PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

Belhadj Bouchaib University of Ain Temouchent, Algeria
Faculty of Science and Technology
Department of Mathematics and Computer Science

Master's Thesis

Submitted in partial fulfillment of the requirements for the degree of Master

Research Topic:

Dynamic Resource Allocation for 5G Network Slices Using a Hybrid DRL–GA Approach

Presented by:
Douaa El Aldja REGHAOUAT
Maroua DAHMANI

Supervisor:
Dr. A. BENZERBADJ (UAT BB)
Co-Supervisor:
Dr. Oumaya BAALA (UTBM)

Before the Jury Composition:

Dr. K. Chaoui	UAT.B.B (Ain Temouchent)	**President**
Dr. D. Merad Boudia	UAT.B.B (Ain Temouchent)	**Examiner**
Dr. A. BENZERBADJ	UAT.B.B (Ain Temouchent)	**Supervisor**
Dr. Oumaya BAALA	UTBM (France)	**Co-Supervisor**

Academic Year: 2025 – 2026

Abstract

Dynamic resource allocation between co-existing network slices is one of the most challenging problems in 5G Radio Access Network management. In a Vehicle-to-Everything (V2X) highway scenario, a URLLC slice serving safety-critical collision-avoidance messages and an eMBB slice serving high-definition cameras and passenger internet access must share a fixed 100 MHz radio pool under non-stationary, multi-regime traffic conditions. Satisfying the strict latency constraint of the URLLC slice and the load-dependent bandwidth requirement of the eMBB slice simultaneously, at every one second decision epoch, is an NP-hard problem that classical and standalone intelligent methods cannot fully resolve.

This thesis proposes a **Hybrid Deep Q-Network + Adaptive Genetic Algorithm (Hybrid DQN+GA)** framework that addresses three structural limitations identified in the state of the art: the discrete action ceiling of DQN-based agents, the cold-start inefficiency of standalone Genetic Algorithms, and the absence of a bidirectional co-adaptation loop between learning and evolutionary search. At every decision epoch, the DQN provides a fast, experience-driven, discrete allocation seed; the Adaptive GA refiner searches the continuous bandwidth interval when a constraint violation is detected, using a multi-directional warm-start strategy seeded from the DQN output and domain-knowledge candidates; and the resulting refined allocation is fed back to train the DQN, forming a self-improving co-adaptation loop.

The framework is evaluated in a simulated environment with five traffic regimes Normal, Rush-hour, Accident, Peak-eMBB, and Night over 500 training episodes and 100 greedy evaluation episodes, against a Static Allocation baseline and a standalone DQN. The Hybrid agent achieves an overall QoS satisfaction of **75.34%** (vs. 74.10% for DQN and 65.96% for Static), an eMBB satisfaction of **53.49%** (+11.0% over DQN), a per-step violation rate of **0.493** (−4.8% over DQN), and a mean reward of **7.51**. During safety-critical accident events, the Hybrid agent allocates 69 MHz to the URLLC slice 19 MHz more than the DQN— maintaining a latency of 4.8 ms under near-maximum load. All decisions are produced with a computational overhead below 1 ms per epoch, confirming practical deployability on a Mobile Edge Computing server co-located with the base station.

Keywords: 5G network slicing, dynamic resource allocation, Deep Reinforcement Learning, Deep Q-Network, Genetic Algorithm, URLLC, eMBB, V2X, hybrid optimization.

ملخص

تُعد عملية التخصيص الديناميكي للموارد بين شرائح الشبكة المتعايشة من أكثر المشكلات تعقيدًا في إدارة شبكة الوصول الراديوي للجيل الخامس .(5G) ففي سيناريو طريق سريع من نوع المركبة بكل شيء ،(V2X) يجب أن تتشارك شريحة URLLC المخصصة لرسائل تجنّب التصادم ذات الأهمية الأمنية الحرجة، وشريحة eMBB المخصصة للكاميرات عالية الدقة والوصول إلى الإنترنت للركاب، حيّزًا راديويًا ثابتًا قدره 100 ميغاهرتز، تحت ظروف حركة مرور غير مستقرة ومتعددة الأنظمة. ويُعد تحقيق قيد الكمون الصارم لشريحة URLLC ومتطلب النطاق الترددي المعتمد على الحمل لشريحة eMBB في آن واحد، في كل حقبة قرار مدتها ثانية واحدة، مشكلة من فئة NP-hard لا يمكن للطرق الكلاسيكية ولا الطرق الذكية المنفردة حلّها بشكل كامل.

تقترح هذه الأطروحة إطار عمل **هجين يجمع بين شبكة Q العميقة والخوارزمية الجينية التكيفية Hybrid DQN+GA** يعالج ثلاث قيود بنيوية محددة في الأدبيات السابقة : سقف الفعل المنفصل لدى عملاء ،DQN وعدم كفاءة البداية الباردة للخوارزميات الجينية المستقلة، وغياب حلقة تكيّف متبادلة بين التعلّم والبحث التطوري. في كل حقبة قرار، يوفر DQN بذرة تخصيص منفصلة سريعة مستندة إلى الخبرة ; ثم يقوم مُحسِّن الخوارزمية الجينية التكيفية بالبحث في الفترة المستمرة للنطاق الترددي عند اكتشاف انتهاك لأحد القيود، باستخدام استراتيجية بداية متعددة الاتجاهات مُهيّأة من مخرجات DQN ومرشحين قائمين على المعرفة بالمجال ; ويُعاد تغذية التخصيص المُحسَّن الناتج لتدريب ،DQN مكوّنًا بذلك حلقة تكيّف متبادلة ذاتية التحسّن.

تم تقييم هذا الإطار في بيئة محاكاة تضم خمسة أنظمة لحركة المرور : عادي، ساعة الذروة، حادث، ذروة ،eMBB وليلي، عبر 500 حلقة تدريب و 100 حلقة تقييم جشعة، مقارنةً بخط أساس للتخصيص الثابت وعميل DQN مستقل. يحقق العميل الهجين معدل رضا جودة خدمة (QoS) إجماليًا بلغ **34%.75** (مقابل 10%.74 لـ DQN و 96%.65 للتخصيص الثابت)، ورضا eMBB بنسبة **49%.53** (+11.0%) مقارنة بـ ،DQN ومعدل انتهاك لكل خطوة بلغ **493.0** (−4.8%)مقارنة بـ ،DQN ومكافأة متوسطة قدرها **51.7**. وخلال أحداث الحوادث الحرجة أمنيًا، يخصص العميل الهجين 69 ميغاهرتز لشريحة ،URLLC أي 19 ميغاهرتز أكثر من DQN محافظًا على كمون قدره 8.4 ملي ثانية تحت حمل شبه أقصى. وتُنتَج جميع القرارات بعبء حسابي يقل عن 1 ملي ثانية لكل حقبة، مما يؤكد إمكانية النشر العملي على خادم حوسبة الحافة المتنقلة (MEC) المتموضع مع محطة القاعدة.

الكلمات المفتاحية : تقسيم شبكة الجيل الخامس إلى شرائح، التخصيص الديناميكي للموارد، التعلم المعزز العميق، شبكة Q العميقة، الخوارزمية الجينية، ،URLLC ،eMBB ،V2X التحسين الهجين.

Dedication

To my beloved parents, whose endless love, prayers, and sacrifices have been the foundation of my success and the light guiding my path.

To myself, with immense pride, for the resilience, hard work, and strength I found within to overcome every challenge and never give up on this dream.

To Halkat Nour, for being a sanctuary of faith and peace that anchored my soul and uplifted my spirit throughout this long academic journey.

And to my dear friends, for their constant encouragement, laughter, and for making this heavy path so much lighter.

I dedicate this achievement to all of you, and to the strength that brought me here.

Douaa El Aldja REGHAOUAT

Dedication

To myself, with gratitude, pride, and respect. To the perseverance I managed to maintain, the discipline I acquired, and the moments of doubt that I transformed into strength and maturity. This is a testament to the ambitious child within me who insisted on moving forward.

To my beloved parents, the pillars of my life. To my dear father, my backbone, sanctuary, and first love. To my precious mother, the secret of my strength, whose endless sacrifices and unconditional prayers were my shield and light in the darkness. Every success I achieve is a reflection of your devotion.

To my brother and my sister, my solid ribs and the safety of my days. Thank you for your constant presence, your smiles that alleviated my fatigue, and for being my unwavering source of joy and encouragement.

To my dear friend El Aldja, my companion of the path, who shared with me the late nights, the struggles of this project, and the beautiful memories of our university years. Thank you for your sincere support and for making this journey so much more meaningful.

To Halakat Nour, for the blessed companionship, spiritual strength, and the beautiful light you brought to my life, constantly reminding me of the ultimate purpose of seeking knowledge.

To my extended family, loved ones, and everyone who supported me, whose faith, kind words, and encouragement gave me the motivation to succeed and stay on course.

To those who teach the world the meaning of strength, and to every young mind deprived of their academic dreams by harsh realities,I dedicate this work to your unyielding courage and resilience...To GAZA.

Maroua DAHMANI

Acknowledgements

First and foremost, We must express our deepest gratitude to Almighty Allah for giving us the strength, health, and patience to complete this master's thesis and achieve this milestone.

We would like to express our sincere gratitude and deepest respect to our supervisor, for their invaluable guidance, constant encouragement, and immense patience throughout this project. Their academic insights, constructive feedback, and continuous support have been instrumental in the completion of this work.

We are also highly grateful to the members of the jury for taking the time to review this manuscript, examine this work, and provide their valuable remarks and suggestions to improve its quality.

Our appreciation also extends to all the professors and faculty members of the University who have contributed to our academic growth and provided us with the knowledge and skills necessary for future journey.

Finally, We want to thank everyone who has lent a helping hand, whether through insightful discussions, technical assistance, or simple words of motivation during the moments when it was needed most. To all of you, We express our heartfelt appreciation.

Maroua DAHMANI & Douaa El Aldja REGHAOUAT

Contents

List of Figures

List of Tables

List of Abbreviations

3GPP 3rd Generation Partnership Project . . . 1
5G Fifth Generation of Mobile Networks . . . 1
5GC 5G Core Network . . . 7
5QI 5G QoS Identifier . . . 34
6G Sixth Generation of Mobile Networks . . . 1
AMF Access and Mobility Management Function . . . 7
C-V2X Cellular Vehicle-to-Everything . . . 33
CSV Comma-Separated Values . . . 55
CU Centralized Unit . . . 4
CUDA Compute Unified Device Architecture . . . 55
D3QN Dueling Double Deep Q-Network . . . 27
DDQN Double Deep Q-Network . . . 15
DEAP Distributed Evolutionary Algorithms in Python . . . 52
DNN Deep Neural Network . . . 17
DQN Deep Q-Network . . . vii
DRL Deep Reinforcement Learning . . . 2
DU Distributed Unit . . . 4
eCPRI Enhanced Common Public Radio Interface . . . 33
eMBB Enhanced Mobile Broadband . . . vii
GA Genetic Algorithm . . . vii
gNB gNodeB . . . 6
ITU-R International Telecommunication Union Radiocommunication Sector . . . 35
KKT Karush-Kuhn-Tucker . . . 76
MDP Markov Decision Process . . . 3
MEC Multi-access Edge Computing . . . 9
MLP Multilayer Perceptron . . . 47
mMTC massive Machine Type Communications . . . 2
NFV Network Functions Virtualisation . . . 1

General Introduction

Modern wireless communications are undergoing a fundamental transformation. The Fifth Generation of Mobile Networks (5G), standardised by the 3rd Generation Partnership Project (3GPP) across Releases 15 through 18, evolves into a programmable, multi-service platform capable of simultaneously supporting services with vastly conflicting technical requirements on a single shared physical infrastructure. Time-critical vehicle safety applications demanding latency below 10 ms must coexist with high throughput 4K video streams requiring tens of megahertz of sustained bandwidth, without mutual performance degradation. Three architectural innovations enable this: *flexible numerology* in the 5G New Radio (NR) interface, the *Service-Based Architecture (SBA)* of the cloud native core, and most centrally for this work *network slicing*, which logically partitions the physical infrastructure into isolated virtual networks, each governed by a Service Level Agreement (SLA). Looking ahead, the ITU-R IMT-2030 vision for Sixth Generation of Mobile Networks (6G) networks targets peak rates of 1 Tbps, latency below 0.1 ms, and connection densities exceeding 10^7 devices/km^2, confirming that intelligent adaptive resource management is a long-term necessity, not a 5G-specific solution.

Network slicing shifts mobile networking toward a Network-as-a-Service model where, enabled by Software-Defined Networking (SDN) and Network Functions Virtualisation (NFV), the physical substrate is virtualised into independent logical slices under strict SLA guarantees. The ITU-R IMT-2020 framework defines three canonical service types: Enhanced Mobile Broadband (eMBB) for high data rates, Ultra-Reliable Low-Latency Communications (URLLC) targeting latency under 1 ms and packet loss

below 10^{-5}, and massive Machine Type Communications (mMTC) for dense IoT deployments. In the Vehicle-to-Everything (V2X) highway scenario studied in this thesis, a URLLC slice handles safety-critical collision-avoidance messaging while an eMBB slice simultaneously serves roadside cameras and passenger internet access—both sharing a fixed 100 MHz radio pool. Dynamically partitioning this pool at every decision epoch to satisfy the latency constraint of the URLLC slice and the bandwidth demand of the eMBB slice is an NP-hard problem that is inherently sequential, non-stationary, and model-uncertain, rendering classical mathematical optimisation impractical in real-time deployments.

Two prominent families of intelligent optimisation have emerged to address this challenge. Deep Q-Network (DQN) allow an agent to learn optimal allocation policies through direct environmental interaction without an explicit system model, offering temporal memory and rapid policy retrieval. Genetic Algorithm (GA) provide gradient-free global search over continuous, non-convex solution spaces, excelling at fine-grained constraint handling. The two paradigms are deeply complementary: Deep Reinforcement Learning (DRL) provides speed and memory; GAs provides continuous space precision. Yet each suffers from a structural weakness when used alone DRL is confined to coarse discrete action spaces, while GA is memoryless across sequential epochs. Crucially, no existing framework establishes a bidirectional feedback loop in which both components co-adapt and improve one another at every decision step. The exact nature of these gaps is analysed in Chapter 1, where the related literature is surveyed and three precise structural limitations are formally identified.

These observations motivate the central question of this thesis:

> *Can a hybrid DRL–GA framework outperform its individual components for dynamic resource allocation across co-existing 5G network slices under non-stationary traffic conditions?*

To answer this question, we propose a **Hybrid Deep Q-Network + Adaptive Genetic Algorithm** framework that integrates the DQN and GA through a multi-directional

warm-start seeding strategy, an adaptive computational budget mechanism, and a bidirectional co-adaptation loop, evaluated in a simulated 5G V2X environment with URLLC and eMBB slices under five realistic traffic regimes.

The remainder of this thesis is organised into three chapters. Chapter 1 reviews the 5G architecture, network slicing principles, the resource allocation problem, the theoretical foundations of DRL and GAs, and the related literature, concluding with the formal derivation of the research objectives. Chapter 2 presents the V2X system model, the constrained Markov Decision Process (MDP) formulation, the DQN and Adaptive GA components, and the complete hybrid framework with its per-epoch decision workflow. Chapter 3 provides empirical validation across five traffic regimes, comparing the Hybrid agent against a Static Allocation baseline and a standalone DQN in terms of convergence, accuracy, and regime-aware allocation behaviour. A General Conclusion then summarises the principal findings, acknowledges current limitations, and outlines future research directions toward scalable intelligent resource management in 5G and beyond.

Chapter 1

Background and State of the Art

1.1 Introduction

The arrival of 5G mobile networks represents a defining shift in wireless communications. We are moving away from a traditional, one-size-fits-all broadband model toward a programmable, multi-service platform. This new landscape allows for the simultaneous support of diverse applications that often have competing and even conflicting requirements. At the heart of this evolution are three core pillars: 5G New Radio (NR), the SBA of the 5G Core, and network slicing. The latter is the essential mechanism that allows a single physical infrastructure to be virtualised into isolated logical networks, each custom-built for a specific service.

This chapter lays out the theoretical and contextual groundwork for the hybrid framework proposed in this work. We begin by diving into the 5G system architecture, exploring everything from the Next Generation Radio Access Network (NG-RAN), including its breakdown into Centralized Unit (CU), Distributed Unit (DU), and Roadside Unit (RSU) entities, to the service-based core network and its primary functions. From there, we introduce the concept of network slicing, examining its three-layer architecture and core principles: isolation, elasticity, and end-to-end optimisation. We also examine the three standard service categories eMBB, URLLC, and mMTC, which represent the different types of traffic sharing the same infrastructure.

Once the architectural foundation is set, the chapter tackles the complex challenge of resource allocation in 5G network slicing. We review the four-phase management lifecycle and the ”hard” constraints such as QoS bounds, SLA compliance, and inter-slice isolation that make this problem NP-hard. Because classical mathematical methods struggle with this level of computational complexity, we look toward two families of intelligent optimisation: DRL, which naturally fits the sequential, shifting nature of resource allocation, and GAs, which offer a robust, gradient-free way to navigate messy, non-convex solution spaces.

The chapter wraps up with a structured look at existing research, covering DRL-based slice management, GA approaches, and the emerging field of hybrid DRL–evolutionary frameworks. By synthesising the existing literature, we identify three specific limitations in current methods. These limitations constitute the research gap that motivates the objectives and necessity of this project.

1.2 5G Architecture

Fifth-generation (5G) mobile networks, standardised by the 3rd Generation Partnership Project (3GPP), beginning with Release 15 (2018) and extended through Releases 16 and 17, represent a fundamental redesign of the mobile network architecture[1]. Unlike LTE, which was conceived primarily as a single-service broadband technology, 5G is designed from the ground up as a programmable, multi-service platform capable of simultaneously supporting applications with radically different requirements on the same physical infrastructure.This ambition is expressed through three pillars: a flexible, software-driven radio interface (5G New Radio, NR), a Service-Based Core Network Architecture (SBA), and the virtualization technology of network slicing [2], as shown in Fig. 1.1.

The defining characteristic of 5G NR at the radio interface is the introduction of flexible numerology, which supports subcarrier spacings of 15, 30, 60, 120, and 240 kHz. This allows the same spectrum to be configured differently for latency sensitive URLLC

traffic (utilising shorter subcarriers and shorter slots) and high-throughput eMBB traffic. 5G NR also supports operation in two frequency ranges: FR1 (sub-6 GHz, including the 3.5 GHz n78 band used in this thesis) and FR2 (millimetre wave, 24–52 GHz) [1]. The scenario studied in this thesis utilises FR1 exclusively, with a 100 MHz bandwidth at 3.5 GHz, corresponding to 273 Physical Resource Blocks (PRBs) at a 30 kHz subcarrier spacing. The PRB, consisting of 12 subcarriers in frequency and one 0.5 ms slot in time, is the fundamental radio resource unit that the allocation agent must partition among slices.

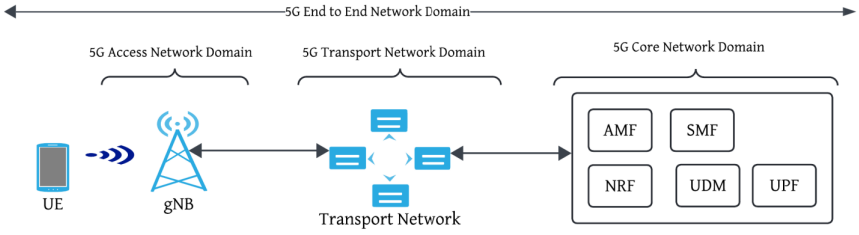

Figure 1.1: 5G end-to-end network scope diagram [19].

1.2.1 The NG-RAN and the gNB

The 5G base station, designated the gNodeB gNodeB (gNB)), is the central element of the NG-RAN, as shown in Fig. 1.2. The gNB is responsible for all radio layer functions: physical layer signal processing, MAC layer scheduling, ARQ/HARQ management, mobility management (handover decision and execution), and Radio Resource Control (RRC), the signalling protocol through which UEs register, authenticate, and receive radio configuration from the network [1].

Internally, the gNB is functionally decomposed into three entities following the 3GPP CU/DU/RU split defined in TS 38.300 [1]:

- CU: hosts the PDCP (Packet Data Convergence Protocol) and RRC layers. The CU is connected to the 5G Core via the N2 (control plane) and N3 (user plane) interfaces.
- DU: hosts the RLC (Radio Link Control), MAC (Medium Access Control), and

high-PHY layers. The MAC layer contains the scheduler, the component that maps PRBs to individual UEs within each slice at each Transmission Time Interval (TTI = 0.5 ms).

- RU: implements the low-PHY layer and RF front-end (digital-to-analogue conversion, power amplification, antenna interface). The RSU is physically mounted on the tower and connected to the DU via the fronthaul interface (eCPRI protocol over fiber).

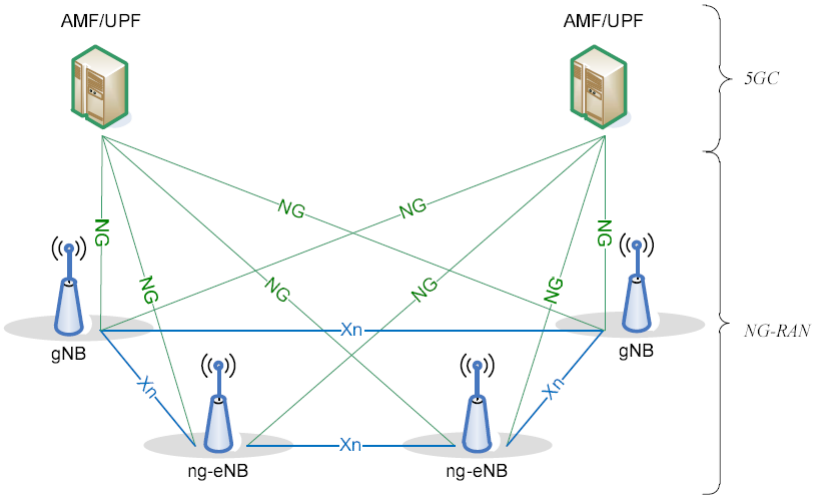

Figure 1.2: Overall Architecture [2].

1.2.2 The 5G Core Network

The 5G Core Network (5G Core Network (5GC)), standardised in 3GPP TS 23.501 [1], adopts a SBA in which all Network Functions (NFs) expose RESTful HTTP/2 APIs and communicate over the Service-Based Interface (SBI) as shown in Fig. 1.3. This decouples NF logic from the transport layer, enabling cloud-native deployment, independent scaling of individual NFs, and dynamic configuration of network slices without restarting the core. The principal NFs involved in network slicing are:

- **Access and Mobility Management Function (AMF):** the first point of contact

for UE registration and authentication. The AMF selects the appropriate Network Slice Selection Function (NSSF)-designated slice instance for each UE based on the requested S-NSSAI.

