Network Flow Optimization Using Reinforcement Learning

Wu, Chien-Cheng

DOI (link to publication from Publisher):
10.54337/aau547738179

Publication date:
2023

Document Version
Publisher's PDF, also known as Version of record

NETWORK FLOW OPTIMIZATION USING REINFORCEMENT LEARNING

BY
CHIEN-CHENG WU

DISSERTATION SUBMITTED 2023

AALBORG UNIVERSITY
DENMARK

Network Flow Optimization Using Reinforcement Learning

Ph.D. Dissertation

Chien-Cheng Wu

Aalborg University
Department of Electronic Systems
Fredrik Bajers Vej 7B
DK-9220 Aalborg

Dissertation submitted:	April 17, 2023
PhD supervisor:	Prof. Cedomir Stefanovic Aalborg University
Assistant PhD supervisors:	Prof. Petar Popovski Aalborg University
	Prof. Zheng-Hua Tan Aalborg University
PhD committee:	Professor Preben E. Mogensen (chairman) Department of Electronic Systems Aalborg University, Denmark
	Associate Professor Lingjie Duan Singapore University of Technology and Design, Singapore
	Professor Daniel Enrique Lucani Roetter Department of Engineering Aarhus University, Denmark
PhD Series:	Technical Faculty of IT and Design, Aalborg University
Department:	Department of Electronic Systems

ISSN (online): 2446-1628
ISBN (online): 978-87-7573-720-8

Published by:
Aalborg University Press
Kroghstræde 3
DK – 9220 Aalborg Ø
Phone: +45 99407140
aauf@forlag.aau.dk
forlag.aau.dk

Printed in Denmark by Stibo Complete, 2023

Abstract

Machine Learning (ML) is widely used in solving optimization problems nowadays. However, applying standard-compatible ML techniques to telecom network optimization is still in its exploratory phase. This dissertation aims to address network flow (traffic) optimization problems in a telecommunication network, leveraging Reinforcement Learning (RL) as an ML technique to extract knowledge from observations within the network and approximate optimal solutions.

Optimal solutions in a telecommunication network involve achieving specific objectives while efficiently utilizing network resources. This dissertation proposes a feasible RL paradigm with efficient algorithms to optimize data flows in telecommunication networks in near real-time. Specifically, the first research work focuses on optimizing the uplink transmission under a 5G network, and the objectives of the optimization are minimizing the age of information (AoI) and maximizing network throughput. The trade-off between AoI minimization and network throughput maximization is formulated and solved using the proposed RL paradigm, employing the Proximal Policy Optimization (PPO) algorithm. Furthermore, the second research addresses the joint periodic and burst traffic scheduling problem and proposes an RL paradigm that outperforms classical methods without prior knowledge of arriving traffic. The results demonstrate a tangible solution to deal with the joint network traffic and obtain robust uplink scheduling policies.

Additionally, this dissertation explores applying RL in online network slice provisioning (ONSP) optimization. The ONSP optimization can be formulated as a Multi-Objective Integer Programming problem and solved by predicting the incoming traffic demand under an RL framework. The research result contributes to optimizing end-to-end NSP in a telecommunication network. The efficiency of the RL approach was shown via numerical simulations. Moreover, we evaluate the performance gain by the computational complexity and the competitive ratio. Finally, this dissertation presents the optimal solution and concludes an RL-augmented framework with a standard-compatible design for the network's control and user planes.

Resumé

Machine Learning (ML) er meget brugt til at løse optimeringsproblemer i dag. Anvendelsen af standardkompatible ML-teknikker til optimering af telekommunikationsnetværk er dog stadig i sin udforskningsfase. Denne afhandling har til formål at adressere netværksflow (trafik) optimeringsproblemer i et telekommunikationsnetværk ved at udnytte Reinforcement Learning (RL) som en ML-teknik til at udtrække viden fra observationer inden for netværket og tilnærme optimale løsninger.

Optimale løsninger i et telekommunikationsnetværk involverer opnåelse af specifikke mål, samtidig med at netværksressourcer udnyttes effektivt. Denne afhandling foreslår et gennemførligt RL-paradigme med effektive algoritmer til at optimere datastrømme i telekommunikationsnetværk i næsten realtid. Specifikt fokuserer det første forskningsarbejde på at minimere informationsalderen (AoI) og maksimere netværksgennemstrømningen i et fjernstyringsmiljø over et 5G-netværk. Afvejningen mellem AoI-minimering og netværksgennemstrømningsmaksimering er formuleret og løst ved hjælp af det foreslåede RL-paradigme, ved at anvende algoritmen Proximal Policy Optimization (PPO). Ydermere adresserer det andet forskningsarbejde det fælles periodiske og burst-trafikplanlægningsproblem og foreslår et RL-paradigme, der udkonkurrerer klassiske metoder uden forudgående kendskab til ankommende trafik. Resultaterne viser den håndgribelige optimering af radioadgangsnetværk i telekommunikation ved hjælp af RL med robusthed.

Derudover udforsker denne afhandling anvendelsen af RL i Online Network Slice Provisioning (ONSP) optimering. ONSP-optimeringen er et problem med Multi-Objective Integer Programming, og det kan løses ved at forudsige den indkommende trafikefterspørgsel under en RL-ramme. Forskningsresultatet bidrager til at optimere end-to-end NSP i et telekommunikationsnetværk. Vi bruger numeriske simuleringer til at verificere effektiviteten af RL-tilgangen. Desuden evaluerer vi præstationsgevinsten ud fra den beregningsmæssige kompleksitet og konkurrenceforholdet. Til sidst præsenterer denne afhandling den optimale løsning og afslutter et RL-augmented framework med et standardkompatibelt design for netværkets kontrol- og brugerplan.

Contents

Dissertation Details

Dissertation Title:	Network Flow Optimization Using Reinforcement Learning
Ph.D. Student:	Chien-Cheng Wu
Supervisors:	Prof. Cedomir Stefanovic, Aalborg University
	Prof. Petar Popovski, Aalborg University
	Prof. Zheng-Hua Tan, Aalborg University

The dissertation comprises three introductory chapters and the following articles.

[A] Chien-Cheng Wu, Petar Popovski, Zheng-Hua Tan, Cedomir Stefanovic, "Design of AoI-Aware 5G Uplink Scheduler Using Reinforcement Learning," *Proceeding of IEEE 4th 5G World Forum*, pp. 176–181, 2021.

[B] Chien-Cheng Wu, Zheng-Hua Tan, Cedomir Stefanovic, "AoI and Throughput Optimization for Hybrid Traffic in Cellular Uplink Using Reinforcement Learning," *2022 IEEE 94rd Vehicular Technology Conference Workshop*, pp. 1–6, 2022.

[C] Chien-Cheng Wu, Vasilis Friderikos, Cedomir Stefanovic, "Multi-Objective Provisioning of Network Slices using Deep Reinforcement Learning," *Arxiv*, pp. 1–15, 2022.

[D] Chien-Cheng Wu, "Online Network Slicing Provisioning with Deep Reinforcement Learning," *Arxiv*, pp. 1–31, 2023.

This dissertation was financially supported by the European Union's Marie Sklodowska-Curie Grant Agreement No. 813999, "WindMill", as part of the Horizon 2020 Program. WindMill is a collaborative European project organized by the H2020 Marie Skodowska-Curie Innovative Training Networks (ITNs). Chien-Cheng Wu contributed to this project as an Early Stage Researcher (ESR), working alongside 14 ESRs to achieve the project's goals.

Acknowledgements

This dissertation was finalized after surmounting numerous substantial challenges that went beyond the realm of scientific research. First and foremost, I am sincerely grateful to all those who have unwaveringly supported me throughout my PhD journey.

The unwavering assistance provided by my family has played a pivotal role in addressing various practical aspects, from university enrollment to financial aid. Furthermore, my friends' encouragement has given me the essential fortitude to persevere. Without the support of my family and friend, I could not complete this dissertation.

Next, I appreciate the committee members, industrial partners, professors, staff members, researchers, and supervisors who contributed to my PhD studies. Your guidance from Aalborg University, Aarhus University, European Union, IT University of Copenhagen, King's College London, Minister for Higher Education and Science, Technical University of Denmark, University of Copenhagen, and other very important institutions have laid the foundation for my PhD journey.

While space limitations prevent mentioning each individual by name, your considerations and suggestions during the most challenging periods will always be cherished. This acknowledgment serves as an initial expression of my permanent appreciation.

Chien-Cheng Wu
Aalborg University, April 17, 2023

Part I

Introductory Chapters

Chapter 1

Introduction

This chapter has four distinct sections. The first section discusses the characteristics and challenges of network flow optimization in detail. The subsequent section presents a brief introduction of the potential benefits and opportunities associated with using a learning-based approach for flow optimization in next-generation telecommunication networks. The third section presents the research objectives and methodologies used in this PhD research. At last, the fourth section summarizes the contributions of the doctoral research and outlines the structure of the dissertation.

1 Network Flows Optimization

Telecommunication networks serve an essential role in contemporary society. The significance of telecommunication networks is contingent upon enhancing network hardware and software functionality. Network software functions' improvement supports the core competency of telecommunication networks, enabling them to operate with increased speed, reduced power consumption, and boosted reliability. Such improvements facilitate the evolution of telecommunication networks from existing generations (e.g., 5G and 6G) to next-generation and future networks. A telecommunication network flow refers to the transmission of data or information between network nodes via interconnecting links. The departure node is called the sender, while the destination node is called the receiver. A network flow can be further classified into different types according to some specific properties. For example, the network flow can be unidirectional or bidirectional between the sender and receiver, depending on the nature of the transmission.

The next-generation networks include many types of network flows. Those divergent network flows often derive complex Quality of Service (QoS) requirements (for example, delay and packet loss). In particular, the networks'

operational complexity is high if the network operators consider maintaining an end-to-end QoS guarantee within all routing links betweem the sender and the receiver. The IEEE 802.1Q Standard [3] and ITU-T Standard [1] defined eight types of network flow to simplify the operation and preserve the high-speed, low-cost characteristics within the network. This PhD research further categorizes the eight network types into four fundamental classes: Non-Sensitive, Loss-Sensitive, Time-Sensitive, and Joint-Sensitive (i.e., time-sensitive and loss-sensitive) network flows.

In the modern digital telecommunication network, one network flow can be generated via packets transmitted with the same communication protocol and IP addresses (and ports) from the same source and destination. Therefore, a tuple with the five properties above can independently and identically represent the network flow. More properties, such as QoS constraints, can also be added to the tuple representation for a specific flow classification at the application level. Network flow optimization involves the management and control of different types of flow to improve network performance, minimize congestion, and ensure efficient resource utilization. The optimization research in this dissertation covers three primary tasks: bandwidth allocation, flow routing, and flow scheduling. Based on the above definition of network flow and optimization, this section organized the following four subsections for advanced discussion: Non-Sensitive, Loss-Sensitive, Time-Sensitive Joint-Sensitive. The last subsection then summarizes the optimality conditions in different flow scenarios.

1.1 Non-Sensitive Network Flows

This type of flow has the least constraining regarding packet loss and delay. For example, a network application of static web page browsing generates a typical non-sensitive network flow. However, the type has less necessary to be optimized because it is generally used for low-priority traffic or applications. The non-sensitive network applications include email or short messages.

1.2 Loss-Sensitive Network Flows

The need for more reliable transmissions over the next-generation networks has recently become more urgent. These flows prioritize minimizing data loss during transmission. Applications like file transfer and banking transactions require to transmit data over loss-sensitive network flows. In order to guarantee reliability over loss-sensitive flows, protocols like Transmission Control Protocol (TCP) use error correction or retransmission to recover the lost packets. Those recovery methods (e.g., Hybrid Automatic Repeat reQuest [11]) lead to inevitable delays on top of the existing transmission delays of the network flows. Hence, the effect of packet loss in network flows

can be transformed to its dual effect, which is the delay penalty in network flows.

1.3 Time-Sensitive Network Flows

Time-Sensitive (or Delay-Sensitive) network flows evolved to encompass the development of multimedia applications. These flows prioritize minimizing latency and delay. Time-sensitive flows focus on delivering data packets as quickly as possible, even at the cost of occasional data loss. Examples of time-sensitive flows include video conferencing, online gaming, and virtual reality. In order to ensure data delivery with low delay (latency), low packet loss, and low jitter (i.e., guaranteed upper bound), standardization organizations (e.g., 3GPP, NGMN, IEEE) agree on some advanced implementations for traffic-aware scheduling and network resource management [8] [27] [6]. Protocols like User Datagram Protocol (UDP), which do not include error-checking or retransmission mechanisms, are suitable for handling time-sensitive traffic. However, the aforementioned implementations focus only on partial network scenarios for improved solutions, and this leaves room for optimizations.

1.4 Joint-Sensitive Network Flows

Joint-sensitive flows are sensitive to both packet loss and delay. This type of flow focuses on providing low latency while maintaining data reliability. These flows require timely delivery of all packets to maintain their QoS performance. An important application using Joint-sensitive flows is URLLCs (Ultra-Reliable and Low Latency Communications, i.e., a category of communication with significantly stringent constraints in reliability and latency). Applications that require real-time communication and high data reliability, such as remote control, teleoperated vehicles, or remote surgery, typically use Joint-sensitive flows.

Table 1.1 summarizes the sensitivity and the typical applications in different network flow categories.

Category	**Latency Sensitivity**	**Packet-Loss Sensitivity**	**Typical Applications**
Non-sensitive	Tolerant	Tolerant	Email transfers
Loss-sensitive	Tolerant	Sensitive	File transfers
Time-sensitive	Sensitive	Tolerant	Video conferencing
Joint-sensitive	Sensitive	Sensitive	Remote control

Table 1.1: Summary of Network Flow Categories

1.5 Optimality Conditions

Next-generation communication networks strive to accommodate diverse network flows. However, as initially designed, the communication networks needed to distinguish between the priority of every traffic flow. Each network flow shall not be often treated identically, regardless of whether it is critical or not. For example, although the current network standards define nine different flow categories [7] [9], mapping a suitable number of categorizations for flow transmission is still an open question. The classical communication networks are ill-suited for supporting the optimal transmission of diverse flows (i.e., transmitting the maximized amount of flows by using minimal network resources). Consequently, the need for optimality has arisen to tackle the restrictions of prior design, and it is particularly enticing for shaping the next-generation networks.

Age-of-information (AoI) was defined as a latency-related performance index in [28] for evaluating time-sensitive network flows optimization results. For example, a general link scheduling problem in 5G communications aims to determine how many links that can simultaneously schedule to transmit flows. Or the link scheduler should decide how many network resources should be allocated at each link. The optimal scheduling policy usually determines the policy parameters using objectives such as the minimum AoI or maximum throughput for all the links. Moreover, applying Network Function Virtualization (NFV) [18] and Software Defined Networking (SDN) [23] concepts to telecommunication networks can construct flexible network flows management for optimization. For instance, network slicing leverages NFV and SDN to transmit time-sensitive traffic and provides flexibility in network resource allocation. Each slice reserves enough network resources to guarantee the desired delay or jitter requirements. Proper slice allocation leads to a potential opportunity to reach optimality in the network.

An additional argument can be made regarding the prediction of user behavior. The accurately predicted user behavior can reduce the minimal task achievement delay (latency) to zero and progressively decrease to less than zero. For example, in some remote control scenarios, a robot can preserve a certain amount of network resources via prediction and avoid the waiting time for just-in-time scheduling with no preservation. Achieving negative delay would involve allocating sufficient network resources in advance to circumvent waiting for resource scheduling. By accurately forecasting user behavior, next-generation networks could further optimize resource management, enhancing the overall performance of time-sensitive network flows.

2 Learning-Based Optimization Approach

Numerous network optimization problems exhibit complex objective or constraint functions. Those functions often include intricate properties such as high dimensionality, nonlinearity, discontinuity, undifferentiability, convexity, and multimodality. Consequently, traditional optimization techniques like gradient descent, dynamic programming, or linear programming face significant challenges when addressing these complex properties and solving the problems. For example, optimization algorithms that rely on gradient information as inputs may struggle with non-differentiable objective functions. As a result, these conventional optimization methods often fail to deliver optimal solutions for complex optimization problems [15].

To tackle such complex problems, researchers have proposed some learning-based optimization algorithms [24]. One category includes heuristic optimization techniques like the Social-spider optimization algorithm [14], ant colony optimization algorithm [16], Krill herd algorithm [17], genetic algorithm [20], artificial bee colony algorithm [21], particle swarm optimization algorithm [22], and Greywolf optimization algorithm [26]. But these algorithms require experts to transform the physical network operation into a mathematical model and fulfill the formulation of algorithms.

Besides, machine learning (ML) methods are popular in reducing the inherent complexity of network optimization problems. These ML approaches can make predictions about incoming network flows based on historical network traffic data [13]. For instance, reinforcement learning (RL), particularly deep reinforcement learning (DRL), has garnered significant interest due to its efficiency and robustness. However, some existing RL approaches consider only under an individual base station and are ill-suited to network-wide management. A feasible RL approach should be embedded in telecommunication networks as Fig. 1.1 and fit its legacy architecture or future evaluation.

3 Research Objectives and Methology

Several international standard bodies have established latency and reliability requirements for next-generation communication services [5] [2] [4]. These stringent requirements make the next-generation mobile networks totally different from the former generation. Furthermore, Ultra-Reliable Low-Latency Communication (URLLC) services have been perceived as the catalyst for mission-critical use cases, including industrial automation, tele-operated control, and consumer-oriented services such as virtual reality/augmented reality (VR/AR) gaming. Consequently, these evolving requirements prompt new research questions and challenges, a warranting investigation into diverse solutions tailored to various scenarios.

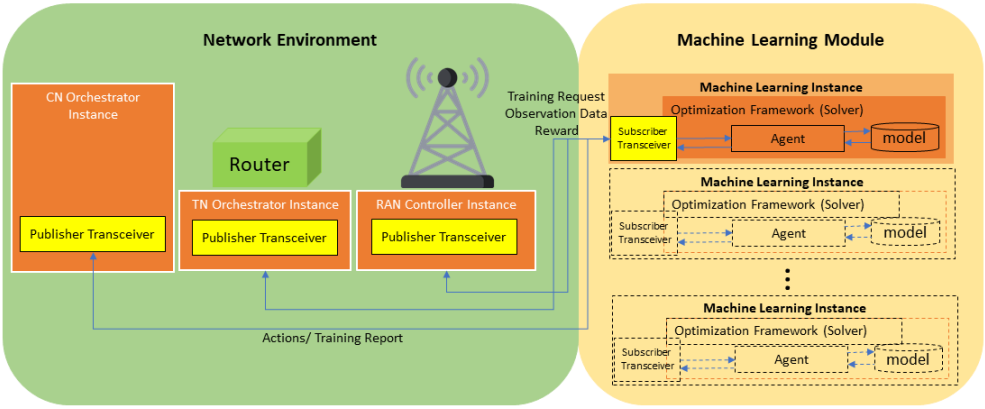

Fig. 1.1: An Example of Feasible RL-embedded Network Model

The PhD research objectives are explained in the following four paragraphs.

Objective 1: Investigate URLLC network flows in mobile networks The first research objective aims to devise a viable data model for URLLC traffic within mobile networks, which additionally functions as a surrogate model for forthcoming networks beyond the 6G. Owing to privacy and deployment concerns, obtaining URLLC traffic from actual networks poses significant challenges. Therefore, a network simulation platform (e.g. NS3 [19]) is used to construct flows and evaluate network performance.

Objective 2: Exploit the correlations in network traffic The second research objective investigates recurrent flow properties across diverse applications and network scenarios. In addition, the properties exhibit corresponding data and metadata flows between UEs and the network. The investigation lies in optimising control-plane procedures, such as radio resource management, connection establishment, and mobility management. Subsequently, the investigation is to devise methodologies for detecting and estimating patterns in the transmission of network traffic.

Objective 3: Design ML frameworks to optimize URLLC network traffic This objective focuses on the development and implementation of machine learning (ML) frameworks that can be incorporated into existing and future networks, to enhance the efficiency of URLLC network flows. The optimization targets various aspects, including signaling, power consumption, latency reduction, and reliability enhancement.

Objective 4: Evaluate the improvement of URLLC traffic flows The goal is to evaluate the advantages of the developed ML models in improving the URLLC network flows, particularly by reducing the transmission cost such as time and energy consumption. This evaluation process aims to demonstrate the practical benefits of implementing ML in modern communication systems.

Upon elucidating the PhD research objectives, the research methods employed in this doctoral study are summarized as follows:
This PhD investigation employs learning-based approaches to address the stochastic network environment, stemming from the dynamic and intricate nature in a practical mobile network. Traditional learn-based methods, however, prove to be ill-suited for mobile networks due to their considerable computational requirements, substantial memory consumption, and performance limitations. Consequently, the research methods adopted in this PhD study jointly consider optimization and learning technologies, diverging from prior art research.

The methods explore temporal and spatial correlations among flows within the mobile network. These correlations are utilized to optimize the latency and reliability of flows across the mobile network. To ensure the deployed ML methods reach a convergent state, a necessary procedure for the implementation conducted in this dissertation, it is crucial to ascertain that: (i) observations of network environments, such as received signal power, traffic load, and users' mobility intentions, are obtained to facilitate the appropriate training of the ML algorithm, and (ii) the number of network nodes remains bounded.

The employed methods predominantly rely on network simulations for system modeling and, potentially, on statistical data analysis. The supporting network simulation is divided into an ML module and a network module. The network module is either adapted from an open-source project, such as NS-3, or individually implemented by the author based on standard documents. Conversely, the ML module is implemented by the author by extending from the open-source ML library such as OpenAI Gym, PySyft, Pytorch, or Tensorflow [12] [25] [10]. The network and ML modules aim to execute on a virtual (or physical) computing platform with storage resources.

4 Dissertation Contribution

This PhD research developed models of network flows and techniques for optimizing network flow transmission. The optimization techniques include an optimization solver and a standard-compatible ML module for the existing mobile networks.

This dissertation concludes this PhD research and organizes the main contributions in the following four perspectives: (1) investigating user behavior for the user-plane network flows' construction and control-plane signaling in mobile networks, (2) simulating network flows based on the user behavior statistics models in mobile networks, (3) integrating the Machine Learning (ML) framework and optimization algorithms with a network simulator, and (4) optimizing the network resource management policy for network flows using the ML framework and optimization algorithm.

In the first perspective, this PhD research investigated user behaviors and derived the dynamic nature of those behaviors with a statistical model. The user-plane network flows in relation to user behavior models, such as the Poisson Pareto Burst Process model under a base station and a large-scale model under a nationwide mobile network, had been implemented for the network flow simulation in this PhD study. In particular, the large-scale model implementation demonstrates that ten percent of the users generate sixty percent of network traffic (i.e., within one standard deviation of the mean in a standard normal distribution). In addition, the control-plane signaling, such as network slice requests, had also been simulated to model the network traffic generation.

Regarding the second perspective, this PhD research includes a survey of cutting-edge ML techniques and results in an implementation of an RL module that can be embedded in existing and future mobile networks. The main achievement in this perspective is extending a network simulator with RL capabilities and establishing the Proximal Policy Optimization.

For the third perspective, an RL paradigm was constructed for network traffic optimization without additional data collection signaling. This RL paradigm is integrated with a software implementation of an optimization solver. The paradigm also focuses on the combination of training, testing, and deployment in a telecommunication network.

Finally, in the fourth perspective, the PhD research investigated RAN-scale optimization regarding AoI minimization and throughput maximization (**Paper A** and **Paper B**). Next, this PhD research studied end-to-end optimization for network slicing provisioning (**Paper C** and **Paper D**). The optimal provisioning has the efficiency of network flows in terms of the enhancements in network operation cost, SLA violation rate, and computation time. The two scenarios are investigated with statistical flow models and entail a deep RL module using the existing protocol signaling without extra message overheads. Both investigations also concluded that the network flow optimization using deep RL module has no closed-form representation. Hence, the evaluation was verified by numerical simulations and comparisons of competitive ratio and time complexity analysis.

5 Dissertation Outline

This dissertation consists of two parts. The first part presents the introductory chapters. This chapter provides an overview of the dissertation, including the backgrounds, objectives, methodologies, contributions, and outlines of the PhD research. The other two chapters of the first part are listed below.

Chapter 2 summarizes the prior arts and provides a detailed survey for the joint research between Machine Learning, Optimation, and Telecommunication Networks. The first session contains the analysis of telecommunication networks. This analysis studied and compared network flow characteristics in the latency and packet-loss aspects. The second session discusses the primary optimization methods that can be used to solve network flow optimization problems for transmission efficiency. The last session covers the ML techniques to enable the learning-based optimization approach in telecommunication networks. This session demonstrates a feasible Reinforcement Learning (RL) embedding in existing and potentially novel telecommunication networks (e.g., 5G and 6G).

Chapter 3 concludes the first part with a detailed discussion and future works in network flow optimizations using RL.

The second part of the dissertation is a collection of the public publications and organized as follows. The part plans to present the case study using RL approaches in the network flows optimization. Various scenarios with applications to support this PhD research are presented in the following papers:

Paper A In this research [31], the Age of Information (AoI) was defined the time passed from generating a packet by an user equipment (UE) to its reception by a base station under the 5G networks. This metric is crucial for evaluating the temporal information in modern wireless networks, particularly for time-sensitive applications. This work proposes a reinforcement learning (RL) method to implement an AoI-aware radio resource scheduler for UEs. The AoI-aware radio resource scheduler was designed for a remote control environment wherein multiple UEs transmit latency-sensitive measurements to a remote controller. The AoI in the overall transmission should be minimized. It can be formulated as a trade-off between maximizing the network throughput and minimizing the average AoI for all UEs. Motivated by the success of RL in addressing significant network optimization problems with low complexity, an RL-based method was developed to tackle the formulated problem. We also compare the performance of state-of-the-art scheduling algorithms with our solution. Simulation results demonstrate that the RL-

based method outperforms the state-of-the-art in minimizing the expected AoI while maintaining network throughput. This simulation highlights the potential of RL-based methods for AoI-aware radio resource scheduling in 5G networks, paving the way for more efficient and timely information delivery in time-critical applications.

Paper B The rapid proliferation of time-sensitive wireless applications leads to optimizing network radio scheduling algorithms. In this study [32], a scheduling problem for joint periodic and burst traffic transmission within a 5G network was considered, designing a reinforcement learning method to optimize the age of information and network throughput. The periodic traffic flow is generated at a fixed frequency, while the Poisson Pareto Burst Process governs the generation of burst traffic flow. We initially formulated the scheduling problem as a non-linear integer programming problem. Subsequently, we implementated a reinforcement learning framework using the Proximal Policy Optimization algorithm. At last, the numerical simulation indicates that the proposed reinforcement learning algorithm outperforms traditional solutions, even without an input of the arriving traffic. This finding underscores the potential of reinforcement learning in optimizing the age of information and throughput in 5G networks, enabling more effective radio scheduling for time-sensitive applications.

Paper C Network Slicing (NS) plays a vital role in efficiently facilitating diverse network applications in next-generation networks. However, the intricate Quality of Service (QoS) requirements and the heterogeneous nature of network services cause high complexity in Network Slice Provisioning (NSP) optimization. Traditional optimization methods struggle to address various low-latency and high-reliability requirements stemming from network applications. Therefore, an Online Network Slice Provisioning (ONSP) scenario was devised to enable NSP in a real-time manner [30]. Specifically, the ONSP problem was modeled as a Multi-Objective Integer Programming Optimization (MOIPO) problem. Next, to solve the MOIPO problem, an RL framework was proposed for predicting traffic demands. Simulation outcomes demonstrate the proposed method's effectiveness compared to state-of-the-art methods, as evidenced by a reduced Service-Level Agreement (SLA) violation rate and lower network operating costs. The research highlights the potential of leveraging PPO for real-time network slice provisioning in next-generation networks, improving the management of diverse QoS requirements and heterogeneous network services.

Paper D Network Slicing has emerged in next-generation networks to support mobile operators in isolated logical networks (slices) construction over the shared physical networks. While Network Slicing Provisioning (NSP)

provides fundamental support for slice management, solutions for optimal NSP are still at an initial stage. In order to deploy optimal NSP solutions to a network environment where a time horizon is considered, augmented learning capabilities are needed to make the NSP optimization adopt the dynamic nature of the real network. The paper [29] designed a fast reinforcement learning framework to solve the NSP optimization problem while achieving robustness against user traffic uncertainties and reserving effective provisioning. A theoretical analysis based on competitive ratios has been carried out to investigate the performance of algorithms. Furthermore, numerical simulations have been conducted to evaluate the designed reinforcement learning framework.

References

[1] "G.1010 : End-user multimedia qos categories," *ITU-T Publications*, pp. 1–18, 2002.

[2] "Minimum requirements related to technical performance for imt-2020 radio interface(s), standard rep. m.2410," 2017.

[3] "Ieee standard for local and metropolitan area network–bridges and bridged networks annex i," *IEEE Std 802.1Q-2018 (Revision of IEEE Std 802.1Q-2014)*, pp. 1–1993, 2018.

[4] "5g for connected industries and automation, 2nd ed., white paper," 2019.

[5] "3gpp tr 38.804: Study on new radio access technology; radio interface protocol aspects," 2020.

[6] "Ieee std 802.1as-2020: Timing and synchronization for time-sensitive applications," 2020.

[7] "3gpp ts 23.203; lte; policy and charging control architecture," 2021.

[8] "3gpp ts 24.539; 5g system (5gs); network to tsn translator (tt) protocol aspects; stage 3," 2021.

[9] "3gpp ts 24.501; non-access-stratum (nas) protocol for 5g system (5gs); stage 3," 2023.

[10] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "{TensorFlow}: a system for {Large-Scale} machine learning," in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 2016, pp. 265–283.

[11] S. Ahmadi, Ed., *5G NR Architecture, Technology, Implementation, and Operation of 3GPP New Radio Standards*. Academic Press, 2019.

