B&B for $\mathcal{P}_3$.

Outputs: power allocation p_k^g for $\mathcal{P}_0$ or time slots allocation z_g for $\mathcal{P}_3$.

2.5 Performance Evaluation

In this section, we evaluate the performance of the deep learning-assisted approaches, in terms of computational time, and performance gaps between the proposed learning approach and the optimal solutions. In the simulation, the BS is equipped with up to 5 antennas, serving up to 10 users. We set $\sigma^2 = 0.01, B = 1$MHz and $\sigma_{h_k}^2 = 1, \forall k$. We limit the cardinality of each group to the number of antennas. If $L = 5$ and $K = 10$, the number of candidate groups is $G = C_{10}^5$ and the number of the group combinations is $N = \frac{C_{10}^5}{2}$. For neural networks implementation, TensorFlow framework is used for building the neural network in Python. During the training phase, mean square error (MSE) is adopted as the loss function which is minimized by the Adam method [79]. Besides, a regularization item is introduced to prevent overfitting. As shown in the results, 100 test sets are used for evaluation.

The average accuracy of the predictions in CNN and FC-DNN is evaluated as follows,

$$\frac{1}{100}\sum_{i=1}^{100}\frac{V_{dl}^{i}-V_{opt}^{i}}{V_{opt}^{i}}, \tag{2.10}$$

where V_{dl} and V_{opt} are the derived objective values from the deep-leaning approaches, i.e., FC-DNN and CNN, and the optimal algorithm, respectively. Since the selected parameters of DNN are able to affect the satisfactory performance, we perform some pretests for evaluating several parameters, e.g., the number of hidden layers and neurons per layer. The FC-DNN and CNN settings are summarized in Table 2.1 and Table 2.2, respectively.

Table 2.1: DNN settings

Parameter	Value
Number of input nodes	$K(L+2)+1$
Number of hidden layers	3
Number of nodes for hidden layers	300
Number of output nodes	N
Activation function in hidden layers	Relu
Activation function in output layer	Sigmoid
Loss function	MSE
Optimizer	Adam optimization

Table 2.2: CNN settings

Parameter	Value
Dimension of input layer	$K \times (L+3)$
Convolution layer-1 kernel	3×3
Convolution layer-2 kernel	2×2
Convolution layer step	1
Convolution layer depth	3
Pooling layer filter	2×2
Pooling layer filter	2
Pooling method	max pooling [29]
Number of nodes for hidden layer-1	200
Number of nodes for hidden layer-2	200
Number of output nodes	N
Activation function in hidden layers	Relu
Activation function in output layer	Sigmoid
Loss function	MSE
Optimizer	Adam optimization

To demonstrate the computation efficiency of the learning-assisted approach, the CPU time (in seconds) of the DNN and the optimal method are compared in Table 2.3. The computational time in FC-DNN test phase (CPU time for executing lines 1-2 in Algorithm 2) is insensitive to the problem's scale. It keeps at the same magnitude in both cases. In general, the computational time of the proposed FC-DNN approach (CPU time for executing lines 1-12 in Algorithm 2) is dramatically reduced compared to the optimal algorithm. We remark that the total CPU time in Algorithm 2 varies with three cases of $\| \hat{\mathbf{y}} \|$. It

is noted that, in the case of $\| \hat{\mathbf{y}} \| > 1$, the time is more than the other two cases. This is because when $\| \hat{\mathbf{y}} \| > 1$, the power/time slots allocation problem needs to be solved for more groups. The CNN method and FC-DNN have similar results in computational time.

Table 2.3: CPU time in computation

Algorithmic Solutions	$K = 4$ $L = 2$	$K = 10$ $L = 5$
Algorithm 1	6.031	354.140
DNN test phase in Alg. 2	0.083	0.139
Alg. 2 for case 1, $\| \hat{\mathbf{y}} \| = 1$	1.211	2.910
Alg. 2 for case 2, $\| \hat{\mathbf{y}} \| > 1$	2.451	32.145
Alg. 2 for case 3, $\| \hat{\mathbf{y}} \| = 0$	1.330	3.039

Fig. 2.3 shows the performance of the FC-DNN-assisted approach on $\mathcal{P}_0$ compared to the optimal solution, with increasing the training set size. The performance gaps on the vertical axis represent the relative delivery time of the FC-DNN-assisted method with respect to the optimum value. In general, using more data in the training set improves the prediction accuracy of the FC-DNN-based method. In particular, the performance reaches the optimum when the size of the training set is around 800, leading to the smallest gap around 8%, 9% and 10% in the case of $\alpha = 2.4$, $\alpha = 2.6$ and $\alpha = 2.8$ respectively.

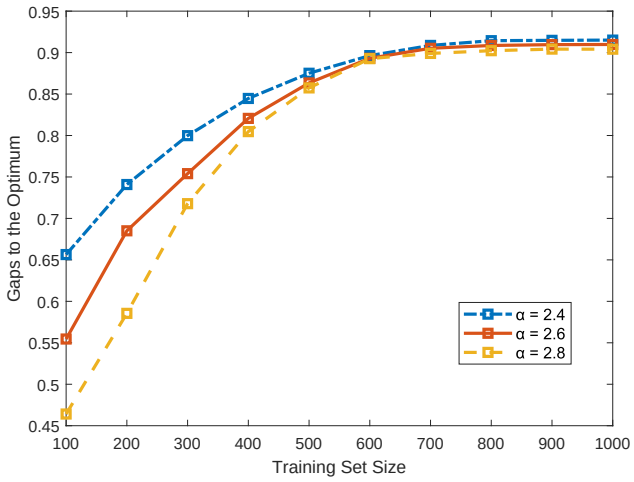


Figure 2.3: Average gaps between DNN and optimum ($K = 10$, $L = 5$)

Fig. 2.4 presents the difference of the total data transmission length T_{tot} between the DNN-assisted approach and the optimal method on all the tested samples of $\mathcal{P}_0$. Over 100 testing samples, the average gaps of objective values between the two algorithms are small, around 8%. This shows that the DNN approach is able to provide a near-optimal solution with high computational efficiency.

Fig. 2.5 demonstrate the energy consumption by performing the CNN-based approach, the DNN-based approach and the optimal algorithm for the cases of $K = 5$, 10 and 15 in $\mathcal{P}_3$, respectively. As shown in the results, 100 test sets are used for evaluation. The average prediction accuracy of the CNN-based approach achieves 99.66%, 97.27%, and 96.08% in the cases of $K = 5$, 10, and 15, respectively, whereas the performance in FC-DNN slightly drops to 99.09%, 95.16%, and 93.78%. Recalling the computational time in Table. 2.3, the learning-based approaches are able to provide a near-optimal solution with high computational efficiency.

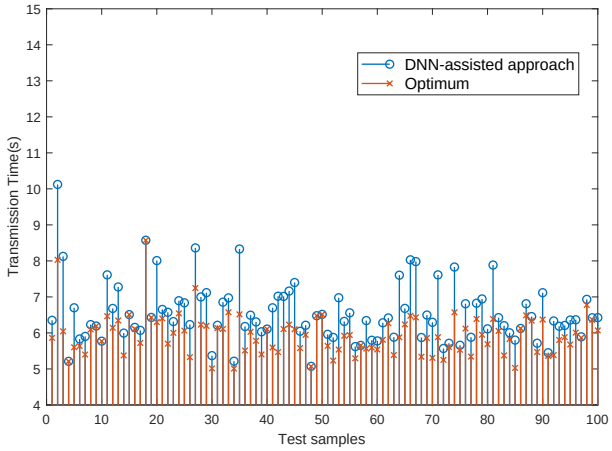
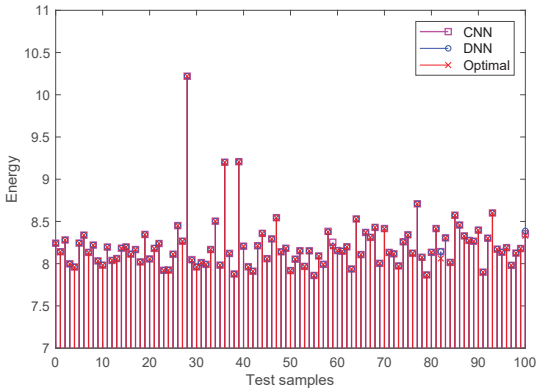
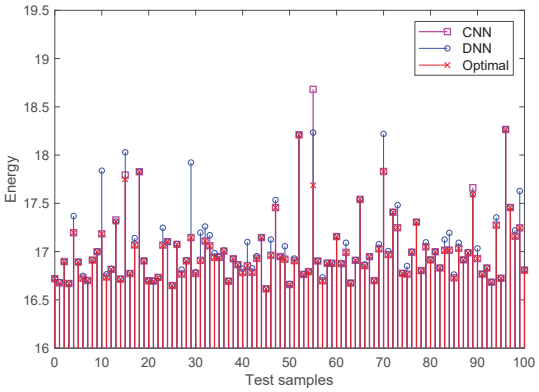


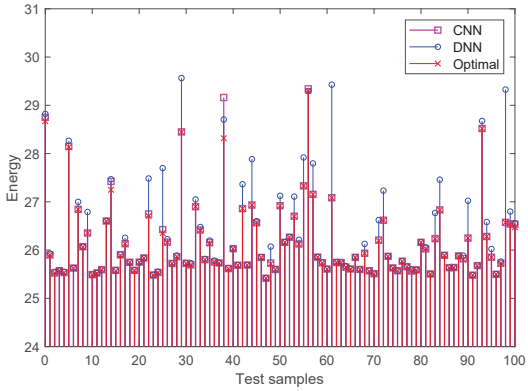
Figure 2.4: Transmission time comparison between the DNN-based algorithm and the optimal algorithm ($K = 10$, $L = 5$, $\alpha = 2.8$, training set size$= 800$)



(a) $K = 5$



(b) $K = 10$



(c) $K = 15$

Figure 2.5: Energy comparison between the learning-based algorithms and the optimal algorithm ($K = 5, 10, 15$, $L = 5$ $\alpha = 2.8$, training set size$= 800$)

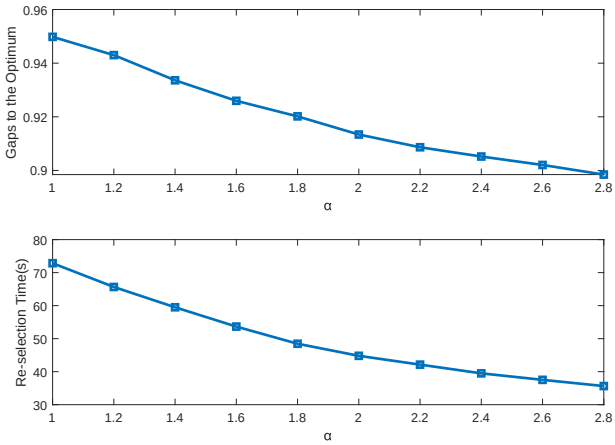


Figure 2.6: Optimality and re-selection time with different α ($K = 10$, $L = 5$, training set size$= 800$)

Next, we evaluate the effect of α on the performance gap and the computation complexity in the re-selection phase. As shown in Fig. 2.6, the time monotonically decreases with the growth of α, whereas the performance of the FC-DNN in approaching the optimum is degraded. As we analyzed in section 2.4, when α is large, a small number of combinations will be in the restricted candidate set $\boldsymbol{\mathcal{C}}^*$, thus leading to less computational time. On the other hand, the fewer candidates in $\boldsymbol{\mathcal{C}}^*$ can possibly result in higher a probability of loss in optimality.

2.6 Conclusion

We developed an ML-based approach for resource allocation to enable a fast, feasible, and near-optimal solution for both time-efficient and energy-efficient content delivery in MISO systems. We formulated a non-linear MIP and an ILP problem via power control and time slots allocation, respectively. In order to obtain the global optimum as the performance benchmark, optimal methods with exponential complexity were designed. Toward an efficient solution, we designed FC-DNN and CNN-based approaches to approximate the combinatorial decisions. Different from the conventional pure ML algorithm, the proposed learning-assisted approach applies NNs to predict the learnable features which can be used to accelerate the optimization process and improve the learning accuracy. Numerical results show that both learning-assisted approaches can well-approximate the optimum with low computation complexity. The CNN-based method slightly outperforms FC-DNN in the respect of prediction accuracy in a large-scale scenario.

Chapter 3

Centralized Learning: Reinforcement Learning-Based User Scheduling

3.1 Introduction

Unmanned aerial vehicles (UAVs) have attracted much attention to high-speed data transmission in dynamic, distributed, or plug-and-play scenarios, e.g., disaster rescue, live concert, or sports events [81]. However, UAVs' limited endurance, energy supply, and storage become critical issues for their applications, which motivates the study of energy efficiency in UAV-aided communication networks. The UAV's energy consumption comes from two aspects, propulsion energy for flying and hovering, and communication energy for data transmission. The flying energy mainly depends on the UAV's velocity and trajectory [81]. The hovering energy is, in general, proportional to the hovering time. Compared to the propulsion energy, the communication energy consumption is not a negligible part, e.g., considerable communication energy can be consumed in the scenarios with high traffic requests from a large number of users. Thus joint energy optimization for both parts is necessary and has attracted considerable attention in the literature [82, 83, 84, 85, 86, 87, 88, 89].

The authors in [83, 84] maximized the energy efficiency, referring to the ratio between transmitted data and propulsion energy. In [85], the authors introduced a complete UAV energy model and proposed a user-timeslot scheduling method to minimize the sum of the propulsion energy and communication energy. Based on the energy model in [85], the authors formulated an energy minimization problem with latency constraints by trajectory design in [82]. The above works in [83, 84, 85, 82] adopted a time division multiple access (TDMA) mode, where the UAV serves one user per timeslot. Besides TDMA, space division multiple access (SDMA) enables simultaneous data transmission to multiple users, such that the hovering time and hovering energy can be reduced. In [86], the authors designed an SDMA-based beamforming scheme to minimize the total transmit power for multi-antenna UAVs. In [87], an energy efficiency maximization problem was investigated in an SDMA-based multi-antenna UAV network via optimizing the flying velocity and power allocation. However, serving multiple users simultaneously may lead to strong inter-user interference and may require more communication energy to fulfill users' demands. In [88, 89], two non-convex combinatorial optimization problems based on SDMA were studied. The authors in [88] proposed an alternative optimization algorithm based on the block coordinate descent method to optimize frequency, transmit power, and UAV's trajectory. In [89], the non-convex problem becomes convex by fixing a sensing-time variable. Then, a single-variable search method was adopted to optimize.

Deterministic optimization algorithms, e.g., [82, 83, 84, 85, 86, 87, 88, 89], might not be suitable for fast decision making in a dynamic wireless environment. To address this issue, deep learning-based solutions have been investigated in the literature. The authors in [90] relied on a deep neural network (DNN) to efficiently predict the resource allocation scheme for mobile edge networks. In [91], a deep learning-based auction algorithm was proposed to determine a dynamic battery charging scheduling for UAV-aided systems. Supervised learning, such as DNN, requires large amounts of training data, which is a non-trivial task in an offline manner [74]. Another category is reinforcement learning (RL). In [92], Q-learning was applied to solve an unconstrained UAVs trajectory design problem to avoid collisions. However, updating the Q-table results in unaffordable computing time/resources. To improve efficiency, deep reinforcement learning (DRL) was developed with the following advantages. Firstly, DRL provides timely solutions, adapted to environment variations. Secondly, DRL integrates DNN to make decisions and improve solution quality. Thirdly, DNN requires an offline data generating and training phase, whereas DRL is less needed for prior knowledge and is able to train by exploring unknown environments and exploiting received feedbacks in an online manner. In [93], the authors applied a deep Q network (DQN) to design an energy-efficient flying trajectory scheme for UAV-aided networks. In general, DQN is used to deal with a relatively small and discrete action space, where the action space refers to the set of all possible decisions [57]. The authors in [94] designed a different deep Q-learning architecture with a high dimensional action space, but it needs to evaluate all of the actions before making a decision, which is time-consuming.

Deep actor-critic is an emerging DRL method with fast convergent properties and the capability to deal with a large action space [95]. In [96], an actor-critic-based DRL (AC-DRL) algorithm was proposed to reduce the UAV's energy consumption and enhance the UAV's coverage of ground users via optimizing UAV's flying direction and distance. In [97], the authors employed deep actor-critic to design a learning algorithm for UAV-aided systems, considering energy efficiency and users' fairness. Note that the AC-DRL in [96, 97] was developed for unconstrained problems. However, most of the problems in UAV systems are constrained and with discrete variables. The conventional AC-DRL algorithms have limitations on tackling constrained combinatorial optimization problems, which may result in slow convergent, infeasible, and degraded solutions. The authors in [98, 99] developed AC-DRL algorithms for a combinatorial optimization problem in a UAV-aided system, where the performance is limited when the problem scale grows. In [100, 101], two AC-DRL algorithms based on deep deterministic policy gradient (DDPG) were developed to optimize UAV trajectory and resource allocation. The adopted reward function can satisfy simple constraints but might not be applicable for complicated combinatorial problems.

In this study, we minimize the UAV's communication and propulsion energy in a downlink UAV-aided communication system. The improvement of solution development lies in three aspects. Firstly, compared to offline optimization approaches, we provide online learning and timely energy-saving solutions based on DRL. Secondly, unlike the conventional DRL or AC-DRL methods, the proposed solution is suited to tackle constrained combinatorial optimization. Thirdly, compared to our previous work [99], we augment the algorithms by developing new theoretical results and tailored approaches to address two challenging issues in guaranteeing feasibility and controlling exponentially-increased action space. The major contributions are summarized as follows:

- We formulate an energy minimization problem for an SDMA-enabled UAV communication system, where user-timeslot allocation and UAV's hovering time assignment are the coupled optimization tasks. The formulated problem is combinatorial and non-convex with bilinear constraints.

- We provide a relax-and-approximate method to approach the optimum. That is, the bilinear terms are addressed by McCormick envelopes relaxation, then the remaining integer linear programming

problem is solved by the branch-and-bound (B&B) algorithm.

- We characterize the interplay among communication energy, hovering time, and hovering energy. Based on the derived analytical results, we develop a golden section search-based heuristic (GSS-HEU) algorithm for benchmarking general instances with lower complexity than the optimal solution.
- We propose an actor-critic-based deep stochastic online scheduling (AC-DSOS) algorithm for UAV energy savings, where the original problem is transformed into a Markov decision process (MDP). AC-DSOS is computationally light and solves the problem in an online manner, against offline high-complexity optimal/sub-optimal algorithms. Unlike conventional DRL, we develop a set of tailored approaches in AC-DSOS, e.g., stochastic policy quantification, action space reduction, and feasibility-guaranteed reward function design, to overcome DRL's limitations in addressing combinatorial optimization problems with multiple constraints and large action space.
- Simulations demonstrate that the proposed AC-DSOS enables a feasible, fast-converging, and dynamically adaptive solution. The designed approaches are effective in reducing action space and guaranteeing feasibility. AC-DSOS achieves promising energy-saving performance compared with two recent AC-DRL methods and three heuristic algorithms.

The rest of the chapter is organized as follows. Section 3.2 provides the system model and Section 3.3 formulates the considered optimization problem. In Section 3.4, we analyze the relationship between the energy consumption and hovering time, and propose a heuristic algorithm. In Section 3.5, we reformulate the problem as an MDP and develop an AC-DSOS algorithm. Numerical results are presented and analyzed in Section 3.6. Finally, we draw the conclusions in Section 3.7.

3.2 System Model and Problem Formulation

3.2.1 System Model

We consider a downlink UAV-aided communication system. A UAV serves as an aerial base station (BS) to deliver data to ground users, e.g., for the scenarios if terrestrial BSs are unavailable or overloaded by high traffic demand from numerous users. We assume that the UAV is equipped with L antennas and each ground user has a single antenna [87]. The UAV is fully loaded with data and energy at a dock station before the task starts. The service area is divided into N clusters considering the UAV's limited coverage area. This setup can be used in many practical scenarios such as emergency rescue and temporary communication[102, 103]. We denote $\mathcal{N} = \{1, ..., n, ..., N\}$ as the set of clusters and $\mathcal{N}^+ = \mathcal{N} \cup \{N+1\}$ as the extended set, where the $(N+1)$-th cluster denotes the dock station. The UAV flies through all the clusters successively according to a pre-optimized trajectory, and transmits data to the users when hovering at a given point, e.g., above the cluster's center. Let K_n and $\mathcal{K}_n$ denote the number and set of the users in the n-th cluster. The demands of user $k \in \mathcal{K}_n$ are denoted by $q_{k,n}$ (in bits). When all the demands in a cluster are satisfied, the UAV leaves the current cluster and visits the next one. After serving all the clusters, the UAV flies back to the dock station. The process of the UAV from leaving to returning the dock station is defined as a round or a task. Fig. 3.1 illustrates an example of the considered system.

The data stored in the UAV typically have a certain life span [104]. Thus, we consider the transmitted data are delay-sensitive, and all data delivery must be completed within T_{max} (in frames), where the time domain is divided by frames in set $\mathcal{T} = \{1, ..., t, ..., T_{max}\}$. One frame consists of I timeslots, and the duration of a timeslot is Φ. With SDMA, the UAV can simultaneously transmit data to more than one

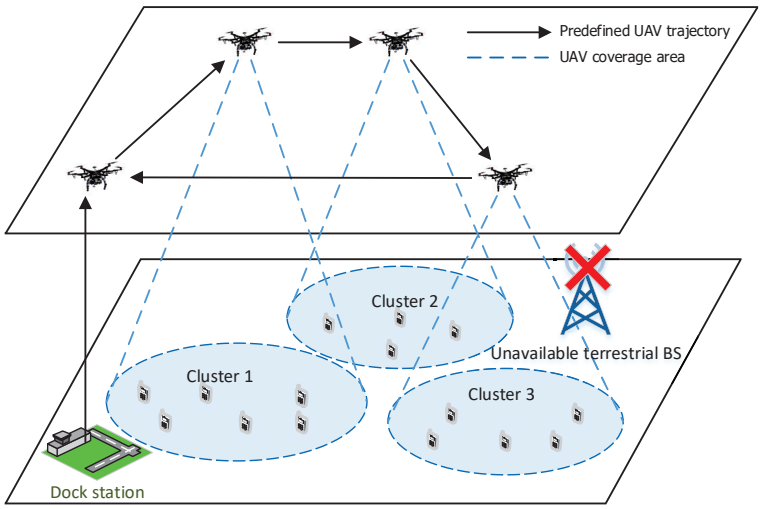


Figure 3.1: An illustrative UAV-aided network.

user in each timeslot. The frame-timeslot structure is shown in Fig. 3.2, where the shaded blocks indicate that the users are scheduled. We define the scheduled users at a timeslot as a user group. The union of the possible groups in cluster n is denoted by $\mathcal{G}_n = \{1, ..., g, ..., G_n\}$. The maximum number of candidate groups in cluster n is $G_n = 2^{K_n} - 1$ [105, 106], which increases exponentially with K_n. The number and set of the users of group g in cluster n are denoted by $K_{g,n}$ and $\mathcal{K}_{g,n}$, respectively.

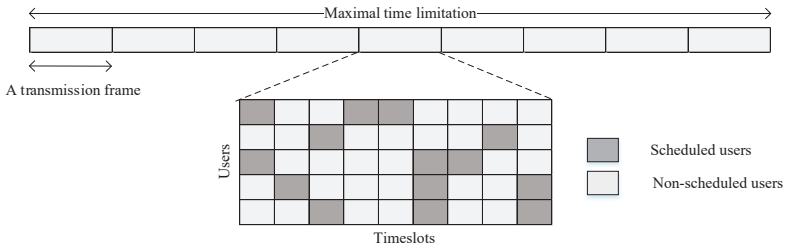


Figure 3.2: An illustration of the frame-timeslot structure.

The channel vector from the UAV antennas to ground user $k \in \mathcal{K}_n$ is denoted as $\mathbf{h}_{k,n} \in \mathbb{C}^{1\times L}$, which can be expressed by $\boldsymbol{\alpha}_{k,n} 10^{-\xi_{k,n}/10}$, where $\boldsymbol{\alpha}_{k,n} \in \mathbb{C}^{1\times L}$ is the multipath Rician fading vector and $\xi_{k,n}$ is the free-space propagation loss between the UAV and ground user $k \in \mathcal{K}_n$. The model comprises a deterministic LoS component and a random multi-path component. The former predicts the propagation loss of a signal encounters air-to-ground scenarios, and the latter captures the effects of reflection, scattering, and diffraction by the ground obstacles. The model is suited and widely adopted for UAV applications in urban/suburban scenarios, e.g., [87, 92, 98, 102, 107, 108]. We collect all the channel vectors of the users in $\mathcal{K}_{g,n}$ to form a matrix $\mathbf{H}_{g,n} \in \mathbb{C}^{K_{g,n}\times L}$. Within a user group, we apply a linear minimum mean square error (MMSE) precoding scheme due to its high efficiency and low computational complexity in mitigating intra-group interference. The precoding vector for user $k \in \mathcal{K}_{g,n}$ is calculated by:

$$\mathbf{w}_{k,g,n} = \sqrt{p_{k,g,n}} \frac{\tilde{\mathbf{h}}_{k,g,n}}{\|\tilde{\mathbf{h}}_{k,g,n}\|}, \tag{3.1}$$

where $p_{k,g,n}$ is the transmit power for user k in group g, $\tilde{\mathbf{h}}_{k,g,n}$ is the k-th column in $\mathbf{H}_{g,n}^H(\sigma^2\mathbf{I} + \mathbf{H}_{g,n}\mathbf{H}_{g,n}^H)^{-1}$, and σ^2 is the noise power. Note that transmit power $p_{k,g,n}$ is fixed as parameters in this work by following practical UAV applications, e.g., constant transmit power can be selected from 0.1 W to 10 W [109]. The signal-to-interference-plus-noise ratio (SINR) for the user $k \in \mathcal{K}_{g,n}$ is given by:

$$\Gamma_{k,g,n} = \frac{\beta_{g,n}^{(kk)} p_{k,g,n}}{\sum_{j \in \mathcal{K}_{g,n} \setminus \{k\}} \beta_{g,n}^{(kj)} p_{j,g,n} + \sigma^2}, \; k \in \mathcal{K}_{g,n}, \; g \in \mathcal{G}_n, \tag{3.2}$$

where $\beta_{g,n}^{(kk)} = |\mathbf{h}_{k,n}\tilde{\mathbf{h}}_{k,g,n}|^2$ and $\beta_{g,n}^{(kj)} = |\mathbf{h}_{k,n}\tilde{\mathbf{h}}_{j,g,n}|^2$ are the effective channel gains.

We assume the channel state information (CSI) updates over frames, and the channel states keep static within a transmission frame. Based on the adopted path-loss and Rician-fading model, we further model the time-varying channel as the first state Markov channel (FSMC) to capture the time-correlation characteristics and allow mathematically tractable analysis. Given the Rician probability density function and the corresponding auto-correlation function, we can discretize the channel into several intervals and derive the transition probabilities. Thus, knowing the initial channel state (sampling by Rician distribution), the following channel states can be forecasted by the transition probabilities [110]. By FSMC modeling, the channel quality in the near future can be forecast based on the knowledge of previous channel conditions. Moreover, FSMC is efficient in quick simulations and system performance evaluations [111, 112]. Since CSI varies over frames, we use $\Gamma_{k,g,n,t}$, $\beta_{g,n,t}^{(kk)}$ and $\beta_{g,n,t}^{(kj)}$ to track SINR and channel coefficients on the t-th frame. We quantify each coefficient $\beta_{g,n,t}^{(kk)}$ and $\beta_{g,n,t}^{(kj)}$ to multiple Markov states and obtain a transition probability such that the variations of $\beta_{g,n,t}^{(kk)}$ and $\beta_{g,n,t}^{(kj)}$ follow a Markov process between frames [113]. If group $g \in \mathcal{G}_n$ is scheduled at timeslot i on frame t, the amount of data transmitted to user $k \in \mathcal{K}_{g,n}$ and the consumed communication energy of group $g \in \mathcal{G}_n$ can be expressed by (3.3) and (3.4), respectively.

$$d_{k,g,n,t} = \Phi B \log_2\left(1 + \Gamma_{k,g,n,t}\right), \; k \in \mathcal{K}_{g,n}, \; g \in \mathcal{G}_n, \; t \in \mathcal{T}, \tag{3.3}$$

$$e_{g,n,t} = \Phi \sum_{k \in \mathcal{K}_{g,n}} p_{k,g,n}, \; g \in \mathcal{G}_n, \; t \in \mathcal{T}, \tag{3.4}$$

where B is the system bandwidth. Note that within a frame, we assume a user's channel condition is identical across all the timeslots, thus index i is omitted in $d_{k,g,n,t}$ and $e_{g,n,t}$.

3.2.2 UAV's Energy Model

We employ a UAV energy model proposed in [85]. The flying power is formulated as a function $f(U)$ of flying velocity U:

$$f(U) = P_0\left(1 + \frac{3U^2}{U_{tip}^2}\right) + P_1\left(\sqrt{1 + \frac{U^4}{4U_{ind}^4}} - \frac{U^2}{2U_{ind}^2}\right)^{\frac{1}{2}} + \frac{1}{2}\rho_1\rho_2 U^3, \tag{3.5}$$

where

- P_0: the blade profile power in hovering status;
- P_1: the induced power in hovering status;
- U_{tip}: the tip speed of the rotor blade;

- U_{ind}: the mean rotor induced velocity;
- ρ_1: the parameter related to the fuselage drag ratio, rotor solidity, and the rotor disc area;
- ρ_2: the air density.

When UAV approaches the hovering point of each cluster, it will fly around the point with a certain velocity $U = U_{hov}$, which is more energy-efficient than $U = 0$ [82]. Thus, the hovering power P_H is $f(U = U_{hov})$. The flying energy with constant velocity U and traveling distance S is expressed as:

$$\begin{aligned} &f(U) \cdot S/U \\ &= SP_0\left(\frac{1}{U} + \frac{3U}{U_{tip}^2}\right) + SP_1\left(\sqrt{\frac{1}{U^4} + \frac{1}{4U_{ind}^4}} - \frac{1}{2U_{ind}^2}\right)^{\frac{1}{2}} + \frac{S}{2}\rho_1\rho_2 U^2. \end{aligned} \tag{3.6}$$

Hovering energy and communication energy need to be jointly optimized since they are coupled by hovering time, whereas the optimization of flying energy is independent. By applying graph-based numerical methods [114], the minimum flying energy E_F^* along with the optimal flying speed U_F^* can be obtained by:

$$E_F^* = f(U_F^*) \cdot S/U_F^*, \tag{3.7}$$

where $U_F^* = \text{argmin}_{U \geq 0} \frac{f(U)}{U}$.

The main notations are summarized in Table 3.1.

Table 3.1: Summary of symbols and notations

Notation	Description
$N, \mathcal{N}$	number and set of clusters
L	number of antennas in UAV
$K_n, \mathcal{K}_n$	number and set of users in cluster n
$G_n, \mathcal{G}_n$	number and set of groups in cluster n
$K_{g,n}, \mathcal{K}_{g,n}$	number and set of users in group g of cluster n
$q_{k,n}$	demands of user k in cluster n
$T_{max}, \mathcal{T}$	maximum number and set of frames in each round
$I, \mathcal{I}$	number and set of timeslots in each frame
Φ	duration of each timeslot (in seconds)
$\Gamma_{k,g,n,t}$	SINR of user $k \in \mathcal{K}_{g,n}$ on frame t
$\beta_{g,n,t}^{(kj)}$	channel coefficient from user j's precoding vector to user k ($k, j \in \mathcal{K}_{g,n}$) on frame t
$d_{k,g,n,t}$	transmitted data of user $k \in \mathcal{K}_{g,n}$ per timeslot on frame t
$e_{g,n,t}$	communication energy of group $g \in \mathcal{G}_n$ per timeslot on frame t
U_F^*	UAV's flying velocity that minimizes flying energy with a predetermined flying path
E_F^*	minimal flying energy with a predetermined flying path

3.3 Problem Formulation

We denote binary variables $\lambda_{i,g,n,t} \in \{0,1\}$ as the scheduling indicator, where $\lambda_{i,g,n,t} = 1$ indicates that user group $g \in \mathcal{G}_n$ is assigned to timeslot i on frame t and $\lambda_{i,g,n,t} = 0$ otherwise. Another class of binary variables $\nu_{n,t} \in \{0,1\}$ indicates that the UAV is hovering above cluster n on frame t ($\nu_{n,t} = 1$), and $\nu_{n,t} = 0$ otherwise. The UAV energy consumption consists of flying energy $E_{_F}$, hovering energy $E_{_H}$, and communication energy $E_{_C}$. Since the minimal flying energy $E_{_F}^*$ can be independently obtained by Eq. (3.7) without loss of optimality, the objective focuses on joint optimization of $E_{_C}$ and $E_{_H}$, which are expressed by:

$$E_{_C} = \sum_{t=1}^{T_{max}} \sum_{n=1}^{N} \sum_{g=1}^{G_n} \sum_{i=1}^{I} \nu_{n,t} \lambda_{i,g,n,t} e_{g,n,t}, \tag{3.8}$$

$$E_{_H} = \sum_{t=1}^{T_{max}} \sum_{n=1}^{N} \Phi I P_H \nu_{n,t}. \tag{3.9}$$

Note that the UAV is battery-limited in practice. We focus on the instances that the minimum consumed energy in (3.10a) is within the UAV's battery storage, otherwise, the task is infeasible. The optimization problem is formulated as:

$$\mathcal{P}_1 : \min_{\substack{\lambda_{i,g,n,t}, \\ \nu_{n,t}}} \quad E_{_C} + E_{_H} \tag{3.10a}$$

$$s.t.$$

$$\sum_{t=1}^{T_{max}} \sum_{g=1}^{G_n} \sum_{i=1}^{I} \nu_{n,t} \lambda_{i,g,n,t} d_{k,g,n,t} \geq q_{_{k,n}}, \ \forall k \in \mathcal{K}_n, \ n \in \mathcal{N}, \tag{3.10b}$$

$$\nu_{_{n,t}} \leq \nu_{_{n,t+1}} + \nu_{_{n+1,t+1}}, \ \forall n \in \mathcal{N}, \ t \in \mathcal{T}, \tag{3.10c}$$

$$\sum_{g=1}^{G_n} \sum_{i=1}^{I} \lambda_{i,g,n,t} = I \cdot \nu_{_{n,t}}, \ \forall n \in \mathcal{N}^+, \ t \in \mathcal{T}, \tag{3.10d}$$

$$\sum_{g=1}^{G_n} \lambda_{i,g,n,t} \leq 1, \ \ \forall i \in \mathcal{I}, \ n \in \mathcal{N}^+, \ t \in \mathcal{T}, \tag{3.10e}$$

$$\sum_{n=1}^{N+1} \nu_{n,t} = 1, \ \forall t \in \mathcal{T}, \tag{3.10f}$$

$$\lambda_{i,g,n,t} \in \{0,1\}, \ \forall i \in \mathcal{I}, \ g \in \mathcal{G}_n, \ n \in \mathcal{N}^+, \ t \in \mathcal{T}, \tag{3.10g}$$

$$\nu_{n,t} \in \{0,1\}, \ \forall n \in \mathcal{N}^+, \ t \in \mathcal{T}. \tag{3.10h}$$

Constraints (3.10b) guarantee that all the users' requests have to be satisfied within T_{max}. Constraints (3.10c) define that the UAV follows a successive and forward manner in visiting clusters. For example, if the UAV is hovering above cluster n on frame t, in the next frame $t+1$, the UAV either chooses to stay at the current cluster n or move to the next cluster $n+1$. The option of flying back to previously visited clusters, e.g., $n-1$, is thus excluded. Note that the UAV takes off from the first cluster, i.e., $\nu_{_{1,1}} = 1$. Constraints (3.10d) represent that all the timeslots on frame t are assigned to a user group when $\nu_{n,t} = 1$, otherwise, no users are scheduled in any timeslot. Constraints (3.10e) and (3.10f) indicate that

no more than one group can be scheduled at a timeslot and only one cluster can be served within a frame. Constraints (3.10g) and (3.10h) confine variables $\lambda_{i,g,n,t}$ and $\nu_{n,t}$ to binary.

Note that $\mathcal{P}_1$ is a combinatorial optimization problem with a non-convex bilinear objective and constraints. The optimum can be approached by a well-established relax-and-approximate method. That is, the non-convex bilinear terms are relaxed and bounded by McCormick envelopes [63], where each variable ($\lambda_{i,g,n,t}$ and $\nu_{n,t}$) is bounded by an upper and a lower bound. The relaxation problem becomes an integer linear programming (ILP) problem which can be optimally solved by B&B. Overall, the optimum of $\mathcal{P}_1$ can be approached by ultimately tightening the bounds, e.g., increase the number of breakpoints in the envelopes, but this results in exponentially increasing complexity which is unaffordable in practice [73]. Thus, we adopt the above relax-and-approximate method to provide an optimal solution for benchmarking small-medium cases. For general cases, we propose a heuristic algorithm in the next section.

