



PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
الجمهورية الجزائرية الديمقراطية الشعبية
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
وزارة التعليم العالي والبحث العلمي
FACULTY OF SCIENCES
DEPARTMENT OF COMPUTER SCIENCES

A Thesis

Presented for obtaining the degree of:

DOCTORATE

In: Computer Science

Specialty: Software Engineering

By:

Mme. Fatima-Zohra BEMRAH

Theme

Towards Adaptive MADRL Approach For Optimization and Resilience in IoT Environments

Thesis defended on July 15, 2026 at Faculty of Science of the University of Tlemcen
in Front of the Jury Composed of:

Mr Abdelkrim BENAMAR	Full Professor	University of Tlemcen	President
Mr Azzedine CHIKH	Full Professor	University of Tlemcen	Supervisor
Mr Samir OUCHANI	Research Director	CESI Lineact, Aix-en-Provence	Co-Supervisor
Mr Mimoun MALKI	Full Professor	ESI of Sidi Bel Abbes	Examiner
Mr Sidi Mohamed BENSLIMANE	Full Professor	ESI of Sidi Bel Abbes	Examiner
Mme Yassamine SELADJI	Full Professor	University of Tlemcen	Examiner

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

﴿ يَرْفَعُ اللَّهُ الَّذِينَ آمَنُوا مِنْكُمْ وَالَّذِينَ أُوتُوا الْعِلْمَ دَرَجَاتٍ ﴾

صَدَقَ اللَّهُ الْعَظِيمُ
سُورَةُ الْمُجَادَلَةِ، الْآيَةُ ١١

﴿ رَبِّ أَوْزِعْنِي أَنْ أَشْكُرَ نِعْمَتَكَ الَّتِي أَنْعَمْتَ عَلَيَّ وَعَلَى
وَالِدَيَّ وَأَنْ أَعْمَلَ صَالِحًا تَرْضَاهُ وَأَدْخِلْنِي بِرَحْمَتِكَ فِي عِبَادِكَ
الصَّالِحِينَ ﴾

صَدَقَ اللَّهُ الْعَظِيمُ
سُورَةُ النَّحْلِ، الْآيَةُ ١٩

الْحَمْدُ لِلَّهِ الَّذِي بِفَضْلِهِ تَتِمُّ الصَّالِحَاتُ

Abstract

IoT systems face critical challenges in managing services efficiently while preserving resilience under dynamic environments. Most existing management approaches rely on static policies or centralized control, which limits scalability, adaptability, and robustness in large-scale IoT deployments. To address these limitations, we propose the MADRL framework, which integrates a hierarchical multi-agent architecture with deep reinforcement learning (DRL). MADRL enables adaptive decision-making, modular service composition, resilience and energy optimization. MADRL framework orchestrates elementary IoT services dynamically into higher-level service behaviors. The hierarchical design decomposes the management problem into coordinated decision layers, improving scalability and tractability. At the high level, a global agent optimizes long-term system objectives, including resilience, energy efficiency, and system stability. The mid-level refines these objectives by coordinating atomic service and adapting QoS constraints to contextual changes, while the low level ensures fine-grained, real-time control through direct interaction with devices and services. Agents are trained using a centralized training–decentralized execution paradigm, enabling coordinated learning while preserving distributed autonomy. We evaluate our approach using a realistic smart-building case study with up to 500 agents. Our analysis shows a environmental comfort and resilience ensuring, QoS satisfaction of 95.3%, increasing reward with fast convergence and 40% reduction in environmental energy consumption compared to DRL baselines.

keywords: Internet of Things (IoT), Multi-Agent Systems, Deep Reinforcement Learning, Hierarchical Architecture, Service Composition, Quality of Service (QoS), Energy Efficiency, Resilience.

Résumé

Les systèmes IoT évoluent dans des environnements dynamiques, hétérogènes et contraints en ressources, rendant la gestion des services complexe. Les approches existantes, souvent centralisées ou statiques, peinent à assurer simultanément la QoS, l'efficacité énergétique et la résilience, tout en garantissant scalabilité, adaptabilité et robustesse. Afin de surmonter ces limitations, nous proposons le framework MADRL, qui intègre une architecture multi-agents hiérarchique avec l'apprentissage par renforcement profond (Deep Reinforcement Learning, DRL). MADRL permet une prise de décision adaptative, une composition modulaire des services, ainsi qu'une optimisation de la résilience et de la consommation énergétique. Le framework MADRL orchestre dynamiquement des services IoT élémentaires en comportements de services de plus haut niveau. La conception hiérarchique permet de décomposer le problème de gestion en couches décisionnelles coordonnées, améliorant ainsi la scalabilité et la tractabilité du système. Au niveau supérieur, un agent global optimise les objectifs à long terme du système, notamment la résilience, l'efficacité énergétique et la stabilité globale. Le niveau intermédiaire affine ces objectifs en coordonnant les services atomiques et en adaptant les contraintes de QoS en fonction du contexte, tandis que le niveau inférieur assure un contrôle fin et en temps réel grâce à une interaction directe avec les dispositifs et les services. Les agents sont entraînés selon le paradigme CTDE, assurant une coordination efficace tout en maintenant l'autonomie distribuée, et l'approche est validée sur un cas de bâtiment intelligent impliquant jusqu'à 500 agents. Les résultats montrent que notre approche assure le confort environnemental et la résilience du système, avec un taux de satisfaction de la QoS atteignant 95,3 %. Elle présente également une convergence rapide avec une augmentation stable de la récompense, ainsi qu'une réduction de 40 % de la consommation énergétique environnementale par rapport aux approches DRL de référence.

mots-clés: Internet des Objets (IoT), systèmes multi-agents, apprentissage par renforcement profond, architecture hiérarchique, composition de services, qualité de service (QoS), efficacité énergétique, résilience.

ملخص

تواجه أنظمة إنترنت الأشياء تحديات كبيرة في إدارة الخدمات بكفاءة مع الحفاظ على المرونة في بيئات ديناميكية. تعتمد معظم أساليب الإدارة الحالية على سياسات ثابتة أو تحكم مركزي، مما يحد من قابلية التوسع، والقدرة على التكيف، والموثوقية في عمليات النشر واسعة النطاق. ولعلاج هذه القيود، نقترح إطاراً باضغى الذي يدمج بنية متعددة الوكلاء هرمية مع التعلم العميق بالتعزيز (ضغى). يتيح هذا الإطار اتخاذ قرارات تكيفية، وتركيباً معيارياً للخدمات، بالإضافة إلى تحسين المرونة وكفاءة استهلاك الطاقة. يقوم إطار بتنسيق الخدمات الأولية لإنترنت الأشياء بشكل ديناميكي لتشكيل سلوكيات خدمية ذات مستوى أعلى. ويسمح التصميم الهرمي بتقسيم مشكلة الإدارة إلى طبقات قرار منسقة، مما يحسن من قابلية التوسع وقابلية المعالجة. على المستوى الأعلى، يعمل وكيل عالمي على تحسين الأهداف طويلة المدى للنظام، بما في ذلك المرونة، وكفاءة الطاقة، واستقرار النظام. أما المستوى المتوسط، فيقوم بتحسين هذه الأهداف من خلال تنسيق الخدمات الأولية وتكييف قيود جودة الخدمة وفقاً للتغيرات السياقية، في حين يضمن المستوى الأدنى تحكماً دقيقاً وفي الوقت الحقيقي من خلال التفاعل المباشر مع الأجهزة والخدمات. يتم تدريب الوكلاء وفق نموذج التدريب المركزي والتنفيذ اللامركزي، مما يتيح تعلماً منسقاً مع الحفاظ على استقلالية الوكلاء. وقد تم تقييم هذا النهج باستخدام دراسة حالة واقعية لبنى ذكي تضم ما يصل إلى ٥٠٠ وكيل. تُظهر النتائج تحقيق راحة بيئية وضمان المرونة، مع معدل رضا عن جودة الخدمة يبلغ ٣٠.٩٥٪، إضافة إلى تقارب سريع ومستقر في التعلم، وتقليل بنسبة ٤٠٪ في استهلاك الطاقة مقارنة بأساليب التعلم العميق بالتعزيز المرجعية.

الكلمات المفتاحية: إنترنت الأشياء، الأنظمة متعددة الوكلاء، التعلم العميق بالتعزيز، البنية الهرمية، تركيب الخدمات، جودة الخدمة، كفاءة الطاقة، المرونة.

To my husband, Youcef
who stood beside me through every long day and every late night.
Your patience, your kindness, and your quiet strength carried me when my own strength
was not enough.
This work is as much yours as it is mine.
To my son Akram and my daughter Alaa,
my two greatest joys,
You filled even the hardest days with laughter and light.
May you read these pages one day
and know that every effort was, in its way for you
To my parents,
who taught me to love learning from my very first steps.
Everything good in me began with you.
Thank you for your endless love, your Douâa, and your belief in me.
To my sisters,
who share this long experience with me.
Your presence made the hard days very easy.
To my second parents,
who welcome me as one of their own
and gave me a second family to lean on.
Your care and support meant more than words can say.
To my supervisors,
who guided me with patience and wisdom
and trusted me to find my own path.
You showed me what it truly means to be a researcher.
And to every person who offered a kind word
when I need it most,
thank you. This thesis carries a piece of each of you.

Acknowledgments

I would like to express my deepest gratitude to all who supported and accompanied me throughout this doctoral journey. This thesis is the result of years of effort, and it would not have been possible without the help and kindness of many people.

My deepest thanks go to my supervisor, **Prof. Azeddine CHIKH**. Thank you for your guidance, your support during this research. Your advice and your encouragements meant a great deal to me, and I am thankful for your availability and kindness, and the trust you placed in me throughout these years. I have learned so much from you, not only about research, but also about what it means to work with care and dedication.

I am equally and deeply grateful to my co-supervisor, **Prof. Samir OUCHANI**. Thank you for guidance, your patience, and your constant support in every stage of this research. Your advice and your knowledge helped me find my way whenever I felt lost. You were always there when I needed help, and your close involvement at every step gave me strength and confidence. I am truly grateful for trusting me and for the freedom you gave me to develop my own ideas.

I would also like to thank the members of the jury. Thank you for honor of evaluating my work, for the time you devoted to reading it, and for your valuable comments and suggestions. Your remarks helped me see my work with new eyes and make it better.

Finally, my heartfelt thanks go to my family. To my husband for your patience and support. To the light of my life Akram and Alaa. To my parents and my second parents, thank you for your constant care and encouragements.

Contents

Acknowledgments	1
1 Introduction	9
1.1 Context	9
1.2 Motivation	10
1.2.1 The Complexity of Large-Scale IoT Environments	10
1.2.2 Limitations of Existing Approaches	10
1.2.3 The Promise of MAS and DRL Integration	11
1.3 Problem Statement	11
1.4 Research Questions	12
1.5 Proposed Approach	12
1.6 Contributions of Thesis	13
1.7 Scope and Delimitations	14
1.8 Thesis Structure	14
2 State of Art	15
2.1 Introduction	15
2.2 Background	15
2.2.1 IoT System Architecture	16
2.2.2 Service Management	16
2.2.3 Key Terminology & Concepts	17
2.2.4 Multi-Agent System Architecture	18
2.2.5 Deep Reinforcement Learning	21
2.2.6 Multi Agent Deep Reinforcement Learning	23
2.2.7 Graph Attention Networks	26
2.2.8 The Unsupervised Learning Paradigm	29
2.2.9 Multi-Objectives Optimization	31
2.2.10 Formal Methods in Architecture Modelling	33
2.3 Service Management in IoT	36
2.3.1 Optimization-Based Approaches	36
2.3.2 AI-Driven and Learning-Based Approaches	38
2.3.3 Decentralized and Blockchain-Based Approaches	39
2.3.4 Comparison & Discussion	40
2.4 Multi-Agent Deep Reinforcement Learning for IoT	41
2.4.1 Domain-Specific Applications of MADRL in IoT	42
2.4.2 Learning Approaches in MADRL for IoT	44
2.4.3 Communication and Coordination Strategies in MADRL	46
2.4.4 Comparison & Discussion	49

2.5	MAS-DRL for IoT Service Composition	49
2.5.1	Service Composition with MAS and DRL	49
2.5.2	MADRL and GAT-based Service Composition	50
2.5.3	MADRL Service Optimization with MO and GAT	51
2.5.4	Comparison and Discussion	51
2.6	Conclusion	53
3	IoT System Modeling and Conceptualization	55
3.1	Introduction	55
3.2	IoT System Agent-based Architecture	55
3.3	IoT Environment Modeling	56
3.3.1	The Physical Layer	56
3.3.2	The Edge-Fog Layer	57
3.3.3	The Network Layer	58
3.3.4	The Platform Layer	58
3.3.5	The Application Layer	60
3.4	Smart City use case	60
3.4.1	Smart Building Devices	61
3.4.2	Smart Building Fog and Edge Nodes	62
3.4.3	Smart Building Services	62
3.4.4	Smart Building Communication links	63
3.5	Conclusion	64
4	Hierarchical and Graph-Attention MADRL Frameworks for IoT Service Optimization	65
4.1	Introduction	65
4.2	Hierarchical Architecture Overview	66
4.3	High-Level Coordination	67
4.3.1	Global Objectives Optimization	68
4.3.2	Coo-MATD3 Learning for Coordination	70
4.4	Middle-Level Optimization	71
4.4.1	Context-aware Service Discovery	72
4.4.2	QoS-based Service Selection	77
4.4.3	Optimization of Service Composition	80
4.4.4	Complexity Analysis	82
4.5	Low-Level Interaction	84
4.6	Conclusion	87
5	Formal Verification & Experimental Validation	88
5.1	Introduction	88
5.2	IoT System Architecture Verification	88
5.2.1	Event-B Architecture and Modeling Approach	89
5.2.2	The static domain: contexts	90
5.2.3	The behavioral model: the refinement chain	90
5.2.4	Safety Properties	91
5.2.5	Verification by proof obligations	91
5.2.6	Discussion and Conclusion	93
5.3	Environment Size Reduction (GAMMA) Verification	94
5.3.1	Event-B Architecture and Refinement Strategy	94

5.3.2	Mathematical Modeling of ESR	95
5.3.3	Invariants and Refinement Soundness	96
5.3.4	Proof-Based Verification	96
5.3.5	Model Checking with ProB	98
5.3.6	Discussion and Conclusion	100
5.4	MDP Reduction Verification (HiMAD)	101
5.4.1	Verification Architecture	101
5.4.2	Verification Process	103
5.4.3	Discussion and Conclusion	105
5.5	Experimental Validation of HiMAD	106
5.5.1	Simulation Environment and Parameters	107
5.5.2	Hi-MAD Framework Evaluation	109
5.5.3	DRL Models Performance Evaluation	115
5.5.4	Statistical Validation and Discussion	116
5.6	Experimental Validation of GAMMA	118
5.6.1	Experimental Settings	118
5.6.2	GAMMA Framework Evaluation	118
5.6.3	Experimental results discussion	125
5.7	Conclusion	126
6	Conclusion and Future Work	127
6.1	Summary of Contributions	127
6.2	Limitations and Practical Considerations	128
6.3	Future Work	129

List of Algorithms

1	Coo-MATD3 for Coordination	71
2	MAPPO-Att+KMeans for Context Discovery	76
3	MAPPO-QoS+MO Selection.	80
4	MADDPG-GNN for Service Composition	83
5	DDQN for Low-Level Agent Learning with Shared Reward	87

List of Figures

2.1	IoT System Architecture	17
2.2	Multi-Agent System Architecture.	19
2.3	Deep Reinforcement Learning loop.	22
2.4	Multi-Agent Deep Reinforcement Learning.	24
2.5	From graph convolution to graph attention.	27
2.6	Architecture overview of a Graph Attention Network layer.	28
2.7	Overview of the clustering task.	29
2.8	Overview of a bi-objective minimisation problem in objective space. . .	31
2.9	Structural foundations of an Event-B model.	34
2.10	Semantics of a single event and its associated proof obligation.	34
2.11	Stepwise refinement as a correct-by-construction architectural pipeline. .	35
2.12	Taxonomy of Service Management Approaches and Techniques in IoT Sys- tems.	37
2.13	Taxonomy of MADRL in IoT Systems	42
3.1	MAS-based IoT System Architecture.	57
3.2	Smart Building	61
4.1	Hierarchical Graph Approach Overview.	67
4.2	The GAMMA Three-Stages Architecture	72
4.3	MDP Reduction Steps.	73
5.1	Event-B refinement architecture of the Hi-MAD system model.	89
5.2	Proof Obligations Discharged	92
5.3	Proof Obligation in Rodin.	93
5.4	Event-B refinement architecture of the Environment Size Reduction (ESR) process	94
5.5	ESR Proof Obligations	97
5.6	Proof Obligations in Rodin.	98
5.7	ProB Visualization	98
5.8	ProB Visualization	99
5.9	ProB Visualization	99
5.10	ProB Visualization	100
5.11	Event-B Architecture of MDP Reduction	102
5.12	Proof Obligations in Rodin.	105
5.13	ProB Visualization	106
5.14	Environmental Sensor State	110
5.15	Quality of Service Evaluation.	111
5.16	Correct vs Incorrect Selections per Episode.	111
5.17	Energy Use Minimization.	114

5.18	Energy Use Baseline	114
5.19	Mid-Agent Stability	114
5.20	Policy Stability over Time	114
5.21	(CD/QS)-MAPPO Reward vs Other DRL Models	115
5.22	Co-MADDPG Reward vs Other DRL Models	115
5.23	Coo-MATD3 Evaluation vs Other DRL Models.	116
5.24	DDQN Reward Performance over Time.	116
5.25	Service Distribution after to the context discovery step	119
5.26	QoS hard-constraints distributions of the QoS-based sub-dataset ($V_S, n=1085$)	120
5.27	QoS Satisfaction of hard and soft constraints	121
5.28	QoS Satisfaction Comparison	121
5.29	Mean normalized utility $\overline{f_j(q(j))}$	122
5.30	Success Rate of Service Composition	123
5.31	Energy Consumption	124
5.32	Reward Evolution and Convergence	125
6.1	Hierarchical-Graph Control Flow	128

List of Tables

1.1	Summary of Contributions	13
2.1	The three primary paradigms of clustering.	30
2.2	Solution strategies grouped by the moment of preference articulation. . .	32
2.3	The two structural components of an Event-B model.	33
2.4	Comparative Analysis of Service Management Approaches in IoT Systems	41
2.5	Comparative Analysis of IoT Multi-Agent DRL Applications and Ap- proaches	48
2.6	Comparison of related works on MAS-based IoT services across key criteria.	53
4.1	Computational profile of GAMMA on QWS2.	84
5.1	Domain safety properties of the Hi-MAD model.	91
5.2	Summary of hyperparameters for DRL algorithms used in training. . . .	108
5.3	Environmental Performance Criteria for Comfort Stability Experiment. .	110
5.4	MDP Reduction via Context and QoS-Based Agents Filtering.	112
5.5	Energy consumption before and after Context and QoS-based filtering. .	113
5.6	Filtering-induced size and energy reduction.	123
1	Experimental settings for MAPPO-Context discovery on QWS2.	130
2	Experimental settings for MAPPO-QoS+MO on QWS2.	131
3	Experimental settings for MADDPG-GNN composition on QWS2.	132

Chapter 1

Introduction

1.1 Context

The rapid proliferation of networked embedded devices over the past two decades has given rise to the **Internet of Things (IoT)** — a paradigm that extends computing into the physical world by interconnecting billions of heterogeneous devices across homes, industries, transportation systems, and healthcare infrastructures. As of 2024, global IoT device deployments have surpassed fifteen billion active endpoints, with projections indicating this figure will exceed thirty billion by 2030 (Ericsson Mobility Report, 2023). This exponential growth is not merely a quantitative expansion; it represents a qualitative transformation in the nature of computing itself, shifting the locus of intelligence away from centralized data centers and toward distributed, context-sensitive, and resource-constrained environments.

IoT systems are canonically organized according to a **multi-layer architecture** consisting of three principal strata. The perception layer encompasses the physical devices — sensors, actuators, embedded controllers, and smart objects — responsible for gathering raw data from the environment and enacting control actions. The network layer provides the communication fabric through which device-generated data is transmitted, including protocols such as MQTT, CoAP, Zigbee, and 5G NR. The application layer hosts the intelligent services and decision logic that interpret data streams and deliver value to end users. Across all three layers, effective coordination is predicated on the seamless exchange of heterogeneous data, timely response to environmental changes, and compliance with stringent **Quality of Service (QoS)** requirements encompassing latency, reliability, throughput, and energy consumption.

A central preoccupation within this architectural context is **IoT service management** — the orchestration, coordination, and optimization of distributed services deployed across devices, edge nodes, and cloud infrastructures. Rooted conceptually in Service-Oriented Computing (SOC), service management promotes modular, interoperable design through the dynamic composition of loosely coupled service components. Yet, as IoT deployments scale to millions of nodes operating under continuously varying conditions — device mobility, intermittent connectivity, fluctuating workloads, and adversarial failures — the traditional assumptions of SOC break down. Static service compositions cannot account for real-time variability; centralized orchestration introduces latency and single points of failure; and predefined rule-based policies lack the adaptability required by open, non-stationary environments.

These tensions have motivated a shift toward **intelligent and adaptive service**

management frameworks capable of autonomously coordinating services, optimizing resource utilization, and sustaining reliable operation under uncertainty. Two complementary paradigms have emerged as particularly promising in this regard: **Multi-Agent Systems (MAS)**, which model the distributed decision-making structure of IoT systems as collections of autonomous, interacting agents; and **Deep Reinforcement Learning (DRL)**, which provides agents with the capacity to learn optimal decision policies through interaction with a dynamic environment. The integration of these two paradigms — yielding **Multi-Agent Deep Reinforcement Learning (MADRL)** — constitutes a powerful and increasingly studied approach to the challenges of large-scale IoT management. This thesis investigates the design, formalization, and evaluation of such an integrated framework.

1.2 Motivation

1.2.1 The Complexity of Large-Scale IoT Environments

IoT environments are inherently complex adaptive systems characterized by large numbers of heterogeneous entities, non-linear interdependencies, and emergent collective behavior. The diversity of device capabilities — from battery-powered microcontrollers with kilobytes of RAM to edge servers with multi-core processors and gigabytes of memory — necessitates adaptive resource management strategies that can simultaneously respect the constraints of the weakest nodes and exploit the capacities of the strongest. Furthermore, the spatial distribution of deployments, spanning local area networks, metropolitan infrastructure, and wide-area cloud backends, introduces multi-scale latency and bandwidth constraints that profoundly affect the feasibility of centralized control.

The dynamic nature of IoT environments further compounds their complexity. Device availability fluctuates due to mobility, energy depletion, and hardware failure; user demand patterns evolve over time; and the service landscape itself is subject to continuous change as new services are deployed, updated, or decommissioned. These conditions demand management frameworks that are not merely reactive but proactively adaptive, capable of anticipating state transitions and adjusting coordination strategies before performance degradation occurs.