[12] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[13] F. Bronzino, P. Schmitt, S. Ayoubi, H. Kim, R. Teixeira, and N. Feamster, "Traffic refinery: Cost-aware data representation for machine learning on network traffic," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 5, no. 3, dec 2021. [Online]. Available: https://doi.org/10.1145/3491052

[14] E. Cuevas, M. Cienfuegos, D. Zaldívar, and M. Pérez-Cisneros, "A swarm optimization algorithm inspired in the behavior of the social-spider," *Expert Systems with Applications*, vol. 40, no. 16, pp. 6374–6384, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417413003394

[15] B. Dogan and T. Ölmez, "A new metaheuristic for numerical function optimization: Vortex search algorithm," *Information Sciences*, vol. 293, pp. 125–145, 2015.

[16] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 28–39, 2006.

[17] A. H. Gandomi and A. H. Alavi, "Krill herd: A new bio-inspired optimization algorithm," *Communications in Nonlinear Science and Numerical Simulation*, vol. 17, no. 12, pp. 4831–4845, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1007570412002171

[18] H. Hawilo, A. Shami, M. Mirahmadi, and R. Asal, "Nfv: state of the art, challenges, and implementation in next generation mobile networks (vepc)," *IEEE Network*, vol. 28, no. 6, pp. 18–26, 2014.

[19] T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, and J. Kopena, "Network simulations with the ns-3 simulator," *SIGCOMM demonstration*, vol. 14, no. 14, p. 527, 2008.

[20] J. H. Holland, "Genetic algorithms and the optimal allocation of trials," *SIAM Journal on Computing*, vol. 2, no. 2, pp. 88–105, 1973.

[21] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm," *Journal of Global Optimization*, vol. 39, pp. 459–471, 2007.

[22] J. Kennedy, *Particle Swarm Optimization*. Boston, MA: Springer US, 2010, pp. 760–766. [Online]. Available: https://doi.org/10.1007/978-0-387-30164-8_630

[23] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Communications Magazine*, vol. 51, no. 2, pp. 114–119, 2013.

[24] M. J. Kochenderfer and T. A. Wheeler, *Algorithms for optimization*. Mit Press, 2019.

[25] D. Machalek, J. Tuttle, K. Andersson, and K. M. Powell, "Dynamic energy system modeling using hybrid physics-based and machine learning encoder-decoder models," *Energy and AI*, vol. 9, p. 100172, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S266654682200026X

[26] S. Mirjalili, S. Saremi, S. M. Mirjalili, and L. dos S. Coelho, "Multi-objective grey wolf optimizer: A novel algorithm for multi-criterion optimization," *Expert Systems with Applications*, vol. 47, pp. 106–119, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417415007435

[27] NGMN, "5g white paper 2," 2020.

[28] Y. Sun, E. Uysal-Biyikoglu, R. Yates, C. E. Koksal, and N. B. Shroff, "Update or wait: How to keep your data fresh," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*. IEEE Press, 2016, pp. 1–9.

[29] C.-C. Wu, "Online network slicing provisioning with deep reinforcement learning," 2023.

[30] C.-C. Wu, V. Friderikos, and C. Stefanovic, "Multi-objective provisioning of network slices using deep reinforcement learning," 2022.

[31] C.-C. Wu, P. Popovski, Z.-H. Tan, and . Stefanovi, "Design of aoi-aware 5g uplink scheduler using reinforcement learning," in *2021 IEEE 4th 5G World Forum (5GWF)*, 2021, pp. 176–181.

[32] C.-C. Wu, Z.-H. Tan, and . Stefanovi, "Aoi and throughput optimization for hybrid traffic in cellular uplink using reinforcement learning," in *2022 IEEE 95th Vehicular Technology Conference: (VTC2022-Spring)*, 2022, pp. 1–6.

Chapter 2

Machine Learning, Optimization and Telecommunication Networks

This chapter firstly discusses the historical background and emerging development of the mobile communication systems in Section 1. Next, Section 2 presents the fundamental principles for network flow optimization. In addition, Section 3 introduces some machine learning approaches in designing a framework for managing network flows. The proposed framework considers the heterogeneity in the mobile networks formed by a range of user equipment and base stations. At last, Section 4 summarizes the main contributions in this chapter.

1 Telecommunication Networks

Telecommunication networks have evolved tremendously over the last thirty years to satisfy increasing demands having high-reliability and low-latency requirements. Regarding next-generation telecommunication networks, a machine learning management component and three generic connectivity scenarios were introduced as in Fig. 2.1: Ultra-Reliable Low Latency Communications (URLLC), enhanced Mobile Broadband (eMBB), and massive Machine-Type Communication (mMTC). On top of next-generation communications, many novel applications, such as tele-operated robots, remote surgery, and vehicle-to-everything (V2X) network, all have stringent requirements for

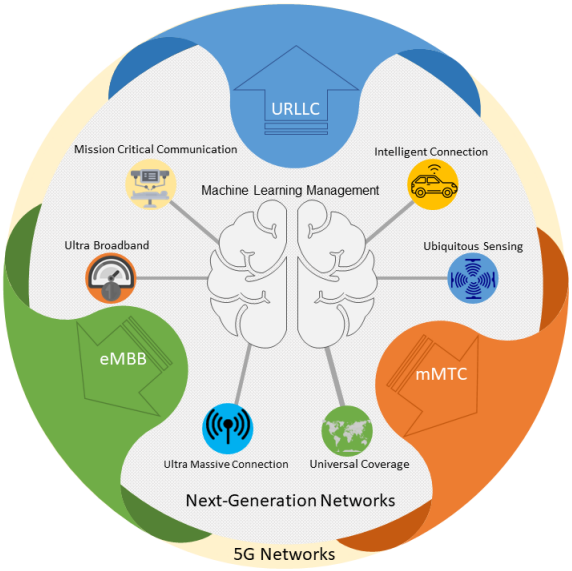

Fig. 2.1: Applications in Next-Generation Telecommunication Networks

transmission latency and reliability, which are difficult to be fulfilled by the legacy systems in a Heterogeneous Networks (HetNets) environment such as multi-mode and multi-RAT [5] [9]. 5G-ACIA describes various scenarios, aiming to satisfy some cyber-physical applications with high reliability and low latency requirements [2]. This research studied some typical scenarios and listed them as follows:

Scenario 1: Ultra Broadband Communication

This usage scenario extends the eMBB scenario of 5G. The typical applications are such as immersive XR and holographic communications. The scenario will require extremely high data rates but only using enough latency and system capacity. This scenario aims to cover all deployments from dense urban hotspots to rural areas.

Scenario 2: Ubiquitous Sensing

This usage scenario refers to the technologies that combine Sensing and Communication Systems to realize ubiquitous sensing. The typical usages are advanced localization, positioning, posture/gesture recognition, tracking, imaging, and mapping, applied to the use cases such as automatic construction, warehouse management, and automated driving. Integration of sensing with communication provides real-time interaction between virtual and physical worlds.

Scenario 3: Mission Critical Communication

This usage scenario applies to cases typically with very stringent transmis-

sion reliability and latency requirements, i.e., extending the URLLC of 5G to extreme URLLC. The typical use cases include full automation and remote control, remote teleoperation, robotics collaboration, autonomous driving, remote medical surgery, etc., envisioned for the Beyond 5G era. This set of use cases is characterized by situations where a failure of the communication service can have severe consequences for safety-related applications.

Scenario 4: Universal Coverage
This scenario tends to cover all humanity's footprints by providing universal coverage. For example, the typical use cases are supposed to provide essential MBB services everywhere people live. This usage scenario requires interworking between non-terrestrial networks and terrestrial cellular networks, such as Very Low Earth Orbit (VLEO) and High Altitude Platform Station (HAPS). In that way, this scenario can also support the rescue and recovery efforts in the event of natural disasters with disaster-resilient infrastructures.

Scenario 5: Ultra Massive Connection
This usage scenario extends the scenario of mMTC of 5G. The typical applications include remote meter reading, environmental monitoring, and connecting the massive amount of wearable devices, electronic devices or sensors with sporadic traffic in daily life. This usage scenario may also require supporting the massive connectivity simultaneously.

Scenario 6: Intelligent Connection
This usage scenario is characterized by incorporating AI-Native functionality into future beyond-5G networks and supports AI-powered applications in conjunction with the in-device and in-network AI capabilities. It will leverage local compute offload, distributed learning/inference and training of AI models performed jointly with many intelligent agents in the network. Typical applications include: Training and inference for collaborative robots. Distributed learning and inference for automated driving. Autonomous collaboration between devices with zero-touch. Another important aspect of this usage scenario is using AI/ML tools to optimize Beyond 5G systems in all network layers to improve performance and efficiency, such as the air interface and network itself.

The predominant design in next-generation telecommunication networks is the Heterogeneous Networks (HetNets) architecture [3], which provides a versatile environment for accommodating various cyber-physical applications. Numerous existing solutions have been used to enhance network resource management strategies in HetNets by utilizing deterministic algorithms and parameters [18], but the machine learning methods also start to be adopted [1].

In these next-generation networks, the overall architecture is divided into three layers: Core Network (CN), Transport Network (TN), and Radio Access Network (RAN). Fig. 2.2 illustrates the functional components and interfaces between CN, RAN, and TN. Several functional components within these lay-

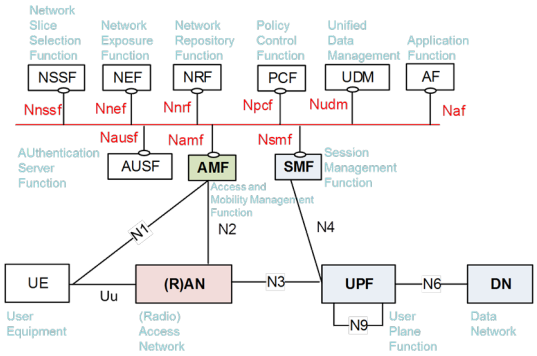

Fig. 2.2: Illustration of 5G CN functional components and interfaces with RAN and TN

ers present opportunities to implement optimization techniques aimed at improving network performance. However, a unified optimization framework that spans across different network layers is rarely documented in the existing literature.

The new network services with extreme latency and reliability demands have shown an urgent need for optimizing the entire network. The most crucial procedure in network flow optimization is to monitor the network status without generating redundant network traffics. Therefore, the ideal solution is leveraging the active devices in the telecommunication network to measure the network status continuously and only report some measurements back to the network orchestrator within standard protocols. These piggyback measurements can be applied to resource allocation and network flow optimization without additional communication signaling exchanges. Using piggyback measurements for flow optimization leads to efficient network flow optimization. This dissertation illustrates how classical and measurements-driven solutions can be applied in network flow optimization in the next two sections, Section 2 and Section 3 respectively.

2 Optimization Technologies

This section introduces the classical network optimization approaches while reserving the discussion on machine learning-specific optimization techniques for the subsequent section. Network optimization comprises a collection of technologies to improve network performance. Fig. 2.3 depicts the classical optimization technologies organized in a tree structure. The optimization tree presents the primary characteristics utilized to classify these optimization technologies. The optimization technologies can be broadly categorized into two groups: those that deal with uncertainty (stochastic) and those that

are deterministic. The examples of the relevant methods within each group are also presented.

First, stochastic optimization methods are used when there is randomness or uncertainty in the problem parameters or constraints. These methods help find the best solution considering the probabilities or likelihoods associated with data inputs. For instance, the Stochastic Gradient Descent is a popular optimization method that adjusts the model's parameters by minimizing an objective function that incorporates random data samples.

Second, deterministic optimization methods assume that all the problem parameters and constraints are known and fixed. These methods can be further divided into various subcategories, such as: a. Integer Programming: This method deals with optimization problems where a part of or all of the decision variables are required to be integers. Examples include 0-1 (binary) integer programming and mixed-integer programming, which combines float and integer variables. b. Linear Programming: This method is used for problems where the objective and constraint functions are linear expressions. It aims to find the optimal solution within the feasible region defined by the constraints, typically using methods like the Simplex algorithm or interior-point methods. c. Nonlinear Programming: This method addresses optimization problems with nonlinear objective functions or constraints. Nonlinear programming techniques can be further classified into unconstrained and constrained optimization methods. Common solution methods include gradient descent, Newton's method, and sequential quadratic programming. d. Mixed Integer Programming: This method combines elements of both integer programming and linear programming, where some variables are required to be integers, while others are continuous. Mixed Integer Linear Programming (MILP) and Mixed Integer Nonlinear Programming (MINLP) are classic methods of this technique. e. Constrained Optimization: This type of optimization method deals with problems where the decision variables are subject to certain constraints or limitations. This can include linear constraints, nonlinear constraints, or a mix of both. f. Multiobjective Optimization: This method is used when there are multiple objectives that need to be optimized simultaneously. The number of optimal solutions may lasrger than one, called Pareto Front that represent trade-offs between those objectives. Common techniques include the epsilon-constraint and weighted-sum methods, and evolutionary algorithms [13].

An optimization method constitutes a systematic process of discovering the optimal system design, subject to a set of constraints. Optimization problems are generally formulated as mathematical representations encompassing objective and constraint functions. A typical optimization problem can be expressed in (2.1):

$$\begin{aligned} &\text{minimize} \quad f(x) \\ &\text{subject to} \quad x \in X \end{aligned} \tag{2.1}$$

where $f(x)$ represents the objective function to be minimized, and the constraint $x \in X$ defines the feasible region for the decision variable x.

The optimal solution to an optimization problem is the objective function's global minima (in primary condition) or maxima (in dual condition). Employing an optimization algorithm facilitates the gradual enhancement of the design until either the objective function can no longer be improved (i.e., attaining the minimum or maximum) or the pre-allocated constraints have been exhausted.

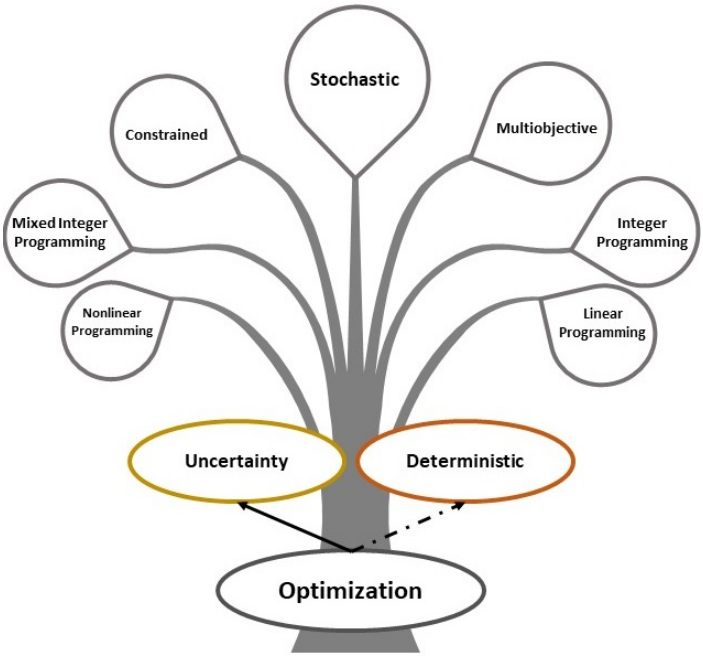

Fig. 2.3: Optimization Technologies

Network optimization organizes a series of technologies to improve network performance. Because of new users and applications, the network flows are inevitably increasing. The increasing network flows require expanding network capabilities, such as more bandwidth or base stations, resulting in more network operation costs. Network optimization cost-effectively offers various benefits such as more significant throughput, eliminating service violations and reducing the delay of application and network. Without network

optimization, the network operators cannot accommodate advanced users and new applications while maintaining or improving network performance.

Due to the broad range of optimization technologies, this chapter structures the related work in network optimization into three classes: Gradient-based, Linear Programming, and Non-Linear Programming methods.

2.1 Gradient-based Methods

Gradient-based methods represent a well-established category of optimization strategies. They primarily rely on the derivatives of objective functions, such as the first-order or second-order derivatives of a given function f. The term gradient originates from the derivative of a multidimensional function, $f(\vec{x})$. For example, $\nabla f(\vec{x}) =< \frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n} >$ represents a vector that each element is a slope of $\vec{x}$ in the corresponding dimension [28]. The fundamental concepts underpinning gradient-based methods are straightforward.

For instance, given a polynomial function, $f(x)$, in one variable, x. The concept starts by computing the gradient at any initial value of x and then iteratively adds to or subtracts from a relevant small positive value at x until reaching the minimal value of $f(x)$. The process denotes as equation (2.2):

$$x = x - \beta f'(x) \tag{2.2}$$

where β represents a small enough positive value, and $f'(x)$ denotes the first-order derivative of $f(x)$. As a result, a positive slope leads x to decrease, while a negative slope will cause x to increase. Because the process above aims to find the minimal $f(x)$, it is also called gradient descent. From the illustration in Fig. 2.4, $f'(x)$ is negative, Hence, x will progressively descend the function until reaching its minimal value, at which point $f'(x)$ becomes zero, halting its progress.

However, gradient-based methods are unsuitable for tackling optimization problems with unknown step lengths that take a long time to converge. For instance, as gradient descent approaches a function minimum, an unfeasible step length may be too large to reach the minimum and move forward in the wrong direction. On the other hand, in the subsequent step, slowly move toward the target in a "zig-zag" manner from the other side of the minimum point. This behavior is intrinsically connected to the function's slope at a particular point, meaning that a steeper slope corresponds to a more significant leap, which can be mitigated by a small modification of the value of β. Nevertheless, some functions (or function regions) necessitate smaller values, while others require larger values. Newton's method enhances this process by considering the function's second derivative, $f''(x)$, as follows:

$$x = x - \beta \frac{f'(x)}{f''(x)}, \tag{2.3}$$

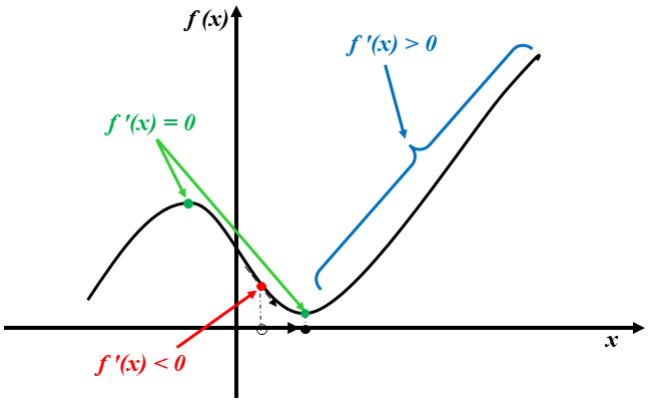

Fig. 2.4: An example of derivative and x is increased by a negative derivative

thereby modifying the value of β as it converges towards a point with zero slope [28].

Another concern was identifying points other than the maximum or minimum points. The reason is that many functions may encompass saddle points (i.e., inflection points in one-dimensional functions) besides maxima and minima. For example, in Fig. 2.5, the first derivative of a saddle point is zero, so the gradient descent will cease searching for the minimum at the saddle point despite not reaching it. In this case, Newton's method does not offer assistance, even attempting to divide by zero. The case explicitly demonstrates how gradient descent becomes trapped in local optima. Local optima of a function are defined as the optimal value (or minimum value in this case) of a local region. Similarly, global optima are defined as the optimal value of the entire domain of a function. It can be inferred that gradient-based methods, such as gradient descent or Newton's method, operate as local optimization algorithms [28].

The last but not most minor design requirement is that the gradient-based methods leverage the derivable property of a function to find an optimal solution. This assumption is valid only when optimizing functions with good mathematical properties. Unfortunately, this condition is often not satisfied, as the gradient is generally non-computable in most cases due to the complex function. Consequently, numerous researchers continue to refine gradient-based optimization techniques.

2.2 Linear Programming Methods

The most widely used optimization techniques is linear programming, which were originally developed in the field of Operations Research to solve various

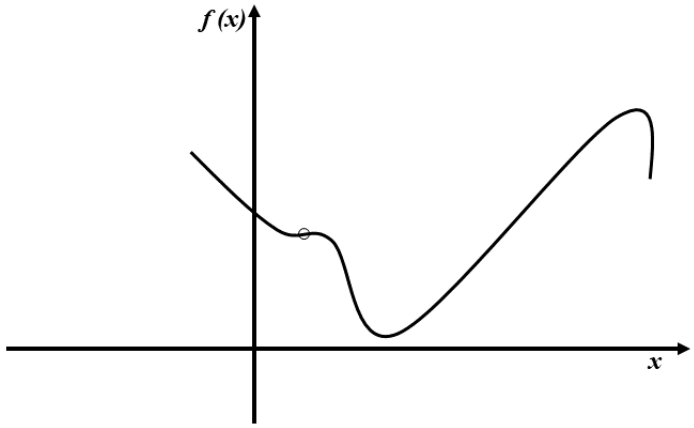

Fig. 2.5: An example for a saddle point (inflection point)

logistical optimization problems.

In the linear programming optimization model, the given objective function f and constraint functions are all linear. These constraints impose limitations on $\vec{x}$, necessitating adherence to specific conditions such as conforming to restricted resource availability. A linear programming optimization problem can be modeled in (2.4) and (2.5).

$$\min f(\vec{x}) = \vec{c} \cdot \vec{x} \tag{2.4}$$

subject to

$$\begin{aligned} A \cdot \vec{x} &\leq \vec{b} \\ \vec{x} &\geq \vec{0} \end{aligned} \tag{2.5}$$

The inequalities defined in Equation (2.5) constitute the constraints for the linear program specified in Equation (2.4). Fig. 2.6 demonstrates a linear-programming example with several constraints. In order to address continuous and linear optimization problems, there are efficient exact algorithms from the prior arts, such as the ellipsoid method [31], the criss-cross method [23], or the interior-points method [21].

In fact, linear programming represents one of the most well-established methods for resolving optimization problems, because the linear objective function and constraints compose a convex property. Consequently, the feasible region of the problem solutions is also convex, and the global optimum corresponds to a vertex of the polytope that can be found in the feasible region [35].

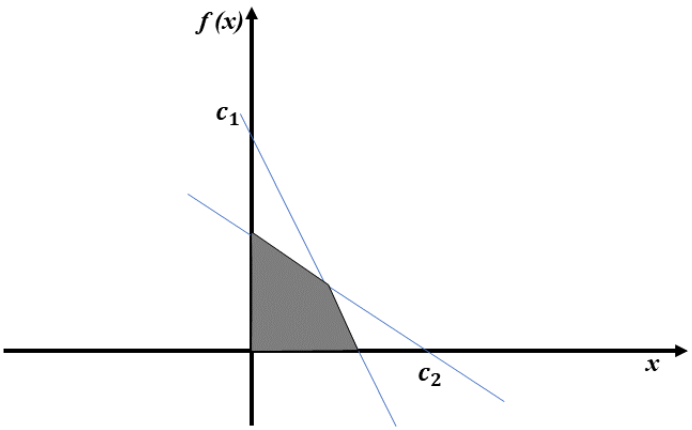

Fig. 2.6: A two-dimension example of linear-programming with constraints, c_1 and c_2. The region of feasible solutions is marked by the grayed polytope area.

2.3 Nonlinear Programming Methods

Nonlinear programming methodologies aim to solve optimization problems characterized by nonlinear objective functions or constraints [8]. Existing techniques that try to simplify these nonlinear objective functions or constraints introduce additional variables and a certain degree of approximation [15]. Furthermore, properties such as high dimensionality and multimodality reduce these techniques' efficiency. In recent telecommunication research, the latest results have concentrated on devising more efficient nonlinear programming methods. These advancements have led to improvements in several areas, and the selected topics in recent research are listed below:

- **Resource allocation:** Nonlinear programming has been applied to resource allocation problems in telecommunication networks, including power control, beamforming, and scheduling [42]. By optimizing the allocation of limited resources, these methods aim to maximize network capacity, throughput, and energy efficiency while maintaining the quality of service (QoS) requirements.

- **Network Planning:** Network planning is complex to adaptively coordinates base stations and antennas to satisfy the QoS requirements from user equipment. This research [24] addresses the non-line-of-sight (NLOS) problem and transforms it into a mixed-integer second-order cone programming, effectively solving NLOS issues and enabling ubiquitous coverage in a telecommunication network.

- **Cross-layer optimization:** In order to account for the interactions be-

tween different layers of the network components, cross-layer optimization techniques have been developed. For example, a method integrate multiple layers' objectives and constraints into a mixed-integer nonlinear programming problem, allowing for a more holistic optimization approach [36].

In the context of real-world problems, the availability of an analytical objective function is not always held. In certain instances, simulations or physical models objective-function evaluation [14]. In these cases, machine learning technologies emerge as promising candidates for addressing various instance sizes of this class of problems.

In conclusion, recent advancements in nonlinear programming for telecommunication have fostered the development of more sophisticated and efficient techniques, expanding the range of problems that can be effectively solved. For example, the research in [40], integrating these innovations with machine learning technologies has significantly increased the potential for tackling complex, real-world optimization challenges.

3 Machine Learning Technologies

Massive-connected devices generate data and require network transmissions under stringent latency, throughput or reliability performance. In order to fulfill the divergent requirements, next-generation networks must increasingly adapt autonomous and cognitive capabilities. For instance, the network edge entities need to spontaneously make decisions using their observation of network states to increase the efficiency and reduce the complexity of the network controller. Those cognitive decision scheme can be used in network flow optimization such as maximizing the network throughput under dynamic network environments without slow manual configurations. On the other hand, the cognitive decision scheme is an integral aspect of machine learning, as it facilitates decision-making processes by leveraging the dataset or observations derived from network environments.

Machine learning algorithms [20] [29] [32] are designed to mimic human cognitive decision-making capabilities such as sensing, mining (e.g., flow classification), prediction (e.g., forecasting future traffic demands), and inference. These algorithms adaptively learn from experience and data patterns, enabling them to make increasingly better decisions over time. The cognitive decision scheme thus serves as a foundation for optimizing network systems that can autonomously analyze complex flows, solve problems, and then make optimal choices, all while continuously refining their understanding of the network environment and improving their decision-making capabilities.

The authors in [19] [10] [4] investigate different ML solutions in solving network optimization problems such as mobility management, resource management and traffic classification. The solutions could be classified as Un-supervised Learning, (Semi-) supervised learning, and Reinforcement Learning (RL) with different implementation such as neural network or decision tree. Moveover, a distributed version on top of the previous three algorithms is discussed in an individual subsection. The four main ML algorithms present their features for the network optimization in the following subsections. This section outlined the primary ML categories listed in the following sub-sections.

3.1 Supervised Learning

Supervised learning extracts input data and output labels from training data and thus expects to map new input data to correctly predicted outputs. However, the Supervised Learning approach has fewer applications over online data flows compared to the other ML categories because this learning algorithm requires labelled data. The data flows are difficult to label in a dynamic network environment, and the existing flows are often encrypted during transmissions. The two reasons above prevent the Supervised Learning approach from distinguishing different flows. In addition, supervised learning may encounter difficulties in attaining satisfactory processing performance when dealing with a complex problem. For example, Mohseni et al. [26] and Hendrycks et al. [16] attributed the performance gap to the potential inability of the method to construct a suitable representation of the data distribution, which is essential for effectively facilitating discriminative tasks.

3.2 Reinforcement Learning

Reinforcement Learning is a machine learning paradigm where an agent strives to find optimal decisions by interactive learning from its environment, receiving feedback in the form of rewards or penalties based on the actions it selects, and striving to find a balance between exploration and exploitation [33]. For instance, an ultra-reliable low-latency communication (URLLC) service implementation employs reinforcement learning to allow network entities to ascertain the optimal policy, including decisions and actions, given their states in scenarios with finite state and action spaces [22]. However, in intricate and large networks, such as heterogeneous networks (HetNets), the large state and action spaces may hinder reinforcement learning from identifying the optimal policy within a reasonable timeframe. Consequently, deep reinforcement learning, which melds reinforcement learning with deep learning, has been developed to address these limitations. A pioneering deep rein-

forcement learning framework is proposed to facilitate model-free URLLC in the downlink of an orthogonal frequency division multiple access (OFDMA) system [38].

In addition, some researches consider the optimization of rare network error events (e.g. burst errors, traffic jam and signal shadowing), whose behaviour cannot be exactly modelled, but mostly affect the URLLC reliability [22]. Such events can potentially be detected and monitored, while also building methods against them through repetitions in time/frequency and higher SNR [27]. On the other hand, the user equipment association and resource allocation could also be determined by each base station through a deep reinforcement learning method [34]. Similarly, in [12], the authors study the UE handover optimization from real 5G traffic and develop a ML approach to reorganize the distributed unit grouping and then reduce some control messages between the distributed units.

3.3 Unsupervised Learning

Despite the prevalence of human learning through supervision (e.g., manual guidance) or reinforcement (e.g., environmental interaction), unsupervised learning remains a crucial technique [17]. Algorithms within unsupervised learning can autonomously self-organize networks, considering features such as online network congestion statistics and QoS requirement of applications [30]. Unsupervised learning is also used in some network optimization scenarios due to its ability to operate without labeled data, which can be costly and challenging to obtain scalably. For example, the authors in [11] employed a scalable Bayesian network for network flow monitoring and analysis. Furthermore, the authors from [37] implemented a hybrid architecture combining unsupervised learning with supervised feature classification to enhance the radio resource management and the QoE-based admission control for a video service.

3.4 Distributed Learning

Some literature extends the centralized ML paradigm to the distributed or federated paradigm [25] [39]. Since ML in telecommunication networks sometimes incurs high costs or privacy issues while persistent data collection, a distributed ML approach [7] is introduced to avoid such conditions. In the distributed ML paradigm, mobile devices train their ML models locally using on-device data without a parameter server. The training procedure is repeated in multiple epoches until a desirable model accuracy is achieved. The procedure can be easily implemented in a distributed network and imply that distributed ML can be embedded in a distributed network with cost-saving and privacy protection.