3.4 Heuristic Approach

We decompose the joint optimization to two sub-problems, i.e., user-timeslot and hovering time allocation, corresponding to optimization of $\lambda_{i,g,t,n}$ and $\nu_{n,t}$, respectively. We then solve one sub-problem when the other is fixed.

3.4.1 User-Timeslot Scheduling

The bilinear items are resolved with the fixed $\nu_{n,t}$. The number of frames at each cluster is determined by:

$$t_n = \sum_{t=1}^{T_{max}} \nu_{n,t},\ \forall n \in \mathcal{N}, \tag{3.11}$$

and ΦIt_n is the hovering duration. The user-timeslot scheduling can be carried out independently in each cluster, and the resulting problem for the n-th cluster is formulated in $\mathcal{P}_2(n)$ with a given t_n. We denote $E_{_{H,n}}$ and $E_{_{C,n}}$ as the hovering and communication energy for the n-th cluster:

$$E_{_{H,n}} = \Phi I P_{_H} t_n, \tag{3.12}$$

$$E_{_{C,n}} = \sum_{t=\tau_n+1}^{\tau_n+t_n} \sum_{g=1}^{G_n} \sum_{i=1}^{I} \lambda_{i,g,n,t} e_{g,n,t}, \tag{3.13}$$

where τ_n refers to the number of elapsed frames before the UAV arriving cluster n, which can be calculated by:

$$\tau_n = \sum_{t=1}^{T_{max}} \sum_{n'=1}^{n-1} \nu_{n',t}. \tag{3.14}$$

The sub-problem $\mathcal{P}_2(n)$ is formulated as:

$$\mathcal{P}_2(n): \min_{\lambda_{i,g,n,t}} \quad E_{_{C,n}} + E_{_{H,n}} \tag{3.15a}$$

$$s.t. \sum_{t=\tau_n+1}^{\tau_n+t_n} \sum_{g=1}^{G_n} \sum_{i=1}^{I} \lambda_{i,g,n,t} d_{k,g,n,t} \geq q_{k,n},\ \forall k \in \mathcal{K}_n, \tag{3.15b}$$

$$\sum_{g=1}^{G_n} \sum_{i=1}^{I} \lambda_{i,g,n,t} = I,\ \forall t \in \{\tau_n+1, ..., \tau_n+t_n\}, \tag{3.15c}$$

$$\sum_{g=1}^{G_n} \lambda_{i,g,n,t} \leq 1,\ \ \forall i \in \mathcal{I},\ t \in \mathcal{T}, \tag{3.15d}$$

$$\lambda_{i,g,n,t} \in \{0,1\},\ \forall i \in \mathcal{I},\ g \in \mathcal{G}_n,\ t \in \mathcal{T}. \tag{3.15e}$$

$\mathcal{P}_2(n)$ is a multi-choice multi-dimensional knapsack problem (MMKP), which can be solved by a guided local search (GLS)-based heuristic algorithm with high-quality sub-optimal solutions and pseudo-polynomial-time complexity [115].

3.4.2 Hovering Time Allocation

To optimize hovering time efficiently, we first investigate the connection between the objective energy and t_n. From Eq. (3.12) and Eq. (3.13), $E_{H,n}$ increases linearly with t_n while $E_{C,n}$ is determined by both t_n and $\lambda_{i,g,n,t}$. Next, we show the relationship between the optimum $E_{C,n}$ and t_n. For cluster n, we denote $E^*_{C,n}(t_n)$ as the communication energy with the optimal scheduling decision $\lambda^*_{i,g,n,t}$ at a given hovering time t_n.

Lemma 1. *$E^*_{C,n}(t_n)$ is a non-increasing function of t_n,*

$$E^*_{C,n}(\hat{t}) \geq E^*_{C,n}(\hat{t}+\Delta t),\ \hat{t} > 0, \Delta t > 0. \tag{3.16}$$

Proof. We denote the optimal user scheduling for $\mathcal{P}_2(n)|_{t_n=\hat{t}}$ as $\lambda^*_{i,g,n,t}$. If t_n increases from $\hat{t}$ to $\hat{t}+\Delta t$, $\lambda^*_{i,g,n,t}$ is still feasible for $\mathcal{P}_2(n)|_{t_n=\hat{t}+\Delta t}$ such that

$$\begin{aligned} E^*_{C,n}(\hat{t}) &= \sum_{t=\tau_n+1}^{\tau_n+\hat{t}} \sum_{g=1}^{G_n} \sum_{i=1}^{I} \lambda^*_{i,g,n,t} e_{g,n,t} \\ =& E^{'}_{C,n}(\hat{t}+\Delta t) = \sum_{t=\tau_n+1}^{\tau_n+\hat{t}+\Delta t} \sum_{g=1}^{G_n} \sum_{i=1}^{I} \lambda^*_{i,g,n,t} e_{g,n,t}. \end{aligned} \tag{3.17}$$

$\lambda^*_{i,g,n,t}$ might not be necessarily optimal for $t_n = \hat{t}+\Delta t$. There exists an optimal scheduling resulting in lower communication energy, i.e.,

$$E^*_{C,n}(\hat{t}+\Delta t) \leq E^{'}_{C,n}(\hat{t}+\Delta t) = E^*_{C,n}(\hat{t}). \tag{3.18}$$

Thus the conclusion. □

From Lemma 1, we can observe that $E^*_{C,n}(t_n)$ is an non-increasing function of t_n, i.e., $\frac{dE^*_{C,n}(t_n)}{dt_n} \leq 0$. For $E_{H,n}(t_n)$, we can derive that $\frac{dE_{H,n}(t_n)}{dt_n} = \Phi I P_H$ based on Eq. (3.12). Thus, the extreme point of

$E^*_{C,n}(t_n) + E_{H,n}(t_n)$ can be obtained at $t_n = t^\dagger$ when

$$\frac{dE^*_{C,n}(t_n)}{dt_n}|_{t_n=t^\dagger} = -\Phi IP_H. \tag{3.19}$$

Since the existence and the number of extreme points are undetermined. There are three possible cases, i.e., unimodal, multimodal, and monotonic, for $E^*_{C,n}(t_n) + E_{H,n}(t_n)$, as illustrated in Fig. 3.3. In case 1, the curve is a unimodal function with only one extreme point. In case 2, the fluctuation of $\frac{dE^*_{C,n}(t_n)}{dt_n}$ leads to multiple extreme points such that the curve is a multimodal function. In case 3, Eq. (3.19) cannot hold, e.g., $\frac{dE^*_{C,n}(t_n)}{dt_n}$ is consistently lager than $-\Phi IP_H$, so the curve is monotonously increasing with no extreme point.

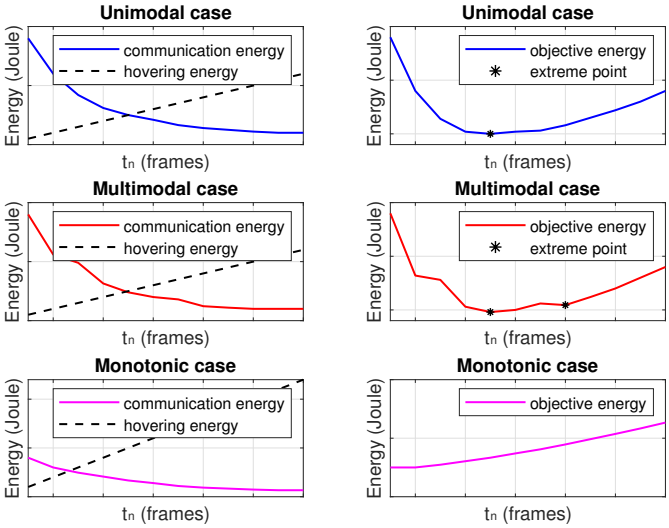


Figure 3.3: Energy curves for three possible cases.

Observing the possible cases, we employ an efficient golden section search (GSS) to find the extreme points [116]. In GSS, we limit the hovering time $t_n \leq \bar{t}_n$ to ensure that the total service duration does not exceed T_{max}, where $\bar{t}_n$ is a maximal time limitation for cluster n. Intuitively, the clusters with more demands need more transmission frames. We assume $\bar{t}_n$ is proportional to the users' demands:

$$\bar{t}_n = T_{max} \frac{\sum_{k=1}^{K_n} q_{k,n}}{\sum_{n=1}^{N} \sum_{k=1}^{K_n} q_{k,n}}. \tag{3.20}$$

3.4.3 Algorithm Summary

We summarize the proposed GSS-HEU in Alg. 3. We denote $\mathcal{B}_{n,t}$ as the set of channel states of cluster n on frame t, which is expressed as:

$$\mathcal{B}_{n,t} = \{\beta^{(kj)}_{1,n,t}, ..., \beta^{(kj)}_{G_n,n,t} | \ \forall k, j \in \mathcal{K}_{g,n}\}. \tag{3.21}$$

In GSS-HEU, the initial search range of GSS $[x_1, y_1]$ is set as $[0, \bar{t}_n]$, which is partitioned into three sections by two points u_1 and v_1 with the golden ratio 0.618 in lines 2-4, where $\lceil . \rceil$ is an operation to round a value up to an integer. When a hovering time is searched in GSS, e.g., $t_n = u_m$ or $t_n = v_m$, the corresponding user-timeslot allocation is obtained by solving $\mathcal{P}_2(n)$ in line 6. In lines 9-13, we compare the objective energy and update the search range. The search process terminates at $|y_m - x_m| \leq 1$. The selected hovering time t_n^* is v_m and the corresponding scheduling scheme $\lambda_{i,g,n,t}^*$ is $\lambda_{i,g,n,t}|_{t_n=v_m}$.

Algorithm 3 GSS-HEU Algorithm

Inputs:

Users' demands: $q_{1,1}$,..., $q_{K_1,1}$,..., $q_{1,N}$,...,$q_{K_N,N}$;
Channel states: $\mathcal{B}_{1,1}$,...,$\mathcal{B}_{1,T_{max}}$,...,$\mathcal{B}_{N,1}$,...,$\mathcal{B}_{N,T_{max}}$;
Search range's upper bound: $\bar{t}_1, ..., \bar{t}_N$.

1: **for** $n = 1; n \leq N; n++$ **do**
2: $x_1 = 0; y_1 = \bar{t}_n;$
3: $u_1 = \lceil y_1 - 0.618(y_1 - x_1) \rceil;$
4: $v_1 = \lceil x_1 + 0.618(y_1 - x_1) \rceil;$
5: **for** $m = 1; |y_m - x_m| > 1; m++$ **do**
6: Solve $\mathcal{P}_2(n)|_{t_n=u_m}$ and $\mathcal{P}_2(n)|_{t_n=v_m}$;
7: Obtain the corresponding user scheduling schemes $\lambda_{i,g,n,t}|_{t_n=u_m}$ and $\lambda_{i,g,n,t}|_{t_n=v_m}$;
8: Obtain the objective energy $(E_{C,n} + E_{H,n})|_{t_n=u_m}$ and $(E_{C,n} + E_{H,n})|_{t_n=v_m}$;
9: **if** $(E_{C,n} + E_{H,n})|_{t_n=u_m} < (E_{C,n} + E_{H,n})|_{t_n=v_m}$ **then**
10: $x_{m+1} = x_m;\ y_{m+1} = v_m;\ v_{m+1} = u_m;$
$u_{m+1} = \lceil y_{m+1} - 0.618(y_{m+1} - x_{m+1}) \rceil;$
11: **else**
12: $x_{m+1} = u_m;\ y_{m+1} = y_m;\ u_{m+1} = v_m;$
$v_{m+1} = \lceil y_{m+1} - 0.618(y_{m+1} - x_{m+1}) \rceil;$
13: **end if**
14: **end for**
15: $t_n^* = v_m; \lambda_{i,g,n,t}^* = \lambda_{i,g,n,t}|_{t_n=v_m}.$
16: **end for**

Outputs:

Heuristic solution: $\lambda_{1,1,1,1}^*, ..., \lambda_{I,G_n,N,T_{max}}^*, t_1^*, ..., t_N^*$

The complexity of GSS-HEU is $\mathcal{O}(\sum_{n=1}^{N} G_n^2 \times \max\{K_n, I\bar{t}_n\} \times \log(2\bar{t}_n))$, which is much lower than that of the optimal method. However, both the optimal and GSS-HEU approaches may have limitations in fast decision-making. The computational time for both algorithms grows exponentially with the number of users since $G_n = 2^{K_n} - 1$ [74]. In addition, both algorithms need the estimated and complete channel states for the whole task frames, i.e., from $t = 1$ to T_{max}. This may result in difficulties in channel estimation. Therefore, we reconsider $\mathcal{P}_1$ from the perspective of DRL to enable the UAV to make decisions intelligently, while the developed optimal and heuristic algorithms are used to benchmark the performance of learning-based solutions.

3.5 Actor-Critic-Based DRL Algorithm

3.5.1 Overview of Actor-Cirtic-Based DRL (AC-DRL)

In DRL, an agent learns to make decisions by exploring the unknown environments and exploiting the received feedbacks. At each learning step[1] t, the agent observes the current state $\boldsymbol{s}_t$ and takes an action $\boldsymbol{a}_t$ based on a policy. Then, a reward r_t will be fed back to the agent. The policy will be updated step by step according to the feedback. Actor-critic is an emerging reinforcement learning method that separates the agent into two parts, an actor and a critic. The actor is responsible for taking actions following a stochastic policy $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t)$, where $\pi(.|.)$ refers to a conditional probability density function. The critic is used to evaluate the decisions via a Q-value, which is given by:

$$Q^{\pi}(\boldsymbol{s}_t, \boldsymbol{a}_t) = \mathbb{E}_{\boldsymbol{a}_t \sim \pi(\boldsymbol{a}_t|\boldsymbol{s}_t)}[R_t|\boldsymbol{s}_t, \boldsymbol{a}_t], \tag{3.22}$$

where $\mathbb{E}_{\boldsymbol{a}_t \sim \pi(\boldsymbol{a}_t|\boldsymbol{s}_t)}[.|.]$ is a conditional expectation under the policy $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t)$, and R_t is the cumulative discounted reward with a discount factor γ, which can be expressed as:

$$R_t = \sum_{t'=t}^{\infty} \gamma^{t'-t} r_{t'}, \ \ \gamma \in [0, 1]. \tag{3.23}$$

However, obtaining the explicit expressions of $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t)$ and $Q^{\pi}(\boldsymbol{s}_t, \boldsymbol{a}_t)$ is difficult. DRL uses DNNs as the parameterized approximators to provide estimations for $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t)$ and $Q^{\pi}(\boldsymbol{s}_t, \boldsymbol{a}_t)$. We denote $\boldsymbol{\theta}_t$ and $\boldsymbol{\omega}_t$ as the parameter vectors for the actor and critic, and $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t; \boldsymbol{\theta}_t)$ and $Q^{\boldsymbol{\theta}}(\boldsymbol{s}_t, \boldsymbol{a}_t; \boldsymbol{\omega}_t)$ as the corresponding parameterized functions[2]. The goal of the agent is to minimize the loss function of the actor $-J(\boldsymbol{\theta}_t)$:

$$-J(\boldsymbol{\theta}_t) = -\mathbb{E}[Q^{\boldsymbol{\theta}}(\boldsymbol{s}_t, \boldsymbol{a}_t; \boldsymbol{\omega}_t)]. \tag{3.24}$$

Based on the fundamental results of the policy gradient theorem [57], the gradient of $J(\boldsymbol{\theta}_t)$ can be calculated by:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t) = \mathbb{E}[\nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{a}_t|\boldsymbol{s}_t; \boldsymbol{\theta}_t) Q^{\boldsymbol{\theta}}(\boldsymbol{s}_t, \boldsymbol{a}_t; \boldsymbol{\omega}_t)]. \tag{3.25}$$

The update rule of $\boldsymbol{\theta}_t$ can be derived based on gradient descent:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha_a \cdot (-\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t)), \tag{3.26}$$

where α_a is the learning rate of the actor. For the critic, the parameter vector $\boldsymbol{\omega}_t$ is updated based on temporal-difference (TD) learning [57]. In TD learning, the loss function of the critic $C_{_Q}(\boldsymbol{\omega}_t)$ is defined as the expectation of the square of TD error $\delta_{_Q}(\boldsymbol{\omega}_t)$, i.e., $\mathbb{E}[(\delta_{_Q}(\boldsymbol{\omega}_t))^2]$. The TD error $\delta_{_Q}(\boldsymbol{\omega}_t)$ refers to the difference between the TD target and estimated Q-value, which is given by:

$$\delta_{_Q}(\boldsymbol{\omega}_t) = r_t + \gamma Q^{\boldsymbol{\theta}}(\boldsymbol{s}_{t+1}, \boldsymbol{a}_{t+1}; \boldsymbol{\omega}_t) - Q^{\boldsymbol{\theta}}(\boldsymbol{s}_t, \boldsymbol{a}_t; \boldsymbol{\omega}_t), \tag{3.27}$$

where $r_t + \gamma Q^{\boldsymbol{\theta}}(\boldsymbol{s}_{t+1}, \boldsymbol{a}_{t+1}; \boldsymbol{\omega}_t)$ is the TD target. The objective of the critic is to minimize the loss function $C_{_Q}(\boldsymbol{\omega}_t)$ and the update rule of $\boldsymbol{\omega}_t$ can be derived by gradient descent:

$$\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t - \alpha_c \nabla_{\boldsymbol{\omega}} C_{_Q}(\boldsymbol{\omega}_t), \tag{3.28}$$

[1] In this chapter, a learning step is equivalent to a transmission frame.

[2] For simplicity, $Q^{\boldsymbol{\theta}}(\boldsymbol{s}_t, \boldsymbol{a}_t; \boldsymbol{\omega}_t) = \mathbb{E}_{\boldsymbol{a}_t \sim \pi(\boldsymbol{a}_t|\boldsymbol{s}_t; \boldsymbol{\theta}_t)}[R_t|\boldsymbol{s}_t, \boldsymbol{a}_t]$.

where α_c is the learning rate for the critic.

However, approximating $Q^{\pi}(\boldsymbol{s}_t, \boldsymbol{a}_t)$ brings about a large variance for the gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t)$, resulting in poor convergence [117]. To solve the problem, a V-value is introduced:

$$V^{\pi}(\boldsymbol{s}_t) = \mathbb{E}_{\boldsymbol{a}_t \sim \pi(\boldsymbol{a}_t|\boldsymbol{s}_t)}[R_t|\boldsymbol{s}_t]. \tag{3.29}$$

Approximating $V^{\pi}(\boldsymbol{s}_t)$ can reduce the variance. With the parameterized V-value $V^{\boldsymbol{\theta}}(\boldsymbol{s}_t; \boldsymbol{\omega}_t)$, the TD error and the loss function of the critic are expressed as:

$$\delta_{_V}(\boldsymbol{\omega}_t) = r_t + \gamma V^{\boldsymbol{\theta}}(\boldsymbol{s}_{t+1}; \boldsymbol{\omega}_t) - V^{\boldsymbol{\theta}}(\boldsymbol{s}_t; \boldsymbol{\omega}_t), \tag{3.30}$$

and

$$C_{_V}(\boldsymbol{\omega}_t) = \mathbb{E}[(\delta_{_V}(\boldsymbol{\omega}_t))^2]. \tag{3.31}$$

In addition, $\delta_{_V}(\boldsymbol{\omega}_t)$ provides an unbiased estimation of Q-value [117]. Thus, we can rewrite $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t)$ in Eq. (3.25) as:

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t) =& \mathbb{E}\left[\nabla_{\boldsymbol{\theta}} \log(\pi(\boldsymbol{a}_t|\boldsymbol{s}_t; \boldsymbol{\theta}_t)) Q^{\pi}(\boldsymbol{s}_t, \boldsymbol{a}_t)\right] \\ =& \mathbb{E}\left[\nabla_{\boldsymbol{\theta}} \log(\pi(\boldsymbol{a}_t|\boldsymbol{s}_t; \boldsymbol{\theta}_t)) \delta_{_V}(\boldsymbol{\omega}_t)\right]. \end{aligned} \tag{3.32}$$

3.5.2 Problem Reformulation

To apply AC-DRL, we reformulate $\mathcal{P}_1$ to an MDP problem, in which the UAV acts as an agent. We define the states, actions, and rewards as follows.

States

The system states $\boldsymbol{s}_t$ consist of the channel states for all the clusters on the current frame, i.e., $\mathcal{B}_{1,t}, ..., \mathcal{B}_{N,t}$, the undelivered demands, and the currently served cluster on frame t. The undelivered demands $b_{n,t}$ is the residual data to be delivered for cluster n on frame t:

$$b_{n,t+1} = b_{n,t} - d^{\pi}_{n,t}, \ \forall n \in \mathcal{N}, \ t \in \mathcal{T}, \tag{3.33}$$

$$b_{n,0} = \sum_{k=1}^{K_n} q_{k,n}, \ \forall n \in \mathcal{N}, \tag{3.34}$$

where $d^{\pi}_{n,t}$ is the delivered data for cluster n on frame t under the policy $\pi(\boldsymbol{s}_t|\boldsymbol{a}_t)$. We denote $o_t \in \mathcal{N}^+$ as an indicator to represent which cluster the UAV is serving on frame t. When the users' requests in the current cluster are completed, the UAV will move to the next cluster on the next frame, otherwise, staying at the current cluster. For example, we assume that the UAV is hovering above cluster n on frame t, i.e., $o_t = n$. For the next frame, o_{t+1} is obtained by:

$$o_{t+1} = \begin{cases} n, & b_{n,t} > 0, \\ n+1, & b_{n,t} = 0. \end{cases} \tag{3.35}$$

When the UAV's duration exceeds T_{max}, the UAV will fly back to the dock station. By assembling the above three parts, the state $\boldsymbol{s}_t$ is defined as:

$$\boldsymbol{s}_t = [\mathcal{B}_{_{1,t}}, ..., \mathcal{B}_{_{N,t}}, b_{_{1,t}} ..., b_{_{N,t}}, o_t]. \tag{3.36}$$

Note that the elements of $\mathcal{B}_{n,t}$ are modeled as FSMC. In addition, based on Eq. (3.33) and Eq. (3.35), the next state of $b_{n,t}$ and o_t only depend on the current state and current policy. Therefore, the transition of the state $\boldsymbol{s}_t$ conforms to MDP [57].

Actions

The action of the UAV is the user-timeslot assignment on frame t, which is given by:

$$\begin{aligned} &\boldsymbol{a}_t = [a_{_{1,t}}, ..., a_{_{I,t}}], \\ &a_{_{i,t}} \in \{1, ..., g, ..., G_n\},\ \forall i \in \mathcal{I},\ t \in \mathcal{T}, \end{aligned} \tag{3.37}$$

where $a_{i,t} = g$ means the g-th group is selected at the i-th timeslot on the t-th frame. Note that the action space G_n can be huge since it increases exponentially with the number of users.

Rewards

The reward functions are commonly related to the objective of the problem. Conventionally, the reward function of $\mathcal{P}_1$ can be designed by Eq. (3.38) and Eq. (3.39), referring to [118] and [119]:

$$r_t = 1/e_t^{\pi}, \tag{3.38}$$

$$r_t = -e_t^{\pi}, \tag{3.39}$$

where e_t^{π} is the energy consumed on frame t under the policy $\pi(\boldsymbol{s}_t|\boldsymbol{a}_t)$. Since both the above reward functions monotonically decrease with e_t^{π}, the UAV updates the policy towards reducing energy consumption.

3.5.3 The AC-DSOS Algorithm

Conventional AC-DRL algorithms may not be able to deal with constrained discrete problems. Firstly, the combinatorial component of $\mathcal{P}_1$ limits the conventional AC-DRL in addressing huge discrete action spaces [120]. Secondly, the increased action space reduces the exploration efficiency in the learning process and degrades overall energy-saving performance. Thirdly, the conventional AC-DRL algorithms cannot guarantee the solution's feasibility in general. This means that a high-reward action can fail to satisfy the constraints in $\mathcal{P}_1$. To overcome the above difficulties and limitations, we propose an AC-DSOS algorithm that is tailored for constrained problems with discrete action representation. The basic actor-critic framework is employed in order to take the advantages of the stochastic policy and TD learning, where the stochastic policy can be quantified to tackle the issue of huge discrete spaces and TD learning can improve the learning efficiency.

We illustrate the actor-critic framework of AC-DSOS in Fig. 3.4, where two DNNs work as the actor and critic, respectively. The stochastic policy $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t)$ is usually modeled as Gaussian distribution with a mean $\boldsymbol{\mu}(\boldsymbol{s}_t)$ and a variance $\boldsymbol{\chi}(\boldsymbol{s}_t)$ [121]. Given the current state $\boldsymbol{s}_t$, the actor does not predict $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t;\boldsymbol{\theta}_t)$ directly but obtains approximations of the mean $\boldsymbol{\mu}(\boldsymbol{s}_t;\boldsymbol{\theta}_t)$ and the variance $\boldsymbol{\chi}(\boldsymbol{s}_t;\boldsymbol{\theta}_t)$. An action $\boldsymbol{a}_t$ can be selected based on $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t;\boldsymbol{\theta}_t)$. Then, the agent receives a reward r_t after taking the action and collects the next state $\boldsymbol{s}_{t+1}$. For the critic, two V-values, $V^{\boldsymbol{\theta}}(\boldsymbol{s}_t;\boldsymbol{\omega}_t)$ and $V^{\boldsymbol{\theta}}(\boldsymbol{s}_{s+1};\boldsymbol{\omega}_t)$, are estimated by DNN with the inputs $\boldsymbol{s}_t$ and $\boldsymbol{s}_{t+1}$, respectively. The TD error $\delta_{_V}(\boldsymbol{\omega}_t)$ can be calculated by Eq. (3.30). A tuple $\{\boldsymbol{s}_t, \boldsymbol{s}_{t+1}, \delta_{_V}(\boldsymbol{\omega}_t), r_t\}$ is stored in a memory at each step t. By applying a memory replay mechanism, the data in the memory can be used for training the DNNs. In each training step, the actor and critic are updated by the gradient descent over a batch of training data. The whole training process consists

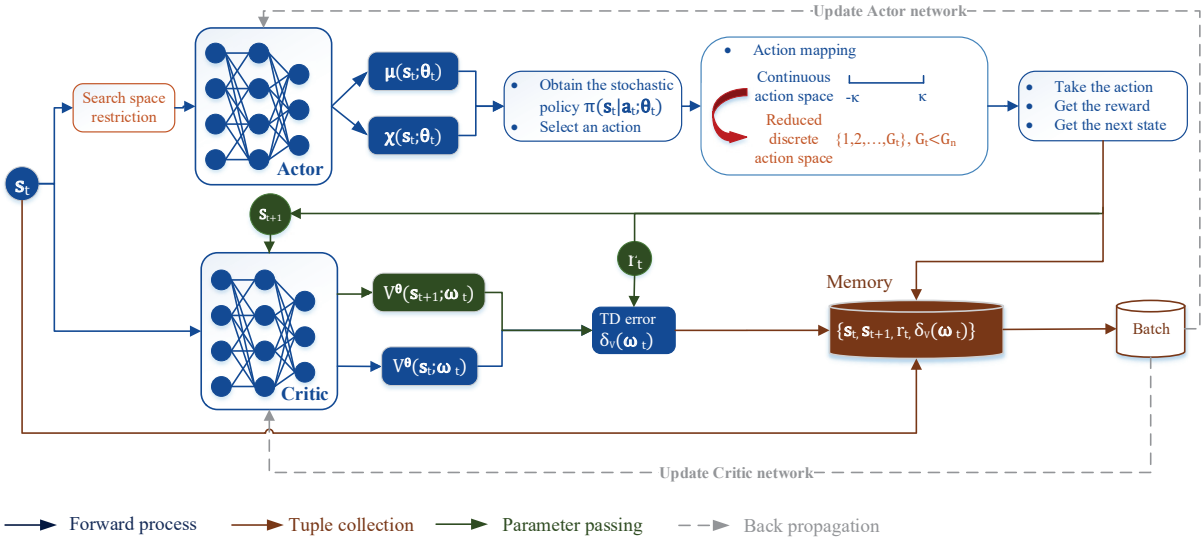


Figure 3.4: The actor-critic framework of AC-DSOS.

of multiple episodes, each episode including T_{max} steps. Based on the above framework, the AC-DSOS algorithm is summarized in Alg. 4.

In AC-DSOS, two DNNs are employed to estimate the stochastic policy and V-value with the input $\boldsymbol{s}_t$. The complexity of AC-DSOS is dominated by the DNNs' forward-propagation and back-propagation process in lines 6, 13, and 16 in Alg. 2. The size of the input is $N(K_{max}^2+1)+1$, where $K_{max} = \max\{K_1, ..., K_N\}$. We assume both DNNs have X hidden layers and the x-th layer has l_x nodes. In the forward propagation (line 6 and line 13 in Alg. 4), the complexity of the actor DNN and critic DNN are identical, i.e., $O((N(K_{max}^2+1)+1)l_1+\sum_{x=1}^{X-1} l_x l_{x+1}+l_X)$, referring to [122]. In the back propagation, (line 16 in Alg. 4), the stochastic gradient is obtained with the complexity $O(M((N(K_{max}^2+1)+1)l_1+\sum_{x=1}^{X-1} l_x l_{x+1}+l_X))$ to update the neural parameters, where M is the batch size. Overall, the complexity of AC-DSOS is $O(2T_{max}(M+1)((N(K_{max}^2+1)+1)l_1+\sum_{x=1}^{X-1} l_x l_{x+1}+l_X))$.

The novelties of the proposed AC-DSOS compared to the conventional AC-DRL are summarized as follows.

Action Mapping to Tackle the Issue of Huge Discrete Action Space

The conventional actor-critic is used for continuous action space. We denote $\hat{\boldsymbol{a}}_t = [\hat{a}_{1,t}, ..., \hat{a}_{I,t}]$ as the original action selected by the stochastic policy, where the element $\hat{a}_{i,t}$ is fractional. However, as the decision variables are integers in $\mathcal{P}_1$, the action space is discrete. To deal with this issue, we adopt an action mapping method in AC-DSOS (line 9 in Alg. 4). Firstly, we confine $\hat{a}_{i,t}$ to a fixed range $[-\kappa, \kappa]$ to avoid its value being too large/small since the domain of Gaussian distribution is $[-\infty, \infty]$. Then, a uniform quantization method is used to map $\hat{a}_{i,t}$ to the discrete action space $\{1, ..., G_n\}$ by:

$$a_{i,t} = \lceil \frac{\kappa + \hat{a}_{i,t}}{2\kappa/G_n} \rceil, \tag{3.40}$$

where $2\kappa/G_n$ is the quantization interval. With the mapping operation, we can support a larger G_n by reducing the interval.

Algorithm 4 AC-DSOS Algorithm

Inputs: The current state $\boldsymbol{s}_t$.

1: Initialize $\boldsymbol{\theta}_1$ and $\boldsymbol{\omega}_1$.
2: **for** each learning episode **do**
3: Observe the initial state $\boldsymbol{s}_1$.
4: **for** $t = 1 : T_{max}$ **do**
5: Remove the groups containing the demand-satisfied users.
6: Predicted mean $\boldsymbol{\mu}(\boldsymbol{s}_t; \boldsymbol{\theta}_t)$ and variance $\boldsymbol{\chi}(\boldsymbol{s}_t; \boldsymbol{\theta}_t)$ by the DNN of the actor.
7: Obtain action's distribution $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t; \boldsymbol{\theta}_t)$ based on Gaussian distribution.
8: Randomly choose $\hat{\boldsymbol{a}}_t$ following $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t; \boldsymbol{\theta}_t)$.
9: Map the elements $\hat{a}_{i,t}$ to $a_{i,t}$ by Eq. (3.40).
10: Take the after-mapped action $\boldsymbol{a}_t$.
11: Obtain reward r_t by Eq. (3.47).
12: Collect the next state $\boldsymbol{s}_{t+1}$.
13: Approximate the value functions $V^{\boldsymbol{\theta}}(\boldsymbol{s}_t; \boldsymbol{\omega}_t)$ and $V^{\boldsymbol{\theta}}(\boldsymbol{s}_{t+1}; \boldsymbol{\omega}_t)$ by the DNN of the critic.
14: Calculate TD error $\delta_{_V}(\boldsymbol{\omega}_t)$ by Eq. (3.30).
15: Form and store a new tuple $\{\boldsymbol{s}_t, \boldsymbol{s}_{t+1}, r_t, \delta_{_V}(\boldsymbol{\omega}_t)\}$.
16: Obtain $\boldsymbol{\theta}_{t+1}$ and $\boldsymbol{\omega}_{t+1}$ by gradient descent.
17: $\boldsymbol{s}_t = \boldsymbol{s}_{t+1}$; $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t+1}$; $\boldsymbol{\omega}_t = \boldsymbol{\omega}_{t+1}$.
18: **end for**
19: **end for**

Outputs: The current action $\boldsymbol{a}_t$.

Action Space Restriction to Improve Solution Quality

Although AC-DSOS can tackle the issue of discrete action space by the above mapping operation, exploring in a huge space remains difficult. To improve the exploration efficiency and the quality of the solution, we design a method to restrict the action space in the learning process (line 5 in Alg. 4). Compared to the conventional DRL method, the difference mainly lies at the action selection. At the beginning of each frame, we first observe which users' demands have been satisfied. Then, we remove the corresponding candidate groups, i.e., the groups containing the successfully served users. In conventional DRL, the action space keeps fixed as $G_n = 2^{K_n} - 1$. This may result in two issues: Firstly, when the action space grows exponentially large, DRL/RL needs more time to search for the optimal action, thus decreases the exploration efficiency; Secondly, the probability of selecting undesirable low-reward actions will be increased. This is because the original actions $\hat{a}_{i,t}$ are selected by a stochastic policy such that a small difference in $\hat{a}_{i,t}$ could lead to different actions after mapping. For example, $\hat{a}_{i,t} = 0.999$ and $\hat{a}_{i,t} = 1.001$ map to $a_{i,t} = 1$ and $a_{i,t} = 2$, respectively. If $a_{i,t} = 2$ is a low-reward action, a small error in $\hat{a}_{i,t}$ could cause a large loss in reward value. As illustrated in Fig. 3.4, the size of the action space in AC-DSOS, denoted by G_t ($G_t \leq G_n$), is not fixed but gradually reduces over $1, ..., T_{max}$. Before taking an action, we remove the redundant actions with lower rewards from the action space, such that the action space can keep concise and with controllable size.

Lemma 2. *At each learning step, $V^{\pi}(\boldsymbol{s}_t) \leq V^{\pi'}(\boldsymbol{s}_t)$, where $V^{\pi}(\boldsymbol{s}_t)$ and $V^{\pi'}(\boldsymbol{s}_t)$ are the V-values under the policy with the fixed action space and the reduced action space, respectively.*

Proof. We denote $\mathcal{A}$ and $\mathcal{A}'$ as the fixed action space and the reduced action space, respectively. Based

on bellman equation [57], $V^{\pi}(\boldsymbol{s}_t)$ can be expressed as:

$$\begin{aligned} V^{\pi}(\boldsymbol{s}_t) &= \sum_{\boldsymbol{a}_t \in \mathcal{A}} \pi(\boldsymbol{a}_t|\boldsymbol{s}_t) \left(r(\boldsymbol{s}_t, \boldsymbol{a}_t) + \gamma V^{\pi}(\boldsymbol{s}_{t+1})\right) \\ &= \sum_{\boldsymbol{a}_t \in \mathcal{A}} \pi(\boldsymbol{a}_t|\boldsymbol{s}_t) r(\boldsymbol{s}_t, \boldsymbol{a}_t) + \gamma \sum_{\boldsymbol{a}_{t+1} \in \mathcal{A}} \pi(\boldsymbol{a}_{t+1}|\boldsymbol{s}_{t+1}) r(\boldsymbol{s}_{t+1}, \boldsymbol{a}_{t+1}) \\ &\quad + \gamma^2 \sum_{\boldsymbol{a}_{t+2} \in \mathcal{A}} \pi(\boldsymbol{a}_{t+2}|\boldsymbol{s}_{t+2}) r(\boldsymbol{s}_{t+2}, \boldsymbol{a}_{t+2}) + \cdots \end{aligned} \tag{3.41}$$

$\mathcal{A}'$ excludes the redundant actions from $\mathcal{A}$, that is, the actions that bring the lowest rewards, thus,

$$r(\boldsymbol{s}_t, \boldsymbol{a}_t|\boldsymbol{a}_t \in \mathcal{A}') > r(\boldsymbol{s}_t, \boldsymbol{a}_t|\boldsymbol{a}_t \in \mathcal{A} \setminus \mathcal{A}'). \tag{3.42}$$

For the probability distribution of the two policies, the following equations hold.

$$\sum_{\boldsymbol{a}_t \in \mathcal{A}} \pi(\boldsymbol{a}_t|\boldsymbol{s}_t) = \sum_{\boldsymbol{a}_t \in \mathcal{A}} \pi'(\boldsymbol{a}_t|\boldsymbol{s}_t) = 1. \tag{3.43}$$

$$\pi'(\boldsymbol{a}_t|\boldsymbol{s}_t) = 0, \ \boldsymbol{a}_t \in \mathcal{A} \setminus \mathcal{A}', \tag{3.44}$$

$$\pi(\boldsymbol{a}_t|\boldsymbol{s}_t) \geq 0, \ \boldsymbol{a}_t \in \mathcal{A} \setminus \mathcal{A}'. \tag{3.45}$$

Based on Eq. (3.42)-(3.45), we can derive:

$$\begin{aligned} &\sum_{\boldsymbol{a}_t \in \mathcal{A}} \pi(\boldsymbol{a}_t|\boldsymbol{s}_t) r(\boldsymbol{s}_t, \boldsymbol{a}_t) \\ &= \sum_{\boldsymbol{a}_t \in \mathcal{A}'} \pi(\boldsymbol{a}_t|\boldsymbol{s}_t) r(\boldsymbol{s}_t, \boldsymbol{a}_t) + \sum_{\boldsymbol{a}_t \in \mathcal{A} \setminus \mathcal{A}'} \pi(\boldsymbol{a}_t|\boldsymbol{s}_t) r(\boldsymbol{s}_t, \boldsymbol{a}_t) \\ &< \sum_{\boldsymbol{a}_t \in \mathcal{A}'} \pi'(\boldsymbol{a}_t|\boldsymbol{s}_t) r(\boldsymbol{s}_t, \boldsymbol{a}_t) + \sum_{\boldsymbol{a}_t \in \mathcal{A} \setminus \mathcal{A}'} \pi'(\boldsymbol{a}_t|\boldsymbol{s}_t) r(\boldsymbol{s}_t, \boldsymbol{a}_t) \\ &= \sum_{\boldsymbol{a}_t \in \mathcal{A}} \pi'(\boldsymbol{a}_t|\boldsymbol{s}_t) r(\boldsymbol{s}_t, \boldsymbol{a}_t). \end{aligned} \tag{3.46}$$

Substituting Eq. (3.46) into Eq. (3.41), the inequality $V^{\pi}(\boldsymbol{s}_t) < V^{\pi'}(\boldsymbol{s}_t)$ can be obtained. Thus the conclusion. □

By recalling Eq. (3.29), the definition of V-value is the average accumulative discounted reward, $V^{\pi}(\boldsymbol{s}_t) = \mathbb{E}_{\boldsymbol{a}_t \sim \pi(\boldsymbol{a}_t|\boldsymbol{s}_t)}[R_t|\boldsymbol{s}_t]$. Based on Lemma 2, as $V^{\pi}(\boldsymbol{s}_t) \leq V^{\pi'}(\boldsymbol{s}_t)$, the policy with the reduced action space provides a higher average R_t than that of the fixed action space. In addition, the reduced action space helps the agent to avoid searching for low-reward actions, thereby reducing the computational time in exploration, which can be verified by simulation.