1.2.2 Limitations of Existing Approaches

Existing approaches to IoT service management can be broadly categorized along three dimensions: centralized versus decentralized control, reactive versus proactive adaptation, and model-based versus learning-based decision-making. Traditional centralized orchestration systems — including cloud-based IoT platforms such as AWS IoT Core, Microsoft Azure IoT Hub, and Google Cloud IoT — offer strong global consistency guarantees but are ill-suited to low-latency, high-reliability applications due to the round-trip delays imposed by cloud communication. Edge computing frameworks partially address this limitation but typically rely on static resource allocation policies that cannot adapt to real-time demand shifts.

Model-based approaches grounded in formal methods, constraint programming, or classical optimization provide provable correctness and predictable behavior, but their applicability is constrained by the need for accurate environment models — a requirement

that is fundamentally at odds with the open-world assumption of large-scale IoT. Rule-based expert systems encode domain knowledge effectively but cannot generalize beyond the scenarios anticipated at design time, and their maintenance costs grow super-linearly with the complexity of the deployment environment.

Learning-based approaches — particularly those leveraging Deep Reinforcement Learning — have demonstrated impressive results in single-agent IoT management tasks, including resource scheduling, power control, and task offloading. However, the straightforward extension of single-agent DRL to multi-agent settings introduces well-documented challenges: non-stationarity of the environment from each agent’s perspective, the exponential growth of joint action and state spaces, and the difficulty of credit assignment in cooperative multi-agent scenarios. Moreover, existing MADRL solutions for IoT have tended to optimize for isolated objectives — typically either latency or energy — without jointly addressing the coupled constraints of QoS compliance, energy efficiency, and system resilience. This gap in the literature constitutes the primary motivation for the research presented in this thesis.

1.2.3 The Promise of MAS and DRL Integration

The integration of Multi-Agent Systems with Deep Reinforcement Learning offers a compelling path toward addressing these limitations. MAS provides a natural architectural metaphor for IoT deployments — distributed, autonomous agents correspond directly to the heterogeneous nodes and services that populate real-world deployments — and supplies a rich theoretical foundation for reasoning about agent interaction, coordination, and emergent collective behaviour. DRL, meanwhile, offers the capacity for agents to learn complex, non-linear policies from high-dimensional inputs without requiring explicit environment models, through trial-and-error interaction guided by scalar reward signals.

The Centralized Training with Decentralized Execution (CTDE) paradigm represents one of the most productive recent developments in MADRL, enabling agents to be trained with access to global information while retaining the capacity for fully autonomous decentralized execution at deployment time. This paradigm is particularly well suited to IoT service management, where global coordination information is available during system design and training phases but must give way to local decision-making at runtime due to communication and latency constraints. Hierarchical extensions of CTDE, in which agents at different levels of an organizational hierarchy specialize in decisions at different temporal and spatial scales, further enhance the scalability and expressiveness of the approach.

1.3 Problem Statement

This thesis addresses the following core problem:

How can a hierarchical multi-agent deep reinforcement learning framework be designed to jointly optimise service composition, energy efficiency, QoS compliance, and system resilience in large-scale, heterogeneous, and dynamic IoT environments, while preserving decentralised autonomy and achieving scalable coordination across multiple levels of system abstraction?

This problem is notable for three interlocking dimensions of difficulty. First, the scale

and heterogeneity of target deployments preclude both exhaustive search and uniform policy design, necessitating hierarchical decomposition and specialised learning strategies at each level. Second, the multi-objective nature of the optimisation — jointly addressing latency, reliability, energy consumption, and fault tolerance — requires careful reward shaping and inter-level coordination mechanisms that prevent objectives from being optimised in isolation at the expense of others. Third, the decentralised autonomy constraint — reflecting the communication and privacy requirements of real IoT deployments — prohibits reliance on a centralised controller at execution time, demanding that agents be capable of high-quality, coordinated decision-making based solely on local observations.

1.4 Research Questions

The research presented in this thesis is organised around three principal research questions, each targeting a distinct facet of the overarching problem:

RQ1: *How can decisions be coordinated across device, service, and system levels in a scalable manner within large-scale IoT deployments?*

RQ2: *How can energy efficiency, QoS compliance, and system resilience be jointly optimised under highly dynamic and uncertain IoT operating conditions?*

RQ3: *How can learning mechanisms be designed to preserve decentralised agent autonomy while enabling effective alignment with global system-level objectives?*

RQ1 targets the architectural and algorithmic foundations of hierarchical multi-agent coordination, asking how a structured decomposition of the service management problem across levels of abstraction can yield scalable performance without sacrificing coordination quality. **RQ2** concerns the reward engineering and multi-objective optimisation dimensions of the framework, examining how competing performance criteria can be balanced within a unified learning paradigm. **RQ3** addresses the fundamental tension between global coherence and local autonomy, investigating how the CTDE paradigm can be extended and refined within a hierarchical setting to achieve both.

1.5 Proposed Approach

To address the identified research questions, this thesis proposes a hierarchical and graph attention MADRL framework specifically designed for adaptive, energy-efficient, and resilient service management in dynamic IoT environments. The framework builds upon and substantially extends our prior work on MAS-DRL for dynamic service optimization in distributed IoT systems, advancing that foundational contribution along three principal dimensions: hierarchical architectural structure, explicit IoT constraint modeling, and scalability and resilience-oriented design.

Our Approach operates through a **three-level hierarchical architecture**, with each level hosting specialized DRL agents responsible for decisions at a distinct spatiotemporal scale. The levels and their respective roles are as follows:

- **High Level — System Coordination:** A global coordination agent optimises long-term, system-wide objectives including stability, resilience, and aggregate energy efficiency. By preventing oscillatory behaviour, pre-empting failure cascades, and minimising redundant reconfiguration events, high-level coordination indirectly reduces system-wide energy consumption while maintaining service continuity.

- **Mid Level — Service Filtering and Composition:** A two-stage filtering process — comprising context discovery followed by QoS-driven selection — dramatically reduces the effective search space for service composition optimization. Context discovery identifies environment-relevant parameters and candidate services; QoS-driven selection ranks and selects compositions based on latency, reliability, user preferences, and energy budget. By constraining the environment complexity, this mid-level process enables both higher decision quality and lower computational overhead.
- **Low Level — Device Interaction and Conflict Resolution:** Atomic agents interacting directly with physical devices and actuators employ the CTDE paradigm to detect and resolve conflicting control actions. By ensuring coordination at the fine-grained level of individual actuations, the low level prevents energy waste from competing control signals and guarantees interoperability across heterogeneous hardware.

The three-level design implements CTDE globally: during training, agents at all levels have access to shared global state information, enabling the learning of cooperative, globally coherent policies. At execution time, each agent acts exclusively on the basis of its local observations, satisfying the decentralized autonomy constraint. This architecture is the first, to our knowledge, to apply hierarchical CTDE specifically to the joint optimization of service composition, energy efficiency, and resilience in IoT environments.

1.6 Contributions of Thesis

The thesis makes the following original contributions to knowledge:

Table 1.1: Summary of Contributions

#	Contribution	Description
C1	Hierarchical MADRL Architecture	A three-level hierarchical multi-agent DRL framework and graph attention framework based on CTDE, designed for scalable and adaptive service management in heterogeneous and dynamic IoT environments.
C2	Context-Aware Service Coordination	A two-stage coordination mechanism combining context discovery with QoS-driven service composition, enabling dynamic, constraint-aware service orchestration.
C3	Resilience-Oriented Coordination	High-level coordination mechanisms explicitly designed to enhance system robustness and maintain service continuity under device failures and disturbances.
C4	Energy-Aware Selective Activation	A selective agent activation strategy that reduces environmental energy consumption by engaging only context-relevant agents, with formal analysis of energy-performance trade-offs.
C5	Formal Verification	a complete Event-B modeling of the proposed IoT system modeling and System with MDP reduction for Well-definedness, safety and correctness.

1.7 Scope and Delimitations

The research presented in this thesis focuses specifically on the service management layer of IoT systems, treating the underlying communication and perception layers as given. While the frameworks are designed to be communication-protocol-agnostic, experimental evaluation is conducted in simulated environments calibrated to realistic IoT deployments; evaluation at full-scale physical infrastructure is identified as a direction for future work. The MADRL algorithms employed — specifically variants of Proximal Policy Optimisation (PPO) and Multi-Agent PPO (MAPPO) — are chosen for their stability and sample efficiency; the thesis does not claim that **Hi-MAD** and **GAMMA** are optimal for all possible DRL algorithmic choices, and sensitivity analyses with respect to algorithm selection are presented in Chapter 4.

Security and privacy aspects of the IoT deployment — including data confidentiality, access control, and adversarial robustness — are treated as orthogonal concerns and are not the primary focus of this thesis, though their interaction with the proposed management framework is discussed in the concluding chapter.

1.8 Thesis Structure

The remainder of this thesis is organized as follows:

- Chapter 2 — State of Art: provides a comprehensive review of the relevant literature, covering IoT service management, Service-Oriented Computing, Multi-Agent Systems, Deep Reinforcement Learning, and prior work on MAS-DRL for IoT. The chapter positions the contributions of this thesis within the current state of the art and identifies the key gaps that motivate the proposed framework.
- Chapter 3 — IoT System Modeling Conceptualization: presents the IoT system model and the foundational formal concepts, including the service composition problem formulation, the multi-agent Markov decision process model, and the mathematical of the components in multiple layers.
- Chapter 4 — Hierarchical and Graph- Attention MADRL Frameworks for IoT Service Optimization: details the hierarchical and graph attention architecture, learning mechanisms, and algorithmic components of the proposed frameworks. The chapter covers the design of each hierarchical level, the CTDE training procedure, the reward engineering approach, and the selective agent activation strategy. **Hi-MAD** and **GAMMA** work complementary to achieve the system objectives and ensure the stability of the environment.
- Chapter 5 — Formal Verification & Experimental Validation: presents the Event-B architecture of the contributions related to **Hi-MAD** and **GAMMA** and verify the safety and the correctness of theirs deployments. And describes the experimental setup, simulation environments, baseline comparisons, evaluation metrics, and results.
- Chapter 6 — Conclusion and Future Work: summarizes the thesis contributions, discusses the limitations of the current work, and outlines directions for future research, including full-scale physical deployment, formal policy verification, and extensions to privacy-preserving and adversarially robust IoT management.

Chapter 2

State of Art

2.1 Introduction

IoT services enable seamless integration of heterogeneous devices and applications, but their efficient management requires sophisticated mechanisms for composition, coordination, and adaptation in highly dynamic environments. This chapter surveys the state of the art in IoT service composition and coordination with a particular focus on MAS-DRL. We first trace the evolution from rule/model-driven and optimization-based methods to AI-enabled solutions and decentralized frameworks (including blockchain-based coordination and federated/edge-centric learning). We then analyze how MAS-DRL instantiates these ideas in practice—through training regimes such as centralized-training-decentralized-execution (CTDE) versus fully decentralized learning, communication topologies and coordination mechanisms, and methods that explicitly account for robustness, safety, privacy, and resource constraints. Throughout, we emphasize transferability to real deployments and draw connections to system-level metrics and service-level agreements (SLAs).

This chapter has four aims. **(A1)** Map the IoT service-composition literature across five families, rule/model-driven, optimization-based, AI/learning-based, decentralized/blockchain-oriented, and federated/edge-centric to establish a consistent vocabulary and baseline (Sec. 2.3, supported by Sec. 2.2). **(A2)** Develop a taxonomy for MAS-DRL in IoT—training regime (CTDE vs. fully decentralized), communication/topology, coordination mechanisms, robustness/safety, privacy/trust, resource-/energy-awareness, and scalability—and organize representative methods accordingly (Sec. 2.4). **(A3)** Provide a comparative analysis across these approaches using QoS/QoE, latency–energy trade-offs, privacy, and resilience criteria, highlighting strengths, limitations, and transferability to real deployments (Sec. 2.5). **(A4)** Synthesize the key insights into design principles and open problems that directly motivate the thesis’s technical contributions (Sec. 2.6).

2.2 Background

This presents the foundational components and formal concepts essential for service management into IoT environments.

2.2.1 IoT System Architecture

An IoT system can be formally modeled as a dynamic environment $\mathcal{E}$ comprising a set of heterogeneous interconnected devices where each device $d_j \in \mathcal{D}$ performs three core functions: sensing, communication, and actuation. At discrete time steps t , device d_j observes a local state $s_j^t \in \mathcal{S}_j$, where the state may encode temperature, occupancy, energy consumption, or other relevant environmental features. Based on s_j^t , the device selects an action $a_j^t \in \mathcal{A}_j$ through a local policy $\pi_j : \mathcal{S}_j \rightarrow \mathcal{A}_j$. Thus, the environment evolves according to stochastic dynamics: $s_j^{t+1} \sim \mathcal{P}_j(s_j^{t+1} | s_j^t, a_j^t)$, and the device receives a scalar reward $r_j^t = \mathcal{R}_j(s_j^t, a_j^t)$, which evaluates the effectiveness of its action relative to system objectives (e.g., energy saving, delay minimization).

IoT generates vast data volumes, requiring a hierarchical architecture that manages processing and storage across tiers to balance latency, scalability, and computational efficiency as illustrated in 2.1. The most widely referenced IoT architecture comprises three distinct layers, each serving a critical function in the data lifecycle:

Perception Layer: It is the physical interface of an IoT system, consisting of sensors and actuators that collect data and execute actions. It incorporates technologies like RFID, embedded systems, and short-range communication protocols to enable local processing and device-level connectivity.

Network Layer: It facilitates reliable communication between IoT systems layers, handling tasks such as routing, data aggregation, and protocol translation. It uses network protocols (e.g., 6LoWPAN, LoRaWAN) and lightweight messaging protocols (e.g., MQTT, CoAP) for efficient data transfer. Gateways play a vital role by managing edge communication, security, and protocol conversion, while broader connectivity is enabled through technologies like 4G/5G, Wi-Fi, etc.

Application Layer: It represents the topmost level of the IoT architecture, delivering intelligent, domain-specific services by leveraging data from lower layers. It handles tasks such as data analysis, visualization, service provisioning, and user interaction. Built on SOA principles and powered by APIs, it ensures modularity and interoperability across diverse platforms. This layer enables real-world applications like industrial monitoring, remote healthcare, and smart city management.

To overcome latency and scalability issues in traditional IoT models, a hierarchical architecture—comprising Edge, Fog, and Cloud layers—has emerged. The *Edge layer* enables ultra-fast local processing near devices, the *Fog layer* supports regional analytics and decision-making, and the *Cloud layer* handles global, resource-intensive tasks. This structure enhances responsiveness, reduces bandwidth use, and supports scalable IoT deployment.

2.2.2 Service Management

Service management in IoT represents a foundational shift from traditional ITSM, which operates in controlled, resource-rich environments with stable network topologies, homogeneous infrastructures, and centralized orchestration models such as ITIL and SOA. In contrast, IoT environments are inherently dynamic, heterogeneous, and resource-constrained, comprising vast numbers of mobile, intermittently connected devices with

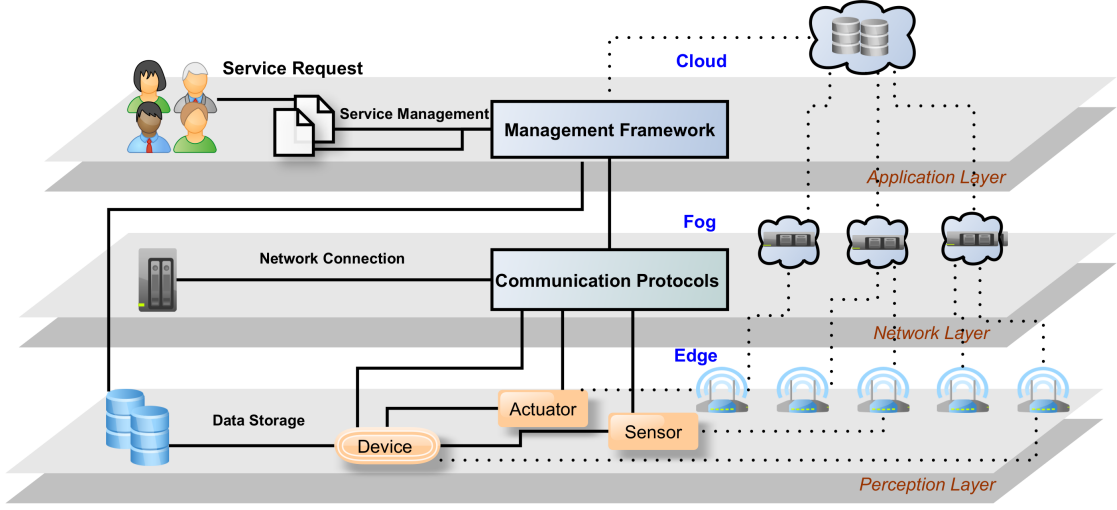


Figure 2.1: IoT System Architecture

limited computation, energy, and bandwidth. Effective IoT service management must therefore support decentralized, scalable, and context-aware orchestration capable of real-time adaptation to variations in user behavior, environmental context, and device conditions. This involves continuous optimization of services across the edge–fog–cloud continuum, where services S_i are dynamically mapped to nodes N_j such that Quality of Service (QoS) constraints are satisfied, expressed as:

$$\min_A \sum_{i=1}^n C(S_i, N_j) \quad \text{subject to} \quad \text{QoS}(S_i, N_j) \geq \theta_i, \\ R(N_j) \geq R_{\min}, \quad E(N_j) \leq E_{\max}$$

where C denotes cost, QoS reflects latency, availability, or throughput thresholds θ_i , and R , E represent resource and energy constraints, respectively. Unlike ITSM's static SLAs, IoT service management requires runtime SLA negotiation and enforcement, supported by lightweight protocols, edge intelligence, and adaptive discovery mechanisms that align with the fluidity and scale of IoT architectures, as illustrated in Figure 2.1.

2.2.3 Key Terminology & Concepts

Service Provisioning: Allocating resources to deliver a service S to a user or device U , typically under constraints:

$$\min_{x \in \mathcal{X}} C(x) \quad \text{s.t.} \quad R(x) \geq R_{\min}, \quad E(x) \leq E_{\max},$$

where $C(x)$ is the cost, $R(x)$ the resource availability, and $E(x)$ the energy consumption.

Service Discovery: Locating available services via protocols such as mDNS, CoAP RD, or decentralized registries. The discovery time is typically modeled as a function of network characteristics: $T_{\text{discovery}} = f(N, B, L)$, where N is the number of nodes, B is the available bandwidth, and L is the network latency.

Service Orchestration: Dynamic coordination of services across devices, often modeled as a directed graph $G(V, E)$, where V represents service tasks and E encodes dependencies between them. The orchestration is typically optimized using scheduling algorithms to ensure efficient execution and resource utilization.

Quality of Service: Ensuring non-functional attributes where QoS parameters are often optimized through multi-objective functions depending on the system's performance requirements.

$$\text{QoS} = \{\text{Latency } (L), \text{Availability } (A), \text{Reliability } (R)\}.$$

Service Level Agreement: A formal contract specifying the expected QoS guarantees. SLA violations are measured as:

$$\text{SLA}_{\text{violation}} = \frac{1}{n} \sum_{i=1}^n \mathbf{1}_{\text{QoS}_i < \text{Threshold}_i} \text{ where } \mathbf{1} \text{ is the indicator function, returning 1 if the QoS for service } i \text{ is below the required threshold, and 0 otherwise.}$$

Context-Awareness: The ability of a system to adapt services based on environmental or user context C . A context-aware service S can be modeled as: $S = f(D, C)$ where D denotes sensor data, and C is the context vector (e.g., location, time, device state).

The overall control challenge lies in designing distributed policies that ensure coordinated, intelligent behavior under partial observability, dynamic environments, and constrained communication.

2.2.4 Multi-Agent System Architecture

Figure 2.2 illustrates a generic *Multi-Agent System (MAS)* architecture, which integrates autonomous agents, the environment, communication channels, and collaboration strategies. MAS are widely recognized as a fundamental paradigm for modeling distributed, intelligent, and adaptive systems, particularly in domains such as Internet of Things (IoT), edge computing, and cyber-physical infrastructures. The distinguishing feature of MAS is the ability of agents to operate both independently and collectively, enabling decentralized problem solving in dynamic and uncertain environments.

Multi-Agent System: A Multi-Agent System (MAS) is a distributed system composed of multiple autonomous agents that perceive their environment, reason about their states, and act upon the environment or communicate with other agents to achieve individual or collective goals. Formally, a MAS can be described as:

$$\mathcal{M} = \langle \mathcal{A}, \mathcal{E}, \mathcal{C}, \mathcal{S}, \mathcal{T}, \mathcal{G} \rangle \quad (2.1)$$

where:

- $\mathcal{A} = \{a_1, a_2, \dots, a_n\}$ is the set of autonomous agents;
- $\mathcal{E}$ is the environment in which agents act, including a physical environment $\mathcal{E}_p$ and a knowledge base $\mathcal{E}_k$:

$$\mathcal{E} = \mathcal{E}_p \cup \mathcal{E}_k; \quad (2.2)$$

- $\mathcal{C}$ represents the communication channels used for message exchange;

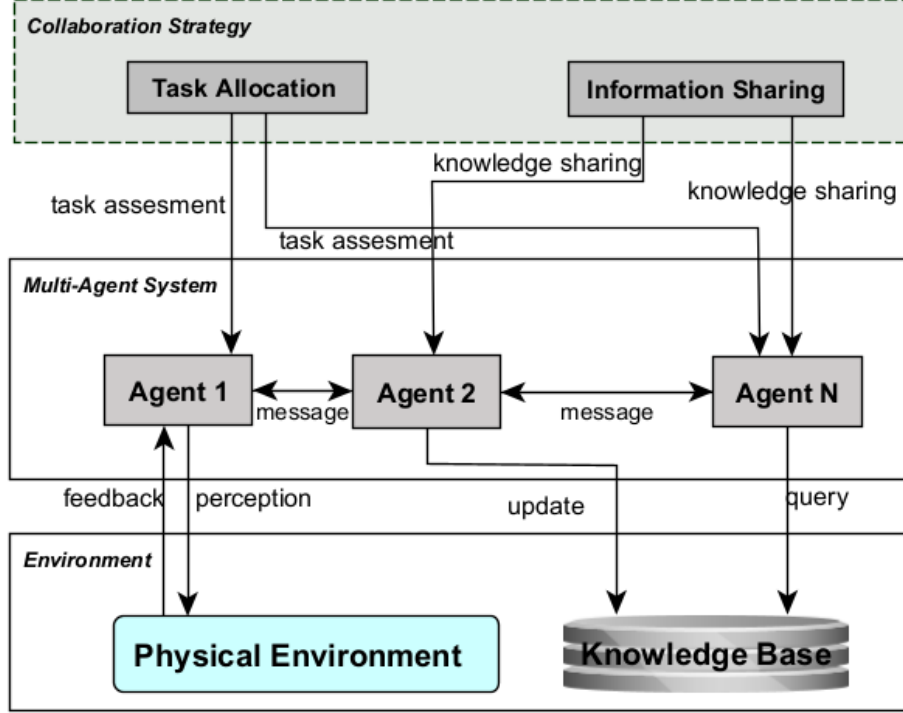


Figure 2.2: Multi-Agent System Architecture.