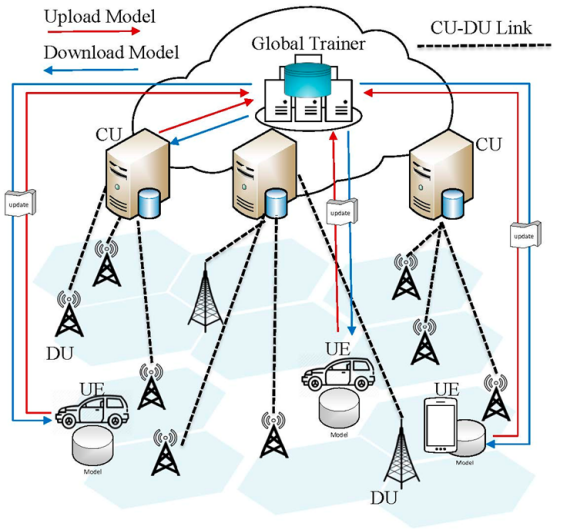

Fig. 2.7: Federated RL Paradigm

Based on the distributed ML concepts, a federated RL paradigm for telecommunication networks has also been proposed [41] [6], as shown in Fig. 2.7. The paradigm only considers reducing the data collection overhead in the network. During the federated RL training, mobile devices send the local model updates (e.g. the model's weights), to a global trainer for aggregation (e.g., to average the weights from users). A RAN consists of base stations, denoted as gNBs. A gNB includes multiple distributed units (DUs) and a centralized unit (CU). UEs download and improve the latest model by learning from data (e.g., downlink signal strength, channel quality) in the local environment and then summarize the model differences in a single update. Only the model updates can be sent to the global trainer to improve the model. All the training data remains on the local device, and the global trainer does not need to keep individual updates. Therefore, only the smallest amount of signaling will be generated in the network and can limit the explosion of communication costs during the RL training.

4 Discussion

This chapter introduced cutting-edge telecommunication network development, optimization techniques, and the most crucial Machine Learning (ML) technologies. Various techniques have been incorporated into contemporary telecommunication networks to enable a flexible orchestration or a robust management for complex flow requirements, such as those embodied by the

Approach	Pros	Cons
Deterministic Algorithm	Obtain optimal solutions for a specific problem	Often difficult to design an algorithm and to accomodate new constraints
(Non-)Linear Programming (LP)	1.Provides optimality gap 2.Can be very efficient	Requires unique optimization modeling expertise to model and tune
Metaheuristics Algorithm	1.Easy to create running model 2.Can be significantly more efficient than LP for some problem types	1.Requires deep understanding of specific algorithm to provide good solution 2.No theoretical optimality gap 3.Often less suitable for continuous decision variables
Reinforcement Learning	1.Use comprehensive abstraction (states/actions/rewards) 2.Inherent handling of uncertainty	1.struggle to handle high-dimensional data/extensive hyperparameter and algorithm tuning 2.Incorporating constraints 3.Inherent uncertainty makes explainability more difficult

Table 2.1: Comparison of Optimization Techniques

Software-Defined Networking (SDN) and Network Function Virtualization (NFV) paradigms. In particular, Reinforcement Learning (RL) has been advocated as the vital building block for network flow optimization and has shown that RL in telecommunication networks has a wave of success in improving network performance. A comparison of classical optimization and RL techniques is listed in the Table 2.1.

The doctoral research investigates some preliminary examples of the potential network flow optimization, including Quality of Service (QoS) improvement, Quality of Experience (QoE) enhancement, and network resource management. The subsequent chapters will present additional optimization research findings to substantiate this doctoral study.

References

[1] "3gpp ts 28.105; 5g; management and orchestration; artificial intelligence/machine learning (ai/ml) management," 2023.

[2] 5G-ACIA, "5g for connected industries and automation, 2nd ed., white paper," 2019.

[3] N. Alliance, "5g white paper," *Next generation mobile networks, white paper*, vol. 1, no. 2015, 2015.

[4] M. A. Alsheikh, S. Lin, D. Niyato, and H.-P. Tan, "Machine learning in wireless sensor networks: Algorithms, strategies, and applications," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 1996–2018, 2014.

[5] J. G. Andrews, "Seven ways that hetnets are a cellular paradigm shift," *IEEE communications magazine*, vol. 51, no. 3, pp. 136–144, 2013.

[6] F. Ang, L. Chen, N. Zhao, Y. Chen, W. Wang, and F. R. Yu, "Robust federated learning with noisy communication," *IEEE Transactions on Communications*, vol. 68, no. 6, pp. 3452–3464, 2020.

[7] D. R. B. McMahan, "Federated learning: Collaborative machine learning without centralized training data," 2017.

[8] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*. Wiley-Interscience, 2006.

[9] S. H. Chae, J.-P. Hong, and W. Choi, "Optimal access in ofdma multi-rat cellular networks with stochastic geometry: Can a single rat be better?" *IEEE Transactions on Wireless Communications*, vol. 15, no. 7, pp. 4778–4789, 2016.

[10] 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 & Tutorials*, vol. 21, no. 4, pp. 3039–3071, 2019.

[11] X. Chen, K. Irie, D. Banks, R. Haslinger, J. Thomas, and M. West, "Scalable bayesian modeling, monitoring, and analysis of dynamic network flow data," *Journal of the American Statistical Association*, vol. 113, no. 522, pp. 519–533, 2018.

[12] R. Dong, C. She, W. Hardjawana, Y. Li, and B. Vucetic, "Deep learning for hybrid 5g services in mobile edge computing systems: Learn from a digital twin," *IEEE Transactions on Wireless Communications*, vol. 18, no. 10, pp. 4692–4707, 2019.

[13] M. T. Emmerich and A. H. Deutz, "A tutorial on multiobjective optimization: Fundamentals and evolutionary methods," *Natural Computing: An International Journal*, vol. 17, no. 3, p. 585609, sep 2018. [Online]. Available: https://doi.org/10.1007/s11047-018-9685-y

[14] M. C. Fu, "Optimization for simulation: Theory vs. practice," *INFORMS Journal on Computing*, vol. 14, no. 3, pp. 192–215, 2002.

[15] F. Glover, "Improved linear integer programming formulations of nonlinear integer problems," *Management Science*, vol. 22, no. 4, pp. 455–460, 1975.

[16] D. Hendrycks, S. Basart, M. Mazeika, A. Zou, J. Kwon, M. Mostajabi, J. Steinhardt, and D. Song, "Scaling out-of-distribution detection for real-world settings," 2022.

[17] G. Hinton and T. J. Sejnowski, *Unsupervised learning: foundations of neural computation*. MIT press, 1999.

[18] H. Holma, A. Toskala, and J. Reunanen, *LTE small cell optimization: 3GPP evolution to Release 13*. John Wiley & Sons, 2016.

[19] C. Jiang, H. Zhang, Y. Ren, Z. Han, K.-C. Chen, and L. Hanzo, "Machine learning paradigms for next-generation wireless networks," *IEEE Wireless Communications*, vol. 24, no. 2, pp. 98–105, 2017.

[20] V. P. Kafle, Y. Fukushima, P. Martinez-Julia, and T. Miyazawa, "Consideration on automation of 5g network slicing with machine learning," in *2018 ITU Kaleidoscope: Machine Learning for a 5G Future (ITU K)*, 2018, pp. 1–8.

[21] N. Karmarkar, "A new polynomial-time algorithm for linear programming," in *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*. ACM, 1984, pp. 302–311.

[22] A. T. Z. Kasgari and W. Saad, "Model-free ultra reliable low latency communication (urllc): A deep reinforcement learning framework," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–6.

[23] T. T. Komei Fukuda, "Criss-cross methods: A fresh view on pivot algorithms," *Mathematical Programming*, vol. 79, p. 369395, 1997.

[24] S.-C. Lin and I. F. Akyildiz, "Dynamic base station formation for solving nlos problem in 5g millimeter-wave communication," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, 2017, pp. 1–9.

[25] R. Mayer and H.-A. Jacobsen, "Scalable deep learning on distributed infrastructures: Challenges, techniques, and tools," *ACM Comput. Surv.*, vol. 53, no. 1, feb 2020. [Online]. Available: https://doi.org/10.1145/3363554

[26] S. Mohseni, M. Pitale, J. Yadawa, and Z. Wang, "Self-supervised learning for generalizable out-of-distribution detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 5216–5223.

[27] M. Polese, R. Jana, V. Kounev, K. Zhang, S. Deb, and M. Zorzi, "Machine learning at the edge: A data-driven architecture with applications to 5g cellular networks," *IEEE Transactions on Mobile Computing*, vol. 20, no. 12, pp. 3367–3382, 2021.

[28] W. B. Powell, *Derivative-Based Stochastic Search*. John Wiley Sons, Ltd, 2022, ch. 5, pp. 221–271.

[29] D. F. Preciado Rojas, F. Nazmetdinov, and A. Mitschele-Thiel, "Zero-touch coordination framework for self-organizing functions in 5g," in *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, 2020, pp. 1–8.

[30] J. Qadir, "Artificial intelligence based cognitive routing for cognitive radio networks," *Artificial Intelligence Review*, vol. 45, no. 1, pp. 25–96, 2016.

[31] S. Rebennack, *Ellipsoid Method*, C. A. Floudas and P. M. Pardalos, Eds. Boston, MA: Springer US, 2009.

[32] V. Sciancalepore, F. Z. Yousaf, and X. Costa-Perez, "z-torch: An automated nfv orchestration and monitoring solution," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1292–1306, 2018.

[33] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[34] V. N. Swamy, N. Naderializadeh, V. N. Ekambaram, S. Talwar, and A. Sahai, "Monitoring under-modeled rare events for urllc," in *2019 IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2019, pp. 1–5.

[35] E.-G. Talbi, *Metaheuristics: From Design to Implementation*. Wiley, 2009, vol. 74.

[36] J. Tang, W. P. Tay, and T. Q. S. Quek, "Cross-layer resource allocation with elastic service scaling in cloud radio access network," *IEEE Transactions on Wireless Communications*, vol. 14, no. 9, pp. 5068–5081, 2015.

[37] A. Testolin, M. Zanforlin, M. D. F. De Grazia, D. Munaretto, A. Zanella, M. Zorzi, and M. Zorzi, "A machine learning approach to qoe-based video admission control and resource allocation in wireless systems," in *2014 13th Annual Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET)*. IEEE, 2014, pp. 31–38.

[38] M. Wang, Y. Cui, X. Wang, S. Xiao, and J. Jiang, "Machine learning for networking: Workflow, advances and opportunities," *IEEE Network*, vol. 32, no. 2, pp. 92–99, 2018.

[39] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 869–904, 2020.

[40] C.-C. Wu, P. Popovski, Z.-H. Tan, and . Stefanovi, "Design of aoi-aware 5g uplink scheduler using reinforcement learning," in *2021 IEEE 4th 5G World Forum (5GWF)*, 2021, pp. 176–181.

[41] H. H. Yang, Z. Liu, T. Q. S. Quek, and H. V. Poor, "Scheduling policies for federated learning in wireless networks," *IEEE Transactions on Communications*, vol. 68, no. 1, pp. 317–333, 2020.

[42] B. Zhang, X. Mao, J.-L. Yu, and Z. Han, "Resource allocation for 5g heterogeneous cloud radio access networks with d2d communication: A matching and coalition approach," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 7, pp. 5883–5894, 2018.

Chapter 3

Conclusions

This dissertation covers the research topics related to fast and reliable transmissions over telecommunication networks. In particular, the dissertation focuses on solving optimization problems in those topics via reinforcement learning. The optimization procedure can be generalized in three research questions below: how to optimally balance the latency and throughput tradeoff in the URLLC service (Q1), how to optimize the network resource reservation across different objectives (Q2), and how to analyze the optimization performance and provide end-to-end robustness using reinforcement learning (Q3). The three research questions are concluded below based on theoretical analysis and numerical simulations in this dissertation.

Q1: How to optimally balance the latency and throughput tradeoff in the URLLC service? The optimality between two primary network performance metrics: latency and throughput, had been investigated in Paper A and Paper B for a RAN-level scenario. In general, the network operator needs to schedule network resources to fulfill the performance requirements of data flow transmission in a dynamic network environment. The scheduling tradeoff between the latency and throughput requirements comes from the discontinuity of traffic flows. This PhD research studied the periodic and burst data flows generated from the Poisson Pareto Burst stochastic model. In addition, a deep reinforcement learning framework was designed to learn the optimal URLLC uplink scheduling based on the flows' temporal-spatial patterns. The deep reinforcement learning method can adopt different flow patterns from the user behaviour changes in the network environment without prior knowledge of the flow model. Therefore, the proposed deep reinforcement learning method is suitable for solving the tradeoff optimization problem in the dynamic network environment.

Q2: How to optimize the online network resource scheduling across different optimization objectives? Several performance metrics measure the optimality of online network resource scheduling. The different performance metrics turn into multiple complex optimization objectives, and sometimes those objectives are in conflict. Although the network resource scheduling scenario is challenging, there exists a statistical relationship between the resource requests. Hence, the proposed reinforcement learning method can still learn an optimal representation of scheduling policies for all objectives over the solution space. The objectives of this PhD research include latency, data rate, fairness (in Paper C), and SLA (Service-Level Agreement) violation rate (in Paper D). This PhD research has designed a reinforcement learning method to find optimal scheduling for multi-objective optimization in network resource scheduling.

Q3: How to analyze the performance of optimization and provide end–to-end robustness using reinforcement learning? The attainment of an optimal solution for both RAN-level and end-to-end network transmissions is paramount. The proposed reinforcement learning framework has been extended to optimize the entire transmission route, encompassing Radio Access Network (RAN), Transport Network (TN), and Core Network (CN). Rather than solely concentrating on optimization within the RAN domain, this end-to-end optimization approach guarantees network performance in real-world use cases and facilitates unbounded network connectivity. The findings of this research not only consider the physical scenarios but also extend beyond simplified modeling by incorporating mathematical formulations. The worst-case comparison of computational complexity and competitive ratio proves the performance guarantee of the proposed method. In addition, the experiments demonstrate that the proposed algorithms exhibit effective solutions and policy approximation to adapt and elicit unknown traffic in the network.

According to the research findings in this dissertation, some potential directions may be valuable to work on in the future. First, instead of focusing on the network traffic flows in a single-machine version algorithm, in the distributed version algorithm, we may deduce the computational time consumption while keeping the accuracy of optimal solutions. In addition, more network topologies can be investigated to identify the impacts of network scale corresponding to the accuracy of optimal solutions. On the other hand, the optimization was inspired by the URLLC scenario sharing similar network traffic with the different connectivity scenarios such as eMBB and mMTC. The prior model can be applied to those scenarios via the transfer learning method.

Part II

Papers

Paper A

Design of AoI-Aware 5G Uplink Scheduler Using Reinforcement Learning

Chien-Cheng Wu; Petar Popovski; Zheng-Hua Tan; Cedomir Stefanovic

The paper has been published in the
Proceeding of IEEE 4th 5G World Forum, pp. 176–181, 2021.

The layout has been revised.

Abstract

Age of Information (AoI) reflects the time that is elapsed from the generation of a packet by a 5G user equipment (UE) to the reception of the packet by a controller. A design of an AoI-aware radio resource scheduler for UEs via reinforcement learning is proposed in this paper. In this paper, we consider a remote control environment in which a number of UEs are transmitting time-sensitive measurements to a remote controller. We consider the AoI minimization problem and formulate the problem as a trade-off between minimizing the sum of the expected AoI of all UEs and maximizing the throughput of the network. Inspired by the success of machine learning in solving large networking problems at low complexity, we develop a reinforcement learning-based method to solve the formulated problem. We used the state-of-the-art proximal policy optimization algorithm to solve this problem. Our simulation results show that the proposed algorithm outperforms the considered baselines in terms of minimizing the expected AoI while maintaining the network throughput.

1 Introduction

Many emerging services and systems such as Autonomous Driving, Industrial Automation, and Tactile Internet require real-time monitoring and low-latency information delivery. The growth of time-sensitive information led to a new data freshness measure named Age of Information (AoI). AoI evaluates the time elapsed from the generation of the last update at a source that was received at the destination [1]. Consider a cyber-physical system such as an automated factory where many robots are transmitting time-sensitive information to a remote observer through a wireless network. Each robot continuously samples the environment at the physical factory and transmits the sample data to the monitoring observer. Due to inevitable bandwidth limitations, the network may not be able to transmit all the data to the observer. Consequently, each robot may be allocated a part of the transmission bandwidth that does not completely satisfy its requests. In such cases, the latest generated data can not be transmitted immediately and will be accumulated at the robot's queue, resulting in the AoI increase. A way to mitigate such scenarios is to optimize the operation of the network scheduler.

In the literature, many works are investigating the time-average AoI [1] and aims to achieve the minimum time-average AoI. If the network traffic from each node is known, the optimal scheduling policy for minimizing time-average AoI can be derived with low computation complexity [2]. However, the computation complexity of finding an optimal scheduling policy without any prior knowledge of the network traffic is high.

In this paper, leveraging the success of machine learning in solving many of the online large-scale networking problems [3], we propose a method

based on Reinforcement Learning (RL) to solve an optimization problem that combines network utility (a measure of matching between the required and allocated resources per user) and minimization of AoI. In the training phase, the RL agent starts to capture network state in terms of the number of user equipments (UEs) and traffic volume of each UE. Then, at the given state, the RL agent is trained to choose a scheduling action that maximizes the total reward. The RL agent continuously interacts with the environment and tries to find the best policy based on the reward fed back from the environment. In the testing phase, obtain a policy capable of optimizing the transmission schedule.

As already hinted, the reward of the minimization problem considered in the paper combines two objective functions. The first is the expected AoI of each node (i.e., user) and the second is the utility of each node.

These two objectives can be juxtaposed, as maximizing utility may adversely affect AoI and vice versa. For example, consider a network with two nodes A and B, sharing the same wireless channel that can accommodate $L > 1$ packets per time slot. Node A and node B have a fixed size queue of length L. Assume that node A generates L packets per timeslot, while node B only generates 1 packet per timeslot. The scheduling policy selects which node is allowed to transmit packets at a given timeslot. Policy $p1$ alternates between node A and node B. Policy $p2$ selects node B every L slots and selects node A otherwise. Policy $p2$ has larger throughput than Policy $p1$ (i.e., every slot is fully used). But Policy $p1$ has lower AoI than Policy $p2$, as node B waits less for the transmission opportunities. In the paper, we consider a generalization of the problem, in which the nodes are generating traffic with an a priori unknown statistics, and design an agent to learn the scheduling strategy that maximizes the reward based on the RL paradigm.

The rest of this paper is organized as follows. Section 2 presents a brief overview of the related work. The system model and the problem formulation are described in Section 3. The proposed learning algorithm is presented in Section 4, whereas the numerical simulations are given in Section 5. Finally, the conclusions are drawn in Section 6.

2 Related Work

Recently, a number of papers tackled the problem of minimizing the AoI of many sources that are competing for the available radio resources. [2] considers the problem of many sensors connected wirelessly to a single monitoring node and formulate an optimization problem that minimizes the weighted expected AoI of the sensors at the monitoring node. The authors of [4] also consider the sum expected AoI minimization problem when constraints on the packet deadlines are imposed. In [5], the minimization of the sum ex-

pected AoI is considered in cognitive shared access.

The scheduling decisions with multiple receivers over a perfect channel are investigated in [6, 7], where the goal is to learn data arrival statistics. Q-learning is used for a generate-at-will model in [6], while policy gradients and DQN methods are used for a queue-based multi-flow AoI-optimal scheduling problem in [7]. In addition, AoI in multi-user networks has been studied in [6] [8]. The authors in [9] show that finding an optimal scheduling decision that minimizes AoI is an NP-hard problem.

Scheduling transmissions to multiple receivers is investigated in [6], focusing on a perfect transmission medium, and the optimal scheduling algorithm is shown to be of threshold type on the AoI. Average AoI has also been studied when status updates are transmitted over unreliable multiple-access channels [10] or multicast networks [11]. In addition, Peak AoI has also been jointly considered with Average AoI in [12] and a UAV's trajectory and average peak AoI optimization problem has been studied at [13]. A source node sending time-sensitive information to several users through unreliable channels is considered in [14], where the problem is formulated as a multi-armed bandit (MAB), and a suboptimal Whittle Index (WI) policy is proposed.

Most literature working on AoI minimization problems assumes perfect statistical knowledge of the random processes governing the status update system. However, in most practical systems (e.g., heterogeneous UEs with different mission-critical traffic co-located in the same network), the characteristics of the system, e.g., mobility pattern, traffic distribution, etc, are not known a priori and must be learned. A limited number of recent works consider the unknown or time-varying characteristics of status update systems, and apply a learning-theoretic approach [7, 15]. To the best of our knowledge, the scheduling for a tradeoff between average AoI minimization and throughput maximization via utility modeling is studied for the first time at a multi-user system.

3 System Model

In this paper, we consider a mission-critical system consisting of a 5G network with one base station (gNB), N UEs and UE's controller as illustrated in Fig. A.1. The figure depicts a centralized autonomous-control factory, which is a mission-critical system. The centralized controller monitors the state of each UE/robot through the 5G wireless network at the remote side. The robots connect to the controller via the base station. The connection between a robot and the controller can be modeled as a virtual link. Since the robots briskly operate at the production line, to keep the system reacting in time, robots shall transmit fresh data such as sensor information or velocity to the controller and fetch control signals back to maintain the system opera-

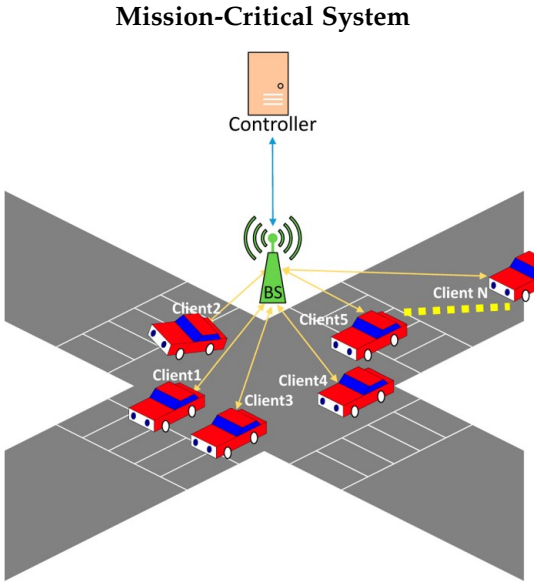

Fig. A.1: An autonomous control factory

tion. All the UE-generated data can be carried in one or multiple packets and transmitted to the controller individually via the wireless link. For the packet transmission scheduling over a link, a time-slotted system is considered, where scheduling decisions are made and transmitted to the UEs at the beginning of each time slot t. Each time slot has a duration of τ, e.g., 1 ms or even smaller by using dynamic transmission time intervals. The total channel bandwidth is limited to B units, where $0 < B < N$. Due to the network resource limitation, the network only allows a subset of UEs (denote as m, $0 < m \leq N$) at a timeslot t to send packets to the central controller. Denote by $S_n(t)$ be a random variable that indicates the selection of UE n by the gNB at time slot t, i.e., $S_n(t) = 1$ if the controller selects UE n at time slot t, and $S_n(t) = 0$ otherwise. When $S_n(t) = 1$, UE n sends a subset of the packets, depending on the amount of allocated resource, to the controller. Otherwise, the UE cannot send any packets to the controller. If the user (say user n) is selected for transmission at slot t, the allocation of channel bandwidth at time slot t to UE n is denoted by $b_n(t)$, where $b_n(t)$ is an integer number of units, $0 < b_n(t) \leq B$ and $\sum_{n \in \mathcal{I}(t)} b_n(t) = B$, where $\mathcal{I}(t)$ is the set of selected users in slot t. The schedule at moment t is denoted by $\mathcal{S}(t) = \{S_1(t), S_2(t), \ldots S_n(t), \ldots S_N(t)\}$, which belongs to the set of all possible schedules S.

The number of packets generated at time slot t by user n is denoted by $X_n(t)$, following a memoryless arrival process that is independent and iden-

tically distributed over users. The generated packets are stored at UE transmitters' queues, following the first-come-first-serve (FCFS) policy. The queue length is fixed and initialized to 0.

The AoI of each UE n is defined as the time elapsed between the current time and the generation of the latest packet that departed from the transmitter; we assume that all the packets that are transmitted are also successfully received. The AoI is computed by the formula:

$$A_n(t) := t - \max_i \{t_G^n(i)) | t_G^n(i)) \leq t\} \tag{A.1}$$

where $t_G^n(i))$ and $t_D^n(i)$ are the moments of generation and departure of the ith packet of UE n. Due to the slotted time assumption, AoI is changing in integer units (i.e., number of slots). If no packet from UE n is received by the controller in time slot t, $A_n(t)$ is increased by 1. Otherwise, if at time slot t, a packet from UE n is successfully received by the controller, the AoI will be updated below:

$$A_n(t+1) = \begin{cases} A_n(t) + 1 & \text{if } X_n(t) b_n(t) S_n(t) = 0 \\ t - t_G^n(i) & \text{if } X_n(t) b_n(t) S_n(t) > 1 \end{cases} \tag{A.2}$$

where $t_G^n(i)$ is the generation moment of the latest packet that was successfully received in slot t (denoted by the dummy index i).

Further, the node and network utility per slot are defined, respectively, as:

$$U_n(t) = \frac{1}{1 + e^{-(1.5 \times b_n(t) - X_n(t))}} S_n(t),\ 0 \leq n \leq N \tag{A.3}$$

$$U(t) = \sum_{n=1}^{N} U_n(t) \tag{A.4}$$

$U_n(t)$ is a sigmoid function, while the network utility $U(t)$ is the summation of all node utility values at time slot t. The higher network utility means a higher network throughput [12].

Our goal is to find suitable scheduling policies for a set of UEs, which attempt to maximize network utilization while minimizing the aggregate AoI. Specifically, we formulate our goal via the function given below. In each time slot t, select a schedule $\mathcal{S}(t)$ s.t.

$$\mathcal{S}(t) = \underset{\mathcal{S}(t) \in \mathbb{S}}{\operatorname{argmax}} \sum_{n=1}^{N} [U_n(t) - \beta A_n(t)] \tag{A.5}$$

where $\beta \in \mathbb{R}$ is a scalar control parameter. If β is larger, the scheduling will be more sensitive than a small β. Problem (A.5) is a non-linear integer programming (NLIP) that is generally complicated to solve [16]. Actually,

the optimization variables in (A.5) include sequential decisions. It was recently shown that Deep RL (DRL) achieves high performance on the long-term sequential decision-making problems without human knowledge [17–19]. Moreover, DRL-based solutions can provide the decisions in an automatic and zero-touch manner. In addition, benefiting from deep neural networks, DRL is capable of handling high-dimensional observation-action spaces. These motivate us to propose policy-based model-free DRL solutions, discussed in the next section.

4 Proposed Model and Algorithm

We assume an RL agent interacting with a network environment to learn a scheduling policy without prior information.[1] To deal with the unknown information in the network environment, an RL agent gradually learns the scheduling policy from the network environment observations. We define the scheduling policy as $\pi(o_t, \theta_t)$ that returns a schedule $\mathcal{S}(t)$ as an action a_t to satisfy $\sum_{n=1}^{N} b_n(t) \leq B$. The action a_t represents which UEs are selected to transmit their packets. For example, if only $S_1(t) = S_2(t) = 1$ in $\mathcal{S}(t)$, it means that only UE1 and UE2 are selected to transmit data with their allocated bandwidth $b_1(t)$ and $b_2(t)$. The other UEs are not allowed to transmit data and have to wait for the next action. In the time-slotted system, every interaction between the agent and the network environment happens at the beginning of a time slot. In each interaction, the agent samples the environment to get an observation o_t and performs an action a_t based on the scheduling policy. After performing the action, the agent receives a reward r_t. Then the agent waits for the time slot $t+1$ to interact with the network environment. The learning process repeats the interactions continuously to approximate the optimal scheduling policy which obtains the maximal reward. Hence, the objective of learning is to maximize the expected cumulative reward.

The optimization goal is defined as

$$J := \max_{\pi_k} \mathbb{E}\left[\sum_{n=1}^{N} \left(U_n(t) - \beta A_n(t)\right)\right] \tag{A.6}$$

$$\text{s.t. } \textstyle\sum_{n=1}^{N} b_n(t) \leq B,\ \pi_k = \mathcal{S}(t)$$

Our RL agent exploits the Proximal Policy Optimization (PPO) Algorithm [17, 18]. This PPO algorithm has become one of the most widely used algorithms in RL due to its better sample efficiency than other tabular-based RL algorithms.

[1] If the base station has prior information such as the incoming traffic from the UEs or the AoI evolution, the base station scheduler can find the optimal scheduling policy by conventional algorithms.

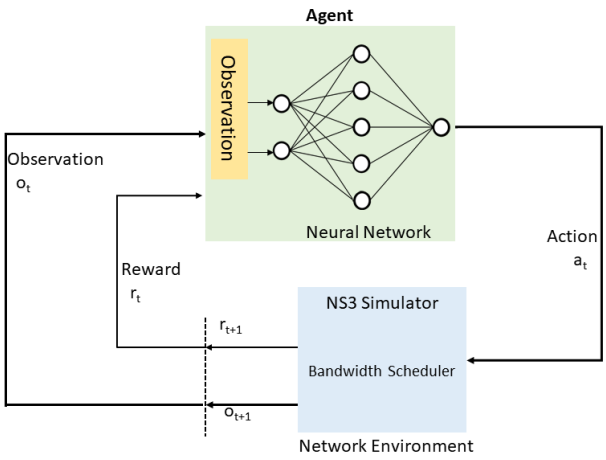

Fig. A.2: A Flowchart of the Reinforcement Learning Method

As shown in Fig. A.2, the learning process is a finite loop of K iterations. In the beginning, the agent will fetch an initial environment observation, $o_{t=0} = \{o_1(t=0)), o_2(t=0)), \ldots o_n(t=0))\}$, from the network environment. The agent observes a set of metrics from $o_n(t)$ including the buffer status, AoI value of every UE, and the throughput achieved in the last k iterations. Then the agent feeds these values to the neural network, which will output the next action. The next action is defined by which UEs are to be chosen for the next iteration $k+1$, as well as how much bandwidth they will be allocated, at time slot $t+1$. The scheduling policy is transformed from the action obtained from the trained neural network. If a UE is selected to transmit packets then the corresponding bandwidth will be reserved for the UE. After the new scheduling is deployed to all the UEs, a reward is observed and fed back to the agent. The agent uses the reward information to train and improve its neural network model.