Re-designed Reward Function to Deal with Feasibility Issues

Without a carefully designed mechanism, the actions made in conventional AC-DRL may easily violate constraints, thus fail to guarantee the solution feasibility. In $\mathcal{P}_1$, the major difficulty comes from constraints (3.10b), whereas (3.10c)-(3.10h) can be satisfied by properly defined actions. Under the commonly-used reward designs, e.g., Eq. (3.38) or Eq. (3.39), constraint (3.10b) may not be satisfied since the criterion of the decision making is to minimize the objective energy without considering con-

straints. To solve the problem, we re-design the reward function by incorporating constraint (3.10b), which is given by:

$$r_t = \frac{\sum_{n=1}^{N} d_{n,t}^{\pi}}{(e_t^{\pi})^{\epsilon}}. \tag{3.47}$$

The rationale is that the proposed reward function is the ratio between the delivered data and the consumed energy on frame t, where ϵ is a control parameter. When ϵ is small, the reward enforces the UAV to deliver more data to meet users' demands. However, transmitting more data results in more energy consumption. To control energy growth, we can increase ϵ such that the agent will reduce the energy consumption to avoid the reward losses. Thus, by tuning an appropriate ϵ, the decisions made by AC-DSOS can achieve good energy-saving performance while satisfying users' demands.

For practical applications, AC-DSOS is designed to overcome the limitations brought by constrained combinatorial problems, e.g., guarantee feasibility and control exponentially-increased action space. Compared to conventional optimization and DRL approaches, AC-DSOS is expected to achieve a good trade-off between solution quality and complexity. From a theoretical perspective, AC-DSOS is of polynomial-time complexity, which provides a theoretical basis for its further real-time applications. The developed Lemma 2 proves that the reduced action space can lead to higher accumulative reward, which justifies the developed approaches and grantees the performance theoretically.

3.6 Numerical Results

In this section, we present numerical results to evaluate the performance of the proposed AC-DSOS algorithm and compare it with other schemes:

- Previous AC-DRL scheme: Deep deterministic policy gradient (DDPG) [58];
- Previous AC-DRL scheme: Proximal policy optimization (PPO) [60];
- High-complexity heuristic scheme: the proposed GSS-HEU in Alg. 3;
- High-complexity heuristic scheme: the alternative optimization algorithm (ALT-HEU) [88];
- Low-complexity heuristic scheme: semi-orthogonal user scheduling-based heuristic algorithm (SUS-HEU) [123];
- Optimal scheme: optimal algorithm (OPT).

DDPG and PPO provide performance benchmarks from the AC-DRL perspective. Both of them are based on stochastic policy gradient with fixed action space. The structure of the DNNs, parameter settings, and reward function, i.e., Eq. (3.47), for AC-DSOS, DDPG and PPO keep the same in order to enable a fair comparison. The proposed GSS-HEU, ALT-HEU in [88], SUS-HEU in [123], and OPT are the benchmark schemes from an optimization perspective. We implement ATL-HEU by applying its core idea of the block coordinate descent method to alternatively optimize two blocks, i.e., hovering time and user scheduling. SUS-HEU adopts a simple user-grouping strategy with lower complexity than GSS-HEU and ALT-HEU.

In the simulation, we first evaluate the performance of energy consumption and computational time. After that, we justify the developed new reward function in guaranteeing solution feasibility by comparing several well-known reward functions. Furthermore, we evaluate the convergence performance of AC-DSOS with different learning rates.

3.6.1 Parameter Settings

The UAV is equipped with $L = 10$ antennas serving $N = 3$ clusters. The ground users are randomly scattered in the service area. Each cluster contains up to $K = 9$ users. The users' demands $q_{k,n}$ are randomly selected from $\{1, 2, 3, 4, 5\}$ (Mbit). We assume the bandwidth $B = 10$ MHz, noise power $\sigma^2 = 0.1$ mW, hovering power $P_H = 10$ W, and transmit power $p_{k,g,n} = 3$ W, referring to [85]. Based on FSMC, we quantize $\beta_{g,n,t}^{(kk)}$ and $\beta_{g,n,t}^{(kj)}$ into 9 levels, $\{0, 0.3, 0.6, 0.9, 1.2, 1.5, 1.8, 2.1, 2.4\}$. The setting of the transfer probability matrix is similar in [124]. Two fully-connected DNNs are employed as the actor and the critic. The adopted parameters for implementing AC-DSOS are summarized in Table 3.2.

Table 3.2: Parameters in AC-DSOS

Parameters	Actor	Critic
Number of hidden layers	3	3
Number of nodes/layer	300	300
Activation function (hidden layers)	ReLU	ReLU
Activation function (output layer)	Sigmoid	None
Learning rate	0.003	0.002
Loss function	Eq. (3.24)	Eq. (3.31)
Optimizer	Adam	Adam
Batch size	64	64
Discount factor γ	0.9	
Memory size	10,000 tuples	
Number of learning episodes	400	
Value range $[-\kappa, \kappa]$ of $\hat{a}_{i,t}$	[-2, 2]	
Software platform	Python 3.6 with TensorFlow 0.12.1	

3.6.2 Results and Analysis

Trade-off Performance between Energy and Computational Time

Firstly, by comparing with six benchmarking algorithms in Fig. 3.5, the proposed AC-DSOS achieves a good trade-off between energy minimization and computational time. Note that for $K > 7$, the optimal energy results are absent due to the high complexity and the corresponding long computational time. From Fig. 3.5, AC-DSOS saves around 29.94% and 24.84% energy compared to DDPG and PPO on average. Overall, AC-DSOS provides a sub-optimal solution, with 19.17% gap to the optimum. GSS-HEU achieves near optimality, and consumes less 9.8% energy than AC-DSOS in average but with paying much higher complexity and time, e.g., see Fig. 3.6. ALT-HEU takes more computational time but the average energy-saving performance is 4.28% inferior to AC-DSOS as the algorithm is sensitive to the initial point. SUS-HEU consumes the highest energy since it schedules users based on channel conditions without considering energy consumption. It is also shown that the total objective energy follows a roughly linear increase in all the algorithms. The gaps between the optimal algorithm and other algorithms become larger as K increases. When K grows from 5 to 7, the gap to the optimum increases from 47.7% to 65.1% for SUS-HEU, and from 31% to 44.5% for DDPG. In AC-DSOS, since the delivery-completed users are deleted during the learning process, the size of the action space will continuously decrease. This improves the searching efficiency and quality, and reduces the growth rate of the gap as K increases, from 11.1% ($K = 5$) to 16.7% ($K = 7$).

Fig. 3.6 compares the computational time with respect to K. The computational time is accounted as the elapsed time of producing an optimized solution per frame. In GSS-HEU, ALT-HEU and OPT, the

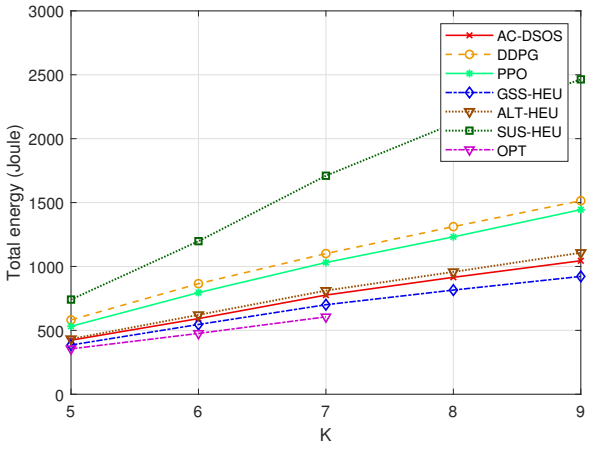


Figure 3.5: Total energy vs. K ($T_{max} = 160$).

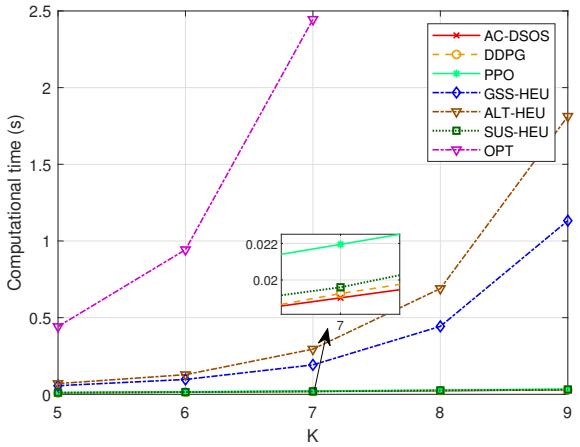


Figure 3.6: Average computational time vs. K ($T_{max} = 160$).

computational time grows exponentially with K, whereas the proposed AC-DSOS along with DDPG, PPO and SUS-HEU maintain at the millisecond magnitude and insensitive to K. In average, AC-DSOS saves 99.23%, 92.86%, and 89.98% computational time compared to OPT, GSS-HEU, and ALT-HEU, respectively. This is due to the fact that DRL can provide online decisions based on the current environment state instead of solving the optimization problem directly. PPO consumes 14.55% more computational time than AC-DSOS since calculating the gradient for a complex loss function consumes extra time. The computational time of AC-DSOS is slightly lower than DDPG and SUS-HEU. However, by recalling Fig. 3.5, AC-DSOS saves 24.84%, 29.94%, and 52.51% energy compared with PPO, DDPG, and SUS-HEU, respectively. In addition, we can observe that the computational time of GSS-HEU and OPT exceeds 1s when $K = 9$ and $K = 7$, respectively, which is impractical in the scenarios with strict delay requirements. For AC-DSOS, the result remains at the millisecond-level, even if the number of users increases from 5 to 9.

Fig. 3.7 demonstrates the total energy consumption with respect to T_{max}, and Fig. 3.8 illustrates the communication energy and hovering energy separately. From Fig. 3.7, AC-DSOS outperforms DDPG

and PPO by saving 21.37% and 18.45% total energy on average. The average gap between GSS-HEU and the optimal solution is 8.91% smaller than that of AC-DSOS, but, from Fig. 3.6, GSS-HEU consumes nearly 126 times higher calculation time than AC-DSOS at $T_{max} = 160$. The energy-saving performance of SUS-HEU is worse than other algorithms and its gap to the optimum reaches 59.44%. Fig. 3.7 also shows that, as T_{max} increases, the objective energy rapidly decreases first then grows steadily. This can be explained via Fig. 3.8. The objective energy consists of the communication energy and hovering energy. From Fig. 3.8, the communication energy drops rapidly when $T_{max} < 140$, and becomes stable after $T_{max} > 180$. Whereas, the hovering energy increases linearly with T_{max} for all the algorithms.

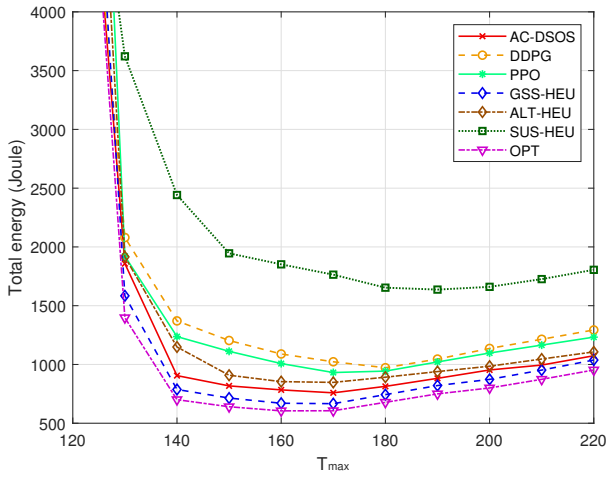


Figure 3.7: Total energy vs. T_{max} $(K = 7)$.

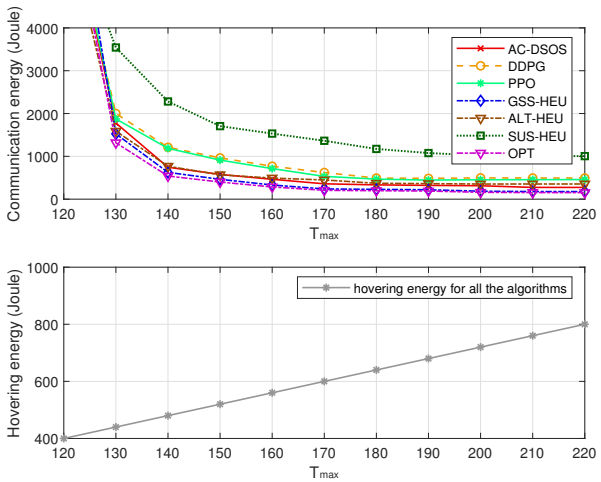


Figure 3.8: Communication and hovering energy vs. T_{max} $(K = 7)$.

Feasibility and Convergence Performance

Fig. 3.9 verifies the capability of the proposed reward function in dealing with feasibility issues, where a feasible solution is obtained only if the ratio of delivered demand over total demand in y-axis achieves 100%. From Fig. 3.9, the reward functions used in Eq. (3.38) and Eq. (3.39) fail to guarantee the feasibility of the solution. For the re-designed reward, we evaluate the performance by setting ϵ to 1, 1.2, and 1.5. A small ϵ means that transmitting more data can bring more rewards gain than saving energy. When ϵ drops below 1.2, the feasibility issue can be solved. Fig. 3.10 shows the objective energy with different ϵ. It can be found that a smaller ϵ leads to more energy consumption. Thus, an appropriate parameter ϵ lies at 1.2, enabling the after-learned solution to guarantee the demands while consuming less energy.

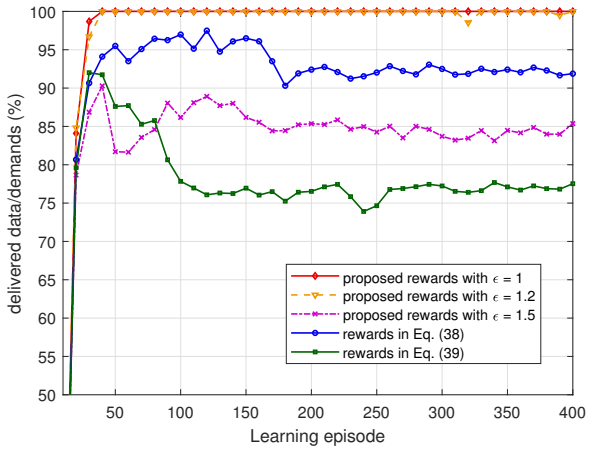


Figure 3.9: Feasibility vs. learning episode.

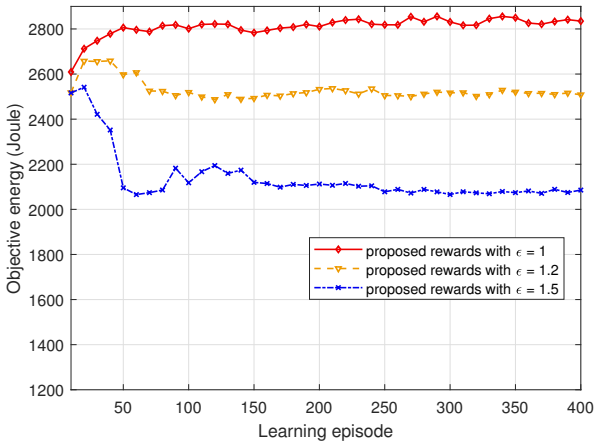


Figure 3.10: Energy vs. learning episode.

Fig. 3.11 demonstrates the convergence of AC-DSOS with different actor's learning rate α_a. The x-axis is the learning episode and the y-axis is the received accumulative reward R_m in the m-th episode. We define that the AC-DSOS algorithm converges if there exist $\bar{R}$ and a sufficiently large integer m_{con},

such that $|R_m - \bar{R}| < \varepsilon$ for all $m > m_{con}$, where ε is a positive tolerance. From 3.11, we can observe that when the learning rate $\alpha_a = 0.001$ and $\alpha_a = 0.003$, the curves converge around 80 episodes. As α_a increases to $\alpha_a = 0.005$, the curve fluctuates due to the large update step. Taking the actor as an example, the learning rate for the critic α_c has the same tendency. In conclusion, the learning rates of the actor and critic are sensitive to the convergence, and need to be properly selected, e.g., 0.003 for the actor.

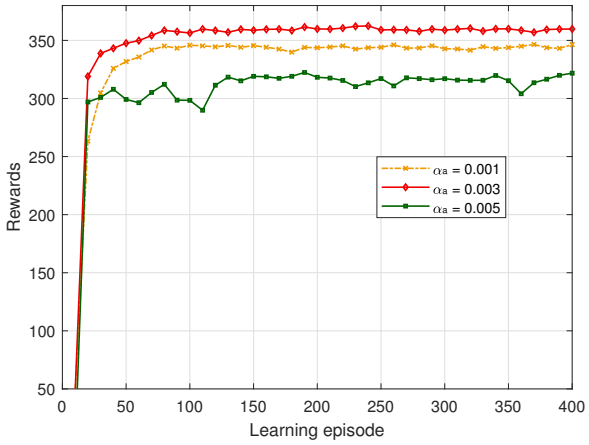


Figure 3.11: Rewards vs. learning episode.

Fig. 3.12 compares the policy with reduced action space and fixed action space in convergence speed and reward evolution. AC-DSOS with reduced action space converges around 60 episodes, against 250 episodes in AC-DSOS with fixed action space, and achieves 8.33% higher reward value than the other in average. In addition, we can observe that, with the fixed-large action space, the agent is likely to get stuck in local points, which can result in more time in exploration to escape from the points, referring to the red curve's step-like effect at the 100-200 episodes. Overall, the policy with the reduced action space is effective in improving learning efficiency and reward quality.

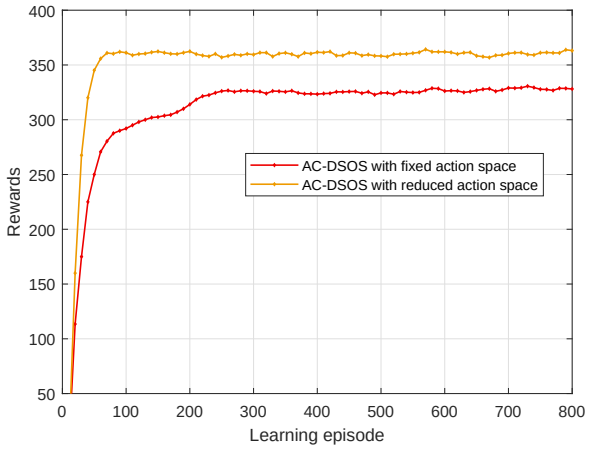


Figure 3.12: Rewards in AC-DSOS with fixed and reduced action space.

Performance Comparison in Dynamic Environments

In Fig. 3.13, we evaluate the capability of AC-DSOS in adapting network dynamics. Unlike previous static scenarios, we consider a dynamic scenario, where users request diverse amounts of data, and their arrival/departure in each cluster are varying over time by following the Poisson distribution. Starting from the 200-th episode, we assume that the entry/leave event happens every 100 episodes. For example, at the 200-th episode, some new users join the clusters and request date services. As a consequence, more energy is consumed (see the optimal energy consumption in OPT). In DDPG and AC-DSOS, the agents need time to learn and train to adapt to the new users due to the lack of their prior/historical knowledge. Both algorithms, therefore, undergo an adjusting period to converge to adapt to the environment change.

From the results, AC-DSOS demonstrates two advantages compared to DDPG. Firstly, AC-DSOS converges faster than DDPG. AC-DSOS is able to converge within 60 episodes such that it is more timely and adaptive to handle the periodically-changed network. Such improved computational efficiency and convergence are benefited by the developed action-space-reduction and policy-quantification approaches. In contrast, DDPG leads to a worse case. That is, the algorithm has not been converged to react to the first environment change but the second has arrived. As a result, DDPG is not able to converge. Secondly, compared to the performance in static cases, e.g., Fig. 3.5, the average gap in energy consumption between AC-DSOS and OPT remains stable, within 20%, whereas the performance of DDPG fluctuates dramatically.

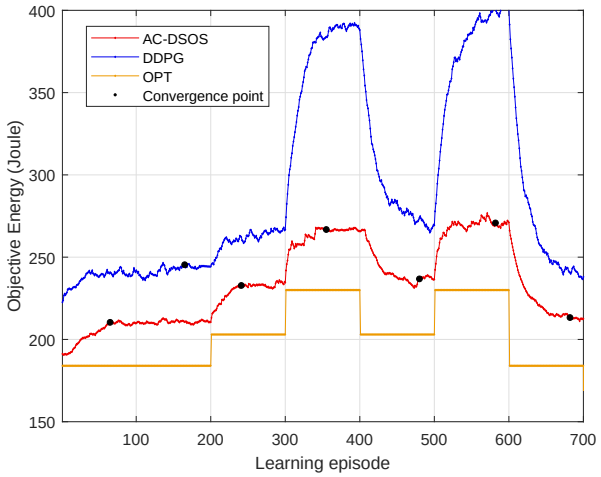


Figure 3.13: Energy comparison in a scenario with dynamic user arrival and departure.

3.7 Conclusion

In this chapter, we have investigated an energy minimization problem for UAV-aided communication systems from the perspective of AC-DRL. The formulated problem is combinatorial and non-convex. We provided an optimal method and proposed a GSS-based heuristic algorithm to solve the problem and serve as benchmarks. To make the solutions adaptive to online operations, we propose an AC-DSOS algorithm. Different from previous AC-DRL methods, the proposed AC-DSOS is able to deal with the huge discrete action space and guarantee feasibility. Numerical results have shown that AC-DSOS provides a good trade-off between energy efficiency and computational efficiency. Furthermore,

the re-designed reward function is effective to deal with the feasibility issue.

As a future extension, the proposed AC-DSOS can be further extended to adapt multi-UAV/multi-agent scenarios in two possible ways, i.e., distributed AC-DSOS and hybrid centralized-distributed AC-DSOS. For the former, a decentralized AC-DSOS is applied to each UAV/agent individually in a distributed multi-UAV system. Each agent learns its own value function network and strategy network without considering the mutual influence of other agents. For the latter, AC-DSOS is extended to enable centralized training and decentralized execution, where AC-DSOS is to use the global Q-value for each agent to update the local policy.

Chapter 4

Centralized Learning: Dynamic-Adaptive Scheduling with Meta Learning

4.1 Introduction

In beyond 5G networks (B5G), the massive number of connected users and their increasing demands for high-data-rate services can lead to overloading of terrestrial base stations (BSs), which in turn results in degraded user experience, e.g., longer delay in requesting data services or lower data rate [125]. In order to improve the network performance and user experience, the integration of satellites, e.g., low earth orbit (LEO) satellites, and terrestrial systems is considered as a promising solution to provide cost-efficient data services [36]. The solutions for terrestrial network optimization and resource management might not be suitable for direct application to integrated satellite-terrestrial systems [37]. In the literature, tailored schemes have been investigated to improve the networks' performance. In [38], the authors proposed a user scheduling scheme to maximize the sum-rate and the number of accessed users by utilizing the LEO-based backhaul. In [126], a joint power allocation and user scheduling scheme was proposed to maximize the network throughput in hierarchical LEO systems with the constraint of transmission delay. In [127], the authors developed a joint resource block allocation and power allocation algorithm to maximize the total transmission rate for LEO systems. It is worth noting that the resource optimization problems in LEO-terrestrial networks are typically combinatorial and non-convex. The conventional iterative optimization methods, e.g., in [38, 126, 127], are unaffordable for real-time operations due to their high computational complexity.

4.1.1 Related Works: State-of-the-art and Limitations

Towards an efficient solution, various learning techniques have been studied. Compared to supervised learning, reinforcement learning (RL) learns the optimal policy from observed samples without preparing labeled data. As one of the promising RL methods, deep reinforcement learning (DRL) adopts deep neural networks (DNNs) for parameterization and rapid decision making. Recent works have applied RL/DRL for resource management in LEO-terrestrial systems [128, 129, 130]. In [128], to maximize the achievable rate in LEO-assisted relay networks, a DQN-based algorithm was proposed to make the online decisions for link association. The authors in [129] adopted multi-agent reinforcement learning to minimize the average number of handovers and improve the efficiency of channel utilization for LEO satellite systems. In [130], the authors applied an actor-critic (AC) algorithm to LEO resource allocation, such as beam allocation and power control. The above RL algorithms in practical LEO systems are

limited by the following issue. That is, the performance of a learning model largely depends on the data originated from the experienced samples or the observed environment, but the wireless environment is highly complex and dynamic. When network parameters vary dramatically, the performance of the learning models can be degraded. To remedy this, one has to re-collect a large number of training data and re-train the learning models, which is time-consuming and inefficient to adapt to fast variations [43].

To address this issue, a variety of studies focus on how to make the learning models quickly respond to dynamic environments. Transfer learning applies the knowledge acquired from a source learning task to a target learning task to speed up the re-training process and reduce the volume of the collected new data sets [131]. The performance of transfer learning is limited by finding correlated tasks. Another approach, joint learning, aims at obtaining a single model that can be adapted to dynamic environments by optimizing the loss function over multiple tasks [132]. Besides, continual learning can also accelerate the adaptation to the new learning task by adding the experienced data from the previous tasks to the re-training data set, thus avoiding completely forgetting previously learned models [133]. Joint learning and continual learning might have good learning performance on average but have limited generalization abilities when different tasks are highly diversified [134]. In contrast, meta-learning extracts meta-knowledge and achieves good performance for specific tasks without requiring the related source tasks. The authors in [135] proposed a model-agnostic meta-learning algorithm (MAML) to obtain the model's initial parameters as meta-knowledge to quickly adapt to new tasks. In [136], an algorithm combining actor-critic with MAML (AC-MAML) was developed to learn a new task from fewer experience data sets. In [137], the authors proposed a promising meta-critic learning framework with better performance than conventional AC and AC-MAML. In [138], a meta-learning-based adaptive sensing algorithm was proposed, which determines the next most informative sensing location in wireless sensor networks. In [139], meta-learning was applied to find a common initialization vector that enables fast training of an autoencoder for the fading channels. Most of the meta-learning methods were applied in the areas of pattern recognition [135], robotics [136, 137], and physical layer communications [139]. The considered learning tasks in these works are simple and the action space is small, e.g., [138, 139]. However, when the learning techniques, e.g., DRL, AC-MAML, or meta-critic learning, are applied to address combinatorial optimization problems in a dynamic LEO-terrestrial network, the action space can be huge and the input-output relationships can become more complex. These may degrade the efficiency of the above learning methods.

4.1.2 Motivations and Contributions

Moving beyond the state-of-the-art, this chapter intends to address the following questions:

- Which learning technique can lead to higher performance gain in addressing resource management problems for dynamic LEO-terrestrial networks?
- How to deal with the huge action space and improve the learning efficiency?
- How to make the learning solutions more adaptive to dynamic environments?

In this study, we design an enhanced meta-critic learning algorithm (EMCL) for dynamic LEO-terrestrial systems. To the best of our knowledge, this is arguably the first work to present meta-critic learning to address resource scheduling problems and emphasize the adaptation to non-ideal dynamic environments. The major contributions are summarized as follows:

- We design a tailored metric for over-loaded LEO systems with dense user distribution, aiming at serving more users and delivering a higher volume of requested data.

- We formulate the resource scheduling problem as a quadratic integer programming (QIP) and provide two offline optimization-based benchmarks, i.e., optimal branch and bound (B&B) algorithm and suboptimal alternating direction method of multipliers-based heuristic algorithm (ADMM-HEU).
- Due to the combinatorial nature and the high complexity of the offline solutions, we solve the problem from the perspective of DRL by reformulating a Markov decision process (MDP) to make online decisions with the identical objective as the original problem.
- To enhance the adaptation to dynamic environments, we propose an EMCL algorithm based on the meta-critic framework. Compared to the conventional meta-learning, the novelty stems from that 1) the critic has a good generalization ability to evaluate any new task such that the learning agent can adjust the policy timely when the environment changes; 2) the tailored design of a hybrid neural network extracts the features from the current and historical samples; 3) the integrated Wolpertinger policy allows the actor to make decisions more efficiently in an exponentially increasing action space.
- To identify a promising solution, we evaluate the proposed EMCL with other benchmarks in three practical dynamic scenarios, i.e., bursty user demands, dramatically fluctuated channel states, and user departure/arrival. The numerical results verify EMCL's effectiveness and fast-response capabilities in adapting to dynamic environments.

The rest of the chapter is organized as follows. The system model is presented in Section 4.2. We formulate a resource scheduling problem and develop optimal and suboptimal solutions for performance benchmarks in Section 4.3. In Section 4.4, we model the problem as an MDP and develop an EMCL algorithm. Numerical results are demonstrated and analyzed in Section 4.5. Finally, Section 4.6 concludes the chapter.

4.2 System Model and Problem Formulation

4.2.1 LEO-Terrestrial Network

In practice, terrestrial BSs can become over-loaded and congested. This common issue has received considerable attention from the academia, industry, and standardization bodies, e.g., 3GPP Release 17 [140]. In this work, we address this challenging issue via developing satellite-aided solutions. As shown in Fig. 4.1, the BSs with limited resources might not be able to serve all the users and deliver all the requested data demands within a required transmission or queuing delay. To relieve the burden of the terrestrial BSs, LEO satellites are introduced to offload traffic from BSs or provide backhauling services. The LEO employs a transparent payload. For the spectrum usage, the system keeps consistent with currently deployed space and ground systems. That is, the LEO satellites operate at the Ka-band to provide broadband services to advanced terminals, e.g., equipped with very small aperture terminals (VSAT), while the 5G terrestrial system adopts sub-6GHz at the C-band to serve normal mobile devices, e.g., smartphones [141]. The co-existence of the C-band or Ka-band between terrestrial and satellite segments can introduce extra co-channel interference and thus require more sophisticated anti-interference approaches, which is out of the scope of this work.

We consider two types of mobile terminals (MTs) in the system. The first type is the normal cellular terminals, e.g., cell phones, that can be served by BSs or terrestrial-satellite terminals (TSTs), but cannot be served by LEO due to the size limitation of dish antennas. The other is the dual-mode terminals, e.g., vehicular terminals, which are equipped with a 3GPP terrestrial-non-terrestrial network (TN-NTN)

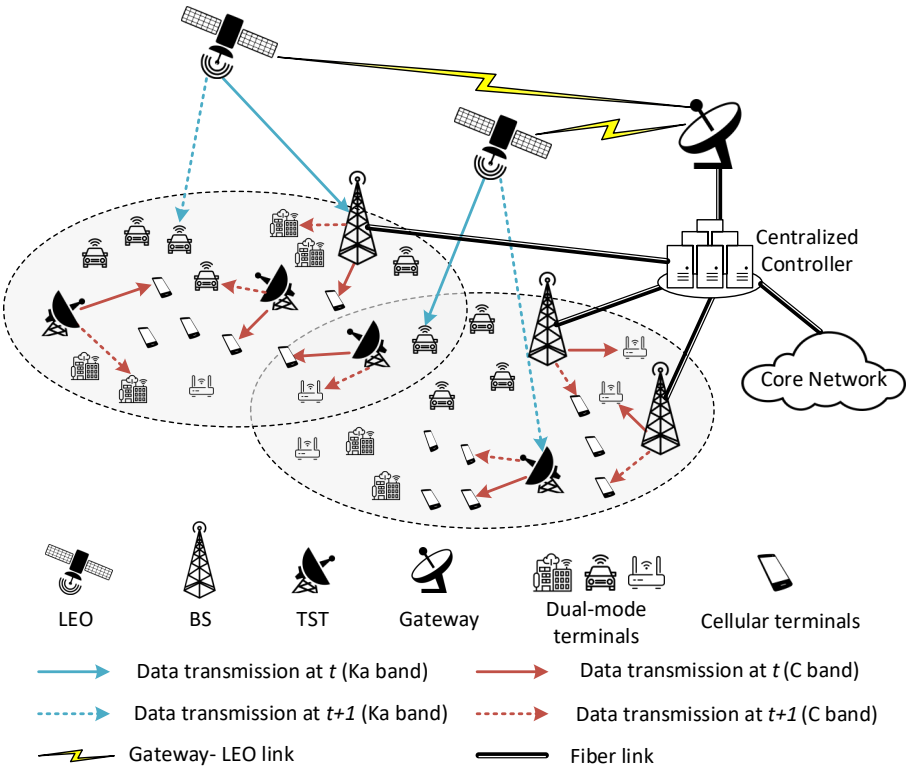


Figure 4.1: An illustrative LEO-terrestrial communication system

compliant dual-mode that can be either served by LEO via Ka-band (in rural areas) or by BS/TST through C-band (in urban areas) [142]. Compared to BS, TST is a small-size terminal that acts as a flexible and cost-saving access point, e.g., Starlink ground terminals. A TST can receive backhauling services from LEO over Ka-band and transmit data to MTs over C-band [38]. The terrestrial BSs can request data from the core network through optical fiber links or from the LEO satellites through the BS-LEO link.

We remark that Fig. 4.1 can be extended to a large-scale network with a massive number of MTs. Specifically, an MT in Fig. 4.1 can represent a cluster of densely-deployed devices. Due to the proximity, the channel states of the devices within a cluster can be assumed identical. When a cluster is scheduled, all the devices within the cluster will be scheduled by the TDMA (or FDMA) mode to avoid intra-cluster interference.

We denote $\mathcal{S}, \mathcal{B}, \mathcal{M}$ and $\mathcal{L}$ as the set of TSTs, BEs, MTs, and LEOs, respectively, where $\mathcal{M}$ is the union of set $\mathcal{M}_1$ (all the cellphone MTs) and $\mathcal{M}_2$ (all the dual-mode MTs). Thus, the union of receivers, i.e., ground devices (GDs), can be expressed as $\mathcal{K} = \mathcal{S} \cup \mathcal{B} \cup \mathcal{M} = \{1, ..., k, ..., K\}$, where $K = |\mathcal{S}|+|\mathcal{B}|+|\mathcal{M}|$. Similarly, the union of transmitters is written by $\mathcal{N} = \mathcal{S} \cup \mathcal{B} \cup \mathcal{L} = \{1, ..., n, ..., N\}$, where $N = |\mathcal{S}| + |\mathcal{B}| + |\mathcal{L}|$. We assume that the transmission tasks are delay-sensitive and to be completed within T time slots. The time domain is divided by time slots, i.e., $\mathcal{T} = \{1, ..., t, ..., T\}$. In data transmission, each transmitter n serves a GD in unicast mode, i.e., no joint transmission and no multi-cast transmission. Within a time slot, multiple transmitter-GD links can be activated, forming a link group. We denote $\mathcal{G} = \{1, ..., g, ..., G\}$ as a set by enumerating all the valid link groups.