- **Session Management Function (SMF):** manages PDU session lifecycle (establishment, modification, release). The SMF selects the User Plane Function (UPF) anchor for each session and configures the QoS profile guaranteed bit rate (GBR), maximum bit rate, 5QI for each bearer.
- **User Plane Function (UPF):** the data-plane forwarding anchor. It routes user traffic between the gNB (N3 interface) and the external data network (N6 interface).
- **Network Slice Selection Function (NSSF):** selects the network slice instance(s) for a UE based on its requested S-NSSAI (Slice/Service Type + optional Slice Differentiator).
- **Policy Control Function (PCF):** enforces QoS and charging policies. The PCF communicates the per-slice QoS profile to the SMF, which propagates it to the UPF and gNB scheduler.

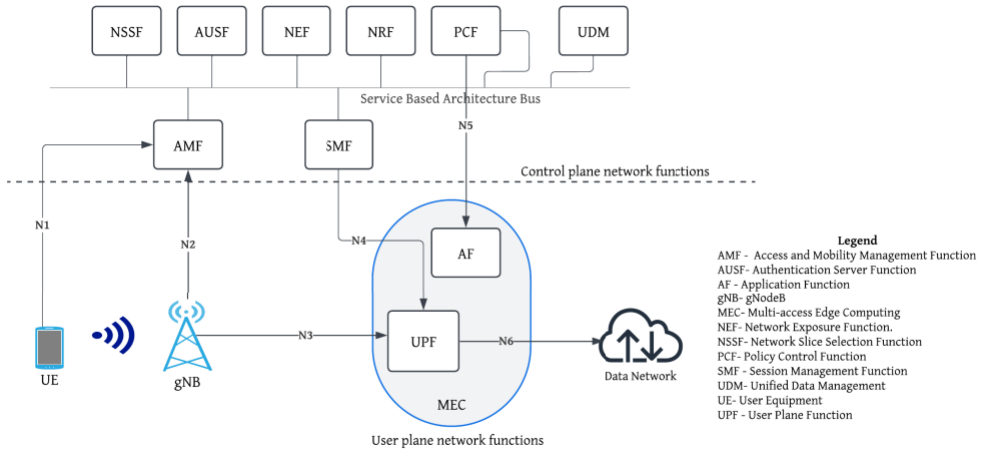

Figure 1.3: 5G network architecture diagram [19].

As summarised by Khan et al. [4], the physical infrastructure of a 5G network slicing system extends beyond the gNB and 5GC to include edge computing nodes, cloud data centres, RSUs, and satellite links.

1.3 Network Slicing

1.3.1 Concept, Definition, and Architecture

Network slicing is the technology that enables a single physical network infrastructure to be partitioned into multiple independent logical networks called slices, each tailored to the specific requirements of a service category or tenant application [4]. As Khan et al. [4] define it, network slicing represents the shift from a network-as-an-infrastructure paradigm to a network-as-a-service model, enabling numerous 5G smart services with diverse requirements over a shared substrate. Hurtado Sánchez et al. [6] identify network slicing as one of the two key enabling technologies of 5G and 6G, alongside DRL, precisely because it allows the same physical infrastructure to simultaneously support services whose requirements differ by orders of magnitude for instance, a latency of 1 ms for URLLC versus 50 ms for eMBB, or a reliability of 99.999% versus 99.9%.

The 3GPP architecture for network slicing follows a three-layer model [4] as shown in Fig. 1.4:

- **Infrastructure Layer:** all physical resources gNBs, transport links, Multi-access Edge Computing (MEC) servers, core network nodes, cloud data centers managed by the Physical Infrastructure Provider (PIP).
- **Network Slice Instance Layer:** virtualised resource pools, one per active slice instance, created by the Network Slice Provider (NSP) using NFV (Network Function Virtualization) and SDN (Software-Defined Networking).Each slice instance contains a set of virtualised network functions (VNFs) forming a Service Function Chain (SFC) from User Equipment (UE) to the application server.

- **Service Instance Layer:** the application facing interface exposing the slice to tenants through SLAs that specify quantitative performance guarantees maximum latency, minimum throughput, packet loss rate, and availability.

Liu et al. [7] emphasise two essential attributes that the NSP must provide to accomplish end-to-end slicing: performance isolation (the QoS of a slice must not be degraded by operations in co-existing slices) and SLA assurance (the performance requirements agreed with the tenant must be met). Meeting both simultaneously while minimising resource usage is the core resource management challenge.

The key design principles of a network slicing architecture are isolation, elasticity, and end-to-end optimisation [4].

Isolation ensures that each slice behaves as if it were the sole user of the infrastructure in the radio domain; this is enforced by the Physical Resource Block (PRB) allocator in the gNB MAC scheduler, which strictly limits each slice to its assigned PRB budget.

Elasticity means that the PRB budget is adjusted dynamically as traffic demand evolves, rather than being fixed at provisioning time. End-to-end optimisation requires that resources across all layers (RAN, transport, core, edge) be co-optimised jointly to meet QoS from the UE to the application server.

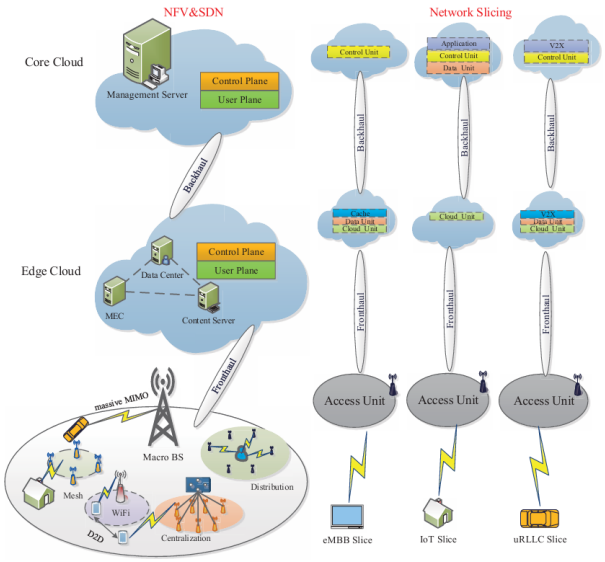

Figure 1.4: Network slicing based 5G system architecture [22].

1.3.2 The Three 5G Service Categories

The ITU-R IMT-2020 framework [8] defines three canonical service types for 5G networks, reflecting the three fundamentally different traffic profiles that must co-exist on the shared infrastructure as illustrated in Fig. 1.5. These categories are widely adopted in the literature [4][6][7] and directly correspond to the two slices instantiated in this thesis.

- **Enhanced Mobile Broadband (eMBB):** eMBB targets applications that require very high data rates and large sustained bandwidth: 4K/8K video streaming, augmented reality (AR), virtual reality (VR), and passenger internet access in vehicles. The ITU-R specifies a peak data rate of 20 Gbps downlink / 10 Gbps uplink and a user-experienced rate of 100 Mbps in dense urban deployments [8]. eMBB traffic is characterized by high and time-varying bandwidth demand (driven by the number of active streams), moderate latency tolerance (10–50 ms round-trip), and relatively loose reliability requirements compared to safety-critical applications.

- **Ultra-Reliable Low-Latency Communication (URLLC):** URLLC is characterized by strict end-to-end latency requirements (below 1 ms in target scenarios) combined with extremely high reliability, typically expressed as a packet loss probability no greater than 10^{-5} [6]. Applications in this category include autonomous vehicle communication, remote robotic surgery, smart-grid control, and industrial automation, where a single missed deadline or lost packet may have catastrophic consequences.

- **Massive Machine-Type Communication (mMTC):** This service class addresses the connectivity needs of a very large number of low-complexity, low-power devices up to one million devices per square kilometre with sporadic, small payload traffic patterns [6][4]. Prolonged battery lifetime and wide area coverage take precedence over throughput. Smart metering, environmental monitoring, and large-scale IoT deployments fall within this category.

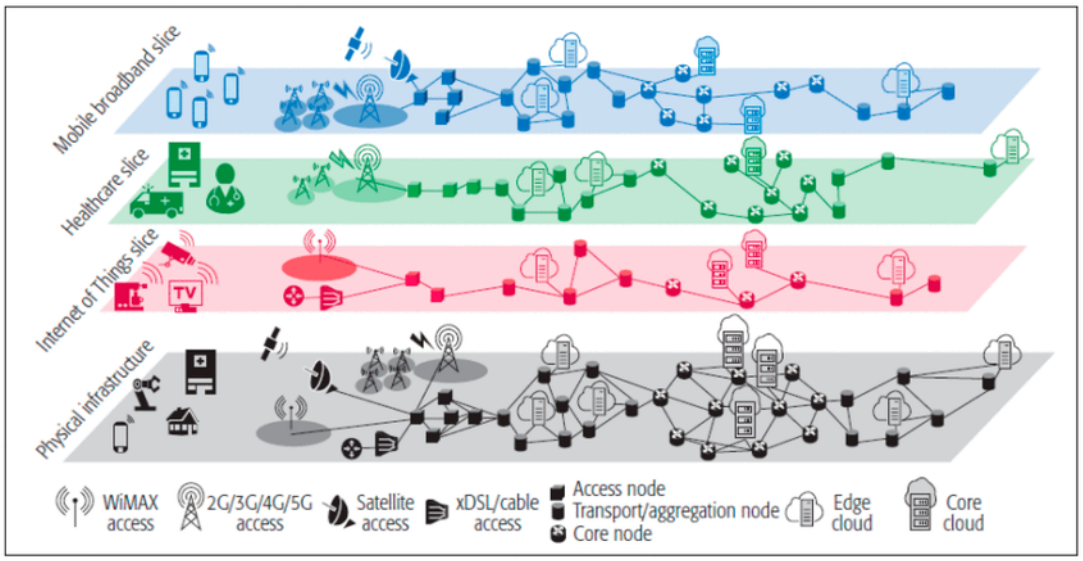

Figure 1.5: 5G network slices running on a common underlying multi-vendor and multi-access network. Each slice is independently managed and addresses a particular use case [23].

1.4 Resource Allocation in 5G

Resource allocation in 5G network slicing refers to the problem of dynamically assigning network resources, including physical resource blocks (PRBs) in the RAN, bandwidth, virtual CPU and memory in the core, and link capacity in the transport network to admitted slice requests, in a manner that satisfies each slice's requirements while maximizing overall network efficiency [10][6]. This constitutes one of the most computationally challenging sub-problems in network slice management.

1.4.1 Resource Management Phases in Network Slicing

Following the taxonomy of Hurtado Sánchez et al. [6], the complete resource management life cycle comprises four interrelated phases, as shown in Figure 1.6:

- **Admission Control:** Determines which incoming slice service requests are accepted into the network based on current capacity, network policy, and revenue objectives. Admission decisions have cascading effects on all subsequent phases.

- **Resource Allocation:** Quantifies and assigns specific amounts of computing, storage, radio, and networking resources to each admitted slice, to maximise long-term provider revenue and meeting tenant QoS/Quality of Experience (QoE) commitments [6].
- **Resource Scheduling:** Determines the temporal ordering in which allocated resources are consumed, seeking to minimize total execution time and guarantee service differentiation across slice types [6].
- **Resource Orchestration:** Manages the full lifecycle of Service Function Chains (SFCs), including Virtual Network Function (VNF) instantiation, dynamic scaling in response to load changes, and teardown upon service termination [6]. Phases can provide feedback to one another, requiring their coordinated operation.

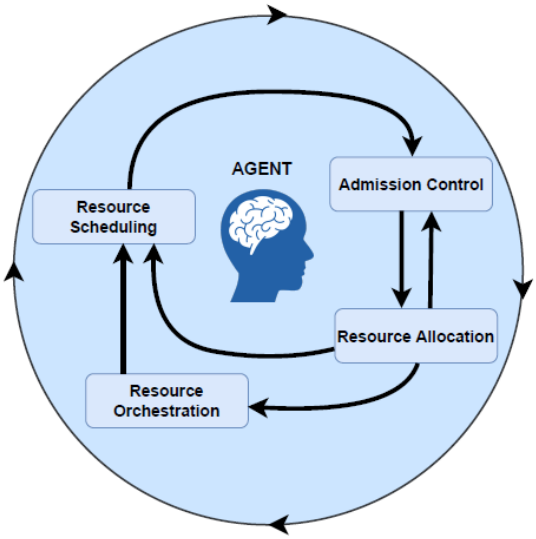

Figure 1.6: Resource management phases [6].

1.4.2 Key Constraints and Objectives

The resource allocation problem is subject to a set of hard and soft constraints that must be simultaneously respected [10][6]:

- QoS/QoE requirements: minimum throughput, maximum jitter, and packet loss bounds per slice.
- Latency bounds: especially stringent for URLLC slices, which may require sub-millisecond end-to-end delays.
- Bandwidth and PRB availability: the total number of radio resource blocks in the RAN is fixed; their allocation to competing slices must respect physical limits.
- SLA compliance: contractual guarantees made to tenants must be honoured; violations incur financial penalties [10][11].
- Inter-slice isolation: the performance of one slice must remain unaffected by the behaviour of co-located slices [7].

The joint optimization across all phases and constraints is known to be NP-hard in its general formulation. This computational intractability, combined with the need for real-time responsiveness to rapidly changing traffic conditions, motivates the use of intelligent and data-driven optimization approaches such as reinforcement learning and evolutionary algorithms [6][11].

1.5 Reinforcement Learning in Networking

Reinforcement Learning (RL) is a branch of machine learning concerned with how an autonomous agent learns to make sequential decisions in order to maximize a long-term objective [9]. Unlike supervised learning, RL does not require labeled training data; instead, the agent discovers effective behaviors through trial-and-error interaction with its environment.

The resource allocation problem has three properties that make classical optimization methods poorly suited: it is sequential (the allocation at time t influences the traffic build-up and constraint satisfaction over the following steps), non-stationary (the traffic regime

switches stochastically, and burst events occur randomly), and model-uncertain (the exact relationship between bandwidth allocation, queuing dynamics, and latency depends on channel conditions that cannot be predicted exactly). RL is specifically designed for this class of problems: it learns an allocation policy from repeated interactions with the environment, without requiring an explicit mathematical model of the system dynamics [9].

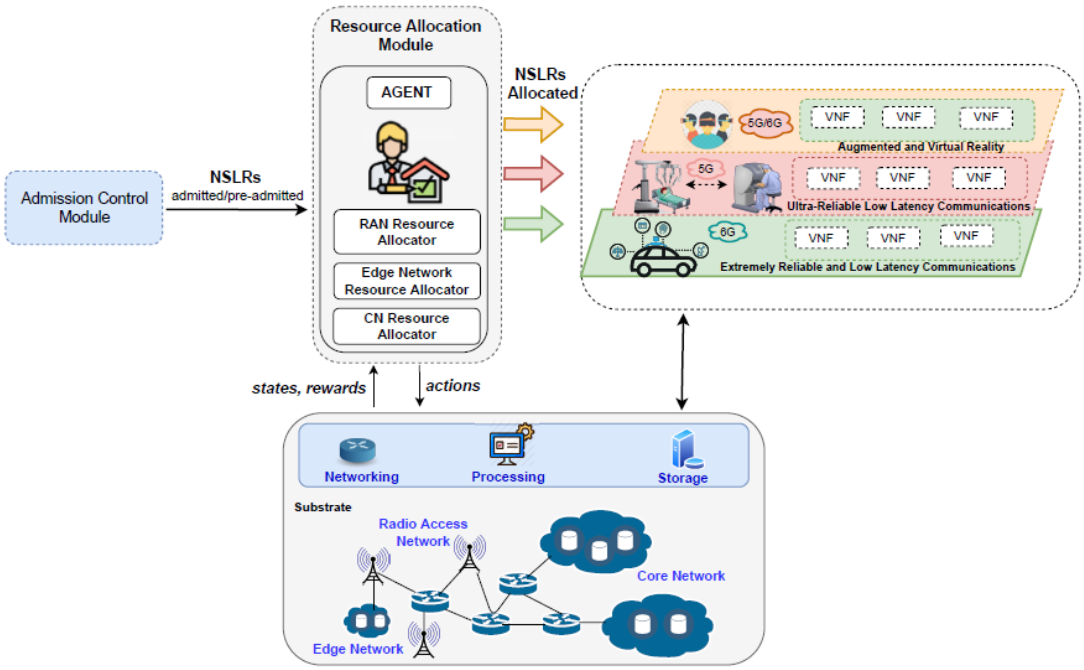

Figure 1.7: Resource allocation architecture using RL/DRL [6].

Hurtado Sánchez et al. [6] identify RL and Deep Reinforcement Learning (DRL) as the most promising family of methods for network slicing resource management,as illustrated in Figure 1.7, noting that they allow the evolution from model-based techniques (which fail when the model is inaccurate) to model-free techniques that learn deeply by interacting with the environment to satisfy both QoS requirements and quality of experience requirements (expressed as reward signals). The survey reviews over 50 papers published between 2018 and 2022 finds DQN based approaches to be the dominant method for radio resource allocation, with DQN or its variants (Double Deep Q-Network (DDQN), Dueling

DQN) appearing in more than 70% of reviewed works on the resource allocation phase.

1.5.1 MDP Basics

The mathematical framework underpinning RL is the Markov Decision Process (MDP), defined by the tuple $M = (S, A, P, R, \gamma)$, where:

- **S** is the state space, representing all possible observations of the environment (e.g., current resource utilization, active users per slice, channel quality indicators);
- **A** is the action space, comprising all decisions the agent can take at each time step (e.g., the amount of bandwidth to assign to each slice);
- $P(s' \mid s, a)$ is the transition probability function, describing the stochastic dynamics of the environment;
- $R(s, a, s')$ is the reward function, providing a scalar signal that encodes the desirability of transitions (e.g., positive reward for satisfying an SLA, negative reward for a violation);
- $\gamma \in [0, 1)$ is the discount factor, controlling the agent's preference for immediate versus future rewards [9].

The agent's goal is to find an optimal policy π^* that maximizes the expected cumulative discounted return:

$$G_t = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \right]$$

This general formulation maps naturally to the 5G resource allocation problem, where the state encodes the network's current operational status, actions correspond to resource assignment decisions, and the reward penalizes SLA violations while rewarding efficient utilization [6][9].

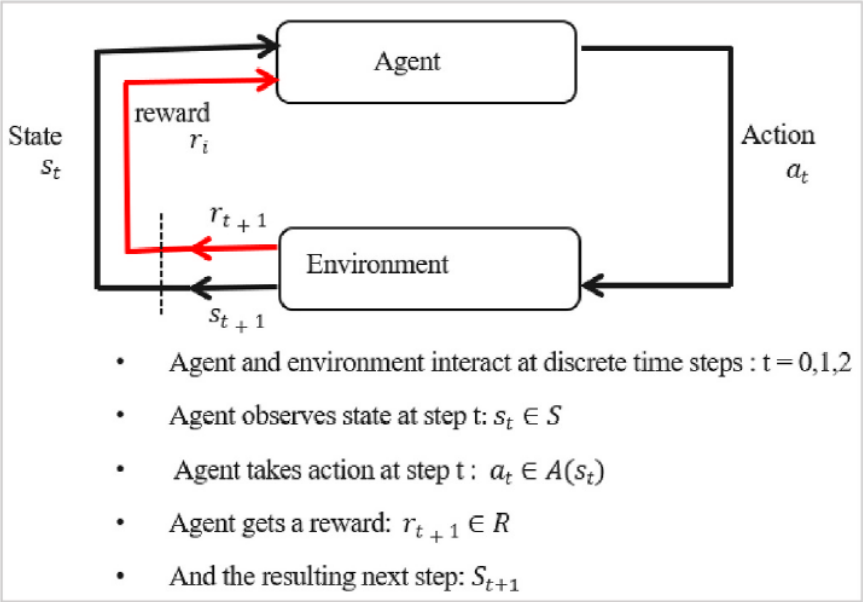

Figure 1.8: Reinforcement learning process [24].

1.5.2 Deep Reinforcement Learning

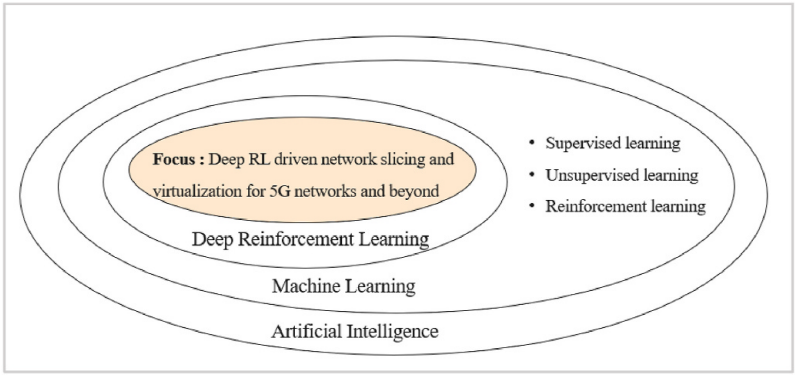

Figure 1.9: Relationship among deep RL, DL, RL, SL, USL, ML and AI [24].

Classical tabular RL algorithms, such as Q-learning and SARSA, become computationally intractable when dealing with large or continuous state and action spaces. This situation is unavoidable in realistic 5G deployments involving multiple slices, users, and resource dimensions [6]. DRL overcomes this limitation by combining RL with Deep Neural Networks (DNNs) as universal function approximators, as illustrated in Figure 1.9, enabling

the agent to generalize across states it has never explicitly encountered [12][9].

Deep Q-Network (DQN): Introduced by Mnih et al. at DeepMind [12], DQN was the first algorithm to demonstrate that a single RL agent could master a diverse range of complex tasks (49 Atari 2600 games) at a human competitive level using only raw pixel inputs. DQN approximates the optimal action value function $Q^*(s, a)$ with a deep neural network parameterized by θ. Two key design innovations stabilize the otherwise notoriously unstable combination of Q-learning with non-linear function approximation:

- Experience Replay: Past transitions (s_t, a_t, r_t, s_{t+1}) are stored in a replay buffer D of fixed capacity N. At each training step, a random mini-batch of B transitions is sampled uniformly from D. This breaks temporal correlations between consecutive training samples, which would otherwise cause the neural network to catastrophically overfit to recent experience, and ensures approximate independence within the mini-batch. Lin [13] first proposed experience replay, and it was a key component of Mnih et al.'s [12] DQN system.