Our implementation of the PPO algorithm in the scheduling problem is detailed in Algorithm 1. Starting from the initial parameters, the PPO algorithm optimizes its policy, π, until converges or reaches K iterations. At each iteration k, the PPO agent collects observation of a time slot. Next, it selects an action with the current policy. After the agent takes the selected action, the agent obtains a reward based on the reward function. The reward function is defined as $R(o_t, a_t)$. In addition, we formulate a Q-value function, a value function and an advantage function which use to compute the intermediate values in each iteration:

$$Q_\pi(o_t, a_t) = \mathbb{E}_{o_{t+1}, a_{t+1}}[\gamma^t R(o_{t+1}) | o_t = o_0] \tag{A.7}$$

$$V_\pi(o_t) = \mathbb{E}_{o_{t+1},a_t}[\gamma^t R(o_{t+1})|o_t = o_0] \tag{A.8}$$

where γ is a discount factor, $\gamma \in [0,1]$, and

$$A_\pi(o_t, a_t) = Q_\pi(o_t, a_t) - V_\pi(o_t) \tag{A.9}$$

Then we construct the surrogate loss on these observations and optimize policy with SGD for e epochs and minibatch size $\mathcal{B}$.

Algorithm 1: Proximal Policy Optimization Algorithm

Input: An initial policy with parameters θ_0 and initial observation o_0
for $k = 1, 2, 3, \cdots$ *until* $k = K$ *or convergence* **do**
  Update age and bandwidth request based on observation o_k.
  Take scheduling action using policy $\pi = \pi(\theta_k)$.
  Compute advantage estimation based on the value function.
  Optimize surrogate function ∇J with respect to θ_k using e epochs and minibatch size $\mathcal{B}$.
  $\theta_k \leftarrow \theta_{k+1}$
end

5 Simulation

In this section, we provide illustrate the performance of the scheme proposed in Section 4. To evaluate the scheme in a realistic cellular network, the simulation is performed in the network simulator (NS3) [20]. In addition, we select the round-robin algorithm as baseline 1 and a proportional-fair algorithm as baseline 2 [21]. The simulation parameters in NS3 are listed in Table A.1.

Table A.1: Simulation Parameters

Parameter Name	Value
Number of UEs N	20
Slot Duration τ	1 ms
Packet Size (d)	2048 bytes
Numerology	0
Duplexing	TDD
Bandwidth	20 MHz
Transmission Power	20 dBm
Propagation Model	TR 38.901

5.1 Simulation Setup

We revised the NS3 LTE module to implement a 5G environment. The modulation and coding scheme and the resource block allocation are chosen based on the standard [22–26], and [2]. Users are distributed uniformly in the service area of 80*80 meters, while the base station is placed at a fixed location in the service area. The simulation time was chosen to be 900 seconds which ensures that enough training samples were collected. For each UE n, the packet arrival rate is randomly set with a normal distribution which has a mean frequency range between [60Hz, 1300Hz]. The variance of packet arrival rate is 9000 Hz. The packet distribution will only be used in the Proportional-Fair algorithm as a known parameter. Regarding the parameters of the actor, we set the batch size $\mathcal{B} = 80$. Then, we set the step size as 0.6 and use a three-layer DNN with hyperbolic tangent (Tanh) activation function, Adam optimizer, and initial learning rate 0.0003. On the other hand, the neural network structure of critic starts with an action-appended input layer. Then, it connects to a fully connected hidden layer and an output layer with N outputs. The critic also uses the Tanh activation function and Adam optimizer. Additionally, the initial critic learning rate is set to 0.001.

5.2 Simulation Results

The proposed scheme is assessed in terms of the average AoI and average UEs throughput. Based on those two metrics, the following scheduling schemes are compared:

1. Round-Robin (RR), where all RBs are evenly allocated to each UE;
2. Proportional-Fair (PF), where all RBs that are allocated depend on the known arrival traffic distribution;
3. Proximal Policy Optimization (PPO), where all RBs are allocated by the prediction from our RL agent.

We now present our simulation results based on the two baseline algorithms and the proposed PPO algorithm 1. Fig. A.3 shows the average AoI and Fig. A.4 presents the network throughput as functions of the mean traffic generation frequency. The RR algorithm has the worst AoI performance and the lowest throughput due to the full fairness for each UE. The PF algorithm has the best AoI and throughput performance because the data generating distributions over the network nodes are known parameters. It can be seen that the PPO algorithm outperforms the round-robin algorithm at the heavy traffic condition without any proprietary parameters from the UEs. Finally, in the PPO algorithm, the scheduler outperforms the RR algorithm and achieves almost as good performance as the PF algorithm, despite the lack of knowledge of the data generating distributions.

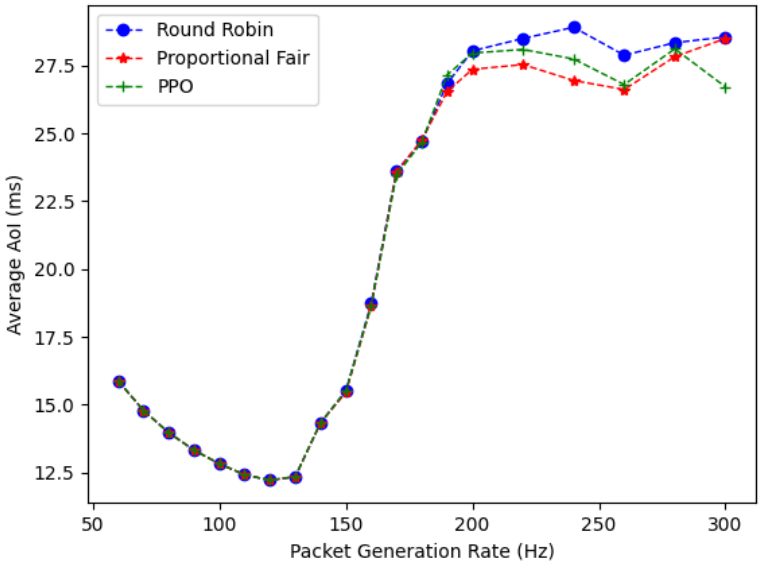

Fig. A.3: Average AoI in different Packet Generation Rates

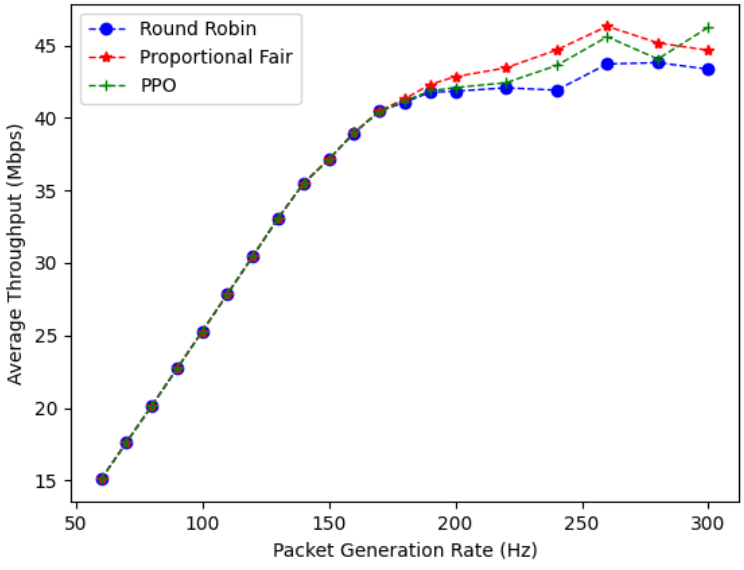

Fig. A.4: Average Throughput in different Packet Generation Rates

5.3 Discussions

If the network bandwidth is larger than the total traffic requirements, there is no network backlog. Thus, any scheduler can achieve the same AoI performance. Therefore, all the algorithms have the same AoI performance at low traffic conditions, as illustrated in Fig. A.3. However, when the total traffic requirements exceed the available bandwidth, the individual traffic from each UE starts backlogging, and AoI starts to increase. This happens in our evaluation when the data generation rate surpasses 120 Hz. The RR algorithm has the worst AoI performance and the lowest throughput because the RR algorithm does not consider the individual traffic load. The PF algorithm has better AoI and throughput performance due to the pre-configured traffic generation distribution. However, in the practical system, the traffic distribution may not be obtained. It can be seen in Fig. A.3 and Fig. A.4 that the PPO algorithm achieves the high throughput while keeping the low AoI without the knowledge of the traffic distribution. Since the PPO algorithm adapts to the traffic generation and AoI evolution, the PPO agent flexibly allocates the network bandwidth.

6 Conclusion

In this paper, we designed a model-free deep reinforcement learning (DRL) method for optimizing the uplink scheduler. DRL agent guided the scheduler to deal with network utilization and UEs age minimization in a scenario with unknown traffic generation. The problem formulation and optimization process of AoI provided a theoretical basis for future studies on next-generation network radio resource management. Moreover, the proposed learning framework of centralized estimation and execution could be deployed in the real network environment. For our future work, we will consider learning-based scheduling in heterogeneous network architectures.

References

[1] J. Doncel and M. Assaad, "Age of information in a decentralized network of parallel queues with routing and packets losses," 2020.

[2] I. Kadota, A. Sinha, and E. Modiano, "Optimizing age of information in wireless networks with throughput constraints," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, 2018, pp. 1844–1852.

[3] 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 Tutorials*, vol. 21, no. 4, pp. 3072–3108, 2019.

[4] C. Kam, S. Kompella, G. D. Nguyen, J. E. Wieselthier, and A. Ephremides, "On the age of information with packet deadlines," *IEEE Transactions on Information Theory*, vol. 64, no. 9, pp. 6419–6428, 2018.

[5] A. Kosta, N. Pappas, A. Ephremides, and V. Angelakis, "Age of information and throughput in a shared access network with heterogeneous traffic," 2018.

[6] Y.-P. Hsu, E. Modiano, and L. Duan, "Age of information: Design and analysis of optimal scheduling algorithms," in *2017 IEEE International Symposium on Information Theory (ISIT)*, 2017, pp. 561–565.

[7] H. B. Beytur and E. Uysal, "Age minimization of multiple flows using reinforcement learning," in *2019 International Conference on Computing, Networking and Communications (ICNC)*, 2019, pp. 339–343.

[8] A. M. Bedewy, Y. Sun, S. Kompella, and N. B. Shroff, "Optimal sampling and scheduling for timely status updates in multi-source networks," 2020.

[9] Q. He, D. Yuan, and A. Ephremides, "Optimal link scheduling for age minimization in wireless systems," *IEEE Transactions on Information Theory*, vol. 64, no. 7, pp. 5381–5394, 2018.

[10] R. D. Yates and S. K. Kaul, "Status updates over unreliable multiaccess channels," in *2017 IEEE International Symposium on Information Theory (ISIT)*, 2017, pp. 331–335.

[11] J. Zhong, E. Soljanin, and R. D. Yates, "Status updates through multicast networks," in *2017 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2017, pp. 463–469.

[12] J. Gang and V. Friderikos, "Optimal resource sharing in multi-tenant 5g networks," in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE Press, 2018, pp. 1–6. [Online]. Available: https://doi.org/10.1109/WCNC.2018.8377326

[13] M. A. Abd-Elmagid and H. S. Dhillon, "Average peak age-of-information minimization in uav-assisted iot networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 2003–2008, 2019.

[14] I. Kadota, A. Sinha, E. Uysal-Biyikoglu, R. Singh, and E. Modiano, "Scheduling policies for minimizing age of information in broadcast wireless networks," 2018.

[15] E. T. Ceran, D. Gündüz, and A. György, "A reinforcement learning approach to age of information in multi-user networks," in *2018 IEEE 29th*

Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), 2018, pp. 1967–1971.

[16] S. Kaul, M. Gruteser, V. Rai, and J. Kenney, "Minimizing age of information in vehicular networks," in *2011 8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, 2011, pp. 350–358.

[17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[18] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," 2018.

[19] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. M. A. Eslami, M. Riedmiller, and D. Silver, "Emergence of locomotion behaviours in rich environments," 2017. [Online]. Available: https://arxiv.org/abs/1707.02286

[20] "Network simulator 3," https://www.nsnam.org.

[21] M. Kaneko, P. Popovski, and J. Dahl, "Proportional fairness in multi-carrier system with multi-slot frames: upper bound and user multiplexing algorithms," *IEEE Transactions on Wireless Communications*, vol. 7, no. 1, pp. 22–26, 2008.

[22] "3gpp tr 38.913 study on scenarios and requirements for next generation access technologies," 2017.

[23] "3gpp tr 36.839 evolved universal terrestrial radio access; mobility enhancements in heterogeneous networks," 2013.

[24] "3gpp ts 24.501 5g; non-access-stratum (nas) protocol for 5g system (5gs); stage 3," 2020.

[25] "3gpp ts 38.331, "nr - radio resource control (rrc) protocol specification -release 16," 2020.

[26] "3gpp ts 38.300, nr; overall description; stage-2," 2019.

Paper B

AoI and Throughput Optimization for Hybrid Traffic in Cellular Uplink Using Reinforcement Learning

Chien-Cheng Wu; Zheng-Hua Tan; Cedomir Stefanovic

The paper has been published in the
IEEE 95th Vehicular Technology Conference: (VTC2022-Spring), pp. 1–6, 2022.

The layout has been revised.

Abstract

The fast growth of time-sensitive wireless applications motivates us to optimize the network radio scheduling algorithm. We consider a joint periodic and burst traffic scheduling problem over a 5G network and design a reinforcement learning method for the age of information and throughput optimization. The periodic traffic flow is generated with a fixed frequency, and the Poisson Pareto Burst Process generates the burst traffic flow. We firstly formulate the scheduling problem as a non-linear integer programming problem. Then, we focus on the reinforcement learning method modelling and solve it via the Proximal Policy Optimization algorithm. Our simulation shows that the suggested reinforcement algorithm outperforms the classical algorithms without any prior knowledge of the arriving traffic.

1 Introduction

Rapidly growing time-sensitive applications, such as virtual reality, autonomous vehicles, and tactile internet, raise the importance of network latency in 5G and beyond 5G mobile communications [1] [2]. One of 5G pillar technologies, Ultra-Reliable Low-Latency Communication (URLLC) requires fresh data arrival at the destination within a certain period [3]. Due to stringent requirements in terms of latency, the network is requested to keep the data at the destination as fresh as possible. The data "freshness" is evaluated by a new metric called Age of Information (AoI) [3]. AoI is defined as the time elapsed since the generation of the status update that was most recently received by a destination.

In time-sensitive multi-users applications, it is of paramount importance to optimize resource-allocation such that AoI is minimized across users. A pivotal aspect to consider in this regard is the nature of the traffic arrivals in the application, where the basic categories are (quasi-)periodic and burst traffic. Fig. B.1 depicts such a mobile virtual reality scenario that users generate the burst traffic from their motions and sensors on their VR glass generate the periodic traffic. Note that most of the available literature on resource allocation that optimizes AoI assumes homogeneous traffic arrivals; a notable exception is presented in [4].

Another key parameter of interest in multi-user scenarios is the network throughput, reflecting the efficiency of the resource use. In practical scenarios, throughput should be taken along the latency-related metrics when optimizing the resource allocation [5]. In the context of AoI, resource allocation that also includes throughput considerations has been seldom investigated.

In this work, we consider a network that serves multiple, time-sensitive users that generate packets to be transmitted in the uplink according to a hybrid arrival model. Each user asynchronously communicates with the base

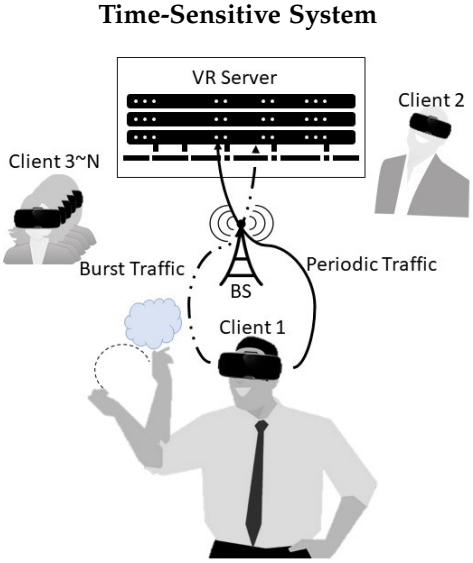

Fig. B.1: Periodic and Burst Traffic over Mobile VR Scenario

station to obtain transmission grants and the corresponding radio resources for a specific transmission. Our goal is to minimize the average AoI while maximizing the network throughput. A key problem is how to allocate communication resources such that the goal is met. To this end, we propose a proactive radio resource allocation method dispatching radio resources according to traffic prediction.

Unlike traditional radio resource allocation, our solution does not need to gather historical traffic information and analysis offline. In contrast, the base station gradually learns the required radio resources online, relying on exploration and exploitation in the context of reinforcement learning. Specifically, we propose that base stations utilize the observations from clients as a guide when making a final resource selection. Based on the observations, base stations gradually improve the AoI and throughput performance over the mobile network. To sum up, our contribution in this paper is to design an age-throughput scheduler using Reinforcement Learning (RL), suitable for the multi-source hybrid traffic scenario with no provision of the traffic models at the server.

The rest of this paper is organized as follows. Section 2 presents a brief overview of the related work. The system model and the problem formulation are described in Section 3. The proposed learning algorithm is presented in Section 4, whereas the numerical simulations are given in Section 5. Finally, the conclusions are drawn in Section 6.

2 Related Work

Recently, AoI modeling and optimization problems for time-sensitive services have been extensively studied [3] [6]. The existing analytical models formulate AoI optimization by considering scheduling policies, packet generation control, or queue management [3]. Especially for the AoI optimization problem under throughput constraints [7], [8], prior AoI studies can be classified into two main categories, single-source traffic and multi-source traffic.

2.1 Single-Source Traffic

The optimal scheduling for a single-source traffic system had been well-studied. The first AoI paper [6] considered a single-source traffic scenario with different queue models, FCFS (First Come First Serve) infinite buffer. The authors in [9] obtain a gamma distribution from Average AoI and Peak AoI at a single-source node network with an LCFS (Last Come First Serve) queue. Average Peak AoI expressions are derived for a queueing system with packet transmission errors with various buffer management schemes in [10]. Expressions for the steady-state distributions of AoI and Peak AoI are derived in [11] for a wide range of single-source systems.

2.2 Multi-Source Traffic

The optimal scheduling for a multi-source traffic system had been discussed in recent research. The analytical results are obtained and the proposed algorithm achieves a desirable optimal solution among known traffic assumptions. At [7], the authors propose an optimal transmission scheduling policy to minimize the average AoI with throughput constraints in a multi-source system with an always-on traffic assumption. The authors in [12] consider the problem of minimizing the age of information in a multi-source system and they show that for any given sampling strategy, the Maximum Age First (MAF) scheduling strategy provides the best age performance among all scheduling strategies. In the paper [13], an age-based scheduler was proposed to combine age with the interarrival times of incoming packets in its scheduling decisions; consequently, the scheduler can achieve improved information freshness at the receiver. The authors of [3] and [8] consider a network serving multi-source traffic using per-source queueing under the assumption of synchronized and random information packet arrivals, respectively, and propose a nearly optimal scheduler. For the particular random arrivals as this paper, the authors in [3] analyze a stationary randomized policy for the single-buffer case with optimal scheduling probabilities depending on the source weights and source success probabilities through a square-root

relationship. Moreover, they propose an age-based Max-Weight scheduler for the same system whose performance is better and close to the lower bound.

3 System Model

3.1 Network Model

We assume a time slotted 5G network consists of N UEs which are served by a single base station (BS). Every UE runs a time-sensitive application that generates uplink traffic according to the model described in 3.2. Without loss of generality, we assume that each UE $n \in N$ has a limited size of uplink queue. Let c denote the size of uplink queue in bytes. Each UE notifies the BS about the size of uplink queue via the 5G New Radio control signal [14] when requesting uplink transmission in the beginning of a time slot. The BS allocates the corresponding radio resources to UEs via its resource allocation algorithm and then reply to the UEs the control information regarding the granted radio resources. The total radio resources in the BS defined as the total B bytes. Due to the radio resource limitation, the BS only allows a subset of m, $0 < m \leq N$, of UEs at a timeslot t to send packets to the server.

3.2 Traffic Model

Data packets that used for time-sensitive services considered in this paper are divided into two types. The first type refers to periodic packets, containing some sensory information that should be regularly reported and whose generation instances are controlled by the users. Specifically, user n generates periodic packets with a fixed frequency f_n, whose value is drawn from a Gaussian distribution and which are independently and identically distributed over users. The size of the periodic packets is also fixed on user basis, and its value for each user is uniformly randomly drawn in the interval $[1, d]$ bytes. Denoting by $x_n(t)$ the size of the periodic data generated by user n in slot t, we have

$$x_n(t) = \begin{cases} d_n & \text{if } \dfrac{t}{f_n} \in \mathbb{Z}^+ \\ 0 & \text{if } \dfrac{t}{f_n} \notin \mathbb{Z}^+ \end{cases} \tag{B.1}$$

where $d_n \sim \mathcal{U}(1, d)$.

The other type of packets are burst packets; the bursts are occurring according to a Poisson Pareto Burst Process (PPBP) model developed in [15]. The model's parameters are $(r; v; l)$. Specifically, bursts arrive according to a Poisson process with rate r, and their lengths are independent and identically

distributed random variables following a Pareto distribution with parameter v. Finally, parameter l specifies the data packet length l, which is a constant value, same for all users. The arrival rate of the burst traffic λ generated from a PPBP model can be calculated from the variables above:

$$\lambda = v \times r \times l \tag{B.2}$$

We denote $x'_n(t)$ as the size of the bursty data generated by user n in slot t,[1] where

$$x'_n(t) = \begin{cases} l & \text{user } n \text{ has a burst packet arrival in slot } t \\ 0 & \text{otherwise} \end{cases} \tag{B.3}$$

The total amount of data generated by user n in time slot t is $x_n^{\text{tot}}(t) = x_n(t) + x'_n(t)$, following a memoryless arrival process that is independent and identically distributed over users. The generated packets are stored at UE transmitters' queues, following the FCFS policy. The queue length is fixed to c and initialized to 0.

3.3 Age of Information

The AoI of each UE is defined as the time elapsed between the current time and the generation of the latest packet that departed from its transmitter; we assume that all transmitted packets are successfully received.

We express AoI in number of slots. Specifically, the value of AoI is updated after every slot in the following way. If no packet from UE n is received by the server in time slot t, AoI is increased by 1. Formally

$$A_n(t) = A_n(t-1) + 1 \tag{B.4}$$

where $A_n(t)$ denotes the value of AoI of user n. Otherwise, if at time slot t, a packet from UE n is successfully received by the server, the AoI will be updated as follows

$$A_n(t) = t - t_n^G(i) \tag{B.5}$$

where $t_n^G(i)$ is the generation moment of the packet that was received in slot t (denoted by the dummy index i).

3.4 Hybrid Traffic Multiplexing

Given N UEs with different PPBP model parameters and periodic frequency f_n, we assume that each UE has a fixed length queue to store a mix of periodic and burst traffic. The packet enqueue also follows the FCFS policy. Each UE

[1] We assume that only a single burst packet can be generated in a slot, which can be justified by a small slot duration. Effectively, the burst duration is then given in number of slots which is determined by the burst duration.

transmits its packets to BS based on the allocated time-slots from the BS scheduler. If an UE obtains transmission time slots, the UE dequeues at least one packet to transmit with a first-in-first-out (FIFO) rule. Otherwise, the UE needs to wait for next time-slots allocation. If the queue was full, all the incoming packets will be dropped until empty spaces available. In addition, given a link-layer transmitter buffer size at each UE, the schedule must ensure that no transmitter buffer violations happen at any UE. For instance, if a UE was selected to transmit, it must have data in its buffer to be transmitted.

3.5 Radio Resource Allocation Algorithm

The radio resource allocation algorithm aims to schedule the BS radio resource blocks to UEs. The scheduling decisions are acknowledged to UEs right after the uplink transmission requests from the UEs. We define that $S_n(t)$ is a random variable which indicates the UE n selection result at the time slot t. To be more specific, $S_n(t) = 1$ if the BS selects UE n at time slot t, otherwise $S_n(t) = 0$. When $S_n(t) = 1$, UE n transmits a subset of the packets to the server via the scheduled resource blocks. If an user is selected for transmission at slot t, the allocated channel bandwidth at time slot t to UE n is denoted by $b_n(t)$, where $b_n(t)$ is an integer number of bytes, $0 < b_n(t) \leq B$ and $\sum_{n \in \{\mathcal{S}(t)=1\}} b_n(t) = B$, where n is the index of selected users in slot t. A feasible schedule at slot t is denoted by $\mathcal{S}(t) = \{S_1(t), S_2(t), \ldots S_n(t), \ldots S_N(t)\}$, which belongs to the set of all schedules $\mathbb{S}$.

4 Proposed Model and Algorithm

The goal of our radio resource allocation algorithm is to find an optimal transmission schedule of all clients to maximize the system-wide utility (i.e. throughput) as well as minimize the average AoI. Formally, the goal is to select a schedule $\mathcal{S}(t)$ in each time slot t such that

$$\mathcal{S}(t) = \underset{\mathcal{S}(t) \in \mathbb{S}}{\operatorname{argmax}} \sum_{n=1}^{N} [U_n(t) - \beta A_n(t)] \tag{B.6}$$

where $\beta \in \mathbb{R}$ is a control parameter; $U_n(t)$ is the utility of UE n at time slot t.

The UE and network utility per slot are defined as:

$$U_n(t) = \frac{1}{1 + e^{-(1.5 \times b_n(t) - X_n(t))}} S_n(t),\ 0 \leq n \leq N \tag{B.7}$$

$$U(t) = \sum_{n=1}^{N} U_n(t) \tag{B.8}$$

The optimization problem in (B.6) is a non-linear integer programming problem [16]. Further, the optimization variables in (B.6) include sequential decisions, which invokes the idea to apply an RL method to solve (B.6). In this respect, deep reinforcement learning (DRL) was shown to achieve a high performance on the long-term sequential decision-making problems without supervision [17–19]. Moreover, DRL-based solutions can provide the decisions in an automatic and zero-touch manner. In addition, DRL is capable of handling high-dimensional observation-action spaces. These motivate us to propose a policy-based model-free DRL solution to (B.6), as elaborated next.

We assume a reinforcement learning (RL) agent interacting with a network environment to learn a scheduling policy without prior information.[2] To deal with the unknown information in the network environment, an RL agent gradually learns the scheduling policy by constructing the policy parameters θ_t from the network environment observations o_t.

In addition, we define the scheduling policy as $\pi(o_t, \theta_t)$ that returns a schedule $\mathcal{S}(t)$ as an action a_t to satisfy $\sum_{n=1}^{N} b_n(t) \leq B$. The action a_t represents which UEs are selected to transmit their packets. For example, if only $S_1(t) = S_2(t) = 1$ in $\mathcal{S}(t)$, it means that only UE1 and UE2 are selected to transmit data with their allocated bandwidth $b_1(t)$ and $b_2(t)$. The other UEs are not allowed to transmit data and have to wait for the next action. In the time-slotted system, every interaction between the agent and the network environment happens at the beginning of a time slot. In each interaction, the agent samples the environment to get an observation o_t and performs an action a_t based on the scheduling policy. After performing the action, the agent receives a reward r_t. Then the agent waits for the time slot $t+1$ to interact with the network environment. The learning process repeats the interactions continuously to approximate the optimal scheduling policy which obtains the maximal reward. Hence, the objective of learning is to maximize the expected cumulative reward.

The optimization goal is defined as

$$J := \max_{\pi_k} \mathbb{E}\left[\sum_{n=1}^{N} \left(U_n(t) - \beta A_n(t)\right)\right] \tag{B.9}$$

$$\text{s.t. } \sum_{n=1}^{N} b_n(t) \leq B,\ \pi_k = \mathcal{S}(t)$$

Our RL agent uses the Proximal Policy Optimization (PPO) Algorithm [17, 18]. This PPO algorithm has become one of the most widely used algorithms in RL due to its better sample efficiency than other tabular-based RL algorithms.

[2] If the base station has prior information such as the incoming traffic from the UEs or the AoI evolution, the base station scheduler can find the optimal scheduling policy by conventional algorithms.

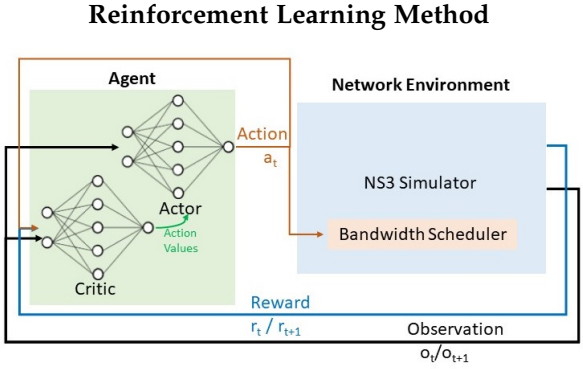

Fig. B.2: A Flowchart of the Reinforcement Learning Method

As shown in Fig. B.2, the learning process is a finite loop of K iterations. In the beginning, the agent will fetch an initial environment observation, $o_{t=0} = \{o_1(t=0)), o_2(t=0)), \ldots o_n(t=0))\}$, from the network environment. The agent observes a set of metrics from $o_n(t)$ including the buffer status, AoI value of every UE, and the throughput achieved in the last k iterations. Then the agent feeds these values to the neural network, which will output the next action. The next action is defined by which UEs are to be chosen for the next iteration $k+1$, as well as how much bandwidth they will be allocated, at time slot $t+1$. The scheduling policy is transformed from the action obtained from the trained neural network. If a UE is selected to transmit packets then the corresponding bandwidth will be reserved for the UE.