To coordinate the link scheduling between terrestrial and satellite parts, a centralized controller is deployed in the system [143]. With the centralized controller, the information from the ground and satellite can be collected and exchanged, which facilitates the coordination of scheduling different types of links. In addition, efficient synchronization approaches can be implemented on the transmitters and receivers to guarantee that the resource scheduling updates are performed accurately in LEO satellite

systems [144].

4.2.2 Channel Modeling

We consider time-varying channels for both satellite and terrestrial communication. At time slot t, the channel state between receiver k and transmitter n can be modeled as:

$$h_{k,n,t} = \begin{cases} G_{leo}^{(T)} \cdot G_{k,n,t}^{(C)} \cdot G^{(R)}, & n \in \mathcal{L}, \\ G_{ter}^{(T)} \cdot G_{k,n,t}^{(C)} \cdot G^{(R)}, & n \in \mathcal{N} \backslash \mathcal{L}, \end{cases} \tag{4.1}$$

where $G_{leo}^{(T)}$ and $G_{ter}^{(T)}$ are the transmit antenna gain of LEO and terrestrial BS/TST, respectively. We assume that all the GDs are equipped with a single receiving antenna, so that their receive antenna gains $G^{(R)}$ are uniform. $G_{k,n,t}^{(C)}$ represents the channel fading between transmitter n and GD k at time slot t. For LEO-to-GD channel, a widely used channel fading model in [38, 127, 145] is adopted, which includes free-space path loss, pitch angle fading, atmosphere fading, and Rician small-scale fading:

$$G_{k,n,t}^{(C)} = \left(\frac{c}{4\pi d_{k,n,t} f_{leo}} \right)^2 \cdot G_{k,n}^{(P)} \cdot A(\Omega) \cdot \varphi, \tag{4.2}$$

where c is the speed of light, $d_{k,n,t}$ is the propagation distance between LEO and the terminals, f_{leo} is the carrier frequency of LEO, $G_{k,n}^{(P)}$ is the pitch angle fading gain, and φ is the Rician fading gain. The atmospheric fading gain $A(\Omega)$ is the function of the angle Ω, where $\sin \Omega = H/d_{k,n,t}$, and H is the altitude of LEO.

$$A(\Omega) = 10^{\left(\frac{3\chi}{10 \sin \Omega}\right)}, \tag{4.3}$$

where χ, in dB/km, is the attenuation through the clouds and rain. In downlink transmission, we assume that Doppler shift caused by the high mobility of LEO can be perfectly pre(post)-compensated in the gateway based on the predictable satellite motion and speed [146]. For terrestrial channels, i.e., TST/BS-to-MT, $G_{k,n,t}^{(C)}$ consists of the path loss and Rayleigh small-scale fading [147], which is given by:

$$G_{k,n,t}^{(C)} = \left(\frac{c}{4\pi d_{k,n,t} f_{ter}} \right)^2 \cdot \phi, \tag{4.4}$$

where f_{ter} is the carrier frequency of TST/BS and ϕ is the Rayleigh fading factor.

Based on the adopted channel fading models (4.2) and (4.4), we further model the time-varying channel as the first state Markov channel (FSMC) to capture the time-correlation characteristics and conduct mathematically tractable analysis [113]. We discretize each channel state $h_{k,n,t}$ into L levels, $\mathcal{H} = \{h_1, ..., h_L\}$, and the transition probability matrix is defined as:

$$\mathbf{P} = \begin{bmatrix} P_{1,1} & \cdots & P_{1,L} \\ \vdots & \ddots & \vdots \\ P_{L,1} & \cdots & P_{L,L} \end{bmatrix}, \tag{4.5}$$

where an element $P_{l,l'}$ can be written as:

$$P_{l,l'} = \text{Prob}\left[h_{k,n,t+1} = h_{l'} | h_{k,n,t} = h_l\right], \; h_l, h_{l'} \in \mathcal{H}. \tag{4.6}$$

That is, at a given time slot t, if $h_{k,n,t} = h_l$, $P_{l,l'}$ refers to the probability of channel state at the next time slot $h_{k,n,t+1}$ transiting from h_l to $h_{l'}$.

4.2.3 Optimization Problem

We formulate a resource scheduling problem for the considered over-loaded LEO-5G systems. We use binary indicators $\alpha_{k,n,g}$ to represent the activated links in group g, where $\alpha_{k,n,g} = 1$ if the transmitter-GD link (n, k) is included in group g and will be activated when group g is scheduled, otherwise, 0. We remark that only valid links, i.e., satisfying the following constraints, can form a link group.

$$\alpha_{k,n,g} = 0,\ \forall k \in \mathcal{K}\backslash\mathcal{M}, \forall n \in \mathcal{N}\backslash\mathcal{L}, \forall g \in \mathcal{G}, \tag{4.7}$$

$$\alpha_{k,n,g} = 0,\ \forall k \in \mathcal{M}_1,\ \forall n \in \mathcal{L},\ \forall g \in \mathcal{G}, \tag{4.8}$$

$$\sum\nolimits_{n\in\mathcal{N}} \alpha_{k,n,g} \leq 1,\ \forall k \in \mathcal{K}, \forall g \in \mathcal{G}, \tag{4.9}$$

$$\sum\nolimits_{k\in\mathcal{K}} \alpha_{k,n,g} \leq 1,\ \forall n \in \mathcal{N}, \forall g \in \mathcal{G}. \tag{4.10}$$

(4.7) and (4.8) exclude certain types of links, i.e., BS-BS, TST-TST, BS-TST, TST-BS, and LEO-cellphone. (4.9) means that each GD k in group g receives data from at most one transmitter, and (4.10) represents each transmitter n in group g serves no more than one GD. For example, consider a simple system with 1 LEO, 1 TST, 1 BS, and 2 MTs (an MT1 in $\mathcal{M}_1$, and an MT2 in $\mathcal{M}_2$). There are four possible receivers, i.e., TST, BS, MT1, and MT2, indexed by $K = \{1, 2, 3, 4\}$, respectively, and three possible transmitters, i.e., TST, BS, and LEO, indexed by $\mathcal{N} = \{1, 2, 3\}$. Filtered by (7)-(8), all the valid links (n, k) are $(1, 3)$ (TST to MT1), $(1, 4)$ (TST to MT2), $(2, 3)$ (BS to MT1), $(2, 4)$ (BS to MT2), $(3, 1)$ (LEO to TST), $(3, 2)$ (LEO to BS), and $(3, 4)$ (LEO to MT2). Confined by (9)-(10), a combination of the above links can be a valid group g, e.g, a group $\{(3, 4), (1, 3)\}$ contains two links. Enumerating all the valid groups forms set $\mathcal{G} = \{\{(1, 3), (2, 4)\}, \{(1, 3), (3, 4)\}, ..., \{(1, 3), (2, 4), (3, 1)\}\}$, which is served as the input set for decision making.

Confined by (4.7) and (4.8), the SINR and the volume of transmitted data of GD k in group g at time slot t are expressed in (4.11) and (4.12), respectively.

$$\gamma_{k,g,t} = \frac{\sum_{n\in\mathcal{L}} h_{k,n,t}\alpha_{k,n,g}p_{k,g}}{\sum_{j\in\mathcal{K}\backslash k}\sum_{n\in\mathcal{L}} h_{j,n,t}\alpha_{j,n,g}p_{k,g} + \sigma^2} + \frac{\sum_{n\in\mathcal{N}\backslash\mathcal{L}} h_{k,n,t}\alpha_{k,n,g}p_{k,g}}{\sum_{j\in\mathcal{K}\backslash k}\sum_{n\in\mathcal{N}\backslash\mathcal{L}} h_{j,n,t}\alpha_{j,n,g}p_{k,g} + \sigma^2}, \tag{4.11}$$

and

$$R_{k,g,t} = \Phi B_{k,g} \log_2(1 + \gamma_{k,g,t}), \tag{4.12}$$

where $p_{k,g}$ is the transmit power to GD k in group g and Φ is the duration of each time slot. We denote B_{leo} and B_{ter} are the fixed bandwidth for LEO and BS/TST, respectively, such that the used bandwidth $B_{k,g}$ for GD k in group g can be calculated by $B_{leo}\sum_{n\in\mathcal{L}} \alpha_{k,n,g} + B_{ter}\sum_{n\in\mathcal{N}\backslash\mathcal{L}} \alpha_{k,n,g}$. We define the decision variables as $\mathbf{x} = [x_{1,1}, ..., x_{g,t}, ..., x_{G,T}]$ where

$$x_{g,t} = \begin{cases} 1, & \text{if group } g \text{ is scheduled at time slot } t, \\ 0, & \text{otherwise.} \end{cases}$$

In a practical over-loaded scenario, not all the terminals can be timely served and their actual demands may not be fully delivered in time due to massive access requests competing for limited resources. Under this undesirable scenario, the optimization task may shift from "serving all the terminals and satisfying all the demands" to "serving as many terminals (and their demands) as possible". On this basis, we

denote D_k and $D'_k (< D_k)$ as the actual demand (in bits) and the threshold, respectively. In the objective design, we consider a composite utility function in (4.13), and define that GD k is served, i.e., $f_k(\mathbf{x}) = 1$, when a threshold D'_k is satisfied.

$$f_k(\mathbf{x}) = \mathbb{1}\left(\sum_{t\in\mathcal{T}}\sum_{g\in\mathcal{G}} R_{k,g,t}x_{g,t} - D'_k\right), \tag{4.13}$$

where $\mathbb{1}(\cdot)$ is an indicator function such that $\mathbb{1}(\beta) = \begin{cases} 1, & \text{if } \beta > 0 \\ 0, & \text{if } \beta \leq 0 \end{cases}$. We set a threshold D'_k in (4.13) since the system may not be able to satisfy the actual demand considering the over-loaded scenario with densely deployed users. We convert the non-linear function $f_k(\mathbf{x})$ to a linear function by introducing auxiliary variables $\mathbf{y} = [y_1, ..., y_k, ..., y_K]$ and linear constrains (4.14d), where $y_k = f_k(\mathbf{x})$. The optimization problem is formulated as:

$$\mathbf{P1}: \min_{x_{g,t}, y_k} f(\mathbf{x}, \mathbf{y}) = \eta_0 \left(\sum_{k\in\mathcal{K}} y_k - K\right)^2 + \sum_{k\in\mathcal{K}} \eta_k \left(\sum_{t\in\mathcal{T}}\sum_{g\in\mathcal{G}} R_{k,g,t}x_{g,t} - D_k\right)^2 \tag{4.14a}$$

$$s.t.\ \bar{\gamma}_k - \gamma_{k,g,t} \leq V\left(1 - x_{g,t}\sum_{n\in\mathcal{N}} \alpha_{k,n,g}\right),\ \forall k \in \mathcal{K}, \forall g \in \mathcal{G}, \forall t \in \mathcal{T}, \tag{4.14b}$$

$$\sum_{g\in\mathcal{G}} x_{g,t} \leq 1,\ \forall t \in \mathcal{T}, \tag{4.14c}$$

$$D'_k y_k \leq \sum_{t\in\mathcal{T}}\sum_{g\in\mathcal{G}} R_{k,g,t}x_{g,t},\ \forall k \in \mathcal{K}, \tag{4.14d}$$

$$x_{g,t} \in \{0, 1\},\ \forall g \in \mathcal{G},\ \forall t \in \mathcal{T}, \tag{4.14e}$$

$$y_k \in \{0, 1\},\ \forall k \in \mathcal{K}, \tag{4.14f}$$

where $\bar{\gamma}_k$ is the SINR threshold of GD k, V is a positive sufficiently large value, and $\eta_0, ..., \eta_K$ are the weight factors. Considering the users' fairness and resource utilization in an over-loaded system, we design a tailored utility function (5.20a) consisting of two components. The first term encourages to serve more users and meet their minimum requirement D'_k since satisfying low-traffic users are more likely to have rewards in the objective. The second term aims at minimizing the supply-demand gap such that the scheduler tends to serve the users with higher demand D_k or higher weights η_k $(k = 1, ..., K)$. The priority or importance of the two parts can be adjusted by pre-defined weight values according to different scenarios. For example, when a large number of delay-sensitive and low-traffic users enter the network, the scheduler may give more priority by increasing η_0 to serve this type of users as many as possible, while the delay-tolerate services with high data demand may have lower priority (with decreased η_k) in this scheduling cycle.

- The constraints (4.14b) represent the SINR requirement in practical satellite and 5G systems. If GD k in group g is scheduled at time slot t, i.e., $x_{g,t}\sum_{n\in\mathcal{N}} \alpha_{k,n,g} = 1$, the SINR of GD k should be higher than the threshold $\bar{\gamma}_k$ to guarantee the link quality. This also implies that scheduling many links with strong co-channel interference may not be a wise option in the optimal solution. The setting of $\bar{\gamma}_k$ refers to the standard of DVB-S2X [148] and 3GPP Release 16 [149].

- The constraints (4.14c) represent no more than one group can be scheduled in a time slot.

- In constraints (4.14d), we define that if GD k is served, i.e., $y_k = 1$, the received data should be larger than D'_k.

4.3 Characterization on Solution Development

In this section, we propose an optimal method and a heuristic approach as the offline benchmarks for small-medium and large-scale instances, respectively. In addition, we outline conventional online-learning solutions and their limitations.

4.3.1 The Proposed Optimal and Sub-optimal Solutions

Towards the optimum of $\mathbf{P1}$, we first identify the convexity of $\mathbf{P1}$ when the binary variables are relaxed.

Lemma 3. *The relaxation problem of* $\mathbf{P1}$ *is convex.*

Proof. See Appendix 6.6. □

Based on Lemma 3, we conclude that $\mathbf{P1}$ is an integer convex optimization problem. The optimum can be obtained by B&B that solves a convex relaxation problem at each node, with the complexity $\mathcal{O}(2^{G\times T+K})$ [150]. Although the complexity increases exponentially, the B&B-based approach can provide a performance benchmark at least for small-medium instances.

To reduce the complexity in solving the large-scale problems, we develop a suboptimal algorithm. We observe that $\mathbf{P1}$ has a variable-splitting structure, which motivates the development of ADMM based approaches [151]. The algorithm is summarized in Alg. 5, first solving the convex relaxation problem of $\mathbf{P1}$ based on ADMM (in lines 2-8), followed by a rounding operation (in lines 9-13). In ADMM, we divide the relaxed variables into $T+1$ blocks $\hat{\mathbf{x}}_1, ..., \hat{\mathbf{x}}_T, \hat{\mathbf{y}}$, where $\hat{\mathbf{x}}_t = [\hat{x}_{1,t}, ..., \hat{x}_{G,t}]$, and introduce auxiliary variables $\mathbf{z} = [z_1, ..., z_K]$, where

$$z_k = D'_k \hat{y}_k - \sum_{g\in\mathcal{G}} \sum_{t\in\mathcal{T}} R_{k,g,t} \hat{x}_{g,t}, \forall k \in \mathcal{K}. \tag{4.15}$$

The inequality constraints (4.14d) are replaced by:

$$z_k \leq 0, \ \forall k \in \mathcal{K}. \tag{4.16}$$

The augmented Lagrangian function is expressed as:

$$\begin{aligned} & L(\hat{\mathbf{x}}_1, ..., \hat{\mathbf{x}}_T, \hat{\mathbf{y}}, \mathbf{z}, \boldsymbol{\lambda}) \\ =& f(\hat{\mathbf{x}}, \hat{\mathbf{y}}) + \sum_{k\in\mathcal{K}} \lambda_k \left(z_k - D'_k \hat{y}_k + \sum_{g\in\mathcal{G}} \sum_{t\in\mathcal{T}} R_{k,g,t} \hat{x}_{g,t} \right) \\ +& \frac{\rho}{2} \sum_{k\in\mathcal{K}} \| z_k - D'_k \hat{y}_k + \sum_{g\in\mathcal{G}} \sum_{t\in\mathcal{T}} R_{k,g,t} \hat{x}_{g,t} \|^2, \end{aligned} \tag{4.17}$$

where $\rho > 0$ is the penalty parameter and $\boldsymbol{\lambda} = [\lambda_1, ..., \lambda_K]$ are the lagrangian multipliers. We define I_{iter} as the total number of iterations of the algorithm. In each iteration i, ADMM update each variable

Algorithm 5 ADMM-HEU

Inputs: D_k, D'_k and $R_{k,n,t}$.

1: Relax $\mathbf{P1}$ to a continuous problem $\mathbf{P1}'$.
2: Initialize $\hat{\mathbf{x}}_t^0$, $\hat{\mathbf{y}}^0$, $\mathbf{z}^0$, $\boldsymbol{\lambda}^0$ and $i = 0$.
3: **for** $i = 0, ..., I_{iter}$ **do**
4: Update $\hat{\mathbf{x}}_t, \hat{\mathbf{y}}$ and $\mathbf{z}$ by Eq. (4.18), (4.19) and (4.20).
5: $\lambda_k^{i+1} = \lambda_k^i + \rho \left(z_k^i - D'_k \hat{y}_k^i + \sum_{g\in\mathcal{G}} \sum_{t\in\mathcal{T}} R_{k,g,t} \hat{x}_{g,t}^i \right)$.
6: **end for**
7: Obtain relaxed solution $\hat{x}_{g,t}$.
8: **for** $t \in \mathcal{T}$ **do**
9: Find $g^\dagger = \underset{g\in\mathcal{G}}{\operatorname{argmax}} \{\hat{x}_{1,t}, ..., \hat{x}_{G,t}\}$.
10: Set $x^*_{g^\dagger,t} = 1$ and $x^*_{g,t} = 0$, $\forall g \neq g^\dagger$.
11: **end for**
12: Calculate y_k^* based on Eq. (4.13).

Outputs: $x^*_{g,t}$ and y_k^*

block as follows (in line 5) and update multipliers (in line 6):

$$\hat{\mathbf{x}}_t^{i+1} = \underset{\hat{\mathbf{x}}_t\in\mathcal{X}_t}{\operatorname{argmin}} L(\hat{\mathbf{x}}_1^i, ..., \hat{\mathbf{x}}_T^i, \hat{\mathbf{y}}^i, \mathbf{z}^i, \boldsymbol{\lambda}^i),\ \forall t \in \mathcal{T}, \tag{4.18}$$

$$\hat{\mathbf{y}}^{i+1} = \underset{\hat{\mathbf{y}}\in\mathcal{Y}}{\operatorname{argmin}} L(\hat{\mathbf{x}}_1^i, ..., \hat{\mathbf{x}}_T^i, \hat{\mathbf{y}}^i, \mathbf{z}^i, \boldsymbol{\lambda}^i), \tag{4.19}$$

$$\mathbf{z}^{i+1} = \underset{\mathbf{z}\in\mathcal{Z}}{\operatorname{argmin}} L(\hat{\mathbf{x}}_1^i, ..., \hat{\mathbf{x}}_T^i, \hat{\mathbf{y}}^i, \mathbf{z}^i, \boldsymbol{\lambda}^i), \tag{4.20}$$

where $\mathcal{X}_t = \{\mathbf{x}_t | (4.14b), (4.14c), (4.14d)\}$, $\mathcal{Y} = \{\mathbf{y} | 0 \leq y_k \leq 1\}$ and $\mathcal{Z} = \{\mathbf{z} | z_k \leq 0\}$. When ADMM terminates, the continuous solution $\hat{x}_{g,t}$ is obtained in line 8. The rounding process is then carried out in lines 10-13 to convert the largest $\hat{x}_{g,t}$ in each time slot to 1 (selecting the most promising group g for each t) and keep others 0.

The developed ADMM-HEU can provide sub-optimal benchmarks within an acceptable time span, since the subproblems in (4.18)-(4.20) can be solved in a parallel manner and with a smaller size than the original problem. However, ADMM-HEU requires $\mathcal{O}(1/\epsilon^2)$ iterations to achieve ϵ-optimality, where ϵ is set as $\frac{\mu}{T(T+3)}$ [152]. At each iteration, we can solve the $T+2$ variable blocks by B&B with the time complexity of $\mathcal{O}(T \cdot 2^G + 2 \cdot 2^K)$. Thus, the total complexity is given by $\mathcal{O}(T^5 \cdot 2^G + T^4 \cdot 2^K)$, which might not sufficient for fast adaptation to network variations.

4.3.2 Conventional Online-Learning Solutions and Limitations

To enable an intelligent and online solution, we address the problem from an RL perspective. Firstly, we briefly introduce actor-critic and meta-critic learning approaches as a basis to present the proposed EMCL. AC is an RL algorithm that takes advantage of both value-based methods, e.g., Q-learning, and policy-based methods, e.g., REINFORCE, with fast convergent properties and the capability to deal with continuous action spaces [153]. The learning agent in AC contains two components, where the actor is responsible for making decisions while the critic is used for evaluating the decisions by the value

functions. Specifically, at each learning step t[1], the actor takes action based on a stochastic policy, i.e., $a_t \sim \pi(a|s_t)$, where $\pi(a|s_t)$ is the probability of taking an action under state s_t, typically following the Gaussian distribution [121]. The critic is to generate a Q-value function $Q(s_t, a_t) = \mathbb{E}_\pi[\bar{r}_t|s_t, a_t]$, where $\bar{r}_t$ is the accumulated reward at step t, and $\mathbb{E}_\pi[\beta]$ is the expected value of β over the policy π. The goal of the learning agent is to find a policy to maximize the expected accumulated reward (or Q-value).

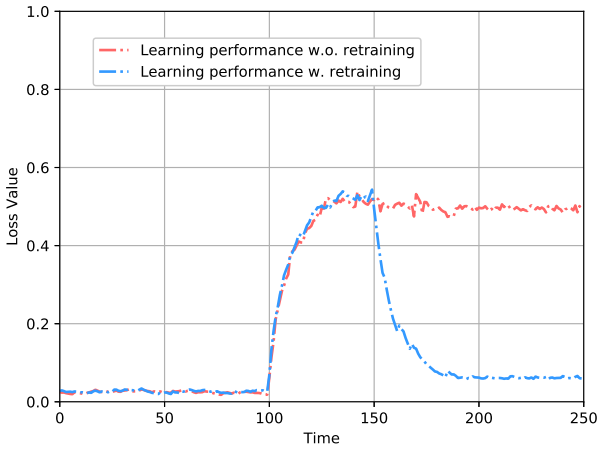


Figure 4.2: Evolution of loss over time-varying demands.

A critical issue in conventional learning approaches, including AC, is that the performance of a learning model largely depends on the adopted training or observed data sets. To illustrate the dynamic environment and its impacts, we consider two types of environmental changes. The first is "foreseen variations". A typical example is a time-varying channel with certain time correlation and statistical characteristics. In this case, a general machine learning algorithm can capture the regular patterns to resolve the mapping from the environment to the desired decision variables. The second is "unforeseen variations", which is much more challenging to address. These changes are usually unexpected and inclined to break the statistical distribution of the original environment. The practical LEO-5G systems are highly complex and dynamic, such as fast and dramatic variations in channel states, user demands, user arrival/departure, and network topologies. This typically causes that the new inputs are no longer relevant to the statistical properties of the historical data [154]. As a consequence, the scheduling decisions made from the previous learning model can become invalid and the model may need to be re-trained to adapt to the new environment.

To illustrate this impact, we use Fig. 4.2, as an example, to depict a typical evolution of AC's loss value over time-varying demands. From 0 to 100 time slots, the demand is time-varying but follows historical statistical properties, e.g., fluctuating within a certain range or following a certain distribution, leading to a well-adapted AC with low and stable loss values. When a surged demand is generated at the 100-th time slot, the new input deviates from the statistics. The AC model becomes inapplicable to the new environment, evidenced by the rapidly deteriorating loss values. When the agent in AC consumes a considerable amount of time in new data collection and re-training, the performance can return to the previous level.

To address this issue of "unforeseen change", meta-critic-based approaches become an emerging technique that takes advantage of a variety of previously observed tasks to infer the meta-knowledge,

[1] In this chapter, a learning step corresponds to a time slot.

such that a new learning task can be quickly trained with few observations [135]. Meta-critic learning combines meta-learning with an AC framework to enhance the generalization ability. However, conventional meta-critic learning is not effective in dealing with the large discrete space in $\mathbf{P1}$. In addition, there is no uniform standard to parameterize the learning model and extract meta-knowledge in dynamic environments. Thus, we propose an EMCL algorithm to enable an efficient dynamic-adaptive solution.

4.4 The Proposed EMCL Algorithm

In this section, we elaborate the proposed EMCL algorithm, firstly starting from outlining the EMCL framework, then detailing the tailored design.

4.4.1 EMCL Framework

MDP Reformulation

First, we reformulate the original problem $\mathbf{P1}$ as an MDP by defining action, state and reward.

- As the actor is to select a group from set $\mathcal{G}$ at each time slot t, the action is defined as an assigned link group,

$$a_t = g \in \mathcal{G}. \tag{4.21}$$

- The state consists of the channel coefficients $h_{k,n,t}$, modeled as FSMC with the transition probability defined in (4.6), and the delivered data for user k up to time slot t, where $b_{k,t} = b_{k,t-1} + R_{k,a_t,t}$.

$$s_t = \{h_{1,1,t}, ..., h_{K,N,t}, b_{1,t}, ..., b_{K,t}\}. \tag{4.22}$$

  All possible states are included in the state space $\mathcal{S}$. The next state only depends on the current state and action but is irrelevant to the past, which means the state transition from s_t to s_{s+1} follows the Markov property [57].

- The reward is closely related to the objective of $\mathbf{P1}$. We define the reward as (4.23).

$$r_t = \sum_{k=0}^{K} \eta_k (\Delta_{k,t-1}^2 - \Delta_{k,t}^2), \tag{4.23}$$

  where $\Delta_{k,t} = \begin{cases} \sum_{k=1}^{K} \mathbb{1}\left(b_{k,t} - D_k'\right) - K, & k = 0, \\ b_{k,t} - D_k, & k \neq 0. \end{cases}$ Then, the accumulated reward at step t is given by $\bar{r}_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$, where $\gamma \in [0, 1]$ is a discounted factor.

Under the designed MDP, we verify the consistency between the goals of the RL algorithm and the original optimization problem such that the policy provided by the learning agent can minimize the objective in $\mathbf{P1}$.

Lemma 4. *When $\gamma = 1$, the objective of the learning agent is equivalent to that of the optimization problem $\mathbf{P1}$.*

Proof. See Appendix 6.4 □

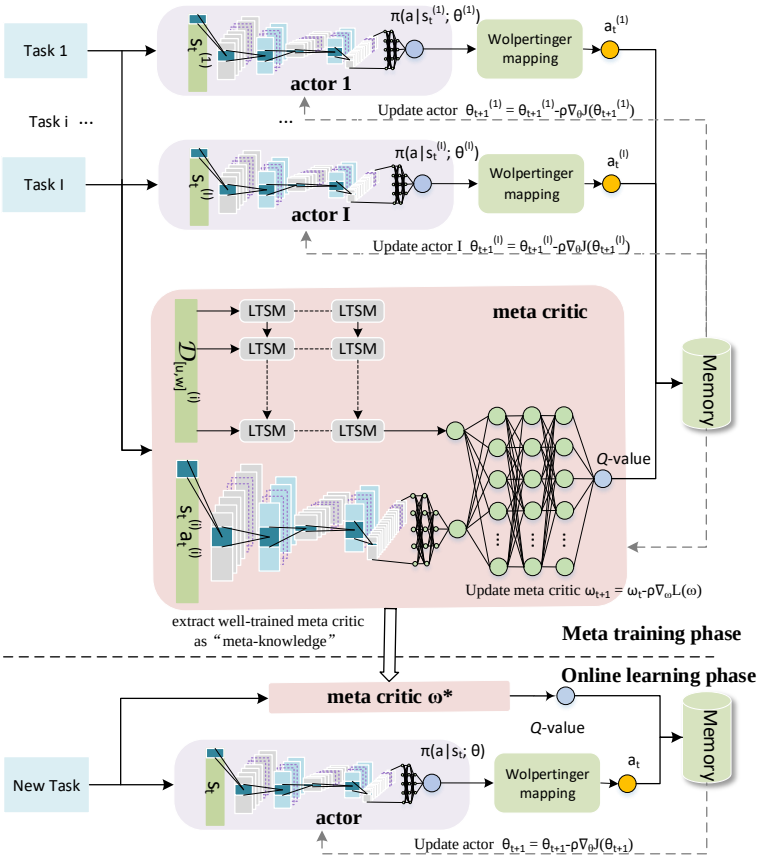


Figure 4.3: The proposed EMCL framework.

Meta Critic and Task-Specific Actor

As shown in Fig. 4.3, we design a hierarchical structure in EMCL containing a meta critic[2] and multiple actors. Meta-learning uses data from previously observed multiple tasks, $\mathcal{J}^{(1)}, ..., \mathcal{J}^{(I)}$, to infer a "meta-knowledge" with good generalization ability and accelerate the training for a new task. In the proposed EMCL, the "meta-knowledge" is the meta critic which can evaluate the task with a Q-value, like the role of the critic in traditional AC, and possesses a strong generalization ability to guide any task-specific actor to provide a policy.

At time step t, $s_t^{(i)}$, $a_t^{(i)}$, and $r_t^{(i)}$ represent the state, action, and reward for task i, respectively. An episode $\mathcal{D}^{(i)} = \{s_1^{(i)}, a_1^{(i)}, r_1^{(i)}..., s_T^{(i)}, a_T^{(i)}, r_T^{(i)}\}$ can be sampled from the first step to the terminal step T. We denote $\mathcal{D}_{[u,w]}^{(i)}$ as a segment of $\mathcal{D}^{(i)}$ from step u to w, i.e., $\mathcal{D}_{[u,w]}^{(i)} = \{s_u^{(i)}, a_u^{(i)}, r_u^{(i)}, ..., s_w^{(i)}, a_w^{(i)}, r_w^{(i)}\}$. Since the explicit meta critic and actors are difficult to obtain, we adopt the function approximation method. The meta critic is parameterized as a neural network (NN) with the weights $\boldsymbol{\omega}$, i.e., $Q(s_t^{(i)}, a_t^{(i)}, \mathcal{D}_{[t-\bar{t},t-1]}^{(i)}; \boldsymbol{\omega})$. We note that, in addition to $s_t^{(i)}$ and $a_t^{(i)}$, the input includes the most recent $\bar{t}$ samples $\mathcal{D}_{[t-\bar{t},t-1]}^{(i)}$. Each task-specific actor is modeled as an NN $\pi(a|s_t^{(i)}; \boldsymbol{\theta}^{(i)})$ with the weights $\boldsymbol{\theta}^{(i)}$.

To optimize the weights, we minimize the loss functions by gradient descend. The loss function of

[2]In this chapter, "meta-critic learning" refers to an algorithm that combines AC and meta-learning while "meta critic" refers to the critic in the framework.

the meta critic $L(\boldsymbol{\omega})$ is defined as the average temporal difference (TD) error over all tasks:

$$L(\boldsymbol{\omega}) = \frac{1}{I}\sum_{i=1}^{I}\mathbb{E}_{\pi(\boldsymbol{\theta}^{(i)})}\left[(Q(s_{t+1}^{(i)}, a_{t+1}^{(i)}, \mathcal{D}_{[t-\bar{t}+1,t]}^{(i)}; \boldsymbol{\omega}) - r_t - \gamma Q(s_t^{(i)}, a_t^{(i)}, \mathcal{D}_{[t-\bar{t},t-1]}^{(i)}; \boldsymbol{\omega})\right]^2, \quad (4.24)$$

where the TD error reflects the similarity between the estimated Q-value and actual Q-value. For the task-specific actor, the loss function $J(\boldsymbol{\theta}^{(i)})$ is the negative Q-value:

$$J(\boldsymbol{\theta}^{(i)}) = \mathbb{E}_{\pi(\boldsymbol{\theta}^{(i)})}\left[-Q(s_t^{(i)}, a_t^{(i)}, \mathcal{D}_{[t-\bar{t},t-1]}^{(i)}; \boldsymbol{\omega})\right], \quad (4.25)$$

such that minimizing $J(\boldsymbol{\theta}^{(i)})$ is equivalent to maximizing the expected accumulated reward. The update rules are given by:

$$\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t - \rho\nabla_{\boldsymbol{\omega}}L(\boldsymbol{\omega}), \quad (4.26)$$

$$\boldsymbol{\theta}_{t+1}^{(i)} = \boldsymbol{\theta}_t^{(i)} - \rho\nabla_{\boldsymbol{\theta}^{(i)}}J(\boldsymbol{\theta}^{(i)}). \quad (4.27)$$

Based on the fundamental results of the policy gradient theorem [57], the gradients of $L(\boldsymbol{\omega})$ and $J(\boldsymbol{\theta}^{(i)})$ are:

$$\nabla_{\boldsymbol{\omega}}L(\boldsymbol{\omega}) = \frac{1}{I}\sum_{i=1}^{I}\left[2L(\boldsymbol{\omega})\nabla_{\boldsymbol{\omega}}(Q(s_{t+1}^{(i)}, a_{t+1}^{(i)}, \mathcal{D}_{[t-\bar{t}+1,t]}^{(i)}; \boldsymbol{\omega}) - Q(s_t^{(i)}, a_t^{(i)}, \mathcal{D}_{[t-\bar{t},t-1]}^{(i)}; \boldsymbol{\omega}))\right], \quad (4.28)$$

$$\nabla_{\boldsymbol{\theta}^{(i)}}J(\boldsymbol{\theta}^{(i)}) = -Q(s_t^{(i)}, a_t^{(i)}, \mathcal{D}; \boldsymbol{\omega})\nabla_{\boldsymbol{\theta}^{(i)}}\log\pi(a|s_t^{(i)}; \boldsymbol{\theta}^{(i)}). \quad (4.29)$$

Algorithm Summary

We summarize the proposed EMCL in Alg. 6, which includes two phases: the meta training phase and the online learning phase. For the former, the meta critic is trained over different learning tasks. At each learning episode, we sample I learning tasks. We obtain the approximated Q-value (in line 6) and stochastic policy (in line 7) by the approximation functions. The final actions are determined by the Wolpertinger approach in line 8, which will be elaborated in the following subsection. In line 9, the memory is used to store the experienced learning tuples $\{s_t^{(i)}, s_{t+1}^{(i)}, a_t^{(i)}, r_t^{(i)}\}$. At each step, we extract a batch of tuples from the memory as the training data for updating $\boldsymbol{\omega}$ and $\boldsymbol{\theta}^{(i)}$ by (4.26) and (4.27) in line 10 and 12, respectively. In the online learning phase, given a new task, the well-trained meta critic $\boldsymbol{\omega}^*$ can be directly used to estimate the Q-value and only the actor needs to be re-trained. We note that the adaptation ability of the meta-learning algorithm depends on the completeness of the tasks provided in the meta-training phase. In general, it is not practical to collect all the possible environments. As an alternative, the selected tasks in the meta-training phase should keep the diversity and representativeness to achieve higher sampling efficiency.

4.4.2 Tailored Designs in EMCL

Parameterization with Hybrid Neural Networks

There is no uniform standard for parameterization in conventional meta-critic learning. Considering dynamic environments, the distribution of the new input data and the previous observations may deviate. Towards fast adaptation to the dynamic environment, the critic should be able to identify different tasks, where the information for task identification can be refined from the experienced data, which usually

Algorithm 6 EMCL

Inputs: Multiple task samples; initial $\boldsymbol{\omega}_0$.

for each learning episode **do**
  Sample I tasks and initialize $\boldsymbol{\omega}_0, \boldsymbol{\theta}_0^{(1)}, ..., \boldsymbol{\theta}_0^{(I)}$.
  for each learning step t **do**
    for each task i **do**
      Obtain Q-value by the meta critic in (4.32).
      Obtain stochastic policy by the actor in (4.34).
      Take actions $a_t^{(i)}$ by the Wolpertinger approach.
      Store tuples $\{s_t^{(i)}, s_{t+1}^{(i)}, a_t^{(i)}, r_t^{(i)}\}$ in the memory.
      Take a batch of data and update $\boldsymbol{\theta}^{(i)}$ by (4.27).
    end for
    Update $\boldsymbol{\omega}$ by (4.26).
  end for
end for
output: The well-trained meta critic $\boldsymbol{\omega}^*$.

Outputs:

input: A new task; initial $\boldsymbol{\theta}_0$; well-trained meta critic $\boldsymbol{\omega}^*$.
for each learning episode **do**
  for each learning step t **do**
    Obtain Q-value by the meta critic in (4.32).
    Obtain stochastic policy by the actor in (4.34).
    Take an action a_t by the Wolpertinger approach.
    Store tuples $\{s_t, s_{t+1}, a_t, r_t\}$ in the memory.
    Take a batch of data and update $\boldsymbol{\theta}$ by (4.27).
  end for
end for
Outputs: The optimal actor $\boldsymbol{\theta}^*$.

forms time-related series [137]. The widely used DNN might have limitations in efficiency and in mining features from time-series data due to massive number of weights and feed-forward structure. In the proposed EMCL, we design tailored neural networks to enable the meta critic and the actors to fit the complex nonlinear relationships and extract the meta-knowledge from historical data.