- $\mathcal{S}$ denotes the collaboration strategies governing coordination, such as task allocation and information sharing;
- $\mathcal{T}$ is the global transition function defining the system dynamics;
- $\mathcal{G} = \{g_1, g_2, \dots, g_m\}$ represents the set of individual or collective goals.

This definition ensures that the MAS captures both the structural components (agents and environment) and the interactional mechanisms (communication and collaboration).

Agent Model: An agent is an autonomous computational entity capable of perceiving its environment, reasoning about its internal state, and executing actions to achieve defined objectives. Each agent $a_i \in \mathcal{A}$ can be formally modeled as:

$$a_i = \langle \mathcal{S}_i, \mathcal{P} \upharpoonright \nabla_i, \mathcal{A} \upharpoonright \sqcup_i, \pi_i \rangle \quad (2.3)$$

with:

- $\mathcal{S}_i$ the internal state space of the agent;
- $\mathcal{P} \upharpoonright \nabla_i : \mathcal{E} \rightarrow 2^{\mathcal{O} \upharpoonright f_i}$ the perception function, mapping environment states to observations;
- $\mathcal{A} \upharpoonright \sqcup_i$ the action space available to the agent;
- $\pi_i : \mathcal{S}_i \times \mathcal{O} \upharpoonright f_i \rightarrow \mathcal{A} \upharpoonright \sqcup_i$ the policy function defining how the agent selects actions given states and observations.

This abstraction highlights the autonomy of agents: each a_i is capable of reasoning locally while contributing to the system's global objectives.

Environment Dynamics: The environment is the external context in which agents operate. It provides stimuli (perceptions) and evolves according to agents' actions, thus forming a feedback loop. The environment evolves according to the transition function:

$$T : \mathcal{E} \times \prod_{i=1}^n \mathcal{A}|_{\sqcup_i} \rightarrow \mathcal{E} \quad (2.4)$$

which updates the environment state based on the joint actions of all agents, represented by the action vector

$$\mathbf{a} = (a_1, a_2, \dots, a_n).$$

This captures the interdependence of agents' decisions: the outcome of one agent's action often depends on the actions of others.

Communication and Collaboration: Communication in MAS is the process through which agents exchange messages to coordinate, negotiate, and share information in order to achieve local or global objectives. Agents coordinate through explicit communication, represented by:

$$COMM : \mathcal{A} \times \mathcal{A} \rightarrow \mathcal{Mf}, \quad (2.5)$$

where $\mathcal{Mf}$ denotes the message space. Such communication supports:

- **Task Allocation:**

$$ALLOC : \mathcal{G} \times \mathcal{A} \rightarrow \mathcal{G}_i \quad (2.6)$$

which distributes global goals $\mathcal{G}$ among agents, ensuring balanced workload and efficiency;

- **Information Sharing:**

$$SHARE : \mathcal{A} \times \mathcal{E}_k \rightarrow \mathcal{E}_k \quad (2.7)$$

enabling agents to collaboratively update and query the shared knowledge base.

These strategies ensure that local reasoning contributes to coherent global behavior, a central challenge in distributed problem-solving.

System Utility: The utility function quantifies the performance of the system or individual agents. It can represent either a shared objective (cooperative MAS) or individual objectives (competitive MAS). The performance of the MAS is evaluated using a utility function:

$$U : \mathcal{E} \times \prod_{i=1}^n \mathcal{A}|_{\sqcup_i} \rightarrow \mathbb{R}, \quad (2.8)$$

which quantifies the quality of outcomes. In cooperative MAS, a shared global utility U is maximized, whereas in competitive MAS, each agent optimizes an individual utility U_i .

This architecture emphasizes the cyclic interactions between agents and their environment: agents perceive, decide, and act, while continuously exchanging knowledge and coordinating tasks. The combination of autonomy, communication, and collaboration makes MAS highly suitable for distributed domains such as IoT, edge/fog computing, and service composition, where scalability, resilience, and adaptability are critical requirements.

2.2.5 Deep Reinforcement Learning

Reinforcement Learning (RL) is a computational paradigm inspired by behavioral psychology, where an *agent* interacts with an *environment* to maximize long-term cumulative rewards. Formally, RL is modeled as a *Markov Decision Process (MDP)* defined by the tuple:

$$\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$$

where:

- $\mathcal{S}$ is the set of states,
- $\mathcal{A}$ is the set of actions,
- $P(s'|s, a)$ is the transition probability from state s to s' after action a ,
- $R(s, a)$ is the reward function,
- $\gamma \in [0, 1]$ is the discount factor balancing immediate and future rewards.

The objective is to find an optimal policy $\pi^*(a|s)$ that maximizes the expected cumulative discounted reward:

$$J(\pi) = \mathbb{E}_{\pi} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right].$$

Value Functions and Bellman Equations: To evaluate policies, RL employs value functions:

1. The **state-value function**:

$$V^{\pi}(s) = \mathbb{E}_{\pi} \left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s \right],$$

2. The **action-value function (Q-function)**:

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi} \left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, a_0 = a \right].$$

3. The recursive Bellman optimality equation is defined as:

$$Q^*(s, a) = \mathbb{E}_{s'} \left[R(s, a) + \gamma \max_{a'} Q^*(s', a') \right].$$

From RL to Deep Reinforcement Learning: Traditional RL struggles with high-dimensional state and action spaces. *Deep Reinforcement Learning (DRL)* addresses this by leveraging deep neural networks (DNNs) as function approximators for policies and/or value functions.

For instance, the Deep Q-Network (DQN) introduced by Mnih *et al.* approximates the action-value function as:

$$Q(s, a; \theta) \approx Q^*(s, a),$$

where θ are the parameters of the neural network. Training minimizes the temporal-difference (TD) loss:

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

with θ^- representing the target network parameters periodically synchronized for stability.

DRL Interaction Loop: The fundamental DRL process is illustrated in Figure 2.3. It represents the closed feedback loop between the agent and the environment:

1. **Agent:** At each time step t , the agent observes the state s_t and processes it through a DNN parameterized by θ . This network outputs either a policy $\pi_\theta(a|s)$ (policy-based) or a value estimate $Q_\theta(s, a)$ (value-based).
2. **Action Selection:** Based on this output, the agent selects an action $a_t \in \mathcal{A}$.
3. **Environment:** The environment executes a_t , transitions to a new state s_{t+1} according to $P(s_{t+1}|s_t, a_t)$, and returns a reward $r_t = R(s_t, a_t)$.
4. **Learning:** The agent updates θ using the observed (s_t, a_t, r_t, s_{t+1}) , refining its policy to maximize:

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

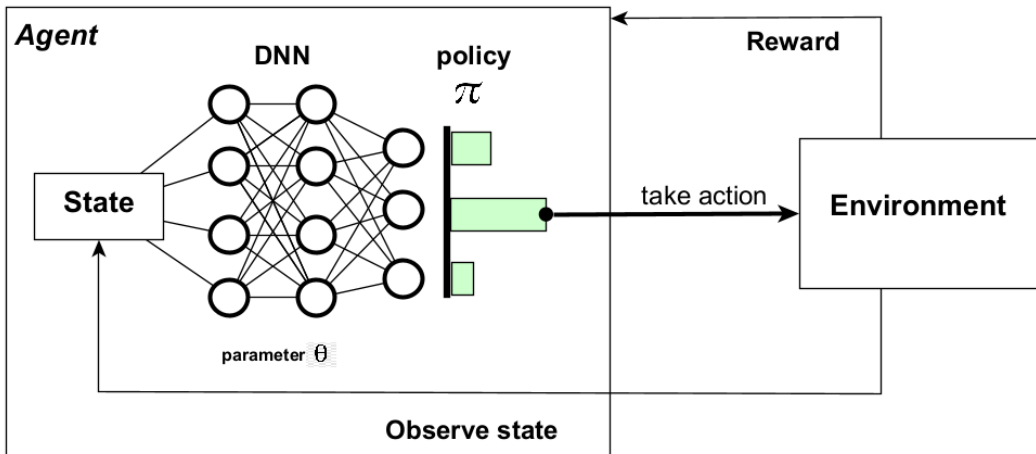


Figure 2.3: Deep Reinforcement Learning loop.

This cycle embodies the essence of DRL: *trial-and-error learning with deep function approximation*.

DRL Paradigms: Deep Reinforcement Learning methods are commonly categorized into three main paradigms:

1. **Value-based methods:** These approaches aim to approximate the optimal action-value function $Q^*(s, a)$. Examples include *Deep Q-Network (DQN)*, *Double DQN*, *Dueling DQN*, and *Rainbow DQN*.
2. **Policy-based methods:** These approaches directly optimize the policy $\pi_\theta(a|s)$ using gradient ascent on the expected return. The policy gradient is given by:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\pi_\theta} [\nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s, a)].$$

Examples include *REINFORCE*, *Advantage Actor-Critic (A2C)*, and *Proximal Policy Optimization (PPO)*.

3. **Actor-Critic methods:** These approaches combine value-based and policy-based techniques. The *actor* updates the policy, while the *critic* estimates value functions. Examples include *Asynchronous Advantage Actor-Critic (A3C)*, *Deep Deterministic Policy Gradient (DDPG)*, *Twin Delayed DDPG (TD3)*, and *Soft Actor-Critic (SAC)*.

2.2.6 Multi Agent Deep Reinforcement Learning

Multi-Agent Deep Reinforcement Learning (MADRL) extends Reinforcement Learning to settings where multiple agents simultaneously interact with a shared environment. As multiple agents adapt and interact, complex emergent behaviors arise, often exceeding the scope of individual policies. These behaviors include cooperation, competition, coordination, and negotiation, as illustrated in Figure 2.4.

Formal Foundations: The formal model underlying Multi-Agent Deep Reinforcement Learning is the Decentralized Partially Observable Markov Decision Process (Dec-POMDP):

$$M = \langle N, S, \{A_i\}_{i=1}^N, T, \{O_i\}_{i=1}^N, Z, r, \gamma \rangle,$$

where:

- N is the number of agents,
- S is the global state space,
- A_i is the action space of agent i , $A = \times_{i=1}^N A_i$ the joint action space,
- $T(s' | s, a)$ defines the state transition probability,
- O_i is the observation space of agent i ,
- $Z(o | s, a)$ the observation distribution,
- $r : S \times A \rightarrow \mathbb{R}$ is the reward function,
- $\gamma \in [0, 1)$ is the discount factor.

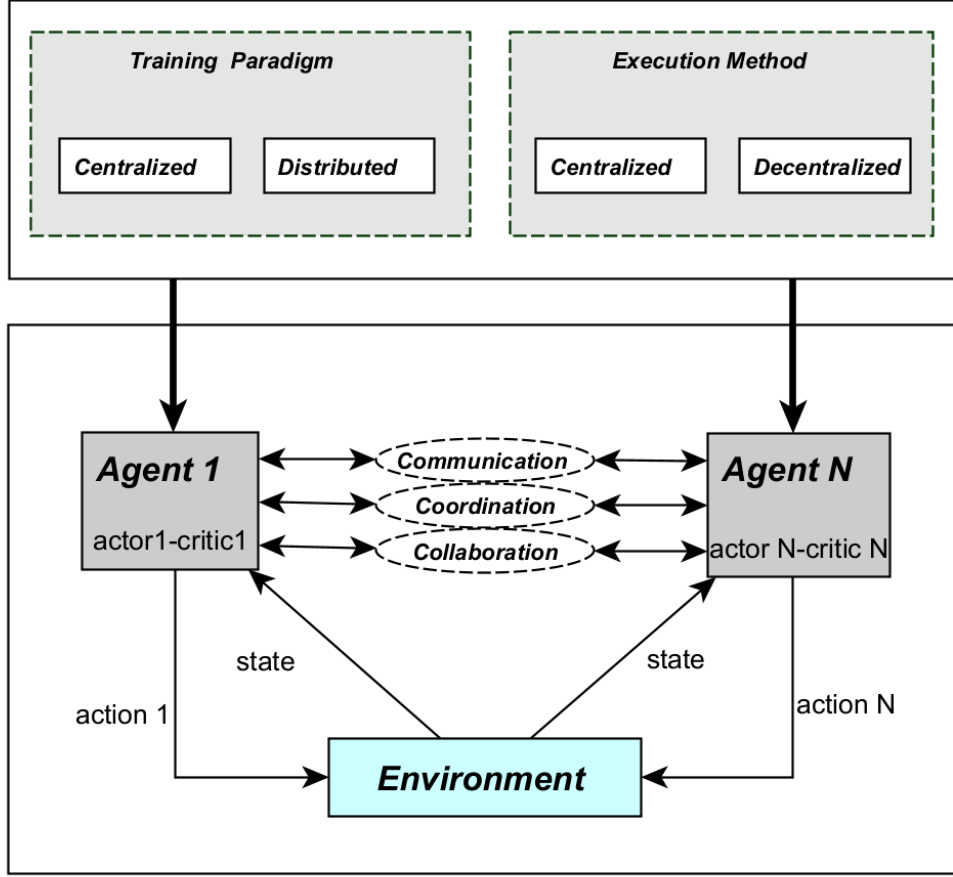


Figure 2.4: Multi-Agent Deep Reinforcement Learning.

Each agent maintains a local policy:

$$\pi_{\theta_i}(a_i^t \mid \tau_i^t), \quad \tau_i^t = (o_i^1, a_i^1, \dots, o_i^t),$$

and the joint policy is factorized as:

$$\pi_{\theta}(\mathbf{a}_t \mid \boldsymbol{\tau}_t) = \prod_{i=1}^N \pi_{\theta_i}(a_i^t \mid \tau_i^t).$$

The objective is to maximize the expected discounted return:

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\pi_{\theta}} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, \mathbf{a}_t) \right].$$

Methodological Paradigms:

1. Independent Learning:

Each agent learns independently, updating its value function:

$$Q_i^{\pi_i}(s, a_i) = \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t r_i(s_t, \mathbf{a}_t) \mid s_0 = s, a_i^0 = a_i \right].$$

The policy gradient for agent i is:

$$\nabla_{\theta_i} J_i(\theta_i) = \mathbb{E} [\nabla_{\theta_i} \log \pi_{\theta_i}(a_i \mid \tau_i) Q_i^{\pi_i}(s, a_i)].$$

2. Centralized Training with Decentralized Execution (CTDE)

A centralized critic evaluates the joint Q-function:

$$Q^\pi(s, a_1, \dots, a_N),$$

while decentralized actors optimize:

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E} [\nabla_{\theta_i} \log \pi_{\theta_i}(a_i \mid \tau_i) A_i(s, \mathbf{a})],$$

with the advantage function:

$$A_i(s, \mathbf{a}) = Q^\pi(s, \mathbf{a}) - \sum_{a'_i} \pi_{\theta_i}(a'_i \mid \tau_i) Q^\pi(s, (\mathbf{a}_{-i}, a'_i)).$$

3. Value Decomposition:

The global Q-function is factorized:

$$Q_{\text{tot}}(s, \mathbf{a}) = f(Q_1(o_1, a_1), \dots, Q_N(o_N, a_N), s),$$

where f satisfies the monotonicity constraint in QMIX:

$$\frac{\partial Q_{\text{tot}}}{\partial Q_i} \geq 0, \quad \forall i.$$

Learning Architectures:

1. **Multi-Agent Actor–Critic:** each agent updates with respect to a centralized critic:

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E} [\nabla_{\theta_i} \log \pi_{\theta_i}(a_i \mid \tau_i) Q_\phi(s, \mathbf{a})].$$

2. **Counterfactual Multi-Agent Baseline (COMA):** credit assignment is computed via the counterfactual advantage:

$$A_i^{\text{CF}}(s, \mathbf{a}) = Q(s, \mathbf{a}) - \sum_{a'_i} \pi_{\theta_i}(a'_i \mid \tau_i) Q(s, (\mathbf{a}_{-i}, a'_i)).$$

3. **Value Decomposition Networks (VDN):** simple additive decomposition:

$$Q_{\text{tot}}(s, \mathbf{a}) = \sum_{i=1}^N Q_i(o_i, a_i).$$

4. **Attention-Based Critics:** introduce a critic that weights interactions between agents dynamically:

$$Q_\phi(s, \mathbf{a}) = g \left(\sum_{j=1}^N \alpha_{ij} h_j(o_j, a_j) \right), \quad \alpha_{ij} = \frac{\exp(e_{ij})}{\sum_k \exp(e_{ik})}.$$

Representative Algorithms:

1. **Independent Q-Learning / PPO:** agents optimize individual returns without coordination.

2. **MADDPG (Multi-Agent Deep Deterministic Policy Gradient):**

Critic:

$$Q_i^\mu(s, a_1, \dots, a_N).$$

Policy gradient:

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E} [\nabla_{\theta_i} \mu_{\theta_i}(o_i) \nabla_{a_i} Q_i^\mu(s, a_1, \dots, a_N)].$$

3. **COMA:** actor-critic with counterfactual baselines for credit assignment.
4. **VDN and QMIX:** value decomposition methods; QMIX enforces monotonicity to ensure consistency.
5. **MAPPO:** multi-agent adaptation of PPO with clipped surrogate objective:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t [\min(\rho_t(\theta) A_t, \text{clip}(\rho_t(\theta), 1 - \epsilon, 1 + \epsilon) A_t)].$$

6. **Transformer-based Critics:** employ attention weights for scalability and adaptability in high-dimensional systems.

2.2.7 Graph Attention Networks

Graph Neural Networks (GNNs) have emerged as the standard paradigm for learning representations from non-Euclidean, graph-structured data. Early and highly successful variants, such as Graph Convolutional Networks (GCNs), operate by aggregating feature information from a node’s local neighborhood. Standard GCNs, however, rely on a static, structure-dependent normalization scheme—typically derived from node degree—to weigh the contribution of neighboring nodes.

This *isotropic* aggregation treats all neighbors equally, or scales their contributions purely according to structural properties, fundamentally limiting the model’s capacity to recognize that certain neighbors are inherently more informative than others. Graph Attention Networks (GATs) resolve this limitation by bridging graph-based deep learning with the self-attention mechanisms originally popularized in natural language processing. The conceptual distinction between the two aggregation regimes is illustrated in Figure 2.5.

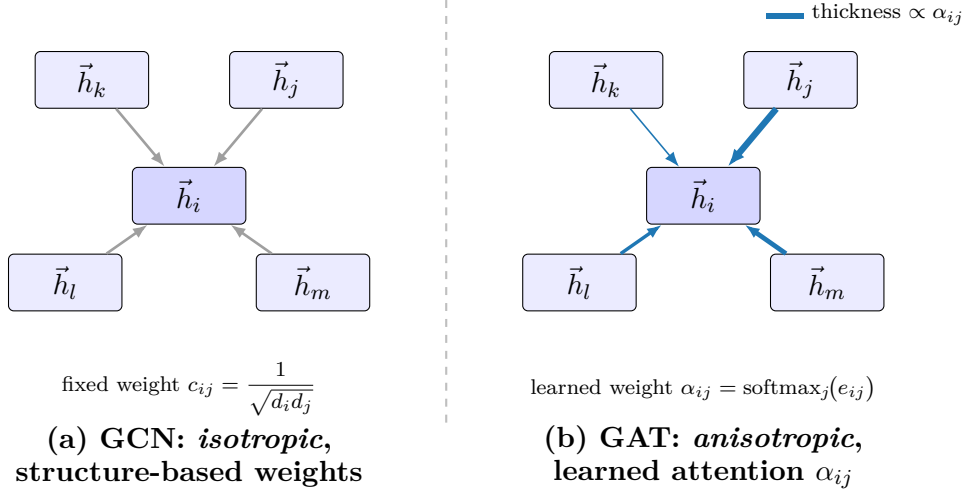


Figure 2.5: From graph convolution to graph attention.

The Core Mechanism: Self-Attention on Graph Nodes

The primary innovation of the GAT architecture is the application of a shared self-attention mechanism to compute the hidden representation of each node in the graph. Rather than relying on graph topology to dictate aggregation weights, a GAT dynamically learns the importance of each neighbour from the actual node features. This *anisotropic* approach allows the network to assign arbitrary, data-driven importances to different nodes within a neighbourhood, regardless of the neighbourhood's size or the node's degree. A high-level overview of the layer is provided in Figure 2.6.

Mathematical Formulation of the Attention Layer

The operation of a single GAT layer begins with a set of input node features, denoted $\mathbf{h} = \{\vec{h}_1, \vec{h}_2, \dots, \vec{h}_N\}$, where N is the number of nodes and $\vec{h}_i \in \mathbb{R}^F$ is the feature vector of node i with F features. The layer outputs a new set of node features, $\mathbf{h}' = \{\vec{h}'_1, \vec{h}'_2, \dots, \vec{h}'_N\}$, with $\vec{h}'_i \in \mathbb{R}^{F'}$.

To transform the input features into higher-level representations, a shared learnable linear transformation parameterised by a weight matrix $\mathbf{W} \in \mathbb{R}^{F' \times F}$ is applied to every node. A shared attentional mechanism $a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \rightarrow \mathbb{R}$ then computes the unnormalised attention coefficients

$$e_{ij} = a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j), \quad (2.9)$$

where e_{ij} indicates the importance of node j 's features to node i . To inject graph structure into the mechanism, the model performs *masked attention*—computing e_{ij} only for nodes $j \in \mathcal{N}_i$, where $\mathcal{N}_i$ is the neighbourhood of node i (typically including node i itself).