After the new scheduling is deployed to all UEs, a reward is observed and fed back to the agent. The agent uses the reward information to train and improve its neural network model.

Our implementation of the PPO algorithm in the scheduling problem is detailed in Algorithm 2. Starting from the initial parameters, the PPO algorithm optimizes its policy, π, until converges or reaches K iterations. At each iteration k, the PPO agent collects observation of a time slot. Next, it selects an action with the current policy. After the agent takes the selected action, the agent obtains a reward based on the reward function. The reward function is defined as $R(o_t, a_t)$. In addition, we formulate a Q-value function, a value function and an advantage function which use to compute the intermediate values in each iteration:

$$Q_\pi(o_t, a_t) = \mathbb{E}_{o_{t+1}, a_{t+1}}[\gamma^t R(o_{t+1}) | o_t = o_0] \tag{B.10}$$

$$V_\pi(o_t) = \mathbb{E}_{o_{t+1}, a_t}[\gamma^t R(o_{t+1}) | o_t = o_0] \tag{B.11}$$

where γ is a discount factor, $\gamma \in [0, 1]$, and

$$A_\pi(o_t, a_t) = Q_\pi(o_t, a_t) - V_\pi(o_t) \tag{B.12}$$

Then we construct the surrogate loss on these observations and optimize policy with stochastic gradient descent (SGD) for e epochs and minibatch size $\mathcal{B}$.

Algorithm 2: Proximal Policy Optimization Algorithm

Input: An initial policy with parameters θ_0 and initial observation o_0
for $k = 1, 2, 3, \cdots$ *until* $k = K$ *or convergence* **do**
  Update age and bandwidth request based on observation o_k.
  Take scheduling action using policy $\pi = \pi(\theta_k)$.
  Compute advantage estimation based on the value function
  Optimize surrogate function ∇J with respect to θ_k using e epochs and minibatch size $\mathcal{B}$.
  $\theta_k \leftarrow \theta_{k+1}$
end

5 Simulation

The performance of our method presented in Section 4 by simulation. The simulation is implemented using the network simulator (NS3) [20] to obtain the system-level performance. Furthermore, we use a round-robin method as baseline 1 and a proportional-fair approach as baseline 2 [21]. Table B.1 lists the simulation parameters in NS3.

Table B.1: Simulation Parameters

Parameter Name	Value
Number of UEs N	20
Slot Duration τ	1 ms
Periodic Packet Size (d)	2048 bytes
Burst Packet Size (l)	1024 bytes
Numerology	0
Duplexing	TDD
Bandwidth	20 MHz
Transmission Power	20 dBm
Propagation Model	TR 38.901

5.1 Simulation Environment

To construct a 5G environment, we updated the NS3 LTE module. The modulation and coding schemes, as well as the assignment of resource blocks

(RBs), are all based on the standard [22–24]. The base station is installed at a permanent point in the service area, and users are scattered evenly in an 80*80 meter service area. The N users will be randomly assign high-load burst traffic according to the ratio of high-load users in total users. A simulation time was set to 900 seconds guarantee that sufficient training samples were gathered. The periodic packet arrival rate for each UE n is a random variable using a normal distribution with a mean frequency range between [60Hz, 1300Hz]. The periodic packet arrival rate variance is 9000 Hz. On the other hand, the mean burst traffic duration is $v = 0.6$ seconds. Both of the periodic and burst packet distribution will be used as a known parameter only in the Proportional-Fair algorithm. In terms of the actor's parameters, we set the batch size to $\mathcal{B} = 80$. Then, to implement the Proximal Policy Optimization (PPO) algorithm, we are using a three-layer deep neural network (DNN) with hyperbolic tangent (Tanh) activation function, Adam optimizer, and initial learning rate of 0.0003. Critic's neural network topology, on the other hand, begins with an action-appended input layer. It then links to a fully connected hidden layer and an N-output output layer. The critic also uses the Tanh activation function and Adam optimizer; the initial critic learning rate is set to 0.001.

5.2 Simulation Results

We compare the performance difference between our method and the following two scheduling algorithms in terms of the average network AoI and average network throughput.

1. Round-Robin (RR), where all RBs are evenly allocated to each UE;
2. Proportional-Fair (PF), where all RBs that are allocated depend on the known arrival traffic distribution;
3. Proximal Policy Optimization (PPO), where all RBs are allocated by the prediction from our RL agent.

We now present our simulation results based on the two baseline algorithms and the proposed PPO algorithm 2. Fig. B.3 shows the average AoI and Fig. B.4 presents the network throughput as functions of the mean traffic generation frequency. The RR algorithm has the worst AoI performance and the lowest throughput due to the full fairness for each UE. The PF algorithm has better AoI and throughput performance than RR algorithm because the data generating distributions over the network nodes are known parameters. It can be seen that the PPO algorithm outperforms the RR and PF algorithms at the heavy traffic condition without any proprietary parameters from the

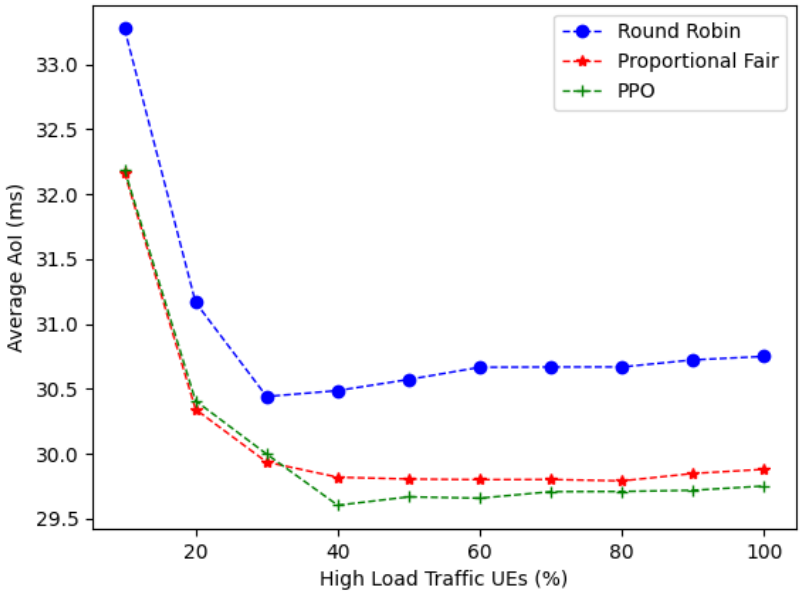

Fig. B.3: Average AoI in different Packet Generation Rates

UEs. Finally, in the PPO algorithm, the scheduler outperforms the RR algorithm and achieves better performance than the PF algorithm, despite the lack of knowledge of the data generating distributions.

5.3 Discussions

If the BS can provide enough radio resources per time slot to satisfy the traffic requirements from all UEs, there is no network backlog. However, if the number of heavy-load UE increases, the network performance becomes worse due to congestion. As illustrated in Fig. B.3, while the congestion condition happened, the RR scheduler allocated bandwidth sequentially that caused the worse AoI and throughput performance under network backlog. From our simulation, when the ratio of heavy traffic UEs achieves 20%, the RR algorithm has the lowest throughput and the highest AoI performance because the RR algorithm does not allocate radio resources reflecting the individual traffic load. The PF algorithm has a better AoI and throughput performance than the RR algorithm due to the proper radio resource allocation from the known network traffic distribution. But in the practical system, it is difficult to obtain the timely traffic distribution in advance. It can be seen in Fig. B.3

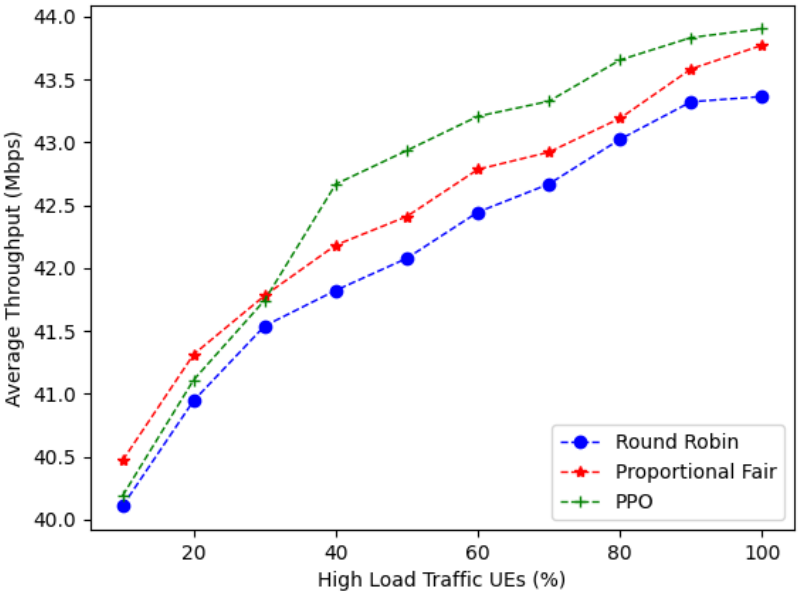

Fig. B.4: Average Throughput in different Packet Generation Rates

and Fig. B.4 that the PPO algorithm achieves the highest throughput while keeping the lowest AoI without the knowledge of the traffic distribution. The PPO agent can approximate the optimal network bandwidth allocation because the agent predicts the AoI evolution and future traffic generation and mitigate the network backlogging.

6 Conclusion

The wireless resource allocation method is investigated in this study in the context of the AoI and utility trade-off. The suggested technique uses deep reinforcement learning to gradually learn from previous transmissions in order to optimize the clients' radio resource occupancy decisions. In addition, a strategy model combining neural networks and reinforcement learning was developed. When compared to a decision algorithm with known traffic patterns, this strategy model can effectively minimize the AoI. The simulation also demonstrates that this technique outperforms the two benchmarks in terms of the minimum AoI and the maximum throughput. Our next step is to explore switching alternative resource management techniques to build a full solution for proactive network resource management based on diverse network conditions. In the future, we'll look at optimizing models for various reinforcement learning approaches depending on the data available on the network.

References

[1] S. K. Kaul, R. D. Yates, and M. Gruteser, "Status updates through queues," in *2012 46th Annual Conference on Information Sciences and Systems (CISS)*, 2012, pp. 1–6.

[2] M. Kaneko, P. Popovski, and J. Dahl, "Proportional fairness in multi-carrier system with multi-slot frames: upper bound and user multiplexing algorithms," *IEEE Transactions on Wireless Communications*, vol. 7, no. 1, pp. 22–26, 2008.

[3] R. D. Yates, Y. Sun, D. R. Brown, S. K. Kaul, E. Modiano, and S. Ulukus, "Age of information: An introduction and survey," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 5, pp. 1183–1210, 2021.

[4] M. K. Abdel-Aziz, C.-F. Liu, S. Samarakoon, M. Bennis, and W. Saad, "Ultra-reliable low-latency vehicular networks: Taming the age of information tail," in *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1–7.

[5] N. Bhushan, J. Li, D. Malladi, R. Gilmore, D. Brenner, A. Damnjanovic, R. T. Sukhavasi, C. Patel, and S. Geirhofer, "Network densification: the dominant theme for wireless evolution into 5g," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 82–89, 2014.

[6] C. She, C. Yang, and T. Q. S. Quek, "Joint uplink and downlink resource configuration for ultra-reliable and low-latency communications," *IEEE Transactions on Communications*, vol. 66, no. 5, pp. 2266–2280, 2018.

[7] "3gpp ts 38.300, nr; overall description; stage-2," September 2019.

[8] Y.-P. Hsu, E. Modiano, and L. Duan, "Age of information: Design and analysis of optimal scheduling algorithms," in *2017 IEEE International Symposium on Information Theory (ISIT)*, 2017, pp. 561–565.

[9] M. Eisen, C. Zhang, L. F. O. Chamon, D. D. Lee, and A. Ribeiro, "Learning optimal resource allocations in wireless systems," *IEEE Transactions on Signal Processing*, vol. 67, no. 10, pp. 2775–2790, 2019.

[10] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017. [Online]. Available: https://arxiv.org/abs/1707.06347

[11] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," 2015. [Online]. Available: https://arxiv.org/abs/1506.02438

[12] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. M. A. Eslami, M. Riedmiller, and D. Silver, "Emergence of locomotion behaviours in rich environments," 2017. [Online]. Available: https://arxiv.org/abs/1707.02286

[13] M. A. Abd-Elmagid and H. S. Dhillon, "Average peak age-of-information minimization in uav-assisted iot networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 2003–2008, 2019.

[14] R. D. Yates, "The age of information in networks: Moments, distributions, and sampling," *IEEE Transactions on Information Theory*, vol. 66, no. 9, pp. 5712–5728, 2020.

[15] A. Kosta, N. Pappas, and V. Angelakis, 2017.

[16] R. D. Yates, E. Najm, E. Soljanin, and J. Zhong, "Timely updates over an erasure channel," in *2017 IEEE International Symposium on Information Theory (ISIT)*, 2017, pp. 316–320.

[17] E. T. Ceran, D. Gündüz, and A. György, "A reinforcement learning approach to age of information in multi-user networks," in *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2018, pp. 1967–1971.

[18] R. D. Yates and S. K. Kaul, "The age of information: Real-time status updating by multiple sources," *IEEE Transactions on Information Theory*, vol. 65, no. 3, pp. 1807–1827, 2019.

[19] S. Kaul, R. Yates, and M. Gruteser, "Real-time status: How often should one update?" in *2012 Proceedings IEEE INFOCOM*, 2012, pp. 2731–2735.

[20] "Network simulator 3." [Online]. Available: https://www.nsnam.org

[21] A. Kosta, N. Pappas, A. Ephremides, and V. Angelakis, "Age of information and throughput in a shared access network with heterogeneous traffic," in *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1–6.

[22] "3gpp ts 24.501 5g; non-access-stratum (nas) protocol for 5g system (5gs); stage 3," January 2020.

[23] "3gpp ts 38.331, nr - radio resource control (rrc) protocol specification -release 16," 2020.

[24] "3gpp ts 38.300, nr; overall description; stage-2," September 2019.

Paper C

Multi-Objective Provisioning of Network Slices using Deep Reinforcement Learning

Chien-Cheng Wu; Vasilis Friderikos; Cedomir Stefanovic

The paper has been published in the
Arxiv, pp. 1–15, 2022.

The layout has been revised.

Abstract

Network Slicing (NS) is crucial for efficiently enabling divergent network applications in next-generation networks. Nonetheless, the complex Quality of Service (QoS) requirements and diverse heterogeneity in network services entail high complexity for Network Slice Provisioning (NSP) optimization. The legacy optimization methods are challenging to meet various low latency and high-reliability requirements from network applications. To this end, we model the real-time NSP as an Online Network Slice Provisioning (ONSP) problem. Specifically, we formulate the ONSP problem as an Multi-Objective Integer Programming Optimization (MOIPO) problem. Then, we approximate the solution to the MOIPO problem by applying the Proximal Policy Optimization (PPO) method to the traffic demand prediction. Our simulation results show the effectiveness of the proposed method compared to the state-of-the-art methods with a lower SLA violation rate and network operation cost.

1 Introduction

Network Slicing (NS) is essential in the next-generation mobile wireless networks [1]. It enables efficient connectivity to various services with diverse requirements by instantiating multiple logical networks on top of the substrate, i.e., the physical network infrastructure. Note that some emerging 5G services, such as those related to the Ultra-Reliable Low Latency Communication (URLLC), require dedicated network resources to achieve the stringent quality of service (QoS) requirements. NS can offer dedicated network resources for multiple network services mapped and managed over a physical wireless network infrastructure [2]. In that respect, a network slice can be considered as a self-contained logical network with its physical network resources, topology and traffic flows with established QoS requirements [3].

In addition, Virtual Network Functions (VNFs) bring higher levels of flexibility as VNFs can be anchored at different network locations and scaled flexibly with NS to meet the fluctuating user traffic demands, thus allowing for efficient on-demand Network Slice Provisioning (NSP). A network slice incorporates a set of VNFs organized in different suitable locations across the transportation paths and depending on the needs of the service. However, real-time and high-quality resource provisioning for multiple network slices is a formidable task, proved to be NP-hard [4].

In order to address this hurdle, the Cloud-native Network Functions (CNFs) have been actively considered and standardized within the 3rd Generation Partnership Project (3GPP) [5]. According to the latest 3GPP architecture [6–8], We construct slices as CNFs' interconnections in our NS paradigm, which align with the conventional notion of VNFs.

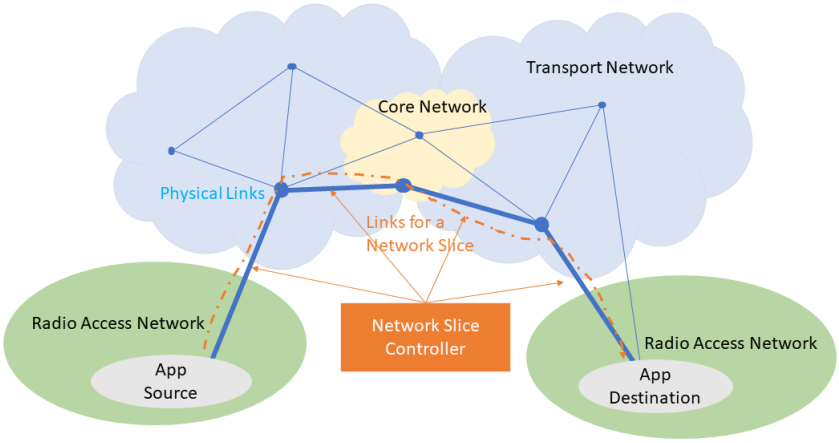

Fig. C.1: Example of a slice constructed from the source node to the destination node.

Within the paradigm, a slice instance should be constructed by related CNFs' interconnections for a user traffic demand request. We further separate the CNFs' interconnections into the data rate and delay arrangements over the physical network infrastructure. Fig. C.1 presents how the network slice controller constructed a network slice via associating network resources with a suitable route. Without loss of generality, a network service provider can append more network resource variables other than the data rates and delays to the paradigm.

This paper investigates efficient slice provisioning as the CNFs' network resource provisioning in the NS paradigm. We focus on the data rate and delay arrangements for the slice provisioning. Our contributions include formulating a multi-objective optimization model and subsequent deep reinforcement learning framework to optimize the network slices provision whilst being robust to unknown traffic demand fluctuation of the users. The robustness in provisioning is achieved by providing a probabilistic guarantee that the amount of provisioned slices will meet the slice QoS requirements. We implement the deep reinforcement learning framework by using Proximal Policy Optimization (PPO) algorithm [9] with two neural networks as a value function and a policy function pair. Under this framework, the network slice controller determines its slice provisioning by the optimal QoS-dependent policy from the PPO algorithm. The problem is then solved by approximately the optimal solution with the joint slices allocation and demands prediction. To illustrate the distinction of the computational performance, we implement a suboptimal approach as a performance benchmark where slice provisioning demands are considered in a batch form, i.e., as several groups sequentially. Both solutions are compared to a nominal provisioning scheme

that only processes the current user demand without considering relations between the slice provisioning demands. Through extensive simulations, we show that our proposed algorithm can reduce the network operation cost while keeping a low Service-Level Agreement (SLA) violation rate.

The rest of this article is structured as follows. Section 2 analyzes related work and highlights our contributions. The mobile network model, user demand requests and the network slices are presented in Section 3. The online NSP problem with uncertainties in the number of users and the associated traffic demands is then formulated in Section 4. Next, we present the existing approaches and our reinforcement learning method to solve the robust NSP problem in Section 5. Section 6 presents numerical results. Finally, Section 7 concludes our perspectives.

2 Related Work

Software-Defined Network (SDN) uses programmable controllers to provide flexible and cost-effective network services in next-generation networks. NSP is a crucial function of SDN for supporting dynamic user demands in different network services. The main challenge for the NSP is to optimize the deployment of slices across the physical infrastructures matching network services constraints. Therefore, we review the existing NSP design and organize the related solutions to the NSP optimization problem.

A slice is realized as a concatenation of communication (wired-cum wireless links) and computing resources that can span across the Radio Access Network, Transport Network, or Core Network [10]. In the context of rule-based solutions, the authors in [11] propose a practical NS implementation. The proposed model provides an efficient solution by analyzing historical NS information but the network slices assigned to the same tenant cannot overlap in time. The research in [12] considers the slice provision with VNF placement for the SDN-based 5G mobile-edge cloud. Their algorithm provides a flexible slice provision by placing VNFs in distributed data centers. Similarly, the work in [13] describes a VNF placement algorithm with emphasis on the mobile core network, exploiting the cost of placement to allocate the VNFs. Their problem formulation takes physical network constraints into account for different network service capacity and connectivity.

On the other hand, the deep reinforcement learning solutions for dynamic demand optimization continuously attract the attention of researchers. The authors of [14] propose a novel framework that uses assured resources which are offered based on the forecast of the user demands. According to the framework, the designed stochastic algorithm can handle the trade-off between the traffic uncertainty in the forecast and overbooking. The work [15] designed a hybrid machine learning model for spatiotemporal prediction,

where an autoencoder models the spatial dependence, and the temporal dependence is captured by a Long Short Term Memory neural network. We also mention [16] that proposes a heuristically-controlled A3C algorithm to demonstrate network slice provision with dynamic traffic.

To the best of our knowledge, this work presents the first attempt to use Proximal Policy Optimization (PPO) algorithm in multi-objective network slice provisioning. More specifically, we formulate the network slice provisioning problem as a multi-objective integer programming optimization problem. Then we propose a deep reinforcement learning framework that allows for online provisioning, which jointly considers the fairness of slice provision and the cost of network operation.

3 System Model

We assume the system is time-slotted. Each action, such as time increment or user demand request arrival, happens at the beginning of a slot and completes before the end of that slot.

Network Model

The network is modeled as an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$, where $\mathcal{V}$ denotes the set of network nodes and $\mathcal{L}$ is the set of undirected wired-cum-wireless links that constitute the deployed physical infrastructure in the network. By $l_{i,j} \in \mathcal{L}$ we denote a link between nodes v_i and v_j ($v_i, v_j \in \mathcal{V}$). The nodes in the graph represent either a radio access network component, e.g., a base-station (BS), a transportation regional network component or a core network element. We assume that all network nodes are virtualized, which means that they are capable of running virtual machines and/or containers over the corresponding physical (bare metal) machine capabilities. Furthermore, we denote the available link capacity as a vector $C = [C_{1,1}, \ldots, C_{i,j}, \ldots, C_{V,V}]$, where each element $C_{i,j} \geq 0$ is the capacity of direct link $l_{i,j}$ between nodes v_i and v_j; if there is no link between v_i and v_j, then the value of $C_{i,j}$ is zero. We also denote the occupied link capacity as a function

$$0 \leq \alpha(i,j) \leq C_{i,j}. \tag{C.1}$$

The delay of the links is given in the vector $D = [D_{1,1}, \ldots, D_{i,j}, \ldots, D_{V,V}]$, where each element $D_{i,j} \geq 0$ express the delay between v_i, and v_j.
The active network operation cost of links in $\mathcal{L}$ is given by the vector $P = [P_{1,1}, \ldots, P_{i,j}, \ldots, P_{V,V}]$, where each element $P_{i,j} \geq 0$ is the active operation cost between nodes v_i and v_j.[1]

[1] If there is no link between v_i and v_j, then we assign the maximum available integer value to $D_{i,j}$ and $P_{i,j}$ in our simulations to represent the disconnected scenario.

User Demand Model

We assume that the user demand requests at a time slot t is a set, denoted by $\mathcal{R}(t)$. Each request r, $r \in \mathcal{R}(t)$, comes from a network application of an user, asking for a network slice template at time slot t. All requests are generated according to a Poisson arrival model determined by a aggregate arrival rate λ (requests/slot). A user demand request is represented as a tuple $r = \{v_s, v_u, b_r, d_r, \mathcal{T}_r, a_r, h_r\}$. where v_s and v_u denote the source and destination nodes respectively, b_r is the required data rate, d_r denotes the delay requirement, and the demand type is expressed as $\mathcal{T}_r$. Further, the initial time slot of the demand is a_r and the demand life time is h_r slots. Thus, the demand will start from the time slot a_r, continue for h_r slots, and becomes terminated at the time slot $t = a_r + h_r$. We assume there are finite number of demand types and network nodes. When the network slice controller receives a request, the controller decides whether to accept or reject it in an online manner.

For example, at a time slot $t = t_1$, request set $\mathcal{R}(t_1)$ is handled by the slice controller and then the appropriate network slices are constructed for the requests. If a request $r \in \mathcal{R}(t_1)$ cannot be served at t_1, it will be stored in a queue until the time reached the time slot ($t = a_r + h_r$). Thus, at any time slot t there are a set of requests, $\mathcal{R}(t)$, that includes both new arrived requests as well the backlogged requests (i.e. the requests that are in the queue, which currently cannot be served and whose termination time has not yet expired).

Network Slice Model

Each slice n can be denoted as

$$n = \{v_{n,s,r}, v_{n,u,r}, E_{n,r}, y_{n,r}, \mathcal{T}_{n,r}\} \tag{C.2}$$

where s is the index of source node $v_{n,s,r}$, u is the index of destination node $v_{n,u,r}$; $(v_{n,s,r}, v_{n,u,r} \in \mathcal{V})$. Further, $E_{n,r}$ is a set of virtual links $E_{n,r} = \{e_{i,j}^{n,r}\}$ with defined provisioning resources. Each virtual link $e_{i,j}^{n,r} \in E_{n,r}$ can be mapped to one and only one physical link $l_{i,j}$ between node v_i and node v_j, and $E_{n,r}$ is a loop-free path, starting from $v_{n,s,r}$ and ending at $v_{n,u,r}$. The loop-free path represents the connectivity constraints of the slice n. The capacity and delay constraints of a virtual link $e_{i,j}^{n,r}$ are the same as the ones of the mapped physical link $l_{i,j}$. The slice load is $y_{n,r}$, which means each $e_{i,j}^{n,r} \in E_{n,r}$ occupies $y_{n,r}$ capacity of link $l_{i,j}$; therefore $e_{i,j}^{n,r} = y_{n,r}$ $y_{n,r} \leq \min_{\forall e_{i,j}^{n,r} \in E_{n,r}} \alpha_{i,j}$. The type of slice is denoted as $\mathcal{T}_{n,r}$ which is allocated according to the demand type. Finally, the slice delay is $\sum_{\forall e_{i,j}^{n,r} \in E_{n,r}} D_{i,j}$, where i, j are in $e_{i,j}^{n,r} \in E_{n,r}$. A feasible slice can only be provided by its slice controller by fulfilling the connectivity, capacity and delay constraints.

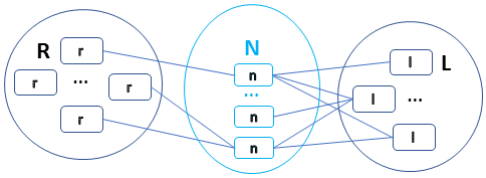

Fig. C.2: Network Slices Mapping Demonstration

4 Problem Formulation

Our goal is to generate network slices that accommodate the underlying user requests in terms of data-rate and latency whilst maximizing the deployment fairness and minimizing the deployment cost, formally defined later in the section. In particular, we avoid network congestion at some links by maximizing the deployment fairness. To this end, we fairly distribute requests to the whole network. We also consider the co-existence of multiple slices on the same physical network and assume that each slice serves network traffic from a single source to a single destination, and therefore we represent a slice by a source-destination pair.

For each user traffic demand r, the NS controller aims to generate an appropriate network slice n to serve r. For example, in Fig. C.2, there is user request set R and link set $\mathcal{L}$. The NS controller aims to construct a set of network slices N to serve R using (a part of) link set $\mathcal{L}$. A request r, $\forall r \in R$, can only be served by a single slice n, $n \in N$, and each n consists of a subset of $\mathcal{L}$. A virtual link $e_{i,j}^{n,r}$ comes from slice n and request r. Moreover, the virtual link $e_{i,j}^{n,r}$ is made by the physical link $l_{i,j}$ between node v_i and node v_j in $\mathcal{L}$.

The slice generation procedure can be decomposed into two steps. The first step is to find potential end-to-end routes which can be utilized for request r from its source node $v_{n,s,r}$ to its destination node $v_{n,u,r}$. The second step is to find a route which can minimize the objective function while satisfying the constraints. To represent the link utilization, we define a utilization matrix X: for each matrix element, $x_{i,j}^{n,r}$, if slice n from request r utilizes link $l_{i,j}$, then $x_{i,j}^{n,r} = 1$, otherwise, $x_{i,j}^{n,r} = 0$. Since multiple slices share the same physical network, the bandwidth allocation at each virtual link shall not exceed the available physical link bandwidth. We formulate the global network constraint as

$$\sum_{\substack{\forall n \in N \\ \forall r \in \mathcal{R}(t)}} x_{i,j}^{n,r} \cdot e_{i,j}^{n,r} \leq C_{i,j}. \tag{C.3}$$

In addition, each virtual route in slice n mapped onto the physical links

should satisfy the requested user traffic volume

$$\min_{\forall e_{i,j}^{n,r} \in E_{n,r}} \alpha(i,j) \geq b_r. \tag{C.4}$$

Moreover, each virtual route in slice n should also satisfy the requested user traffic latency

$$\sum_{\substack{\forall n \in N \\ \forall r \in \mathcal{R}(t)}} x_{i,j}^{n,r} \cdot D_{i,j} \leq d_r. \tag{C.5}$$

From the above constraints and matrices, we formulate the Online Network Slicing Provisioning (ONSP) problem as: *Given the user demand requests, $\mathcal{R}(t)$, how to generate a set of slices N with minimum cost and maximum fairness to transmit the user traffic?*

Specifically, the cost objective function is defined as the summation of all the slices' cost in Eq. (C.6) below.

$$f_1 = \min \sum_{\substack{\forall n \in N \\ \forall l_{i,j} \in \mathcal{L}}} x_{i,j}^{n,r} \cdot P_{i,j} \tag{C.6}$$

The cost optimization strategy tends to prioritize the low-cost virtual links during slice provisioning. However, the priority of low-cost links should be limited within a proper level. Therefore, we refer [17] to define the fairness objective function as the ratio of allocated data rate and available data rate Eq. (C.7) below.

$$f_2 = \max \frac{\left(\sum_{\substack{\forall n \in N \\ \forall l_{i,j} \in \mathcal{L}}} \frac{\alpha(i,j)}{C_{i,j}} \right)^2}{|N| \sum_{\substack{\forall n \in N \\ \forall l_{i,j} \in \mathcal{L}}} \left(\frac{\alpha(i,j)}{C_{i,j}} \right)^2}. \tag{C.7}$$

The ONSP problem is then reformulated as a Multi-Objective Integer Programming Optimization (MOIPO) problem to construct a route for a slice, and Breadth-First Search (BFS) search algorithm is used to find the minimal cost paths through the network. In general, at a timeslot t, given a set of user demand requests, $\mathcal{R}(t)$ as an input, our algorithm aims to generate a set of network slices N and keep the related bandwidth and latency guarantees for every network slice.