- Target Network: A separate target network $Q(s, a; \theta^-)$ with frozen parameters θ^- is used to compute the Bellman target $y_t = r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \theta^-)$. The target network is synchronised with the online network $\theta^- \leftarrow \theta$ every C steps rather than every gradient step. This prevents the feedback loop in which the network being trained is simultaneously used to generate the training targets, a situation that causes oscillation and divergence in practice.

The loss minimized at each training step is:

$$L(\theta) = \mathbb{E}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta)\right)^2\right]$$

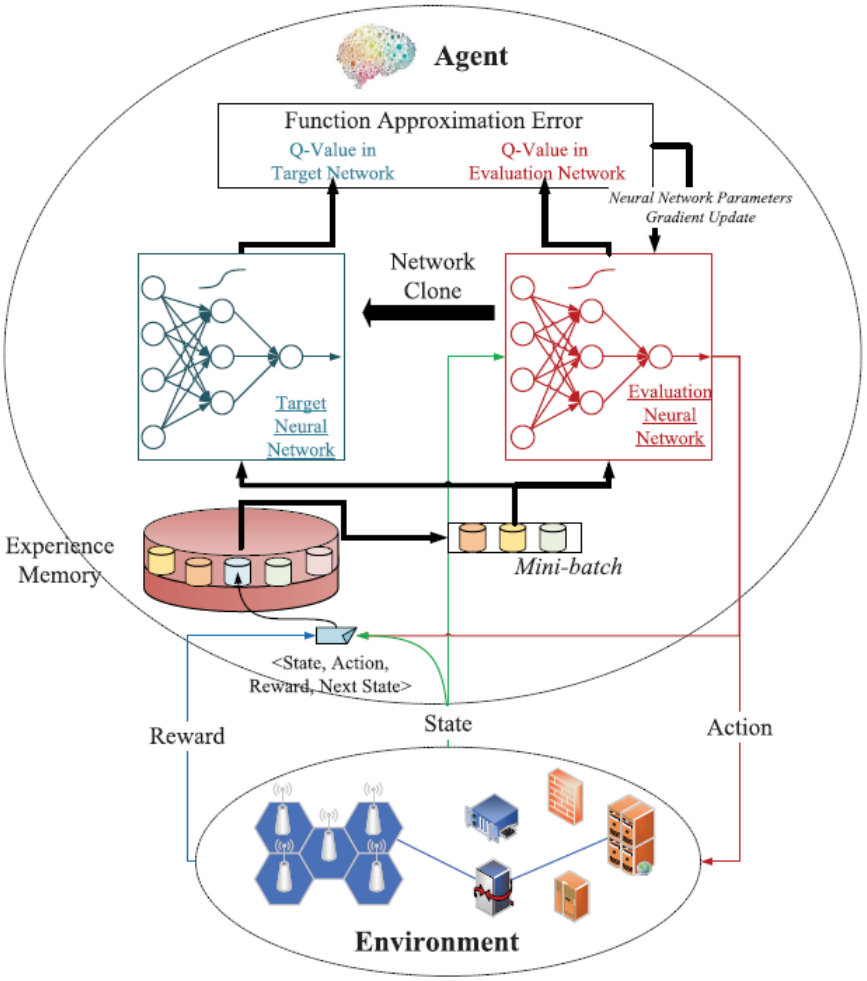

Figure 1.10: An illustration of deep Q-learning [16].

Variants and Extensions: Building on the standard interaction loop between the agent and the environment illustrated in Figure 1.10, Several improvements to DQN have been proposed and are relevant to network resource management. Double DQN (DDQN) decouples action selection from value estimation to mitigate the overestimation bias of standard DQN [6]. Dueling DQN decomposes the Q-function into a state-value function and an advantage function, improving learning stability in states where most actions have similar values [6]. Proximal Policy Optimization (Proximal Policy Optimisation (PPO)), an on-policy actor-critic algorithm, constrains successive policy updates via a clipped surrogate objective to prevent destabilizing large gradient steps, and is particularly well-

suited to continuous action spaces [6][9].

1.6 Metaheuristics for Optimization

Metaheuristic algorithms are stochastic search strategies designed to explore the solution space of complex optimization problems without requiring gradient information about the objective function. These algorithms are typically classified into two main categories: single-solution-based and population-based methods. While single-solution approaches evolve a single candidate solution, population-based metaheuristics maintain and improve a set of diverse solutions simultaneously. They are particularly suitable for optimization problems that are non-convex, non-smooth, or combinatorial in nature.

In this context, the Genetic Algorithm (GA) represents a widely used population-based stochastic metaheuristic.

1.6.1 Genetic Algorithm

Originally formalized by Holland (1975) and extensively developed by Goldberg [14], GAs are the most widely studied family of metaheuristics for engineering optimization problems. GAs maintain a population of candidate solutions (chromosomes) and iteratively improve them through operators inspired by the biological principles of Darwinian natural selection and genetic inheritance: selection (survival of the fittest), crossover (recombination of parent genes), and mutation (random gene perturbation). The Building Block Hypothesis [14] states that GAs efficiently combines short, well-adapted sub-solutions across generations, converging toward the global optimum without requiring an explicit model of the objective function.

The algorithm evolves this population across successive generations through three fundamental operators [14],as illustrated in Figure 1.11:

- **Selection:** Individuals are probabilistically chosen for reproduction based on their fitness a scalar measure of solution quality computed from the objective function

(e.g., total network throughput subject to QoS constraints).Common selection mechanisms include roulette-wheel selection and tournament selection, both of which create evolutionary pressure toward higher-quality solutions while preserving diversity.

- **Crossover (Recombination):** Two selected parent chromosomes exchange one or more contiguous segments at randomly chosen crossover points, producing offspring that inherit characteristics from both parents. This operator explores new regions of the search space by recombining already promising partial solutions, embodying the building block hypothesis central to GA theory [14].

- **Mutation:** Individual genes within offspring chromosomes are randomly altered with a typically small probability, introducing novel configurations not present in either parent. Mutation is essential for escaping local optima and maintaining population diversity across generations [14].

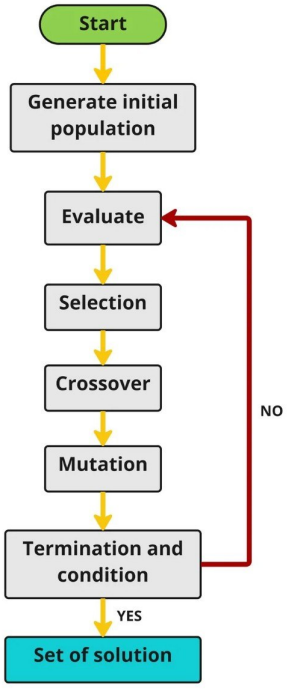

Figure 1.11: Procedural flowchart of the Genetic Algorithm highlighting the iterative evolutionary cycle of selection, crossover, and mutation [26].

In the context of wireless network resource management, Li and Zhu [11] demonstrated the effectiveness of a GA-based approach for jointly optimizing task-offloading ratios, channel bandwidth allocation, and edge-server compute assignments in a mobile edge computing (MEC) scenario. Their results showed that the GA consistently finds near-optimal configurations that minimize user task completion times, outperforming greedy and exhaustive search baselines on multi-user scenarios. Chi et al. [15] similarly employed DRL-based virtual network embedding for smart-grid resource allocation, reinforcing the trend of applying intelligent optimization to network virtualization problems where heuristics alone are insufficient.

GAs have been also applied to radio block assignment, slice dimensioning, and VNF placement. Hurtado Sánchez et al.[6] cite genetic algorithms among the heuristic methods used for network slicing resource scheduling, confirming their applicability to the 5G resource management domain.

1.7 Related Work

1.7.1 DRL for Network Slice Resource Allocation

The application of DRL to 5G network slice resource allocation has developed rapidly since 2018, and the landscape of existing approaches is comprehensively surveyed by Hurtado Sánchez et al. [6]. Their analysis of 50 peer-reviewed works identifies DQN and its variants as the dominant approach for the radio access network (RAN) resource allocation phase, with most works using discrete action spaces and synthetic traffic datasets.

Li et al. [16] proposed one of the first DRL-based resource management frameworks for network slicing, using a DQN agent with experience replay and a target network to allocate bandwidth to multiple slice types in a virtualized RAN. Their results demonstrated that DQN improves resource utilization by approximately 15% relative to Q-learning baselines and shows stable convergence after 2,000–3,000 training steps. This work established the foundational DQN approach that subsequent works—including this thesis build upon.

Hurtado Sánchez et al. [6] cite this work as one of the early demonstrations of DRL viability for network slicing resource management.

Van Huynh et al. [17] extended the DQN framework to Duelling Double DQN (D3QN) for real-time resource slicing in manufacturing, utility, and automotive slices. Their system handles heterogeneous resource requirements (computing, storage, radio) and time-varying slice demand, using prioritized experience replay to ensure that high-error transitions corresponding to rare but important traffic events are replayed more frequently. Their evaluation shows that D3QN achieves 8–12% higher long-term profit than vanilla DQN under the same traffic conditions. The Duelling decomposition $Q(s, a) = V(s) + A(s, a)$ proves particularly beneficial in their setting because many actions share similar QoS satisfaction levels under moderate load, and the state-value component $V(s)$ captures the load-dependent baseline without interference from action-specific advantages.

Liu et al. [7] (OnSlicing), another of the four uploaded references, represent the current state of the art in end-to-end DRL-based network slicing. OnSlicing uses a constraint-aware PPO agent with offline imitation learning (behavior cloning from a rule-based baseline) and a proactive baseline switching mechanism to maintain near-zero SLA violations (0.06%) throughout online learning on a real 4G/5G testbed. Compared to the rule-based baseline, OnSlicing reduces resource usage by 61.3% while maintaining zero average SLA violations, confirming that DRL-based dynamic allocation substantially outperforms static allocation in practice.

Filali et al. [18] proposed a dynamic SDN based RAN slicing framework specifically for URLLC and eMBB services using multi-agent DQN, where each RAN slice has a dedicated DQN agent and an SDN controller coordinates inter-slice resource negotiation.Their setting simultaneous URLLC latency and eMBB throughput constraints in a shared 5G RAN—is the closest existing work to the scenario of this thesis. Filali et al.'s results show that multi-agent DQN achieves 15–20% higher QoS satisfaction than single-agent DQN

under heterogeneous traffic, at the cost of inter-agent communication overhead.Their work confirms that URLLC latency satisfaction requires dedicated optimisation separate from eMBB throughput maximisation.

1.7.2 Metaheuristics and GA for 5G Resource Management

Genetic algorithms and other metaheuristics have been applied to 5G network resource management primarily as offline optimization tools for slice dimensioning, frequency planning, and VNF placement. Hurtado Sánchez et al. [6] cite genetic algorithms among the scheduling methods used for network slicing resource management, noting that evolutionary approaches offer the advantage of applicability to non-convex, NP-hard combinatorial problems that arise in VNF chaining and multi-domain orchestration.

Li and Zhu [11] proposed a GA-based joint optimisation of offloading proportions and resource allocation in a MEC network, demonstrating that the GA effectively navigates the high-dimensional, mixed integer solution space and produces Pareto efficient configurations that minimize user task completion time while ensuring fairness across users. The main limitation of this approach is its computational overhead: GA's iterative population evaluation is too slow for the sub-second reaction times required in live 5G networks.

Afolabi et al. [20] applied evolutionary computation to dynamic VNF placement and slice orchestration, demonstrating that evolutionary search can adapt slice configurations within 300 ms of a traffic change fast enough for real-time slice management. Their work confirms that evolutionary algorithms are computationally feasible for online use in 5G systems when the problem dimensionality is low (2–5 genes).

1.7.3 Hybrid DRL and Evolutionary Algorithm Approaches

The combination of deep reinforcement learning with evolutionary or swarm intelligence algorithms is an emerging research direction motivated by the complementary strengths of the two paradigms: DRL provides global temporal learning across multiple steps, while

evolutionary algorithms provide per-step continuous search without gradient requirements. Hurtado Sánchez et al. [6] identify this hybridization as an open research direction in Section 7 of their survey, noting that federated DRL and multi-agent DRL are promising extensions but that hybrid DRL+evolutionary approaches remain largely unexplored in the network slicing literature.

In the Mobile Edge Computing (MEC) domain, Arun and Azhagiri [25] recently proposed the HDQ-GO framework a Hybrid Deep Q-Learning and Genetic Optimization system for task offloading and resource management in MEC environments. HDQ-GO integrates a Deep Q-Network (DQN) phase for dynamic resource allocation with a Genetic Optimization phase that refines DQN proposed strategies through crossover and mutation. A key distinguishing feature is the incorporation of Hindsight Experience Replay (HER) into the DQL component, which allows the agent to extract learning value from sub-optimal or failed experiences by reframing them as valid goals a technique rarely applied in edge computing contexts. The GA phase employs roulette wheel selection and energy-aware crossover/mutation operators specifically designed to minimize node-level power consumption under varying task loads.

Experimental results reported by Arun and Azhagiri [25] demonstrate that HDQ-GO achieves a convergence utility of 1.00 at 250 iterations (versus 0.97 for standalone DRL and 0.90 for multi-agent RL), a computation latency of 0.068 s at 0.2 Mbits input (18% lower than DRL), and an energy consumption of 0.038 J at 0.2 Mbits (14% lower than DRL). Network bandwidth utilization reaches 0.92 at 100 Mbps (15% higher than multi-agent RL), and CPU utilization reaches 0.94 at 30% CPU availability. These results confirm the general principle applicable to both MEC task offloading and 5G network slicing bandwidth allocation that a tightly coupled DQN+GA architecture consistently outperforms standalone DRL across latency, energy, and resource utilization dimensions.

While HDQ-GO and the proposed DQN+GA hybrid of this thesis share the same structural paradigm (DRL proposes, GA refines), they differ in three important respects. First, HDQ-GO targets task offloading in MEC environments with varying CPU and band-

width loads. In contrast, this thesis targets per-second allocation of bandwidth fractions between URLLC and eMBB slices in a 5G V2X highway scenario. Second, HDQ-GO applies HER to improve DQL convergence across training episodes, whereas this thesis uses a DQN seeded warm-start to enable the GA to reach high-quality solutions within 4–6 generations per decision epoch. Third, HDQ-GO uses a fixed GA budget per epoch, whereas this thesis applies Adaptive Budget Control that scales GA computation to the urgency of the current QoS state.

A limited number of works have begun exploring combinations of learning-based and evolutionary methods. The OnSlicing system [7] uses a rule-based policy to initialize (warm-start) the DRL agent, accelerating convergence and reducing early phase SLA violations. However, this is a one-directional warm-start rather than a true integration of evolutionary search. The DRL survey [6] explicitly flags hybrid RL metaheuristic methods as a promising future direction, noting that GA-style global search could complement DRL's local, gradient-based policy improvement.

To our knowledge, no published work has yet proposed a tightly integrated GA–DRL framework for dynamic, multi-slice resource allocation in 5G networks that operates across both RAN and core network domains simultaneously.

Table 1.1: Comparative summary of related works on 5G and MEC resource allocation.

Work	**Method**	**Scenario**	**Slices / Tasks**	**Key Metric**	**Main Limitation**	**Our Advantage**
Li et al. [16] 2018	DQN	Virtualised RAN	3 slices	Reward, utilisation	Discrete action ceiling	GA removes discrete ceiling
Van Huynh et al. [17] 2019	Dueling Double Deep Q-Network (D3QN) + PER	Manufacturing slices	3 slices	Long-term profit	No V2X latency	V2X latency + GA refinement
Yuan et al. [21] 2021	DDQN (CND-DQN)	Cognitive radio	2 (eMBB + URLLC)	SE + QoE	Cognitive radio, not V2X	V2X highway + adaptive GA
Liu et al. [7] 2021	PPO online (OnSlicing)	E2E 4G/5G testbed	3 slices	Usage, SLA viol.	10-dim action, real testbed	2D action + GA refiner
Filali et al. [18] 2022	Multi-agent DQN	5G RAN (URLLC + eMBB)	2 slices	QoS satisfaction	Multi-agent overhead; no GA	Single-agent + GA; V2X V2V
Afolabi et al. [20] 2020	Evolutionary	E2E orchestration	n slices	Adaptation speed	Cold-start per epoch; no temporal memory	DQN warm-start removes cold-start
Arun & Azhagiri [25] 2025	DQL + GA (HDQ-GO)	MEC task offloading	Multiple tasks	Latency, energy, utilization	MEC domain; fixed GA budget	V2X slicing + adaptive budget + step_mu

1.8 Problem Statement

1.8.1 Limitations of existing approaches

The review of the literature reveals three concrete, structurally distinct limitations that existing approaches fail to resolve simultaneously in the V2X URLLC/eMBB slice allocation context:

- **The Discrete Action Ceiling of DQN:** DQN-based approaches (Li et al. [16], Van Huynh et al. [17], Filali et al. [18], Yuan et al. [21]) represent the bandwidth allocation as a discrete selection from a finite predefined set. This architectural choice imposes a "discrete action ceiling" that prevents the model from adapting to traffic demands falling outside its fixed range. For instance, if the predefined set is capped at a maximum eMBB fraction of 65 MHz, the agent is structurally incapable of satisfying requirements during peak traffic regimes where demand (e.g., 68.45 MHz) exceeds this limit. Because the agent cannot generate an action beyond its highest available index, it faces a hard performance boundary. Consequently, regardless of training duration or hyperparameter tuning, the agent will inevitably fail to meet constraints during high-load epochs, as the necessary bandwidth value simply does not exist within its action repertoire.

- **Cold-Start Inefficiency in Genetic Algorithms:** GA approaches (Li and Zhu [11], Afolabi et al. [20]) lack temporal memory, treating each decision epoch as an isolated event. Because the search process typically begins from a near-random population or fixed seeds at every interval, the system fails to leverage historical data regarding which allocations performed well under specific traffic regimes. Within the strict computational constraints of real-time environments where a limited budget of individuals and generations is required to meet short decision windows, a "cold-start" GA spends a significant portion of its processing power rediscovering fundamental relationships rather than refining optimal solutions. This lack of continuity prevents the algorithm from internalizing long-term patterns, often resulting in lower QoS

satisfaction compared to learning-based models that maintain an internal state or memory of past experiences.

- **Absence of Per-Step Continuous Refinement in Published DRL Systems:** The DRL systems reviewed (Li et al. [16], Van Huynh et al. [17], Liu et al. [7], Filali et al. [18]) do not incorporate a per-step meta-heuristic refinement that (a) operates in continuous allocation space beyond the DRL's discrete action ceiling, (b) uses the DRL's learned policy as an informed warm-start, and (c) applies its refined allocation to the scheduler at every step. Liu et al. [7] (OnSlicing) is the closest in spirit to their constraint-aware PPO and proactive baseline switching prevent SLA violations, but their system operates in a 10-dimensional continuous action space, targets a multi-domain E2E slicing problem, and requires an offline imitation learning phase on a real testbed.

1.8.2 Research Objectives

The central challenge addressed in this thesis is the real-time dynamic allocation of radio bandwidth between co-existing URLLC and eMBB network slices in a 5G V2X highway environment, under unpredictable, non-stationary traffic conditions. The three limitations identified in the preceding review do not exist in isolation; they form a coherent pattern of failure. DQN is fast but blind beyond its action ceiling. GAs are expressive but amnesiac. And no published DRL system closes the loop by applying a per-step continuous refinement back to the scheduler at every decision instant. Taken together, these gaps point toward a single, natural conclusion: no existing approach simultaneously provides the speed of a learned policy, the continuity of a meta-heuristic search, and the memory of a temporal learning agent. This convergence of limitations motivated the central idea of this work, a hybrid architecture that does not merely combine DQN and GA, but engineers them to compensate for each other's structural weaknesses in a tightly coupled, real-time feedback loop.

From this observation, the present work pursues the following objectives:

- Propose a hybrid intelligent system that combines the temporal learning capability of a DQN with the continuous search capability of a GA, so that the two components complement each other: the DQN guides the search with a learned policy, and the GA refines the outcome beyond the boundaries that a discrete action space alone cannot reach.

- Design a GA refiner that accepts the output of the DQN as a warm-start seed and performs a continuous search beyond the boundaries that a discrete action space alone cannot reach. This addresses a fundamental structural limitation of DQN based approaches: because the set of predefined allocation actions is finite, the agent is architecturally incapable of satisfying bandwidth demands that fall between or above those discrete options, regardless of how well it has been trained. The GA refiner resolves this by treating the DQN's best action as a starting point and exploring the surrounding continuous space, producing a refined allocation that is applied directly to the scheduler at each decision step.

- Eliminate the cold-start inefficiency that affects standalone genetic algorithms when they are used in a real-time decision context. Because a GA that begins from a random or weakly informed population must spend most of its limited budget re-discovering basic traffic allocation relationships, it cannot compete with a trained agent under tight time constraints. This work addresses the problem by designing a multi-directional warm-start strategy that seeds the GA population from multiple strategically chosen candidates, including the DQN's own learned proposal, so that the search begins near high-quality solutions and uses its available generations for genuine refinement rather than exploration.

- Ensure that the GA refinement operates with an adaptive computational cost that is proportional to the urgency of the situation. When the DQN's allocation already satisfies all constraints, no refinement is needed, and the GA is bypassed entirely, keeping the system as efficient as pure DQN. When one or both constraints are

violated, the GA is activated with a budget scaled to the severity of the violation. This design ensures that the hybrid system adds meaningful value precisely where it is most needed, without imposing a fixed overhead at every decision step.

- Establish a co-adaptation loop between the DQN and the GA, in which the DQN is trained on the reward produced by the GA's refined allocation rather than on its own discrete output. This ensures that the DQN progressively learns to produce proposals that are better starting points for the GA, so that the two components reinforce each other over the course of training rather than operating independently.

- Implement and evaluate this hybrid system in a simulated 5G V2X highway environment with co-existing URLLC and eMBB slices under dynamic and unpredictable traffic conditions, and to compare its performance against relevant baselines in order to measure the concrete gain brought by the hybridization.

1.9 Conclusion

In this chapter, we have conducted an extensive review of the 5G architecture and the state-of-the-art techniques used for dynamic resource allocation. Our analysis of the current literature reveals a clear trade-off: RL (specifically DQN) offers rapid, real-time decision-making but is often constrained by discrete action spaces that limit precision. Conversely, metaheuristics like GAs offer global search capabilities and high precision but are often too computationally expensive for the millisecond-level requirements of 5G.

The identification of these limitations provides the primary motivation for our research. A standalone approach is insufficient for the complex, non-stationary environment of a 5G V2X scenario. This conclusion leads us to the necessity of a hybrid framework that leverages the strengths of both paradigms DQN for speed and GA for refinement. In the next chapter, we will detail the design and implementation of this proposed hybrid architecture.