To make the coefficients comparable across different nodes, they are normalised with a softmax function over all choices of j :

$$\alpha_{ij} = \text{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}. \quad (2.10)$$

In the standard GAT implementation, the attention mechanism a is a single-layer feedforward neural network parameterised by a weight vector $\vec{\mathbf{a}} \in \mathbb{R}^{2F'}$, followed by a

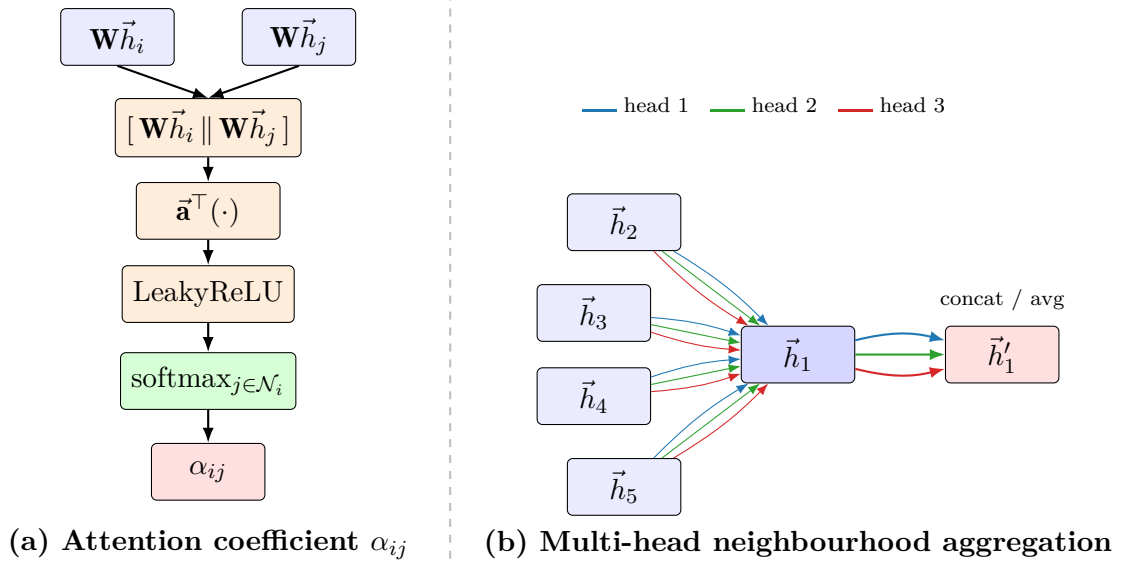


Figure 2.6: Architecture overview of a Graph Attention Network layer.

LeakyReLU nonlinearity. Fully expanded, the normalised coefficients are

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{a}^\top [\mathbf{W}\vec{h}_i \parallel \mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(\text{LeakyReLU}\left(\vec{a}^\top [\mathbf{W}\vec{h}_i \parallel \mathbf{W}\vec{h}_k]\right)\right)}, \quad (2.11)$$

where $\parallel$ denotes concatenation. Once the normalised coefficients are obtained, they are used to form a linear combination of the corresponding transformed features, followed by an activation σ , yielding the output feature of node i :

$$\vec{h}'_i = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W}\vec{h}_j\right). \quad (2.12)$$

Stabilisation via Multi-Head Attention

To stabilise the learning process of self-attention, GATs employ *multi-head attention*, a concept akin to the Transformer architecture. K independent attention mechanisms execute the transformation of Equation (2.12), and their outputs are concatenated:

$$\vec{h}'_i = \parallel_{k=1}^K \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j\right), \quad (2.13)$$

where α_{ij}^k is the normalised attention coefficient computed by the k -th mechanism and $\mathbf{W}^k$ is the corresponding input linear transformation. If multi-head attention is applied in the final prediction layer, concatenation is typically replaced by averaging to produce the output.

Key Advantages and Expressivity

GATs offer several critical theoretical and practical advantages:

Goal: high intra-cluster similarity, high inter-cluster dissimilarity ($\times$ = centroid μ_k ; a_i = mean intra-cluster distance, b_i = mean nearest-cluster distance)

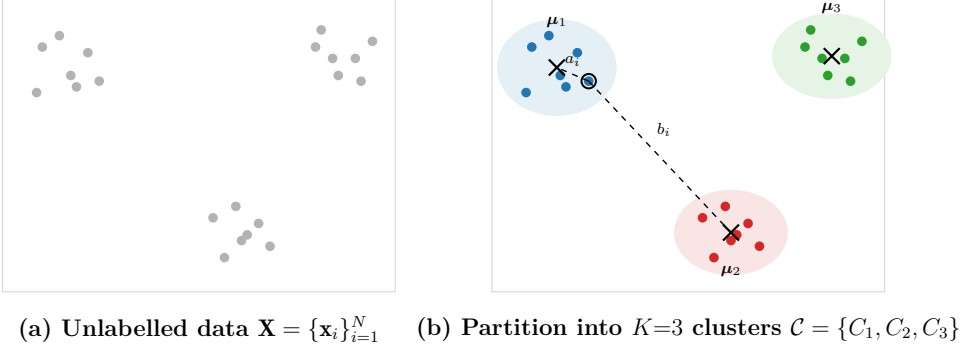


Figure 2.7: Overview of the clustering task.

- **Computational efficiency.** The operations of the self-attention layer can be highly parallelised across all edges, and the computation of output features can be parallelised across all nodes.
- **Inductive capability.** Unlike spectral GCNs, which are bound to a specific graph structure during training and require access to the full graph Laplacian, GATs rely purely on local neighbourhoods. This allows them to perform inductive learning seamlessly—evaluating nodes on completely unseen graphs.
- **Interpretability.** The learned attention coefficients α_{ij} inherently provide a degree of interpretability, as they quantify the relative influence of neighbouring nodes on the target node’s representation.

2.2.8 The Unsupervised Learning Paradigm

In the landscape of machine learning, *unsupervised learning* involves identifying hidden structures within unlabelled data. Unlike supervised learning, which relies on ground-truth annotations to guide the optimisation process, unsupervised models must infer the natural topology of the data space autonomously. *Clustering* is the most prominent task within this paradigm. Its fundamental objective is to partition a dataset into meaningful subgroups (clusters) such that data points within the same cluster exhibit high *intra-cluster similarity*, while points in different clusters demonstrate high *inter-cluster dissimilarity*. This principle is illustrated in Figure 2.7.

Formal Definition and Similarity Metrics

The clustering problem can be formally defined over a dataset of N observations, $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$, where each observation $\mathbf{x}_i \in \mathbb{R}^d$ is a d -dimensional feature vector. The goal is to assign each observation to one of K clusters, $\mathcal{C} = \{C_1, C_2, \dots, C_K\}$, such that the clusters jointly reconstruct the dataset, $\bigcup_{i=1}^K C_i = \mathbf{X}$, and remain mutually disjoint, $C_i \cap C_j = \emptyset$ for $i \neq j$.

The notion of *similarity* is foundational to any clustering algorithm and is strictly dictated by the chosen distance metric. The most ubiquitous choice for continuous variables

is the Euclidean distance,

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{m=1}^d (x_{im} - x_{jm})^2}. \quad (2.14)$$

Depending on the nature of the feature space, alternative metrics—such as the Manhattan distance, cosine similarity, or Mahalanobis distance—may be employed to better capture the underlying data distribution.

Taxonomy of Clustering Methodologies

Clustering algorithms are generally categorised according to their underlying theoretical assumptions regarding what constitutes a cluster. The field is broadly divided into three primary paradigms, summarised in Table 2.1.

Table 2.1: The three primary paradigms of clustering.

Methodology	Core concept	Example algorithm	Key strength
Partitioning	Divides data into non-overlapping subsets based on distance to centroids.	K-Means	Computationally efficient and highly scalable for large datasets.
Hierarchical	Builds a tree-like structure (dendrogram) of nested clusters based on linkage criteria.	Agglomerative clustering	Does not require pre-specifying the final number of clusters.
Density-based	Groups contiguous regions of high data density separated by sparse areas.	DBSCAN	Highly effective at discovering clusters of arbitrary, non-convex shapes.

Mathematical Formulation: Partitioning via K-Means

As the most widely implemented partitioning method, K-Means seeks to minimise the within-cluster sum of squares (WCSS), also known as *inertia*. Given a predetermined number of clusters K , the algorithm iteratively refines a set of centroids $\boldsymbol{\mu} = \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_K\}$. The objective function to be minimised is

$$J = \sum_{i=1}^K \sum_{\mathbf{x}_j \in C_i} \|\mathbf{x}_j - \boldsymbol{\mu}_i\|^2, \quad (2.15)$$

where $\|\mathbf{x}_j - \boldsymbol{\mu}_i\|^2$ is the squared Euclidean distance between a data point $\mathbf{x}_j$ and the centroid $\boldsymbol{\mu}_i$ of its assigned cluster C_i . Because finding the global optimum of this problem is NP-hard, the algorithm employs a heuristic (Lloyd’s algorithm) that alternates between an *assignment step* and an *update step* until convergence is achieved.

Evaluation of Cluster Quality

Assessing the performance of a clustering algorithm is inherently challenging due to the absence of ground-truth labels. Consequently, evaluation relies heavily on *internal validation metrics* that quantify the structural integrity of the resulting clusters.

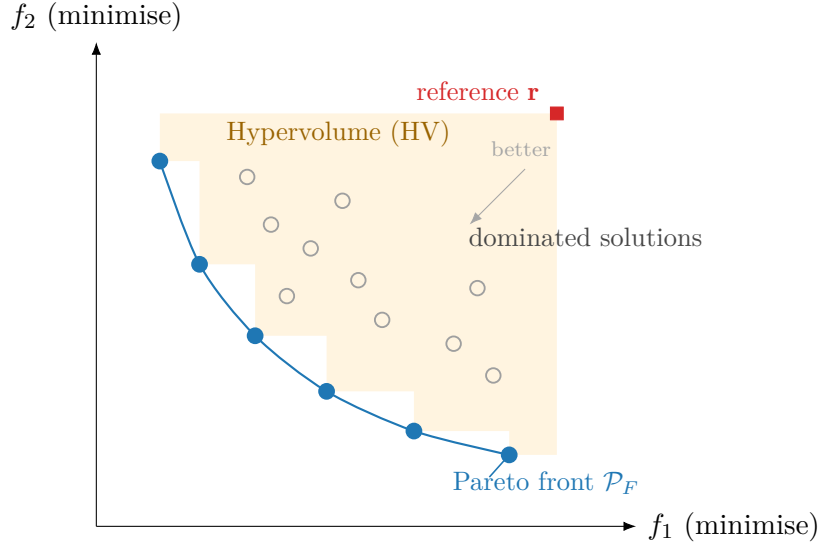


Figure 2.8: Overview of a bi-objective minimisation problem in objective space.

A prominent measure is the *Silhouette Coefficient*, which evaluates how similar an object is to its own cluster compared with other clusters. For a given point i , the silhouette score is

$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)}, \quad (2.16)$$

where a_i is the mean intra-cluster distance and b_i is the mean nearest-cluster distance (cf. Figure 2.7b). The score ranges from -1 to 1 , where a value closer to 1 indicates that the point is well matched to its own cluster and clearly separated from neighboring clusters. Other standard internal metrics include the Davies–Bouldin Index and the Calinski–Harabasz Index, both of which mathematically balance intra-cluster variance against inter-cluster separation.

2.2.9 Multi-Objectives Optimization

In classical mathematical programming, the optimisation process is generally driven by a singular goal: finding a solution that minimises or maximises a solitary scalar objective function. The majority of complex real-world engineering, economic, and scientific systems, however, inherently require the simultaneous optimisation of two or more *conflicting* objectives. Multi-Objective Optimisation (MOO) provides the fundamental framework for addressing these multifaceted scenarios.

Unlike single-objective problems—where the superiority of one solution over another is straightforwardly determined by comparing scalar values—MOO lacks a single global optimum. Because the objectives are in conflict, improving the performance of one objective typically necessitates a trade-off, or a degradation, in at least one other. The resulting structure of optimal trade-offs is depicted in Figure 2.8.

Mathematical Formulation

Formally, a multi-objective optimisation problem seeks a decision vector $\mathbf{x}$ that optimises a vector-valued function of m distinct objectives. Assuming a minimisation paradigm

Table 2.2: Solution strategies grouped by the moment of preference articulation.

Strategy	Articulation of preference	Methodology & characteristics
A priori	Before optimisation	Aggregates objectives into a single scalar function using predefined weights (e.g. the Weighted Sum Method). Computationally inexpensive, but requires deep prior domain knowledge.
A posteriori	After optimisation	Generates a representative set of Pareto optimal solutions, leaving the final selection to the user. Multi-Objective Evolutionary Algorithms (MOEAs) such as NSGA-II are prominent examples.
Interactive	During optimisation	Iteratively involves the decision-maker to refine preferences and progressively guide the search toward the most desirable region of the Pareto Front.

without loss of generality, the problem reads

$$\min_{\mathbf{x}} F(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_m(\mathbf{x}))^\top, \quad (2.17)$$

subject to the constraints that define the feasible decision space Ω ,

$$g_j(\mathbf{x}) \leq 0, \quad j = 1, \dots, p; \quad h_k(\mathbf{x}) = 0, \quad k = 1, \dots, q, \quad (2.18)$$

where $\mathbf{x} = (x_1, x_2, \dots, x_n)^\top \in \mathbb{R}^n$ is the n -dimensional decision vector and $f_i(\mathbf{x}) : \Omega \rightarrow \mathbb{R}$ is the i -th objective function. The image of the feasible decision space Ω under F constitutes the objective space $\mathcal{Y} \subset \mathbb{R}^m$.

Pareto Dominance and Optimality

Because the components f_i are typically in conflict, standard relational operators cannot totally order the solutions in Ω . Instead, MOO relies on the fundamental concept of *Pareto dominance*. A decision vector $\mathbf{x}_1 \in \Omega$ is said to (strictly) dominate another vector $\mathbf{x}_2 \in \Omega$, denoted $\mathbf{x}_1 \prec \mathbf{x}_2$, if and only if

$$f_i(\mathbf{x}_1) \leq f_i(\mathbf{x}_2) \quad \forall i \in \{1, \dots, m\} \quad \text{and} \quad f_j(\mathbf{x}_1) < f_j(\mathbf{x}_2) \quad \text{for at least one } j. \quad (2.19)$$

A solution $\mathbf{x}^* \in \Omega$ is *Pareto optimal* if no other feasible solution dominates it. The set of all such optimal decision vectors is the *Pareto Set* $\mathcal{P}_S$, and its image in the objective space is the *Pareto Front*,

$$\mathcal{P}_F = \{ F(\mathbf{x}) \mid \mathbf{x} \in \mathcal{P}_S \}. \quad (2.20)$$

The ultimate goal of many MOO algorithms is to find a discrete set of solutions that approximates the true Pareto Front as closely and as diversely as possible (Figure 2.8), providing the decision-maker with a complete spectrum of optimal trade-offs.

Taxonomy of Solution Strategies

Approaches to solving MOO problems are broadly categorised according to *when* the decision-maker's subjective preferences are articulated relative to the optimisation process, as summarised in Table 2.2.

Evaluation Metrics for Approximate Fronts

When using a posteriori approximation methods, evaluating algorithmic performance is complex because it requires measuring two competing goals: *convergence* (proximity to the true theoretical Pareto Front) and *diversity* (uniform distribution and spread of the solutions across the front).

Hypervolume (HV). This metric calculates the m -dimensional volume of the objective space strictly dominated by the obtained approximation set, bounded by a predefined reference point $\mathbf{r}$ (the shaded region in Figure 2.8). It is highly regarded because it is strictly Pareto-compliant, capturing both convergence and diversity simultaneously.

Inverted Generational Distance (IGD). This metric computes the average Euclidean distance from a set of uniformly distributed reference points sampled from the true Pareto Front to their nearest respective solutions in the obtained set. A lower IGD value indicates better overall performance.

2.2.10 Formal Methods in Architecture Modelling

In the design of complex, safety-critical systems, the system architecture serves as the fundamental blueprint defining components, their interactions, and the overarching system behaviours. Architectures are traditionally modelled using semi-formal graphical languages such as UML (Unified Modeling Language) or SysML (Systems Modeling Language). While these languages excel at visualisation and stakeholder communication, they lack rigorous mathematical semantics. This ambiguity can allow critical design flaws, deadlocks, and safety violations to propagate undetected into the implementation phase.

To bridge this gap, *formal methods* employ discrete mathematics and logic to specify and verify system architectures. Event-B, an evolution of the classical B-method developed by Jean-Raymond Abrial, has emerged as a premier formal modelling language. It allows architects to mathematically prove that a system’s architectural design satisfies strict safety and functional properties *before a single line of code is written*.

Structural Foundations of Event-B

Event-B is a state-based formal method grounded in first-order logic and Zermelo–Fraenkel set theory. An Event-B model represents a system through a clear separation of concerns, dividing the specification into static and dynamic components, as summarised in Table 2.3 and illustrated in Figure 2.9.

Table 2.3: The two structural components of an Event-B model.

Component	Core elements	Architectural purpose
Context	Sets, Constants, Axioms, Theorems	Defines the <i>static</i> structural topology of the architecture (node types, immutable network configurations, system constants).
Machine	Variables, Invariants, Events	Defines the <i>dynamic</i> behavioural state of the architecture (component states, message passing, active connections).

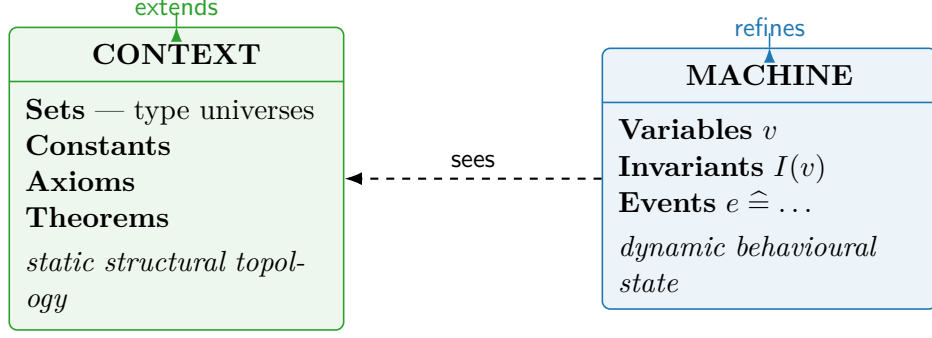


Figure 2.9: Structural foundations of an Event-B model.

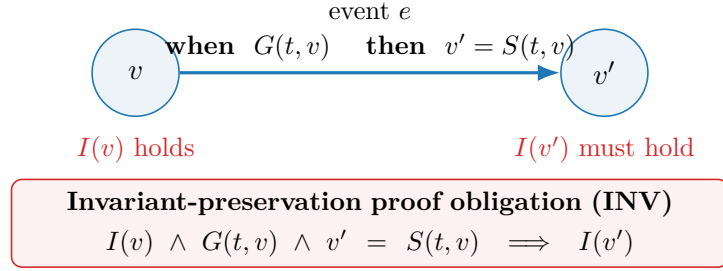


Figure 2.10: Semantics of a single event and its associated proof obligation.

Contexts can be extended by other contexts, and machines can *see* contexts to use their static definitions. This modularity aligns naturally with component-based system architecture.

Mathematical Formulation of State and Events

The dynamic state of an Event-B machine is represented by a vector of state variables v . The valid state space is constrained by an *Invariant* $I(v)$, a strictly defined logical predicate. In the architectural setting, the invariant formally expresses the critical safety properties that the architecture must never violate (e.g. mutual exclusion, resource limits, or secure state transitions).

State transitions are driven by *Events*. An event e models a specific architectural behaviour—such as a component fault or a message transmission—and is formally defined as

$$e \triangleq \mathbf{any} \ t \ \mathbf{where} \ G(t, v) \ \mathbf{then} \ v' = S(t, v) \ \mathbf{end}, \quad (2.21)$$

where t are the local parameters of the event, $G(t, v)$ is the *Guard*—a first-order predicate dictating exactly when the event is enabled—and $S(t, v)$ is the *Action* (generalised substitution) defining how the state transitions from v to its successor v' . This transition, together with the property the architecture must preserve, is depicted in Figure 2.10.

System Architecture through Stepwise Refinement

The most powerful architectural paradigm in Event-B is *stepwise refinement*. Designing a complex architecture in a single step is highly error-prone; Event-B instead adopts a top-down, *correct-by-construction* approach (Figure 2.11).

The architect begins with a highly abstract machine M_0 that models the core purpose of the system. Through refinement, new structural details, components, and communi-

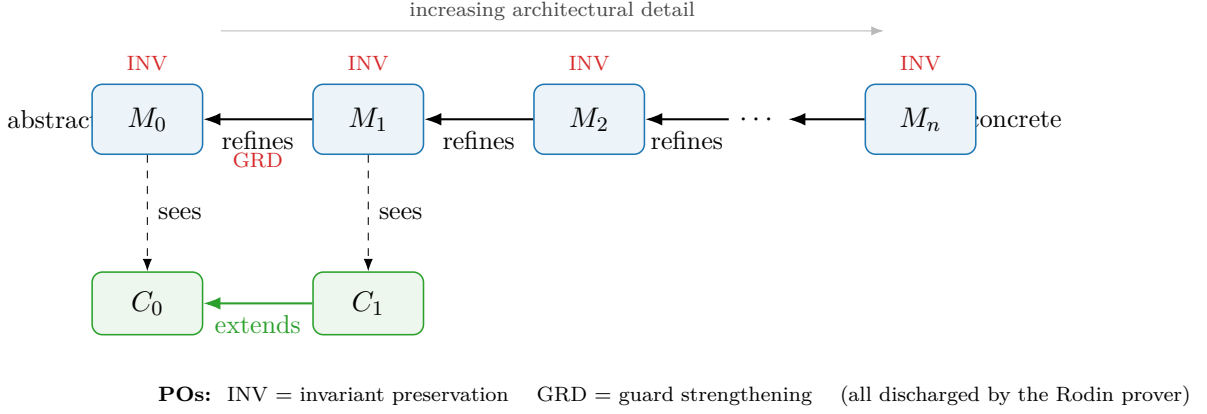


Figure 2.11: Stepwise refinement as a correct-by-construction architectural pipeline.

cation protocols are incrementally introduced in subsequent machines $M_1, M_2, \dots, M_n$. During this process, abstract variables may be replaced by more concrete architectural representations (*data refinement*), and abstract events may be decomposed into finer-grained interactions (*event refinement*).

Proof Obligations for Architectural Verification

The rigor of Event-B lies in the automated generation of *Proof Obligations* (POs): mathematical theorems that must be proven true—typically with automated provers such as the Rodin platform—to guarantee the consistency and correctness of the model. For an architecture to be deemed correct, two critical classes of PO must be satisfied.

Invariant preservation (INV). It must be proven that whenever the system is in a valid state (the invariant holds) and an enabled event is triggered (the guard is true), the resulting state still satisfies the invariant:

$$I(v) \wedge G(t, v) \wedge v' = S(t, v) \implies I(v'). \quad (2.22)$$

Guard strengthening (GRD). When an abstract architecture is refined into a concrete one, a concrete event must not occur under conditions in which its abstract counterpart would be forbidden. With abstract guard $G_A(v_A)$, concrete guard $G_C(v_C)$, and a gluing invariant $J(v_A, v_C)$, the obligation is

$$I(v_A) \wedge J(v_A, v_C) \wedge G_C(v_C) \implies G_A(v_A). \quad (2.23)$$

Mapping Architectural Semantics to Event-B

To apply Event-B to architecture modelling, practitioners map structural elements to formal constructs. System components are typically modelled as elements of defined sets, while architectural connectors and communication channels are modelled using mathematical relations or functions mapping sender components to receiver components. The verification process then mathematically guarantees that the proposed architecture is structurally sound, behaviourally consistent, and entirely free from the design faults targeted by the invariants.