5 Proposed Framework and Algorithms

Greedy Approach

The greedy algorithm is used as a less computationally intensive benchmark because it only considers one possible provisioning sequence when allocating

virtual links. The computational complexity is $\mathcal{O}(\mathcal{V} + \mathcal{L} + \mathcal{R})$. Algorithm 3 shows a greedy ONSP algorithm. An undirected graph, $\mathcal{G}$, and user demand requests set, $\mathcal{R}(t)$, are given as input. We use the cardinality of a set such as $|\mathcal{R}(t)|$ and $|\Omega|$ to measure the number of elements in the set $|\mathcal{R}(t)|$ and the set $|\Omega|$ respectively. From the graph $\mathcal{G}$ and the set $\mathcal{R}(t)$, we can obtain the bandwidth constraint functions and latency constraint functions, see Eq. C.3, C.4 and C.5. Then, the greedy algorithm iterates all requests in $\mathcal{R}(t)$ by a loop with finite steps. For each user demand request $r \in \mathcal{R}(t)$, we use the BFS algorithm to search the graph for all possible sets of links that satisfy the connectivity constraint of r.

A set of links can construct a unique path with a fixed sequence from the BFS algorithm. If a path can satisfy the connectivity constraint of r, we define the path is a feasible path. Otherwise, the path is not feasible. Next, we consider all feasible paths and check the link availability of each path based on the given bandwidth and latency constraints. If a link is not available, the corresponding path is skipped. Otherwise, we continue to the next step.

Next, we calculate the cost and fairness values based on the objective functions. After that, we compare the cost value and fairness value in the current step with the cost value in the previous step. If the cost value in the current step is less than the cost value in the previous step and the fairness value in the current step is higher than the fairness value in the previous step, we assign the path ω to be a candidate for slice n with respect to the request r. Next, to generate a slice n by a non-empty candidate. Otherwise, move to the next feasible path. Finally, we aggregate each slice n as a set $\mathcal{N}$ for output.

Integer Programming Approach

Integer Programming (IP) was introduced to model a series of optimization problems. Many algorithms had tried to efficiently solve the Integer Programming problem such as the Constraint Integer Programming (CIP) [18]. In addition, The CIP method had been proven to be able to obtain the optimal solution in [18]. The general form of CIP is the following: given an finite set $\mathcal{H}$ of constraints $h_i : \mathcal{R}^n \rightarrow \mathcal{Z}, i = \{1, \ldots, m\}$, a variable set $\mathcal{X}$, $x \in \mathcal{Z}, \forall x \in \mathcal{X}$, and a vector of objective functions $f \in \mathcal{R}^n$, derive an optimal solution θ from $\mathcal{X}$ if the CIP is satisfiable as $\theta = \min\{f^T\mathcal{X}|h_i(X) = \text{true}, \forall h_i \in \mathcal{H}\}$. Each request at a specific timeslot t in the ONSP problem in Section 4 can be written

Algorithm 3: Greedy Algorithm

Input: A undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$ and user demand requests, $\mathcal{R}$
Output: network slices, $\mathcal{N}$
Sort the set $\mathcal{R}$ by the initial time
for $r = 1, 2, 3, \cdots$ *until* $r = |\mathcal{R}|$ **do**
  Find all feasible paths, Ω, based on the BFS algorithm
  for $\omega = 1, 2, 3, \cdots$ *until* $\omega = |\Omega|$ **do**
    Check the link availability in ω based on the constraint functions.
    Calculate the cost value and fairness value based on the objective functions.
    Compare with the previous cost value and fairness value.
    if *current value is better than previous value* **then**
      Assign the current ω to candidate
    end
  end
  if *candidate is not empty* **then**
    Generate a network slice n by the candidate
  end
end
Aggregate all network slices to $\mathcal{N}$

in the CIP form:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{k=1}^{2} \sum_{n=1}^{|N|} \sum_{i=1}^{|V|} \sum_{j=1}^{|V|} f_k \cdot x_{ij} \\
\text{subject to} \quad & \sum_{n=1,\ldots,|N|} \alpha(i,j) \cdot x_{ij} \leq C_{i,j} \{i,j : e_{ij} \in \mathcal{L}\} \\
& \sum_{\forall n \in N} D_{i,j} \cdot x_{ij} \leq d_{i,j} \qquad \{i,j : e_{ij} \in \mathcal{L}\} \\
& x_{i,j} \in \{0,1\}, \qquad i,j = 1,\ldots,|V|
\end{aligned}
$$

The considered optimization problem is indeed an integer programming problem [18]. In each time slot, the slice controller receives a set of user demands as a request instance, $\mathcal{R}(t)$. The goal of IP algorithm is to construct a set of slices and to satisfy all or part of the requests. The computational complexity for IP algorithm to search for the optimal solution is equal to $\mathcal{O}((\mathcal{V} + \mathcal{L}) \times \mathcal{R}(t) \times \mathcal{R}(t))$ [19] .

Deep Reinforcement Learning Approach

Deep Reinforcement Learning (DRL) had been discussed in [20] to solve IP optimization problems. Hence, we enhance the DRL framework to solve the formulated ONSP problem under the IP model. The framework aims to construct an agent in the slice controller for the network slices provisioning.

The agent will not only learn to find an optimal solution at a specific NSP instance, but reuse what it has learned in previous instances. In such cases, the PPO algorithm has been shown to achieve a higher performance than the other DRL algorithms [21]. This motivates us to embed the PPO algorithm to our framework.

We assume an PPO agent interacting with a network environment to learn a provisioning policy without prior information. The state space only includes the demand requests and network graph. Given a set of demand requests $\mathcal{R}(t)$ the provisioning policy $\pi(o_t, \theta_t)$ returns a subset of $\mathcal{R}(t)$ as an action a_t to satisfy the network constraints in Eq. (C.3) and (C.4) and the traffic demand constraints in Eq. (C.5).

The action space consists of a set of actions a_t to represent which requests are accepted to transmit in a network slice in slot t. For example, if only $r_1^t = 1$ for all $r_i^t \in \mathcal{R}(t)$, it means that only request 1 is accepted to obtain network resources and to transmit traffic network slice at a time slot t. The requests for which $r_i^t = 0$ are rejected to construct their network slices and have to wait for the next action.

Every interaction between the agent and the network environment happens at the beginning of a slot. In each interaction, the agent samples the environment to get an observation o_t and performs an action a_t based on the provisioning policy. After performing the action, the agent receives a reward $R(t)$. Then the agent waits for the time slot $t+1$ to interact with the network environment. The learning process repeats the interactions continuously to approximate the optimal provisioning policy, which obtains the maximal reward.

The reward function is defined as follows. If the request r is accepted, the reward function is

$$R(t) = \sum_{\substack{\forall n \in N \\ \forall l_{i,j} \in \mathcal{L}}} x_{i,j}^{n,r} \cdot P_{i,j}, \ \forall r_i^t = 1. \tag{C.8}$$

Otherwise, the reward function is

$$R(t) = \sum_{\substack{\forall n \in N \\ \forall l_{i,j} \in \mathcal{L}}} x_{i,j}^{n,r} \cdot P_{i,j} \cdot (-1), \ \forall r_i^t \in \mathcal{R}(t). \tag{C.9}$$

In the beginning, the agent will fetch an initial environment observation o_0 from the network environment. The agent observes a set of metrics in each slot t, o_t, including the links' capacity, bandwidth and latency requirements of received user demands, and the cost achieved in the last k iterations. Then the agent feeds these values to the neural network, which outputs the next action. The next action is defined by which requests are to be chosen for the next iteration $k+1$, as well as how much bandwidth they will be allocated, at time slot $t+1$. The provisioning policy is transformed from the action

obtained from the trained neural network. If a request is selected to transmit packets, the required bandwidth will be fully reserved. Otherwise, the request waits until it gets allocated. After the new provisioning is deployed to all requests, a reward is observed and fed back to the agent. The agent uses the reward information to train and improve its model. Our implementation of the PPO algorithm in the provisioning problem is detailed in Algorithm 4. Starting from the initial parameters, the PPO algorithm optimizes its policy π until converging or reaching K iterations. At each iteration k, the PPO agent collects observation of a time slot. Next, it selects an action with the current policy. After the agent takes the selected action, the agent obtains a reward based on the reward functions in Eq. (C.8) and (C.9).

In addition, we formulate a Q-value function, a state value function and an advantage function, used to compute the intermediate values in each iteration.

$$Q_\pi(o_t, a_t) = \mathbb{E}_{o_{t+1}, a_{t+1}}[\gamma^t R(t)] \tag{C.10}$$

$$V_\pi(o_t) = \mathbb{E}_{o_{t+1}, a_t}[\gamma^t R(t)] \tag{C.11}$$

$$A_\pi(o_t, a_t) = Q_\pi(o_t, a_t) - V_\pi(o_t) \tag{C.12}$$

where γ is a discount factor, $\gamma \in [0, 1]$.

Then we construct the surrogate loss on these observations and optimize policy with stochastic gradient descent (SGD) for e epochs and minibatch size $\mathcal{B}$. The computational complexity of PPO algorithm in testing phase is also $\mathcal{O}(\mathcal{V} + \mathcal{L} + \mathcal{R})$.

Algorithm 4: Proximal Policy Optimization Algorithm

Input: An initial policy with parameters θ_0 and initial observation o_0, a set of user demand requests $\mathcal{R}(t)$
Output: A set of neural network parameters
for $k = 1, 2, 3, \cdots$ *until* $k = K$ *or convergence* **do**
  Fetch user demand requests, $\mathcal{R}(t)$, based on observation o_k.
  Take action to select user demand requests using policy $\pi = \pi(\theta_k)$.
  Compute advantage estimation based on the value function.
  Optimize surrogate function $\nabla R(t)$ with respect to θ_k using e epochs and minibatch size $\mathcal{B}$.
  $\theta_k \leftarrow \theta_{k+1}$
end

Table C.1: Simulation Parameters

Parameter Name	Value/Distribution(mean, variance)
Number of Nodes $\|V\|$	8
Number of Physical Links $\|L\|$	12
Capacity of Links	Uniform Distribution (100, 200)
Latency of Links	Uniform Distribution (1, 10)
Cost of Links	Uniform Distribution (1, 20)
Simulation Duration	1000 slots
Request Demand Requirement (b)	Normal Distribution (0, 0.1)
Request Latency Requirement (d)	Normal Distribution (1, 0.1)
Request Initial Time a_r	Normal Distribution (0, 0.1)
Request Life Time h_r	Normal Distribution (1, 0.1)
Request Arrival Model	Poisson Process

6 Numerical Investigations

Simulation Environment

Our simulation is executed on a desktop PC with an Intel i7 CPU and 8 GB memory. To construct a general simulation environment, we refer to the model in [22] to construct the network graph in our simulation. Because the MOIPO problem is NP-hard and intractable in a network with large network node sets and edge sets, we verify the proposed approximate method with the other two benchmarks on a reduced network scale. Some algorithms using high-performance computing clusters have been used to solve large-scale problems, but it is beyond the scope of this paper. Table C.1 lists the environment parameters in our implementation.

Simulation Results

We validate the performance of the PPO method in terms of the SLA violation rate and network operation cost. We define the network operation cost is the summation of all the slices' costs with respect to the their physical links and average the cost by all requests. In addition, we define the SLA violation rate as the ratio between the number of the provisioning requests and the number of all requests. We also compare the results of the PPO method with the two benchmarks, Greedy and IP. Fig. C.3 shows the comparison of slice operation cost, and Fig. C.4 presents the performance difference of SLA violation rate. The Greedy algorithm has the worst cost performance and the highest SLA violation rate because the sequential requests processing cannot closely approximate the optimal solution. The IP algorithm has better performance in terms of the cost and SLA violation rate as it considers the complete request

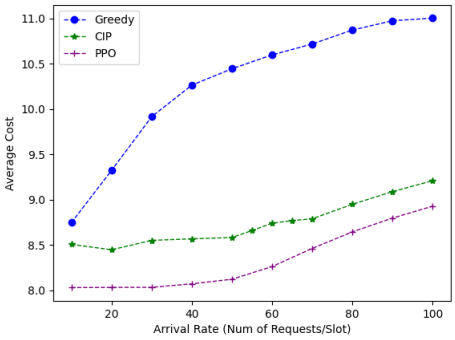

Fig. C.3: Slices Operation Cost with Different User Request Arrival Rate

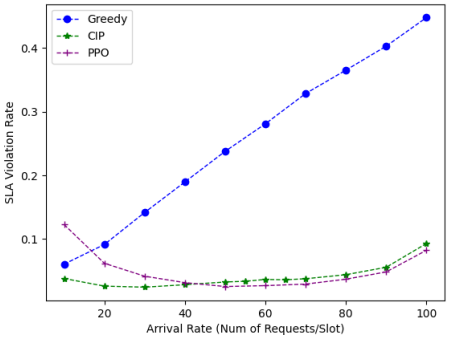

Fig. C.4: SLA Violation Rate with Different User Request Arrival Rate

set in the course of optimization. On the other hand, we verify the PPO algorithm can learn to approximate the similar network operation cost and SLA violation rate as the IP algorithm without any prior information used in the IP algorithm. From Fig. C.4, the reduction in SLA violation rate compared between CIP and PPO algorithms was ranging from 1.15 to 1.28 times.

7 Conclusion

We studied the NSP optimization problem by considering the cost of network operation and the fairness of slice provisioning in a virtualized network. To minimize the network operation cost while maximizing the provisioning fairness, we implement the PPO algorithm in our DRL framework to predict the incoming user demand and search for the optimal solution to the optimization problem. We also implemented two benchmark algorithms to demonstrate the performance difference. Simulation shows the effectiveness of our DRL framework compared with the benchmarks.

References

[1] P. Rost *et al.*, "Network slicing to enable scalability and flexibility in 5g mobile networks," *IEEE Commun. Mag.*, 2017.

[2] P. Popovski *et al.*, "5g wireless network slicing for embb, urllc, and mmtc: A communication-theoretic view," *IEEE Access*, vol. 6, pp. 55 765–55 779, 2018.

[3] R. Gouareb *et al.*, "Virtual network functions routing and placement for edge cloud latency minimization," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2346–2357, 2018.

[4] E. Amaldi *et al.*, "On the computational complexity of the virtual network embedding problem," *Discrete Mathematics*, vol. 52, pp. 213–220, 2016.

[5] NGMN, "Future networks cloud native platform," Next Generation Mobile Networks, Final Deliverable 5.2, 03 2021, version 16.12.0.

[6] 3GPP, "System architecture for the 5g system," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 23.501, 09 2021, version 16.10.1.

[7] ——, "5g system network slice selection services," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 29.531, 04 2018, version 16.6.0.

[8] ——, "5g; management and orchestration; provisioning," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 28.531, 01 2022, version 16.12.0.

[9] J. Schulman *et al.*, "Proximal policy optimization algorithms," 2017.

[10] V.-G. Nguyen *et al.*, "Sdn/nfv-based mobile packet core network architectures: A survey," *IEEE Commun. Surv. Tutor.*, vol. 19, no. 3, pp. 1567–1602, 2017.

[11] V. Sciancalepore *et al.*, "Onets: Online network slice broker from theory to practice," *IEEE Trans. Wireless Commun.*, vol. 21, no. 1, pp. 121–134, 2022.

[12] R. Ford *et al.*, "Provisioning low latency, resilient mobile edge clouds for 5g," in *Proc. IEEE INFOCOM Workshops 2017*, 2017, pp. 169–174.

[13] A. Baumgartner *et al.*, "Mobile core network virtualization: A model for combined virtual core network function placement and topology optimization," in *Proc. IEEE NetSoft 2015*, 2015, pp. 1–9.

[14] C. Sexton *et al.*, "On provisioning slices and overbooking resources in service tailored networks of the future," *IEEE ACM Trans. Netw.*, pp. 2106–2119, 2020.

[15] J. Wang *et al.*, "Spatiotemporal modeling and prediction in cellular networks: A big data enabled deep learning approach," in *Proc. IEEE INFOCOM 2017*, 2017, pp. 1–9.

[16] A. Esteves *et al.*, "On the robustness of controlled deep reinforcement learning for slice placement," *J. Netw. Syst. Mag.*, vol. 30, no. 3, jul 2022.

[17] R. Jain, D.-M. Chiu, and W. R. Hawe, *A quantitative measure of fairness and discrimination for resource allocation in shared computer system*. Eastern Research Laboratory, Digital Equipment Corporation Hudson, MA, 1984, vol. 38.

[18] T. Achterberg *et al.*, "Constraint integer programming: A new approach to integrate cp and mip," 2008, pp. 6–20.

[19] F. Eisenbrand *et al.*, "An algorithmic theory of integer programming," 2019.

[20] W. Kool *et al.*, "Attention, learn to solve routing problems!" in *International Conference on Learning Representations*, 2019.

[21] N. Heess *et al.*, "Emergence of locomotion behaviours in rich environments," *arXiv*, pp. 1–9, 2017.

[22] M. E. Newman *et al.*, "Random graphs as models of networks," *Handbook of graphs and networks*, vol. 1, pp. 35–68, 2003.

Paper D

Online Network Slicing Provisioning with Deep Reinforcement Learning

Chien-Cheng Wu

Arxiv, pp. 1–31, 2023.

The layout has been revised.

Abstract

Network Slicing has emerged in next-generation networks, enabling a logical network (slice) construction and isolation over shared physical infrastructures. While Network Slicing Provisioning (NSP) provides fundamental support for slice management, solutions for optimal NSP are still at an initial stage. To deploy optimal NSP solutions to a network environment where a time horizon is considered, augmented learning capabilities are needed to make the NSP optimization adopt the dynamic nature of the real network. This paper proposes a reinforcement learning framework to solve an online NSP optimization problem while achieving robustness against user traffic uncertainties and reserving effective provisioning. A theoretical analysis based on competitive ratios has been carried out to investigate the performance of algorithms. Furthermore, we conduct numerical simulations to evaluate the designed reinforcement learning framework.

1 Introduction

Contemporary telecommunication networks are challenged to simultaneously satisfy the different quality of service (QoS) requirements from their users. Communications service providers need to evolve their network implementations and management protocols to meet those emerging requirements and then be able to fulfill service-level agreements (SLA) for the users. The challenges come from the 'elastic' user traffic demand where the user traffic loads and arrivals are dynamic and stochastic. Network Slicing (NS) is a crucial mechanism enabled by Software Defined Networking (SDN) to allow for flexible network resource reservations and to accommodate the elastic user traffic demand [1], and [2]. Under the NS framework, service providers can manage their networks by exploiting NS for proper SLA fulfillment [3]. Within that context, Network Slicing Provisioning (NSP) is then designed to support the fundamental NS operations at a slice controller. As illustrated in Fig. D.1, the slice controller has to consider multiple functional groups, including the radio access networks (RANs), transport networks (TNs), and core networks (CNs) to establish end-to-end network slices. The NSP optimization aims to offer robust NS service which can keep low network operation costs and SLA violations when the traffic load or arrivals change frequently. Whilst the proposed approaches towards the optimal NSP problem under stochastic traffic demand are at an embryonic stage, previous literature proved that the NSP optimization is an NP-hard problem [4].

One of the major NSP research challenges is how to construct the slices for different user demands while keeping optimal network operation cost and demand acceptance rate. For example, over-provisioning allocates all resources to concurrent slices and incurs a high costs. On the other hand,

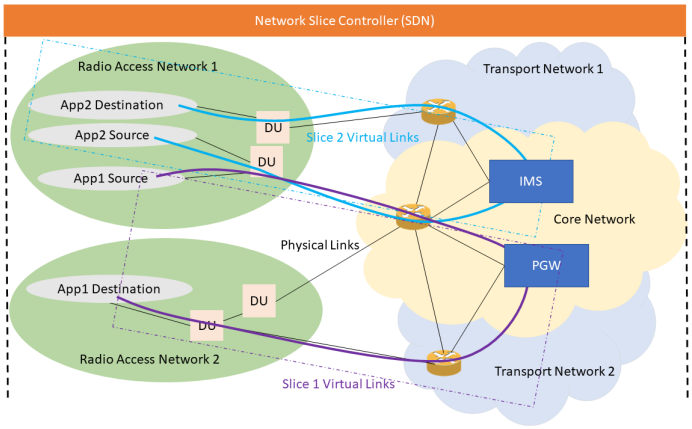

Fig. D.1: An illustration of end-to-end network slices

under-provisioning leaves more resources to future slices and thus might incurs a low demand acceptance rate and a QoS degradation. NSP optimization in this regard was studied using a static optimization approach, which is inadequate when there is uncertainty in user traffic. Such rule-based approaches cannot adequately adapt to the temporal dynamic traffic changes.

Williamson et al. [5] studied that daily user traffic repeats with some patterns that depend on the weekday. These results have been verified by [6–9] and [10]. Their results inspire us to exploit the user traffic prediction using reinforcement learning approaches for NSP optimization as outlined in the following. The slice controller can learn to construct the slices based on different user demands, where the historical user traffic guides the controller toward an optimal slicing provisioning policy. We leverage this observation in that work, showing that reinforcement learning could be one of the solutions to approximate the optimal NSP.

Specifically, the contributions of this paper are summarized as follows:

1. We formulate the NSP optimization problem as an online multi-objective integer programming problem.

2. We implement a reinforcement learning framework to approximate the optimal solutions under stochastic user traffic demands.

3. We provide a theoretical analysis based on competitive ratios and show the error-dependent performance guarantee for all methods under investigation.

4. From a discrete event simulation, we verify that the proposed framework provides a faster approximation to the solutions from the integer programming method.

The rest of this paper is organized as follows: Section 2 discusses related works. The telecommunication network model, user request model, and network slice model are presented in Section 3. The online network slice provisioning problem is formulated in Section 4. Next, the existing approaches and the proposed reinforcement learning framework are introduced to solve the online network slice provisioning problem in Section 5. Section 6 shows theoretical analysis based on competitive ratios. Section 7 presents numerical results. Finally, Section 8 concludes this research.

2 Related Work

2.1 Slice Provisioning Problem

In [11], the authors have demonstrated their network slicing implementation. Their suggested model offers an effective slice provisioning solution by analyzing historical network slice requests. The paper in [12] considers an NSP problem with partially unknown resources and user demands. Their model defined the available network resources and slice resource utilization as a normal distribution. A customizable factor is then provided to manage the likelihood of the NSP solution. Instead of providing a sub-optimal method for the NSP problem, this research investigates algorithms to approximate the optimal solution to adaptively provision network resources for elastic user traffic.

2.2 Slice Provisioning Optimization

Due to the variations in slice requests, the NSP optimization problem usually considers the Virtual Network Functions (VNFs) migration and virtual links' construction. Many algorithms were used to approximate the optimal solutions of NSP [4]. Some approximated or heuristic algorithms, such as differential evolution (DE), particle swarm optimization (PSO), or genetic algorithm (GA) have been used to solve the NSP optimization problem. However, the existing research has a few discussions on the topic of online NSP optimization.

The study in [13] suggested where the VNFs placement should be optimized for the edge part in a telecommunication network. Their optimization approach aims to distribute VNFs optimally among several data centers. In addition, the authors in [14] also provided the optimal VNF placement for the core network. The VNFs were distributed based on the optimal placement cost. During their VNFs allocation, they generally considered physical network constraints such as processing time, storage capacity, switching capacity, and services in the physical substrate network. The authors in [15]

suggested using a GA approach to minimize the compute cost required to solve the resilient NSP problem, which was demonstrated to outperform the existing Mixed-Integer Linear Programming solver.

This study extends from the first work delineated in [16], while supplementing it with rigorous theoretical analysis and an additional set of performance evaluations. The method presented in [16] conceptualizes the migration of Virtual Network Functions (VNFs) as a merged process of constructing virtual links. The proposed DRL framework is designed to efficiently approximate an optimal Network Service Provider (NSP) solution for dynamic user traffic in an online fashion. Hence, we continue the prior research and provide detailed analysis and findings in this article.

3 System Model

The notations used in this research article have been itemized in Table D.1. The system was defined as a time-slotted system, and each action, such as the arrival of a user demand request, occurs at the commencement of a slot and completes before the termination of that slot.

3.1 Network Model

We assume the network is an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$, where $\mathcal{V}$ represents the set of network nodes and $\mathcal{L}$ signifies the set of undirected wired and wireless links that comprise the deployed physical infrastructure in the network. A link between nodes v_i and v_j ($v_i, v_j \in \mathcal{V}$) is denoted by $l_{i,j} \in \mathcal{L}$. The nodes in the graph embody either a radio access network component, such as a base station (BS), a transportation regional network component, or a core network element (e.g. IMS or PGW). Specifically, we assume that all network nodes are virtualized, indicating that they possess the capacity to operate virtual machines or containers atop the corresponding physical (bare metal) machine capabilities. Moreover, we denote the available link capacity as a vector $\mathbf{C} = [C_{1,1}, \ldots, C_{i,j}, \ldots, C_{V,V}]$, where each element $C_{i,j} > 0$ represents the capacity of the direct link $l_{i,j}$ between nodes v_i and v_j. If no link exists between v_i and v_j, then the value of $C_{i,j}$ is -1.

We also denote the occupied link capacity as $\alpha(i, j)$, where

$$0 \leq \alpha(i, j) \leq C_{i,j}. \tag{D.1}$$

The delay of the links is provided in the vector $\mathbf{D} = [D_{1,1}, \ldots, D_{i,j}, \ldots, D_{V,V}]$, where each element $D_{i,j} > 0$ expresses the delay between v_i and v_j. The active network operation cost of links in $\mathcal{L}$ is given by the vector: $\mathbf{P} = [P_{1,1}, \ldots, P_{i,j}, \ldots, P_{V,V}]$, where each element $P_{i,j} > 0$ constitutes the active operation cost between nodes v_i and v_j. If there is no link between v_i and v_j, we

assign the maximum available integer value to $D_{i,j}$ and $P_{i,j}$ in our simulations to represent the disconnected scenario.

3.2 User Demand Model

The set of user demand requests received by the slice controller at the time slot t is denoted by $\mathcal{R}(t)$, comprising both newly arrived request set at t, $\mathcal{A}(t)$, and the active backlogged request set, $\mathcal{Q}(t-1)$.[1] According to the standard document [17], each request r, $r \in \mathcal{R}(t)$, represents a network slice demand of a network application from a user at the time slot t. A user demand request is represented as a septuple $r = \{v_s, v_u, b_r, d_r, \iota_r, a_r, h_r\}$, $r \in \mathcal{R}(t)$, where v_s and v_u denote the source and destination nodes respectively. b_r is the required data rate, d_r denotes the delay requirement, and the demand type is expressed as κ_r. The request's initial time slot is a_r and the request life time is h_r slots. All the parameters of user demand requests are unknown in advance.

A request's life cycle follows the following procedure. When a request arrives at a time slot t', the slice controller must make an irrevocable decision of whether to (1) accept, (2) buffer, or (3) reject the request at t'. If the request can be accepted at t', then the slice controller must construct a slice to serve it. Then the request will be served starting from the time slot t' and continue for h_r slots (i.e. if the request can be served at t', the request will be terminated at the time slot $t = t' + h_r$). Otherwise, if the request cannot be accepted at t' and $t' < a_r$, the request will be buffered in a queue to wait for a provisioning opportunity at the next time slot $t' + 1$. In the last case, if the request cannot be served before $t' \geq a_r$ (i.e. the request expired), the request will be rejected.

All requests arrive according to a Poisson arrival model determined by an arrival rate $\lambda_t = \lambda$ (requests/slot). In addition, the requests' parameters are determined when the requests are generated. We assume that user demand requests' generation follows regular temporal patterns [6] and the parameter generation at every user is based on a priori unknown distribution drawn from the aforementioned regular temporal patterns. But we assume that the value of each parameter lies in a range known as a priori. To this end, we have $b_{min} \leq b_r \leq b_{max}$, $d_{min} \leq d_r \leq d_{max}$, $a_{min} \leq a_r \leq a_{max}$, and $h_{min} \leq h_r \leq h_{max}$, for some known constants: b_{min}, b_{max}, d_{min}, d_{max}, a_{min}, a_{max}, h_{min} and h_{max}.

A network slice controller can accept, buffer or reject the user demand requests by its decisions in an online manner. For example, at a time slot $t = t_1$, we assume that a request set $\mathcal{R}(t_1)$ is handled by the slice controller. And if the appropriate network slices can be constructed for all the requests in $\mathcal{R}(t_1)$, the network slice controller accepts the set $\mathcal{R}(t_1)$. On the other

[1] The backlogged requests are buffered in a queue at the NS controller if they cannot be served in the previous time slots. The queue size at the NS controller is large enough to avoid overflow. A backlogged request is active if its initial time has not yet expired.