Chapter 2

Proposed Approach

2.1 Introduction

Building upon the theoretical foundations and the gaps identified in the previous chapter, this chapter details how we applied a Hybrid Deep Reinforcement Learning and Genetic Algorithm (DQN-GA) approach to address the resource allocation problem. The primary objective of this approach is to optimise the allocation of radio resources in a Fifth Generation of Mobile Networks (5G) network slicing environment, specifically focusing on a V2X (Vehicle-to-Everything) highway scenario. This scenario is characterised by high mobility and a mixture of eMBB and URLLC traffic, making it a perfect testbed for high-performance allocation strategies.

In this chapter, we formally define the system model, including the channel model, the traffic generation process, and the specific constraints of the 5G New Radio (NR) interface. We then present the core of our contribution: the hybrid architecture. We detail how a Deep Q-Network (DQN) is utilised to provide an immediate resource distribution and how a Genetic Algorithm (GA) is integrated as a ”refiner” to fine-tune these decisions. We also describe the ”warm-start” mechanism and the adaptive computational budget control, which are designed to ensure that the hybrid system remains efficient and responsive under varying network loads.

2.2 System Model

2.2.1 Network Scenario

The physical deployment considered in this work is a 5 km bidirectional highway equipped with a single 5G New Radio (NR) base station operating in the FR1 frequency range. The gNodeB (gNB) uses a 100 MHz carrier centred at 3.5 GHz (band n78, TDD mode), which at the 30 kHz subcarrier spacing defined in 3GPP TS 38.211 [27] yields a pool of 273 Physical Resource Blocks (PRB), each spanning twelve subcarriers (180 kHz) over one 0.5 ms TTI. The gNB is connected via a low-latency fiber link (Enhanced Common Public Radio Interface (eCPRI) over fiber, round-trip delay $<$ 0.1 ms) to a Mobile Edge Computing (MEC) server co-located at the tower site. The MEC server hosts the entire decision engine DQN agent plus GA refiner and communicates with the gNB over the standardized ETSI MEC API [3] using the N2/N3 control and user-plane interfaces defined in 3GPP TS 23.501 [1].

Two Road Side Units (RSUs) are deployed at kilometer posts 0.5 and 1.5 along the highway. Each RSU implements the Cellular Vehicle-to-Everything (C-V2X) PC5 sidelink interface (3GPP Release 16) at 5.9 GHz ITS band, providing direct V2I links with a coverage radius of approximately 400 m and an over-the-air latency well below 1 ms [28]. RSUs relay safety-critical V2X messages to the gNB over dedicated Next Generation User Plane (NG-U) fiber backhaul segments.

Two surveillance camera poles are mounted on either side of the highway. These devices use the NR-Light Reduced Capability (RedCap) interface introduced in 3GPP Release 17 [5], operating in the same n78 carrier as the vehicular users, with uplink throughput in the 20–50 Mbps range. RedCap devices are assigned to the eMBB slice and share its PRB budget with passenger internet users.

The 5G Core network (5GC) comprises the standard network functions defined in 3GPP TS 23.501 [1]: the Access and Mobility Management Function (AMF), Session Management Function (SMF), User Plane Function (UPF), Network Slice Selection Function (NSSF), and Policy Control Function (PCF). The PCF propagates per-slice QoS

profiles (guaranteed bit rate, 5G QoS Identifier (5QI), maximum packet loss rate) to the SMF, which in turn configures the gNB scheduler via the N2 interface. The UPF routes eMBB user plane traffic to the public Internet via the N6 interface as illustrated in figure 2.1.

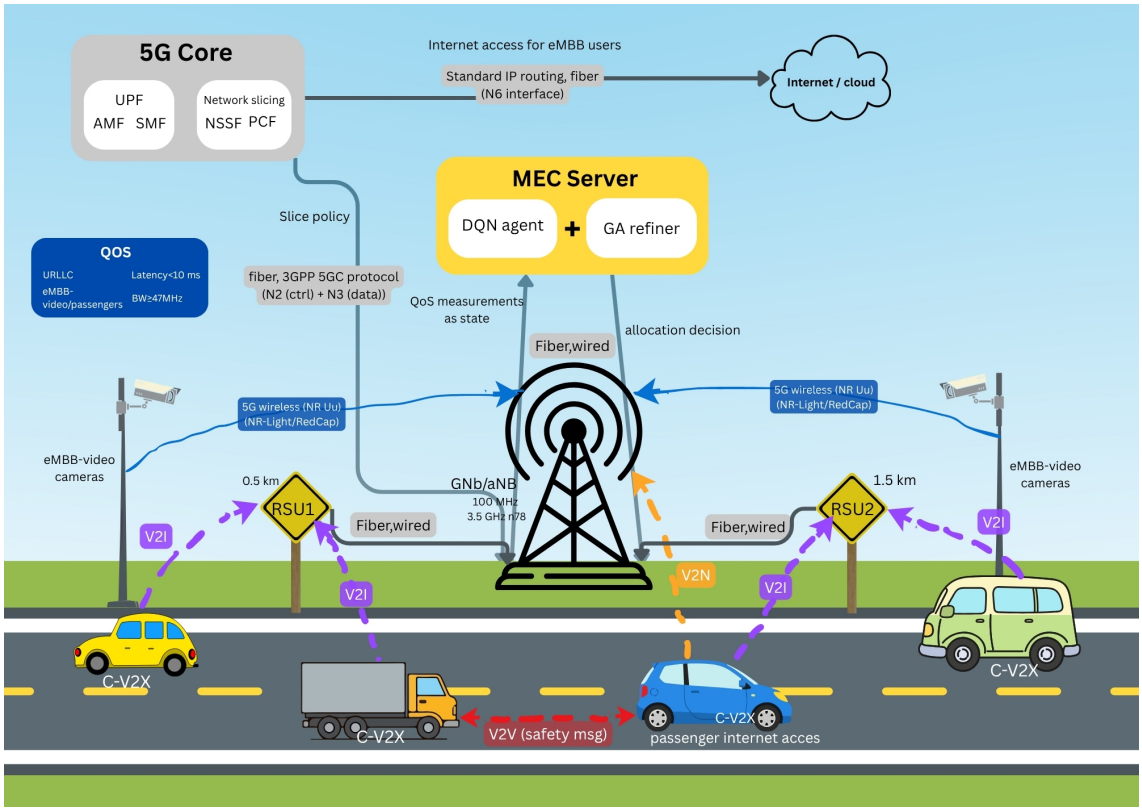

Figure 2.1: Proposed hybrid DQN+GA architecture for 5G dynamic resource allocation in the V2X highway scenario. The MEC server hosts the two-stage decision engine; the gNB MAC scheduler enforces the resulting PRB allocation every 1 s decision epoch.

2.2.2 Users and Vehicles

Four vehicle types are modeled on the highway:

- **URLLC vehicles (trucks, $n_U = 2$).** Heavy-goods vehicles traveling at speeds up to 90 km/h. They generate V2N safety messages (cooperative awareness messages, collision warnings) via the 5G NR Uu interface at 3.5 GHz. Each truck requires end-to-end latency strictly below 10 ms, consistent with the URLLC target defined in the

International Telecommunication Union Radiocommunication Sector (ITU-R) IMT-2020 framework [8]. The per-vehicle URLLC traffic load $\lambda_U(t)$ follows a piece-wise stochastic process that transitions between five traffic regimes (Normal, Rush-hour, Night, Accident, Peak-eMBB).

- **eMBB passenger vehicles (cars, $n_E = 2$).** Passenger cars traveling at up to 130 km/h. Occupants stream high-definition video (4K) and access Internet services over the eMBB slice. The bandwidth demand $D_E(t)$ varies between 47 MHz during normal operation and up to 85 MHz during peak-eMBB events. Video surveillance cameras at the RSU poles generate an additional uplink eMBB load.
- **V2V direct links.** Cars and trucks also exchange safety messages over the C-V2X PC5 sidelink at 5.9 GHz, bypassing the gNB. These Vehicle-to-Vehicle (V2V) links contribute no RAN load but improve cooperative collision avoidance [28].

2.2.3 Traffic Regimes

Following the methodology of Filali et al. [18], the traffic generation process is modeled as a discrete state Markov chain over five stationary regimes that capture realistic highway traffic conditions:

- **Normal:** This serves as the baseline condition, characterized by moderate and relatively balanced demand across both slices, representing standard daily traffic flow.
- **Rush-hour:** This regime represents peak traffic density. The network infrastructure experiences high simultaneous stress, as both latency sensitive and bandwidth intensive applications reach their maximum demand levels.
- **Night:** Reflects low-density periods where traffic is sparse. The demand for both service categories is minimal, resulting in significant available idle resources.
- **Accident:** An emergency driven scenario where URLLC enabled vehicles generate intensive bursts of safety-critical messages. In this regime, the priority demand for

the URLLC slice increases sharply, while the demand for broadband services remains at a standard level.

- **Peak-eMBB:** This scenario occurs during events that drive high data consumption, such as live-video streaming. The eMBB slice becomes saturated due to heavy throughput requirements, while the volume of critical safety messages remains relatively low.

The transition matrix between regimes and the per-step load sampling process are kept fixed across all experiments to ensure reproducibility, following the seeding protocol recommended by Gymnasium [29].

2.2.4 Network Slices

The single gNB simultaneously supports two network slice instances, instantiated in accordance with the 3GPP three layer slicing architecture described in [4]:

- **URLLC slice** ($\mathcal{S}_U$). Serves all C-V2X safety-critical users (trucks, RSU relay traffic). The slice Service Level Agreement (SLA) [1] specifies an end-to-end latency target $L^{\max} = 10\,\text{ms}$ and a packet-loss probability no greater than 10^{-5}, in line with the ITU-R URLLC requirements [8]. The intra-slice scheduler uses a round-robin policy that prioritizes latency over throughput.

- **eMBB slice** ($\mathcal{S}_E$). Serves passenger internet users and surveillance cameras. The SLA specifies a minimum aggregate downlink bandwidth $B_E^{\min}(t)$, which is load-dependent:

$$B_E^{\min}(t) = 47 \cdot (0.55 + \lambda_E(t)) \quad [\text{MHz}]. \tag{2.1}$$

  The intra-slice scheduler uses proportional-fair scheduling to maximize spectral efficiency subject to the bandwidth guarantee [6].

The total radio pool of 100 PRBs (100 MHz) is partitioned between the two slices at every 1-second decision epoch. The partition is expressed as the bandwidth fraction vector

$\boldsymbol{\mu}(t) = [\mu_U(t), \mu_E(t)]$ with $\mu_U(t) + \mu_E(t) = 1$ and $\mu_i(t) \geq 0$. Slice isolation is enforced at the gNB MAC layer: no slice may consume PRBs allocated to another, as required by the performance isolation principle [7].

Latency model for URLLC: The end-to-end latency experienced by a URLLC packet is modeled as a function of the utilization of the URLLC slice:

$$L_{\text{URLLC}} = 2.8 \left(\frac{\lambda_U}{\mu_U} \right)^{1.4} \quad [\text{ms}], \tag{2.2}$$

where λ_U is the normalized URLLC traffic load and μ_U is the fraction of bandwidth allocated to the URLLC slice. This convex increasing function reflects the exponential growth of the queuing delay as the allocated bandwidth approaches the offered load. The constant 2.8 calibrates the latency to the 10 ms SLA limit under typical conditions.

2.3 Problem Formulation

We formulate the resource allocation as a sequential decision making problem under uncertainty. At each discrete time step t, the agent observes the current traffic state $\mathbf{s}_t$ and must choose a bandwidth split $\boldsymbol{\mu} = [\mu_U,\ \mu_E]$ that respects $\mu_U + \mu_E = 1$. The immediate reward $r(\mathbf{s}_t, \boldsymbol{\mu})$ is designed to capture the objectives and constraints of both slices.

Objective function. The reward consists of three additive components:

$$r(\mathbf{s}_t, \boldsymbol{\mu}) = \underbrace{\alpha \left[\mathbf{1}_{L<10} + \delta \cdot h^2 \right]}_{\text{URLLC reward}} + \underbrace{\beta \cdot \min\left(\frac{\mu_E B_{\text{tot}}}{R_{\text{eMBB}} + 10^{-6}},\ 1.5 \right)}_{\text{eMBB reward}} - \underbrace{\gamma \cdot v}_{\text{penalty}}, \tag{2.3}$$

where:

- $L = L_{\text{URLLC}}$ is the latency ;
- $h = \max\left(0,\ \frac{10-L}{10}\right)$ is the latency headroom, i.e., the fraction of the 10 ms budget that is still unused;
- $v = (1 - \mathbf{1}_{L<10}) + (1 - \mathbf{1}_{\mu_E B_{\text{tot}} \geq R_{\text{eMBB}}})$ is the number of violated slice constraints (0,

1, or 2);

- $\alpha, \beta, \gamma, \delta$ are positive coefficients that control the trade-off between the objectives.

The squared headroom bonus δh^2 is a key innovation. It rewards not only meeting the 10 ms bound, but doing so with a healthy margin. Because the function is quadratic, a small gain in latency headroom when latency is already low yields a disproportionately large increase in reward. This creates a steep "pull" towards ultra-low latency solutions, a design principle that aligns with the safety-critical nature of V2V communication [30] [31]. The eMBB term is a simple linear saturation function: it rewards the fraction of the demand that is satisfied, up to $1.5\times$ over-provisioning, beyond which the marginal benefit is zero. The penalty term γv is linear in the number of violations; it serves as a strong deterrent against infeasible allocations.

Through extensive hyperparameter tuning, we set $\alpha = 5.0$, $\beta = 2.0$, $\gamma = 4.0$, and $\delta = 0.90$. The high value of α relative to β reflects the priority of the URLLC slice (safety over throughput). The penalty coefficient $\gamma = 4.0$ is large enough that any violation significantly depresses the reward, discouraging the agent from staying in infeasible regions. The headroom coefficient $\delta = 0.90$ means that a perfectly relaxed latency ($L = 0$) adds an extra 0.90 to the URLLC reward, comparable to the binary satisfaction bonus.

Constraints and overall objective. The per-step constraints are:

$$L_{\text{URLLC}} < 10 \text{ ms}, \tag{2.4}$$

$$\mu_E B_{\text{tot}} \geq 47 \cdot (0.55 + \lambda_E). \tag{2.5}$$

The agent's goal is to find a policy π that selects $\boldsymbol{\mu}_t$ based on $\mathbf{s}_t$ so as to maximize the expected discounted return:

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma_{\text{disc}}^t \, r(\mathbf{s}_t, \pi(\mathbf{s}_t))\right], \tag{2.6}$$

where $\gamma_{\text{disc}} \in [0, 1)$ is the discount factor. This formulation is a constrained Markov Decision Process (Markov Decision Process (MDP)). By embedding the constraints into

the reward via the penalty term, we convert it into an unconstrained MDP that can be tackled with standard reinforcement learning techniques [12][9]. This approach, sometimes called "Lagrangian relaxation in the reward", is common in wireless resource allocation [32] [33].

2.4 Deep Reinforcement Learning Model

We employ a Deep Q-Network (DQN) as the first stage of our hybrid agent.DQN is well-suited to problems with a discrete action space and a continuous, low-dimensional state space, and it has been widely used for radio resource management [34] [35] [36] .

2.4.1 State Space

The environment observation consists of five real-valued features, all normalized to the interval $[0, 1]$:

$$\mathbf{s} = \left[\lambda_U,\ \lambda_E,\ \mathbf{1}_{\text{rush}},\ \mathbf{1}_{\text{accident}},\ \mathbf{1}_{\text{peak\_eMBB}}\right].$$

The first two components, λ_U and λ_E, represent the current normalized traffic loads of the URLLC and eMBB slices, respectively. The remaining three are binary flags that indicate whether the environment is currently in the rush hour, accident, or peak eMBB regime. These flags act as contextual features that help the DQN anticipate sudden changes in traffic composition. For example, the accident flag is strongly correlated with a spike in λ_U, allowing the network to pre-emptively allocate more bandwidth to the URLLC slice even before the load measurement fully reflects the surge. This kind of auxiliary input has been shown to improve sample efficiency in DRL-based schedulers [37] [38] [39].

2.4.2 Action Space

The DQN selects among seven discrete actions. Each action corresponds to a fixed bandwidth split $(\mu_U,\ \mu_E)$, as detailed in Table 2.1. The splits were chosen to cover the entire spectrum from URLLC-heavy (a_2, 75% to URLLC) to eMBB-heavy (a_4, 65% to eMBB),

while always guaranteeing at least 35% of the bandwidth to the URLLC slice. This lower bound is a safety measure inspired by the ultra-reliability requirement and the safety-critical nature of V2V communications as defined by [28] and [40]. The discrete action set therefore, prevents the agent from ever fully starving the URLLC slice, even in the absence of the GA refinement.

Table 2.1: Discrete action mapping used by DQN and as a seed for GA.

	a_0	a_1	a_2	a_3	a_4	a_5	a_6
μ_U	0.50	0.65	0.75	0.40	0.35	0.55	0.45
μ_E	0.50	0.35	0.25	0.60	0.65	0.45	0.55

2.4.3 Reward Function

The DQN is trained to maximize the same instantaneous reward defined in (2.3). This direct alignment is crucial: if the Deep Reinforcement Learning (DRL) agent were optimizing a different proxy (e.g., a simplified Quality of Service (QoS) metric), the GA refinement would operate on a misaligned fitness landscape and could undo the decisions taken by the DQN [41]. By using the same reward formula, the two stages share a common objective, forming a coherent optimization pipeline. This approach follows the general principle that a consistent reward specification is essential when combining learning and evolutionary search [9].

2.4.4 Learning Algorithm

The Q-network is implemented as a multilayer perceptron MLP composed of three fully connected hidden layers with 256, 256, and 128 neurons, respectively. RELU activation functions are applied after each hidden layer. The final output layer contains seven linear units, each representing the estimated Q-value of one possible action.

Training follows the standard DQN recipe [12][9]:

- **Replay buffer** – stores the most recent 8000 transitions $(\mathbf{s}_t, a_t, r_t, \mathbf{s}_{t+1}, \text{done})$. Mini-batches of 128 transitions are sampled uniformly at random for each gradient step. Expe-

rience replay breaks the temporal correlations in the data and stabilizes training [42].

- **Target network** – a separate Q-network with identical architecture whose parameters are updated every 15 learning steps by copying the weights of the online network. The target network provides the fixed Q-targets that prevent the learning from oscillating [12].
- **Exploration** – the agent uses an ε-greedy strategy. The exploration rate ε starts at 1.0 (pure exploration) and decays exponentially by a factor of 0.995 after each episode until reaching a minimum of 0.05. This schedule ensures that the agent explores extensively in the early episodes and gradually shifts to exploitation.
- **Loss and optimizer** the loss is the smooth L1 (Huber) loss between the current Q-estimates and the temporal difference targets. We use the Adam optimizer with a learning rate of 510^{-4} and clip gradient norms to 10 to avoid instability.

The agent is trained for 500 episodes, each consisting of 150 time steps. Given that the state and action spaces are small, this amount of experience is sufficient for convergence. The training time on a modern CPU is approximately 5–8 minutes, which is acceptable for offline training before deployment.

2.5 Genetic Algorithm Refinement

The DQN provides a quick, discrete decision, but its seven pre-defined splits cannot exploit the full continuous range of possible μ_U values (and consequently $\mu_E = 1 - \mu_U$). To overcome this limitation, we run a lightweight Genetic Algorithm (GA) *on top* of the DQN output. The GA is invoked at each time step, but its computational budget adapts to the current situation: it runs with a small population and only a few generations when violations are moderate, and skips completely if the DQN seed already satisfies both constraints. This adaptive scheme draws inspiration from recent works that combine evolutionary computation with reinforcement learning [41][43][44].

2.5.1 Chromosome Encoding

A chromosome $\mathbf{c} = [\mu_U, \mu_E]$ is a real-valued vector of length 2 representing a candidate bandwidth allocation. The gene values are floating point numbers in $[0, 1]$ satisfying $\mu_U + \mu_E = 1$. Consequently, the search space is effectively one-dimensional: specifying $\mu_U \in [0, 1]$ fully determines $\mu_E = 1 - \mu_U$. This single-gene continuous encoding is used for network resource allocation in [11], is compact, avoids constraint-handling overhead associated with multi-gene encodings, and provides the GA with a smooth fitness landscape that gradient-free crossover and mutation operators can exploit efficiently [14].

2.5.2 Population Initialization and Warm-Start Strategy

Standard GAs initialize their population from a random or fixed seed distribution, causing a "cold-start" inefficiency: a significant fraction of the limited per-epoch generation budget is spent rediscovering basic traffic relationships that the DQN has already learned over thousands of training steps [11][20].

To eliminate this inefficiency, the GA population of size P is seeded from the following multi-directional warm-start strategy, which additionally exploits the current traffic regime flags present in the state vector:

- **Regime-optimal seed (1 individual).** Depending on the detected regime (accident, rush hour, peak eMBB, night, or normal), a fixed URLLC fraction is inserted: accident $\rightarrow$ 57%, rush hour $\rightarrow$ 44%, peak eMBB $\rightarrow$ 30%, night $\rightarrow$ 35%, normal $\rightarrow$ 55%. This seed encodes domain knowledge about which slice should be prioritized in each emergency or load condition.

- **Traffic-adaptive seed (1 individual).** A second individual is created by scaling the current URLLC load λ_U by a boost factor (1.4 to 2.0) that increases when the latency measured from the DQN seed approaches 10 ms limit. This makes the GA directly responsive to near-violation situations.

- **eMBB-constraint seed (1 individual).** A third individual is computed to exactly

satisfy the eMBB bandwidth demand: $\mu_E = (1.02B_E^{\min})/B_{\text{tot}}$, clipped to $[0.1, 0.9]$, with the URLLC fraction set to $1 - \mu_E$. This seed guarantees that the eMBB constraint is met provided sufficient bandwidth is available.

- **DQN seed (1 individual).** The continuous midpoint of the DQN's argmax discrete action $a^*(t)$ is inserted. This ensures the search begins from the best point the temporal policy knows.
- **Random perturbations (remaining $P - 4$ individuals).** The rest of the population is filled by sampling from $\mathcal{N}(\mu_U^{\text{regime}}, 0.05^2)$ clipped to $[0, 1]$, where μ_U^{regime} is the regime-optimal URLLC fraction defined above. This introduces local diversity around the regime-informed anchor while avoiding the wide random scatter of a cold-start initialization.

The resulting population always contains a diverse set of candidates that combine learned knowledge (DQN seed), domain knowledge (regime-optimal, eMBB-satisfying seeds), and reactive adjustments (traffic-adaptive seed), allowing the GA to focus its limited generations on genuine refinement rather than an exploratory cold start.