2.3 Service Management in IoT

Emerging paradigms such as intent-based networking (IBN), AI-driven orchestration, and blockchain-enabled trust frameworks [58] are reshaping the landscape of service coordination. These innovations need the intelligent management of ephemeral service workflows—supporting predictive maintenance, automated trust negotiation, and cross-layer resilience—where microservices are dynamically orchestrated to enforce security policies and optimize resource utilization. Despite significant advances, existing IoT service management approaches exhibit paradigm fragmentation (e.g., isolated AI or optimization models), architectural rigidity, and unresolved trade-offs between scalability, security, and efficiency. This review proposed by Bemrah et al. [18] addresses these limitations by synthesizing insights across multiple paradigms and evaluating hybrid frameworks that enable integrated, autonomous, and resource-aware service orchestration in next-generation IoT ecosystems.

Specifically, we systematically examine the state-of-the-art in IoT service management, analyzing their strengths, limitations, and domain applicability. We establish a comprehensive taxonomy encompassing rule-based and model-driven methods, heuristic and metaheuristic optimization strategies, AI-driven learning techniques, and decentralized federated learning mechanisms. This review provides a consistent and richly detailed synthesis, capturing both foundational principles and cutting-edge innovations across diverse application domains. By consolidating recent advances, we identify critical research gaps and outline future directions—including quantum-resistant security models and privacy-preserving federated analytics—that serve as a foundation for scalable, resilient, and intelligent service management in latency-sensitive, large-scale IoT deployments.

IoT service management requires a multi-disciplinary strategy that balances real-time constraints, distributed architectures, energy efficiency, and security. Recent approaches span AI-based scheduling, federated learning, blockchain identity systems, and context-aware orchestration. Figure 2.12 represents a taxonomy of major techniques with representative studies.

2.3.1 Optimization-Based Approaches

These methods leverage mathematical modeling, heuristics and metaheuristic algorithms to find near-optimal solutions for complex IoT service management problems under dynamic, large-scale, and constrained environments

Deterministic and Heuristic Optimization Techniques

IoT service management often involves solving highly complex optimization problems, where deterministic methods guarantee reproducible exact solutions while heuristic and metaheuristic approaches offer efficient ways to reach near-optimal results in dynamic environments. For instance, Nabavizadeh et al. [66] develop a MILP model to optimize home healthcare delivery during COVID-19, improving caregiver routing, workload balancing, and uncertainty management in IoT-based scheduling. Similarly, heuristic extensions like Tang et al. [83]’s ISFLA enhance solution diversity and convergence through weighted inertia and advanced exploration, though scalability in real-time environments remains a challenge.

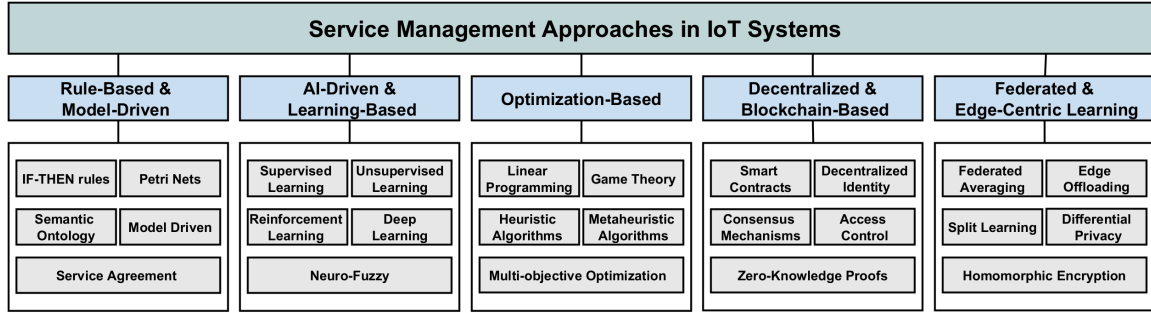


Figure 2.12: Taxonomy of Service Management Approaches and Techniques in IoT Systems.

Beyond scheduling, optimization techniques are applied to network reliability and resource allocation. For example, Chen et al. [28] introduce EDLoG, a real-time greedy detection mechanism for LoRaWAN IoT networks, ensuring reliable communication in critical domains such as intelligent transportation. In cloud and fog infrastructures, Masoori et al. [63] improve the efficiency of server allocation by refining the FirstFit algorithm, while Wibowo et al. [95] show that Round-Robin scheduling can significantly reduce latency and energy consumption in wireless sensor networks, thereby enhancing service responsiveness. More advanced metaheuristic approaches focus on multi-objective QoS optimization. Vakili et al. [85] propose a GWO–MapReduce framework that optimizes latency, cost, and reliability in cloud–IoT systems, while Liu and Deng [56] integrate Whale Optimization with a Genetic Algorithm for QoS-aware service discovery, improving both energy and access efficiency. Along similar lines, Bey et al. [24] design a quantum-inspired PSO for edge service placement, which improves energy use, delay, and load balancing simultaneously. Hybridization between heuristics and learning-based methods has also gained traction. For instance, Farhadi et al. [39] combine ACO with DRL to enable group-based service recommendations in Social IoT, boosting adaptability and personalization. Likewise, Dhaini et al. [34] propose EnergyNetAware2-SA, a simulated annealing–based VM placement model for fog computing, effectively reducing energy consumption while improving overall resource efficiency.

Deterministic and heuristic optimization techniques provide a rich toolkit for addressing IoT service management challenges, from scheduling and routing to QoS-aware composition and resource allocation. While deterministic approaches ensure correctness and rigor, heuristic and metaheuristic methods offer scalability and adaptability, though many still face limitations in energy efficiency and real-time responsiveness.

Strategic and Multi-Objective Decision Techniques

In IoT service management, many problems involve optimizing multiple conflicting objectives—such as latency, energy efficiency, cost, and reliability—simultaneously, which requires advanced multi-objective and strategic decision-making approaches. For example, Bandyopadhyay et al. [14] propose EdgeMatch, a preference-based scheduling mechanism that enables IoT devices and edge nodes to autonomously select optimal partners by balancing delay and resource criteria. Building on this trend of bio-inspired decision-making, Ait Hacène Ouhadda et al. [4] introduce DALOA, a lion-inspired algorithm designed for multi-objective QoS-aware service composition, which improves efficiency but struggles to adapt to highly dynamic distributed IoT systems.

Other works focus on strengthening QoS management and resource efficiency. Singh et al. [80] apply multi-objective optimization (MOO) techniques in IoT/WSN environments to enhance robustness and resource allocation at the edge, while Li et al. [52] integrate NSGA-II into a large language model-assisted operating system for dynamic service function chain deployment, thereby improving adaptability and service quality. Complementing these approaches, Sharma et al. [79] propose a hybrid framework combining fuzzy logic with NSGA-II, which increases solution diversity and convergence in QoS-aware service composition, although it lacks scalability and real-time responsiveness.

In summary, strategic and multi-objective optimization techniques provide powerful tools for balancing competing QoS requirements in IoT environments. While preference-based, evolutionary, and hybrid methods demonstrate strong adaptability and efficiency, challenges remain in ensuring scalability, responsiveness, and applicability to dynamic real-world deployments.

2.3.2 AI-Driven and Learning-Based Approaches

These techniques leverage artificial intelligence to enable adaptive, data-driven decision-making in dynamic and uncertain IoT environments, allowing systems to learn optimal behaviors from experience rather than relying solely on predefined rules.

Learning Techniques

Learning techniques aim to enable IoT services to autonomously improve performance and make intelligent decisions from data, moving beyond rigid rule-based methods toward adaptive, data-driven intelligence. For example, Dangore et al. apply a Random Forest-based strategy to improve prediction accuracy and adaptability in dynamic IIoT systems using real-time sensor data. Building on the same principle of learning from data, Alangari [9] address IoT security by combining K-means clustering for trust-based grouping with a hybrid Genetic-Firefly algorithm, thereby achieving secure and efficient routing. In a different application domain, Wadhwa et al. [87] demonstrate the effectiveness of DBSCAN for IoT-based weather forecasting, showing its ability to detect complex, non-linear patterns in spatially distributed data without requiring predefined cluster counts. Complementing these unsupervised and ensemble approaches, Hizal et al. [43] propose a deep learning-based intrusion detection system that integrates DNN, CNN, and LSTM models to enhance real-time DDoS attack detection in IoT networks through a two-level classification strategy.

Learning techniques provide powerful tools for extracting knowledge and optimizing performance across diverse IoT domains, from prediction and routing to forecasting and intrusion detection. However, while effective in specific tasks, they often lack adaptability to highly dynamic, multi-objective scenarios—limitations that motivate the use of adaptive and hybrid learning approaches, which are explored in the next subsection.

Adaptive and Hybrid Learning Techniques

Adaptive and hybrid learning techniques combine multiple algorithms to dynamically adjust to changing IoT environments, enhancing robustness, efficiency, and generalization across heterogeneous tasks. For instance, Abdulazeez and Askar [1] propose a hybrid fog offloading approach that integrates fuzzy logic for selecting the appropriate layer with a Deep Q-Network (DQN) for optimizing fog node allocation, thereby improving QoS and

balancing load. Along similar lines, Khansari and Sharifian [48] develop DRLMCF, a Deep Reinforcement Learning-based algorithm for decentralized microservice composition in serverless fog-based IoT systems, eliminating the need for expert tuning or labeled data. Other works exploit hybrid models to enhance automation and QoS prediction. For example, Harrabi et al. [42] design a multi-agent smart home system with contextual learning to support automation and energy management, though its dependency on advanced infrastructure constrains deployment in resource-limited environments. Similarly, Jithendra et al. [47] introduce ANFIS-COOT, a model that combines Adaptive Neuro-Fuzzy Inference Systems with COOT optimization for accurate QoS prediction in IoT web services. Extending this idea, Sundaram and Vijayan [81] present NFGOA, which integrates neuro-fuzzy clustering and genetic optimization to improve QoS in wireless sensor networks, achieving higher throughput, energy efficiency, and reliability. At a larger architectural scale, Abdulazeez and Askar [2] propose a hybrid offloading mechanism within an Edge-Fog-Cloud environment that combines fuzzy logic and DRL to reduce latency and optimize task execution, though its reliance on substantial computational resources limits adaptability in dynamic networks.

Overall, adaptive and hybrid approaches extend the strengths of single learning techniques by combining fuzzy logic, reinforcement learning, and evolutionary optimization, enabling improved scalability, energy efficiency, and QoS assurance in IoT systems. However, these solutions often face challenges in balancing computational overhead with real-time adaptability, highlighting the need for more efficient hybridization strategies.

2.3.3 Decentralized and Blockchain-Based Approaches

These approaches leverage distributed architectures and cryptographic consensus mechanisms to enable secure, transparent, and autonomous IoT service management, particularly suited for trustless and resource-constrained environments.

Decentralized Infrastructure and Trust Enforcement Techniques

Ensuring secure, transparent, and scalable service management is a critical challenge in IoT environments, where decentralized infrastructures such as blockchain offer promising solutions. To address consensus scalability and security, Chen et al. [30] propose DAG-D, a dual-DAG consensus protocol that improves throughput and communication efficiency in heterogeneous IIoT networks. Along similar lines, but with a focus on lightweight validation, Kim et al. [50] develop *PoA*², a Proof-of-Authority-and-Association protocol that achieves secure, low-latency, and energy-efficient transaction processing tailored to IoT constraints. Expanding the scalability dimension, Pang et al. [71] introduce TrustPay, a dual-layer blockchain framework leveraging REFEREE consensus to enhance transaction trust and support large-scale IoT service exchanges.

Complementing blockchain-based consensus methods, Mandaric et al. [61] propose a MAS-driven framework where agents collaboratively negotiate and adapt service delivery in real time. While this work demonstrates the benefits of distributed intelligence for dynamic IoT service environments, it highlights a common limitation of many decentralized infrastructures—namely, the lack of scalability strategies for large-scale deployments.

Secure Service Control and Privacy Assurance Techniques

As IoT ecosystems handle sensitive and large-scale data flows, ensuring secure service control and privacy-preserving mechanisms has become essential for trustworthy service management. To tackle these issues in resource-constrained environments, Zhang et al. [102] design a blockchain-based architecture optimized for lightweight decentralized processing, providing a secure foundation for IoT services. Building on access control, Zhang et al. [104] propose PE-RBAC, a privacy-enhanced system that integrates APSI and garbled Bloom filters to manage roles efficiently while safeguarding sensitive information. Extending this direction, Sareddy and Khan [78] introduce an AI-driven mHealth framework that combines HIBE, RBAC, and SMC to secure collaborative healthcare data access in IoT-driven medical applications. To further strengthen privacy and authentication, Pathak et al. [72] develop Z-PMA, a Zero-Knowledge Proof-based mutual authentication protocol integrated with permissioned blockchain, achieving both fast and scalable privacy-preserving service access.

Complementing these cryptographic and blockchain solutions, Arif et al. [12] propose a DRL-based routing model that not only enhances QoS and QoE but also supports threat detection through a layered architecture, though its computational demands and energy limitations constrain its deployment in resource-limited IoT systems.

Privacy-Preserving Mechanisms

Ensuring privacy in IoT service management is critical, particularly as federated and collaborative learning approaches introduce new risks of data leakage and security breaches. To address this, Pauu et al. [73] introduce an IRS-aided UAV framework that integrates federated learning with dynamic differential privacy, thereby enhancing both model accuracy and data protection in emergency communication scenarios. Building on the combination of federated learning and privacy-preserving techniques, Adnan et al. [3] develop a vehicular IoT (IoV) framework that couples federated learning with differential privacy, enabling accurate localization while safeguarding sensitive user data. Extending privacy protection to encrypted training, Manh et al. [62] propose a blockchain-based IoT security system that leverages homomorphic encryption, allowing AI models to detect cyberattacks in real time without compromising confidentiality.

Together, these approaches highlight the potential of combining federated learning, differential privacy, and encryption to protect sensitive data in IoT environments. However, balancing strong privacy guarantees with efficiency and scalability in highly dynamic, resource-constrained systems remains a pressing challenge for future research. In summary, secure service control and privacy assurance techniques—ranging from blockchain-based architectures and advanced cryptographic protocols to AI-driven security models—provide robust foundations for safeguarding IoT services. However, balancing strong security guarantees with scalability and energy efficiency remains a pressing challenge for real-world deployments.

2.3.4 Comparison & Discussion

The comparative analysis of IoT service management techniques represented in Table 2.4 reveals a clear trend toward hybrid, context-aware, and decentralized solutions that can scale with the increasing complexity and distribution of IoT systems. While rule-based and model-driven approaches offer foundational transparency and domain

Table 2.4: Comparative Analysis of Service Management Approaches in IoT Systems

Approach	Technique Types	Strengths	Limitations	Use Cases
Rule-Based & Model-Driven	IF-THEN rules [32], Petri Nets [77], MDD-based DSL [16], Semantic Ontologies [17]	Verifiable, interpretable, strong domain abstraction	Poor scalability, low adaptability to dynamic environments	Smart homes, agriculture, SLA analysis, digital twins
AI-Driven & Learning-Based	Random Forest [33], K-means & AHGFFA [9], DBSCAN [87], DNN/CNN/LSTM [43], Deep Q-Networks [1], DRL [48] DRL-MAS [19], ANFIS-COOT [47], NFGOA [81]	Adaptive, predictive, handles non-linear and large-scale data, anomaly detection	Data and compute-intensive require labeled data in many cases	Industrial IoT, weather forecasting, cyberattack detection, fog computing
Optimization-Based Methods	MILP [66], Greedy [28], FirstFit [63], Round-Robin [95], WO+GA [56], PSO [24], SA [34], ACO [39], Game Theory [14], MOO[80], NSGA2 [52]	Mathematical rigor balances multiple objectives (e.g. latency, cost, energy)	Solution quality vs. time trade-off, sometimes unsuitable for real-time tasks	Healthcare logistics, edge-fog resource allocation, smart cities, cloud-fog orchestration
Decentralized & Blockchain-Based	DAG-based consensus [30], TrustPay [71], PE-RBAC [104], HIBE+RBAC [78], ZKPs [72], Smart Contracts [102], PoA ² [50]	Trustless coordination, immutable logs, privacy-enhancing access control	Throughput and latency concerns, integration overhead	mHealth, consumer IoT, smart grids, blockchain-IoT fusion platforms
Federated & Edge-Centric Learning	SOH-FL [59], StatAvg [26], COMSPLIT [69], DQN-based Offloading [58], IRS-FL+DP [73], FL-DP [3], HE-based FL [62]	Privacy-preserving, scalable learning without data centralization	Coordination complexity affected by non-IID and poor connectivity	IoT security, smart cities, emergency response, connected vehicles

specificity, their static nature and limited adaptability underscore their unsuitability for rapidly evolving environments. In contrast, AI-driven and learning-based methods demonstrate significant and efficient advancements in adaptability, prediction, optimization and anomaly detection, particularly in dynamic contexts like industrial IoT and network security [18].

However, these methods often demand substantial labeled data and high computational resources, making them less viable for constrained edge devices. Optimization-based techniques, both deterministic and heuristic, show strength in structured resource allocation and SLA fulfillment but often struggle with real-time responsiveness and scalability. Emerging research on blockchain-integrated service management addresses trust, access control, and privacy through verifiable, decentralized protocols, yet concerns around latency and integration overhead persist, especially in low-power devices. Finally, federated and edge-centric learning marks a pivotal shift toward privacy-preserving intelligence at the edge, but challenges remain in managing data heterogeneity, device participation variability, and secure model aggregation. Collectively, these findings highlight a key research gap: the need for lightweight, multi-modal frameworks that blend semantic reasoning, learning adaptability, optimization efficiency, and trust-aware decentralization for IoT service management.

2.4 Multi-Agent Deep Reinforcement Learning for IoT

To enhance scalability, robustness, and distributed intelligence in complex IoT environments, Multi-Agent Systems (MAS) have been adopted as they enable multiple autonomous agents to collaboratively or competitively address tasks that exceed the capabilities of individual agents [54]. This decentralized approach suits IoT systems requiring

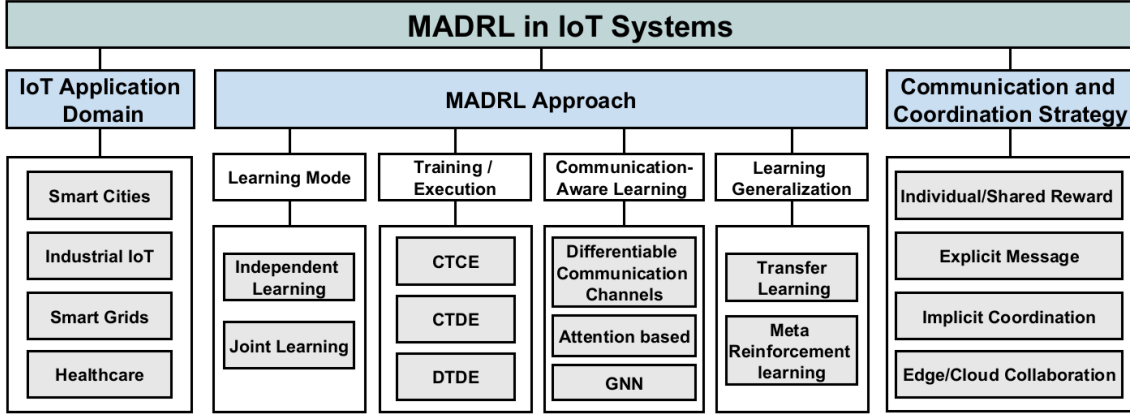


Figure 2.13: Taxonomy of MADRL in IoT Systems

local decisions under partial observability, but traditional MAS methods lack adaptive learning, limiting their responsiveness to dynamic conditions. [82]. Consequently, to overcome these limitations, Multi-Agent Deep Reinforcement Learning (MADRL) integrates the collaborative and distributed nature of MAS with the adaptive learning strength of DRL. MADRL offers a flexible framework for optimizing behavior across large-scale and dynamically evolving IoT systems. Its applications span a range of domains, including edge-cloud collaboration, adaptive routing, energy management, and security [35].

Despite the growing interest in MADRL for IoT, there remains a critical need for a unified, in-depth review that not only maps current advances but also systematically evaluates their effectiveness in addressing the core challenges of IoT environments. Existing works are often fragmented, limited to specific domains or techniques, and lack a comparative perspective across learning paradigms, coordination strategies, and application scenarios. We, Bemrah et al. [20] fill that gap by delivering a comprehensive and structured synthesis of the state of the art in MADRL for IoT. It categorizes the recent contributions across key IoT domains and contrasts major learning frameworks. Furthermore, it highlights unresolved issues in scalability, non-stationarity, and partial observability. By bridging theoretical foundations with practical applications, this review provides a critical reference point to develop next-generation IoT systems that are intelligent, distributed, adaptive, and resilient.

This subsection explores MADRL approaches in IoT by examining its applications. Figure 2.13 presents a comprehensive taxonomy of MADRL in IoT systems Bemrah et al. [20], categorizing approaches by domain applicability, learning models, and communication paradigms. This visual overview highlights the diverse methods and innovations applied to enable intelligent, adaptive, and distributed control within IoT ecosystems, offering a structural lens to understand current trends and research directions.

2.4.1 Domain-Specific Applications of MADRL in IoT

MADRL has shown substantial potential across various IoT application domains by enabling distributed, adaptive, and scalable decision-making. This subsection outlines key areas where MADRL is effectively applied.

Smart Cities

Smart cities increasingly depend on intelligent, decentralized systems to manage urban infrastructure, ensure efficient resource utilization, and support sustainable development. To this end, Vishnoi et al. [86] propose PS-DC-DDQN, a MADRL-based framework designed for RIS-assisted, NOMA-enabled UAV-to-D2D communications in 6G environments, which enhances resource allocation and energy efficiency in complex urban networks. Complementing this, Moudoud et al. [65] introduce a privacy-preserving framework that combines multi-agent federated learning with DRL to secure Internet of Drones (IoD) networks, where a reputation-aware scheduler prioritizes reliable UAVs while minimizing communication overhead.

Focusing on urban mobility, Antonio and Maria-Dolores [11] present adv.RAIM, a MADRL-based autonomous intersection manager trained via self-play, which enables connected and autonomous vehicles (CAVs) to coordinate traffic flows without traditional signaling systems, improving safety and efficiency. Similarly, Bansal et al. [15] address service deployment in smart urban environments through UrbanEnQoSPlace, a DRL-based service placement framework that optimizes multiple QoS metrics using a dueling deep Q-network. While effective in improving latency, reliability, and efficiency, this approach struggles with multi-user scalability and high computational demands.