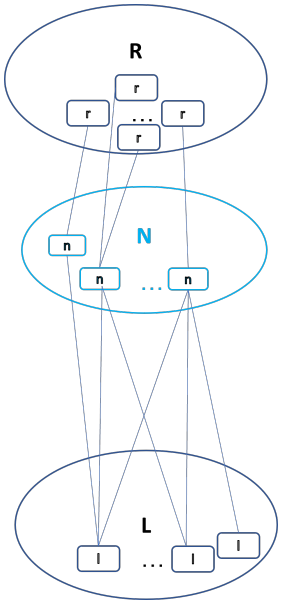

Fig. D.2: Network Slices Mapping Demonstration

hand, if a request $r \in \mathcal{R}(t_1)$ cannot be served at t_1, it can be buffered in a queue and waits for a provisioning opportunity at the next time slot until the time slot ($t = a_r$). Furthermore, the request will be rejected if the request cannot be served after the time slot ($t = a_r$).

3.3 Network Slice Model

Each slice n can be denoted by

$$n = \{v_{n,s}, v_{n,u}, E_n, y_n, \kappa_n, \iota_n\} \tag{D.2}$$

where $v_{n,s}$ is the source node and $v_{n,u}$ is the destination node, $v_{n,s}, v_{n,u} \in \mathcal{V}$. $E_n = \{e^n_{i,j}\}$ is a set of virtual links, with corresponding network resource reservations. Each virtual link $e^n_{i,j} \in E_n$ can be mapped to one and only one physical link $l_{i,j}$ between node v_i and node v_j, and E_n is a loop-free path, starting from $v_{n,s}$ and ending at $v_{n,u}$. If the loop-free path exists between the source node and destination node, then the connectivity constraints of the slice n can be satisfied. If there is no loop-free path between the source node and destination node, then the slice n cannot be created. [2]

As presented in Fig. D.2, each slice n consists of at least one physical link l and can carry exactly one request r. The capacity and delay constraints of a

[2]The loop-free path search doesn't include the capacity and delay constraints of the path.

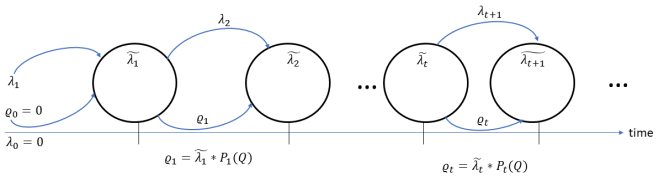

Fig. D.3: Time Series of Network Slices Provisioning

virtual link $e^n_{i,j}$ are the same as the ones of the mapped physical link $l_{i,j}$. The slice load is y_n, which means that each $e^n_{i,j} \in E_n$ occupies y_n capacity of link $l_{i,j}$. Therefore

$$y_n \leq \min_{\forall e^n_{i,j} \in E_n} \alpha(i,j) \tag{D.3}$$

The slice delay κ_n is $\sum_{i,j \text{ s.t. } e^n_{i,j} \in E_n} D_{i,j}$. The slice type is denoted as ι_n, allocated as the same as the demand type. A feasible slice can only be provided by its slice controller by fulfilling the connectivity, capacity and delay constraints of its user demand request.

In addition, we assume the coexistence of multiple slices on the same physical network, and each slice serves network traffic from a single user demand request. Therefore, we represent a slice by a path containing the source-destination of the request.

4 Online Network Slice Provisioning Problem Formulation

This section introduces the formulation for generating network slices that accommodates the underlying user requests regarding data rate and latency while minimizing the network operation cost and SLA violation rate.

4.1 Time Series Model

The network operates in a time-slotted environment, and the entire time interval is divided into $\mathcal{T}$ short time slots. For each time slot $t = 1, 2, \ldots \mathcal{T}$, the new user demand requests dynamically arrive according to an unknown independently and identically distributed (i.i.d) Poisson distribution with arrival rate $\lambda_t = \lambda$.

For each user demand request r at a specific time slot t, the NS controller aims to generate a network slice n to serve r. If a request is unserved at t, it will be buffered at the controller's queue until its initial time expires. The new arrival requests at time slot $t+1$ and buffered requests at time slot t compose an artificial arrival rate, $\widetilde{\lambda}_{t+1}$. The buffered requests at the

controller's queue are denoted by the backlogged request set, $\mathcal{Q}(t)$, which can be carried to time slot $t+1$. In addition, we define the backlog rate as ϱ_t, which means the rate of unserved requests buffering at the NS controller's queue. Let $P_t(\mathcal{Q})$ be the probability of user requests being buffered at the NS controller under a network environment in time slot t.

Hence, the backlog rate ϱ_t is given by

$$\varrho_t = \widetilde{\lambda}_t P_t(\mathcal{Q}) \tag{D.4}$$

We assume that $\lambda_0 = \varrho_0 = 0$ in slot $t = 0$ and $\lambda_t = \lambda$ and then we can get the total rate recursively through

$$\widetilde{\lambda}_{t+1} = \lambda_{t+1} + \varrho_t = \lambda + \widetilde{\lambda}_t P_t(\mathcal{Q}) \tag{D.5}$$

Fig. D.3 illustrates four instances of user demand requests with the time slots are displayed on the x-axis. We assume the arrival rate, λ_t, and the backlog rate, ϱ_t, are zero in the beginning (i.e. $\lambda_0 = \varrho_0 = 0$). We expect λ_1 requests arrived at slot $t = 1$ and therefore the artificial arrival rate $\widetilde{\lambda_1} = \lambda_1$. If $\varrho_1 = \widetilde{\lambda_1} P_1(\mathcal{Q})$ requests are buffered, we will have ϱ_1 requests to be conveyed to the next time slot $t = 2$. Such iterations will continuously repeat until time slot $\mathcal{T}$.

By using the adjusted Pollaczek-Khinchin's equation [18], the expected queue length, $\mathbb{E}[\mathcal{R}(t)]$, at the NS controller can be estimated. We assume the effective service rate of the NS controller is μ, and $\mu > \widetilde{\lambda}_t(1 + P_t(\mathcal{Q}))$. Moreover, the utilization of the NS controller is denoted by $\rho = 1/(1 + P_t(\mathcal{Q}))$. Because of $\rho < 1$, the expected queue length is finite. The accurate value of queue length will be statistically calculate by simulations.

In order to construct numerical simulation, we define the number of requests at time slot t is $|\mathcal{R}(t)|$, and the number of slices at time slot t is $|\mathcal{N}_t|$. We derive the constraint functions as following: since multiple slices share the same physical network, the data rate allocation at each virtual link shall not exceed the available physical link data rate. We can obtain the global network data rate constraint in Equation (D.6)

$$\sum_{n=1}^{|\mathcal{N}_t|} y_n x_{i,j}^n \leq C_{i,j},\ \forall l_{i,j} \in \mathcal{L}. \tag{D.6}$$

Furthermore, for the slice n of request r, at each time slot t, every virtual route mapped to the physical links should satisfy the volume of traffic requested by the user. Hence, the data rate constraint for each link is shown in Equation (D.7)

$$\min_{\forall e_{i,j}^n \in E_n} \alpha(i,j) \geq y_n,\ \forall n \in \mathcal{N}_t. \tag{D.7}$$

On the other hand, in the slice n of request r, each virtual route assigned to the physical links should also satisfy the requested user traffic latency.

$$\sum_{\forall e^n_{i,j} \in E_n} D_{i,j} x^n_{i,j} \leq \kappa_n, \ \forall n \in \mathcal{N}_t. \tag{D.8}$$

The network operation cost objective function is defined as

$$f_1 = \min \sum_{\forall l_{i,j} \in \mathcal{L}} P_{i,j} x^n_{i,j}, \ \forall n \in \mathcal{N}_t \tag{D.9}$$

Next, we can define the SLA violation rate as

$$\frac{|\mathcal{R}(t)| - |\mathcal{N}_t|}{|\mathcal{R}(t)|} \tag{D.10}$$

And the SLA objective function is defined as

$$f_2 = \min \frac{|\mathcal{R}(t)| - |\mathcal{N}_t|}{|\mathcal{R}(t)|} (\max_{\forall i,j \in \boldsymbol{P}} P_{i,j}) |\mathcal{V}| |\mathcal{R}(t)| \tag{D.11}$$

The SLA objective function demonstrates the penalty of the rejected requests at the NS Controller. We use the maximum link operation costs to normalize the value difference between SLA violation rates and network operation costs. Therefore, we can incorporate SLA violation rate into an unified evaluation of the ONSP optimization problem.

The NSP optimization problem is then formulated as a multi-objective optimization problem to construct a route for a slice. Moreover, the details of online optimization will be elaborated in the next section.

4.2 Online NSP Optimization Problem

The user demand requests are indeterminate before they arrive at the NS controller; thus, Network Slice Provisioning (NSP) requires online optimization to process the uncertainty of requests. Specifically, we consider the uncertainty in the number of requests per time slot $|\mathcal{R}(t)|$, as well as the requests' source node, destination node, required data rate, delay, and the requests' initial time and life time. The NS controller generates network slices to serve the user demand requests and the slice generation procedure can be decomposed into two steps. The first step aims to find potential end-to-end routes which can be utilized for request r from its source node v_s to its destination node v_u. Then a slice's source and destination nodes can be determined as $v_{n,s}$ and $v_{n,u}$ accordingly. The second step is to find the optimal route that can minimize the objective function while satisfying the constraints. To this end, we define the matrix X, and for each element of the matrix, $x^n_{i,j}$, if slice n utilizes the link $l_{i,j}$, then $x^n_{i,j} = 1$, otherwise $x^n_{i,j} = 0$.

From the above steps, we can further construct the NSP optimization problem as follows: given a set of user demand requests, $\mathcal{R}(t) = \mathcal{A}(t) + \mathcal{Q}(t-1)$, at time slot t, how can a set of slices $\mathcal{N}_t$ be generated with the minimum cost and minimum SLA violation to transmit user traffic satisfying network constraints?

If we jointly consider a series of time slots, $\mathcal{T}$, in the practical network environment, the NSP optimization problem can be cast as an online NSP (ONSP) optimization problem. Because of the total network capacity limitation or the bottleneck over a period of time, there may be only a subset of user demand requests that can be served by the slice controller (i.e. $|\mathcal{N}_t| \leq |\mathcal{R}(t)|$). In this ONSP problem, we use $\mathbf{z}$ to represent a decision vector at each time slot t. For each vector element, z_r, is a binary variable representing the decision of request r such that

$$z_r = \begin{cases} 1 & \text{if request } r \text{ is selected to be served} \\ 0 & \text{otherwise} \end{cases} \tag{D.12}$$

For every time slot in a period, $\mathcal{T}$, each arrival request has input parameters about its demand information; the slice controller then processes each of its demand and reserves network resources for the corresponding network slice. The decisions to be made at every time slot are which requests are being processed and which slices are being constructed. A solution must satisfy the slice demands and the network constraints, while minimizing the network operation cost and the SLA violation rate. We can further formulate the ONSP problem via a set of objectives and constraints as follows in (D.13). The summation (D.13a) represents a joint objective function of this problem where each slice n is associated with its rate and latency requirements. In addition, the inequalities in (D.13b) and (D.13c) express that no slice data rate allocation can exceed the available physical link data rate, and no slice delay is larger than the user-tolerable delay. The constraints from (D.13d) to (D.13f) limit the size of solutions.

$$\text{minimize} \sum_{t=0}^{\mathcal{T}} \left(\sum_{n=1}^{|\mathcal{N}_t|} f_1 + f_2 \right) \tag{D.13a}$$

$$\text{subject to} \sum_{i,j}^{l_{i,j} \in \mathcal{L}} \sum_{n=1}^{|\mathcal{N}_t|} y_n x_{i,j}^n \leq C_{i,j} \tag{D.13b}$$

$$\sum_{\forall e_{i,j}^n \in E_n} D_{i,j} x_{i,j}^n \leq \kappa_n \quad (\forall n \in \mathcal{N}_t) \tag{D.13c}$$

$$\sum_{r=1}^{|\mathcal{R}(t)|} z_r = |\mathcal{N}_t| \quad (\forall r \in \mathcal{R}(t)) \tag{D.13d}$$

$$x_{i,j}^n \in \{0,1\} \quad (\forall n \in \mathcal{N}_t;\ i,j : l_{i,j} \in \mathcal{L}) \tag{D.13e}$$

$$z_r \in \{0,1\} \quad (\forall r \in \mathcal{R}(t)) \tag{D.13f}$$

The ONSP problem in (D.13) has only a theoretical interest since it assumes that we have arrivals information in each timeslot. Hence, it can be solved by an oracle (i.e., the demand arrivals are fixed in every slot, or all the demands can be obtained offline when the provisioning is decided), and the solution will be optimal. Hereafter, we propose a reinforcement learning framework in Section 5 to approximate the optimal solution derived by the oracle.

5 Methods in the Proposed Framework

5.1 Reinforcement Learning Framework

We propose an Reinforcement Learning (RL) framework (shown in Fig. D.4) to approximate the optimal solution of ONSP problem, i.e., the solution after checking all requests in all timeslots. The RL framework is used to predict the information of future request arrivals which an optimization solver required in each timeslot. The framework operates as follows. Users send varied network slice requests to the slice controller. The slice controller consists of an agent, an optimization solver (e.g. integer programming solver or greedy solver). Next, the agent determines the batch size of integer programming solver by deploying an RL agent. The agent calculates the batch size according to the batch size prediction model. Then, the integer programming solver uses the agent's outputs to provision network slice requests. The model should achieve the network operation cost objective and SLA violation objective. The agent learns to decide the batch size that minimize the network operation cost and the penalty of SLA violation by considering the

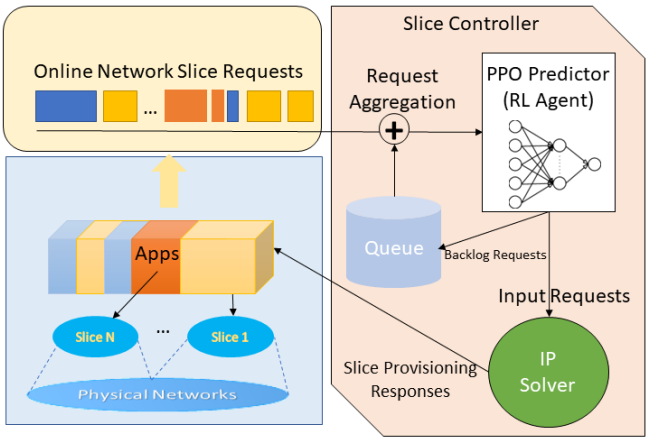

Fig. D.4: The Proposed Reinforcement Learning Framework

information on states and rewards from interaction with the network by the reinforcement algorithm in Section 5.4.

5.2 Greedy Approach

The most rudimentary algorithm to address the ONSP problem is the greedy algorithm, as demonstrated in Algorithm 5. A bidirected graph, $\mathcal{G}$, and a user demand request set, $\mathcal{R}(t)$, are provided as inputs. We further define Ω as the set of all feasible paths originating from $\mathcal{R}(t)$. The cardinality of a set, such as $|\mathcal{R}(t)|$ and $|\Omega|$, indicates the number of elements in the set $\mathcal{R}(t)$ and the set Ω, respectively.

For each user demand request $r \in \mathcal{R}(t)$, we employ the Breadth-First Search (BFS) algorithm to search in $\mathcal{G}$ for all feasible paths that satisfy the connectivity constraint of r. If a path fulfills the connectivity constraint of r, we designate the path as a feasible path; otherwise, the path is deemed infeasible. Furthermore, from the graph $\mathcal{G}$ and the set $\mathcal{R}(t)$, we can derive the data rate constraint functions and latency constraint functions, as shown in Equation (D.6), (D.7), and (D.8). Next, we consider all feasible paths and evaluate the link availability of each path based on the given data rate and latency constraints. If a link is unavailable, the corresponding path is pruned; otherwise, we proceed to the subsequent step. The greedy algorithm then iterates all requests in $|\mathcal{R}(t)|$ within a loop containing a finite number of steps.

Subsequently, we calculate the cost value and SLA violation rate based on the objective functions. We then compare the cost value and SLA violation value in the current step with the cost value in the previous step. If the

cost value in the current step is lower than the cost value in the previous step and the SLA violation value in the current step is higher than the SLA violation value in the previous step, we assign the path ω to be a candidate for slice n with respect to request r. Next, we generate a slice n with a non-empty candidate; otherwise, we advance to the next feasible path. Finally, we aggregate each slice n to form a set $\mathcal{N}_t$ for output.

Algorithm 5: Greedy Algorithm

Input: Given a set of user demand requests, $\mathcal{R}(t)$ and a bidirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$
Output: a set of network slices, $\mathcal{N}_t$
Sorting the set $\mathcal{R}(t)$
for $r = 1, 2, 3, \cdots$ *until* $r = |\mathcal{R}(t)|$ **do**
  Find all feasible paths, Ω, based on a searching algorithm
  for $\omega = 1, 2, 3, \cdots$ *until* $\omega = |\Omega|$ **do**
    Check the link availability in ω based on the constraint functions.
    Calculate the cost value and SLA violation value based on the objective functions.
    Compare with the previous cost value and SLA violation value.
    if *current value is better than previous value* **then**
      Assign the current ω to candidate
    end
  end
  if *candidate is not empty* **then**
    Generate a network slice n by the candidate
  end
end
Aggregate all network slices to $\mathcal{N}_t$

5.3 Integer Programming Approach

Integer Programming (IP) is a mathematical formulation model to solve optimization problems. Many algorithms have been proposed to efficiently solve the IP problem such as the Constraint Integer Programming (CIP) algorithm [19]. In particular, the CIP method was proven to be able to obtain the optimal solution for an IP problem [19]. The general form of CIP is the following: given a finite set $\mathcal{H}$ of constraints $h_i : \mathbb{R}^n \rightarrow \mathbb{Z}$, $i = \{1, \ldots, m\}$, a variable set $\mathcal{X}$, $x \in \mathbb{Z}, \forall x \in \mathcal{X}$, and a vector of objective functions $\boldsymbol{f} \in \mathbb{R}^n$, derive an optimal solution θ from $\mathcal{X}$ if the CIP is satisfiable as $\theta = \min\{\boldsymbol{f}^T \mathcal{X} | h_i(\mathcal{X}) = \text{true}, \forall h_i \in \mathcal{H}\}$.

In addition, at each time slot t, the ONSP problem in Section 4, can be

presented in the IP form with two constraint functions as following:

$$\begin{aligned} \text{minimize} \quad & \sum_{n=1}^{|\mathcal{N}_t|} f_1 + f_2 \\ \text{subject to} \quad & \sum_{\forall n \in \mathcal{N}_t} y_n x_{i,j}^n \leq C_{i,j} \;\; \{\forall i,j : l_{i,j} \in \mathcal{L}\} \\ & \sum_{\forall e_{i,j}^n \in E_n} D_{i,j} x_{i,j}^n \leq \kappa_n \{\forall n \in \mathcal{N}_t\} \\ & x_{i,j}^n \in \{0,1\} \qquad \{\forall n \in \mathcal{N}_t;\; i,j : l_{i,j} \in \mathcal{L}\} \end{aligned} \tag{D.14}$$

As a result, we implement the IP algorithm as a benchmark method for solving the ONSP problem. The complete IP algorithm is presented as Algorithm 6.

Algorithm 6: Integer Programming Algorithm

Input: An initial state with batch size β, a set of user demand requests $\mathcal{R}(t)$
Output: A set of slices
for $k = \beta, \beta-1, \beta-2, \cdots$ *until* $k = 0$ *or found optimal solution* **do**
    Fetch user demand requests, $\mathcal{R}(t)$, based on k
    Solve problem (D.14)
end

In each time slot t, the slice controller receives a set of user demands as a request instance, $\mathcal{R}(t)$ and executes the IP algorithm to construct a set of slices to serve the request instance. However, the IP algorithm can only construct a set of slices within its predefined batch size and satisfy all or a part of the request instance. The time complexity of search for the optimal solution in the predefined batch is exponential [20] [21], causing a significant processing delay in the controller.

5.4 Deep Reinforcement Learning Approach

Deep Reinforcement Learning (DRL) was discussed in [22] to solve IP optimization problems. An agent from the DRL-based approach could directly learn policies based on the rewards by interacting with the network environment. Hence, we enhance the general DRL-based approach to solve the formulated ONSP optimization problem under the IP model. As shown in Fig. D.4, our DRL approach aims to construct an agent in the slice controller for the decisions of network slice provisioning. The agent could not only learn to find a solution at a specific NSP instance but also reuse what it has learned in previous instances to approximate a nearly optimal solution.

In particular, we implement the Proximal Policy Optimization (PPO) algorithm [23] in our DRL approach because PPO has been shown to achieve higher performance than the other Actor-Critic based algorithms [24].

PPO algorithm leverages the advantages of the policy-based and value-based DRL algorithms, and therefore, it is suitable in the network slicing environment with a discrete state space and a continuous action space. On the contrary to the classical DRL algorithms, the PPO algorithm can share parameters for both policy and value functions to reduce the computational complexity in the slice controller. The complexity enhancement is crucial because the slice controller could respond to slice requests quickly using less computing resources. Our goal is to exploit the advantages of the PPO algorithm to approximate the optimal solution to the ONSP optimization problem. To the best of our knowledge, this approach is the first work to embed the PPO algorithm in the slice controller for solving the ONSP optimization problem.

In the following paragraphs, we define the state space, action space, reward function and PPO implementation in our DRL approach.

State Space

The PPO agent learns to provision the slice requests from the network environment without any prior traffic information. The agent only needs the current network state information as follows:

1. the set of physical network nodes ($\mathcal{V}$): the nodes represent either a radio access network component, a transportation regional network component, or a core network component.

2. the set of wired-cum-wireless links ($\mathcal{L}$): the links include the information about 1. the available link capacity, 2. link delay, and 3. link operation cost.

3. a set of user demand requests, $\mathcal{R}(t)$: the user demand requests include the information about 1. the source and destination node 2. the requested data rate, 3. the delay requirement, 4. demand type, 5. demand's initial time slot, and 6. demand's life time

Action Space

The agent's learning process includes a series of interactions with the network environment. In each interaction, the agent obtains an observation o_t from the environment's state sampling and takes action a_t based on the provisioning policy, $\pi()$ and policy parameters, θ_t. The provisioning policy $\pi(o_t, \theta_t)$ returns an action a_t, including a subset of $\mathcal{R}(t)$, to satisfy the network constraints in Equation (D.6) and (D.7) and the user traffic demand constraints

in Equation (D.8). The action a_t represents which requests are accepted to be served in a network slice in slot t. For example, if at time slot t, only $z_1 = 1$ for all $z_r \in \mathbf{z}$, it means that only request $r = 1$ is accepted to obtain network resources and to transmit network traffic at a time slot t. The other requests in $\mathcal{R}(t)$ for which $z_i = 0$ are rejected to construct their network slices and have to wait for the next action or be removed from the slice controller.

Upon completion of the action, the agent gets a reward $reward_t$ and finishes an interaction. Then the agent waits for the time slot $t + 1$ to interact with the network environment again. After that, the agent starts another interaction with the network environment until reaching the next time slot $t + 1$. The agent repeats the learning process to obtain the optimal provisioning policy.

Reward Function

Equation (D.15) and Equation (D.16) are the defined reward functions. At every time slot t, if the request r is accepted, the reward function is

$$reward_t = \sum_{\substack{\forall n \in \mathcal{N}_t \\ \forall l_{i,j} \in \mathcal{L}}} P_{i,j} \cdot x_{i,j}^n, \ \forall z_r = 1, r \in \mathcal{R}(t) \tag{D.15}$$

Otherwise, the reward function is

$$reward_t = \sum_{\substack{\forall n \in \mathcal{N}_t \\ \forall l_{i,j} \in \mathcal{L}}} (-1) \cdot P_{i,j} \cdot x_{i,j}^n, \ \forall z_r = 0, r \in \mathcal{R}(t) \tag{D.16}$$

PPO Implementation

In Algorithm 7, we describe how we implemented the PPO algorithm with respect to the ONSP problem. Starting with the default parameters, the agent uses an initial environment observation o_0 from default system inputs as the first observation. Afterwards, the agent fetches the environment observation o_t in each slot t, including the links' capacity, data rate and latency requirements of received user demands, and the cost obtained in the previous time slots. Next, the neural network receives these values from the agent and generates the subsequent action. The policy-compliant action is formulated by selecting requests for the following time slot $t + 1$ and how much data rate will be allotted to them at time slot $t + 1$. The provisioning policy is then derived from the action. The slice controller will reserve the necessary data rate if a request is selected to be severed. Otherwise, the request is buffered until it is allocated. Following the deployment of the updated provisioning to all requests, a reward is captured and returned to the agent. The reward is obtained from the reward functions in Equation (D.15) and (D.16). The agent

uses the reward to train and enhance its neural network for the provisioning policy, π. The agent optimizes its policy, π, iteratively until it reaches convergent conditions or K iterations.

For each iteration, we compute intermediate values using the following state-action value function, $Q_\pi(o_t, a_t)$, state value function, $V_\pi(o_t)$, and advantage function, $A_\pi(o_t, a_t)$.

$$Q_\pi(o_t, a_t) = \mathbb{E}_{o_{t+1}, a_{t+1}}[\sum_{l=0}^{\infty} \gamma^l (reward_{t+l})] \tag{D.17}$$

$$V_\pi(o_t) = \mathbb{E}_{o_{t+1}, a_t}[\sum_{l=0}^{\infty} \gamma^l (reward_{t+l})] \tag{D.18}$$

$$A_\pi(o_t, a_t) = Q_\pi(o_t, a_t) - V_\pi(o_t) \tag{D.19}$$

where γ is a discount factor and $\gamma \in [0, 1]$.

After that, we calculate the surrogate loss based on these observations and use stochastic gradient descent (SGD) to optimize the policy for e epochs and minibatch sizes, $\mathcal{B}$.

Algorithm 7: Proximal Policy Optimization

Input: A default policy with initial parameters θ_0 and initial network observation o_0, a set of user demand requests $\mathcal{R}(t)$
Output: A set of neural network parameters
Set counter $t \leftarrow 0$
for $k = 1, 2, 3, \cdots$ *until* $k = K$ *or convergence* **do**
  Fetch user demand requests, $\mathcal{R}(t)$ and observation o_t.
  Take action, a_t, to select user demand requests using policy $\pi = \pi(\theta_t)$.
  Compute advantage estimation, $A_\pi(o_t, a_t)$, based on the value function, $V_\pi(o_t)$.
  Update the policy using optimize surrogate loss with $\nabla reward_t$ and θ_t in e epochs and minibatch size $\mathcal{B}$.
  $t \leftarrow t + 1$
  $\theta_t \leftarrow \theta_{t+1}$
end

6 Performance Analysis

In this section, we state the sufficient conditions for the optimality of the ONSP optimization problem. We also analyze our algorithms' computational complexity and competitive ratio concerning at the ONSP optimization problem. Our analysis starts by estimating the computational complexity of the ONSP optimization problem. First of all, the classical NSP optimization

problem has been proven to be NP-hard [4]. Because the classical NSP optimization problem is a special case of the ONSP optimization problem with deterministic user demand arrivals, finding optimal solutions to the ONSP optimization problem is still NP-hard. Then we compared the greedy and integer programming methods proposed in [25] and modified their objective functions for our problem formulation. Specifically, our problem formulation can be solved by a greedy algorithm (Section 5, Algorithm 5) and a constraint integer programming algorithm (Section 5, Algorithm 6) which tend to minimize network operation costs and SLA violation rates. The constraint integer programming algorithm can obtain a nearly optimal solution to the ONSP problem, but the constraint integer programming algorithm has exponential-worse time complexity [20] [21]. On the other hand, the greedy algorithm can achieve polynomial time complexity but produce a non-optimal solution. Both methods do not suffice to quickly approximate a nearly optimal solution. Therefore, we design the reinforcement learning framework to reduce the solution searching time of integer programming algorithm using Algorithm 7 presented in Section 5. The PPO algorithm has a constant time complexity during the prediction phase. In addition, the proofs for the optimality criterion and competitive ratio are established in the following theorem.

Theorem 6.1
Assume that we are given a finite set of user demand, $\mathcal{R}$. The parameters of demands sample from a vector of random variables with corresponding probability density functions. Let $|\mathcal{N}| = |\mathcal{R}|$. For a given network, $\mathcal{G} = (\mathcal{V}, \mathcal{L})$ with $P_{i,j}$, $C_{i,j}$, *and* $D_{i,j} > 0$, a slice provisioning strategy in problem (D.13) is optimal if and only if there exists an integer $\zeta, (1 \leq \zeta \leq |\mathcal{N}|)$ to construct a subset $\mathcal{M}$ where $|\mathcal{M}| = \zeta$, such that all of the following constraints are satisfied:

$$\sum_{n=1,\ldots,\zeta} y_n x^n_{i,j} \leq C_{i,j} \;\; \{\forall i,j : l_{i,j} \in \mathcal{L}\}$$
$$\sum_{i,j:e^n_{i,j} \in E_n} D_{i,j} x^n_{i,j} \leq \kappa_n \{n = 1 \ldots \zeta\}$$
$$x^n_{i,j} \in \{0,1\}, \qquad \{\forall n \in \mathcal{M};\; i,j : l_{i,j} \in \mathcal{L}\}$$

The proof of the theorem is given in the appendix.

Further, recall that an algorithm for solving an online optimization problem is evaluated by a competitive ratio ς, i.e., it is said to be ς-competitive if the algorithm's objective is at least ς times the objective obtained by the optimal solution from the oracle using an offline calculation removing all uncertainties [26]. For the ONSP problem, a worst case instance denoted by $\phi \in \Phi$ can be defined as an input series of user demand requests over a period of time T with the minimal value from its objective function $Obj()$

$$\phi = \min[Obj(\mathcal{R}(t))](t = T)$$

and Φ is used to denote the set of all possible instances, i.e.,

$$\Phi = [Obj(\mathcal{R}(t))](t = T)$$

Given the definition of instance, the performance of an online algorithm can be measured using the competitive ratio, ς, which is referred to as the maximum ratio of the cost earned by the offline oracle, $OPT()$, and a solution from a particular online approximation algorithm, $ALG()$.