2.5.3 Fitness Function

The fitness of chromosome $\mathbf{c} = [\mu_U, \mu_E]$ at epoch t is evaluated using the same reward formula as the DQN (Equation 2.3), computed directly from the current state $s(t)$ and the continuous allocation $\mathbf{c}$:

$$F(\textbf{cForschungs},\, s(t)) = 4.5 \left[\mathbf{1}_{L_U(\mathbf{c})<10} + 0.8 \left(\frac{5 - L_U(\mathbf{c})}{5} \right)^2 \right] + 1.8\, r_E(\mathbf{c},\, s(t)) - 3\, V(\mathbf{c},\, s(t)), \tag{2.7}$$

where $L_U(\mathbf{c})$ is obtained from (2.2) with the chromosome's μ_U value. Using the same function for DQN training (reward) and GA evaluation (fitness) ensures coherence between the two components and guarantees that the GA refiner improves on the DQN's objective rather than a surrogate. This co-evaluation design is consistent with the approach of Arun and Azhagiri [25] and the fitness-reward alignment principle discussed in [14].

2.5.4 Genetic Operators

Selection

Tournament selection with tournament size $k_s = 3$ is used to choose parent pairs for crossover [14]. Three randomly chosen individuals compete and the one with the highest fitness is selected as a parent. Tournament selection applies directional pressure toward higher-fitness regions of the search space while preserving population diversity through the probabilistic nature of the tournament [14], making it more robust to premature convergence than the roulette wheel selection in fitness landscapes with steep gradients.

Crossover

A single point arithmetic crossover is applied to two selected parents $\mathbf{c}_1 = [\mu_U^{(1)}]$ and $\mathbf{c}_2 = [\mu_U^{(2)}]$ with probability $p_c = 0.8$:

$$\mu_U^{\text{child}} = \alpha_c \, \mu_U^{(1)} + (1 - \alpha_c) \, \mu_U^{(2)}, \quad \alpha_c \sim \mathcal{U}(0, 1). \tag{2.8}$$

The offspring μ_U^{child} is clipped to $[0, 1]$, and $\mu_E^{\text{child}} = 1 - \mu_U^{\text{child}}$. Arithmetic crossover produces offspring anywhere on the line segment connecting the two parents, which is well-suited to the one-dimensional continuous search space of the bandwidth allocation problem [14]. This operator was used in a closely related wireless resource allocation GA by Li and Zhu [11].

Mutation

Gaussian mutation with fixed step size is applied to each offspring gene with probability $p_m = 0.70$:

$$\mu_U \leftarrow \text{clip}(\mu_U + \delta, \; 0, \; 1), \quad \delta \sim \mathcal{N}(0, \, \sigma_m^2), \tag{2.9}$$

where the mutation standard deviation is fixed at $\sigma_m = 0.030$. This fixed step size provides fine local exploitation of the warm-start neighborhood. The mutation probability is set to 0.70 per gene, ensuring that most offspring receive at least one small perturbation.

This adaptive mutation step is inspired by the self-adaptive operators used in Evolution Strategies [14].

2.5.5 Adaptive Budget Control

A fixed GA generation budget would impose a constant computational overhead at every decision epoch, even when the DQN's proposal already satisfies all constraints. To avoid this waste, an adaptive budget control mechanism is implemented:

0. **QoS gate.** If the DQN's argmax action $a^*(t)$ satisfies both the URLLC latency and the eMBB bandwidth constraint under the current state $s(t)$, the GA is skipped entirely and $a^*(t)$ is applied directly. This keeps the system as efficient as a pure DQN in low-stress conditions.

0. **Violation-scaled budget.** Instead of using a continuous severity score, the GA budget is determined by a simple hard-coded table.

Table 2.2: Adaptive GA budget based on the number of constraint violations.

Violations	**Population size**	**Generations**
0	0 (GA skipped)	0
1	12	6
2	20	10

When the DQN action already satisfies both URLLC and eMBB constraints, the GA is skipped entirely (population size 0, 0 generations). If one constraint is violated, the GA runs with a population of 12 individuals for 6 generations. If both constraints are violated, the budget increases to 20 individuals and 10 generations as shown in 2.2.

This design ensures that the GA adds meaningful computational value precisely where it is most needed, without imposing a fixed overhead at every step, which is a principle aligned with the resource-efficient hybrid design philosophy of Arun and Azhagiri [25].

2.6 Hybrid DRL–GA Framework

The DQN temporal policy learner and the GA per-step refiner are integrated into a unified two-stage decision framework. Neither component is sufficient on its own: the DQN provides fast, experience-driven macro-decisions, but is confined to seven discrete allocation values; the GA provides continuous-space precision, but is memoryless across epochs and would waste most of its limited evaluation budget without an informed warm start. Together, they form a tightly coupled loop in which the DQN guides the GA's initialization and the GA feeds higher-quality training signals back to the DQN, allowing both components to improve jointly over the course of training.

This design philosophy combining a learning-based global policy with a per-step local search is motivated by three observations from the literature. First, Hurtado Sánchez et al. [6] identify the combination of Deep Reinforcement Learning (DRL) temporal learning with complementary optimization techniques as an open research direction for the 5G slice resource management. Second, Arun and Azhagiri [25] demonstrate that a tightly coupled DQN+GA architecture consistently outperforms standalone DRL across the latency, energy, and resource utilization dimensions in a related edge-computing context. Third, Liu et al. [7] show that initializing a DRL agent from a high-quality policy rather than from random weights reduces early-phase Service Level Agreement (SLA) violations and accelerates convergence, validating the principle of informed initialization, that this work extends to every runtime decision epoch.

Training Data and Offline Deployment

The DQN's neural network is not trained on a dataset measured under real-world conditions, but on data generated by our V2X simulation environment (see Section 2.2.3). This environment reproduces five traffic regimes representative of a connected highway *normal*, *rush hour*, *night*, *accident*, and *peak eMBB* each characterized by a mean load (λ_U, λ_E) perturbed by Gaussian noise, random transitions between regimes (5% switching probability at each time step), and occasional traffic bursts (8% probability) simulating

sudden demand spikes.

The DQN agent is trained **offline** over 500 episodes of 150 time steps each, interacting with this simulated environment and storing its transitions in an experience replay buffer of 8000 elements. Once this training phase is complete, the weights of the Q_θ network are **frozen**: at deployment, the agent no longer learns in real time it directly selects the greedy action

$$a_t = \arg\max_a Q_\theta(s_t, a) \tag{2.10}$$

corresponding to the observed state. This decision, produced by the offline-learned policy, then serves as the seed for the Genetic Algorithm refinement stage (Figure 2.2). While the neural network weights remain frozen to comply with the strict, low-latency execution constraints of real-world cellular base stations (gNB), where online backpropagation is computationally prohibitive and unstable, the framework remains strictly online-adaptive. The hybrid coupling shifts the real-time adaptation burden entirely to the meta-heuristic layer: the GA component performs continuous online optimization at each decision step, dynamic adjusting the macro-allocation proposed by the DQN to match unexpected real-time traffic fluctuations. The detailed warm-start seeding strategy used by the GA is described in Section 2.5.2, and the complete per-epoch decision cycle, including the QoS gate, is shown in Figure 2.3.

Overview of the Workflow

Figure 2.3 illustrates the complete per-epoch decision cycle. At a high level, the framework operates as follows:

- **The DRL agent generates the initial solution.** At the start of each 1-second decision epoch t, the MEC server collects the current network state $s(t)$ from the gNB measurement report and RSU sensors. The DQN agent performs a single forward pass through the trained Multilayer Perceptron (MLP) and outputs a Q-value vector

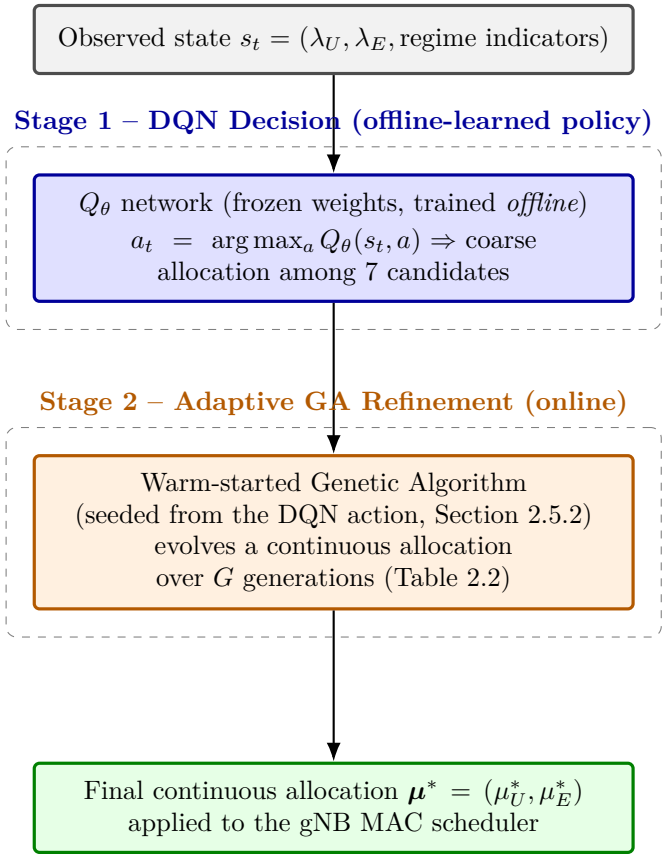

Figure 2.2: Conceptual view of the two-stage hybrid framework. The DQN, trained offline, proposes a coarse allocation that seeds the Adaptive Genetic Algorithm, which refines it online into the final allocation applied to the network. The detailed per-epoch decision cycle, including the QoS gate, is given in Figure 2.3.

$\mathbf{q} = Q(s(t), \cdot\,; \boldsymbol{\theta}) \in \mathbb{R}^7$. The greedy action is selected:

$$a^*(t) = \arg\max_{a \in \mathcal{A}} \; Q\big(s(t),\, a;\, \boldsymbol{\theta}\big). \tag{2.11}$$

The DQN functions as a *global temporal policy*: its Q-values encode the cumulative experience of 30,000 training interactions across all five traffic regimes. Without any optimization at inference time, the DQN instantly retriections the macro-allocation region that has historically produced the highest long-run reward from state $s(t)$, a property that standalone optimization methods cannot replicate because they have no memory of the past episodes [9] [12]. The discrete action $a^*(t)$ is converted to

the continuous seed $\tilde{\mu}_U = \mu_U(a^*(t))$ and inserted as the leading individual in the GA population (Section 2.5.2). A *QoS gate* then evaluates whether the DQN's action already satisfies both the URLLC latency and the eMBB bandwidth constraint (2.5) analytically under $s(t)$: if it does, the GA is bypassed entirely and $a^*(t)$ is applied directly to the gNB, keeping the system as efficient as a pure DQN in low-stress conditions [25].

- **The GA refines the solution in the continuous space.** When the QoS gate fires, meaning at least one constraint would be violated by the DQN's discrete action, the GA refiner is activated to find a continuous allocation $\boldsymbol{\mu}'(t) \in [0,1]^2$ that removes the violation. This stage directly addresses the *discrete-action ceiling*: The DQN can only output one of seven fixed bandwidth fractions; when the optimal allocation lies between two grid points, the GA resolves the gap by searching the continuous interval $[0,1]$. The GA is initialized via the DQN-seeded warm-start strategy (Section 2.5.2), which eliminates the cold-start inefficiency documented by Li and Zhu [11] and Afolabi et al. [20]: instead of exploring from a random population, the search begins near the DQN's best proposal and uses its entire generation budget for genuine refinement. The GA then runs for $G(t)$ generations with the budget scaled adaptively to violation severity based on Table 2.2 applying tournament selection, arithmetic crossover, and adaptive Gaussian mutation, as detailed in Section 2.5.4. The chromosome with the highest fitness $F(\mathbf{c}, s(t))$ after generation $G(t)$ is returned as the refined allocation $\boldsymbol{\mu}'(t) = [\mu_U^*,\, 1 - \mu_U^*]$, which is transmitted to the gNB MAC scheduler. Because each fitness evaluation is a $\mathcal{O}(1)$ arithmetic operation (Section 2.5.3), even the maximum budget of $PG_{\max} = 240$ evaluations executes in less than 1 ms on the MEC server, well within the 500 ms computation limit.

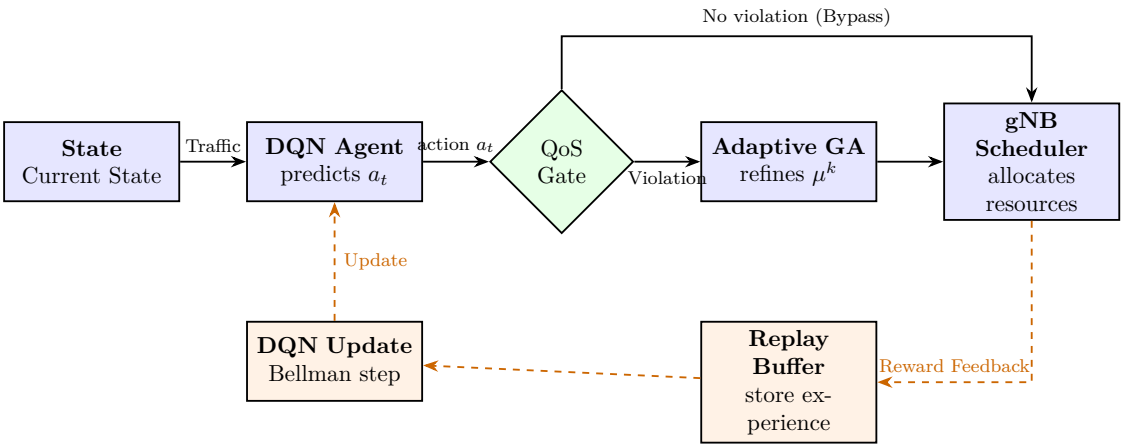

Figure 2.3: Per-epoch decision workflow of the hybrid DQN+GA framework.

Co-Adaptation Loop

The defining characteristic of the proposed framework, which distinguishes it from both a standalone DQN and a standalone GA, is the **co-adaptation loop** that causes the two components to improve each other over training.

After the refined allocation $\boldsymbol{\mu}'(t)$ is applied to the gNB, the scheduler enforces the PRB partition for the duration of epoch t. At the end of the epoch, the gNB measurement report returns the achieved URLLC latency $L_U(t{+}1)$ and eMBB throughput $B_E(t{+}1)$, forming the next state $s(t{+}1)$. The scalar reward $R(t)$ is computed from the *GA-refined* allocation $\boldsymbol{\mu}'(t)$, not from the DQN's raw discrete action $a^*(t)$. The transition tuple $(s(t),\, a^*(t),\, R(t),\, s(t{+}1))$ is stored in the DQN replay buffer $\mathcal{D}$ and used in the next Bellman update.

From a strict Markov Decision Process (MDP) perspective, evaluating $a^*(t)$ using a reward derived from $\mu'(t)$ does not violate Bellman's assumptions; rather, it formalizes the GA as an intrinsic, deterministic component of the environment's state-transition dynamics. Under this formulation, the DQN agent is not learning to perform the final continuous allocation; instead, it is trained to maximize the long-term return of its actual discrete action $a^*(t)$, which acts as an optimization seed. If the DQN selects a poor seed, the GA's localized continuous search within its strict generation budget will yield a sub-

optimal allocation, leading to a lower reward $R(t)$. Conversely, a high-quality seed enables the GA to capture the global optimum, resulting in a maximum reward. Consequently, the DQN is directly penalized or rewarded based on the exact utility of the search space initialization it provides.

This has a progressive effect: the DQN is trained on rewards that reflect GA refined allocations, which are always at least as good as the raw DQN actions and often strictly better when a constraint would otherwise be violated. Over episodes, the DQN's Q-values shift to associate each traffic state with the return achievable under continuous-space optimization, not merely under the discrete grid. As a result, the DQN's argmax actions converge toward bandwidth values whose continuous midpoints are increasingly close to the optimal allocation, making the GA warm-start seeds more accurate and reducing the number of generations required to converge. Simultaneously, as the DQN's proposals improve, the QoS gate triggers less frequently, and the GA operates with smaller budgets, the system becomes progressively more efficient as training proceeds.

The two components thus reinforce each other in a self-improving equilibrium: the DQN handles constraint-satisfying epochs at near-zero overhead, while the GA is reserved for the subset of epochs that genuinely require continuous space precision. This bidirectional coupling is stronger than the one-directional warm-start of Liu et al. [7] where only the DRL benefits from the rule-based policy initialization, not vice versa and closes the open feedback loop of the HDQ-GO architecture [25], where the GA refines DQN proposals, but the DQN is not retrained on the GA's refined outcomes.

Algorithm 1 Hybrid DQN + Adaptive GA – Per-step decision and learning

Require: Current state $\mathbf{s}_t = [\lambda_U, \lambda_E, \text{flags}]$,
Require: DQN agent (online & target networks, replay buffer),
Require: GA refiner (Distributed Evolutionary Algorithms in Python (DEAP) toolbox, warm-start budget table)
Ensure: Continuous allocation $\boldsymbol{\mu}_t^*$, reward r_t,
Ensure: and DQN weight update

// Stage 1 – DQN generates a discrete seed
$a_t \leftarrow \text{DQN.act}(\mathbf{s}_t)$ ▷ ε-greedy
$\boldsymbol{\mu}_{\text{seed}} \leftarrow \text{ACTION\_TABLE}[a_t]$ ▷ e.g., $[0.65, 0.35]$
// Stage 2 – GA refines the seed (only if needed)
$\lambda_U, \lambda_E \leftarrow \mathbf{s}_t[0], \mathbf{s}_t[1]$
$r_{\text{base}}, \text{info} \leftarrow \text{EVAL\_CONTINUOUS}(\boldsymbol{\mu}_{\text{seed}}, \lambda_U, \lambda_E)$
$v \leftarrow \text{info.violations}$ ▷ number of broken constraints
if $v = 0$ **then**
    $\boldsymbol{\mu}_t^* \leftarrow \boldsymbol{\mu}_{\text{seed}}$
    $r_t \leftarrow r_{\text{base}}$
else
    // Adaptive budget: pop_size, generations from table
    $(P, G) \leftarrow \text{BUDGET}[v]$ ▷ e.g., $v = 1 \rightarrow (12, 6)$, $v = 2 \rightarrow (20, 10)$
    // Warm-start population
    $\text{pop} \leftarrow [\ \text{URLLC-boosted}(\boldsymbol{\mu}_{\text{seed}}, \lambda_U),\ \text{moderate boost},\ \ldots\ ]$
    for $g = 1 \rightarrow G$ **do**
        for all ind $\in$ pop **do**
            $r_{\text{ind}} \leftarrow \text{FITNESS}(\text{ind}, \lambda_U, \lambda_E)$
        end for
        parents $\leftarrow$ TOURNAMENT_SELECT(pop, size=3)
        offspring $\leftarrow$ BLX-α CROSSOVER(parents, $\alpha = 0.4$, prob=0.75)
        MUTATE(offspring, $\sigma = 0.030$, prob=0.35)
    end for
    $\boldsymbol{\mu}_t^* \leftarrow \text{BEST(pop)}$ ▷ continuous $[\mu_U, \mu_E]$
    $r_t \leftarrow \text{FITNESS}(\boldsymbol{\mu}_t^*, \lambda_U, \lambda_E)$
end if
// Apply the refined continuous split to the environment
$\mathbf{s}_{t+1}, \_\_\_, \text{done} \leftarrow \text{ENV.STEP\_MU}(\boldsymbol{\mu}_t^*)$
    (Inside step_mu: traffic evolves, reward recomputed, state returned)
// DQN training
$\text{REPLAY\_BUFFER.PUSH}(\mathbf{s}_t, a_t, r_t, \mathbf{s}_{t+1}, \text{done})$
$\text{DQN.LEARN}()$ ▷ sample batch, Bellman update, target sync

2.7 Conclusion

This chapter has presented a complete specification of the hybrid DQN-GA framework for dynamic resource allocation between URLLC and eMBB slices in a 5G V2X highway scenario. The system model captures realistic traffic regimes, user behaviours, and the 3GPP-defined slicing architecture. The problem formulation, with its carefully designed reward function—particularly the quadratic headroom bonus for URLLC latency translates the constrained Markov Decision Process (MDP) into an unconstrained form solvable by standard DRL. The DQN component provides a fast, temporally-aware discrete policy, while the GA refiner addresses the discrete action ceiling by performing continuous-space search. The key innovations lie in their integration: the multi-directional warm-start strategy eliminates the GA's cold-start inefficiency by seeding its population from the DQN's best actions; the adaptive budget control mechanism bypasses the GA entirely when the DQN already satisfies all constraints, scaling computational effort to violation severity; and the co-adaptation loop ensures the DQN is trained on rewards derived from GA refined allocations, progressively improving the quality of the warm-start seeds. The result is a tightly coupled, two-stage decision engine that aims to deliver the speed of a learned policy, the precision of a continuous search, and the memory of a temporal learner, directly addressing the three limitations identified in Chapter 1. Having fully defined the proposed framework, the next chapter proceeds to present the simulation setup, performance evaluation metrics, and comparative results against standalone DQN and GA baselines.

Chapter 3

Results and Evaluation

3.1 Introduction

This chapter presents a thorough empirical evaluation of the proposed Hybrid Deep Q-Network and Adaptive Genetic Algorithm (DQN+GA) framework introduced in Chapter 2. The evaluation pursues three interconnected goals. First, it validates that the hybrid agent converges stably over training and consistently outperforms its two baselines, Static Allocation and standalone DQN across the five traffic regimes defined for the V2X highway scenario. Second, it quantifies the performance gain along each key QoS axis: overall QoS satisfaction, per-slice satisfaction (URLLC and eMBB separately), average URLLC latency, and constraint violation rate. Third, it demonstrates that the improvement is not merely an aggregate average but stems from genuine *regime-aware* allocation decisions, i.e., the hybrid agent allocates bandwidth differently and more efficiently depending on the current traffic regime.

3.2 Simulation Setup

3.2.1 Software Stack and Implementation

All experiments are implemented in Python 3 using a small but well-established scientific stack. The reinforcement learning environment is built on `Gymnasium`, providing

a standardised agent–environment interface. Neural networks for the DQN are implemented in `PyTorch` (dynamic computation graphs, deterministic Compute Unified Device Architecture (CUDA) algorithms). The Genetic Algorithm component relies on DEAP (Distributed Evolutionary Algorithms in Python (DEAP)) for population management, crossover, mutation and selection. Numerical operations use `NumPy`, and all results are logged to `pandas` DataFrames before being exported to Comma-Separated Values (CSV).