As shown in Fig. 4.3, for the meta critic, a hybrid neural network (HNN) combing convolutional neural network (CNN), long-short term memory (LSTM), and DNN is applied to learn the features from the current state-action pairs and historical trajectories [79]. Thereinto, CNN is computation-efficient via adopting the parameter sharing and pooling operations, and is effective to extract spatial features from the input data. LSTM, as a type of recurrent neural network, has advantages in extracting features from time-related sequential data. Thus, in the designed meta critic, the CNN is used to extract a feature from the current action-state pair $s_t^{(i)}, a_t^{(i)}$ to evaluate the decisions made by the actor. The LSTM is adopted for task-identification based on the time-series data $\mathcal{D}_{[t-\bar{t},t-1]}^{(i)}$. We denote $f_{cnn}(\mathbf{x};\mathbf{w})$, $f_{lstm}(\mathbf{x};\mathbf{w})$ and $f_{dnn}(\mathbf{x};\mathbf{w})$ as the outputs of CNN, LSTM, and DNN, respectively, which are the functions of input $\mathbf{x}$ and weight $\mathbf{w}$. The features output from CNN and LSTM are:

$$\xi_1 = f_{cnn}(s_t^{(i)}, a_t^{(i)}; \boldsymbol{\omega}_{cnn}), \tag{4.30}$$

$$\xi_2 = f_{lstm}(\mathcal{D}^{(i)}_{[t-\bar{t},t-1]}; \boldsymbol{\omega}_{lstm}). \tag{4.31}$$

Then, we take the features as input and pass them through a fully-connected DNN to obtain the approximated Q-value:

$$Q^{\pi}(s_t^{(i)}, a_t^{(i)}, \mathcal{D}^{(i)}_{[t-\bar{t},t-1]}; \boldsymbol{\omega}) = f_{dnn}(\xi_1, \xi_2; \boldsymbol{\omega}_{dnn}). \tag{4.32}$$

For the task-specific actors, we adopt CNN as the approximator which takes the current state as the input and outputs the mean μ and variance ϑ^2 of the stochastic policy. We assume the stochastic policy follows Gaussian distribution $N(\mu, \vartheta^2)$, such that

$$[\mu, \vartheta^2] = f_{cnn}(s_t^{(i)}; \boldsymbol{\theta}^{(i)}), \tag{4.33}$$

$$\pi(a|s_t^{(i)}; \boldsymbol{\theta}^{(i)}) = N(\mu, \vartheta^2). \tag{4.34}$$

Action Mapping with the Wolpertinger Policy

The decision variables in $\mathbf{P1}$ are discrete such that we need to map the action from the stochastic policy to a discrete action space. However, the previous action mapping policies in meta-critic learning are not efficient since the action space is large for $\mathbf{P1}$. Thus, in EMCL, the Wolpertinger policy is adopted for faster convergence [155].

Following the stochastic policy π, the actor first produces an action $\hat{a}$ with continuous value, i.e.,

$$f_{\pi} : \mathcal{S} \rightarrow \hat{\mathcal{A}}, \;\; f_{\pi}(s) = \hat{a}, \tag{4.35}$$

where f_{π} is a mapping from the state space $\mathcal{S}$ to a continuous action space $\hat{\mathcal{A}}$ under the policy π. As the real action space $\mathcal{G}$ is discrete in $\mathbf{P1}$, the following two conventional approaches can be used for discretization [57]:

- Simple approach: $a_s^* = \underset{a \in \mathcal{G}}{\operatorname{argmin}} |a - \hat{a}|^2$.
- Greedy approach: $a_g^* = \underset{a \in \mathcal{G}}{\operatorname{argmax}} Q(s, a)$.

The simple approach is to select the closest integer value to $\hat{a}$. This approach may result in a high probability of deviating from the optimum, especially at the beginning of learning, and further lead to slow convergence [57]. The greedy approach optimizes Q-value at each step but the complexity is proportional to the exponentially increasing space $\mathcal{G}$ [57]. To achieve a trade-off between the complexity and learning performance, the Wolpertinger mapping approach is considered.

- Wolpertinger approach: $a_w^* = \underset{a \in \mathcal{M}^*}{\operatorname{argmax}} Q(s, a)$,

where $\mathcal{M}^*$ is a subset of $\mathcal{G}$ and contains M nearest neighbors of $\hat{a}$. In the Wolpertinger approach, the final action is determined by selecting the highest-scoring action from $\mathcal{M}^*$. The Wolpertinger mapping becomes the greedy approach and simple approach when $M = |\mathcal{G}|$ and $M = 1$, respectively, and the solution of simple approach a_s^* is included in $\mathcal{M}^*$.

Lemma 5. *We assume* $\mathcal{M}^* = \{a_1, ..., a_M\}$ *and*

$$\begin{cases} Q(s, a_m) \sim U(Q(s, a_s^*) - \kappa, Q(s, a_s^*) + \kappa), & m \neq m' \\ Q(s, a_m) = Q(s, a_s^*), & m = m', \end{cases} \tag{4.36}$$

where $U(a,b)$ refers to uniform distribution and κ is a constant, then

$$\mathbb{E}\left[Q(s,a_w^*)\right] = Q(s,a_s^*) + \kappa\left(1 - \frac{2(2^M-1)}{M\cdot 2^M}\right). \tag{4.37}$$

Proof. See Appendix 6.5 □

From Lemma 5, when $M > 1$, $\mathbb{E}\left[Q(s,a_w^*)\right] > Q(s,a_s^*)$, which means that the Wolpertinger approach finds the actions with higher Q-values than the simple approach at each learning step, and a larger M leads to a higher expected Q-value. In addition, the complexity of the Wolpertinger approach is lower than the greedy approach as the size of searching space decreases from $|\mathcal{G}|$ to $|\mathcal{M}^*|$. Thus, for the problems with huge discrete spaces, the Wolpertinger approach enables fast convergence to the maximum Q-value with a proper M.

4.4.3 Complexity Analysis for EMCL

For the meta critic, an HNN, composed of CNN, LSTM, and DNN, is employed to estimate the Q-value. We assume CNN includes V_1 convolutional layers. We denote $o_{c,v}$, $o_{k,v}$, $o_{f,v}$ are the number of convolutional kernels, the spatial size of the kernel, and the spatial size of the output feature map in the v-th layer, respectively. The stripe of kernel is 1, and the input size is $o_{c,0} = K(N+1)+1$. The time complexity of CNN is $\mathcal{O}\left(\sum_{v=1}^{V_1} o_{c,v-1}\varrho_v\right)$, where $\varrho_v = o_{k,v}^2 o_{c,v} o_{f,v}^2$ [156]. For the LSTM, we consider V_2 layers, and denote $o_{l,v}$ and $o_{e,v}$ are the input size and number of memory cells for layer v, respectively, where $o_{l,0} = m(K(N+1)+1)$. The time complexity is given by $\mathcal{O}\left(\sum_{v=1}^{V_2} o_{e,v}(4o_{l,v-1}+\varsigma_v)\right)$, where $\varsigma_v = 4o_{e,v} + o_{l,v} + 3$ [157]. For the fully-connected DNN, the time complexity is $\mathcal{O}\left(2o_{d,1} + \sum_{v=2}^{V_3} o_{d,v-1}o_{d,v}\right)$, where V_3 is the number of layers of DNN, $o_{d,v}$ is the input size for layer v [158]. For the actor, as the stochastic policy is approximated by a CNN, the time complexity is identical with that of CNN in the meta critic. Overall, the total time complexity of EMCL is calculated by $\mathcal{O}\left(TK(N+1)L_1 + L_2\right)$, where $L_1 = \varrho_1 + 4mo_{e,1}$ and $L_2 = \varrho_1 + o_{e,1}(4m+\varsigma_1) + 2o_{d,1} + \sum_{v=2}^{V_1} o_{c,v-1}\varrho_v + \sum_{v=2}^{V_2} o_{e,v}(4o_{l,v-1}+\varsigma_v) + \sum_{v=2}^{V_3} o_{d,v-1}o_{d,v}$. When the parameters of the learning model are determined, the complexity increases linearly with $\mathbf{P1}$'s input size, i.e., K and N.

4.5 Numerical Results

In the simulation, the parameter settings are similar as in [38, 159]. The adopted parameters for implementing EMCL are summarized in Table 4.1. We compare the performance of the proposed EMCL algorithm with the following five benchmark algorithms:

- OPT: optimal solution (B&B).
- ADMM-HEU: suboptimal solution (Alg. 1).
- GRD: a greedy suboptimal algorithm proposed in [160].
- AC-DDPG: a classic AC algorithm with deep deterministic policy gradient proposed in [58].
- AC-MAML: AC with model-agnostic meta-learning proposed in [136].

The first three provide benchmarks from an optimization perspective, while the last two compare with EMCL from a learning perspective. For the AC benchmarks, the actor and critic are parameterized by two DNNs with the complexity $\mathcal{O}(TK(N+1)L_3 + L_4)$, where L_3 and L_4 are constants, thus keeping the same magnitude with the proposed EMCL [158].

Table 4.1: Parameter settings

Total number of GDs in network	500-1000
Number of transmitters	1 LEO, 1 BS and 2 TSTs
Time limitation T	10 time slots
Duration of time slot Φ	0.1 s
Altitude of LEO	780 km
Transmit power of LEO	100 W
Transmit power of BS	40 W
Transmit power of TST	2 W
Bandwidth for C-band	20 MHz
Bandwidth for Ka-band	400 MHz
Carrier frequency of C-Band	4 GHz
Carrier frequency of Ka-Band	30 GHz
Noise power spectral density	-174 dBm/Hz
Weights values	$0 \leq \eta_0 \leq 10$ $\eta_1 + ... + \eta_K = 1$
Parameterized meta critic	HNN
Parameterized actor	CNN
Distribution of stochastic policy	Gaussian
Learning rate	0.001
Batch size	128
Memory size	10,000
Discount factor	0.9
Size of search space in Wolpertinger policy	10
Environment update interval	200 time slots
Software platform	Python 3.6 with TensorFlow 1.12.0

4.5.1 Capability in Dealing with Dynamic Environments

To verify the capability of the proposed EMCL in dealing with dynamic environments, Fig. 4.4-4.6 compare EMCL with AC-MAML and AC-DDPG in three dynamic scenarios. In Fig. 4.4, we consider the first scenario with users' irregular access and departure, which can be disruptive to the typical statistical properties. For instance, the adopted simulator generates user arrivals by following the Poisson distribution as the normal case, while it also periodically generates abnormal events (every 200 slots) with randomly large/small number of arrived users. We update the environment information every 200 time slots. From Fig. 4.4, both EMCL and AC-MAML are able to converge before each update, but EMCL saves 28.66% recovery time and reduces 45.42% objective value than AC-MAML, where we define a recovery time counting from the moment of dramatic performance degradation until the performance recoveries to the normal level. For AC-DDPG, the convergence performance is inferior to the others, and fails to converge when updating occurs at the 200-th and 600-th time slot. We remark that the case of user departure is easier to be adapted. Fewer users in the system reduce the problem dimension, and thus simplify the learning task, leading to a halved recovery time and flat curves between the 200-th and

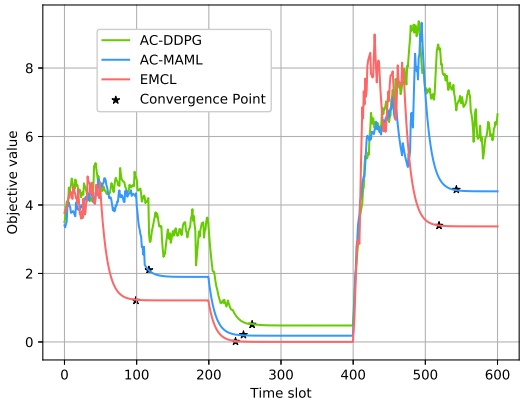


Figure 4.4: Performance in adapting to dynamic scenario 1: user entry and leave.

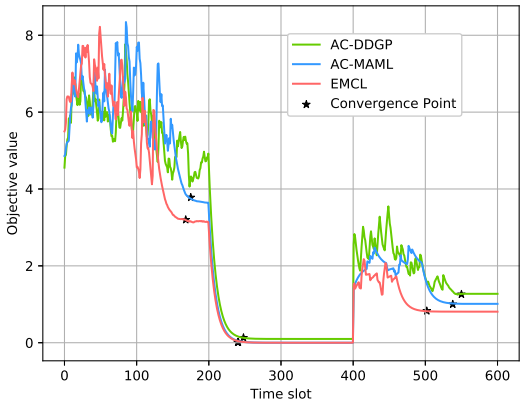


Figure 4.5: Performance in adapting to dynamic scenario 2: bursty demands.

the 400-th slots in three algorithms. In contrast, it is more difficult to deal with the case of user arrival, referring to the large fluctuation after the 400-th slot, mainly due to lacking relevant new-user data and the exponentially increasing dimension. We can observe that EMCL has strong capabilities in adapting to this difficult case and achieves more performance gains than the other two algorithms.

In Fig. 4.5, we evaluate the algorithms' capabilities in adapting to unforeseen dynamic demands. The simulator generates the volume of users' requested data by following the uniform distribution as the normal case. Then, the distribution changes due to the abnormal bursty demands, e.g., switching from a low-speed voice call to a data-hungry HD video service, or vice versa. In Fig. 4.6, we consider the channel states can undergo non-ideal large fluctuations, e.g., sharply deteriorated channel conditions due to the large obstacles or the rain/cloud blocks appearing in the transmission path. Similarly to Fig. 4.4, we collect the updated environment information every 200 time slots. From Fig. 4.5 and Fig. 4.6, AC-DDPG has poor convergence performance, since AC-DDPG needs to re-train the learning model from scratch when the environment changes, leading to a slow adaptation, while EMCL and AC-MAML extract the meta-knowledge from multiple tasks to accelerate the convergence speed. EMCL re-fits the

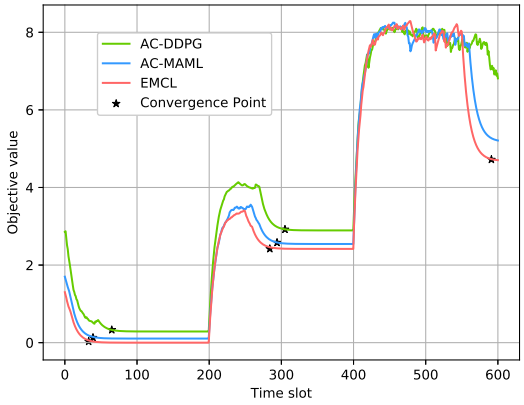


Figure 4.6: Performance in adapting to dynamic scenario 3: unforeseen channel variations.

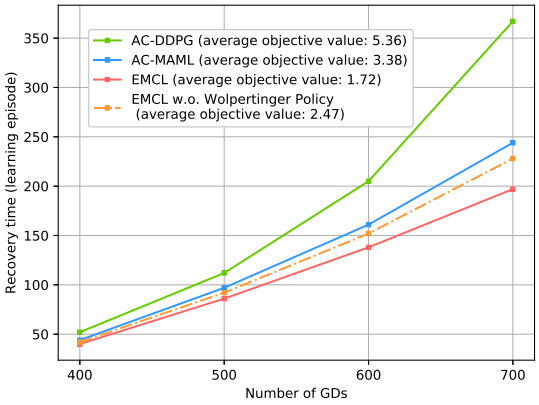


Figure 4.7: Recovery time vs. number of users

learning model in a timely manner than AC-MAML. This is because EMCL uses meta critic to guide the actor to adjust scheduling schemes more effectively in a dynamic environment, and the designed HNN and Wolpertinger mapping approach can improve the learning accuracy and efficiency in large discrete action spaces.

Fig. 4.7 further summarizes the average recovery time with respect to the numbers of GDs based on Fig. 4.4. In general, the more GDs in the system, the longer the recovery time required to adapt to the new environment. On average, EMCL saves 29.83% and 13.49% recovery time compared to AC-DDPG and AC-MAML, respectively, and the time-saving gain of EMCL becomes even larger when more GDs in the system. In addition, we compare the EMCL algorithm with and without the Wolpertinger policy to demonstrate the effectiveness of the adopted action mapping method. The recovery time of the latter is 10.11% increased than the former but less than AC-DDPG and AC-MAML. At the convergence, EMCL can decrease the average objective value by 30.36% compared to EMCL without the Wolpertinger policy.

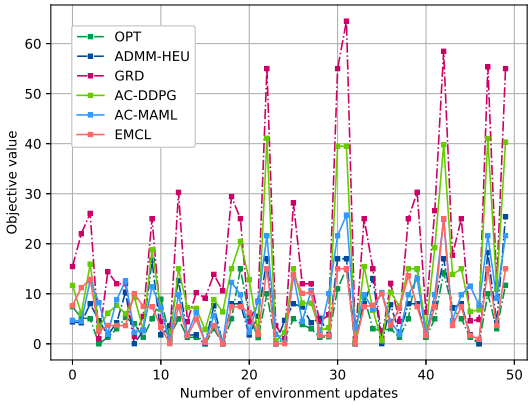


Figure 4.8: Objective value vs. number of updates

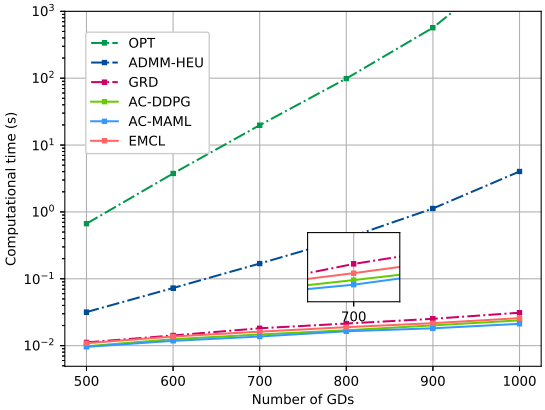


Figure 4.9: Computational time vs. number of users

4.5.2 Trade-offs between Computational Time and Optimality

To demonstrate EMCL's trade-off performance between approaching the optimum (Fig. 4.8) and computational time (Fig. 4.9), we compare EMCL with five benchmarks. In Fig. 4.8, we observe 50 environmental information updates and record the average objective values within each update cycle. For AC-MAML and AC-DDPG, the average gaps to the optimum are 45.26% and 57.23%, respectively, while for EMCL, the average gap drops to 27.58%. The performance of EMCL is slightly better than ADMM-HEU, around 3.54%. For GRD, the average gap to the optimum is 74.15%, which is inferior to the AC-based algorithms.

Fig. 4.9 compares the computational time with respect to the number of GDs. OPT is the most time-consuming algorithm, as expected. Compared to OPT, ADMM-HEU saves 98.14% computational time by decomposing variables into multiple blocks and performing parallel computations. The computational time in ECML, two AC algorithms, and GRD keep at the millisecond level, but the proposed EMCL achieves smaller gaps to the optimum, hence concluding the better trade-off performance of EMCL than other benchmarks.

4.6 Conclusion

We have investigated a resource scheduling problem in dynamic LEO-terrestrial communication systems to address the mismatch issue in a practical over-loaded scenario. Due to the high computational time of the optimal algorithm and the proposed ADMM-HEU algorithm, we solve the problem from the perspective of DRL to obtain online solutions. To enable the learning model to fast adapt to dynamic environments, we develop an EMCL algorithm that is able to handle the environmental changes in wireless networks, such as bursty demands, users' entry/leave, and abrupt channel change. Numerical results show that, when encountering an environmental variation, EMCL consumes less recovery time to re-fit the learning model, compared to AC-DDPG and AC-MAML. Furthermore, EMCL achieves a good trade-off between solutions quality and computation efficiency compared to offline and AC-based benchmarks. An extension of the current work is to combine other techniques, e.g., continuous learning and behavior regularization, to further improve the sample efficiency and model adaptability.

Chapter 5

Distributed Learning: Energy-Efficient Wireless Federated Edge Learning

5.1 Introduction

Over the past few years, the wireless community has witnessed a paradigm shift from model-driven approaches to learning-based methods in wireless system design and analysis. Machine learning is becoming an important tool for next-generation wireless communication networks. Typical machine learning schemes used in wireless edge systems are based on centralized training [161, 162]. That is, the edge devices collect data and transmit to a centralized learning server for computing and processing. However, most of the collected data is privacy-sensitive and such data exchange may have potential risks of information leakage. To this end, federated learning (FL), as a distributed learning paradigm, prominently protects the users' privacy since the edge devices directly upload their local learning models rather than their original data [163, 164]. In addition, FL allows training the learning model in the devices locally in parallel thus saving the training time compared to centralized learning. Therefore, FL has attracted considerable attention in many wireless applications, such as wireless edge computing.

In an FL-based edge computing system, several issues need to be considered. Firstly, edge devices may undertake computing tasks and transmission tasks, and their energy supply and battery storage are limited. Thus, an energy-efficient FL scheme is of importance [50]. Secondly, deploying complex learning models on the edge may not be the best choice mainly due to the resulted massive parameters in local models, large data volume to be uploaded, and typical high consumption in energy and computing resources. Hence, a simple learning model with sparse weight parameters is essential for energy-efficient FL in the edge. Thirdly, when a learning model is sparsified, conventional FL algorithms might not converge. For example, sparse neural networks (SNNs) and binary neural networks (BNNs) are two possible sparsification schemes. They may introduce non-smooth regularization items (for SNN) and the constraints on weights (for BNN). Both cases are challenging to deal with for conventional FL algorithms [66]. To this end, developing a proper learning algorithm for training the sparsified learning models is necessary.

5.1.1 Related Work

In this subsection, we review the state-of-the-art of FL algorithms and their applications over wireless networks.

Federated Learning Algorithms

FL is a type of distributed machine learning structure, which enables terminal devices to collaboratively learn a global learning model while keeping all the training data at the devices. The training process of FL is iterative. At a learning cycle, each edge device trains a local model based on the collected data and the up-to-date global model. All the edge devices then upload their individual local models to a central server. After the aggregated process, the server broadcasts an updated global model to the edge devices for preparing the next learning cycle. Federated stochastic gradient descent (FedSGD) and federated averaging (FedAvg) are the most well-known FL algorithms, which minimize the local loss functions in each device and update the learning models by stochastic gradient descent (SGD) [165]. Compared to FedSGD, FedAvg increases the number of local training epochs performed on the edge device to reduce the frequency of communication between server and edge devices [166]. To further improve the learning efficiency, the authors in [167] proposed an FL method with periodic averaging and quantization (FedPAQ), where the local models are periodically averaged at the server and only partial devices participate in the training process at each round. In addition, the local devices quantize their updated models before uploading to the server. In [168], the authors proposed a proximal FL (FedProx) to tackle the heterogeneity in federated networks, e.g., non-identically distributed (non-i.i.d.) data across the networks and significant variability on each device, by adding a proximal term into the local loss function.

The previous FL algorithms in [165, 166, 167, 168] have been proven their performance in normal cases. However, it is studied to a limited extent for dealing with the non-smooth cases, e.g., L1-norm regularization in SNNs, and the weight constrained cases, e.g., binary constraints in BNNs. In [66], the authors proposed a new optimizer (ProxSGD) to train the sparse neural networks with non-smooth loss functions for centralized learning, but the analysis of convergence and learning accuracy is absent for the distributed learning. This prompts us to develop a suited FL algorithm for efficient local training with guaranteed convergence.

Applications in Wireless Networks

A number of recent works have applied FL to wireless communication networks. In [169], the authors provided a general idea of FL applications in 5G wireless Internet-of-Things (IoT) networks and mobile edge computing. In [170], the authors developed an edge learning framework to analyze the convergence performance of the FL algorithm while considering the features of wireless networks, e.g., user scheduling policy, channel fading and inter-cell interference. In [171], a joint user selection and wireless resource allocation problem was investigated to minimize the loss function with limited transmit power. To minimize FL training time in multiple-input multiple-output systems, in [172], an optimization problem was formulated by allocating transmit power, users' processing frequency and local accuracy. Considering the scarcity of energy in the local devices of wireless federated edge learning networks, the authors in [50], [49] and [48] formulated energy minimization problems, including local computing energy and transmission energy, via optimizing wireless and computation resources. Furthermore, the authors in [48] proposed an improved FL algorithm based on FedProx to handle the non-i.i.d. data and heterogeneity of the devices. In [173], an optimization problem was formulated to minimize to capture the trade-off between FL convergence time and energy consumption of UEs with heterogeneous computing and power resources.

In [171, 172, 49, 50, 48, 173], the energy minimization of FL is achieved in resource optimization by assuming conventional learning models at the device side, e.g., adopting fully-connected deep neural networks (DNNs). This might not be energy-efficient for FL-based edge computing because DNNs need to consume computational energy on mathematical operations between layers and send a heavy-weight

local model to the central server by consuming transmission energy. A lightweight learning model can lighten the computation burden or decrease the data volume to be uploaded [174]. This motivates us to further save energy by replacing dense learning models with SNNs or BNNs.

5.1.2 Contributions

The main contributions and novelty of this work are summarized as follows:

- A majority of the previous FL schemes reduce energy consumption by relying on resource allocation. In this work, we design a joint sparsification and resource optimization scheme (JSRO) for energy-efficient FL in wireless edge networks, such that energy conservation can benefit from two aspects.

- Unlike previous works, we exploit the energy-saving gain in FL from the sparsification of learning models, i.e., replace DNN with SNN or BNN in local training. To handle the non-smoothness and weight constraints in training SNN and BNN, we propose an enhanced SGD algorithm tailored for non-smoothness or constrained learning models (NSC-SGD) with guaranteed convergence.

- Based on model sparsification, we jointly minimize computing, uploading, and broadcasting energy by optimizing resource allocation and learning parameters. All three types of energy consumption are non-trivial for wireless edge networks. Besides, we characterize the interplay between learning performance and energy optimization.

- Numerical results demonstrate that the proposed JSRO can reduce more than 50% energy compared to benchmark FL schemes. From the result analysis, we observe that SNN is effective in reducing uploading energy, whereas BNN can save more energy in local training and data uploading than SNN and DNN. In addition, the proposed NSC-SGD outperforms previous FL algorithms in terms of learning accuracy and convergence.

The remainder of this chapter is organized as follows. In Section 5.2, we provide the preliminary of FL framework and its wireless application. Section 5.3 introduces the energy conservation by sparsification. Section 5.4 formulates an energy minimization problem. In Section 5.5 we elaborate the proposed joint energy-efficient FL scheme, i.e., JSRO. Numerical results are presented and analyzed in 5.6. Finally, we draw the conclusions in 5.7.

5.2 Preliminary

5.2.1 FL Framework in Wireless Edge

We consider a wireless federated edge learning scenario in Fig. 5.1. Thereinto, a base station or an access point (AP) with computing and signaling capabilities can act as the centralized server [169]. We assume that all the edge devices with a certain computing ability to train individual local models based on the collected data. The communication between the edge devices and server is based on wireless links.

FL is designed to allow multiple devices to cooperatively execute a learning task by only uploading the local learning model to a centralized server. The objective of FL is given by:

$$\min_{\mathbf{w}\in\mathcal{W}} F(\mathbf{w}) = \sum_{k=1}^{K} \frac{D_k}{D} F_k(\mathbf{w}), \tag{5.1}$$

where $\mathbf{w}$ is the parameter vector of the learning model, $\mathcal{W}$ is the space of the parameters, K is the number of the edge devices participating in the training, D_k is the size of the training data at edge device k, and D is the total size of the training data. The local objective function $F_k(\mathbf{w})$ consists of a loss function $f_k(\mathbf{w})$ and a regularization item $r_k(\mathbf{w})$, which can be expressed as:

$$F_k(\mathbf{w}) = \underbrace{\frac{1}{D_k}\sum_{l=1}^{D_k}\mathcal{L}_{k,l}(\mathbf{w})}_{\triangleq f_k(\mathbf{w})} + r_k(\mathbf{w}), \tag{5.2}$$

where $\mathcal{L}_{k,l}(\mathbf{w})$ describes the prediction error of the l-th single sample at edge device k, which can be represented by mean square error (MSE) or cross entropy. The regularization item $r_k(\mathbf{w})$ is a penalty to prevent overfitting or confining the weight values.

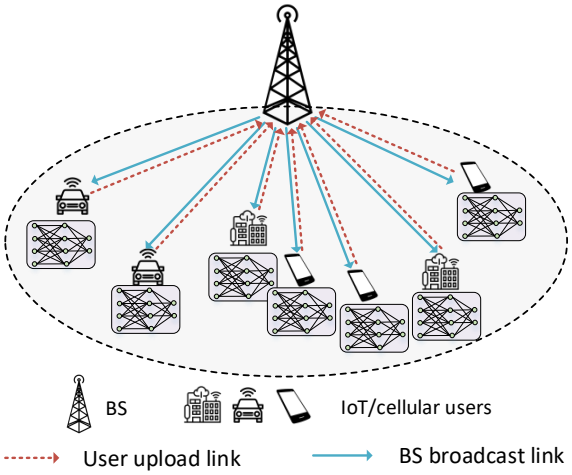


Figure 5.1: Illustration of wireless federated edge learning.

The process of the FL is conducted in an iterative manner as the following steps [165, 166, 167, 168]:

- Step 1: The centralized server randomly selects edge devices to participate in the training process and broadcasts a global FL model.
- Step 2: Each selected edge device downloads the global FL model for training a local FL model. The well-trained local models will be uploaded to the centralized server.
- Step 3: The centralized server aggregates the local information to update the global learning model.
- Step 4: Repeating steps 1-3 (i.e., a learning cycle) until the termination condition is met.

In FL, problem (5.1) is decomposed to K independent problems that are solved locally at each edge device. Thus, each device trains the local learning model by minimizing the local loss function:

$$\min_{\mathbf{w}\in\mathcal{W}} : F_k(\mathbf{w}). \tag{5.3}$$

The local problems can be solved by stochastic gradient descent (SGD) algorithm, which updates the local parameters iteratively and consumes local iterations. We denote i as the index of global iterations

(or learning cycles), and j as the index of local iterations. At each local iteration, taken the received global model $\mathbf{w}^i$ as the initial point, the update rule is written by:

$$\mathbf{w}_k^{i,j+1} = \mathbf{w}_k^{i,j} - \varepsilon_{loc} \nabla F_k(\mathbf{w}_k^{i,j}). \tag{5.4}$$

where ε_{loc} is the local update step and $\mathbf{w}_k^{i,j}$ is the local FL model for user k at the j-th local iteration and the i-th global iteration. We assume that the SGD terminates at the maximum number of local iterations, denoted by J_k. The server updates the global model by:

$$\mathbf{w}^{i+1} = \mathbf{w}^i + \epsilon \left(\sum_{k=1}^{K} \mathbf{w}_k^i - \mathbf{w}^i \right), \tag{5.5}$$

where ϵ is the learning rate and $\mathbf{w}_k^i = \mathbf{w}_k^{i,J_k}$. According to [175], for SGD algorithm, the gap between the optimum and the intermediate result at local iteration j is expressed as:

$$\Gamma(j, d) = \alpha \left(\frac{1}{\sqrt{jd}} + \frac{1}{j} \right), \tag{5.6}$$

where d is the size of training data and α is a positive constant. We remark that a small gap $\Gamma(j, d)$ means high local learning accuracy. Since the local model needs to be uploaded to the server for global aggregation, the local accuracy directly affects the global accuracy of FL. To avoid the performance degradation caused by the accumulation error of local computing, the gap $\Gamma(j, d)$ should be less than a threshold γ_{th}.

5.2.2 Energy Models

The limited energy supply and computing resources at the edge devices are critical issues for FL. In the following, we introduce the adopted energy models.

Communication Energy

At each global iteration (or learning cycle), the channel gain for edge device k is h_k, $\forall k \in \{1, ..., K\}$. Under the assumption of the block fading channel, the channel states keep constant within a learning cycle. Considering general scenarios, the server for edge learning could be an AP with limited energy, the energy consumed on both edge devices' upload and server's broadcast needs to be conserved. To facilitate the data aggregation in the server, the learning models, e.g., DNN, for all the edge devices are identical. The volume of transmitted data is equivalent to the volume of data required to build the ML model. We denote A_k as (in bits) the upload data volume of edge device k and A_0 (in bits) as the broadcast data volume of the server.

In the upload phase, the transmission time of edge device k within a learning cycle is:

$$T_k^u = \frac{A_k}{R_k} = \frac{A_k}{B_k \log(1 + \frac{p_k h_k}{\sigma^2})}, \tag{5.7}$$

where R_k, B_k, p_k are the uplink transmission rate, the uplink bandwidth, and the transmit power of edge device k, respectively, and σ^2 is the noise power. The total energy consumption E^u for all the edge

devices can be expressed as:

$$E^u = \sum_{k=1}^{K} p_k T_k^u. \tag{5.8}$$

In the broadcast phase, since the devices' transmission rates are different, the server broadcasts the global FL model until the last edge device receives the data. Thus, the broadcast time T_0^d can be calculated by:

$$T_0^d = \max_k \left\{ T_k^d \right\} = \max_k \left\{ \frac{A_0}{B_0 \log(1 + \frac{p_0 h_k}{\sigma^2})} \right\}, \tag{5.9}$$

where T_k^d is the transmission time consumed from server to edge device k, and B_0 is the bandwidth of server. We denote p_0 as the transmit power of the server, the broadcast energy E^d is given by:

$$E^d = p_0 T_0^d. \tag{5.10}$$

Computation Energy

The CPU power dissipation includes static and dynamic power, where static power that arises from bias and leakage currents can be dramatically reduced by careful hardware design and dominant CPU power consumption is therefore dynamic power [176]. According to [177], the dynamic power of a CPU P_k^c for edge device k depends on the CPU's supply voltage V_k and CPU computation capacity f_k (the number of CPU cycles/second), i.e., $P_k^c \propto V_k^2 f_k$. In addition, another relation in CPU operation is that f_k is positively proportional to V_k, i.e., $V_k \propto f_k$. Thus, the computation power for edge device k can be written as $P_k^c = \kappa f_k^3$, where κ is the effective switched capacitance depending on the chip architecture. We denote, for edge device k, J_k is the maximum number of local iterations, C_k is the number of CPU cycles required for processing a single sample data, and D_k is the number of collected training samples. The processing time needed can be expressed as:

$$T_k^c = \frac{C_k J_k D_k}{f_k}, \tag{5.11}$$

and the total computation energy for local data processing within a learning cycle is given by:

$$E^c = \sum_{k}^{K} P_k^c T_k^c = \sum_{k}^{K} \kappa J_k C_k D_k f_k^2 = \sum_{k=1}^{K} M_k D_k. \tag{5.12}$$

where $M_k = \kappa I_k C_k f_k^2$. The computation energy at the server can be ignored as the aggregation operation is simple, i.e., averaging [49].

5.2.3 Time Consumption

Fig. 5.2 illustrates the timeline of the FL process. Firstly, the edge devices that have received the broadcast data can perform local calculations immediately. Then, for the data upload, as we consider a half-duplex mode, the server cannot receive and transmit data simultaneously. Before uploading the local FL model to the server, the edge devices need to confirm the server has completed the broadcast task, e.g., receive a control signal allowing to upload. The edge devices that finish the local calculations faster may need to wait for the other devices. Therefore, the time when edge device k starts uploading

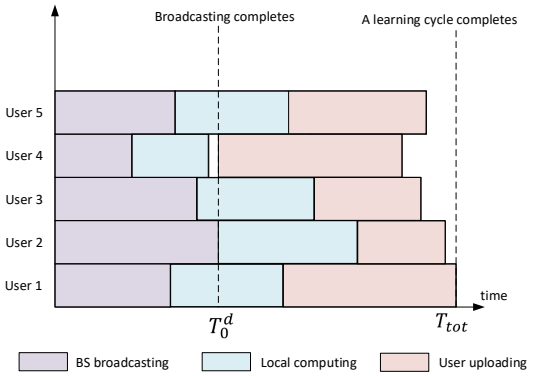


Figure 5.2: Timeline of the FL process.

local learning model is $\max\{T_k^d + T_k^c, T_0^d\}$, where T_k^d, T_0^d and T_k^c can be obtained from (5.9) and (5.11). For example, in Fig. 5.2, user 4 finishes the computation task fast and needs to wait for the broadcast completion. It uploads the local model at T_0^d. For other users, they start uploading data upon finishing the computation, i.e., at $T_k^d + T_k^c$. When all the edge devices complete their upload tasks, we can calculate the total time of a learning cycle by:

$$T_{tot} = \max_k \{T_k^u + \max\{T_k^d + T_k^c, T_0^d\}\} \tag{5.13}$$

5.3 Sparsification in Local Learning Models

From (5.9) and (5.10), the communication energy can be reduced when A_k decreases. In addition, based on (5.12), the computation energy increases with C_k. This motivates us to implement a learning model with fewer weight vectors and simpler computation operations. Although sophisticated learning models, like fully-connected DNNs, can fit non-linear and complex functions, they consume much memory and computing resources. In practice, the learning models with fewer weights or simpler constructions are sufficient to extract the data features with acceptable learning accuracy [79]. Sparse NN (SNN) is one of the methods for simplification. It removes the majority of weights (force them exactly or closely to 0) such that only a percentage of the possible connections exist between layers. Another method is to train a binary NN (BNN) where the weights are driven to either 1 or -1 [174]. The binarized model can dramatically reduce the computational complexity and the amount of data required to compose the model.