We define $OPT(\phi) = \min_{\phi \in \Phi} Obj(\phi)$ and $ALG(\phi) = \min_{\phi \in \Phi} Obj(\phi)$. And an approximation algorithm may return $\phi' \in \Phi$ such that $Obj(\phi') \neq OPT(\phi)$. According to the definition, we have

$$\varsigma = \max_{\Phi, Obj} \frac{OPT(\phi)}{ALG(\phi)} \tag{D.20}$$

Therefore the best and also the maximum competitive ratio is $\varsigma = 1$. A better algorithm has a higher competitive ratio because we want to find a minimal value of the objective function.

Using the definition above, we can derive the following theorems. Their proofs are exposed in the appendix.

Theorem 6.2
Algorithm 5.2 has a competitive ratio of $\varsigma_{greedy} = \Omega(\frac{|\mathcal{V}|+\lambda\mathcal{T}}{(1+\lambda\mathcal{T}|\mathcal{V}|)})$ to the optimal solution, where $\mathcal{T} > 0$ is a fixed period, $|\mathcal{V}|$ is the number of nodes and $\lambda > 0$ is the arrival rate.

Theorem 6.3
Algorithm 5.3 has a competitive ratio of $\varsigma_{ip} = \Omega(\frac{1+\mathcal{T}-|\mathcal{V}|+\mathcal{T}|\mathcal{V}|}{-1+\mathcal{T}+|\mathcal{V}|+\mathcal{T}|\mathcal{V}|})$ to the optimal solution, where $\mathcal{T} > 0$ is a fixed period, $|\mathcal{V}|$ is the number of nodes and $\lambda > 0$ is the arrival rate.

Theorem 6.4
Algorithm 5.4 has a competitive ratio of $\varsigma_{rl} = \Omega(\frac{1-|\mathcal{V}|+\mathcal{T}+|\mathcal{V}|\mathcal{T}}{1-|\mathcal{V}|+\mathcal{T}+\mathcal{T}|\mathcal{V}|+2\epsilon(|\mathcal{V}|-1)})$ to the optimal solution, where $\mathcal{T} > 0$ is a fixed period, $|\mathcal{V}|$ is the number of nodes, ϵ is the prediction error ($0 < \epsilon < 1$) and $\lambda > 0$ is the arrival rate.

From the competitive ratios of the three algorithms, derived in Theorem 6.2, 6.3 and 6.4, we are able to compare their performance. We firstly compare the Greedy algorithm and IP algorithm. It is clear that the competitive ratio of the IP algorithm, ς_{ip}, is closer to one than competitive ratio of the Greedy algorithm, ς_{greedy}, under the same environment parameters such as $|\mathcal{V}|$ and $\mathcal{T}$. Next, we note that, the PPO algorithm in the proposed RL framework has a bounded competitive ratio, ς_{rl}. The ratio ς_{rl} is almost the same as ς_{ip} when the prediction error approximates to zero. Therefore, the proposed RL framework is verified to reach similar performance in terms of the bounded competitive ratio.

7 Simulation

We analyze the performance of our proposed framework using simulations and show that it achieves better performance than the benchmark methods that do not consider the time-series relationship between users in terms of CPU time, network operation cost and SLA violation rate. In our simulation, two different algorithms (i.e., Greedy and IP) were used as the benchmark methods under our proposed framework.

7.1 Simulation Environment

Our simulation is implemented on a laptop computer with an NVIDIA GPU, an Intel Core i7 CPU, and 16 GB RAM. We refer [27] to provide a general network model and construct a random network topology for a scalable simulation. We also assume our simulation's network topology parameters, $|V|$ vertices and $|L|$ edges, are static during our simulation. Then all possible graphs can be obtained from the combinations of $|V|$ and $|L|$ but exclude the graphs with wrong orders between RANs, TNs and CNs. The network topology is then selected from all possible graphs and randomly assigned latency, capacity and cost with their corresponding distributions. Moreover, we simulate the user demand request from different mobile applications with different resource demands and latency metrics. We further emulate those user requests asynchronously reach the slice controller, and the controller serves user requests in parallel. The arrival model follows the Poisson process with a mean arrival rate where the intervals between arrival requests are sampled from an exponential distribution. Table D.2 lists the environment parameters in our simulation.

In addition, our IP algorithm solver uses the optimization library from the Google OR-tool [28]. The library is well-verified and tuned by much existing research and often outperforms a solver using self-implemented heuristic parameters. Therefore, we use the Google OR-tool library to implement our IP algorithm solver. Moreover, our PPO algorithm implementation consists of an actor neural network and a critic neural network. The actor neural network comprises an input layer, three hidden layers, and two fully connected layers. On the other hand, the critic neural network consists of two hidden layers, and three fully connected layers. The input layer transfers the features of the slicing network state to the hidden layers, whereas the output of the input layer is forwarded to the fully connected layers. All hidden layers use 64 neurons and are designed by the hyperbolic tangent (Tanh) activation function. The other parameters of the PPO algorithm list in Table D.3.

The actor network computes, $\pi(o_t, \theta_t)$, the probability distribution across the available actions (a subset of $\mathcal{R}(t)$) for all user demand requests. The critic network evaluates the expectation value, $V_\pi(o_t)$ for the action taken by the

actor. According to this evaluation, the policy function is updated to increase the reward. Indeed, the learning process is a mutual interaction where the actor continuously explores to obtain a higher probability of selecting suitable actions in different states, and the critic keeps assessing to return a more accurate value evaluation. And the agent learns the policy and stops the learning process when the reward converges.

7.2 Simulation Results

In this section, we introduce our simulation result to validate the performance of our framework in terms of the CPU time, SLA violation rate and network operation cost. We use the network slice controller's CPU time to evaluate the computational performance in different approaches without the impact from other components in our simulation. We define the network operation cost as the summation of all the slices' costs with respect to their physical links and average the cost by all requests as, $Cost_{avg}$, in Equation (D.21).

$$Cost_{avg} = \sum_{t=0}^{\mathcal{T}} \sum_{\substack{\forall n \in \mathcal{N}_t \\ \forall l_{i,j} \in \mathcal{L}}} \frac{P_{i,j} x_{i,j}^n}{|\mathcal{R}(t)|}, \ \forall r \in \mathcal{R}(t) \tag{D.21}$$

In addition, we define the SLA violation rate as the ratio between the number of provisioning requests and the number of all requests.

$$SLA = \sum_{t=0}^{\mathcal{T}} \frac{|\mathcal{R}(t)| - |\mathcal{N}_t|}{|\mathcal{R}(t)|\mathcal{T}} \tag{D.22}$$

After defining the performance metrics, we can compare our framework with the Greedy-only and IP-only benchmarks. The results demonstrate that our framework can fast approximate the optimal solution from the IP-only algorithm. Fig. D.5 presents the performance difference of CPU time to solve the ONSP problem. The results also show that our framework obtains a better solution than the Greedy-only algorithm. Fig. D.6 shows the comparison of slice operation cost, and Fig. D.7 presents the performance difference of SLA violation rate. In addition, Fig. D.8 presents the average waiting time in different traffic loads under different algorithms.

The Greedy algorithm has the worst cost performance and the highest SLA violation rate for two reasons. First, the iterative processing in the Greedy algorithm is sub-optimal because it ignores the dependency between requests. Second, the greedy solution only considers the slice provisioning within a time slot and neglects the time-series relations between time slots. The IP algorithm performs better than the Greedy algorithm regarding the cost and SLA violation rate in both low and high demands scenarios as it

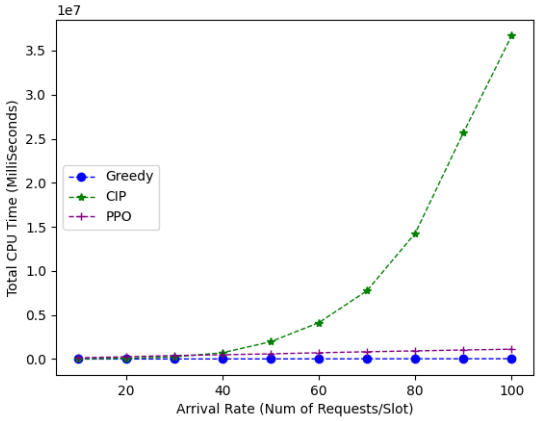

Fig. D.5: CPU Time Evaluation when Network Traffic Load Changing

considers the complete request set in the same time slot during optimization. However, the IP algorithm is proven to consume pseudo-polynomial time to search for the optimal solution. Comparing all the presented results, we demonstrate that our DRL framework has the lowest time consumption (see Fig. D.5). In addition, the DRL approach performs similarly to the IP algorithm regarding the SLA violation rate and network operation costs. Our framework can also mitigate the jitter of average waiting time between the different traffic loads (see Fig. D.8).

8 Conclusion

This research studied the Online Network Slicing Provisioning (ONSP) optimization problem and proposed a reinforcement learning framework to minimize network operation costs and Service-Level Agreements (SLA) violation rates. The reinforcement learning framework provides an improved slice controller based on a dynamic batch size prediction model. More specifically, the Proximal Policy Optimization (PPO) algorithm is used to learn the proposed model. A user demand pattern aims to be predicted from the model to obtain a suitable batch size for the ONSP optimization. Our framework efficiently approximates the solutions offered by an integer programming (IP) algorithm, albeit at a less computational time than the IP algorithm. Moreover, we evaluated the performance of the PPO algorithm and other benchmarks and derived the lower bound of algorithm performance. Numerical results also demonstrate the proposed reinforcement learning framework can approximate a similar performance with less computational time.

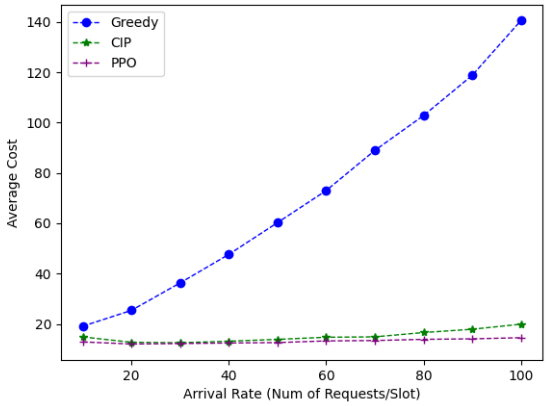

Fig. D.6: Slices Operation Cost Evaluation when Network Traffic Load Changing

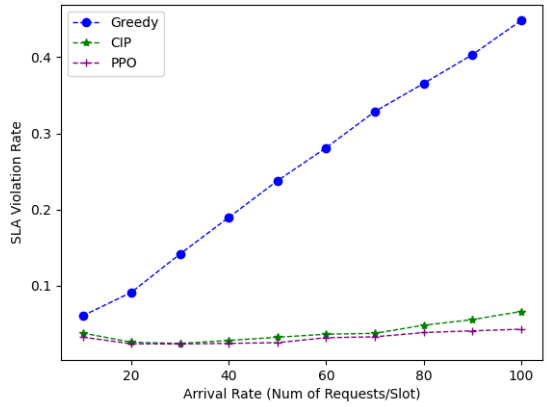

Fig. D.7: SLA Violation Rate Evaluation when Network Traffic Load Changing

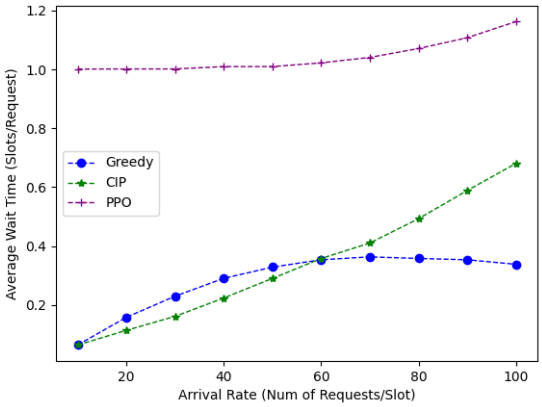

Fig. D.8: Wait Time Evaluation when Network Traffic Load Changing

9 Appendix

9.1 Proof of Theorem 6.1

Proof. To prove the theorem, we can firstly define the total network operation cost by calculating the function as: $f(\mathbf{z}, \mathbf{P}) = \sum_{\forall r \in \mathcal{R}} \sum_{\substack{\forall n \in \mathcal{N} \\ \forall l_{i,j} \in \mathcal{L}}} x_{i,j}^{n} P_{i,j} z_r$. Since the two objectives are linearly independent, the Lagrange multipliers exist. Let $|\mathbf{z}| = \sum_{z_r = 1 \in \mathbf{z}} z_r$. Then, we can consider its Lagrange function and therefore we will have two cases:

Case 1: There exists a vector $\mathbf{z}$ such that $|\mathbf{z}| > |\mathcal{N}|$. If we consider $|\mathbf{z}| = |\mathcal{N}| + 1$, then the network operation cost is larger due to $P_{i,j} > 0$. Hence, the objectives cannot be satisfied in case 1.

Case 2: There exists an vector $\mathbf{z}$ such that $|\mathbf{z}| < 1$. If we consider $|\mathbf{z}| = 0$, then the network utilization is also zero and SLA is fully violated. Hence, the objectives cannot be satisfied in case 2.
According to the two cases, the optimal solutions can only validate when $|\mathbf{z}| = \zeta$. □

9.2 Proof of Theorem 6.2

Proof. To prove the theorem, we consider a worst case that many user demands request the same source and destination nodes and exhaust the capacity at a link of network (i.e. generate a bottleneck). For example, a request with the demand value $C_{i,j}$ and the life time $\mathcal{T}$ slots comes be-

fore $\lambda\mathcal{T}$ requests with the demand value $\frac{C_{i,j}}{\lambda\mathcal{T}}$ and the life time 1 slot (i.e. $[(C_{i,j},\mathcal{T}),(\frac{C_{i,j}}{\lambda\mathcal{T}},1),\ldots,(\frac{C_{i,j}}{\lambda\mathcal{T}},1)]$). We use a link $l_{i,j}$ as an example but the example can be applied to all links in all slices. The greedy algorithm can only admit the first request and will miss the rest, since the network already offer all the link capacity to admit the first request. Therefore, the cost obtained from the greedy algorithm is:

$$ALG(\phi) = P_{i,j} + \lambda\mathcal{T}|V|P_{i,j} \tag{D.23}$$

On ther other hand, the cost of the optimal algorithm is:

$$OPT(\phi) = \lambda\mathcal{T}P_{i,j} + P_{i,j}|V| \tag{D.24}$$

In this way, we show that the competitive ratio of the greedy algorithm is at least $\Omega(\frac{|\mathcal{V}|+\lambda\mathcal{T}}{(1+\lambda\mathcal{T}|\mathcal{V}|)})$ □

9.3 Proof of Theorem 6.3

Proof. To prove the theorem, we can construct the worst case of the ONSP problem that repeat for every $2\mathcal{T}$ period if we assume the algorithm batch size is $\mathcal{T}-1$. Hence, we aim to have $2\lambda\mathcal{T}$ requests in that period. Then the worst case condition can be that the slice controller firstly receives the first $\lambda(\mathcal{T}-1)$ requests in the first $\mathcal{T}-1$ slots where the ith request's demand value and life time value are $\frac{C_{i,j}}{\lambda\mathcal{T}}$ and $\mathcal{T}$ respectively. (i.e. $[r_1 = (\frac{C_{i,j}}{\lambda\mathcal{T}},\mathcal{T}),\ldots,r_i,\ldots,r_{\mathcal{T}-1} = (\frac{C_{i,j}}{\lambda\mathcal{T}},\mathcal{T})]|i = 1\ldots\mathcal{T}-1$). Next, the rest of $\lambda(\mathcal{T}+1)$ requests arrive in the slice controller within the next $\mathcal{T}+1$ slots. Those $\lambda(\mathcal{T}+1)$ requests contain demand value and life time value are larger than $\frac{2C_{i,j}}{\lambda\mathcal{T}}$ (at $i=\mathcal{T}$) or $\frac{C_{i,j}}{\lambda\mathcal{T}}$ (others) and 1, respectively, i.e. $[r_{\mathcal{T}} = (\frac{2C_{i,j}}{\lambda\mathcal{T}},1), r_{\mathcal{T}+1} = (\frac{C_{i,j}}{\lambda\mathcal{T}},1),\ldots,r_i = (\frac{C_{i,j}}{\lambda\mathcal{T}},1),\ldots,r_{2\mathcal{T}} = (\frac{C_{i,j}}{\lambda\mathcal{T}},1)]|i = \mathcal{T}\ldots 2\mathcal{T}$. The IP algorithm can only admit the first $\lambda(\mathcal{T}-1)$ requests and will miss the rest, since the network already uses all the link capacity to admit the first $\lambda(\mathcal{T}-1)$ requests. Hence, the IP algorithm can only obtain the cost $P_{i,j}\lambda((\mathcal{T}-1)+|\mathcal{V}|(\mathcal{T}+1))$.

$$\begin{aligned} ALG(\phi) &= (\mathcal{T}-1)\lambda P_{i,j} + (\mathcal{T}+1)\lambda|V|P_{i,j} \\ &= (\mathcal{T}|V| + |V| + \mathcal{T} - 1)\lambda P_{i,j} \end{aligned} \tag{D.25}$$

However, the optimal algorithm should reject the first $\lambda(\mathcal{T}-1)$ requests and admit the next $\lambda(\mathcal{T}+1)$ requests. Therefore, the optimal cost should be $P_{i,j}\lambda((\mathcal{T}-1)|\mathcal{V}| + (\mathcal{T}+1))$

$$\begin{aligned} OPT(\phi) &= (\mathcal{T}-1)\lambda|V|P_{i,j} + (\mathcal{T}+1)\lambda P_{i,j} \\ &= (\mathcal{T}|V| - |V| + \mathcal{T} + 1)\lambda P_{i,j} \end{aligned} \tag{D.26}$$

To this end, we show that the competitive ratio of the IP algorithm is at least $\Omega(\frac{1+\mathcal{T}-|\mathcal{V}|+\mathcal{T}|\mathcal{V}|}{-1+\mathcal{T}+|\mathcal{V}|+\mathcal{T}|\mathcal{V}|})$. □

9.4 Proof of Theorem 6.4

Proof. To prove the theorem, we can also construct the worst case of the ONSP problem that repeat for every $2^k T$ slots, where $k \geq 1$. In addition, we assume the optimal cost value in the problem (D.13) is α

$$OPT(\phi) = \alpha \tag{D.27}$$

and therefore each request expects to spend average cost $\frac{\alpha}{2^k \mathcal{T} \lambda}$.

$$\begin{aligned}
&\mathbb{E}[ALG] = \sum_{i=1}^{2^k \mathcal{T}\lambda} (1-\epsilon)(\frac{\alpha}{2^k \mathcal{T}\lambda}) + \\
&\epsilon((\frac{\alpha}{2^k \mathcal{T}\lambda})(\frac{-1+|\mathcal{V}|+\mathcal{T}+|\mathcal{V}|\mathcal{T}}{1-|\mathcal{V}|+\mathcal{T}+|\mathcal{V}|\mathcal{T}})) \\
&= 2^k \mathcal{T}\lambda \frac{\alpha + \alpha\mathcal{T} - \alpha|\mathcal{V}| + \alpha\mathcal{T}|\mathcal{V}|(1-\epsilon) - \alpha\epsilon - \alpha\epsilon\mathcal{T} + \alpha\epsilon|\mathcal{V}|}{2^k \mathcal{T}\lambda(1+\mathcal{T}-|\mathcal{V}|+|\mathcal{V}|\mathcal{T})} \\
&+ 2^k \mathcal{T}\lambda \frac{-\alpha\epsilon + \alpha\epsilon\mathcal{T} + \alpha\epsilon|\mathcal{V}| + \alpha\epsilon|\mathcal{V}\mathcal{T}|}{2^k \mathcal{T}\lambda(1+\mathcal{T}-|\mathcal{V}|+|\mathcal{V}|\mathcal{T})} \\
&= \frac{\alpha(1-|\mathcal{V}|+\mathcal{T}+\mathcal{T}|\mathcal{V}|+2\epsilon(|\mathcal{V}|-1))}{1+\mathcal{T}-|\mathcal{V}|+|\mathcal{V}|\mathcal{T}}
\end{aligned} \tag{D.28}$$

In this way, we can estimate the cost from the reinforcement learning algorithm, ALG, with a small prediction error, ϵ. Refer the result from the Theorem 6.3, we can expect the average cost is at least $(\frac{\alpha}{2^k \mathcal{T}\lambda})(\frac{-1+|\mathcal{V}|+\mathcal{T}+\mathcal{T}|\mathcal{V}|}{1-|\mathcal{V}|+\mathcal{T}+\mathcal{T}|\mathcal{V}|})$ if the request was not served in a correct batch. The expected cost of ALG denoted as $\mathbb{E}[ALG]$ should be $\sum_{i=1}^{2^k \mathcal{T}\lambda}(1-\epsilon)(\frac{\alpha}{2^k \mathcal{T}\lambda}) + \epsilon((\frac{\alpha}{2^k \mathcal{T}\lambda})(\frac{-1+|\mathcal{V}|+\mathcal{T}+\mathcal{T}|\mathcal{V}|}{1-|\mathcal{V}|+\mathcal{T}+\mathcal{T}|\mathcal{V}|})$. And thus the competitive ratio of algorithm 5.4 is

$$\frac{OPT(\phi)}{ALG(\phi)} = \frac{\alpha}{\mathbb{E}[ALG]} = \frac{1-|\mathcal{V}|+\mathcal{T}+|\mathcal{V}|\mathcal{T}}{1-|\mathcal{V}|+\mathcal{T}+\mathcal{T}|\mathcal{V}|+2\epsilon(|\mathcal{V}|-1)} \tag{D.29}$$

□

References

[1] P. Rost, C. Mannweiler, D. S. Michalopoulos, C. Sartori, V. Sciancalepore, N. Sastry, O. Holland, S. Tayade, B. Han, D. Bega, D. Aziz, and H. Bakker, "Network slicing to enable scalability and flexibility in 5g mobile networks," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 72–79, 2017.

[2] X. Li, C. Guo, L. Gupta, and R. Jain, "Efficient and secure 5g core network slice provisioning based on vikor approach," *IEEE Access*, vol. 7, pp. 150 517–150 529, 2019.

[3] R. Gouareb, V. Friderikos, and A.-H. Aghvami, "Virtual network functions routing and placement for edge cloud latency minimization," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2346–2357, 2018.

[4] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 4, pp. 1888–1906, 2013.

[5] C. Williamson, E. Halepovic, H. Sun, and Y. Wu, "Characterization of cdma2000 cellular data network traffic," in *in 30th Annual IEEE Conference on Local Computer Networks*, 2005, pp. 712–719.

[6] U. Paul, P. Subramanian, M. M. Buddhikot, and S. Das, "Understanding traffic dynamics in cellular data networks," in *In IEEE Infocom*, 2011.

[7] R. Keralapura, A. Nucci, L. Gao, and Z. li Zhang, "Profiling users in a 3g network using hourglass co-clustering," 2010.

[8] M. Zubair, S. Lusheng, J. Alex, X. Liu, and J. Wang, "Characterizing and modeling internet traffic dynamics of cellular devices," 2011.

[9] Y. Zhang and Åke Arvidsson, "Understanding the characteristics of cellular data traffic," in *in ACM SIGCOMM CellNet Workshop*, 2012.

[10] E. Mucelli Rezende Oliveira, A. C. Viana, K. P. Naveen, and C. Sarraute, "Measurement-driven mobile data traffic modeling in a large metropolitan area," in *2015 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2015, pp. 230–235.

[11] V. Sciancalepore, L. Zanzi, X. Costa-Pérez, and A. Capone, "Onets: Online network slice broker from theory to practice," *IEEE Transactions on Wireless Communications*, vol. 21, no. 1, pp. 121–134, 2022.

[12] Q.-T. Luu, S. Kerboeuf, and M. Kieffer, "Uncertainty-aware resource provisioning for network slicing," *IEEE Trans. on Netw. and Serv. Manag.*, vol. 18, no. 1, pp. 79–93, mar 2021.

[13] R. Ford, A. Sridharan, R. Margolies, R. Jana, and S. Rangan, "Provisioning low latency, resilient mobile edge clouds for 5g," in *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2017, pp. 169–174.

[14] A. Baumgartner, V. S. Reddy, and T. Bauschert, "Mobile core network virtualization: A model for combined virtual core network function placement and topology optimization," in *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, 2015, pp. 1–9.

[15] T. Bauschert and V. S. Reddy, "Genetic algorithms for the network slice design problem under uncertainty," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ser. GECCO '19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 360–361.

[16] C.-C. Wu, V. Friderikos, and C. Stefanovic, "Multi-objective provisioning of network slices using deep reinforcement learning," *arXiv*, pp. 1–15, 2022.

[17] 3GPP, "5g; management and orchestration; provisioning," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 28.531, 01 2022, version 16.12.0. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3274

[18] S. Ross, *Introduction to Probability Models*, eleventh edition ed. Academic Press, 2014.

[19] T. Achterberg *et al.*, "Constraint integer programming: A new approach to integrate cp and mip," in *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, 2008, pp. 6–20.

[20] F. Eisenbrand *et al.*, "An algorithmic theory of integer programming," 2019.

[21] A. Gleixner, "Exact and fast algorithms for mixed-integer nonlinear programming," Ph.D. dissertation, Technischen Universitat Berlin, 2015. [Online]. Available: http://dx.doi.org/10.14279/depositonce-4938

[22] Q. Cappart, T. Moisan, L.-M. Rousseau, I. Pr'emont-Schwarz, and A. A. Ciré, "Combining reinforcement learning and constraint programming for combinatorial optimization," *ArXiv*, vol. abs/2006.01610, 2021.

[23] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *ArXiv*, vol. abs/1707.06347, 2017.

[24] A. Stooke and P. Abbeel, "Accelerated methods for deep reinforcement learning," *ArXiv*, vol. abs/1803.02811, 2018.

[25] L. Qu, C. Assi, K. Shaban, and M. J. Khabbaz, "A reliability-aware network service chain provisioning with delay guarantees in nfv-enabled

enterprise datacenter networks," *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 554–568, 2017.

[26] H.-J. Böckenhauer, D. Komm, R. Královič, R. Královič, and T. Mömke, "On the advice complexity of online problems," in *Algorithms and Computation*, Y. Dong, D.-Z. Du, and O. Ibarra, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 331–340.

[27] M. E. Newman *et al.*, "Random graphs as models of networks," *Handbook of graphs and networks*, vol. 1, pp. 35–68, 2003.

[28] L. Perron and V. Furnon, "Or-tools," 2022. [Online]. Available: https://developers.google.com/optimization/

Table D.1: Notations

Notation	Description
$\mathcal{G}$	network graph
$\mathcal{V}$	the set of network nodes
v_i	different network nodes
$\mathcal{L}$	the set of wired-cum-wireless network links
$l_{i,j}$	a link between nodes v_i and v_j
$\mathbf{C}$	the vector of available link capacity
$C_{i,j}$	the available capacity of direct link $l_{i,j}$
$\alpha(i,j)$	the occupied capacity of direct link $l_{i,j}$
$\mathbf{D}$	the vector of the link delay
$D_{i,j}$	the delay of direct link $l_{i,j}$
$\mathbf{P}$	the vector of the link active network operation cost
$P_{i,j}$	the active operation cost of direct link $l_{i,j}$
$\mathcal{A}(t)$	the set of new arrival user demand requests at t
$\mathcal{R}(t)$	the set of all user demand requests at t
$\mathcal{Q}(t)$	the set of backlog user demand requests at t
r	a user demand request in the set $\mathcal{R}(t)$
v_s	the source node of a user demand request
v_u	the destination node of a user demand request
b_r	the required data rate of a user demand request
d_r	the delay requirement of a user demand request
ι_r	the demand type of a user demand request
a_r	the initial time slot of a user demand request
h_r	the number of slots of a user demand request life time
n	a network slice
$v_{n,s}$	the source node of a network slice
$v_{n,u}$	the destination node of a network slice
E_n	a set of virtual links of a network slice
κ_n	slice delay of a network slice
y_n	slice load of a network slice
ι_n	the slice type of a network slice
X	a slice provisioning matrix
$x_{i,j}^n$	an element of the matrix X
$\mathcal{T}$	a slotted time interval
$P_t(\mathcal{Q})$	the buffering probability at t
λ_t	arrival rate at t
$\widetilde{\lambda}_t$	artificial arrival rate at t
ϱ_t	backlog rate at t
z_r	a decision variable for request r
$\boldsymbol{z}$	a vector of decision variables
Υ_t	a set of reject request at t
$\mathcal{N}_t$	a set of network slice
f_k	objective function k

Table D.2: Simulation Parameters

Parameter Name	Value/Distribution(mean, var)
Number of Nodes $\|V\|$	8
Number of Physical Links $\|L\|$	12
Capacity of Links	Uniform Distribution (100, 200)
Latency of Links	Uniform Distribution (1, 10)
Cost of Links	Uniform Distribution (1, 20)
Simulation Duration $\mathcal{T}$	5000 slots
Request Demand Requirement (b)	Normal Distribution (0, 0.1)
Request Latency Requirement (d)	Normal Distribution (1, 0.1)
Request Initial Time a_r	Normal Distribution (0, 0.1)
Request Life Time h_r	Normal Distribution (1, 0.1)
Request Source/Destination Nodes	Normal Distribution draw from $\|V\|$
Request Arrival Model	Poisson Process

Table D.3: Hyperparameters of PPO Algorithm

Parameter Name	Value
Number of epochs	10
Number of minibatch size $\mathcal{B}$	32
Discount (γ)	0.90
Number of iterations K	8000
Initial parameters θ_0	0

ISSN (online): 2446-1628
ISBN (online): 978-87-7573-720-8

AALBORG UNIVERSITY PRESS