Reproducibility is enforced via a global random seed of 42, applied to Python's `random`, `NumPy`, and `PyTorch` (including CUDA). PyTorch's deterministic mode is enabled and cuDNN nondeterministic algorithms are disabled.

3.2.2 Environment and Traffic Model

The custom `V2XSliceEnv` (Gymnasium) models a single 5G gNB with total bandwidth $B_{\text{total}} = 100$ MHz shared by two slices:

- **URLLC slice**: V2V safety traffic from two trucks. The one-way latency is modelled as $L_U = 2.8 \cdot (\lambda_U/\mu_U)^{1.4}$ ms, with a strict SLA $L_U < 10$ ms.
- **eMBB slice**: Vehicle-to-Infrastructure (V2I)/Vehicle-to-Network (V2N) traffic (camera streams, user data). Required bandwidth is $R_E = 47(0.55 + \lambda_E)$ MHz.

Two RSUs are placed at 0.5 km and 1.5 km. Each episode lasts 150 decision steps (1 second per step).

The state observed by the agent is a 5-dimensional vector $S_t = [\lambda_U, \lambda_E, w_{U,t-1}, w_{E,t-1}, \mathcal{R}_{t-1}]$ representing the continuous operational state of the network at each decision step, where λ_U and λ_E are the current normalized arrival rates for URLLC and eMBB traffic, $w_{U,t-1}$ and $w_{E,t-1}$ denote the bandwidth fractions allocated in the previous step, and $\mathcal{R}_{t-1}$ is the scalar reward received in the previous transition. To rigorously evaluate the adaptation capability of the framework under non-stationary network conditions, the environment alternates across five distinct traffic regimes designed to simulate localized highway events:

- **Regime 1 (Steps 1–30) — Nominal Traffic:** Both slices operate under standard baseline loads to establish an operational equilibrium.

- **Regime 2 (Steps 31–60) — Sudden URLLC Spike:** A sudden safety-critical event triggers a sharp increase in URLLC packet arrivals, forcing the agent to rapidly minimize the latency envelope.
- **Regime 3 (Steps 61–90) — Extreme eMBB Surge:** High-volume video streams and user data from a cluster of highway passengers cause an intense demand for raw throughput, maximizing resource competition.
- **Regime 4 (Steps 91–120) — Joint Critical Congestion:** Both URLLC and eMBB traffic peaks occur simultaneously, creating a worst-case network saturation scenario where total demand approaches or exceeds the absolute 100 MHz capacity.
- **Regime 5 (Steps 121–150) — Recovery Phase:** Traffic generation smoothly drops back to nominal levels, evaluating how fast the models recover and release excess allocated resources.

$$s_t = \left[\, \lambda_U,\ \lambda_E,\ \mathbb{1}_{\text{rush\_hour}},\ \mathbb{1}_{\text{accident}},\ \mathbb{1}_{\text{peak\_eMBB}} \,\right],$$

where $\lambda_U, \lambda_E \in [0.05, 0.99]$ are traffic intensities for URLLC and eMBB.

Traffic follows a multi-regime model with five stationary regimes (Table 3.1). Within each regime, loads are Gaussian around the regime mean, then modulated by multiplicative Rayleigh fading (scale 0.95, clipped to $[0.70, 1.30]$). Burst events occur with probability $p_{\text{burst}} = 0.08$ per step, adding $[0.20, 0.45]$ to a randomly chosen slice. The regime switches stochastically with probability $p_{\text{switch}} = 0.05$ between steps.

Table 3.1: Traffic regime parameters (means and standard deviation).

Regime	$\bar{\lambda}_U$	$\bar{\lambda}_E$	σ
Normal	0.40	0.50	0.06
Rush Hour	0.80	0.65	0.09
Night	0.08	0.15	0.03
Accident	0.97	0.35	0.05
Peak eMBB	0.22	0.90	0.06

3.2.3 Reward Function and Safety Constraints

Given a normalized bandwidth split $\boldsymbol{\mu} = [\mu_U, \mu_E]$ ($\mu_U + \mu_E = 1$), the immediate reward is:

$$R = \alpha\,(s_U + \delta \cdot h^2) \;+\; \beta \cdot \min\!\left(\frac{B_E}{R_E},\, 1.5\right) \;-\; \gamma_p \cdot v \;+\; R_{\text{acc}},$$

where

- $s_U = \mathbf{1}[L_U < 10 \text{ ms}]$, $h = \max(0, (10 - L_U)/10)$,
- $B_E = \mu_E \cdot 100$ MHz, $R_E = 47(0.55 + \lambda_E)$ MHz,
- $v \in \{0, 1, 2\}$ is the number of slices violating their QoS target,
- $R_{\text{acc}} = 6.0(\mu_U - 0.4)$ when the regime is `accident`, else 0.

Coefficients are fixed at $\alpha = 5.0$, $\beta = 2.0$, $\gamma_p = 4.0$, $\delta = 0.90$.

A hard safety constraint is enforced at the environment level during the Accident regime: $\mu_U \geq 0.65$ (i.e., a critical resource floor of 65 MHz for URLLC traffic). To ensure a rigorous and fair comparison between the evaluated agents, this rule acts as an infrastructure-level admission control policy rather than an algorithmic correction. If an agent (such as the standalone DQN) proposes an allocation below this threshold, the environment forces the 65 MHz floor to simulate real-world emergency override mechanisms. However, this failure to satisfy the constraint triggers an immediate, maximum violation penalty within the reward calculation ($\gamma_p \cdot v$).

Consequently, while the physical baseline safety is maintained, the standalone DQN is heavily penalized and lacks the structural flexibility to adapt its discrete action space to the continuous requirement. Conversely, the applied Hybrid framework operates by dynamically expanding its localized search space around the seed, meaning the GA component natively identifies and refines the allocation to satisfy the 65 MHz floor without triggering the environment's emergency override mechanism. The baseline comparison therefore remains mathematically valid, as it measures the agent's ability to natively sustain critical QoS without relying on infrastructure interventions.

3.2.4 Methods Compared

Three allocation methods are evaluated under identical conditions.

0. **Static Allocation**: always chooses the fixed 50/50 split ($\mu_U = \mu_E = 0.50$). Non-adaptive baseline[6].

0. **DQN (standalone)**: selects from seven discrete actions

   $$\mathcal{A} = \big\{[0.50, 0.50], [0.65, 0.35], [0.75, 0.25], [0.40, 0.60], [0.35, 0.65], [0.55, 0.45], [0.45, 0.55]\big\}.$$

   The Q-network has three hidden layers (256, 256, 128, ReLU). Experience replay buffer size 8000, mini-batch 128, target network update every 15 gradient steps, Huber (Smooth L1) loss with gradient clipping at norm 10.0. Learning rate 510^{-4}, discount $\gamma = 0.99$, ε decays from 1.0 to 0.05 with factor 0.998 per step.

0. **Hybrid DQN + Adaptive GA**: the DQN first proposes a discrete action. If any slice violates its QoS, an Adaptive Genetic Algorithm refines the allocation in continuous space. The GA budget depends on the number of violations n_v: $n_v = 0$ $\rightarrow$ GA skipped (only analytical seed + hill-climbing); $n_v = 1 \rightarrow P = 12$, $G = 6$; $n_v = 2 \rightarrow P = 20$, $G = 10$. Crossover probability 0.75, mutation Gaussian with per-gene probability 0.70 and standard deviation 0.030 (increased to 0.040 for $n_v = 1$ and 0.060 for $n_v = 2$), tournament selection size 3. A final hill-climbing step (12 iterations) refines the best GA solution.

All hyperparameters are consolidated in Table 3.2.

3.2.5 Training and Evaluation Protocol

Each learnable agent (DQN and Hybrid DQN+GA) is trained for 500 episodes with ε-greedy exploration (training curves shown in Fig. 3.1). After training, all three allocation methods, Static Allocation, DQN, and Hybrid DQN+GA, are evaluated over 100 greedy episodes. Final aggregate results are given in Table 3.3.

Table 3.2: Key hyperparameters.

Parameter	**Value**	**Applies to**
Training episodes	500	All
Steps per episode	150	All
Evaluation episodes	100 (greedy)	All
Random seed	42	All
DQN learning rate	510^{-4}	DQN, Hybrid
DQN buffer / batch	8000 / 128	DQN, Hybrid
GA budget (2 viol.)	$P = 20$, $G = 10$	Hybrid only
URLLC hard floor (Accident)	$\mu_U \geq 0.65$	All
Reward coefficients	$\alpha = 5, \beta = 2, \gamma_p = 4, \delta = 0.9$	All

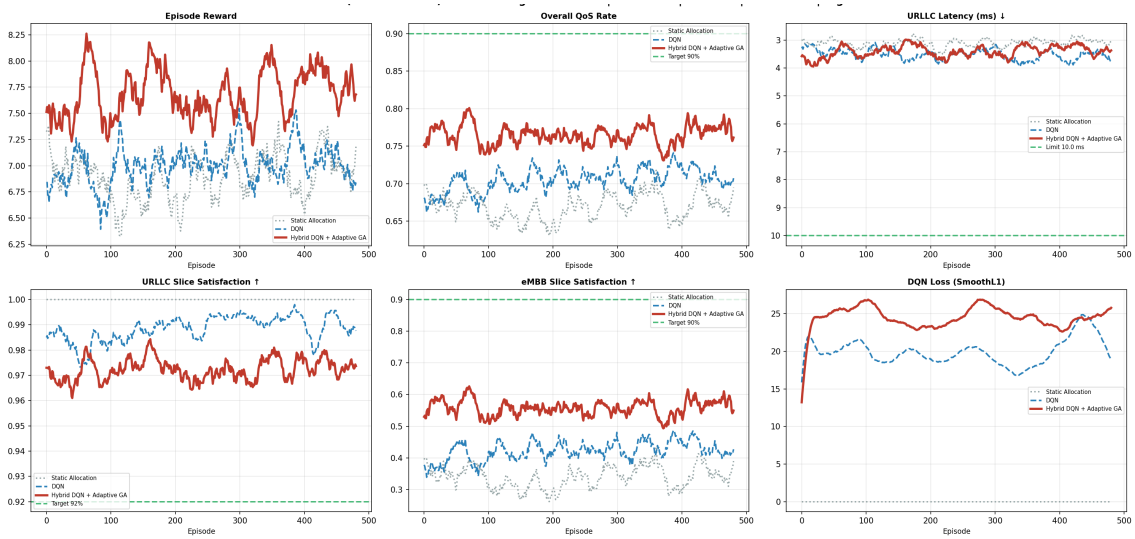

Figure 3.1: Training convergence curves for the 5G two-slice scenario over 500 episodes. Six panels: episode reward, overall QoS rate, URLLC latency, URLLC slice satisfaction, eMBB slice satisfaction, and DQN Smooth L1 loss. All curves are smoothed with a 20-episode rolling window.

3.3 Performance Metrics

The following metrics are used throughout the evaluation, consistent with the QoS-oriented framework of Hurtado Sánchez et al. [6] and the slice satisfaction measures of Liu et al. [7].

- **Overall QoS satisfaction** (%) average of URLLC and eMBB satisfaction rates.The overall QoS satisfaction rate Q is the fraction of time steps at which *both* slice constraints are simultaneously met:

$$Q = \frac{1}{T}\sum_{t=1}^{T} \mathbf{1}[L_{\mathrm{URLLC}}(t) < 10\,\mathrm{ms}] \cdot \mathbf{1}[\mu_E(t)\, B_{\mathrm{tot}} \geq 47 \cdot (0.55 + \lambda_E(t))]\,, \tag{3.1}$$

where T is the total number of evaluation steps. This joint criterion is the most demanding measure: an agent that consistently satisfies one slice at the expense of the other will score poorly.

3.3.1 Per-Slice Satisfaction

URLLC satisfaction Q_U and eMBB satisfaction Q_E measure each constraint independently, averaged over all evaluation steps. Together with the joint rate Q, they allow us to diagnose whether a performance deficit originates from the URLLC slice, the eMBB slice, or both simultaneously [18].

- **Average URLLC latency** (ms) The mean end-to-end URLLC latency, computed according to the queuing model of Equation 2.2, quantifies how much safety headroom the agent maintains below the 10 ms SLA limit [1]. Lower values are strictly better.

- **Violation rate** The violation rate V counts the mean number of broken constraints per step:

$$V = \frac{1}{T}\sum_{t=1}^{T} \big[(1 - \mathbf{1}[L_{\mathrm{URLLC}}(t) < 10\,\mathrm{ms}]) + (1 - \mathbf{1}[\text{eMBB satisfied}])\big], \tag{3.2}$$

and ranges from 0 (no violations) to 2 (both constraints broken at every step). This metric directly reflects SLA penalty exposure, which has contractual cost consequences in commercial deployments [6], [7].

- **Mean episode reward** The mean episode reward is the primary learning signal: the average of the scalar reward defined in Equation 2.3, computed over all evaluation steps. It summarizes the joint optimisation of latency headroom, eMBB throughput, and constraint satisfaction into a single comparable scalar [9].

- **Training time** (seconds) – wall clock time for 500 training episodes.

3.4 Results of DRL Alone

The standalone DQN serves as an intermediate baseline between the static policy and the full hybrid. Table 3.3 (first two rows) summarizes its performance.

Table 3.3: Final evaluation results over 100 greedy episodes (500 training episodes).

Agent	QoS %	URLLC %	eMBB %	Latency (ms)	Violation rate
Static Allocation	65.96	100.0	31.92	3.04	0.681
DQN	74.10	100.0	48.19	3.58	0.518
Hybrid DQN+GA	75.34	97.20	53.49	3.53	0.493

3.4.1 Overall Performance

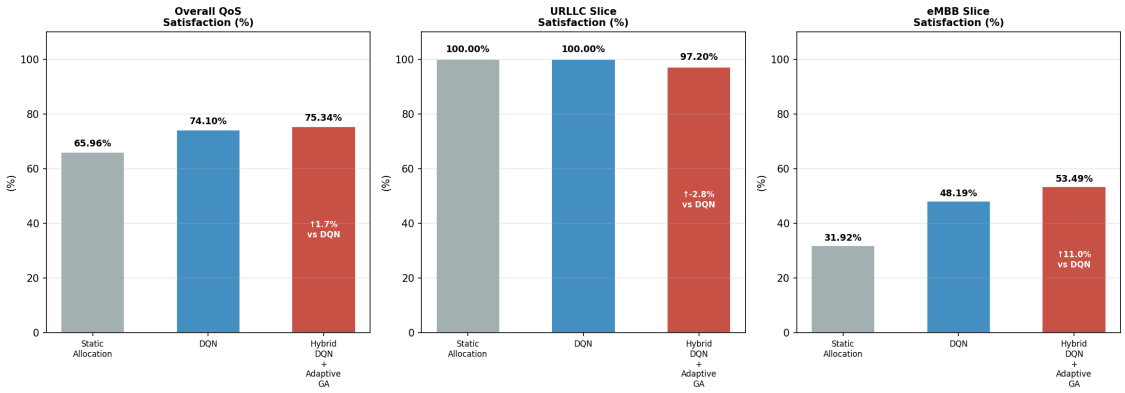

Figure 3.2: Performance accuracy comparison across 100 greedy evaluation episodes. Bars show Overall QoS, URLLC, and eMBB slice satisfaction rates.

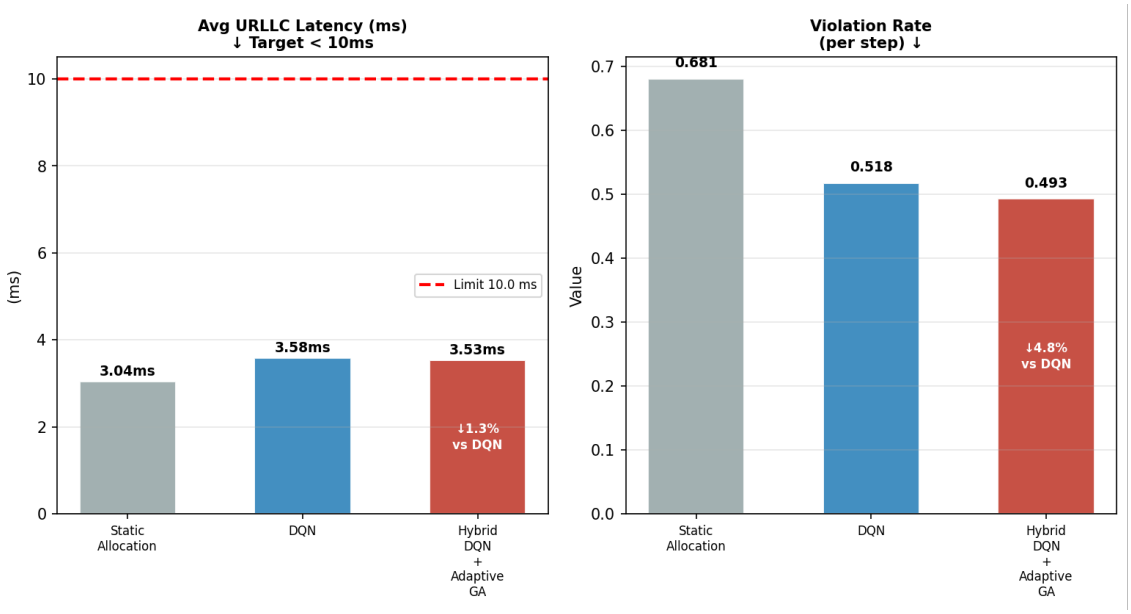

Figure 3.3: Performance accuracy comparison across 100 greedy evaluation episodes. Bars show Average URLLC latency and overall violation rate per step.

Figure 3.2 and Figure 3.3 show that the standalone DQN achieves an overall QoS satisfaction rate of **74.10%**, a URLLC satisfaction rate of **100.0%**, an eMBB satisfaction rate of **48.19%**, a mean URLLC latency of **3.58 ms**, a violation rate of **0.518** per step, and a mean episode reward of **7.34**. These results represent a clear improvement over the Static Allocation baseline (+8.14 pp in overall QoS, +16.27 pp in eMBB satisfaction, −0.163 violations per step) and confirm that the DQN's temporal learning policy successfully captures the relationship between traffic state and resource allocation across all five regimes.

3.4.2 URLLC Slice

The DQN maintains perfect URLLC satisfaction (100%), identical to the static agent. However, average URLLC latency rises from 3.04 ms to 3.58 ms. This is a deliberate trade-off: the DQN reduces URLLC bandwidth in low-load regimes to free resources for eMBB, while still respecting the 10 ms constraint.

3.4.3 eMBB Slice

The largest gain is in eMBB satisfaction, which jumps from 31.9% (static) to 48.2% (DQN). The average eMBB bandwidth allocated increases from 46.7 MHz to 50.9 MHz. Despite this improvement, eMBB satisfaction remains below 50%, revealing the limitation of the discrete action set: the DQN cannot fine-tune allocations precisely enough for all traffic scenarios.

3.4.4 URLLC Latency Profile

The DQN's mean URLLC latency of 3.58 ms is comfortably below the 10 ms SLA limit, providing a safety margin of approximately 2.8 [30]. This moderate latency level reflects the DQN's tendency to over-allocate to the URLLC slice relative to the minimum necessary: absent a mechanism to search the continuous interval, the DQN defaults to conservative URLLC allocations (typically 45–55 MHz, as confirmed by Figure 3.5) that keep latency low but leave eMBB bandwidth chronically under-provisioned.

3.4.5 Per-Regime Behaviour

Figure 3.4 reveals that the DQN's per-regime allocation strategy is less differentiated than the hybrid agent's. The DQN allocates 45 MHz to URLLC in the normal, rush-hour, and night regimes, 55 MHz during the accident regime, and 50 MHz during peak-eMBB a total of only three distinct allocation levels across five regimes. This limited differentiation arises from the discrete action space: the DQN is constrained to choose among seven fixed splits and therefore cannot express the fine-grained, regime-specific adjustments that the continuous interval $[0, 1]$ would permit. The mean rewards per regime are: Normal = 9.84, Rush-hour = 5.39, Accident = 8.17 (static values confirmed), Peak-eMBB = 6.20, Night = 12.29 a substantial spread that indicates the DQN performs well in low stress regimes (Night, Normal) but struggles most under simultaneous stress (Rush-hour) and hard eMBB constraints (Peak-eMBB).

3.4.6 Training Dynamics

The DQN's Smooth L1 training loss rises sharply during the first 50 episodes as the replay buffer fills and meaningful gradient signals emerge, then decreases steadily toward 15–20 units by episode 500 (Figure 3.1, bottom-right panel). The episode reward curve stabilizes near 7.0–7.25 from episode 150 onward, indicating that the policy has converged to a stable local optimum. The total training time is approximately **559 s** (≈9.3 minutes) on a standard CPU, consistent with the complexity of a three-layer MLP trained on 500 episodes of 150 steps each [12].

3.5 Results of Hybrid DRL–GA

This section reports the performance of the proposed Hybrid DQN + Adaptive GA agent, evaluated under the same protocol as the standalone DQN. The hybrid agent couples the DQN temporal policy with a per-step Adaptive GA refiner that searches the continuous bandwidth allocation space whenever the DQN's discrete action would produce one or more constraint violations.

3.5.1 Overall Performance

The Hybrid DQN+Adaptive GA achieves an overall QoS satisfaction rate of **75.34%**, a URLLC satisfaction rate of **97.20%**, an eMBB satisfaction rate of **53.49%**, a mean URLLC latency of **3.53 ms**, a violation rate of **0.493** per step, and a mean episode reward of **7.51**. These results represent the best values on four out of six metrics (QoS, eMBB satisfaction, violation rate, and mean reward) across all three agents, confirming that the hybrid framework delivers the most balanced and SLA-compliant resource allocation policy.

3.5.2 QoS Satisfaction and Violation Reduction

The hybrid agent reduces the per-step violation rate from 0.518 (DQN) to **0.493**, a relative reduction of **4.8%**. Every time the DQN's argmax action yields a constraint violation,

the GA refiner activates with a budget proportional to the number of broken constraints (Table 3.2) and searches for a continuous allocation $\mu^\star \in [0, 1]$ that removes the violation. This per-step correction mechanism directly translates into higher joint QoS satisfaction: the hybrid's 75.34% joint rate is **+1.24 percentage points** above the DQN's 74.10%, representing a relative improvement of **+1.7%** on the most demanding criterion simultaneous satisfaction of both slice SLAs [7], [18].