Overall, these works demonstrate the potential of MADRL and hybrid learning frameworks to optimize communication, security, mobility, and service placement in smart cities. However, scalability, real-time adaptability, and computational efficiency remain open challenges in deploying these solutions at metropolitan scale.

Industrial Internet of Things (IIoT)

The Industrial Internet of Things (IIoT) connects machines, sensors, and controllers in industrial environments to enable intelligent automation, predictive control, and real-time optimization. In this domain, MADRL has emerged as a powerful tool for handling dynamic task allocation and resource management. For instance, Wang et al. [92] propose a two-stage MADRL framework leveraging MAPPO and attention mechanisms to optimize task offloading in IIoT systems, significantly reducing latency and energy consumption in edge computing environments. Building on similar principles, Ling et al. [53] address dense SIIoT scenarios by integrating attention-based coordination with prioritized learning, which enhances load balancing and improves task completion success rates under high-demand conditions.

Together, these studies illustrate the promise of MADRL and hybrid learning techniques for addressing the complexity of industrial IoT service management. Nonetheless, balancing scalability, computational efficiency, and real-time responsiveness remains a critical barrier to their deployment in large-scale industrial environments.

Smart Grids

Smart grids are evolving into intelligent energy systems where distributed generation, demand response, and decentralized control must be managed dynamically. MADRL has become a key enabler in this context, allowing autonomous agents to coordinate energy flows, optimize resource allocation, and ensure grid stability. For example, Alferidi et al. [10] propose a DRL-based system for smart microgrids that enables real-time device scheduling and peer-to-peer energy trading, effectively lowering electricity costs and

reducing dependence on centralized grids. Complementing this, Wang et al. [93] develop a MADRL-driven local control strategy for large-scale PV inverter scheduling, where inference-aided actor structures and composite loss optimization improve decentralized coordination across distributed energy resources.

Together, these approaches demonstrate the potential of MADRL to enhance flexibility, resilience, and cost-efficiency in smart grid management, though challenges remain in scaling to ultra-large networks and maintaining robust performance under uncertain demand and supply conditions.

Healthcare

Healthcare systems represent highly dynamic, sensitive, and decentralized environments where effective decision-making requires both adaptability and strong privacy guarantees. MADRL offers promising capabilities to address these challenges by enabling distributed agents to collaborate in real time while ensuring security and efficiency. For instance, Sahoo et al. [75] enhance federated reinforcement learning for medical image diagnosis by improving fairness and enabling personalized convergence across heterogeneous clients, thus tackling the variability of patient data. Building on this need for secure and adaptive solutions, Al-Marridi et al. [5] integrate blockchain with deep multi-agent Q-learning to optimize decision-making, reduce latency, and strengthen privacy in decentralized healthcare networks.

Together, these works highlight the transformative role of MADRL in healthcare, demonstrating its potential to improve diagnostic accuracy, decision efficiency, and trust, though challenges remain in balancing scalability with strict privacy and resource constraints.

2.4.2 Learning Approaches in MADRL for IoT

The design of learning architectures in Multi-Agent Deep Reinforcement Learning (MADRL) plays a crucial role in shaping agent coordination, learning stability, convergence speed, and the overall effectiveness of deployed policies in IoT environments. This subsection presents a structured overview of core MADRL learning paradigms and their relevance to the decentralized, dynamic nature of IoT systems.

Independent vs. Joint Learning

Independent and joint learning strategies form the foundation of coordination paradigms in MADRL, offering different trade-offs between autonomy and cooperation. In Independent Learning, each agent treats other agents as part of the environment and learns its own policy $\pi_i(a_i|s_i)$ based solely on local observations and rewards [40]. This method supports scalability and simplicity, allowing agents to operate autonomously in loosely coupled environments.

In contrast, Joint Learning emphasizes inter-agent cooperation. Agents coordinate their actions, share observations or learned parameters, and optimize a joint policy or value function. This collaborative strategy improves performance in tightly coupled or interdependent tasks common in IoT environments, such as coordinated sensing or cooperative control [88]. Leveraging shared knowledge enables agents to converge faster and perform more efficiently in complex, multi-agent tasks.

Overall, the choice between independent and joint learning depends on the nature of agent interactions in IoT systems, with independent learning favoring scalability and simplicity, while joint learning provides superior performance in highly collaborative or interdependent scenarios.

Centralized Training with Decentralized Execution (CTDE)

A key challenge in MADRL for IoT lies in reconciling the need for scalable execution with the benefits of stable, sample-efficient training. To address this, the Centralized Training with Decentralized Execution (CTDE) paradigm has emerged as the dominant approach.

During training, agents can access global state and joint actions to learn centralized critics (e.g., $Q_i(s, \mathbf{a})$), which significantly improves stability and convergence [91]. At run-time, however, each agent relies only on its local observation o_i and policy $\pi_i(a_i|o_i)$ to act independently, ensuring scalability in distributed IoT environments. Despite its advantages, CTDE can introduce considerable communication overhead in resource-constrained networks. In contrast, Decentralized Training and Decentralized Execution (DTDE) eliminates this overhead by relying solely on local learning, but it suffers from degraded performance in highly coupled environments where coordination is essential [100]. To overcome these limitations, hybrid strategies that combine centralized critics with decentralized actors have been proposed, offering a robust balance between accuracy and scalability that is well-suited to heterogeneous and latency-sensitive IoT systems [54]. Along these lines, Nouredine et al. [70] proposed a DRL-enhanced agent-based architecture for real-time, autonomous IoT decision-making. Their approach demonstrated significant improvements in latency, resource utilization, and adaptability, illustrating the potential of hybrid learning strategies. However, it also highlighted the persistent challenge of training overhead, which remains a critical barrier in resource-constrained IoT settings.

In summary, while CTDE provides a powerful learning paradigm for MADRL in IoT, its practical deployment requires careful balancing of coordination benefits against communication costs, motivating continued exploration of lightweight hybrid strategies.

Communication-Aware Learning

In highly dynamic IoT environments, where tasks are interdependent and devices operate under heterogeneous and resource-constrained conditions, effective communication among agents becomes a cornerstone of robust coordination. To address these challenges, MADRL frameworks increasingly integrate communication-aware mechanisms that enable agents to exchange goals, observations, and policies in real time.

Differentiable Communication Channels represent one such solution, allowing agents to learn both when and what to communicate through backpropagation-compatible mechanisms. By filtering out irrelevant messages, these methods enhance communication efficiency and significantly reduce bandwidth usage in dynamic IoT networks [46]. Building upon this principle of selective sharing, **Attention-Based Message Filtering** introduces mechanisms to prioritize important signals while suppressing noise, thereby strengthening cooperation under resource constraints or network congestion [53]. Complementing these selective communication strategies, **Graph Neural Networks (GNNs)** provide a structured way to model inter-agent interactions as dynamic graphs. By enabling scalable reasoning, structured message passing, and policy synchronization, GNNs

support more advanced forms of decentralized coordination and task decomposition across IoT agents [82].

Taken together, these approaches illustrate a clear evolution from simple message exchange to intelligent, structured, and context-aware communication, highlighting the importance of communication-aware learning for achieving scalability, adaptability, and efficiency in IoT service management.

Learning Generalization

Ensuring robust generalization is a critical challenge for MADRL agents deployed in heterogeneous IoT environments, where variations in network dynamics, topologies, and task distributions require adaptability beyond task-specific optimization. To address this, research has primarily advanced along two directions: transfer learning and meta-reinforcement learning.

Transfer Learning (TL) enables agents to reuse knowledge from prior tasks or domains, thereby accelerating learning in new environments. For example, Cho et al. [31] propose a distributed DQN framework with daily model transfer to optimize power control in solar-powered small cells, demonstrating the efficiency of knowledge reuse across time-varying conditions. Complementing this, Namiki et al. [68] develop a stable transfer mechanism designed to mitigate negative transfer, effectively reducing learning time in multi-agent control tasks such as base station sleeping control. Together, these approaches illustrate how TL can reduce training overhead while maintaining performance in evolving IoT scenarios.

Meta-Reinforcement Learning (Meta-RL), often described as “learning to learn,” equips agents with meta-learned priors that allow rapid adaptation to novel tasks. For instance, Hossain et al. [44] introduce an MRL-based task offloading and power control mechanism in MEC environments, enabling fast adaptation and reduced power consumption in both binary and partial offloading schemes. Similarly, Faramarzi et al. [38] apply a meta-learning-enhanced DRL algorithm to UAV-mounted active RIS systems, achieving efficient dynamic resource management in highly variable aerial networks. These studies highlight the strength of Meta-RL in enabling agents to generalize across diverse and rapidly changing IoT contexts.

Overall, by combining knowledge reuse through TL with rapid adaptability via Meta-RL, MADRL frameworks are becoming increasingly capable of handling the diversity and unpredictability of real-world IoT service management scenarios.

2.4.3 Communication and Coordination Strategies in MADRL

In MADRL-based IoT systems, effective communication and coordination among agents are critical to ensuring scalable, stable, and intelligent behavior across distributed environments. This subsection outlines key strategies that facilitate inter-agent interaction and decision synchronization in IoT contexts.

Shared vs. Individual Rewards

Reward design plays a pivotal role in guiding agent behavior within MADRL frameworks, directly influencing coordination, scalability, and learning efficiency in IoT environments.

The central debate lies in whether agents should optimize shared rewards or individual rewards, as each strategy brings distinct advantages and trade-offs.

Shared rewards align agent objectives by promoting collective performance, thereby encouraging cooperation and improving global outcomes in interconnected IoT tasks [55]. However, while this approach fosters strong collaboration, it may also lead to credit assignment issues, where individual contributions to the overall performance become difficult to distinguish. Conversely, individual rewards incentivize agents to optimize their own performance, enabling specialization and independent decision-making [90]. This can enhance scalability in loosely coupled environments but risks creating conflicts when global coordination is required.

In practice, the choice between shared and individual rewards often depends on task structure: tightly coupled IoT applications benefit from shared incentives, whereas heterogeneous or weakly coupled scenarios are better suited to individual ones. Hybrid reward schemes that combine both paradigms are emerging as promising solutions, balancing cooperation with specialization to ensure efficiency and fairness in complex IoT ecosystems.

Explicit Message Passing vs. Implicit Coordination

Communication is a cornerstone of multi-agent interaction in IoT systems, and MADRL frameworks typically adopt either explicit message passing or implicit coordination strategies. The choice between these approaches strongly influences scalability, energy efficiency, and robustness in dynamic environments.

In explicit message passing, agents directly exchange observations, beliefs, or policy parameters to align decisions in real time. For example, Zhang et al. [101] address communication constraints in dynamic IoT networks by introducing a DNA-based multi-agent reinforcement learning algorithm, which allows each UE to optimize its offloading strategy using historical data while still benefiting from message exchanges. By contrast, implicit coordination reduces communication overhead by enabling agents to learn cooperative behaviors without direct message passing. Li et al. [51] exemplify this approach with a value-based MADRL algorithm, where IoT devices independently learn offloading policies from experience data, achieving lower energy consumption while maintaining coordination.

Overall, explicit communication provides more precise coordination but increases bandwidth and energy costs, whereas implicit approaches scale better in resource-constrained IoT systems but may suffer from slower convergence. Hybrid strategies that balance both paradigms are increasingly explored as a way to combine the strengths of efficiency and collaboration.

Edge–Cloud Collaboration

IoT systems increasingly rely on hierarchical architectures that combine cloud platforms, edge servers, and local devices. MADRL frameworks can effectively exploit this structure by distributing computation, learning, and decision-making across layers, thereby improving scalability, responsiveness, and resource efficiency.

For instance, Giannopoulos et al. [41] propose COOLER, a latency-aware MADRL framework that enables cooperative offloading decisions across edge and cloud layers. By optimizing local, horizontal, and vertical computation flows, their approach reduces task drop rates and latency in the edge–cloud continuum. Extending this idea, Xu et al.

Table 2.5: Comparative Analysis of IoT Multi-Agent DRL Applications and Approaches

Specific Aspect	Works Methodologies & Innovations	Advantages	Challenges	Domain Affinity
Smart Cities	RIS-assisted NOMA UAV-to-D2D (PS-DC-DDQN) [86], Privacy-preserving FL/DRL with reputation [65], Signal-free CAV coordination [11].	Dynamic resource allocation, Reduced comms overhead, Autonomous traffic flow.	High environmental dynamics, Security vulnerabilities, Heterogeneous agents.	Urban infrastructure, Mobility.
Industrial IoT	Attention-based MAPPO for offloading [92], Prioritized learning in dense deployments [53].	Latency/energy reduction, Improved load balancing, Scalable coordination.	Real-time constraints, Safety-critical ops, Partial observability.	Manufacturing, Supply chain.
Smart Grids	P2P energy trading systems [10], Inference-aided actor structures for PV control [93].	Reduced grid dependency, Cost optimization, Decentralized coordination.	Energy volatility, Large-scale coordination, Cyber-physical threats.	Energy distribution, Microgrids.
Healthcare	Fairness-aware federated MRL [75], Blockchain-integrated Q-learning [5].	Privacy preservation, Personalized convergence, Secure decision-making.	Data sensitivity, Regulatory compliance, Device heterogeneity.	Medical diagnostics, Patient monitoring.
CTDE	Hybrid centralized critics [91], Decentralized actors [54].	Policy consistency, Mitigated non-stationarity, stable training.	Single-point failure risk, limited post-deployment adaptability.	Cross-domain.
Communication Aware	Differentiable comm channels [46], Attention-based filtering [53], GNN-structured messaging [82].	Bandwidth optimization, Adaptive resilience, Noise suppression.	Reward design complexity, Coordination-quality trade-offs.	Bandwidth-constrained networks.
Generalization	Transfer learning (daily model transfer) [31, 68], Meta-RL (rapid adaptation) [44, 38].	Reduced retraining, Cross-domain adaptation, Fast deployment.	Catastrophic forgetting, Over-generalization risks.	Dynamic environments.
Comm. & Coordination	Shared Rewards [55], Individual Rewards [90].	Shared: strong cooperation, Individual: device-specific optimization.	Shared: credit assignment, Individual: selfish behavior risks.	IIoT (shared); Healthcare (individual).
Explicit vs. Implicit Coord.	DNA-based MADRL (explicit) [101], Value-based independent learning (implicit) [51].	Explicit: precise coordination, Implicit: low comms overhead.	Explicit: bandwidth intensive, Implicit: sub-optimal decisions.	Safety-critical (explicit), large-scale (implicit).
Edge-Cloud Collaboration	Latency-aware COOLER framework [41], Dynamic blockchain + R-MADRL [97].	Latency optimization, hierarchical intelligence, trustworthy processing.	Synchronization complexity, cross-layer security risks.	Latency-sensitive applications.

[97] integrate a dynamic blockchain (DBC) scheme with a rotating multi-agent DRL (R-MADRL) algorithm to jointly optimize task offloading, resource allocation, and consensus in end-edge collaborative IIoT systems. Their solution enhances trustworthy processing efficiency and reduces task timeout rates, showcasing the benefits of secure and coordinated edge-cloud interactions. These studies highlight that edge-cloud collaboration not only improves latency and task allocation but also strengthens scalability and fault tolerance. By ensuring that agents can continue operating under communication disruptions, decentralized architectures preserve service reliability and responsiveness in critical IoT applications such as smart grids, healthcare, and industrial automation.

Together, these communication and coordination strategies define the structural axes along which MADRL algorithms can be tailored for specific IoT applications. The optimal choice depends on task complexity, inter-agent coupling, bandwidth availability, and real-time execution demands. Carefully balancing these strategies enables the deployment of robust, adaptive, and efficient MADRL-powered IoT systems.

2.4.4 Comparison & Discussion

This section synthesizes and critically compares the findings presented in previous sections, highlighting key trends, trade-offs, and unresolved challenges in applying MADRL to IoT domains. The analysis focuses on three core dimensions: application domains, learning approaches, and communication/coordination strategies shown in Table 2.5 of [20]. MADRL enables domain-specific IoT optimizations—smart cities prioritize latency-aware coordination, healthcare emphasizes privacy-preserving individualized rewards, IIoT requires safety-driven explicit messaging, and smart grids leverage edge-cloud stability—yet faces fundamental trade-offs between CTDE’s training stability and real-time adaptability. Communication strategies further reveal tensions: explicit coordination ensures precision at high bandwidth costs, while implicit methods sacrifice optimality for scalability, with edge-cloud collaboration (e.g., COOLER/R-MADRL) emerging as a balancing mechanism. Critical gaps persist in real-world validation, adversarial security for centralized components, and certifiable safety for rapid adaptation techniques like Meta-RL. Ultimately, cross-domain success hinges on orchestrating specialized learning, coordination, and infrastructure layers within deployment-ready frameworks.

2.5 MAS-DRL for IoT Service Composition

In IoT environments, service composition requires balancing multiple QoS criteria such as latency, reliability, and energy efficiency under dynamic conditions. Traditional approaches often lack the adaptability and intelligence needed to handle such complexity. Multi-agent systems (MAS) provide distributed coordination and autonomy, while deep reinforcement learning (DRL) enables continuous optimization through experience. Their integration offers a powerful framework for context-aware, QoS-driven service composition. This section explores how MAS–DRL combinations advance IoT service management beyond conventional methods.

2.5.1 Service Composition with MAS and DRL

Service composition in IoT must adapt to highly dynamic conditions, where fluctuating resources, heterogeneous devices, and diverse QoS requirements create significant challenges for traditional methods. Recent studies highlight the integration of Multi-Agent Systems (MAS) with Deep Reinforcement Learning (DRL) as a powerful paradigm for enabling adaptive, autonomous, and scalable service orchestration. By combining the distributed intelligence of MAS with the learning capabilities of DRL, these frameworks optimize service selection, execution, and coordination under uncertainty—making them highly relevant for next-generation IoT applications. Recent research has explored the integration of MAS and DRL to enhance the performance, adaptability, and efficiency of IoT service composition under dynamic conditions. In this part of studying the related work, we survey approaches combining MAS coordination with DRL learning, embodying the convergence of autonomy and intelligence required for next-generation IoT systems. The selected works highlight cooperative multi-agent architectures and hybrid learning models that seek to balance scalability, energy efficiency, and QoS. Zeng et al. [99] proposed Multi-D3QN, a multi-strategy DRL method using Double DQNs to enhance service composition accuracy and adaptability, effectively balancing QoS and cost, though limited by high training costs and untested scalability. Wang et al. [89] proposed a DRL-based

adaptive service composition framework designed for dynamic and partially observable environments. By integrating recurrent neural networks and a heuristic behavior selection strategy, their method enhances scalability, adaptability, and efficiency in large-scale IoT service composition under uncertain conditions. Boutarfa et al. [25] introduced a cooperative multi-agent approach for service composition in multi-cloud environments, integrating DQL to jointly optimize service selection and composition. Their method effectively addresses the NP-hard nature of multi-cloud service composition by enabling localized decision-making and achieving optimal combinations of atomic services across heterogeneous clouds.

Zhuo et al. [105] proposed a DRL-based service composition strategy for Cyber-Physical-Social Systems using Proximal Policy Optimization. By customizing the action space and refining the reward function, it enables adaptive orchestration under varying QoS constraints and surpasses conventional DRL and heuristic methods in scalability and efficiency, though its applicability to complex real-time CPSS remains limited. Similarly, Chen et al. [29] introduced a DQN-based model for smart manufacturing, integrating noise networks and prioritized experience replay to balance exploration and convergence. Despite improved performance, its evaluation does not fully reflect real-world IoT conditions. Yi et al. [98] presented a pretraining-and-policy-based deep reinforcement learning approach for QoS-aware service composition, integrating maximum likelihood estimation and policy scoring to balance exploration and exploitation. They demonstrated that the method efficiently finds near-optimal compositions with superior performance compared to other algorithms. Ma and Xu [60] proposed SkyDRL, combining skyline optimization with advanced DQN techniques to improve energy efficiency and QoS in multi-cloud environments. While achieving significant energy savings, it lacks QoS prediction and multi-agent coordination mechanisms. Zhang et al. [103] introduced a MADRL approach for collaborative edge-cloud offloading in Augmented Intelligence of Things vehicular environments. By integrating caching mechanisms and clustered edge server coordination, the approach significantly improves offloading success rates, reduces latency, and enhances overall system resource utilization. Baek and Ko [13] introduced a multidimensional multi-agent federated reinforcement learning framework for optimizing IoT services in dynamic environments. By employing a parallel neural structure for independent cluster learning and an environment-centric communication protocol for context interpretation, MultiFedRL enhances adaptability and user satisfaction, and improved training efficiency over federated learning approaches.

Together, these works demonstrate that MAS-DRL approaches are transforming IoT service composition by addressing adaptability, scalability, and QoS-awareness in ways traditional methods cannot. However, challenges such as training efficiency, computational overhead, and real-world validation remain open. This makes MAS-DRL not only a promising research direction but also a cornerstone for future IoT ecosystems, where autonomous, intelligent service orchestration will be essential for enabling smart cities, healthcare, manufacturing, and beyond.

2.5.2 MADRL and GAT-based Service Composition

Mohajer et al. [64] proposed a graph-attentive multi-agent TD3 framework for SLA-aware service offloading and orchestration that improves QoS, reduces delay, and enhances resource utilization under dynamic edge traffic conditions. Xiao et al. [96] proposed MACNS, a GNN-enhanced PPO-based multi-agent reinforcement learning framework for

collaborative navigation that improves route optimization, adaptability, and convergence in dynamic environments. Huang et al. [45] proposed a graph attention-based collective reinforcement learning (GAT-CRL) approach that enables knowledge collaboration among distributed DRL agents for efficient and adaptive resource allocation in 6G-enabled IoT environments. Fan et al. [36] presented a multi-agent graph reinforcement learning framework with attention mechanisms for service restoration in active distribution systems, improving resilience and coordination under partially observable conditions. Cao et al. [27] proposed a Web service recommendation method that combines heterogeneous graph attention networks with FiBiNET to model feature interactions and improve recommendation accuracy and QoS-aware ranking. Liu et al. [57] proposed ADSAC, an attention-diffusion-aided DRL approach for QoS-aware multi-AIGC service orchestration in edge networks, improving user utility and system reliability under resource constraints. Fan et al. [37] proposed a graph-based heterogeneous multi-agent reinforcement learning approach that leverages GAT and multi-head self-attention to enhance coordination and achieve efficient service restoration in distribution systems.

2.5.3 MADRL Service Optimization with MO and GAT

We introduced DRL-MAS Bemrah et al. [19], a hybrid framework that combines multi-agent systems with deep reinforcement learning to enhance adaptability and resource efficiency in distributed IoT environments. The framework achieves decentralized optimization of service composition while maintaining stringent Quality of Service requirements. Then **Hi-MAD** significantly extends our previous work Bemrah et al. [19] by introducing a hierarchical architecture of DRL agents, enhancing and refining the underlying DRL models, enabling efficient data distribution across edge and fog nodes, and ensuring greater resilience through advanced coordination and adaptive decision-making mechanisms. And we propose a hybrid service composition framework **GAMMA** that integrates MADRL, multi-objective optimization, and GAT to jointly address scalability, adaptability, and complex service dependencies. MADRL enables decentralized decision-making across distributed agents, while multi-objective optimization ensures Pareto-efficient trade-offs among conflicting QoS metrics such as latency, cost, and reliability. In addition, GAT captures structural relationships within heterogeneous service graphs, allowing agents to focus on the most relevant interactions. This combined design improves convergence, solution quality, and generalization in dynamic large-scale service composition environments compared to existing single-method approaches.