To realize SNN, we set regularization items after the loss function such as L1-norm in (5.14) or L2-norm in (5.15).

$$\min_{\mathbf{w}\in\mathbb{R}^n} F_k(\mathbf{w}) = f_k(\mathbf{w}) + \mu\|\mathbf{w}\|, \tag{5.14}$$

$$\min_{\mathbf{w}\in\mathbb{R}^n} F_k(\mathbf{w}) = f_k(\mathbf{w}) + \frac{\mu}{2}\|\mathbf{w}\|^2, \tag{5.15}$$

where μ is the regularization rate selected from [0,1]. To realize BNN, we can use hard thresholding to binarize the weights, e.g., set weights less than 0 as -1, greater than 0 as +1. However, the hard thresholding may cause severe information loss and its discontinuity brings difficulty to the optimization of the learning model. To address this issue, we adopt another way that adds a specifically-designed regularization term to penalize the weights if they are not -1 or +1 instead of binarizing the weights directly [66]. We extent $\mathbf{w}$ to a $2n$-dimensional vector i.e., $\mathbf{w} = [w_1, ..., w_n, a_1, ..., a_n]$, where $w_1, ..., w_n$ are the

weight values and $a_1, ..., a_n$ are the auxiliary variables. The local optimization problem is rewritten by:

$$\min_{\mathbf{w}\in\mathcal{W}} F_k(\mathbf{w}) = f_k(\mathbf{w}) + \frac{\mu}{4}\sum_{m=1}^{n}(a_m(w_m+1)^2 + (1-a_m)(w_m-1)^2), \tag{5.16}$$

where $\mathcal{W} = \{[w_1, ..., w_n, a_1, ..., a_n] | x_m \in \mathbb{R}, a_m \in [0,1], m = 1, ..., n\}$. If μ is sufficiently large, at the optimum, $a_1, ..., a_n$ will be close to 0 or 1. When a_m is close to 0, the effect of $a_m(w_m+1)^2$ is negligible and $(1-a_m)(w_m-1)^2$ is activated. To minimize the loss function, w_m has a strong tendency to be 1. Analogously, when a_m approaches to 1, w_m is forced to -1. To obtain a model with optimal performance, the regularization rate μ should be properly tuned and cannot be excessively large. Therefore, in the implementation, only part of the auxiliary variables reach 0 or 1, and only part of the weights are binary.

In order to quantitatively analyze the energy-saving gain brought by the sparse/binary models, we calculate A_k and C_k for three different learning models, i.e., the DNN, SNN, and BNN. For edge device k, we define Θ_k as the average unit bits corresponding to the numeric data type of the weights and Ψ_k as the total number of the weights of the learning model. Thus, A_k, as the total data volume, is calculated by:

$$A_k = \Theta_k \cdot \Psi_k. \tag{5.17}$$

We consider DNN, SNN and BNN have the same structure with L layers and x_l neurons at the l-th layer, so that Ψ_k is the same for the three models, expressed as $\sum_{l=1}^{L} x_{l-1}x_l + x_l$. The numeric type of the parameter in DNN is floating-point with $\Theta_k = 32$ (bits). For SNN, we assume $\rho_k^1(\%)$ of the weights are zero and the others are floating-point numbers [180]. For BNN, we assume $\rho_k^2(\%)$ of the weights are binary i.e., {+1, -1}, of which the numeric data type is boolean with 1 (bit), and the others are floating-point numbers [181]. Therefore, the average unit bits for SNN and BNN are $\Theta_k = 32(1-\rho_k^1)$ and $\Theta_k = \rho_k^2 + 32(1-\rho_k^2)$, respectively.

To calculate the CPU cycles per sample C_k, we first introduce Φ_k as the required number of floating-point operations (FLOPs) to process a single training sample, where an addition, multiplication, or division is defined as a FLOP. For DNN and SNN, they both need $2\Psi_k$ addition and multiplication operations [158]. For BNN, when the weights are binary, the number of operations is halved as it does not require multiplication operations [181]. The number of CPU cycles needed for each operation is fixed, denoted by e_k, which depends on the performance of the edge device. The number of CPU cycles required for a single training sample C_k is expressed as:

$$C_k = \Phi_k \cdot e_k. \tag{5.18}$$

For clarification, we summarize the expressions of Θ_k, Ψ_k, Φ_k, A_k and C_k for DNN, SNN and BNN in Tab. 5.1.

Table 5.1: Values of Θ_k, Ψ_k, Φ_k, A_k and C_k

	DNN	SNN	BNN
Ψ_k	$\sum_{l=1}^{L} x_{l-1}x_l + x_l$		
Θ_k (bits)	32	$32(1-\rho_k^1)$	$\rho_2 + 32(1-\rho_k^2)$
Φ_k (FLOPs)	$4\Psi_k$	$4\Psi_k$	$2\Psi_k\rho_k^2 + 4\Psi_k(1-\rho_k^2)$
A_k	$\Theta_k \cdot \Psi_k$		
C_k	$\Phi_k \cdot e_k$		

From Tab. 5.1, we can conclude that SNN and BNN can lessen the amount of data to be transferred, and BNN also reduces the computation operations. The energy consumption of SNN and BNN can be decreased with reduced A_k and C_k.

5.4 Resource Optimization in FL

Due to the capacity limitation of the total uplink rate, a large A_k may cause transmission congestion and require more communication resources, e.g., bandwidth and power. In addition, A_k and C_k are proportional to the energy consumption in wireless federated edge learning systems. Hence, resource optimization needs to be considered for energy-efficient FL. Based on the aforementioned sparse/binary models at the edge, we formulate an optimization problem to joint minimize the total energy consumption in FL local training, data uploading, and data broadcasting, while considering the wireless resource constraints and learning performance requirements.

The edge devices, such as smartphones or small-sized sensors, have a low maximum transmit power (e.g., 23 dBm for most LTE users) and weak ability to adjust the uplink power (e.g., open loop for transmit power control) [178]. Thus, to minimize the uplink energy, we optimize bandwidth B_k and with a fixed transmit power p_k suitable for the receiver. For the downlink transmission, all the edge devices receive the broadcasted data via the shared spectrum without interference. To improve the data rate, we set B_0 as the maximum available bandwidth of the server. The downlink energy can be optimized by adjusting transmit power p_0. In addition, we also minimize the computation energy by selecting a proper training data size D_k. This is because the edge devices' computation capabilities are different. If a bad allocation of D_k is made, e.g., uniform D_k for all devices, the devices with higher computation capacity f_k will consume more communication energy, which may exceed the local energy limitation. The minimization problem for a learning cycle is formulated as:

$$\mathbf{P0}: \min_{p_0, B_k, D_k} E^u + E^d + E^c \tag{5.19a}$$

$$s.t.\ T_{tot} \leq T_{th}, \tag{5.19b}$$

$$M_k D_k \leq E^C_{max}, \forall k, \tag{5.19c}$$

$$\Gamma(J_k, D_k) \leq \gamma_{th},\ \ \forall k, \tag{5.19d}$$

$$\sum\nolimits_{k=1}^{K} R_k \leq R_{cap}, \tag{5.19e}$$

$$\sum\nolimits_{k=1}^{K} b_k \leq B_{tot}, \tag{5.19f}$$

$$p_0 < P_{max}. \tag{5.19g}$$

The objective function (5.19a) in $\mathbf{P0}$ includes three components, i.e., local computing energy and upload energy of the edge devices, and broadcast energy of the server. The constraint (5.19b) captures delay requirement. That is, the time consumption at each learning cycle T_{tot} should be less than T_{th}. Due to the limited battery storage, the computation energy for each device should be less than a limit value E^C_{max}, as expressed in (5.19c). Towards minimizing the local computing energy, the optimizer tends to reduce D_k from (5.12). However, a smaller D_k leads to a large gap $\Gamma(J_k, D_k)$ and the learning accuracy deteriorates. To guarantee the learning accuracy of FL, the constraint (5.19d) confines that the local computing gap should be under the threshold γ_{th}. The constraints (5.19e) represent the total uplink rate should not excess the uplink capacity R_{cap}. The constraints (5.19f) and (5.19g) are the limitations of the spectrum and power, respectively.

As the original problem $\mathbf{P0}$ contains the maximum operators, we reformulate $\mathbf{P0}$ to an equivalent

problem $\mathbf{P1}$ with auxiliary variables z_k and T_0^d.

$$\mathbf{P1}: \min_{\substack{p_0,B_k,D_k,\\ z_k,T_0^d}} \sum_{k=1}^{K} \frac{Wp_k}{B_k \log(1+\frac{p_k h_k}{\sigma^2})} + p_0 T_0^d + \sum_{k=1}^{K} M_k D_k$$

$$s.t. \frac{W}{B_0 \log(1+\frac{p_0 h_k}{\sigma^2})} \leq T_0^d, \ \forall k, \tag{5.20a}$$

$$\frac{W}{B_0 \log(1+\frac{p_0 h_k}{\sigma^2})} + \frac{M_k}{f_k} \leq z_k, \ \forall k, \tag{5.20b}$$

$$z_k + \frac{W_k}{B_k \log(1+\frac{p_k h_k}{\sigma^2})} \leq T_{th}, \ \forall k, \tag{5.20c}$$

$$M_k D_k \leq E_{max}^C, \forall k, \tag{5.20d}$$

$$\alpha \left(\frac{1}{\sqrt{J_k D_k}} + \frac{1}{J_k} \right) \leq \gamma_{th}, \ \forall k, \tag{5.20e}$$

$$T_0^D \leq z_k, \ \forall k, \tag{5.20f}$$

$$(5.19e), (5.19f), (5.19g).$$

$\mathbf{P1}$ is a non-convex problem due to the bilinear item $p_0 T_0^d$. To solve the problem, the bilinear term can be relaxed and bounded by McCormick envelopes [63]. Specifically, we set an upper and a lower bound for p_0 and T_0^d, respectively, i.e., (p^L, p^U) and (T^L, T^U). We substitute $p_0 T_0^d$ with an auxiliary variable v and add the following constraints in $\mathbf{P1}$:

$$v \geq p^L T_0^d + p_0 T^L - p^L T^L, \tag{5.21}$$

$$v \geq p^U T_0^d + p_0 T^L - p^U T^L, \tag{5.22}$$

$$v \geq p^U T_0^d + p_0 T^U - p^U T^U, \tag{5.23}$$

$$v \geq p^L T_0^d + p_0 T^U - p^L T^U. \tag{5.24}$$

Therefore, the relaxation problem of $\mathbf{P1}$ becomes a convex problem that can be optimally solved by the interior point method. By using bounds tightening techniques, e.g., optimization-based bounds tightening (OBBT) in [179], the optimal solution of $\mathbf{P1}$ can be approached with ultimately tightened bounds.

We remark that $\mathbf{P1}$ is solved offline at the beginning of each learning cycle for the optimal network and learning configuration. Then, based on the outcome of $\mathbf{P1}$, we perform FL in wireless networks. In $\mathbf{P1}$, the other learning parameters, such as 1) the data to be transmitted at each global iteration A_k and 2) the CPU cycles required for a single training sample C_k, are determined and cannot be optimized under a given machine learning model, e.g., the widely-used deep neural networks (DNNs) [171, 173].

5.5 The Proposed Joint Sparsification and Resource Optimization Scheme

Based on the above sparsification and optimization components, we conclude an energy-efficient FL scheme in wireless networks, i.e., JSRO, as illustrated in Fig. 5.3. The first step is to choose a machine learning model for each edge device. A proper model can achieve a balance between learning accuracy and costs of energy and resource. Specifically, dense DNNs can fit some complex models with better performance than sparse models. However, considering the practical requirements, e.g., limited battery and memory storage, SNN and BNN are more preferred than dense DNN due to the easy implementa-

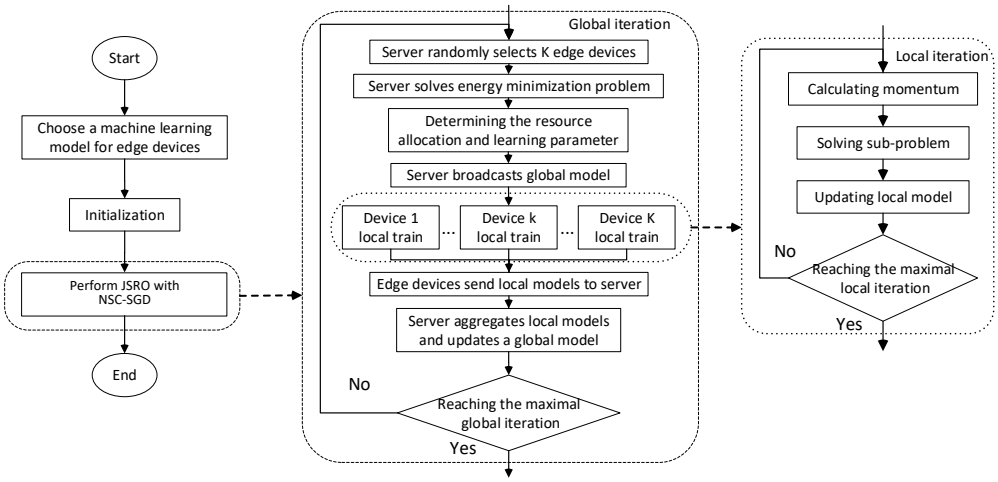


Figure 5.3: The process of JSRO.

tion, lower memory requirement, and less computation and communication energy. The second step is to initialize the learning hyperparameters and network configurations. The final step is to execute the proposed energy-efficient FL scheme JSRO.

The scheme includes global and local iterations. From Fig. 5.3, in the global iteration, the server first randomly selects K edge devices. Then, the energy minimization problem $\mathbf{P0}$ is solved to determine wireless resource allocation, i.e., transmit power of server and uplink bandwidth, and learning parameter assignment, i.e., size of training data. With the optimal solutions of $\mathbf{P0}$, the server broadcasts the global learning model and each edge device trains a local model that will be sent back to the server for aggregation. As all the edge devices collect and process the training data locally and in parallel, user privacy is protected and training efficiency is improved. After receiving the local models, the server can update the global model.

The local iteration refers to the local training process performed at edge devices. Firstly, a minibatch of data is selected to estimate the momentum of the gradient. With the momentum, we formulate a convex sub-problem to deal with the issues of constrained and non-smooth learning models. Then, we calculate the direction of gradient descent which guides the update of the local model. It is worth noting that the algorithm adopted in local training plays a key role in JSRO. In the following, we discuss the developed NSC-SGD for local training in detail.

5.5.1 NSC-SGD for Local Training in JSRO

The existed FL algorithms, e.g., FedAvg, FedDANE and FedProx, guarantee the convergence when the following two conditions are satisfied:

- The loss function $F_k(\mathbf{w})$ is differentiable.
- The space of parameters $\mathcal{W} = \mathbb{R}^n$, where n is the dimension of $\mathbf{w}$.

However, the above algorithms cannot deal with non-smoothness and constraints in SNN and BNN, e.g., L1-norm regularization $\mu\|\mathbf{w}\|$ for SNN and constraints on variables $a_m \in [0, 1]$ for BNN, which may lead to performance degradation on the learning performance. ProxSGD algorithm is able to deal with the non-smooth and constrained learning models but it is specialized for centralized learning. To tackle this issue, we propose an NSC-SGD algorithm by extending ProxSGD in FL framework. We first

adopt ProxSGD to train the local FL models on edge devices. Then, the global FL model is updated by aggregating the outputs of ProxSGD optimizer. In addition, the convergence proof of ProxSGD is not applicable for FL. Thus, we perform an analysis of the convergence of NSC-SGD in FL framework.

Based on Fig. 5.3, the proposed scheme with enhanced NSC-SGD is summarized in Alg. 7, where NSC-SGD refers to the process in iteratively training local models (lines 10-15). Considering a general machine learning model with non-smooth and constrained loss functions, the following two assumptions hold for relaxing the conditions.

Assumption 1. *(Smoothness and convexity) $F_k(\mathbf{w}) = f_k(\mathbf{w}) + r_k(\mathbf{w})$, where $f_k(\mathbf{w})$ is smooth, i.e., continuously differentiable, but not necessarily convex, and $r_k(\mathbf{w})$ is proper convex and lower semicontinuous but not necessarily smooth.*

Assumption 2. *(Convex set) $\mathcal{W}$ is convex and compact set.*

In a learning cycle (lines 5-19), the update of the global FL model (lines 17-18) is consistent with the general FL framework. In the local iteration of each device (lines 10-15), we aim at solving the local loss function (5.3). Firstly, a minibatch $\mathcal{M}^{i,j}$ is randomly selected from the training data to estimate the gradient of $f_k(\mathbf{w})$, i.e.,

$$\bar{\nabla} f_k(\mathbf{w}^{i,j}) = \frac{1}{|\mathcal{M}^{i,j}|} \sum_{l \in \mathcal{M}^{i,j}} \nabla \mathcal{L}_{k,l}(\mathbf{w}^{i,j}). \tag{5.25}$$

To keep the gradient updated in the right direction and accelerate the converge speed, we introduce momentum $\mathbf{v}_k^{i,j}$, which is formed in a recursive manner:

$$\mathbf{v}_k^{i,j} = (1 - \varrho^j)\mathbf{v}_k^{i,j-1} + \varrho^j \bar{\nabla} f_k(\mathbf{w}^{i,j}), \tag{5.26}$$

where $\varrho^j \in (0, 1]$ is the step size for the momentum. Then, we propose to solve an approximation sub-problem and obtain an intermediate solution $\hat{\mathbf{w}}^{i,j}$:

$$\hat{\mathbf{w}}_k^{i,j} = \underset{\mathbf{w} \in \mathcal{W}}{\operatorname{argmin}} \, (\mathbf{w} - \mathbf{w}^{i,j})^{\mathrm{T}} \mathbf{v}_k^{i,j} + \frac{\tau}{2} \|\mathbf{w} - \mathbf{w}^{i,j}\|^2 + r_k(\mathbf{w}). \tag{5.27}$$

Based on successive convex approximation (SCA), the objective in (5.27) is a convex approximation of $F_k(w)$ around the point $\mathbf{w}^{i,j}$ by incorporating first-order Taylor expansion, quadratic regularization and non-smooth term. Compared to the conventional SGD, the update rule in (5.27) can resolve the issues of weights' constraints and non-smooth (non-differentiable) objectives. The difference between $\hat{\mathbf{w}}^{i,j}$ and $\mathbf{w}^{i,j}$ specifies the update direction which is used to refine the parameters:

$$\mathbf{w}_k^{i,j+1} = \mathbf{w}_k^{i,j} + \varepsilon_{loc}(\hat{\mathbf{w}}_k^{i,j} - \mathbf{w}_k^{i,j}). \tag{5.28}$$

When the local training terminates at the J_k-th iteration, each edge device uploads the local learning model to the server. The server collects all the information and updates the global model by (5.5). The training process ends when reaching the maximal number of global iterations I.

5.5.2 Convergence Analysis

To analyze the convergence of Alg. 7, we first make the following assumptions referring to [166, 168, 171, 48].

Assumption 3. *(Lipschitz gradient continous) $\nabla f_k(\mathbf{w})$ is L-Lipschitz continuous, i.e., $f_k(\mathbf{w}') + \nabla f_k(\mathbf{w}')^{\mathrm{T}}(\mathbf{w} - \mathbf{w}') + \frac{L}{2}\|\mathbf{w} - \mathbf{w}'\|^2 \geq f_k(\mathbf{w})$, where L refers to the Lipschitz constant.*

Algorithm 7 JSRO for Energy-Efficient FL

Inputs:
Initial global FL model: $\mathbf{w}^0$;
Maximum number of global iteration: I;
Number of local iterations for each edge device: J_k;
Learning rate: ϵ.

for $i = 0 : I$ **do**
  Server selects K devices randomly.
  Server determines the optimal resource allocation and learning parameter by solving $\mathbf{P0}$.
  Server sends w^i to all chosen devices.
  for $k = 1 : K$ (do in parallel) **do**
    for $j = 0 : J_k$ **do**
      Edge device selects minibatch of collected data.
      Edge device calculates momentum by (5.26).
      Edge device solves sub-problem (5.27).
      Edge device updates $\mathbf{w}_k^{i,j}$ by (5.28).
    end for
  end for
  All the devices send $\mathbf{w}_k^i$ back to the server.
  Server updates $\mathbf{w}^i$ by (5.5).
end for

Outputs: Well-trained local model $\mathbf{w}_k^*$.

Assumption 4. *(Strongly convex) $\nabla f_k(\mathbf{w})$ is χ-strongly convex, i.e., $f_k(\mathbf{w}') + \nabla f_k(\mathbf{w}')^{\mathrm{T}}(\mathbf{w} - \mathbf{w}') + \frac{\chi}{2}\|\mathbf{w} - \mathbf{w}'\|^2 \leq f_k(\mathbf{w})$, where χ is a constant coefficient.*

Assumption 5. *(Bounded gradient) Defining an approximated gradient of $F(\boldsymbol{w})$ by:*

$$\bar{\nabla} F(\mathbf{w}; \mathbf{w}') = \|\mathbb{E}_k[\nabla f_k(\mathbf{w})]\| + \|\frac{\mathbb{E}_k[r_k(\mathbf{w}) - r_k(\mathbf{w}')]}{\mathbf{w} - \mathbf{w}'}\|,$$

$\bar{\nabla} F(\mathbf{w}; \mathbf{w}')$ is bounded such that, for a positive constant M, $\|\bar{\nabla} F(\mathbf{w}; \mathbf{w}')\| \leq M\|\mathbf{w} - \mathbf{w}'\|$.

Assumption 6. *(Bounded dissimilarity) For some ξ>0, there exists a constant S such that for all the points $\mathbf{w} \in \{\mathbf{w} | \|\bar{\nabla} F(\mathbf{w}; \mathbf{w}^{i+1})\|^2 > \xi\}$, $\mathbb{E}_k\left[\bar{\nabla} F_k(\mathbf{w}; \mathbf{w}_k^i)^2\right] \leq S^2 \bar{\nabla} F(\mathbf{w}; \mathbf{w}^{i+1})^2$, where*

$$\bar{\nabla} F_k(\mathbf{w}; \mathbf{w}') = \|\nabla f_k(\mathbf{w})\| + \|\frac{r_k(\mathbf{w}) - r_k(\mathbf{w}')}{\mathbf{w} - \mathbf{w}'}\|. \tag{5.29}$$

Theorem 6. *(Convergence) If χ, K, M, L, S and ϵ are selected such that*

$$\eta = \left(\frac{2S\epsilon(1 + \epsilon M)}{\chi} + \frac{2\epsilon^2 S^2 L}{\chi^2} - \epsilon + \frac{\epsilon\sqrt{8}S}{\sqrt{K}\chi} + \frac{8\sqrt{2}LS^2\epsilon^2}{\sqrt{K}\chi^2} + \frac{8LS^2\epsilon^2}{K\chi^2} \right) < 0, \tag{5.30}$$

then the expected decrease on the global objective holds, i.e.,

$$\mathbb{E}_{\mathcal{K}_i}[F(\mathbf{w}^{i+1})] \leq F(\mathbf{w}^i) + \eta \bar{\nabla} F(\mathbf{w}^i; \mathbf{w}^{i+1})^2, \tag{5.31}$$

Table 5.2: Parameter settings

Data set	MNIST
Total number of devices	1000
Number of selected devices per learning cycle K	10
Transmit power on edge device p_k	0.5 Watt
Bandwidth of server B_0	3 MHz
SINR	15-23 dB
Input size	28$\times$ 28 images
Number training data samples	69035
Learning model	DNN, SNN, BNN
Loss function	cross entropy
learning rate ϵ	0.001
Number of global iterations I	500
Batch size	50
Number of local iterations J_k	50
Penalty parameter μ	0.0025
Unit bits of a single weight	float type: 32 boolean type: 1
Computation capacity f_k	6.725 GHz
FLOPs per CPU cycle e_k	4
Effective switched capacitance κ	10^{-28}

where $\mathcal{K}_i$ is the set of selected devices at global iteration i.

Proof. The proof of Theorem 6 is shown in the Appendix. □

The JSRO is designed to reduce the energy consumption via resource optimization and local model sparsification. There exists an interplay between energy minimization and learning performance.

- From the aspect of learning model selection, SNN and BNN save energy with fewer transmitted parameters and simpler computing operations but the learning performance degrades as the simplified ML models may lead to the loss of information and underfitting. To improve the learning performance while keeping the learning model simple, we design Alg.1 to provide a convergence guarantee and speed up the learning process for a general learning model.
- From the aspect of optimization, the computing energy in $\mathbf{P0}$ can be saved by decreasing the training data size D_k. However, the gap to the optimal $\Gamma(J_k, D_k)$ at local training enlarges with a small D_k. When the local learning accuracy is substandard, the global learning performance will decline. Thus, in $\mathbf{P0}$, we optimize D_k to ensure that the local devices consume minimal energy while satisfying the gap threshold γ_{th}.

5.6 Simulation Results

In this section, we evaluate the performance of the proposed JSRO. We compare the performance of different learning models, i.e., DNN, SNN, and BNN, in terms of distribution of the weight values and the total energy consumption, and verify the energy conservation brought about by resource optimization.

In addition, we compare the proposed NSC-SGD algorithm adopted in JSRO with other FL algorithms, i.e., FedAvg and FedProx, on the performance of learning accuracy. The adopted parameters for implementing JSRO are summarized in Table 5.2.

5.6.1 Performance on Energy Conservation

To demonstrate the impact of different learning models on energy consumption, we simulate the cumulative distribution function (CDF) of weight values and show the total energy consumption of DNN, SNN, and BNN in Fig. 5.4 and 5.5, respectively. From Fig. 5.4, in DNN, around 80% weights are evenly distributed in the interval [-0.2, 0.2], and few weights are with $\|\mathbf{w}\| < 0.01$. Thus, all the weights in DNN should be uploaded or broadcasted as floating-point values. For SNN, around 50% of the weights are exactly equal to 0, such that only half of the weight needs to participate in the communication. For BNN, we can observe that the penalty and constraint in the designed loss function (5.16) is effective in promoting binary weights: more than 60% weights are -1 or +1, which can be transmitted as boolean values.

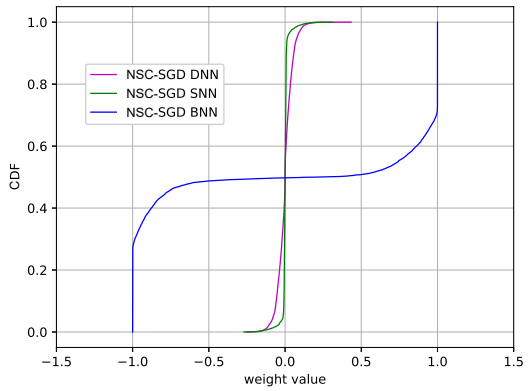


Figure 5.4: Distribution of weights of different neural networks.

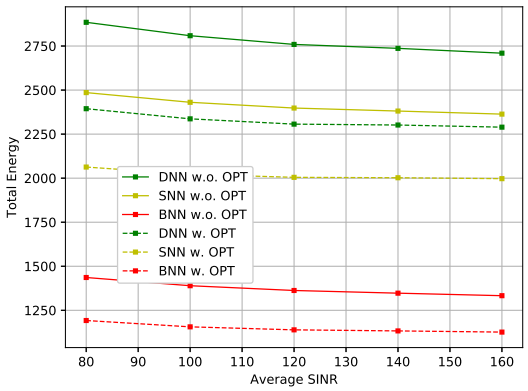


Figure 5.5: Total energy vs. SINR.

In Fig. 5.5, we show the total energy of different learning models adopted in JSRO with regard to the average SINR. Generally, the total energy consumption decreases with a higher value of SINR. This

is because, in a better channel condition, the transmission rate is higher and the consumed energy can be saved with less transmission time. The proposed JSRO with BNN and energy optimization saves 56.21% total energy compared to the FL without JSRO, i.e., DNN-based FL without energy optimization. Among the three learning models, BNN consumes the least total energy, 39.81% less than DNN, followed by SNN which saves 12.72% total energy compared to DNN. Besides, we can also find that, for each learning model, the optimization, i.e., wireless resource allocation and learning parameter selection, can further reduce energy consumption by 16.43%.

To illustrate which part of the energy is reduced by SNN and CNN, three energy components are depicted in Fig. 5.6. We can observe that the computation energy dominates the total energy consumption. Compared to DNN, SNN and BNN save uploading energy by 59.32% and 65.46%, respectively. This is because the zero-weights in SNN and binary-weights in BNN reduce the volume of transmitted data. For the broadcasting energy, BNN and SNN are not significantly different from DNN as the aggregated global FL model may become back to dense. The computation energy of SNN and DNN are close but BNN is 47.22% less than DNN. The reason is that the computing operations in DNN and SNN are based on floating-point calculation while BNN benefits from boolean operations that reduce the computation energy by simplifying the computation and lessening the required CPU cycles.

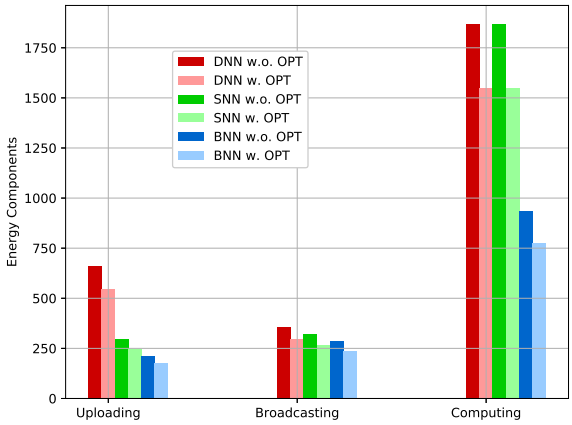


Figure 5.6: Energy parts.

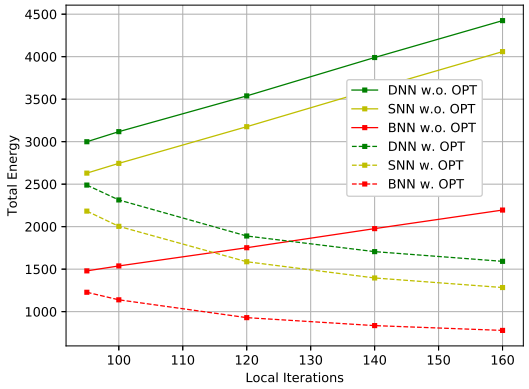


Figure 5.7: Total energy vs. Local iterations.

Fig. 5.7 demonstrates the total energy with respect to the maximal of local iterations J_k. Based on the energy models in section III, the computation energy is positively proportional to J_k and the size of training data D_k. With the optimization (OPT), when J_k increases, the optimizer will reduce D_k to minimize the total energy. As the optimal solution of D_k decreases faster than J_k increases, from Fig. 5.9, the total energy with OPT decreases with J_k. However, without OPT, D_k is constant and non-optimized. Thus, the total energy increases linearly with J_k.

5.6.2 Performance on Learning Accuracy

In addition to energy consumption, another indicator of JSRO is learning accuracy. Fig. 5.8 shows the learning accuracy of JSRO with different learning models. For the learning task, i.e., MNIST dataset, with the identical learning rate and training data, DNN outperforms the others, reaching 89% accuracy. For SNN and BNN, the accuracy decreases to around 88% and 80%, respectively. The reason lies that SNN and BNN drive the weights of the learning model to zero or binary by adding the penalty terms and constraints, leading to the issue of underfitting and degradation of the accuracy. However, recalling Fig. 5.5-5.7, the SNN and BNN simplify the local model and reduce the energy consumption.

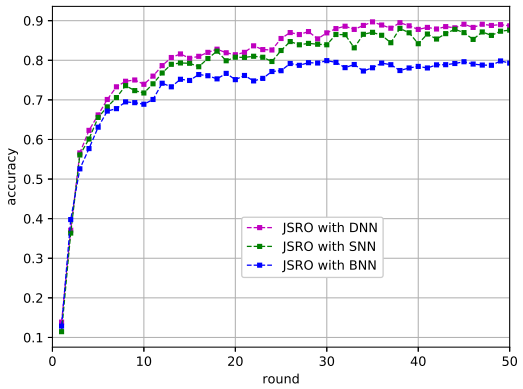


Figure 5.8: Accuracy of different neural networks with JSRO.

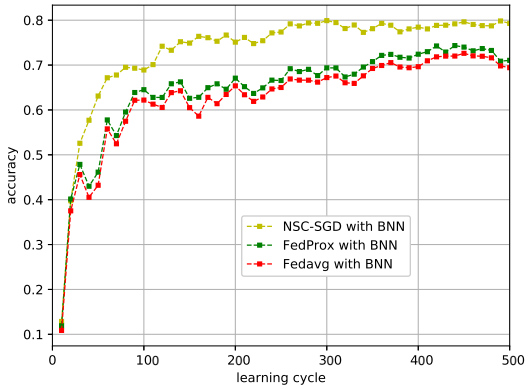


Figure 5.9: Accuracy of different federated optimizers with BNN.

To verify the effectiveness of the NSC-SGD algorithm adopted in JSRO on improving the learning performance for general learning models. Fig. 5.9 compares the learning accuracy of three FL algorithms under the case of BNN. The results show that the learning accuracy of NSC-SGD is 8% higher than that of FedProx. The gap between the FedProx and Fedavg is small, around 2%. In addition, the proposed NSC-SGD converges much faster and is more stable than FedProx and Fedavg. The performance improvement is due to the fact that the update rule of local training is based on solving a sub-problem (5.27) which applies to general learning models with constraints, non-smooth regularization, and penalty terms.

5.7 Conclusion

In this chapter, we investigated an energy-efficient federated scheme, i.e., JSRO, implemented in wireless federated edge learning networks to economize energy from two perspectives. Firstly, we formulated an optimization problem to minimize the energy consumption, including communication and computation energy, via wireless resource management and learning parameter allocation. Secondly, based on the analysis of the energy consumption of different learning models, the energy can be further saved by selecting sparse or binary NN instead of traditional DNN. To improve the learning performance, i.e., the convergence and accuracy, we proposed NSC-SGD to provide convergence guarantee for general learning models with non-smooth and constrained loss functions. Numerical results show that BNN consumes around 40% less energy than DNN, and resource optimization can further save 16.43% of energy. In addition, the proposed JSRO with embedded NSC-SGD achieves better performance on convergence and learning accuracy compared to FedProx and Fedavg.

Chapter 6

Conclusions and Future Works

In this chapter, we conclude the researches of this thesis in section 6.1 and provide an outlook for possible future works in section 6.2.

6.1 Conclusions

In this thesis, we focus on designing ML-based resource management schemes in future wireless systems to achieve energy-, time-, and resource-efficient communication with fast and online solutions. We selected four promising and interesting future wireless scenarios, including multi-antenna systems, UAV-assisted communication, satellite-terrestrial networks, and distributed wireless architecture. To overview the background of the topic, we introduce the adopted methodologies and describe the selected research problems and their motivations. Then, each research problem is elaborated chapter by chapter in the main body.

Firstly, we developed an ML-assisted approach for resource allocation to enable a fast, feasible, and near-optimal solution for both time-efficient and energy-efficient content delivery in MISO systems. As the optimization-based methods have exponential complexity, FC-DNN and CNN are adopted for real-time, intelligent, and efficient solutions. Different from the conventional pure ML algorithm, the proposed learning-assisted approach applies NNs to predict the learnable features which can be used to accelerate the optimization process and improve the learning accuracy. The learning-assisted algorithm can deal with the feasibility and achieve a good trade-off between the complexity and solution quality.

Secondly, we investigated a joint user-timeslot scheduling and hovering time allocation problem for energy minimization in UAV-aided communication systems. We provided an optimal method and proposed a heuristic algorithm, which served as a benchmark, to solve the formulated combinatorial and non-convex problems. To make the solutions adaptive to online operations, an AC-DRL algorithm was developed. The proposed algorithm outperforms other algorithms in energy efficiency and computational efficiency. In addition, the action space restriction and re-designed reward function can deal with the huge discrete action space and guarantee feasibility, respectively.

Thirdly, we investigated a resource scheduling problem in dynamic LEO-terrestrial communication systems to address the mismatch issue in a practical over-loaded scenario. The problem was solved from the perspective of AC-DRL to tackle the issue of high computational time of the optimization-based algorithms and obtain online solutions. To enable the learning model to fast adapt to dynamic environments, we developed a meta-critic-based algorithm to handle the environmental changes in wireless networks, such as bursty demands, users' entry/leave, and abrupt channel change. We verified that, when

encountering an environmental variation, the proposed method consumes less recovery time to re-fit the learning model, compared to other AC-DRL methods.

Fourthly, we designed an energy-efficient FL scheme for distributed wireless communication systems. We first observed and analyzed that the energy can be saved by selecting sparse NN (SNN) or binary NN (BNN) instead of dense DNN. To improve the learning performance for general learning models with non-smooth and constrained loss functions, such as BNN and SNN, an enhanced SGD-based federated learning algorithm was proposed with the proof of convergence. Then, the communication and computation energy were further minimized by wireless resource management and learning parameter allocation. By integrating the above strategies, we designed a comprehensive FL wireless scheme that consumes much less energy and increases learning accuracy.