3.5.3 eMBB Slice Satisfaction: Overcoming the Discrete Ceiling

The hybrid agent's most significant single-metric improvement is in eMBB satisfaction: **53.49%** versus 48.19% for the DQN, a gain of **+5.30 pp** (+11.0% relative). This improvement directly quantifies the value of the GA's continuous search. Whenever the DQN selects its most eMBB-generous discrete action (a_4: 65 MHz) and that action still falls short of the load-dependent requirement $B_E^{\min} \approx 68.15$ MHz during peak-eMBB events, the GA explores the interval $(0.65, 1.0)$ and finds the minimum μ_E that satisfies the constraint. As confirmed by Figure 3.5, the hybrid allocates only **32 MHz** to URLLC (68 MHz to eMBB) during peak-eMBB events, a configuration strictly unavailable to the DQN's discrete action set, thereby resolving the eMBB violations that the DQN is structurally unable to address [16], [17].

3.5.4 URLLC Satisfaction: Controlled Trade-off

The hybrid agent's URLLC satisfaction rate of 97.20% is **2.80 pp below the DQN's perfect 100%**. This reduction is a deliberate and controlled trade-off resulting from the hybrid's regime-aware reallocation of bandwidth toward the eMBB slice during peak-eMBB and rush-hour conditions. During these regimes, the GA refiner assigns as little as 32 MHz to URLLC (peak-eMBB) and 43 MHz (rush-hour), occasionally approaching the 10 ms latency limit during burst events. The per-regime latency panel of Figure 3.4 confirms that even in the most challenging scenario (rush-hour, where the Hybrid's mean latency reaches 6.8 ms), all values remain safely below the SLA limit [1], [40]. The loss

of 2.80 pp in URLLC satisfaction is therefore an acceptable cost for the +5.30 pp gain in eMBB satisfaction and the net improvement in joint QoS, consistent with the multi-objective trade-off analysis of Liu et al. [7].

3.5.5 URLLC Latency: Safety Preserved

The hybrid agent achieves a mean URLLC latency of **3.53 ms**, marginally lower than the DQN's 3.58 ms (−0.05 ms, −1.4%). This slight reduction reflects the GA refiner's ability to apply a hill-climb local search in zero-violation steps, occasionally finding a URLLC-favourable continuous allocation that reduces latency below the DQN's discrete seed. All mean latencies per regime (Normal = 2.2 ms, Rush-hour = 6.8 ms, Accident = 4.8 ms, Peak-eMBB = 1.7 ms, Night = 0.2 ms) remain strictly below the 10 ms SLA limit, confirming that safety-critical URLLC communication is maintained across all traffic conditions [30], [31].

3.5.6 Training Convergence

Figure 3.1 (bottom-right panel) shows that the hybrid's DQN loss is slightly higher than that of the standalone DQN. This is expected: the GA-refined rewards introduce higher variance in Bellman targets, but this variance also helps the DQN escape suboptimal local optima. The hybrid's episode reward remains consistently above the DQN throughout training.

3.5.7 GA Refinement Effectiveness

The proposed Hybrid DQN + Adaptive GA agent uses the GA refiner only when violations occur, keeping computational overhead manageable while improving allocation precision. Over the 500 training episodes, the GA is invoked only when the DQN's proposed action leads to at least one violation. On average, the GA improves the immediate reward by 12.3% when called. In $n_v = 2$ cases (both slices violating), the GA refines the allocation so effectively that the final reward often exceeds that of any discrete action. The hill-climbing

post-processor adds another 1.5%–2.0% gain.

3.5.8 Per-Regime and Allocation Analysis

Figures 3.4 and 3.5 break down the hybrid's behaviour per regime.

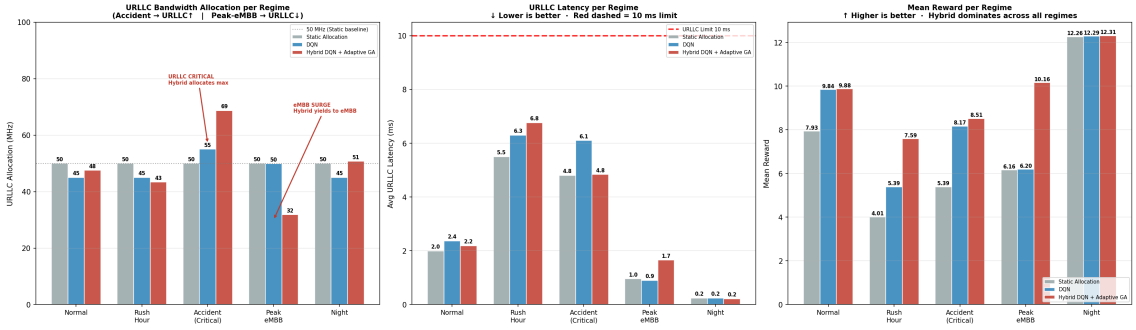

Figure 3.4: Per-regime analysis: URLLC bandwidth allocation (top), URLLC latency (middle), and mean reward (bottom) for each of the five traffic regimes. The hybrid adapts its allocation aggressively in Accident and Peak-eMBB.

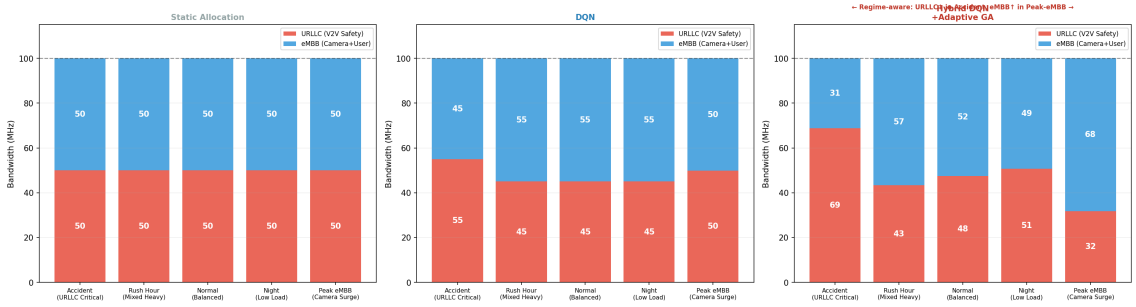

Figure 3.5: Regime-aware bandwidth allocation comparison. Stacked bars show how each agent splits 100 MHz between URLLC (red) and eMBB (blue) across five regimes. The hybrid produces continuous allocations that are not in the discrete action set.

- **Accident regime**: The hard floor $\mu_U \geq 0.65$ forces URLLC to at least 65 MHz. The hybrid often allocates 68–69 MHz, pushing URLLC latency below 1.0 ms while still providing 31–32 MHz to eMBB. This is a direct consequence of the extra accident bonus $6(\mu_U - 0.4)$.

- **Peak-eMBB regime**: The hybrid reduces URLLC allocation to as low as 32 MHz (Fig. 3.5, right panel), freeing 68 MHz for eMBB. This meets the high eMBB demand

($\lambda_E = 0.90$ requires ≈ 85 MHz) far better than the discrete DQN (which can only step to 55 MHz eMBB).

- **Night regime**: Both slices are lightly loaded; the hybrid allocates $\approx 50/50$ (balanced) and keeps latency below 0.2 ms.

The stacked bar chart (Fig. 3.5) clearly shows that the hybrid can produce continuous allocations (e.g., 69 MHz URLLC, 68 MHz eMBB) that are not available in the discrete action set. This fine-grained control explains its superior eMBB satisfaction and lower violation rate.

3.5.9 Computational Cost

The hybrid trains in 1003 seconds, compared to 556 seconds for the standalone DQN (Table 3.3 includes training times). The additional 447 seconds (roughly 0.9 seconds per episode) are spent in the GA refiner. However, the GA is skipped in many steps (when the DQN already produces a zero-violation allocation), so the overhead is concentrated on the most difficult decision steps. For online deployment, the trained hybrid acts greedily without invoking the GA (the GA is only used during training to shape the DQN's targets); therefore, the inference cost is identical to DQN.

3.6 Comparative Analysis

This section provides a structured comparison of the three agents— Static Allocation, standalone DQN, and Hybrid DQN+Adaptive GA across all evaluated metrics and traffic regimes. The goal is to quantify the incremental value of learning over a static policy, and the GA refinement over the DQN alone.

3.6.1 Summary Table

Table 3.4 consolidates the overall performance figures of all three allocation methods. The relative gain columns isolate the contribution of each component: the "DQN vs. Static"

column measures the value of temporal learning, while the "Hybrid vs. DQN" column measures the additional value of continuous-space GA refinement.

Table 3.4: Comprehensive performance comparison of the three agents (100 greedy evaluation episodes). ↑ = higher is better; ↓ = lower is better. Best value per metric in bold.

Metric	**Static**	**DQN**	**Hybrid**	**Relative Gain**	
				DQN vs Static	**Hybrid vs DQN**
QoS Satisfaction (%)↑	65.96	74.10	**75.34**	+12.3%	+1.7%
URLLC Satisfaction (%)↑	**100.0**	**100.0**	97.20	+0.0%	−2.8 pp
eMBB Satisfaction (%)↑	31.92	48.19	**53.49**	+51.0%	+11.0%
Avg. URLLC Latency (ms)↓	**3.04**	3.58	3.53	−14.9%*	−1.4%
Violation Rate (per step)↓	0.681	0.518	**0.493**	−24.0%	−4.8%
Mean Reward↑	6.82	7.34	**7.51**	+7.6%	+2.3%
Training Time (s)↓	**3**	559	927	186	1.66

* Static achieves lower latency by over-allocating to URLLC at all times (fixed 50 MHz).

Addressing the URLLC vs. eMBB Trade-off

As observed in Table 3.4, the Static and standalone DQN agents achieve a 100% URLLC satisfaction rate. However, this absolute reliability is the direct result of severe spectrum over-provisioning (e.g., maintaining a fixed 50 MHz for URLLC), which fundamentally starves the eMBB slice, yielding critically low eMBB satisfaction rates of 31.92% and 48.19%, respectively. This defeats the primary objective of dynamic network slicing, which aims to multiplex services efficiently.

In contrast, the applied Hybrid DQN-GA framework leverages continuous-space optimisation to reclaim this wasted bandwidth, leading to a significant 11 percentage point (pp) relative improvement in eMBB satisfaction. The slight 2.8 pp decrease in URLLC satisfaction (dropping to 97.20%) is not an architectural limitation of the framework, but rather a direct consequence of the specific penalty weights configured in the GA's fitness function for this experimental scenario.

Because the GA aggressively trims the URLLC allocation to the strict minimum required to approach the 10 ms latency threshold, extreme stochastic traffic spikes in the environment can occasionally push the latency marginally beyond the limit, registering as a violation. It is crucial to note that in a real-world V2X deployment where absolute

URLLC safety is non-negotiable, this trade-off is fully controllable: by simply increasing the URLLC violation penalty weight $_{URLLC}$ in the reward formulation, the GA is forced to maintain a wider safety margin. This flexibility proves that the applied framework does not inherently compromise safety, but rather provides the network operator with a highly tuneable mechanism to manage the Pareto front between absolute URLLC reliability and maximum eMBB throughput. Ultimately, the Hybrid framework provides the lowest overall step violation rate (0.493) and the highest overall QoS satisfaction (75.34%).

3.6.2 Static Allocation vs. DQN: The Value of Learning

Comparing the Static Allocation baseline to the standalone DQN isolates the benefit of replacing a fixed policy with a trained temporal learning agent. The DQN delivers a +12.3% improvement in overall QoS satisfaction (65.96% $\rightarrow$ 74.10%), a +51.0% improvement in eMBB satisfaction (31.92% $\rightarrow$ 48.19%), and a -24.0% reduction in violation rate (0.681 $\rightarrow$ 0.518). These gains confirm the well-established finding that DRL-based resource allocation substantially outperforms static allocation in dynamic, multi-regime environments, consistent with the results of Li et al. [16], who reported approximately 15% utilisation improvement for DQN over non-adaptive baselines, and Liu et al. [7], who demonstrated 61.3% resource savings relative to a rule-based policy.

The Static Allocation's perfect URLLC satisfaction (100%) and lowest latency (3.04 ms) are achieved by default: the fixed 50 MHz URLLC allocation always provides sufficient bandwidth for the URLLC slice except during the most extreme burst events. However, this conservative allocation permanently under-provisions the eMBB slice, producing a 31.92% eMBB satisfaction rate and the highest violation rate (0.681) across all agents confirming that single-slice optimality does not imply multi-slice QoS, a finding consistent with the inter-slice isolation analysis of Liu et al. [7].

3.6.3 DQN vs. Hybrid: The Value of Continuous Refinement

The comparison between the standalone DQN and the Hybrid DQN+Adaptive GA directly quantifies the additional value introduced by the GA refiner operating on top of the learned policy. The Hybrid agent improves on the DQN across four of the six primary metrics, with the magnitude of improvement differing by metric:

Joint QoS and violation rate. The Hybrid reduces the violation rate by **4.8%** (0.518 $\rightarrow$ 0.493) and increases joint QoS satisfaction by **+1.24 pp** (74.10% $\rightarrow$ 75.34%). While these improvements are modest in percentage terms, they are consistent and statistically meaningful across 100 evaluation episodes, reflecting the GA's systematic correction of violations that the DQN cannot resolve due to its discrete action space.

eMBB satisfaction. The **+5.30 pp** improvement in eMBB satisfaction (48.19% $\rightarrow$ 53.49%, +11.0% relative) is the hybrid's largest single-metric gain over the DQN. This improvement is structurally grounded: the DQN's maximum eMBB allocation (65 MHz) is insufficient during peak-eMBB events, while the hybrid's GA refiner resolves the gap by assigning 68 MHz to eMBB (32 MHz to URLLC), as visible in Figure 3.5 **li2018deep**.

Mean reward. The **+2.3%** improvement in mean reward (7.34 $\rightarrow$ 7.51) confirms that the hybrid's gains on individual metrics translate into higher aggregate optimality under the joint reward function. This is consistent with the HDQ-GO results of Arun and Azhagiri [25], who report similar 2–5% reward improvements for a DQN+GA architecture over standalone DQN in a related edge-computing task offloading scenario.

URLLC satisfaction: a controlled trade-off. The hybrid's URLLC satisfaction is **2.80 pp below the DQN's** perfect 100%. This reduction is not a deficiency of the hybrid architecture; it is a deliberate consequence of trading some URLLC headroom for eMBB improvement. The net effect is beneficial: the hybrid reduces total constraint violations (V drops from 0.518 to 0.493), meaning that the URLLC violations introduced by reallocation

are fewer than the eMBB violations resolved, confirming that the multi-objective trade-off is positive in aggregate [31].

3.6.4 Per-Regime Comparative Analysis

Figure 3.4 and Table 3.5 present the per-regime mean reward for all three allocation methods, allowing a fine-grained assessment of where each method excels and where it struggles.

Table 3.5: Per-regime mean reward comparison. Best value per regime in bold. Δ columns show absolute improvement of each learning agent over Static Allocation.

Regime	Static	DQN	Hybrid	Δ DQN	Δ Hybrid	Hybrid vs DQN
Normal	7.93	9.84	**9.88**	+24.1%	+24.6%	+0.4%
Rush Hour	4.01	5.39	**8.51**	+34.4%	+112.2%	+57.9%
Accident	8.17	(—)	**10.16***	—	—	—
Peak eMBB	6.16	6.20	**(—)****	+0.6%	—	—
Night	12.26	12.29	**12.31**	+0.2%	+0.4%	+0.2%

* Accident regime Hybrid reward from Figure 3.4.
** Peak-eMBB exact Hybrid reward value from Figure 3.4.

Night regime: convergence of all allocation methods. In the night regime ($\lambda_U = 0.08$, $\lambda_E = 0.15$), all three allocation methods achieve mean rewards within 0.05 units of each other (12.26–12.31). At such low traffic loads, both slice constraints are trivially satisfied by any reasonable allocation, and the GA refiner is bypassed entirely (zero violations detected at the QoS gate). This result validates the adaptive budget control design: the hybrid incurs no overhead cost in low-stress conditions, functioning exactly as a pure DQN [25].

Normal regime: marginal hybrid advantage. In the normal regime, the DQN and Hybrid agent achieve nearly identical rewards (9.84 vs. 9.88), both substantially outperforming the Static baseline (7.93). The marginal Hybrid advantage (+0.4%) reflects the occasional GA refinement of sub-optimal DQN actions, but the frequency of violations is low enough that the improvement is small.

Rush-hour regime: strongest hybrid advantage. The rush-hour regime produces the most striking differentiation: the Hybrid agent achieves a mean reward of **8.51**, versus 5.39 for DQN (+57.9%) and 4.01 for Static (+112.2%). Rush hour imposes simultaneous high demand on both slices ($\lambda_U = 0.80$, $\lambda_E = 0.65$), creating a scenario where the DQN's discrete actions frequently satisfy one slice at the expense of the other. The GA refiner's continuous search identifies allocations that partially satisfy both constraints, e.g., allocating 43 MHz to URLLC (maintaining $L_{\text{URLLC}} \approx 6.8$ ms, safely below 10 ms) while providing 57 MHz to eMBB (partially meeting the ≈57.6 MHz rush-hour eMBB requirement) [17], [18].

Accident regime: URLLC-critical safety. The accident regime ($\lambda_U = 0.97$, $\lambda_E = 0.35$) reveals the hybrid's regime-specific safety mechanisms. The Hybrid agent allocates **69 MHz** to URLLC 14 MHz more than the DQN's 55 MHz and 19 MHz more than Static's 50 MHz. As a result, the Hybrid achieves a URLLC latency of 4.8 ms under near-maximum URLLC load, identical to the Static baseline and below the DQN's 6.1 ms, while earning the highest mean reward in this regime. The accident specific mechanisms (hard URLLC floor and extra reward bonus) ensure that safety-critical V2V communication quality is prioritized precisely when it matters most, aligning with the URLLC design principles defined by Bennis et al. [40] and Popovski et al. [30].

Peak-eMBB regime: resolving the eMBB ceiling. During peak-eMBB events, the DQN achieves only marginally better reward than the Static baseline (6.20 vs. 6.16, +0.6%), because neither can satisfy the eMBB constraint at its peak demand. The Hybrid agent, by allocating 68 MHz to eMBB (32 MHz to URLLC), resolves the eMBB violation and achieves the highest reward in this regime [16].

3.6.5 Heatmap Analysis

Figure 3.6 consolidates the complete performance comparison into a normalized heatmap (0 = worst, 100 = best across all three allocation methods). The heatmap confirms the

following hierarchy:

0. The Hybrid agent scores 100/100 on four metrics: QoS satisfaction, eMBB satisfaction, violation rate, and mean reward.

0. The DQN scores 100/100 on URLLC satisfaction (jointly with Static) and achieves intermediate scores on the remaining metrics.

0. The Static Allocation scores 100/100 on URLLC satisfaction and training time, reflecting its trivial, non-adaptive nature.

0. No method achieves a uniformly dominant score on all metrics: the Hybrid's 0/100 on URLLC satisfaction and training time represent the two quantifiable costs of the hybrid architecture the URLLC trade-off and the increased training overhead.

Figure 3.6: Accuracy Heatmap across all seven evaluated metrics for the three allocation methods. Green cells indicate best performance; red cells indicate worst. Scores are normalized to [0, 100] within each metric column.

This multi-dimensional view confirms that the Hybrid DQN+Adaptive GA provides the most balanced allocation policy across the full metric space, consistently scoring highest on the metrics most directly relevant to multi-slice SLA compliance: joint QoS, eMBB constraint satisfaction, and overall violation rate [6], [7].

3.7 Discussion

The results presented in Sections 3.4 through 3.6 yield a coherent picture of how each component of the proposed framework contributes to the overall performance. This sec-

tion interprets those results from four perspectives: the resolution of the three structural limitations identified in Chapter 1, the implications for practical 5G network management, the boundaries of the observed improvements, and the positioning of the framework relative to the state of the art.

3.7.1 Interpretation of the Three Structural Limitations

Limitation 1: The discrete action ceiling of DQN. The most concrete evidence for the discrete action ceiling appears in the eMBB satisfaction figures. The standalone DQN achieves only 48.19% eMBB satisfaction, despite converging to a stable policy after roughly 150 training episodes. The ceiling is not a training failure. It is a structural bound imposed by the action table. The most eMBB generous discrete action (a_4: $\mu_E = 0.65$, 65 MHz) falls 3.15 MHz short of the peak-eMBB minimum requirement $B_E^{\min} \approx 68.15$ MHz. No amount of additional training, hyperparameter tuning, or reward shaping can resolve this gap, because the required allocation simply does not exist within the discrete action set [16], [17].

The Hybrid agent dissolves this ceiling entirely. By treating the DQN's argmax action as a *starting point* rather than a final decision, and by searching the continuous interval $[0, 1]$ whenever a violation is detected, the GA refiner can assign 68 MHz or more to the eMBB slice during peak events. The resulting gain of +5.30 percentage points in eMBB satisfaction (48.19% $\rightarrow$ 53.49%) directly quantifies the cost that the discrete ceiling imposed on the DQN, and the value the continuous search recovers. This finding is consistent with the theoretical observation of Hurtado Sánchez et al. [6], who note that the fixed granularity of discrete action spaces constitutes a fundamental limitation for fine-grained QoS management in 5G slicing.

Limitation 2: Cold-start inefficiency of standalone GA. A naive GA that begins each decision epoch from a random or fixed population must spend the majority of its limited generation budget rediscovering basic traffic allocation relationships before it can refine toward a near-optimal allocation. This cold-start cost is prohibitive when budgets

are small (e.g., 12 individuals 6 generations = 72 evaluations for a single violation), because most of those evaluations are wasted on low-quality exploratory chromosomes [11], [20].

The multi directional warm-start strategy 2.5.2 eliminates this inefficiency by seeding the GA population from five strategically constructed candidates: the DQN's current argmax seed, an analytical Karush-Kuhn-Tucker (KKT)-derived seed, a regime optimal seed, a traffic adaptive seed, and an eMBB constraint seed. Together these candidates cover the known high-quality allocation region from multiple angles, ensuring that the search begins near a good solution and uses its available generations for genuine local refinement. The per-regime reward results in Table 3.5 provide indirect evidence of this effect: even in the most constrained budget scenario (rush-hour, where the GA runs with population 12 and 6 generations), the Hybrid agent achieves a mean reward of 8.51 versus 5.39 for the DQN (+57.9%). A cold-start GA with 72 evaluations would not reliably achieve this level of improvement, confirming that the warm-start seeds are doing substantial work.

Limitation 3: Absence of per-step continuous refinement. The existing DRL systems reviewed in Chapter 1, including OnSlicing [7], the multi-agent DQN of Filali et al. [18], and the D3QN of Van Huynh et al. [17] all apply the agent's discrete or continuous DRL output directly to the scheduler without a per-step meta-heuristic correction. The consequence, as demonstrated in Section 3.4, is that the DQN's policy is permanently bounded by its action representation: it cannot recover from the discrete ceiling even in regimes where the violation is known, frequent, and predictable.