2.5.4 Comparison and Discussion

The reviewed literature highlights a growing convergence between Multi-Agent Systems (MAS) and Deep Reinforcement Learning (DRL) for IoT service composition, alongside increasing interest in formal methods to ensure correctness and reliability in such complex environments. While each approach contributes valuable strengths, persistent limitations open important opportunities for further advancement. Table 2.6 highlights that existing approaches only partially address the key requirements of IoT service composition. While several works incorporate MADRL or GAT-based modeling, they often lack a unified treatment of context-awareness, multi-objective optimization, QoS guarantees, energy efficiency, and scalability. For instance, some studies focus primarily on QoS optimization or energy efficiency but overlook contextual filtering or graph-based

relational modeling. Others leverage GAT or MADRL individually, yet fail to integrate them with multi-objective selection strategies or scalable IoT-specific composition mechanisms. Moreover, clustering-based context discovery and comprehensive MO formulations are largely underexplored across the literature:

- *IoT Service Composition:* Studies such as Tang et al. [83] and Sharma et al. [79] confirm the central role of service composition in enabling flexibility, context-awareness, and efficient functionality within IoT systems. However, existing approaches often fall short in handling dynamic service variability and integrating seamlessly across heterogeneous environments, particularly when scalability and resource constraints are critical.
- *MAS-based Approaches:* Contributions including Berrani et al. [23], Harrabi et al. [42], and Mandaric et al. [61] demonstrate that MAS frameworks excel at managing distributed interactions, supporting adaptability, and tailoring services in real time. Yet, these solutions frequently depend on advanced infrastructure and still lack robust strategies for scaling efficiently or addressing energy-aware operation in resource-limited IoT deployments.
- *DRL-based Optimization:* DRL frameworks, such as those by Zeng et al. [99], Wang et al. [89], and Ma and Xu [60], highlight the potential of learning-driven optimization to enhance QoS, accelerate convergence, and improve adaptability under uncertainty. Nonetheless, their widespread adoption is hindered by computationally expensive training, limited support for multi-agent coordination, and insufficient integration of contextual or resilience-aware mechanisms.
- *Hybrid and Cooperative Strategies:* Efforts like Boutarfa et al. [25] and Zhuo et al. [105] explore hybrid models that combine MAS, DRL, or heuristic methods to address multi-objective optimization. These hybrid approaches often outperform single-method solutions in flexibility and scalability, but they still face gaps in rigorous real-time validation and large-scale deployment readiness.

Overall, the comparative analysis indicates that while MAS and DRL offer complementary strengths—distributed intelligence on one hand and adaptive optimization on the other—their integration remains underexplored in practice. Most existing frameworks address isolated aspects of service composition but fall short in delivering holistic solutions that can balance scalability, energy efficiency, and resilience in dynamic IoT environments. Bridging this gap calls for novel hybrid architectures, real-world validation in edge/fog contexts, and deeper attention to resource-awareness, making this an especially promising and impactful area for future research.

As shown in Table 2.6, most existing approaches fail to simultaneously address the full spectrum of critical factors—including QoS, context-awareness, energy efficiency, scalability, and resilience—within a unified framework. In the table, ● indicates full support for a criterion, ◐ indicates partial or limited consideration, and ○ indicates no support or mention of the criterion in the respective work. Many existing models remain narrowly focused on a single dimension (e.g., QoS optimization or scheduling efficiency) or are constrained by high computational costs and restrictive infrastructural assumptions. To overcome these shortcomings, our proposed framework integrates a hierarchical MAS architecture with DRL-based optimization across three operational layers. This design

not only ensures adaptive and resilient service composition but also explicitly incorporates energy constraints and supports decentralized execution in resource-constrained IoT environments. Moreover, the framework introduces a dual optimization mechanism that combines context discovery with QoS-aware service selection, enabling holistic decision-making that dynamically adapts to both environmental and user requirements.

	Work	Year	IoT System	Edge/Fog	SC	QoS	Context	MAS	DRL	Energy	Resilience	Scalability
1	Toman et al. [84]	2023	●	○	●	●	○	○	○	○	○	●
2	Tang et al. [83]	2024	●	○	●	●	○	○	○	●	○	○
3	Ait Hacène Ouhabda et al. [4]	2024	●	○	●	○	○	○	○	◐	○	○
4	Sharma et al. [79]	2024	●	○	●	●	○	○	○	○	○	○
5	Vakili et al. [85]	2024	●	○	●	●	○	○	○	○	○	●
6	Berrani et al. [23]	2018	●	○	●	◐	◐	●	○	○	○	●
7	Harrabi et al. [42]	2024	●	○	◐	○	◐	●	○	◐	○	○
8	Mandaric et al. [61]	2024	●	○	◐	○	●	●	○	○	○	○
9	Nouredine et al. [70]	2021	●	○	○	○	○	◐	●	○	○	●
10	Bansal et al. [15]	2022	●	○	◐	●	○	○	●	○	○	○
11	Abdulazeez and Askar [2]	2024	○	●	○	●	○	○	●	○	○	○
12	Wang et al. [94]	2024	●	◐	○	○	○	○	●	○	○	○
13	Arif et al. [12]	2024	●	○	○	●	○	○	●	◐	○	○
14	Zeng et al. [99]	2021	○	○	●	●	○	○	●	○	○	●
15	Wang et al. [89]	2019	○	○	●	●	○	○	●	○	○	○
16	Boutarfa et al. [25]	2022	○	○	●	○	○	●	●	○	○	○
17	Zhuo et al. [105]	2023	○	○	●	●	○	●	○	○	○	●
18	Chen et al. [29]	2023	○	○	●	●	○	●	○	○	○	○
19	Yi et al. [98]	2022	●	○	●	●	○	○	○	○	○	○
20	Ma and Xu [60]	2023	○	○	●	●	○	○	●	●	○	●
21	Hi-MAD	–	●	●	●	●	●	●	●	●	●	●

Table 2.6: Comparison of related works on MAS-based IoT services across key criteria.

In conclusion, this comparative analysis underscores the novelty and strength of our contribution. Unlike prior works, our framework unifies adaptability, resilience, scalability, and energy efficiency into a single, coherent solution, positioning it as a comprehensive and practical approach for managing complex, real-world IoT service ecosystems.

2.6 Conclusion

This chapter mapped IoT service-composition literature into five methodological families, introduced a taxonomy for MAS-DRL in IoT, and compared existing approaches across key criteria such as QoS, energy–latency trade-offs, privacy, and resilience. From

this analysis, we distilled design principles and identified open challenges that directly motivate the technical contributions developed in the following chapters. Specifically, our contributions focus on IoT system modeling in edge/fog environments, managed through a MADRL framework with a hierarchical architecture designed to optimize performance while ensuring quality of service.

Chapter 3

IoT System Modeling and Conceptualization

3.1 Introduction

Traditional cloud-centric solutions often fail to meet these requirements due to communication delays and scalability limitations. To address these challenges, edge and fog computing have emerged as paradigms that extend computation and decision-making closer to end devices, enabling low-latency service delivery, improved scalability, and better resource utilization. However, the distributed and dynamic nature of edge/fog-based IoT systems demands intelligent management strategies capable of adaptive resource allocation, service composition, and coordination across multiple layers of the architecture. In this context, Multi-Agent Deep Reinforcement Learning (MADRL) provides a powerful framework for enabling decentralized decision-making and cooperative optimization. By leveraging agents that learn policies through interaction with the environment, MADRL can support scalability, robustness, and adaptability in complex IoT ecosystems.

This chapter introduces the modeling of IoT systems under edge/fog architectures and presents a MADRL-based hierarchical management framework. The framework is designed to optimize system performance while ensuring quality of service (QoS), balancing computation, communication, and energy efficiency across heterogeneous layers. The chapter first details the system modeling approach, then elaborates on the hierarchical architectural design, and finally explains how MADRL is applied to manage resources, coordinate agents, and adapt to dynamic IoT scenarios.

The remainder of this chapter is organized as follow; Section 3.2 describe the layered proposed architecture. Section 3.3 details the IoT system modeling across the five layers. Section 3.4 presents the use case for the system modeling in the smart city. And Section 3.5 concludes the chapter.

3.2 IoT System Agent-based Architecture

To manage the complexity and dynamic nature of large-scale IoT environments, the proposed approach by Bemrah et al. [21] relies on a formal IoT system model that combines hierarchical decision-making with distributed intelligence. The IoT environment is modeled as a Multi-Agent Partially Observable Markov Decision Process (MPOMDP) within a Multi-Agent Deep Reinforcement Learning framework, enabling decentralized

and coordinated decision-making among heterogeneous IoT entities. In this setup, agents operate under partial observability, learning adaptive control policies through shared experiences and limited communication, which together ensure scalable, intelligent, and context-aware management across the IoT ecosystem.

The IoT system is defined formally as $IoT = \langle D, N, C, MAD, SVC \rangle$, where:

- D denotes the set of IoT devices,
- N signifies the set of edge-fog computing nodes,
- C represents the underlying communication network assuring interactions,
- MAD denotes the multi-agent DRL governed by **Hi-MAD**. It empowers agents to learn, collaborate, and make joint decisions, dynamically optimizing performance metrics; and
- SVC indicates the provided services.

Our proposed modeling architecture structures the IoT ecosystem into five coordinated layers that collectively ensure scalability, energy efficiency, and resilience through hierarchical interaction among devices, edge/fog nodes, and intelligent agents.

3.3 IoT Environment Modeling

Figure 3.1 illustrates the multi-layer architecture of the proposed **Hi-MAD** framework within a typical IoT ecosystem. The architecture is organized into five layers. At the Physical Layer, various IoT devices and sensors—such as temperature, humidity, air quality, and carbon detectors—collect real-time environmental data. The Edge-Fog Layer performs local processing and filtering through edge nodes, while fog nodes handle intermediate computation and storage to reduce latency and communication load. The Network Layer manages data transfer and connectivity between nodes using standard communication protocols. The Platform Layer hosts the Hierarchical MADRL framework, composed of high, mid, and low-level agents responsible for adaptive decision-making and coordination. Finally, the Application Layer includes end-user domains such as smart homes, healthcare, industry, and traffic systems, where our frameworks enable intelligent, efficient, and context-aware management across diverse IoT services.

3.3.1 The Physical Layer

The physical layer corresponds to the set of IoT devices D where each IoT device $d_i \in D$ acts as a data-producing or service-requesting endpoint (e.g., sensors, actuators, and smart thermostats). These devices supply raw sensor data to edge and fog nodes for further processing and decision-making. Additionally, they generate service requests based on contextual information or predefined user objectives. Devices are dynamically managed and reconfigured through the MAS framework, allowing them to respond effectively to environmental changes and ensure coordinated system-wide performance aligned with global objectives. D is defined as $d_i = \langle ID, Sens, Act, Dat \rangle$, where:

- ID : A set of unique identifiers distinguishing each device within the system.

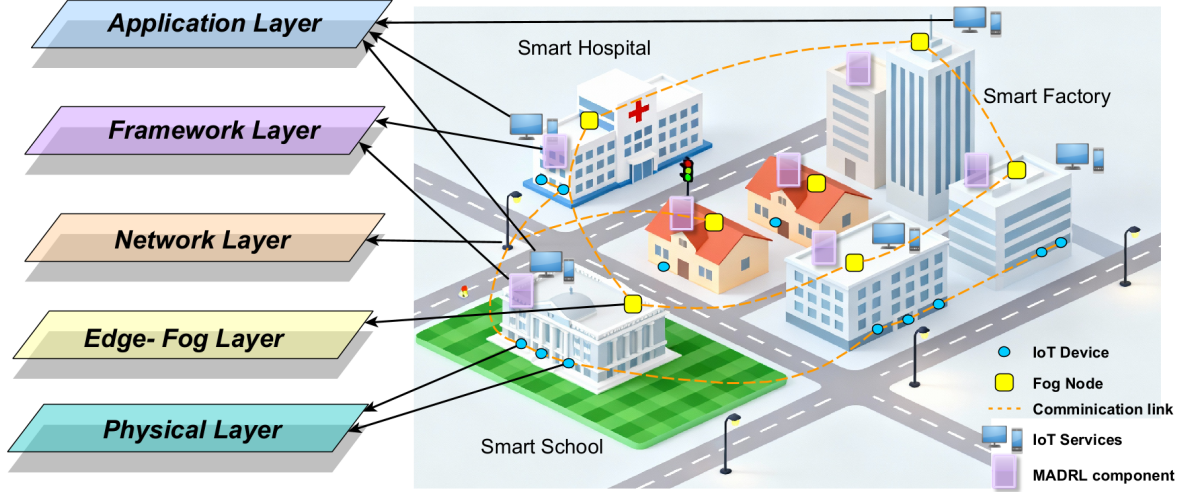


Figure 3.1: MAS-based IoT System Architecture.

- $Sens = \{sens_1, sens_2, \dots, sens_n\}$: A set of sensors, with each sensor $sens_i$ collecting specific environmental data.
- $Act = \{act_1, act_2, \dots, act_m\}$: A set of actuators, with each actuator act_j performing defined actions in response to system commands or established policies.
- $Dat(t) = \{dat_1(t), dat_2(t), \dots, dat_n(t)\}$: Raw data acquired by the device at time t , encompassing unprocessed measurements, anomalies, or irrelevant data points.

3.3.2 The Edge-Fog Layer

The edge-fog layer corresponds to the set of computing nodes N that are pivotal in IoT environments, providing essential localized computing and storage capabilities. Positioned up the device layer, these nodes handle real-time processing of sensor and device data, thereby enabling rapid response times and minimizing reliance on centralized cloud infrastructure. They perform preliminary analyses, data filtering, and contextual processing, forwarding pertinent information to higher layers for comprehensive decision-making and analytics. They are represented as $N = \langle ID, L, I, F, P, S_{host}, O \rangle$, where:

- ID : A set of unique identifiers distinguishing each node within the system.
- L : Physical or virtual location information of the edge and fog servers, critical for latency management.
- I : A set of input data, with each element $i \in I$ representing raw data sourced from IoT devices at edge nodes and from agents at fog nodes.
- F : A registry of available local processing functions within the node, which can be selectively applied to the input data I .
- $P = F'(I)$: A local processing operation composed of selected functions $F' \subseteq F$, applied sequentially to input data I : $F' = f_k \circ f_{k-1} \circ \dots \circ f_1$, where $f_i \in F$ for all $i = 1, \dots, k$.

- S_{host} : Set of deployed services selected from the global service repository, comprising IoT-specific services for edge nodes and composite services for fog nodes.
- O : Output data generated by processing function P , including transformed and preprocessed data ready for further analytics or higher-level decision-making.

3.3.3 The Network Layer

The network layer corresponds to the set of communication links C , representing the set of communication channels between IoT devices D , nodes N , and agents A to facilitate their interactions. A communication link is modeled as $C \subseteq (D \cup N \cup A) \times (D \cup N \cup A)$. These links rely on lightweight protocols such as MQTT, CoAP, HTTP, and gRPC, and support communication patterns including Publish/Subscribe and Request/Response, which are well-suited for decentralized and distributed architectures. The exchanged data includes service requests, performance metrics, configuration updates, and policy-related information. Ensuring low-latency, fault-tolerant communication is especially critical between edge and fog layers to maintain real-time responsiveness and system resilience. This layer ensures reliable, low-latency data exchange and fault-tolerant connectivity among devices, nodes, and agents. Maintaining robust communication across heterogeneous infrastructures is vital for sustaining scalability and real-time coordination.

3.3.4 The Platform Layer

The platform layer hosts the Multi-Agent DRL framework MAD , serving as the core intelligence layer in the distributed IoT architecture. It orchestrates collaborative learning and decision-making among autonomous agents, organized hierarchically into high, mid, and low-level roles with distinct decision scopes. These agents continuously learn, adapt, and optimize system performance. The MADRL-based framework is defined as $MAD = \langle A, G, Mgt, Obj \rangle$ where:

Autonomous Agents (A): they are stratified into functional tiers, each with distinct responsibilities that collectively enable adaptive and intelligent IoT system behavior. Organized into three categories: the *Low-Level* agents continuously monitor and report the operational states of devices and nodes. The *Mid-Level* agents dynamically select, compose, and adapt services based on contextual information and QoS requirements. The *High-Level* agents coordinate with peer agents to oversee global objectives such as service continuity, energy optimization, and system resilience. Each agent operates based on partial local observability, learning through environmental feedback and inter-agent interactions. An agent is defined as $A_i = \langle ID, role, S_{local}, \pi_{\theta_i}, DRL_{Algo}, R_i \rangle$ where:

- ID : A unique identifier assigned to each agent.
- $role$: The agent's hierarchical function—*Low-Level*, *Mid-Level*, or *High-Level*.
- S_{local} : The agent's local view of the environment, comprising observations and performance metrics.
- π_{θ_i} : The agent's policy, modeled as a neural network parameterized by weights θ_i , mapping observations O_i to actions A_i : $\pi_{\theta_i}(O_i) = A_i$. For a global system containing many agents, each agent learns an optimal global policy π_l^* that maximizes the expected cumulative discounted reward over time $\pi_l^* = \arg \max_{\pi} \mathbb{E} [\sum_{t=0}^{\infty} \gamma^t r_t]$ where:

- π_t : The agent’s policy function, mapping observed states to actions.
- γ : The discount factor, which balances short-term and long-term rewards.
- r_t : The immediate reward received at time t .
- DRL_{Algo} : The deep reinforcement learning algorithm employed, such as *MATD3*, *MAPPO*, *MADDPG*, or *DDQN*.
- R_i : The reward function that guides the agent’s learning process by encoding task-specific objectives and constraints.

Service Interaction Graph ($\mathbf{G}$): the multi-agent system operates over a dynamic directed graph $G = \langle V, \mathcal{E}, X \rangle$ where:

- $V = \{v_1, \dots, v_N\}$: the set of nodes, each representing an active service agent or IoT resource.
- $E \subseteq V \times V$: the set of directed edges encoding service dependencies, composition relationships, or communication links, derived from each service’s **Comp** field and updated dynamically at each time step.
- $X \in \mathbb{R}^{N \times d}$: the node feature matrix, where the i -th row $x_i \in \mathbb{R}^d$ encodes the QoS attribute vector of service v_i (ResponseTime, Availability, Throughput, ...).

The graph structure is rebuilt at each episode to reflect dynamic service availability, topology changes, and fault events, ensuring that agent policies remain conditioned on up-to-date relational context.

Management Functions ($\mathbf{Mgt}$): autonomous agents execute decentralized management and orchestration functions to provide intelligent, distributed control in IoT environments. By interacting continuously with their surroundings, they adapt their policies in real time to handle resource limits and varying service demands, achieving local responsiveness while maintaining global coordination and efficiency. The set of core management functions is defined as $Mgt = \{\text{monitoring}, \text{decision_making}, \text{adaptation}\}$ where:

- **monitoring**: Continuous observation of the system state, including node health, service performance, and network conditions. It provides agents with real-time feedback on resource utilization, anomaly detection, and operational metrics.
- **decision-making**: Intelligent orchestration of actions such as task allocation, service composition, routing, and migration. Decisions are informed by real-time data and system constraints to optimize resource distribution and meet QoS targets.
- **adaptation**: Dynamic reconfiguration of policies and operational parameters in response to performance feedback, environmental dynamics, or user-specific requirements. This function ensures system resilience, scalability, and long-term efficiency.

Objectives and Optimization Goals ($\mathbf{Obj}$): the overarching goal of our system is to enable intelligent and adaptive optimization of key operational functions in distributed IoT environments. To this end, we define five interrelated and mutually reinforcing objectives:

- *Service Orchestration*: Dynamically deploy, compose, and coordinate services across heterogeneous edge and fog nodes in response to evolving system and environmental conditions.

- *Energy Management*: Minimize aggregate energy consumption—particularly for resource-constrained devices—without compromising system performance or responsiveness.
- *System Stability*: Sustain consistent performance under fluctuating workloads and environmental variability, ensuring predictable behavior.
- *Resilience*: Maintain fault tolerance and enable rapid recovery in the presence of hardware failures, network disruptions, or degraded resources.
- *User Requirement Satisfaction*: Deliver services in alignment with user-specified constraints, such as latency thresholds, cost ceilings, and QoS preferences.

3.3.5 The Application Layer

The application layer corresponds to the set of services SVC , dynamically composed in response to user requests and the current state of the system, ensuring that service delivery aligns with contextual needs. These services are flexibly deployed across available edge/-fog nodes, enabling localized execution and minimizing latency. Deployment decisions are guided by mechanisms that ensure QoS compliance and efficient resource utilization, allowing the system to adapt to changing demands while maintaining performance and reliability across the distributed infrastructure. Each service $svc_i \in SVC$ represents an atomic or composite IoT functionality defined as $svc_i = \langle ID, Req, QoS, Comp \rangle$ where:

- *ID*: A unique identifier assigned to each service svc_k , ensuring distinct referencing within the system.
- *Req*: The computational and resource requirements of the service, including CPU cycles, memory allocation, and latency budget.
- *QoS*: The desired Quality of Service profile, specifying performance metrics such as reliability, throughput, availability, or energy efficiency.
- *Comp*: The composition structure indicating whether the service is atomic or composite. This is represented as a dependency graph with a list of required sub-services for composite services.

Together, these layers create a unified architecture where data moves upward from sensing to decision-making, and control flows downward from intelligence to action. This interaction allows **Hi-MAD** to operate in a context-aware, resilient, and energy-efficient manner, even under changing conditions.

3.4 Smart City use case

We consider a smart city district with several smart buildings (e.g., office towers, a hospital, a shopping mall, and a residential complex). The goal of the IoT system is to optimize energy consumption across the entire district, minimize costs, reduce strain on the main power grid, and maintain occupant comfort, all while addressing real-time changes such as fluctuating energy prices and varying occupancy levels.