To sum up, this thesis reveals that 1) there is no omnipotent ML algorithm that can solve all the practical problems in future wireless systems. To achieve better performance, the ML tools need to be properly selected, e.g., supervised learning is not suitable for complicated cases where the labeled training data is difficult to acquire and centralized learning methods are not as efficient and secure as FL in distributed network structures. 2) The ML-based algorithms can notably reduce the time complexity and realize real-time decision-making while guaranteeing the solution quality. 3) ML methods have limitations in designing communication schemes. Specifically, we observe from chapter 2, there exists difficulties in finding the complicated relationship between input and output due to, e.g., the noise in the training data. Chapter 3 demonstrates the conventional ML algorithms have issues of feasibility and curse of dimensionality when the original problems are constrained and combinatorial. From chapter 4, the performance of the original ML models can be degraded in highly dynamic wireless environments. In this thesis, we designed some techniques to solve or relieve these problems. 4) In certain situations, there exists an interplay between optimization and ML performance. For example, in chapter 5, the volume of the training data affects the learning accuracy but, from the perspective of energy optimization, more training data means more local computation energy consumption.

6.2 Future Works

In this section, we discuss the extensions of the thesis and potential future work, which either focus on addressing some of the unsolved difficulties or explore the new directions following the current research trends.

6.2.1 Feasibility Problem in ML

The feasibility of the results produced by ML algorithms is a restriction for the application of AI in real-world scenarios. For example, when NNs are used for spectrum allocation, the total bandwidth of all the users may exceed the system limitation. This is because ML methods train their models based on the data or experiences without considering the practical constraints during the process of inference. In this thesis, in chapter 2, we use the learning-assisted approach by learning the extracted features instead of the original labels to avoid the feasibility issue. In chapter 3, we reduce the action space to filter the infeasible decisions. However, the above methods are limited and are unsuitable for general cases. Specifically, for the learning-assisted methods, ML predicts the extracted non-constrained features rather than the constrained original outputs. If the extracted features are constrained, the model cannot ensure feasibility. For the action space restriction, it can only filter the actions violate the simple constraints, e.g., $p_k < P_{tot}$. When the constraints are complex, e.g., $\frac{p_k h_k}{\sum_{j \neq k} p_j h_j + \sigma^2} > SINR_{th}$, the agent cannot easily filter out infeasible p_k. Therefore, one direction of future work could be finding a technique to

automatically and efficiently identify and remove the solutions that violate the constraints during model training.

6.2.2 Embedding ML in Optimization Algorithm

In the current research, there are three general ways to apply ML to optimization problems in wireless scenarios. The first is to directly apply an ML algorithm to solve an optimization problem, i.e., E2E learning discussed in chapter 2. However, E2E learning may perform poorly in terms of solution quality, especially when the problem is complex and the input-output correspondence is difficult to learn. In this case, the complexity of conventional optimization-based methods may be too high to satisfy the strict delay requirement. The second is the learning-assisted method adopted in chapter 2 and previous works [73, 182, 76], which utilizes ML to extract important information from the collected data and then execute a conventional optimization algorithm with the extracted features. This method can speed up the optimization algorithm to provide optimal solutions and avoid infeasible solutions. Nevertheless, for the learning-assisted method, the learning and optimization are separated. We need to manually find out the features which are suitable for learning and useful for reducing the complexity of the optimization. The third method is embedding ML models into the optimization algorithms. Different from the learning-assisted method, this approach nests the learning algorithm within the process of optimization thus the pre-training process can be avoided. For example, in [183], to improve the efficiency of B&B algorithm, which is widely used in combinatorial optimization, the authors use ML to learn the node selection policy and node pruning policy to guide the decision-making in B&B tree. Based on this direction, for most optimization algorithms based on iterative updating, we can embed an ML model into each iteration to reduce the computation time or the number of needed iterations. The method of ML embedding can achieve a good trade-off between efficiency and quality, but its difficulty lies in (1) finding a proper ML model to quickly solve the most time-consuming part in each iteration; (2) tackling the feasibility issue of ML outputs, which could be the focus in our future works.

6.2.3 Multi-Agent Learning Structure

In chapters 3 and 4, the considered UAV-aided and LEO-terrestrial systems are centralized communication structures with a single learning agent to make decisions. This may cause much communication and traveling costs and much time to coordinate training schedules. In chapter 5, although FL allows the devices to train their models locally, it also needs a centralized controller to exchange and integrate global information. To facilitate coordination among devices and enhance the learning performance of all participants, the multi-agent ML model is proposed, e.g., multi-agent reinforcement learning (MARL), in which the agents set policies according to their own observations and observations of other agents. In the literature review, there exist studies applying multi-agent ML to distributed communication systems. In [184], the authors design an MARL algorithm in multi-UAV networks where each UAV acts as a learning agent to make the decision of resource allocation solution based on its local observation. In [185], a multi-agent DQN algorithm is applied to dynamic power allocation in wireless networks. However, in wireless scenarios, multi-agent ML still faces two main challenges. The first is the imperfect inter-agent communication due to, e.g., limited transmission rate. This may lead to difficulty in acquiring the precise global information of each agent thus degrading the learning performance. To solve this issue, in [186], a state aggregation method was proposed to enable the agents to communicate their information in a compact manner. Whereas, when the number of agents is large, the operation of compression could be complex and time-consuming. Thus, in future work, we can design a more efficient strategy to save resources required for information exchange and processing among the agents while maintaining the learning quality. The second challenge is the heterogeneity of the learning agent. In many multi-agent

ML applications, all the agents are identical but, in heterogeneous networks, this is not necessarily the case. For example, in a multi-layer non-terrestrial communication system, UAVs and satellites can serve ground users, and satellites also provide backhaul links to UAVs. If we regard both UAVs and satellites as the agents to allocate the wireless resources, their channel quality, transmission rate, resource reserve, and other properties could be significantly different. The authors in [187] proposed an MARL algorithm for resource allocation in heterogeneous networks but the actions of the agents are different and there is no resource competition between different types of agents. Thus, a possible extension of our work is investigating multi-agent ML in heterogeneous systems and discovering suitable relations between the agents, i.e., competition, cooperation, or mixed competition and cooperation, based on the differences in their abilities.

6.2.4 Interpretability and Theoretical Supports of ML Models

Poor interpretability is a common issue for ML models. For example, NNs are deemed as a black box as they cannot provide any insights into the structures of the approximated functions. In general, with higher interpretability, it is easier to understand the reason why certain decisions or predictions have been made. In the area of computer science, many researchers have explored the interpretability of ML models. For example, pre-model interpretability methods, such as PCA [188] and t-distributed stochastic neighbor embedding [189], can provide a good understanding of the data before ML model selection. In wireless systems, the poor performance of ML algorithms is probably due to the complexity and noise in the original data. If we can find the characteristics or reduce the dimension of the data in advance, this can help to select correct ML models and improve prediction accuracy. In addition, due to the uncertainty and stochastic nature of wireless scenarios, the optimality and convergence of certain ML algorithms cannot be guaranteed, which impacts their usefulness [190]. Thus, in future works, we could focus on the interpretability, optimality, and convergence of ML under the wireless networks to acquire more explanation on *why selecting the model, how the model works, and can the algorithm fast converge*.

6.2.5 AI application areas in B5G/6G scenarios

As AI is considered one of the most promising solutions in future wireless systems, we extend the proposed ML approaches to more communication scenarios to improve the systems' performance, robustness, and automaticity. The potential applications include (i) Network slicing resource management. The main challenge of resource management in network slicing is the high complexity as the cross-slice resource allocation and admission control problems are generally non-convex and their decision variables are integer values, e.g., '0' and '1' for declining and accepting admission, respectively [191, 192, 193, 194]. Thus, towards efficient resource allocation and automatic admission control, we may apply the enhanced AC algorithm in Chapter 3, which can avoid solving the complicated MIP problem and reduce the dimension of the decision variables by action space restriction. (ii) Beamforming in massive MIMO. Considering the combinations of different antenna parameters, selecting an optimal beam pattern could be time-consuming in massive MIMO systems. AI-based beamforming design has been widely investigated to select the optimal beam pattern in real-time. In [195],[196] and [197], the authors applied ML models to antenna selection, CSI feedback, and beam selection to improve beamforming gain and reduce complexity. However, the above works adopted E2E learning, which could not guarantee optimality. In future work, we could extend the learning-based method in Chapter 2 to beam pattern selection, where the ML model is used to learn some useful features to pre-exclude some beam patterns and accelerate the computational time of generating the optimal solution. (iii) Device-to-device (D2D) communication. AI can also be used to design resource management [198], power control [199], and clustering schemes [200] in D2D communication toward efficient and intelligent online decisions.

However, ML techniques still face two challenges in D2D networks [201]. Firstly, highly dynamic D2D networks. Most ML algorithms are targeted for stochastic stationary environments, but the environments in D2D communications are dynamic, e.g., frequently change in topology due to the high-speed trains and vehicles. In this situation, the meta-learning approach adopted in Chapter 4 can be considered to achieve a self-adaptive ML model. Secondly, D2D networks for FL. D2D, as a decentralized network, each device can train the ML model locally. In [202], a decentralized stochastic gradient descent algorithm was implemented for large-scale FL wireless D2D networks. However, exploiting FL techniques for user scheduling and resource allocation with energy constraints in D2D networks is a challenging task, which could be the extension work of Chapter 5.

Bibliography

[1] S. Chen, S. Sun, and S. Kang. System integration of terrestrial mobile communication and satellite communication—the trends, challenges and key technologies in b5g and 6g. *China communications*, 17(12):156–171, December 2020.

[2] NGMN Alliance. Ngmn 5g white paper. Technical report, Next Generation Mobile Networks Ltd., Frankfurt am Main, March 2015.

[3] S. Han, I. Chih-Lin, G. Li, S. Wang, and Q. Sun. Big data enabled mobile network design for 5g and beyond. *IEEE Communications Magazine*, 55(9):150–157, September 2017.

[4] C. Larsson. *5G networks: planning, design and optimization.* Academic Press, August 2018.

[5] Z. Mlika and S. Cherkaoui. Massive iot access with noma in 5g networks and beyond using online competitiveness and learning. *IEEE Internet of Things Journal*, 8(17):13624–13639, March 2021.

[6] Z. Ning, X. Wang, J. J. Rodrigues, and F. Xia. Joint computation offloading, power allocation, and channel assignment for 5g-enabled traffic management systems. *IEEE Transactions on Industrial Informatics*, 15(5):3058–3067, May 2019.

[7] S. D'Oro, L. Bonati, F. Restuccia, and T. Melodia. Coordinated 5g network slicing: How constructive interference can boost network throughput. *IEEE/ACM Transactions on Networking*, 29 (4):1881–1894, August 2021.

[8] R. S. Michalski and J. G. Carbonell. *Machine learning: An artificial intelligence approach*. Springer Science and Business Media, April 2013.

[9] O. Sholev, H. H. Permuter, E. Ben-Dror, and W. Liang. Neural network mimo detection for coded wireless communication with impairments. In *IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–8, 2020.

[10] P. Goswami, A. Mukherjee, M. Maiti, S. K. Tyagi, and L. Yang. A neural network based optimal resource allocation method for secure iiot network. *IEEE Internet of Things Journal*, May 2021.

[11] M. Safaldin, M. Otair, and L. Abualigah. Improved binary gray wolf optimizer and svm for intrusion detection system in wireless sensor networks. *Journal of ambient intelligence and humanized computing*, 12(2):1559–1576, February 2021.

[12] D. Liu and O. Simeone. Channel-driven monte carlo sampling for bayesian distributed learning in wireless data centers. *IEEE Journal on Selected Areas in Communications*, October 2021.

[13] H. Qi, Z. Hu, H. Huang, X. Wen, and Z. Lu. Energy efficient 3-d uav control for persistent communication service and fairness: A deep reinforcement learning approach. *IEEE Access*, 17 (8):53172–53184, March 2020.

[14] K. Feng, Q. Wang, X. Li, and C. K. Wen. Deep reinforcement learning based intelligent reflecting surface optimization for miso communication systems. *IEEE Wireless Communications Letters*, 9 (5):745–749, 2020.

[15] H. Yang, Z. Xiong, J. Zhao, D. Niyato, L. Xiao, and Q. Wu. Deep reinforcement learning-based intelligent reflecting surface for secure wireless communications. *IEEE Transactions on Wireless Communications*, 20(1):375–388, September 2020.

[16] N. Garg, M. Sellathurai, V. Bhatia, and T. Ratnarajah. Function approximation based reinforcement learning for edge caching in massive mimo networks. *IEEE Transactions on Communications*, 69(4):2304–2316, April 2021.

[17] H. Sun and et al. Learning to optimize: Training deep neural networks for interference management. *IEEE Transactions on Signal Processing*, 66(20):5438–5453, October 2018.

[18] Y. Yuan, T. X. Vu, L. Lei, S. Chatzinotas, and B. Ottersten. Joint user grouping and power allocation for miso systems: Learning to schedule. In *27th European Signal Processing Conference (EUSIPCO)*, pages 1–5, 2019.

[19] R. Li, Z. Zhao, X. Chen, J. Palicot, and H. Zhang. Tact: A transfer actor-critic learning framework for energy saving in cellular radio access networks. *IEEE Transactions on Wireless Communications*, 13(4):2000–2011, April 2014.

[20] M. E. Morocho-Cayamcela, H. Lee, and W. Lim. Machine learning for 5g/b5g mobile and wireless communications: Potential, limitations, and future directions. *IEEE Access*, 7:137184–137206, September 2019.

[21] K. He, Z. Wang, D. Li, F. Zhu, and L. Fan. Ultra-reliable mu-mimo detector based on deep learning for 5g/b5g-enabled iot. *Physical Communication*, 43, December 2020.

[22] L. Bai, Z. Huang, Y. Li, and X. Cheng. A 3d cluster-based channel model for 5g and beyond vehicle-to-vehicle massive mimo channels. *IEEE Transactions on Vehicular Technology*, 70(9): 8401–8414, July 2021.

[23] M .Alodeh, D. Spano, A. Kalantari, C. G. Tsinos, D. Christopoulos, S. Chatzinotas, and B. Ottersten. Symbol-level and multicast precoding for multiuser multiantenna downlink: A state-of-the-art, classification, and challenges. *IEEE Communications Surveys and Tutorials*, 20(3): 1733–1757, 2018.

[24] J. Choi, N. Lee, S. Hong, and G. Caire. Joint user scheduling, power allocation, and precoding design for massive mimo systems: A principal component analysis approach. In *IEEE International Symposium on Information Theory (ISIT)*, 2018.

[25] A. Bandi, R. B. S. Mysore, S. Chatzinotas, and B. Ottersten. Joint user scheduling, and precoding for multicast spectral efficiency in multigroup multicast systems. In *International Conference on Signal Processing and Communications (SPCOM)*, 2020.

[26] T. X. Vu, S. Chatzinotas, and B. Ottersten. Dynamic bandwidth allocation and precoding design for highly-loaded multiuser miso in beyond 5g networks. *IEEE Transactions on Wireless Communications*, August 2021.

[27] Z. Chang, L. Lei, Z. Zhou, S. Mao, and T. Ristaniemi. Learn to cache: Machine learning for network edge caching in the big data era. *IEEE Wireless Communications*, 25(3):28–35, June 2018.

[28] W. Lee, M. Kim, and D. Cho. Deep power control: Transmit power control scheme based on convolutional neural network. *IEEE Communications Letters*, 22(6):1276–1279, June 2018.

[29] A. Zappone, M. Renzo, and M. Debbah. Wireless networks design in the era of deep learning: Model-based, ai-based, or both? *IEEE Transactions on Communications*, 67(10):7331–7376, June 2019.

[30] N. N. Ei, M. Alsenwi, Y. K. Tun, Z. Han, and C. S. Hong. Energy-efficient resource allocation in multi-uav-assisted two-stage edge computing for beyond 5g networks. *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[31] M. Li, N. Cheng, J. Gao,, Y. Wang, L. Zhao, and X. Shen. Energy-efficient uav-assisted mobile edge computing: Resource allocation and trajectory optimization. *IEEE Transactions on Vehicular Technology*, 69(3):3424–3438, 2020.

[32] Z. Ning and et al. 5g-enabled uav-to-community offloading: Joint trajectory design and task scheduling. *IEEE Journal on Selected Areas in Communications*, 39(11):3306–3320, November 2021.

[33] H. Yang, J. Zhao, J. Nie, N. Kumar, K. Y. Lam, and Z. Xiong. Uav-assisted 5g/6g networks: Joint scheduling and resource allocation based on asynchronous reinforcement learning. In *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2021.

[34] Collaborative Computation Offloading and Resource Allocation in Multi-UAV-Assisted IoT Networks: A Deep Reinforcement Learning Approach. A. m. Seid and g. o. Boateng and s. Anokye and t. Kwantwi and g. Sun and g. Liu. *IEEE Internet of Things Journal*, 8(15):12203–12218, 2021.

[35] P. Luong, F. Gagnon, L. N. Tran, and F. Labeau. Deep reinforcement learning-based resource allocation in cooperative uav-assisted wireless networks. *IEEE Transactions on Wireless Communications*, 20(11):7610–7625, 2021.

[36] O. Kodheli, E. Lagunas, N. Maturo, S. K. Sharma, B. Shankar, J. F. M. Montoya, J. C. M Duncan, D. Spano, S. Chatzinotas, S. Kisseleff, and J. Querol. Satellite communications in the new space era: a survey and future challenges. *IEEE Communications Surveys and Tutorials*, 23(1):70–109, 2021.

[37] L. You, K. Li, J. Wang, X. Gao, X. Xia, and B. Ottersten. Massive mimo transmission for leo satellite communications. *IEEE Journal on Selected Areas in Communications*, 38(8):1851–1865, June 2020.

[38] B. Di, H. Zhang, L. Song, Y. Li, and G. Y. Li. Ultra-dense leo: integrating terrestrial-satellite networks into 5g and beyond for offloading. *IEEE Transactions on Wireless Communications*, 18 (1):47–62, January 2019.

[39] O. Kodheli, S. Andrenacci, N. Maturo, S. Chatzinotas, and F. Zimmer. Resource allocation approach for differential doppler reduction in nb-iot over leo satellite. In *9th Advanced Satellite Multimedia Systems Conference and the 15th Signal Processing for Space Communications Workshop (ASMS/SPSC)*, 2018.

[40] Y. Yin, C. Huang, D. F. Wu, S. Huang, M. Ashraf, and Q. Guo. Reinforcement learning-based routing algorithm in satellite-terrestrial integrated networks. *Wireless Communications and Mobile Computing*, 2021.

[41] D. Zhou, M. Sheng, Y. Wang, J. Li, and Z. Han. Machine learning-based resource allocation in satellite networks supporting internet of remote things. *IEEE Transactions on Wireless Communications*, 20(10):6606–6621, October 2021.

[42] C. Jiang and X. Zhu. Reinforcement learning based capacity management in multi-layer satellite networks. *IEEE Transactions on Wireless Communications*, 19(7):4685–4699, 2020.

[43] L. Lei, Y. Yuan, T. X. Vu, S. Chatzinotas, M. Minardi, and J. F. Mendoza Montoya. Dynamic-adaptive ai solutions for network slicing management in satellite-integrated b5g systems. *IEEE Network Magazine*, 2021.

[44] M. Hasan and E. Hossain. Distributed resource allocation in 5g cellular networks. *Towards 5G: Applications, requirements and candidate technologies*, pages 129–161, September 2014.

[45] K. B. Letaief, W. Chen, Y. Shi, J. Zhang, and Y. J. A. Zhang. The roadmap to 6g ai: empowered wireless networks. *IEEE communications magazine*, 2019.

[46] L. U. Khan, S. R. Pandey, N. H. Tran, W. Saad, Z. Han, M. N. Nguyen, and C. S. Hong. Federated learning for edge networks: Resource optimization and incentive mechanism. *IEEE Communications Magazine*, 58(10):88–93, 2020.

[47] R. Yu and P. Li. Toward resource-efficient federated learning in mobile edge computing. *IEEE Network*, 35(1):148–155, January 2021.

[48] V. D. Nguyen, S. K. Sharma, T. X. Vu, S. Chatzinotas, and B. Ottersten. Efficient federated learning algorithm for resource allocation in wireless iot networks. *IEEE Internet of Things Journal*, 8 (5):3394–3409, September 2020.

[49] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei. Energy efficient federated learning over wireless communication networks. *IEEE Transactions on Wireless Communications*, 20(3): 1935–1949, March 2021.

[50] Q. Zeng, Y. Du, K. Huang, and K. K. Leung. Energy-efficient radio resource allocation for federated edge learning. In *IEEE International Conference on Communications Workshops (ICC Workshops)*, June 2020.

[51] C. M. Bishop and N. M. Nasrabadi. *Pattern recognition and machine learning*. New York: springer, August 2006.

[52] A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, August 2004.

[53] I. A. Basheer and M. Hajmeer. Artificial neural networks: fundamentals, computing, design, and application. *Journal of microbiological methods*, 43(1):3–31, December 2000.

[54] S. Albawi, T. A. Mohammed, and S. Al-Zawi. Understanding of a convolutional neural network. In *International Conference on Engineering and Technology (ICET)*, 2017.

[55] A. Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, March 2020.

[56] M. E. Celebi and K. Aydin, editors. *Unsupervised learning algorithms*. Berlin: Springer International Publishing, April 2016.

[57] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. Cambridge, Massachusetts, USA: MIT Press, 2018.

[58] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In *Neural Information Processing Systems (NIPS)*, June 2014.

[59] S. Ilager S. Tuli, K. Ramamohanarao, and R. Buyya. Dynamic scheduling for stochastic edge-cloud computing environments using a3c learning and residual recurrent neural networks. *IEEE transactions on mobile computing*, August 2017.

[60] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, July 2017.

[61] D. L. Applegate, W. Cook, S. Dash, and D. G. Espinoza. Exact solutions to linear programming problems. *Operations Research Letters*, 35(6):693–699, 2007.

[62] R. D. Monteiro and I. Adler. Interior path following primal-dual algorithms. part ii: Convex quadratic programming. *Mathematical Programming*, 44(1):43–66, 1989.

[63] Garth P McCormick. Computability of global solutions to factorable nonconvex programs: Part i: Convex underestimating problems. *Mathematical programming*, 10(1):147–175, 1976.

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

[65] D. S. Chen, R. G. Batson, and Y. Dang. *Applied integer programming: modeling and solution*. John Wiley and Sons, September 2011.

[66] Y. Yang, Y. Yuan, A. Chatzimichailidis, R. J. van Sloun, L. Lei, and S. Chatzinotas. Proxsgd: Training structured neural networks under regularization and constraints. In *International Conference on Learning Representations(ICLR)*, September 2019.

[67] M. Agiwal, A. Roy, and N. Saxena. Next generation 5g wireless networks: A comprehensive survey. *IEEE Communications Surveys and Tutorials*, 18(3):1617–1655, 2016.

[68] V. Angelakis, A. Ephremides, Q. He, and D. Yuan. Minimum-time link scheduling for emptying wireless systems: solution characterization and algorithmic framework. *IEEE Transactions on Information Theory*, 60(2):1083–1100, February 2014.

[69] L. Lei, D. Yuan, C. K. Ho, and S. Sun. Optimal cell clustering and activation for energy saving in load-coupled wireless networks. *IEEE Transactions on Wireless Communications*, 14(11):6150–6163, November 2015.

[70] S. Lahoud, K. Khawam, S. Martin, G. Feng, Z. Liang, and J. Nasreddine. Energy-efficient joint scheduling and power control in multi-cell wireless networks. *IEEE Journal on Selected Areas in Communications*, 34(12):3409–3426, December 2016.

[71] G. D. Nguyen, S. Kompella, C. Kam, J. E. Wieselthier, and A. Ephremides. Minimum-energy link scheduling for emptying wireless networks. *13th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, pages 207–212, 2015.

[72] K. Rahimi Malekshan and W. Zhuang. Joint scheduling and transmission power control in wireless ad hoc networks. *IEEE Transactions on Wireless Communications*, 16(9):5982–5993, September 2017.

[73] L. Lei, L. You, Q. He, T. X. Vu, S. Chatzinotas, D. Yuan, and B. Ottersten. Learning-assisted optimization for energy-efficient scheduling in deadline-aware noma systems. *IEEE Transactions on Green Communications and Networking*, 2019.

[74] L. Lei, L. You, G. Dai, T. X. Vu, D. Yuan, and S. Chatzinotas. A deep learning approach for optimizing content delivering in cache-enabled hetnet. In *2017 International Symposium on Wireless Communication Systems (ISWCS)*, pages 449–453, 2017.

[75] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y. Liang, and D. I. Kim. Applications of deep reinforcement learning in communications and networking: A survey. *IEEE Communications Surveys and Tutorials*, 21(4):3133–3174, May 2019.

[76] L. Lei, T. X. Vu, L. You, S. Fowler, and D. Yuan. Efficient minimum-energy scheduling with machine-learning based predictions for multiuser miso systems. In *IEEE International Communication Conference (ICC)*, 2018.

[77] T. X. Vu, S. Chatzinotas, and B. Ottersten. Edge-caching wireless networks: Performance analysis and optimization. *IEEE Transaction on Wireless Communcation*, 17(7):2827–2839, April 2018.

[78] M. R. Gary and D. S. Johnson. *Computers and Intractability: A Guideto the Theory of NP-Completeness*. W. H. Freeman, 1979.

[79] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[80] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Conference on Neural Information Processing Systems (NIPS)*, 2012.

[81] M. Mozaffari, W. Saad, M. Bennis, Y. H. Nam, and M. Debbah. A tutorial on uavs for wireless networks: Applications, challenges, and open problems. *IEEE Communications Surveys and Tutorials*, 21(3):2334–2360, 2019.

[82] H. D. Tran, T. X. Vu, S. Chatzinotas, S. Shahbazpanahi, and B. OTTERSTEN. Coarse trajectory design for energy minimization in uav-enabled wireless communications with latency constraints. *IEEE Transactions on Vehicular Technology*, 2020.

[83] S. Ahmed, M. Z. Chowdhury, and Y. M. Jang. Energy-efficient uav-to-user scheduling to maximize throughput in wireless networks. *IEEE Access*, 8:21215–21225, 2020.

[84] J. Zhang, Y. Zeng, and R. Zhang. Spectrum and energy efficiency maximization in uav-enabled mobile relaying. In *2017 IEEE International Conference on Communications (ICC)*, pages 1–6, 2017.

[85] Y. Zeng, J. Xu, and R. Zhang. Energy minimization for wireless communication with rotary-wing uav. *IEEE Transactions on Wireless Communications*, 18(4):2329–2345, 2019.

[86] S. Zhu, K. Yang, J. Ouyang, and Y. Du. Cooperative beamforming for uav-assisted cognitive relay networks with partial channel state information. In *2018 IEEE 4th International Conference on Computer and Communications (ICCC)*, pages 158–162, 2018.

[87] Q. Song and F. Zheng. Energy efficient multi-antenna uav-enabled mobile relay. *China Communications*, 15(5):41–50, 2018.

[88] F. Zhou, Y. Wu, R. Q. Hu, and Y. Qian. Computation rate maximization in uav-enabled wireless-powered mobile-edge computing systems. *IEEE Journal on Selected Areas in Communications*, 36(9):1927–1941, 2018.

[89] F. Zhou, N. C. Beaulieu, J. Cheng, Z. Chu, and Y. Wang. Robust max–min fairness resource allocation in sensing-based wideband cognitive radio with swipt: Imperfect channel sensing. *IEEE Systems Journal*, 12(3):2361–2372, 2018.

[90] Z. Chang, L. Lei, Z. Zhou, S. Mao, and T. Ristaniemi. Learn to cache: Machine learning for network edge caching in the big data era. *IEEE Wireless Communications*, 25(3):28–35, June 2018.

[91] M. Shin, J. Kim, and M. Levorato. Auction-based charging scheduling with deep learning framework for multi-drone networks. *IEEE Transactions on Vehicular Technology*, 68(5):4235–4248, 2019.

[92] Y. Hsu and R. Gau. Reinforcement learning-based collision avoidance and optimal trajectory planning in uav communication networks. *IEEE Transactions on Mobile Computing*, June 2020.

[93] W. Liu, P. Si, E. Sun, M. Li, C. Fang, and Y. Zhang. Green mobility management in uav-assisted iot based on dueling dqn. In *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019.

[94] J. He, M. Ostendorf, X. He, J. Chen, J. Gao, L. Li, and L. Deng. Deep reinforcement learning with a combinatorial action space for predicting popular reddit threads. *arXiv preprint arXiv:1606.03667*, 2016.

[95] V. Konda and J. Tsitsiklis. Actor-critic algorithms. In *Advances in neural information processing systems*, 2000.

[96] C. H. Liu, Z. Chen, J. Tang, J. Xu, and C. Piao. Energy-efficient uav control for effective and fair communication coverage: A deep reinforcement learning approach. *IEEE Journal on Selected Areas in Communications*, 36(9):2059–2070, 2018.

[97] C. H. Liu, Z. Dai, Y. Zhao, J. Crowcroft, D. O. Wu, and K. Leung. Distributed and energy-efficient mobile crowdsensing with charging stations by deep reinforcement learning. *IEEE Transactions on Mobile Computing*, pages 1–1, 2019.

[98] Y. Cao, L. Zhang, and Y. Liang. Deep reinforcement learning for channel and power allocation in uav-enabled iot systems. In *2019 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2019.

[99] Y. Yuan, L. Lei, T. X. Vu, S. Chatzinotas, and B. Ottersten. Actor-critic deep reinforcement learning forenergy minimization in uav-aided networks. In *2020 European Conference on Networks and Communications (EuCNC)*, 2020.

[100] R. Ding, F. Gao, and X. Shen. 3d uav trajectory design and frequency band allocation for energy-efficient and fair communication: A deep reinforcement learning approach. *IEEE Transactions on Wireless Communications (Early Access)*, August 2020.

[101] W. Shi, J. Li, H. Wu, C. Zhou, N. Cheng, and X. Shen. Drone-cell trajectory planning and resource allocation for highly mobile networks: A hierarchical drl approach. *IEEE Internet of Things Journal (Early Access)*, August 2020.

[102] S. Zhang, Y. Zeng, and R. Zhang. Cellular-enabled uav communication: A connectivity-constrained trajectory optimization perspective. *IEEE Transactions on Communications*, 67(3): 2580–2604, 2019.

[103] S. Yin, L. Li, and F. R. Yu. Resource allocation and basestation placement in downlink cellular networks assisted by multiple wireless powered uavs. *IEEE Transactions on Vehicular Technology*, 69(2):2171–2184, 2020.

[104] F. Iannello and O. Simeone. On the optimal scheduling of independent, symmetric and time-sensitive tasks. *IEEE Transactions on Automatic Control*, 58(9):2421–2425, 2013.

[105] Y. Yuan, T. X. Vu, L. Lei, S. Chatzinotas, and B. Ottersten. Joint user grouping and power allocation for miso systems: Learning to schedule. In *2019 27th European Signal Processing Conference (EUSIPCO)*, pages 1–5, 2019.

[106] L. Lei, Y. Yuan, T. X. Vu, S. Chatzinotas, and B. Ottersten. Learning-based resource allocation: Efficient content delivery enabled by convolutional neural network. In *IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2019.

[107] E. Yanmaz, R. Kuschnig, and C. Bettstetter. Channel measurements over 802.11a-based uav-to-ground links. In *IEEE Global Communications Conference Workshops*, pages 1280–1284, December 2011.

[108] C. You and R. Zhang. 3d trajectory optimization in rician fading for uav-enabled data harvesting. *IEEE Transactions on Wireless Communications*, 18(6):3192–3207, 2019.

[109] C. Yan, L. Fu, J. Zhang, and J. Wang. A comprehensive survey on uav communication channel modeling. *IEEE Access*, 4:107769–107792, August 2019.

[110] Q. Pan, S. Liu, M. Xu, and C. Jia. Finite-state markov model for the aeronautical channel. In *IEEE International Conference on Wireless Communications, Networking and Mobile Computing*, September 2009.

[111] J. Yang, P. Liu, and H. Mao. Model and simulation of narrowband ground-to-air fading channel based on markov process. In *International Conference on Network Computing and Information Security*, July 2011.

[112] Y. Zhou, J. Li, L. Lamont, and CA. Rabbath. A markov-based packet dropout model for uav wireless communications. *Journal of Communications*, 7(6):418–426, June 2012.

[113] H. Wang and N. Moayeri. Finite-state markov channel-a useful model for radio communication channels. *IEEE Transactions on Vehicular Technology*, 44(1):163–171, 1995.

[114] Antonio Filippone. *Flight performance of fixed and rotary wing aircraft*. Elsevier, 2006.

[115] M. Hifi, M. Michrafy, and A. Sbihi. Heuristic algorithms for the multiple-choice multidimensional knapsack problem. *Journal of the Operational Research Society*, 55(12):1323–1332, December 2004.

[116] J. Guillot, D. R. Leal, C. R. Algarín, and I. Oliveros. Search for global maximal in multimodal functions by applying numerical optimization algorithms: A comparison between golden section and simulated annealing. *Computation*, 7(3), 2019.

[117] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[118] Liang Wang, Peiqiu Huang, Kezhi Wang, Guopeng Zhang, Lei Zhang, Nauman Aslam, and Kun Yang. Rl-based user association and resource allocation for multi-uav enabled mec. In *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, pages 741–746. IEEE, 2019.

[119] Y. Lu, H. Lu, L. Cao, F. Wu, and D. Zhu. Learning deterministic policy with target for power control in wireless networks. In *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018.

[120] G. Dulac-Arnold, R. Evans, H. van Hasselt, P. Sunehag, T. Lillicrap, J. Hunt, T. Mann, T. Weber, T. Degris, and B. Coppin. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*, 2015.

[121] Y. Wei, F. R. Yu, M. Song, and Z. Han. User scheduling and resource allocation in hetnets with hybrid energy supply: an actor-critic reinforcement learning approach. *IEEE Transactions on Wireless Communications*, 17(1):680–692, November 2017.

[122] N. Yang, H. Zhang, K. Long, H. Hsieh, and J. Liu. Deep neural network for resource management in noma networks. *IEEE Transactions on Vehicular Technology*, 69(1):876–886, January 2020.

[123] Taesang Yoo and A. Goldsmith. On the optimality of multiantenna broadcast scheduling using zero-forcing beamforming. *IEEE Journal on Selected Areas in Communications*, 24(3):528–541, 2006.

[124] Y. He, Z. Zhang, F. R. Yu, N. Zhao, H. Yin, V. C. M. Leung, and Y. Zhang. Deep-reinforcement-learning-based optimization for cache-enabled opportunistic interference alignment wireless networks. *IEEE Transactions on Vehicular Technology*, 66(11):10433–10445, 2017.

[125] Y. Li, E. Pateromichelakis, N. Vucic, J. Luo, W. Xu, and G. Caire. Radio resource management considerations for 5g millimeter wave backhaul and access networks. *IEEE Communication Magazine*, 55(6):86–92, June 2017.

[126] Y. Li, N. Deng, and W. Zhou. A hierarchical approach to resource allocation in extensible multi-layer leo-mss. *IEEE Access*, 8:18522–18537, January 2020.

[127] S. Wang Y. Li, Q. Wang, M. Su, and W. Zhou. Dynamic downlink resource allocation based on imperfect estimation in leo-hap cognitive system. In *IEEE International Conference on Wireless Communications and Signal Processing (WCSP)*, October 2019.

[128] J. H. Lee, J. Park, M. Bennis, and Y. C. Ko. Integrating leo satellite and uav relaying via reinforcement learning for non-terrestrial networks. In *IEEE Global Communications Conference (GLOBECOM)*, December 2020.

[129] S. He, T. Wang, and S. Wang. Load-aware satellite handover strategy based on multi-agent reinforcement learning. In *IEEE Global Communications Conference (GLOBCOM)*, December 2020.

[130] B. Deng, C. Jiang, H. Yao, S. Guo, and S. Zhao. The next generation heterogeneous satellite communication networks: integration of resource management and deep reinforcement learning. *IEEE Wireless Communications*, 27(2):105–111, April 2020.

[131] Y. Shen, Y. Shi, J. Zhang, and K. B. Letaief. Lorm: learning to optimize for resource management in wireless networks with few training samples. *IEEE Transactions on Wireless Communications*, 19(1):665–679, January 2020.

[132] O. Simeone, S. Park, and J. Kang. From learning to meta-learning: reduced training overhead and complexity for communication systems. *6G Wireless Summit (6G SUMMIT)*, pages 1–5, 2020.

[133] H. Sun, W. Pu, M. Zhu, X. Fu, T. H. Chang, and M. Hong. Learning to continuously optimize wireless resource in episodically dynamic environment. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4945–4949, 2021.

[134] K. Javed and M. White. Meta-learning representations for continual learning,. In *Neural Information Processing Systems (NIPS)*, pages 1820–1830, 2019.

[135] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning (ICML)*, pages 1126–1135, July 2017.

[136] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International Conference on Machine Learning (ICML)*, page 5331–5340, 2019.

[137] F. Sung, L. Zhang, T. Xiang, T. Hospedales, and Y. Yang. Learning to learn: meta-critic networks for sample efficient learning. *arXiv preprint arXiv:1706.09529*, June 2017.

[138] H. Wu, Z. Zhang, C. Jiao, C. Li, and T. Q. S. Quek. Learn to sense: a meta-learning-based sensing and fusion framework for wireless sensor networks. *IEEE Internet of Things Journal*, 6 (5):8215–8227, October 2019.