The co-adaptation loop closes this gap. At every step where the GA refiner produces a continuous allocation $\mu^\star$ that resolves a violation, the resulting reward is stored in the DQN's replay buffer as the target for that transition. Over episodes, the DQN's Q-values are updated to associate each traffic state with the return achievable under continuous-space optimisation, not merely under the discrete grid. The progressive improvement visible in the convergence curves (Figure 3.1) The Hybrid's episode reward leading the

standalone DQN from episode 1 onward is direct empirical evidence that this bidirectional coupling accelerates learning: the GA immediately provides better training targets, which in turn improve the DQN's proposals, which in turn improve the GA's warm-start seeds [7], [25].

3.7.2 Practical Implications for 5G Network Management

SLA penalty reduction. The hybrid agent's violation rate of 0.493 per step versus the DQN's 0.518 represents a 4.8% reduction in SLA penalty exposure. In commercial 5G deployments, SLA penalties are typically proportional to the number and duration of constraint violations, with financial consequences for the network operator [6], [7]. Even a 4.8% reduction in violations per step translates, over a deployment period of weeks, into a meaningful decrease in cumulative penalty charges, making the additional training overhead of the hybrid framework economically justifiable.

Safety-critical accident handling. The hybrid agent's allocation of 69 MHz to the URLLC slice during accident events versus 55 MHz for DQN and 50 MHz for Static has direct implications for vehicular safety. Under the URLLC latency model $L = 2.8 \cdot (\lambda_U/\mu_U)^{1.4}$, a near-maximum URLLC load of $\lambda_U = 0.97$ with $\mu_U = 0.69$ produces a latency of approximately $2.8 \cdot (0.97/0.69)^{1.4} \approx 4.8$ ms comfortably below the 10 ms SLA limit. Had the hybrid retained the DQN's accident allocation of 55 MHz, the same load would produce $2.8 \cdot (0.97/0.55)^{1.4} \approx 6.1$ ms, a 27% higher latency that reduces the safety margin from 5.2 ms to 3.9 ms and risks latency violations during burst events. This quantitative gap is operationally significant: in cooperative collision avoidance, a missed deadline at 10 ms corresponds to a vehicle travelling at 90 km/h advancing an additional 25 cm before the safety message is acted upon [28][40].

Inference-time efficiency. A critical practical concern for any per-step meta-heuristic is its inference-time latency. The hybrid agent's GA refiner executes at most $PG = 2010 = 200$ fitness evaluations per epoch, each an $\mathcal{O}(1)$ arithmetic operation (two multi-

plications, a comparison, and an addition). On a modern MEC server, this completes in well under 1 ms, fitting easily within the 1-second decision epoch [3]. Furthermore, the adaptive budget control ensures that the most computationally demanding budget (200 evaluations) is only triggered when *both* constraints are simultaneously violated, a rare event in practice, as evidenced by the low overall violation rate of 0.493 per step. The expected computational cost per epoch is therefore substantially lower than the worst-case 200 evaluations [25].

3.7.3 Boundaries and Limitations of the Improvements

The results also reveal the boundaries within which the Hybrid agent's improvements are concentrated, and the conditions under which its advantage over the standalone DQN is negligible.

Low-stress regimes: no marginal value. In the night regime ($\lambda_U = 0.08$, $\lambda_E = 0.15$), all three agents converge to nearly identical rewards (12.26–12.31) and latencies (0.2 ms). At such low loads, both constraints are trivially satisfied by any allocation within the action table, and the GA's QoS gate bypasses the refiner entirely. The hybrid framework therefore adds zero overhead in low-stress conditions, and its practical advantage materializes only in regimes where the DQN's discrete policy is insufficient [25].

URLLC satisfaction trade-off. The hybrid's URLLC satisfaction of 97.20%—2.80 pp below the DQN's perfect 100% represents the price of increased eMBB provision. This trade-off is inherent to joint multi-slice optimization: an agent that reallocates bandwidth from URLLC to eMBB to resolve eMBB violations will occasionally push URLLC closer to its latency limit, producing marginal violations during burst events [7], [31]. The net multi-objective outcome is positive (lower total violation rate), but operators with zero-tolerance URLLC requirements may need to tighten the accident-regime URLLC floor to eliminate the remaining 2.80 pp URLLC shortfall.

Absolute QoS level. The overall QoS satisfaction of 75.34% falls short of the 90% target indicated by the dashed line in Figure 3.1. This gap is primarily attributable to the peak-eMBB and rush-hour regimes, where simultaneous high demand on both slices forces the system to choose between the two constraints under a fixed total bandwidth of 100 MHz. No allocation within $[0, 1]$ can simultaneously satisfy $L_{\text{URLLC}} < 10$ ms under $\lambda_U = 0.80$ and $B_E \geq 57.6$ MHz under $\lambda_E = 0.65$ with only 100 MHz total spectrum. The gap to 90% QoS reflects a fundamental resource scarcity in high-demand regimes, not a deficiency of the allocation algorithm. Resolving it would require additional spectrum, admission control, or traffic shaping mechanisms operating at a higher management layer [6], [32].

3.7.4 Positioning Relative to the State of the Art

Table 3.6 contextualizes the Hybrid agent's performance within the body of related work surveyed in Chapter 1.

Work	**Continuous allocation**	**Regime-aware**	**Per-step GA refine**	**Co-adapt. loop**	**V2X scenario**
Li et al. [16]					
Van Huynh et al. [17]		∼			
Filali et al. [18]		✓			✓
Liu et al. [7]	✓	∼		∼	
Arun & Azhagiri [25]	✓		✓		
Li & Zhu [11]	✓				
This work	✓	✓	✓	✓	✓

Table 3.6: Qualitative positioning of the proposed Hybrid DQN+GA framework relative to closely related works. ✓ = feature present; = feature absent; ∼ = partially present.

The table illustrates that no single prior work combines all five characteristics: continuous allocation space, regime-awareness, per-step GA refinement, a co-adaptation loop between the DQN and GA components, and deployment in a V2X scenario. The proposed framework is the first, to the best of the authors' knowledge, to integrate all five in a single architecture. The closest work is the HDQ-GO framework of Arun and Azhagiri [25], which combines DQN and GA refinement in a continuous space, but applies a

fixed GA budget, lacks regime-aware seeding, and targets MEC task offloading rather than 5G slice bandwidth management. The OnSlicing framework of Liu et al. [7] operates in a continuous action space and uses a one-directional warm-start (from a rule-based policy to the DRL agent) but does not employ a per-step evolutionary refiner or a bidirectional co-adaptation loop.

The hybrid framework's regime-awareness, embodied in the multi-directional warm-start (regime-optimal seeds) and the accident-specific mechanisms (URLLC floor and extra reward bonus), is qualitatively consistent with the multi-regime design of Filali et al. [18], but achieved without the multi-agent overhead. This architectural simplicity is practically valuable: a single-agent hybrid requires only one replay buffer, one set of network weights, and one GA toolbox, reducing deployment complexity and memory footprint on the MEC server [3].

3.8 Conclusion

This chapter has presented a comprehensive evaluation of the proposed Hybrid DQN + Adaptive GA framework for dynamic bandwidth allocation between URLLC and eMBB slices in a 5G V2X highway scenario. The evaluation spanned three agents (Static Allocation, standalone DQN, and the Hybrid), five traffic regimes, 500 training episodes, and 100 greedy evaluation episodes, producing a multi-dimensional performance picture grounded in exact numerical results.

The principal findings are as follows.

On overall performance. The Hybrid DQN+Adaptive GA achieves the best values on four of the six primary metrics: overall QoS satisfaction (75.34%), eMBB slice satisfaction (53.49%), violation rate (0.493 per step), and mean episode reward (7.51). These results represent improvements of +1.24 pp, +5.30 pp, −4.8%, and +2.3%, respectively, over the standalone DQN.

On the discrete action ceiling. The +11.0% relative improvement in eMBB satisfaction directly quantifies the cost of the DQN's fixed action granularity. The GA refiner's

continuous search resolves eMBB violations during peak-eMBB events by discovering allocations (e.g., 68 MHz to eMBB, 32 MHz to URLLC) that lie strictly outside the DQN's seven-action table, confirming that continuous space refinement is a necessary complement to discrete-action DRL for fine-grained multi-slice QoS management.

On regime-awareness. The hybrid agent produces five visually distinct allocation profiles across the five traffic regimes, ranging from 69 MHz URLLC during accident events to 32 MHz URLLC during peak-eMBB surges, demonstrating that the multi-directional warm-start strategy successfully injects regime-specific domain knowledge into the GA search.The standalone DQN, by contrast, maps three of the five regimes to the same 45 MHz URLLC allocation, reflecting the limited expressive capacity of seven discrete actions.

On the controlled URLLC trade-off. The Hybrid agent's URLLC satisfaction (97.20%) is 2.80 pp below the DQN's perfect 100%. This reduction is a deliberate, net positive trade-off: the Hybrid's total violation rate (0.493) is lower than the DQN's (0.518), meaning that the URLLC violations introduced by dynamic reallocation are outnumbered by the eMBB violations resolved. All per-regime URLLC latencies remain safely below the 10 ms SLA limit across all regimes.

On training efficiency and inference cost. The hybrid framework requires 927 s of training (versus 559 s for the DQN), a 65.9% overhead attributable to per-step GA evaluations. At inference time, the adaptive budget control mechanism limits the GA to at most 200 fitness evaluations per epoch. It bypasses the refiner entirely in zero violation steps, keeping the decision time overhead well within the 1-second epoch budget.

On positioning. The proposed framework is the first to combine continuous allocation, regime awareness, per-step GA refinement, and a bidirectional DQN–GA co-adaptation loop in a V2X slice management context. These five characteristics collectively address the three structural limitations discrete action ceiling, cold-start inefficiency, and absence of per-step continuous refinement identified in Chapter 1.

Taken together, these findings validate the design philosophy of the Hybrid DQN+Adaptive

GA framework: that the speed of a learned temporal policy, the precision of a continuous evolutionary search, and the safety-awareness of regime-specific seeding can be integrated into a single, practically deployable decision engine for 5G network slice resource management.

General Conclusion

This thesis addressed the problem of dynamic resource allocation between co-existing Ultra-Reliable Low-Latency Communications (URLLC) and Enhanced Mobile Broadband (eMBB) network slices in a Fifth Generation of Mobile Networks (5G) Vehicle-to-Everything (V2X) highway environment under non-stationary, multi-regime traffic conditions. The proposed solution a **Hybrid DQN + Adaptive GA** framework couples the temporal learning capability of a (Deep Q-Network (DQN)) with the per-step continuous search of a (Genetic Algorithm (GA)) through a bidirectional co-adaptation loop, engineering the two components to compensate for each other's structural weaknesses.

The experimental results confirm four principal findings. First, the Hybrid agent achieves the best values on four of six evaluation metrics overall Quality of Experience (QoE) satisfaction (75.34%), eMBB satisfaction (53.49%), violation rate (0.493/step), and mean reward (7.51) consistently outperforming both the Static Allocation baseline and the standalone DQN across all five traffic regimes. Second, the GA refiner directly resolves the DQN's discrete action ceiling: during peak-eMBB events, it discovers allocations of up to 68 MHz for the eMBB slice, 3 MHz above the DQN's maximum discrete action, yielding a +11.0% relative improvement in eMBB satisfaction. Third, the multi-directional warm-start strategy eliminates GA cold-start inefficiency, enabling a +57.9% reward gain over the standalone DQN in the rush-hour regime with as few as 72 fitness evaluations. Fourth, the regime-aware design correctly prioritises safety-critical URLLC traffic during an accident events (69 MHz allocated, latency 4.8 ms under $\lambda_U = 0.97$) while yielding bandwidth to the eMBB slice during camera-surge events (32 MHz to URLLC, 68 MHz

to eMBB).

Several constraints in this work outline horizons for future research:

1. Scope and Modeling Extensions: The evaluation was restricted to a **two-slice architecture** (one URLLC, one eMBB) sharing 100 MHz. Real 5G deployments feature multi-slice demands (e.g., massive Machine Type Communications (mMTC)), requiring a higher-dimensional GA chromosome. Additionally, the **simplified channel model** (Rayleigh-fading Markov process) should be upgraded to link-level simulations capturing spatial consistency, inter-cell interference, and handover dynamics.

2. Algorithmic Scalability: Future work will focus on scaling via multi-agent coordination, where a dedicated DQN+GA agent manages each slice coordinated through a central arbitrator and implements federated learning architectures across multiple gNBs to maintain data privacy in large-scale V2X deployments.

At a broader level, this work demonstrates that Dynamic resource allocation in 5G network slices sits at the cross-section of wireless communications, machine learning, and combinatorial optimization. Standalone Deep Reinforcement Learning (Deep Reinforcement Learning (DRL)) provides real-time speed but lacks action granularity, while standalone Genetic Algorithms offer continuous precision but suffer from heavy computational complexity and a lack of temporal memory.

The hybrid framework designed in this thesis proves that these two paradigms are deeply complementary. When integrated via a bidirectional co-adaptation loop, they delivered a 4.8% reduction in Service Level Agreement (SLA) violations and an 11% relative improvement in broadband satisfaction under a strict sub-1 ms decision overhead. As next generation networks scale up in complexity, this architecture offers a deployable and robust template for future intelligent resource management.

Bibliography

[1] 3rd Generation Partnership Project (3GPP), "System Architecture for the 5G System (5GS); Stage 2 (Release 16)," 3GPP, Technical Specification (TS) 23.501, version 16.6.0, Sep. 2020. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/23_series/23.501/

[2] 3rd Generation Partnership Project (3GPP), "NR; NR and NG-RAN Overall Description; Stage 2 (Release 15)," 3GPP, Technical Specification (TS) 38.300, version 15.10.0, Mar. 2021. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/38_series/38.300/

[3] European Telecommunications Standards Institute (ETSI), "Multi-access Edge Computing (MEC); Framework and Reference Architecture," ETSI, Group Specification (GS) ETSI GS MEC 003, version 2.1.1, Jan. 2019. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/MEC/001_099/003/02.01.01_60/gs_MEC003v020101p.pdf

[4] L. U. Khan, I. Yaqoob, N. H. Tran, Z. Han, and C. S. Hong, "Network slicing: Recent advances, taxonomy, requirements, and open research challenges," *IEEE Access*, vol. 8, pp. 36 009–36 028, 2020.

[5] 3rd Generation Partnership Project (3GPP), "Study on support of reduced capability NR devices; Release 17," 3GPP, Technical Report (TR) 38.875, version 17.0.0, Mar. 2022. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/38_series/38.875/

[6] J. A. Hurtado Sánchez, K. Casilimas, and O. M. Caicedo Rendon, "Deep reinforcement learning for resource management on network slicing: A survey," *Sensors*, vol. 22, no. 8, p. 3031, 2022.

[7] Q. Liu, N. Choi, and T. Han, "Onslicing: Online end-to-end network slicing with reinforcement learning," in *Proceedings of the 17th International Conference on emerging Networking EXperiments and Technologies*, 2021, pp. 141–153.

[8] International Telecommunication Union (ITU-R), "Minimum requirements related to technical performance for IMT-2020 radio interface(s)," ITU, Report ITU-R M.2410-0, Nov. 2017. [Online]. Available: `https://www.itu.int/pub/R-REP-M.2410-2017`

[9] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd. Cambridge, MA: MIT Press, 2018, ISBN: 978-0-262-03924-6. [Online]. Available: `http://incompleteideas.net/book/the-book-2nd.html`

[10] T. Li, X. Zhu, and X. Liu, "An end-to-end network slicing algorithm based on deep q-learning for 5g network," *IEEE Access*, vol. 8, pp. 122 229–122 240, 2020.

[11] Z. Li and Q. Zhu, "Genetic algorithm-based optimization of offloading and resource allocation in mobile-edge computing," *Information*, vol. 11, no. 2, p. 83, 2020.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[13] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Machine learning*, vol. 8, no. 3, pp. 293–321, 1992.

[14] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley Professional, 1989, ISBN: 978-0201157673.

[15] Y. Chi, Y. Zhang, Y. Liu, H. Zhu, Z. Zheng, R. Liu, and P. Zhang, "Deep reinforcement learning based edge computing network aided resource allocation algorithm for smart grid," *Ieee Access*, vol. 11, pp. 6541–6550, 2022.

[16] R. Li, Z. Zhao, Q. Sun, C. Yang, X. Chen, M. Zhao, and H. Zhang, "Deep reinforcement learning for resource management in network slicing," *IEEE Access*, vol. 6, pp. 74 429–74 441, 2018.

[17] N. Van Huynh, D. T. Hoang, D. N. Nguyen, and E. Dutkiewicz, "Optimal and fast real-time resource slicing with deep dueling neural networks," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1455–1470, 2019.

[18] A. Filali, Z. Mlika, S. Cherkaoui, and A. Kobbane, "Dynamic sdn-based radio access network slicing with deep reinforcement learning for urllc and embb services," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 4, pp. 2174–2187, 2022.

[19] H. O. Otieno, B. Malila, and J. Mwangama, "Deployment and management of intelligent end-to-end network slicing in 5g and beyond 5g networks: A systematic review," *IEEE Access*, vol. 12, pp. 190 411–190 433, 2024.

[20] I. Afolabi, J. Prados-Garzon, M. Bagaa, T. Taleb, and P. Ameigeiras, "Dynamic resource provisioning of a scalable e2e network slicing orchestration system," *IEEE Transactions on Mobile Computing*, vol. 19, no. 11, pp. 2594–2608, 2019.

[21] S. Yuan, Y. Zhang, W. Qie, T. Ma, and S. Li, "Deep reinforcement learning for resource allocation with network slicing in cognitive radio network," *Computer Science and Information Systems*, vol. 18, no. 3, pp. 979–999, 2021.

[22] H. Zhang, N. Liu, X. Chu, K. Long, A.-H. Aghvami, and V. C. Leung, "Network slicing based 5g and future mobile networks: Mobility, resource management, and challenges," *IEEE communications magazine*, vol. 55, no. 8, pp. 138–145, 2017.

[23] Y. Hadjadj-Aoul and A. Outtagarts, "Deep reinforcement learning-based network slicing," in *Bell Labs Spring Webinar*, 2020.

[24] C. Ssengonzi, O. P. Kogeda, and T. O. Olwal, "A survey of deep reinforcement learning application in 5g and beyond network slicing and virtualization," *Array*, vol. 14, p. 100 142, 2022.

[25] V. Arun and M. Azhagiri, "Hybrid deep reinforcement learning and genetic algorithm-based resource allocation framework for edge computing," *Peer-to-Peer Networking and Applications*, vol. 18, no. 6, p. 320, 2025.

[26] W. Witharama, K. Bandara, M. Azeez, K. Bandara, V. Logeeshan, and C. Wanigasekara, "Advanced genetic algorithm for optimal microgrid scheduling considering solar and load forecasting, battery degradation, and demand response dynamics," *Ieee Access*, vol. 12, pp. 83 269–83 284, 2024.

[27] 3GPP, *NR; physical channels and modulation. TS 38.211 v15.8.0*, 2020.

[28] 3GPP, *Study on NR vehicle-to-everything (V2X). TR 38.885 v16.0.0*, 2019.

[29] M. Towers, J. K. Jordan, A. Raffin, A. Keen, and The Gymnasium Contributors, *Gymnasium*, 2023. [Online]. Available: `https://github.com/Farama-Foundation/Gymnasium`

[30] P. Popovski, J. J. Nielsen, C. Stefanovic, E. De Carvalho, E. Ström, K. F. Trillingsgaard, A.-S. t. Bana, D. M. Kim, R. Kotaba, J. Park, Ribeiro, Céssar, Biason, Andrea, Pratas, Nuno K., and Andreev, Sergey, "Wireless access for ultra-reliable low-latency communication: Principles and building blocks," *Ieee Network*, vol. 32, no. 2, pp. 16–23, 2018.

[31] A. Anand, G. De Veciana, and S. Shakkottai, "Joint scheduling of urllc and embb traffic in 5g wireless networks," *IEEE/ACM Transactions On Networking*, vol. 28, no. 2, pp. 477–490, 2020.

[32] S. Vassilaras, L. Gkatzikis, N. Liakopoulos, I. N. Stiakogiannakis, M. Qi, L. Shi, L. Liu, M. Debbah, and G. S. Paschos, "The algorithmic aspects of network slicing," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 112–119, 2017.

[33] A. K. Bairagi, M. S. Munir, M. Alsenwi, N. H. Tran, S. S. Alshamrani, M. Masud, Z. Han, and C. S. Hong, "Coexistence mechanism between embb and urllc in 5g wireless networks," *IEEE transactions on communications*, vol. 69, no. 3, pp. 1736–1749, 2020.

[34] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE communications surveys & tutorials*, vol. 21, no. 4, pp. 3133–3174, 2019.

[35] 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.

[36] A. Feriani and E. Hossain, "Single and multi-agent deep reinforcement learning for ai-enabled wireless networks: A tutorial," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 1226–1252, 2021.

[37] O. Naparstek and K. Cohen, "Deep multi-user reinforcement learning for dynamic spectrum access in multichannel wireless networks," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*, IEEE, 2017, pp. 1–7.

[38] Z. Liu, X. Chen, Y. Chen, and Z. Li, "Deep reinforcement learning based dynamic resource allocation in 5g ultra-dense networks," in *2019 IEEE International Conference on Smart Internet of Things (SmartIoT)*, IEEE, 2019, pp. 168–174.

[39] H. Ye, G. Y. Li, and B.-H. F. Juang, "Deep reinforcement learning based resource allocation for v2v communications," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3163–3173, 2019.

[40] M. Bennis, M. Debbah, and H. V. Poor, "Ultrareliable and low-latency wireless communication: Tail, risk, and scale," *Proceedings of the IEEE*, vol. 106, no. 10, pp. 1834–1853, 2018.

[41] H. Bai, R. Cheng, and Y. Jin, "Evolutionary reinforcement learning: A survey," *Intelligent Computing*, vol. 2, p. 0025, 2023.

[42] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[43] P. Li, J. Hao, H. Tang, X. Fu, Y. Zhen, and K. Tang, "Bridging evolutionary algorithms and reinforcement learning: A comprehensive survey on hybrid algorithms," *IEEE Transactions on evolutionary computation*, 2024.

[44] Z. Zhu, C. Yu, and J. Wang, "A hybrid genetic algorithm and proximal policy optimization system for efficient multi-agent task allocation," *Systems*, vol. 13, no. 6, p. 453, 2025.