[139] S. Park, O. Simeone, and J. Kang. Meta-learning to communicate: fast end-to-end training for fading channels. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5075–5079, 2020.

[140] 3GPP TR 23.737. Study on architecture aspects for using satellite access in 5g (release 17). Technical report, 2019.

[141] M. Cheng, J. B. Wang, J. Cheng, J. Y. Wang, and M. Lin. Joint scheduling and precoding for mmwave and sub-6ghz dual-mode networks. *IEEE Transactions on Vehicular Technology*, 69 (11):13098–13111, November 2020.

[142] X. Fang, W. Feng, T. Wei, Y. Chen, N. Ge, and C. X. Wang. 5g embraces satellites for 6g ubiquitous iot: Basic models for integrated satellite terrestrial networks. *IEEE Internet of Things Journal*, 8(18):14399–14417, March 2021.

[143] X. Zhu, C. Jiang, L. Kuang, N. Ge, and J. Lu. Energy efficient resource allocation in cloud based integrated terrestrial-satellite networks. In *IEEE International Conference on Communications (ICC)*, May 2018.

[144] W. Wang, T. Chen, R. Ding, G. Seco-Granados, L. You, and X. Gao. Location-based timing advance estimation for 5g integrated leo satellite communications. *IEEE Transactions on Vehicular Technology*, 70(6):6002–6017, June 2021.

[145] A. Alsharoa and M. S. Alouini. Improvement of the global connectivity using integrated satellite-airborne-terrestrial networks with resource optimization. *IEEE Transactions on Wireless Communications*, 19(8):5088–5100, April 2020.

[146] 3GPP TSG RAN WG1. Doppler compensation, uplink timing advance and random access in ntn (r1-1906087). Technical report, May 2019.

[147] K. Guo, M. Lin, B. Zhang, J. B. Wang, Y. Wu, W. P. Zhu, and J. Cheng. Performance analysis of hybrid satellite-terrestrial cooperative networks with relay selection. *IEEE Transactions on Vehicular Technology*, 69(8):9053–9067, August 2020.

[148] Digital video broadcasting (dvb); implementation guidelines for the second generation system for broadcasting, interactive services, news gathering and other broadband satellite applications; part 2: S2 extensions (dvb-s2x). Technical report, DVB Document A171-2, April 2020.

[149] Study on lte-based 5g terrestrial broadcast (release 16). Technical report, 3GPP TR 36.776.

[150] C. H. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Mineola, NY, USA: Dover, 1998.

[151] S. Boyd, N. Parikh, and E. Chu. *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Hanover, MA, USA: Now Publishers Inc., 2011.

[152] T. Lin, S. Ma, and S. Zhang. Iteration complexity analysis of multi-block admm for a family of convex minimization without strong convexity. *Journal of Scientific Computing*, 69(1):52–81, October 2016.

[153] Y. Yuan, L. Lei, T. X. Vu, S. Chatzinotas, S. Sun, and B. Ottersten. Actor-critic learning-based energy optimization for uav access and backhaul networks. *EURASIP Journal on Wireless Communications and Networking*, 1:1–27, December 2021.

[154] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang. Learning under concept drift: a review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12):2346–2363, December 2019.

[155] J. Ye and H. Gharavi. Deep reinforcement learning-assisted energy harvesting wireless networks. *IEEE Transactions on Green Communications and Networking*, 5(2):990–1002, June 2021.

[156] K. He and J. Sun. Convolutional neural networks at constrained time cost. In *IEEE conference on computer vision and pattern recognition (CVPR)*, pages 5353–5360, 2015.

[157] H. Sak, A. W. Senior, and F. Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *International Speech Communication Association (ISCA)*, 2014.

[158] Y. Yuan, L. Lei, T. X. Vu, S. Chatzinotas, S. Sun, and B. Ottersten. Energy minimization in uav-aided networks: actor-critic learning for constrained scheduling optimization. *IEEE Transactions on Vehicular Technology*, 70(5):5028–5042, May 2021.

[159] O. Popescu. Power budgets for cubesat radios to support ground communications and inter-satellite links. *IEEE Access*, 5:12618–12625, 2017.

[160] Z. Jiang, S. Chen, S. Zhou, and Z. Niu. Joint user scheduling and beam selection optimization for beam-based massive mimo downlinks. *IEEE Transactions on Wireless Communications*, 17(4): 2190–2204, January 2018.

[161] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah. Artificial neural networks-based machine learning for wireless networks: A tutorial. *IEEE Communications Surveys and Tutorials*, 21(4): 3039–3071, July 2019.

[162] Y. Sun, M. Peng, Y. Zhou, Y. Huang, and S. Mao. Application of machine learning in wireless networks: Key techniques and open issues. *IEEE Communications Surveys and Tutorials*, 21(4): 3072–3108, July 2019.

[163] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, H. B. McMahan, and O. Van. Towards federated learning at scale: System design. In *System Machine Learning Conference*, 2019.

[164] J. Konečný, H. B. McMahan F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon. Federated learning: strategies for improving communication efficiency. In *arXiv preprint arXiv:1610.05492*, October 2016.

[165] K. Thonglek, K. Takahashi, K. Ichikawa, H. Iida, and C. Nakasan. Federated learning of neural network models with heterogeneous structures. In *IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 735–740, December 2020.

[166] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas. Communication-efficient learning of deep networks from decentralized data. *International Conference on Artificial Intelligence and Statistics (AISTATS)*, April 2017.

[167] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani. Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization. In *International Conference on Artificial Intelligence and Statistics*, pages 2021–2031, June 2020.

[168] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith. Federated optimization in heterogeneous networks. In *arXiv preprint arXiv:1812.06127*, April 2020.

[169] S. Niknam, H. S. Dhillon, and J. H. Reed. Federated learning for wireless communications: Motivation, opportunities, and challenges. *IEEE Communications Magazine*, 58(6):46–51, July 2020.

[170] H. H. Yang, Z. Liu, T. Q. Quek, and H. V. Poor. Scheduling policies for federated learning in wireless networks. *IEEE Transactions on Communications*, 68(1):317–333, September 2019.

[171] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui. A joint learning and communications framework for federated learning over wireless networks. *IEEE Transactions on Wireless Communications*, 20(1):269–283, January 2021.

[172] T. T. Vu, D. T. Ngo, N. H. Tran, H. Q. Ngo, M. N. Dao, and R. H. Middleton. Cell-free massive mimo for wireless federated learning. *IEEE Transactions on Wireless Communications*, 19(10): 6377–6392, June 2020.

[173] C. T. Dinh, N. H. Tran, M. N. Nguyen, C. S. Hong, W. Bao, A. Y. Zomaya, and V. Gramoli. Federated learning over wireless networks: Convergence analysis and resource allocation. *IEEE/ACM Transactions on Networking*, 29(1):398–409, November 2020.

[174] H. Qin, R. Gong, X. Liu, X. Bai, J. Song, and N. Sebe. Binary neural networks: A survey. *Pattern Recognition*, 105, 2020.

[175] M. Li, T. Zhang, Y. Chen, and A. J. Smola. Efficient mini-batch training for stochastic optimization. In *ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 661–670, August 2014.

[176] T. D. Burd and R. W. Brodersen. Processor design for portable systems. *Journal of VLSI signal processing systems for signal, image and video technology*, 13(2):203–221, August 1996.

[177] I. Ahmad. *Handbook of Energy-Aware and Green Computing, Volume 1*. CRC Press, 2012.

[178] A. Haider and S. H. Hwang. Maximum transmit power for ue in an lte small cell uplink. *Electronics*, 8(7):796–821, July 2019.

[179] C. Coffrin, H. L. Hijazi, and P. Van Hentenryck. Strengthening convex relaxations with bound tightening for power network optimization. *Principles and Practice of Constraint Programming*, pages 39–57, 2015.

[180] S. Srinivas, A. Subramanya, and R. Venkatesh Babu. Training sparse neural networks. In *IEEE conference on computer vision and pattern recognition workshops*, pages 138–145, 2017.

[181] M. Courbariaux, Y. Bengio, and J. P. David. Binaryconnect: Training deep neural networks with binary weights during propagations. *Advances in neural information processing systems*, pages 3123–3131, 2015.

[182] L. Lei, E. Lagunas, Y. Yuan, M. G. Kibria, S. Chatzinotas, and B. Ottersten. Deep learning for beam hopping in multibeam satellite systems. In *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, pages 1–5, 2020.

[183] H. He, H. Daume III, and J. M. Eisner. Learning to search in branch and bound algorithms. In *Advances in neural information processing systems (NIPS)*, 2014.

[184] J. Cui, Y. Liu, and A. Nallanathan. Multi-agent reinforcement learning-based resource allocation for uav networks. *IEEE Transactions on Wireless Communications*, 19(2):729–743, February 2020.

[185] Y. S. Nasir and D. Guo. Multi-agent deep reinforcement learning for dynamic power allocation in wireless networks. *IEEE Journal on Selected Areas in Communications*, 37(10):2239–2250, 2019.

[186] A. Mostaani, O. Simeone, S. Chatzinotas, and B. Ottersten. Learning-based physical layer communications for multiagent collaboration. In *IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pages 1–6, 2019.

[187] M. S. Allahham, A. A. Abdellatif, N. Mhaisen, A. Mohamed, A. Erbad, and M. Guizani. Multi-agent reinforcement learning for network selection and resource allocation in heterogeneous multi-rat networks. *IEEE Transactions on Cognitive Communications and Networking*, March 2022.

[188] I. T. Jolliffe and J. Cadima. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065), April 2016.

[189] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), November 2008.

[190] M. Elsayed and M. Erol-Kantarci. Ai-enabled future wireless networks: Challenges, opportunities, and open issue. *IEEE Vehicular Technology Magazine*, 14(3):70–77, July 2019.

[191] O. Sallent, J. Perez-Romero, R. Ferrus, and R. Agusti. On radio access network slicing from a radio resource management perspective. *IEEE Wireless Communications*, 24(5):166–174, April 2017.

[192] D. Wu, Z. Zhang, S. Wu, J. Yang, and R. Wang. Biologically inspired resource allocation for network slices in 5g-enabled internet of things. *IEEE Internet of Things Journal*, 6(6):9266–9279, December 2018.

[193] B. Han, J. Lianghai, and H. D. Schotten. Slice as an evolutionary service: Genetic optimization for inter-slice resource management in 5g networks. *IEEE ACCESS*, 6:33137–33147, June 2018.

[194] B Han, D. Feng, and H. D. Schotten. A markov model of slice admission contro. *IEEE Networking Letters*, 1(1):2–5, October 2018.

[195] A. M. Elbir and K. V. Mishra. Deep learning design for joint antenna selection and hybrid beamforming in massive mimo. In *IEEE International Symposium on Antennas and Propagation and USNC-URSI Radio Science Meeting*, 2019.

[196] J. Guo, C. K. Wen, and S. Jin. Deep learning-based csi feedback for beamforming in single-and multi-cell massive mimo systems. *IEEE Journal on Selected Areas in Communications*, 39(7): 1872–1884, December 2020.

[197] I. Ahmed, M. K. Shahid, H. Khammari, and M. Masud. Machine learning based beam selection with low complexity hybrid beamforming design for 5g massive mimo systems. *IEEE Transactions on Green Communications and Networking*, 5(4):2160–2173, June 2021.

[198] A. Ortiz, A. Asadi, M. Engelhardt, A. Klein, and M. Hollick. Cbmos: Combinatorial bandit learning for mode selection and resource allocation in d2d systems. *IEEE Journal on Selected Areas in Communications*, 37(10):2225–2238, August 2019.

[199] J. Kim, J. Park, J. Noh, and S. Cho. Autonomous power allocation based on distributed deep learning for device-to-device communication underlaying cellular network. *IEEE Access*, 8:107853–107864, June 2020.

[200] X. Huang, M. Zeng, J. Fan, X. Fan, and X. Tang. A full duplex d2d clustering resource allocation scheme based on a k-means algorithm. *Wireless Communications and Mobile Computing*, May 2018.

[201] S. Hashima, B. M. ElHalawany, K. Hatano, K. Wu, and E. M. Mohamed. Leveraging machine-learning for d2d communications in 5g/beyond 5g networks. *Electronics*, 10(2):169, January 2021.

[202] H. Xing, O. Simeone, and S. Bi. Decentralized federated learning via sgd over wireless d2d networks. In *IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pages 1–5, May 2020.

[203] J. Nocedal and S. Wright. *Numerical optimization*. Springer Science and Business Media, 2006.

Appendices for Chapter 4

6.3 Proof of Lemma 1

We relax all the binary variables of $\mathbf{P1}$ to continuous variables $\hat{\mathbf{x}} = [\hat{x}_{1,1}, ..., \hat{x}_{g,t}, ..., \hat{x}_{G,T}]^{\mathrm{T}}$ and $\hat{\mathbf{y}} = [\hat{y}_1, ..., \hat{y}_k, ..., \hat{y}_K]^{\mathrm{T}}$, where $\hat{x}_{g,t}, \hat{y}_k \in [0, 1]$. The relaxed objective function is written by:

$$f(\hat{\mathbf{x}}, \hat{\mathbf{y}}) = \eta_0 \left(\mathbf{1}^{\mathrm{T}}\hat{\mathbf{y}} - K\right)^2 + \sum_{k\in\mathcal{K}} \eta_k \left(\mathbf{r}_k^{\mathrm{T}}\hat{\mathbf{x}} - D_k\right)^2, \tag{6.1}$$

where $\mathbf{1} = [1, ..., 1]^{\mathrm{T}}$ and $\mathbf{r}_k = [R_{k,1,1}, ..., R_{k,g,t}, ..., R_{k,G,T}]^{\mathrm{T}}$. We expand $\left(\mathbf{1}^{\mathrm{T}}\hat{\mathbf{y}} - K\right)^2$ and $\left(\mathbf{r}_k^{\mathrm{T}}\hat{\mathbf{x}} - D_k\right)^2$ as follows:

$$\left(\mathbf{1}^{\mathrm{T}}\hat{\mathbf{y}} - K\right)^2 = \hat{\mathbf{y}}^{\mathrm{T}}\mathbf{E}\hat{\mathbf{y}} - 2D_k\mathbf{1}_k^{\mathrm{T}}\hat{\mathbf{y}} + K, \tag{6.2}$$

$$\left(\mathbf{r}_k^{\mathrm{T}}\hat{\mathbf{x}} - D_k\right)^2 = \hat{\mathbf{x}}^{\mathrm{T}}\mathbf{R}\hat{\mathbf{x}} - 2D_k\mathbf{r}_k^{\mathrm{T}}\hat{\mathbf{x}} + D_k^2, \tag{6.3}$$

where $\mathbf{E}$ is an all-ones matrix and

$$\mathbf{R} = \begin{bmatrix} R_{k,1,1}^2 & R_{k,1,1}R_{k,1,2} & \cdots & R_{k,1,1}R_{k,G,T} \\ R_{k,1,2}R_{k,1,1} & R_{k,1,2}^2 & \cdots & R_{k,1,2}R_{k,G,T} \\ \vdots & \vdots & \ddots & \vdots \\ R_{k,G,T}R_{k,1,1} & R_{k,G,T}R_{k,1,2} & \cdots & R_{k,G,T}^2 \end{bmatrix}. \tag{6.4}$$

Referring to the theorem of quadratic programming, a quadratic function is convex when its corresponding real symmetric matrix is positive semi-definite [203]. According to the definition, $\mathbf{E}$ and $\mathbf{R}$ are positive semi-definite matrices since, given an arbitrary vector $\mathbf{v} = [v_1, ..., v_{G\times T}] \neq \mathbf{0}$, we can calculate $\mathbf{v}^T\mathbf{E}\mathbf{v} = \left(\mathbf{1}^{\mathrm{T}}\mathbf{v}\right)^2 \geq 0$ and $\mathbf{v}^T\mathbf{R}\mathbf{v} = \left(\mathbf{r}^{\mathrm{T}}\mathbf{v}\right)^2 \geq 0$ [203]. Therefore, $f(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ is convex as it is the summation of $K + 1$ convex functions. Besides, the constraints Eq. (4.14b)-(4.14d) are linear, hence the conclusion.

6.4 Proof of Lemma 2

The objective of the learning agent is to find a policy $\pi(a|s_t)$ that maximizes the expected accumulated reward $\sum_{t=0}^{T} \gamma^t r_t$. With r_t in Eq. (4.23), we expand $\sum_{t=0}^{T} \gamma^t r_t$ as:

$$\sum_{t=0}^{T} \gamma^t r_t = \sum_{k=0}^{K} \eta_k \left[\gamma\Delta_{k,0}^2 + \sum_{t=1}^{T}(\gamma^{t+1} - \gamma^t)\Delta_{k,t}^2 - \gamma^T\Delta_{k,T}^2\right]$$

$$\begin{aligned}
&\overset{\gamma=1}{=} \sum_{k=0}^{K} \eta_k \left(\Delta_{k,0}^2 - \Delta_{k,T}^2\right) \\
&= \sum_{k=0}^{K} \eta_k \Delta_{k,0}^2 - \eta_0 \left(\sum_{k=1}^{K} \mathbb{1}\left(b_{k,T} - D_k'\right) - K \right)^2 - \sum_{k=1}^{K} \eta_k \left(b_{k,T} - D_k\right)^2, \qquad (6.5)
\end{aligned}$$

where $b_{k,T} = \sum_{t=1}^{T} R_{k,a_t,t}$. Thus, we can obtain the optimal policy $a_t^* \sim \pi^*(a|s_t)$ by solving the following problem:

$$\max_{\pi(a|s_t)} -\mathbb{E}_{\pi(a|s_t)} \left[\eta_0 \left(\sum_{k=1}^{K} \mathbb{1} \left(\sum_{t=1}^{T} R_{k,a_t,t} - D_k' \right) - K \right)^2 - \sum_{k=1}^{K} \eta_k \left(\sum_{t=1}^{T} R_{k,a_t,t} - D_k \right)^2 \right], \qquad (6.6)$$

which is equivalent to the objective Eq. (5.20a), thus the conclusion.

6.5 Proof of Lemma 3

Denote $Q(s,a_1),...,Q(s,a_M)$ as random variables $X_1,...,X_M$, where $X_{m'} = Q(s,a_s^*)$ and $X_m \sim U(Q(s,a_s^*) - \kappa, Q(s,a_s^*) + \kappa),\ \forall m \neq m'$. Thus, $Q(s,a_w^*)$ can be expressed as a random variable $\Psi = \max\{X_1,...,X_M\}$. The cumulative distribution function of Ψ is expressed as:

$$\begin{aligned}
F_\Psi(\psi) =& \mathbb{P}[\Psi \leq \psi] = \mathbb{P}[\max\{X_1,...,X_M\} \leq \psi] \\
=& \mathbb{P}[X_1 \leq \psi]\mathbb{P}[X_2 \leq \psi]...\mathbb{P}[X_M \leq \psi] \\
=& F_{X_1}(\psi)F_{X_2}(\psi)...F_{X_M}(\psi) \qquad (6.7)
\end{aligned}$$

For $m \neq m'$, based on the cumulative distribution function of uniform distribution, we can derive:

$$F_{X_m}(\psi) = \frac{\psi - Q(s,a_s^*) + \kappa}{2\kappa},\ \psi \in [Q(s,a_s^*) - \kappa, Q(s,a_s^*) + \kappa]. \qquad (6.8)$$

For $m = m'$, as $X_{m'} = Q(s,a_s^*)$, the cumulative distribution function is:

$$F_{X_{m'}}(\psi) = \mathbb{P}[X_{m'} \leq \psi] = \begin{cases} 1, & \psi \geq Q(s,a_s^*), \\ 0, & \psi < Q(s,a_s^*). \end{cases} \qquad (6.9)$$

By substituting Eq. (6.8) and Eq. (6.9) into Eq. (6.7),

$$F_\Psi(\psi) = \begin{cases} \left(\frac{y - Q(s,a_s^*) + \kappa}{2\kappa}\right)^{M-1}, & \psi \in [Q(s,a_s^*), Q(s,a_s^*) + \kappa], \\ 0, & \psi \in [Q(s,a_s^*) - \kappa, Q(s,a_s^*)). \end{cases} \qquad (6.10)$$

Then, the probability density function of Ψ can be calculated by solving the first derivative:

$$\begin{aligned}
&f_\Psi(\psi) = [F_\Psi(\psi)]' \\
=& \begin{cases} \frac{1}{2^{M-1}}\delta(\psi - Q(s,a_s^*)), & \psi = Q(s,a_s^*), \\ \frac{M-1}{2\kappa}\left(\frac{\psi - Q + \kappa}{2\kappa}\right)^{M-2}, & Q(s,a_s^*) < \psi \leq Q(s,a_s^*) + \kappa, \\ 0, & \text{otherwise}, \end{cases} \qquad (6.11)
\end{aligned}$$

where $\delta(\cdot)$ is Dirac function. The expectation of Ψ is:

$$\begin{aligned}
&\mathbb{E}\left[\Psi\right] = \mathbb{E}\left[Q(s, a_w^*)\right] = \int_{Q(s,a_s^*)}^{Q(s,a_s^*)+\kappa} \psi f_\Psi(\psi) d\psi \\
=&\frac{1}{2^{M-1}} \int_{Q(s,a_s^*)^-}^{Q(s,a_s^*)^+} \psi \delta(y - Q(s, a_s^*)) dy + \int_{Q(s,a_s^*)^+}^{Q(s,a_s^*)+\kappa} \psi \frac{M-1}{2\kappa} \left(\frac{\psi - Q(s, a_s^*) + \kappa}{2\kappa} \right)^{M-2} d\psi \\
=&\frac{Q(s, a_s^*)}{2^{M-1}} + Q(s, a_s^*) + \kappa - \frac{2\kappa}{M} - \frac{Q(s, a_s^*)}{2^{M-1}} + \frac{\kappa}{M \cdot 2^{M-1}} \\
=&Q(s, a_s^*) + \kappa \left(1 - \frac{2(2^M - 1)}{M \cdot 2^M} \right).
\end{aligned} \tag{6.12}$$

Thus the conclusion.

Appendices for Chapter 5

6.6 Proof of Theorem 1

Based on the L-Lipschitz continuous gradient of $F(\mathbf{w})$ in assumption 3 and update rule $\mathbf{w}^{i+1} = \mathbf{w}^i + \epsilon(\bar{\mathbf{w}}^i - \mathbf{w}^i)$, where $\bar{\mathbf{w}}^i = \frac{1}{K}\sum_{k=1}^{K}\mathbf{w}_k^i$, we have:

$$
\begin{aligned}
& F(\mathbf{w}^{i+1}) - F(\mathbf{w}^i) \\
=& \mathbb{E}_k[f_k(\mathbf{w}^{i+1})] + \mathbb{E}_k[r_k(\mathbf{w}^{i+1})] - \mathbb{E}_k[f_k(\mathbf{w}^i)] - \mathbb{E}_k[r_k(\mathbf{w}^i)] \\
\leq& (\mathbf{w}^{i+1} - \mathbf{w}^i)^{\mathrm{T}}\mathbb{E}_k[\nabla f_k(\mathbf{w}^i)] + \frac{L}{2}\|\mathbf{w}^{i+1} - \mathbf{w}^i\|^2 + \mathbb{E}_k[r_k(\mathbf{w}^{i+1}) - r_k(\mathbf{w}^i)] \\
\leq& \epsilon\left[(\bar{\mathbf{w}}^i - \mathbf{w}^i)^{\mathrm{T}}\mathbb{E}_k[\nabla f_k(\mathbf{w}^i) + \frac{r_k(\mathbf{w}^{i+1}) - r_k(\mathbf{w}^i)}{\mathbf{w}^{i+1} - \mathbf{w}^i}]\right] + \frac{L\epsilon^2}{2}\|\bar{\mathbf{w}}^i - \mathbf{w}^i\|^2 \\
=& \epsilon\left[(\bar{\mathbf{w}}^i - \mathbf{w}^i)^{\mathrm{T}}\bar{\nabla}F(\mathbf{w}^i; \mathbf{w}^{i+1})\right] + \frac{L\epsilon^2}{2}\|\bar{\mathbf{w}}^i - \mathbf{w}^i\|^2.
\end{aligned} \tag{6.13}
$$

Then, with the assumption 5, we can further drive:

$$
\begin{aligned}
& (\bar{\mathbf{w}}^i - \mathbf{w}^i)^{\mathrm{T}}\bar{\nabla}F(\mathbf{w}^i; \mathbf{w}^{i+1}) \\
=& (\bar{\nabla}F(\mathbf{w}^i; \mathbf{w}^{i+1}) - \bar{\nabla}F(\mathbf{w}^i; \mathbf{w}^{i+1}) + \bar{\mathbf{w}}^i - \mathbf{w}^i)^{\mathrm{T}} \cdot \bar{\nabla}F(\mathbf{w}^i; \mathbf{w}^{i+1}) \\
\leq& \left(\|\bar{\nabla}F(\mathbf{w}^i; \mathbf{w}^{i+1})\| + \|\bar{\mathbf{w}}^i - \mathbf{w}^i\|\right)\|\bar{\nabla}F(\mathbf{w}^i; \mathbf{w}^{i+1})\| - \bar{\nabla}F(\mathbf{w}^i; \mathbf{w}^{i+1})^2 \\
\leq& (1 + \epsilon M)\|\bar{\mathbf{w}}^i - \mathbf{w}^i\| \cdot \|\bar{\nabla}F(\mathbf{w}^i; \mathbf{w}^{i+1})\| - \bar{\nabla}F(\mathbf{w}^i; \mathbf{w}^{i+1})^2
\end{aligned} \tag{6.14}
$$

In NSC-SGD, the edge devices minimize $F_k(\mathbf{w})$ at each global iteration, we denote the optimal solution at the i-th iteration is $\mathbf{w}_k^i$ such that $F_k(\mathbf{w}_k^i) \leq F_k(\mathbf{w}^i)$. Based the assumption 4, we have:

$$
\begin{aligned}
& F_k(\mathbf{w}_k^i) \leq F_k(\mathbf{w}^i) \\
\Rightarrow& f_k(\mathbf{w}_k^i) - f_k(\mathbf{w}^i) + r_k(\mathbf{w}_k^i) - r_k(\mathbf{w}^i) \leq 0 \\
\Rightarrow& \frac{\chi}{2}\|\mathbf{w}_k^i - \mathbf{w}^i\|^2 + (\mathbf{w}_k^i - \mathbf{w}^i)^{\mathrm{T}}\nabla f_k(\mathbf{w}^i) + r_k(\mathbf{w}_k^i) - r_k(\mathbf{w}^i) \leq 0 \\
\Rightarrow& \|\mathbf{w}_k^i - \mathbf{w}^i\|^2 \leq -\frac{2}{\chi}(\mathbf{w}_k^i - \mathbf{w}^i)^{\mathrm{T}}\bar{\nabla}F_k(\mathbf{w}^i; \mathbf{w}_k^i) \\
& \leq \frac{2}{\chi}\|\mathbf{w}_k^i - \mathbf{w}^i\|\|\bar{\nabla}F_k(\mathbf{w}^i; \mathbf{w}_k^i)\| \\
\Rightarrow& \|\mathbf{w}_k^i - \mathbf{w}^i\| \leq \frac{2}{\chi}\|\bar{\nabla}F_k(\mathbf{w}^i; \mathbf{w}_k^i)\|.
\end{aligned} \tag{6.15}
$$

Combing the (6.15) and assumption 6, we can derive:

$$\begin{aligned}\|\bar{\mathbf{w}}^i - \mathbf{w}^i\| \leq \mathbb{E}_k[\|\mathbf{w}_k^i - \mathbf{w}^i\|] \leq \frac{2}{\chi}\mathbb{E}_k[\bar{\nabla}F_k(\mathbf{w}^i;\mathbf{w}_k^i)] \\ \leq \frac{2}{\chi}\sqrt{\mathbb{E}_k[\bar{\nabla}F_k(\mathbf{w}^i;\mathbf{w}_k^i)^2]} \leq \frac{2S}{\chi}\|\bar{\nabla}F(\mathbf{w}^i;\mathbf{w}^{i+1})\|. \end{aligned} \tag{6.16}$$

Based on (6.13) and (6.16), we have:

$$F(\mathbf{w}^{i+1}) - F(\mathbf{w}^i) \leq \left(\frac{2S\epsilon(1+\epsilon M)}{\chi} + \frac{2\epsilon^2 S^2 L}{\chi^2} - \epsilon\right)\bar{\nabla}F(\mathbf{w}^i;\mathbf{w}^{i+1})^2. \tag{6.17}$$

We remark that (6.17) holds for a single learning cycle. However, in NSC-SGD, the selected subset $\mathcal{K}_i$ ($|\mathcal{K}_i| = K$) are random over the learning cycles. The real weight vector, denoted as $\hat{\mathbf{w}}^i$, can be regarded as a random variable. Thus, it is necessary to analyze the relationship between $F(\hat{\mathbf{w}}^i)$ and the expectation of $F(\hat{\mathbf{w}}^{i+1})$ over the learning cycles, i.e., $\mathbb{E}_{\mathcal{K}_i}[F(\hat{\mathbf{w}}^{i+1})]$. We assume that $\hat{\mathbf{w}}^{i+1}$ can be expressed as $\hat{\mathbf{w}}^{i+1} = \mathbf{w}^i + \epsilon(\tilde{\mathbf{w}}^i - \mathbf{w}^i)$, where $\tilde{\mathbf{w}}^i$ is also a random variable. Based on L_0-local Lipschitz continuity [168], we have:

$$\begin{aligned} F(\hat{\mathbf{w}}^{i+1}) \leq & F(\mathbf{w}^{i+1}) + L_0\|\hat{\mathbf{w}}^{i+1} - \mathbf{w}^{i+1}\| \\ = & F(\mathbf{w}^{i+1}) + L_0\epsilon\|\tilde{\mathbf{w}}^i - \bar{\mathbf{w}}^i\|, \end{aligned} \tag{6.18}$$

where L_0 is the local Lipschitz continuity constant, which satisfies

$$\begin{aligned} L_0 = & \|\bar{\nabla}F(\mathbf{w}^i;\mathbf{w}^{i+1}) + \frac{L\epsilon}{2}(\tilde{\mathbf{w}}^i - \bar{\mathbf{w}}^i)\| \\ \leq & \bar{\nabla}F(\mathbf{w}^i;\mathbf{w}^{i+1}) + \frac{L\epsilon}{2}\|\tilde{\mathbf{w}}^i - \bar{\mathbf{w}}^i\| \\ \leq & \bar{\nabla}F(\mathbf{w}^i;\mathbf{w}^{i+1}) + L\epsilon\max(\|\tilde{\mathbf{w}}^i - \mathbf{w}^i\|, \|\bar{\mathbf{w}}^i - \mathbf{w}^i\|) \\ \leq & \bar{\nabla}F(\mathbf{w}^i;\mathbf{w}^{i+1}) + L\epsilon(\|\tilde{\mathbf{w}}^i - \mathbf{w}^i\| + \|\bar{\mathbf{w}}^i - \mathbf{w}^i\|). \end{aligned} \tag{6.19}$$

Then, we take expectation for the left and right sides of (6.18).

$$\mathbb{E}_{\mathcal{K}_i}[F(\hat{\mathbf{w}}^i)] \leq F(\mathbf{w}^i) + Q_i, \tag{6.20}$$

where $Q_i = \mathbb{E}_{\mathcal{K}_i}[L_0\epsilon\|\tilde{\mathbf{w}}^i - \bar{\mathbf{w}}^i\|]$. By applying Eq. (6.19) into Q_i, we get:

$$\begin{aligned} Q_i \leq & \epsilon\mathbb{E}_{\mathcal{K}_i}\left[\bar{\nabla}F(\mathbf{w}^i;\mathbf{w}^{i+1}) + L\epsilon(\|\tilde{\mathbf{w}}^i - \mathbf{w}^i\| + \|\bar{\mathbf{w}}^i - \mathbf{w}^i\|)\right] \cdot \mathbb{E}_{\mathcal{K}_i}\left[\|\tilde{\mathbf{w}}^i - \bar{\mathbf{w}}^i\|\right] \\ \leq & \epsilon\left(\bar{\nabla}F(\mathbf{w}^i;\mathbf{w}^{i+1}) + L\epsilon\|\bar{\mathbf{w}}^i - \mathbf{w}^i\|\right)\mathbb{E}_{\mathcal{K}_i}\left[\|\tilde{\mathbf{w}}^i - \bar{\mathbf{w}}^i\|\right] + L\epsilon^2\mathbb{E}_{\mathcal{K}_i}\left[\|\tilde{\mathbf{w}}^i - \mathbf{w}^i\|\|\tilde{\mathbf{w}}^i - \bar{\mathbf{w}}^i\|\right] \\ \leq & \epsilon\left(\bar{\nabla}F(\mathbf{w}^i;\mathbf{w}^{i+1}) + L\epsilon\|\bar{\mathbf{w}}^i - \mathbf{w}^i\|\right)\mathbb{E}_{\mathcal{K}_i}\left[\|\tilde{\mathbf{w}}^i - \bar{\mathbf{w}}^i\|\right] \\ & + L\epsilon^2\left(\|\bar{\mathbf{w}}^i - \mathbf{w}^i\| + \mathbb{E}_{\mathcal{K}_i}[\|\tilde{\mathbf{w}}^i - \bar{\mathbf{w}}^i\|]\right)\mathbb{E}_{\mathcal{K}_i}\left[\|\tilde{\mathbf{w}}^i - \bar{\mathbf{w}}^i\|\right] \\ = & \epsilon\left(\bar{\nabla}F(\mathbf{w}^i;\mathbf{w}^{i+1}) + 2L\epsilon\|\bar{\mathbf{w}}^i - \mathbf{w}^i\|\right)\mathbb{E}_{\mathcal{K}_i}\left[\|\tilde{\mathbf{w}}^i - \bar{\mathbf{w}}^i\|\right] + L\epsilon^2\mathbb{E}_{\mathcal{K}_i}[\|\tilde{\mathbf{w}}^i - \bar{\mathbf{w}}^i\|^2] \end{aligned} \tag{6.21}$$

For $\mathbb{E}_{\mathcal{K}_i}[\|\tilde{\mathbf{w}}^i - \bar{\mathbf{w}}^i\|]$ and $\mathbb{E}_{\mathcal{K}_i}[\|\tilde{\mathbf{w}}^i - \bar{\mathbf{w}}^i\|^2]$ we can derive:

$$\mathbb{E}_{\mathcal{K}_i}[\|\tilde{\mathbf{w}}^i - \bar{\mathbf{w}}^i\|] \leq \sqrt{\mathbb{E}_{\mathcal{K}_i}[\|\tilde{\mathbf{w}}^i - \bar{\mathbf{w}}^i\|^2]}, \tag{6.22}$$

and,

$$\mathbb{E}_{\mathcal{K}_i}[\|\tilde{\mathbf{w}}^i - \bar{\mathbf{w}}^i\|^2] \leq \frac{1}{K}\left(\frac{1}{K}\sum_{k\in\mathcal{K}_i}\|\mathbf{w}_k^i - \bar{\mathbf{w}}^i\|^2\right)$$
$$\leq \frac{1}{K}\mathbb{E}_k[\|\mathbf{w}_k^i - \bar{\mathbf{w}}^i\|^2] \leq \frac{2}{K}\max(\mathbb{E}_k[\|\mathbf{w}_k^i - \mathbf{w}^i\|^2], \|\mathbb{E}_k[\mathbf{w}_k^i] - \mathbf{w}^i\|^2])$$
$$= \frac{2}{K}\mathbb{E}_k[\|\mathbf{w}_k^i - \mathbf{w}^i\|^2] \leq \frac{8S^2}{K\chi^2}\bar{\nabla}F(\mathbf{w}^i;\mathbf{w}^{i+1})^2. \tag{6.23}$$

By substituting (6.22) and (6.23) into (6.21), we get:

$$Q_t \leq \epsilon\left(1 + \frac{4LS\epsilon}{\chi}\right)\sqrt{\frac{8S^2}{K\chi^2}}\bar{\nabla}F(\mathbf{w}^i;\mathbf{w}^{i+1})^2 + \frac{8LS^2\epsilon^2}{K\chi^2}\bar{\nabla}F(\mathbf{w}^i;\mathbf{w}^{i+1})^2. \tag{6.24}$$

By combining (6.17), (6.20) and (6.24), the conclusion can be obtained:

$$\mathbb{E}_{\mathcal{K}_i}[F(\hat{\mathbf{w}}^{i+1})] - F(\mathbf{w}^i) \leq F(\mathbf{w}^{i+1}) - F(\mathbf{w}^i) + Q_i$$
$$\leq \left(\frac{2S\epsilon(1+\epsilon M)}{\chi} + \frac{2\epsilon^2S^2L}{\chi^2} - \epsilon + \frac{\epsilon\sqrt{8}S}{\sqrt{K}\chi} + \frac{8\sqrt{2}LS^2\epsilon^2}{\sqrt{K}\chi^2} + \frac{8LS^2\epsilon^2}{K\chi^2}\right)\bar{\nabla}F(\mathbf{w}^i;\mathbf{w}^{i+1})^2. \tag{6.25}$$