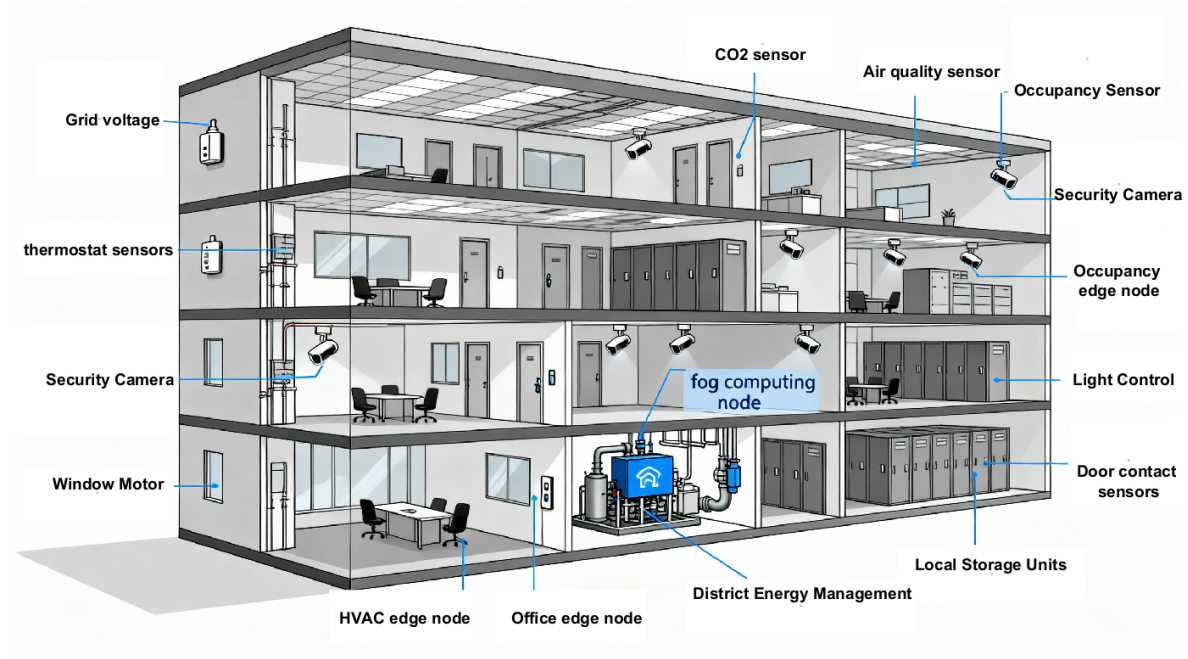


Figure 3.2: Smart Building

3.4.1 Smart Building Devices

In our scenario, each building is equipped with thousands of these devices, continuously collecting raw data.

Office HVAC Device OFFICE-BLDG-A-FL00R12-HVAC25

Sensors: temperature, humidity, occupancy

Actuators: compressor, chiller, burner, fan-speed, damper

Data: {"temp-c": 22.5, "humidity": 45, "occupancy": true, "power": "on", "mode": "heat", "fan-speed": "medium", "damper-position": 75, "unit": "percent", "timestamp": "2024-10-27T14:30:00Z"}

Smart Light Device ROOM-BLDG-A-FL00R15-LIGHT5

Sensors: ambient-light, motion, color

Actuators: dimmer, RGB-LED controller, color-temperature

Data: {"ambient-light": 300, "motion-detected": true, "color-temp": 4000, "brightness": 80, "state": "on", "unit": "lux", "timestamp": "2024-10-27T14:40:00Z"}

Humidity Device OFFICE-BLDG-A-FL00R20-HUM10

Sensor: capacitive-humidity

Actuators: humidifier, dehumidifier

Data: {"humidity": 35, "target-humidity": [45, 50], "power": "on", "unit": "percentRH", "timestamp": "2024-10-21T14:30:00Z"}

CO₂ (Carbon Dioxide) Device OFFICE-BLDG-A-FL00R12-CO2-25

Sensor: Non-Dispersive Infrared (NDIR)

Actuators: HRV, damper, thermostat, occupancy alert

Data: {"co2-ppm": 1200, "fan-speed": "high", "bypass": false, "window-lock": "unlock", "window-position": 25, "led-color": "red", "buzzer": "beep-once", "timestamp": "2024-10-27T14:30:00Z"}

Particulate Matter (PM) Device OFFICE-BLDG-A-FL00R12-PM-5

Sensors: laser scattering PM, particle count, VOC

Actuator: air purifier

Data: {"pm1-0": 8, "pm2-5": 22, "pm10": 34, "particles-0-3um": 15000, "particles-0-5um": 8500, "voc": 220, "fan-speed": 4, "mode": "auto", "power": "on", "unit": "g/m³", "timestamp": "2024-10-27T14:05:00Z"}

Energy Measurement Device OFFICE-BLDG-A-FL00R12-ENER27

Sensors: current transformer, voltage, power factor, frequency

Actuators: relay, audio-visual alert, communication

Data: {"current-a": 5.2, "voltage-v": 240.1, "power-factor": 0.98, "frequency-hz": 59.9, "alert": "high-usage", "cost-this-hour": 1.20, "timestamp": "2024-10-27T14:05:00Z"}

3.4.2 Smart Building Fog and Edge Nodes

Each smart building has its own fog computing node related to local fog and edge nodes. The building's sensors primarily communicate with these nodes.

Floor-HVAC Edge Node EDGE-OFFICE-BLDG-A-FL00R12

Location: Building A, Floor 12, IT Room 12B (GPS)

Inputs: Raw sensor data from all 25 zones on Floor 12

Functions: data-aggregate(), anomaly-detect(), local-policy-enforce(), occupancy-analyze()

Processing functions: $P = \text{local-policy-enforce} \circ \text{occupancy-analyze} \circ \text{data-aggregate}(I)$

Services deployed: zone-comfort-service, energy-monitoring-service, occupancy-tracking-service

Output: {"floor-avg-temp": 23.1°C, "total-occupancy": 45, "energy-usage": 12.5kW, "anomalies-detected": 0}

Building-Level Fog Node FOG-OFFICE-BLDG-A-MAIN

Location: Building A, Basement (Data Center Room)

Inputs: Aggregated data from all edge nodes + directives from Building Agent

Functions: load-forecast(), qos-optimize(), negotiate-energy(), fault-diagnose()

Processing functions: $P = \text{negotiate-energy} \circ \text{qos-optimize} \circ \text{load-forecast}(I)$

Services deployed: building-energy-optimizer, predictive-maintenance-service, grid-interface-service
Data Snapshot: {"predicted-peak-load": 850kW, "optimal-energy-schedule": 70.5kW, "maintenance-alerts": "fair"}

3.4.3 Smart Building Services

Services are deployed across the edge-fog continuum based on latency and resource needs related to several smart building.

Temperature Control (Atomic Service) ZONE12-TEMPERATURE-CONTROL

Resource Profile: CPU: 0.5 cores, RAM: 256MB, Latency: < 50ms

QoS Metrics: Availability: 90%, Response Time: < 150ms, Reliability: 99%

Type: Atomic Service

Energy Management (Composite Service) BLDG-A-ENERGY-MANAGER

Resource Profile: CPU: 4 cores, RAM: 8GB, Latency: < 2s

QoS Metrics: Availability: 99.99%, Energy Efficiency: optimized, Throughput: 1000 req/s

Type: Composite Service

Service Composition: {Data Aggregation Service, Load Prediction Service, Grid Communication Service, Policy Management Service}

3.4.4 Smart Building Communication links

These are the channels enabling interaction, using appropriate protocols for each link.

- **Device-to-Edge ($D \times N$)**
 - **Link:** (OFFICE_BLDG_A_FLOOR12_ZONE3.temperature_sensor, EDGE_OFFICE_BLDG_A_FLOOR12)
 - **Protocol:** MQTT
 - **Pattern:** Publish/Subscribe
 - **Data:** Sensor readings published to topic: "buildingA/floor12/zone3/temp"
- **Edge-to-Fog ($N \times N$)**
 - **Link:** (EDGE_OFFICE_BLDG_A_FLOOR12, FOG_OFFICE_BLDG_A_MAIN)
 - **Protocol:** gRPC
 - **Pattern:** Request/Response
 - **Data:** Aggregated metrics, service health status, local decisions
- **Fog-to-Agent ($N \times A$)**
 - **Link:** (FOG_OFFICE_BLDG_A_MAIN, Building_A_Agent)
 - **Protocol:** HTTP/REST
 - **Pattern:** Request/Response + Publish/Subscribe
 - **Data:** Processed observations, policy updates, coordination messages
- **Agent-to-Agent ($A \times A$)**
 - **Link:** (Building_A_Agent, Grid_Operator_Agent)
 - **Protocol:** AMQP
 - **Pattern:** Message Queuing
 - **Data:** Energy negotiation messages, load reduction agreements, price signals
- **Fog-to-Cloud ($N \times \text{Cloud}$)**
 - **Link:** (FOG_OFFICE_BLDG_A_MAIN, FOG_OFFICE_BLDG_B_MAIN)
 - **Protocol:** gRPC
 - **Pattern:** Service-to-Service
 - **Data:** Coordinated load shedding plans, shared resource allocation

3.5 Conclusion

In conclusion, this chapter introduced the modeling of IoT systems under edge/fog architectures and laid the foundation for their intelligent management. Building in this model, we propose in the next chapter, a hierarchical MADRL framework building on this model that enables context-aware service composition while optimizing performance and ensuring QoS through adaptive coordination of computation, communication, and energy resources across heterogeneous layers.

Chapter 4

Hierarchical and Graph-Attention MADRL Frameworks for IoT Service Optimization

4.1 Introduction

The growing complexity and dynamic nature of IoT networks necessitate intelligent, adaptive frameworks to ensure scalability, resilience, and strict Quality of Service (QoS) compliance. To address this, Multi-Agent Systems (MAS) offer a decentralized approach for autonomous, collaborative decision-making [61]. Furthermore, Deep Reinforcement Learning (DRL) enables agents to continuously learn optimal, context-aware service orchestration policies [49]. Crucially, these intelligent strategies must prioritize energy efficiency to sustain battery-powered and resource-constrained devices [76, 74].

This chapter presents the two complementary frameworks developed in this thesis for the autonomous management and composition of services in large-scale IoT systems: **Hi-MAD** in [21], a hierarchical MADRL framework for resilience and service optimization, and **GAMMA** in [22], a graph-attention, multi-objective MADRL framework for QoS-aware service composition. Although the two frameworks were designed with different primary objectives—Hi-MAD emphasizing hierarchical coordination, resilience, and energy efficiency, and GAMMA emphasizing relational modeling of inter-service dependencies and Pareto-optimal QoS selection—they share a common conceptual backbone. Both adopt CTDE paradigm, both decompose the service management problem into a discovery, selection, and composition pipeline, and both rely on a family of policy-gradient and value-based learners (MAPPO, MADDPG, MATD3, and DDQN). The chapter first establishes the shared foundations that unify the two designs, then develops each framework in turn, and finally discusses how the hierarchical perspective of Hi-MAD and the graph-attentive perspective of GAMMA reinforce one another.

The remainder of this chapter is organized as follows: Section 4.2 describe the overall hierarchical architecture and graph based. Section 4.3 presents the high level operates in the cloud to ensure global coordination, focusing on resilience, energy efficiency, and system stability. Section 4.4, the mid level, deployed in the fog, handle context discovery, QoS-based service selection, and service composition, simplifying the environment into smaller through a two-step reduction process. Section 4.5, the low level, deployed at the edge, multiple agents control individual IoT devices, using shared rewards during centralized training and acting independently during decentralized execution.

4.2 Hierarchical Architecture Overview

To achieve robust and adaptive management of complex Internet of Things ecosystems, we developed a dual-paradigm architecture. As illustrated in Figure 4.1, this approach integrates two highly complementary frameworks **Hi-MAD** and **GAMMA**, that target distinct architectural dimensions: IoT infrastructure components and IoT service orchestration, respectively. While they address different levels of the IoT environment, both frameworks are anchored by an identical, centrally depicted operational pipeline. They diverge significantly, however, in their structural organization and multi-agent coordination strategies, establishing a comprehensive defense against network complexity.

At the core of the proposed architecture lies a shared, sequential pipeline that drives decision-making for both infrastructure and service management. This pipeline ensures that whether the system is managing physical devices or logical services, it follows a standardized optimization trajectory:

1. **Context Discovery:** Agents first perceive and map the current state of the environment, identifying available resources, active services, and environmental constraints.
2. **QoS-Aware Service Selection:** Based on the discovered context, the system evaluates and filters options according to strict QoS requirements, ensuring that only viable candidates proceed.
3. **Service Composition:** Finally, the selected elements are orchestrated into cohesive, optimized structures (either deployed physical configurations or logical service chains) to execute complex user requests.

As depicted on the left side of the architectural diagram, **Hi-MAD** (Hierarchical Multi-Agent Deep Reinforcement Learning) governs the physical and structural components of the IoT system. Deployed across the network hierarchy, spanning the Cloud, Fog, Edge, and Device layers. Hi-MAD organizes its intelligent agents according to their specific decision scope. The diagram highlights this vertical stratification, showing High, Mid, and Low Agents distributed across the topology. To handle the immense complexity of a sprawling network, Hi-MAD employs a two-step environmental reduction process that partitions the global state into smaller, manageable sub-environments tailored to localized optimization tasks. Within this hierarchical structure, Hi-MAD executes the shared operational pipeline using three specialized DRL models.

Conversely, the right side of the diagram illustrates **GAMMA** (Graph Attention Multi-Objective based MADRL), which operates strictly at the logical service level. Rather than relying on a static vertical hierarchy, GAMMA organizes agents horizontally across a dynamically structured service interaction graph. Because IoT services constantly fluctuate based on mobility and demand, GAMMA mediates multi-agent coordination through **attention-weighted message passing**. This Graph Attention Network approach allows agents to dynamically weigh the importance of their neighboring nodes and continuously adapt to changing service topologies. To fulfill the shared pipeline at this service level, GAMMA leverages:

- **Multi-head attention coupled with binary K-means clustering** for Context-Aware Discovery.

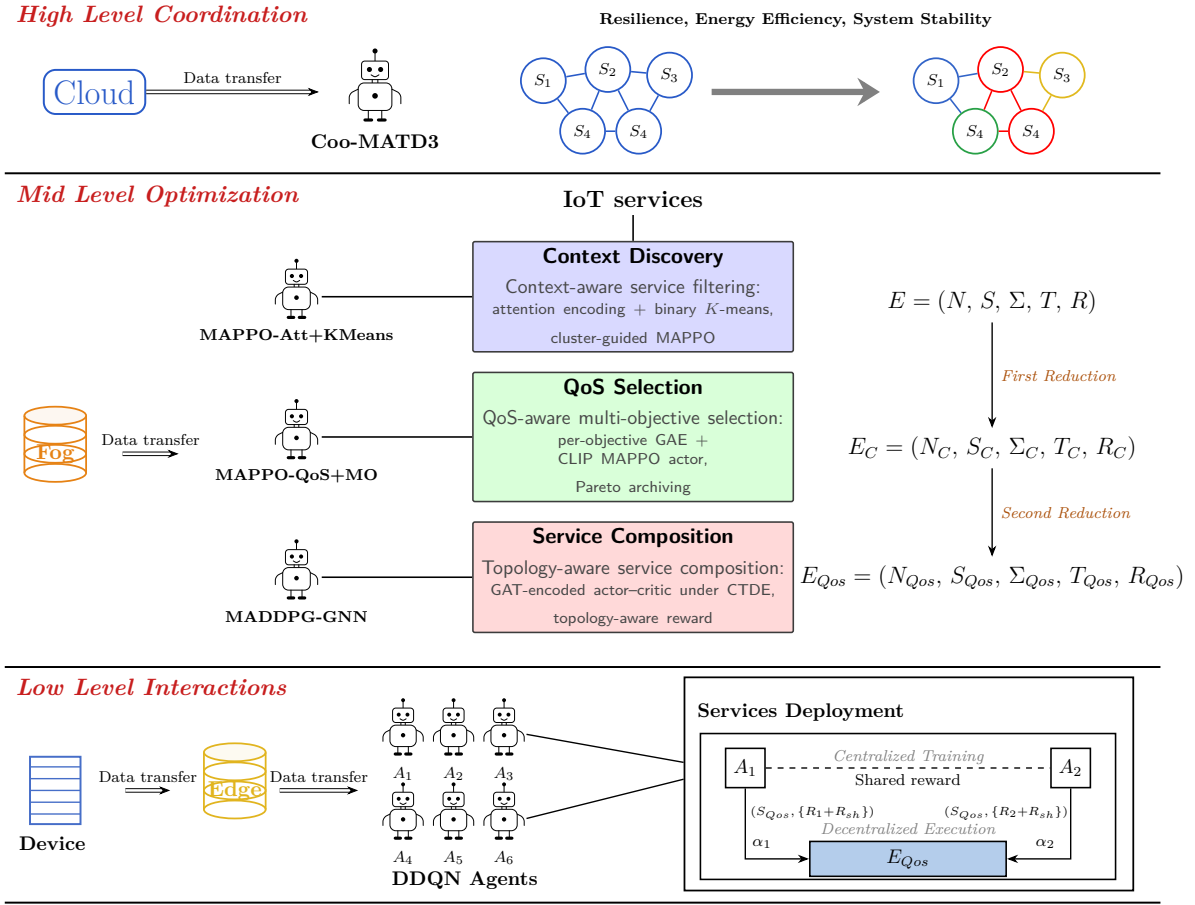


Figure 4.1: Hierarchical Graph Approach Overview.

- A Pareto-archived multi-objective MAPPO for QoS-Aware Selection.
- A GAT integrated with MADDPG for graph-based Service Composition.

Ultimately, the visual representation underscores that these two frameworks are profoundly complementary. By uniting Hi-MAD’s hierarchical, scope-based control of physical IoT components with GAMMA’s dynamic, graph-driven orchestration of logical IoT services, the combined architecture bridges the gap between hardware constraints and software agility. Together, they establish a comprehensive, highly adaptive, and resilient management architecture capable of sustaining performance in large-scale, unpredictable IoT environments.

4.3 High-Level Coordination

In **Hi-MAD**, the high-level agents govern global coordination by supervising the distributed activities of mid-level agents to ensure consistent fulfillment of system-wide objectives. Dynamically, the high-level agent allocates sub-goals, monitors aggregated performance metrics, and refines global strategies in response to real-time environmental feedback. To achieve robust and scalable decision-making under dynamic IoT conditions, **Hi-MAD** adopts the Coo-MATD3 algorithm, an enhanced extension of the Multi-Agent Twin Delayed Deep Deterministic Policy Gradient (MATD3) method specifically

adapted for hierarchical coordination. The choice of MATD3 at this level is justifying by its ability to handle continuous control spaces while mitigating the overestimation bias common in traditional actor–critic algorithms such as MADDPG. By employing dual critic networks and delayed policy updates, MATD3 stabilizes learning across multiple cooperative agents operating under partial observability. Building on these strengths, Coo-MATD3 introduces explicit inter-agent coordination terms within the reward and policy update processes, promoting synchronized policy refinement and improved consensus among agents across hierarchical levels. This enhancement accelerates convergence, strengthens system-level resilience, and enables graceful recovery from agent or network disturbances. By harmonizing decentralized autonomy with global consistency, Coo-MATD3 allows **Hi-MAD**’s high-level agent to achieve adaptive, scalable, and resilient coordination throughout complex, heterogeneous IoT environments.

4.3.1 Global Objectives Optimization

In Coo-MATD3, the high-level agent optimizes a reward that covers scalability, stability, resilience, and energy efficiency. Each goal uses simple measures: changes in performance for scalability, steady actions for stability, quick recovery from faults for resilience and energy use per task for efficiency. These are combined in a policy update with agent coordination, enabling synchronized, reliable, and energy-saving decisions that go beyond standard MATD3.

- *Scalability* enables efficient coordination among an increasing number of mid-level agents, allowing the system to expand without loss of performance. This is achieved through a coordination gradient term that captures the influence of other agents’ actions α_i on agent i ’s policy, aligning policy updates and maintaining stable, synchronized learning without centralized control, defined as:

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}[\nabla_{\alpha_i} Q_i(s, \alpha_i, \alpha_{-i}) \nabla_{\theta_i} \mu_i(s_i) + \lambda \nabla_{\theta_i} \Omega(\alpha_i, \alpha_{-i})], \quad (4.1)$$

where:

- $\nabla_{\theta_i} J(\theta_i)$: Gradient of the objective function of agent i , guiding the optimization of its parameterized policy.
- $\nabla_{\alpha_i} Q_i(s, \alpha_i, \alpha_{-i})$: Sensitivity of the agent’s action-value function with respect to its own action, conditioned on the global state s and the joint actions of the other agents α_{-i} ; captures inter-agent strategic dependencies.
- $\nabla_{\theta_i} \mu_i(s_i)$: Derivative of agent i ’s deterministic policy with respect to its parameters θ_i , indicating how policy updates influence the selected action.
- λ : Regularization coefficient controlling the influence of the cooperative or stabilizing term.
- $\Omega(\alpha_i, \alpha_{-i}) = -\frac{1}{2} \sum_{j \neq i} \psi_{ij} \|a_i + \rho_{ij} a_j\|^2$: Coordination-oriented regularization term (e.g., consensus, alignment, conflict penalization) modeling structured interactions among agents.
- $\lambda \nabla_{\theta_i} \Omega(\alpha_i, \alpha_{-i}) = -\sum_{j \neq i} \psi_{ij} (a_i + \rho_{ij} a_j)$: Contribution of the coordination mechanism to the overall policy gradient, enhancing stability, cooperation, and robustness.

- $\mathbb{E}[\cdot]$: Expectation over the distribution of visited states and actions, ensuring statistically representative updates.

Unlike the original MATD3—which treats agents’ updates independently, Co-MATD3 maintains scalability through implicit policy coupling, allowing the system to expand while preserving coordination efficiency.

- *System Stability* is ensured by the Co-MATD3 Algorithm 1, where dual-critic and delayed updates reduce overestimation and oscillations, and the coordination gradient aligns multi-agent policies to prevent divergence. This design stabilizes continuous control, lowers performance variance, and ensures smooth and reliable long-term operation, with each high-level agent maintaining two independent critic networks, $Q_{i,1}(s, a_1, \dots, a_N)$ and $Q_{i,2}(s, a_1, \dots, a_N)$, parameterized respectively by $\theta_{i,1}$ and $\theta_{i,2}$. During training, the target Q -value is computed as the minimum of the two target critics in order to mitigate overestimation bias:

$$y_i(t) = R_i(t) + \gamma \min_{k=1,2} Q_{i,k}^{\text{target}}(s'(t), a'_1(t), \dots, a'_N(t)), \quad (4.2)$$

where:

- $R_i(t)$ is the cooperative reward at time t ,
- γ is the discount factor,
- $Q_{i,k}^{\text{target}}$ are the slowly updated target critics $Q_{i,1}$ and $Q_{i,2}$.
- the next-step actions are computed as $\alpha'_j(t) = \mu_j^{\text{target}}(s'_j(t)) + \varepsilon$, $\varepsilon \sim \mathcal{N}(0, \sigma^2)$.

Using the minimum of the two critics ensures conservative value estimation, reducing the risk of policy oscillations and improving stability during cooperative learning across agents.

- *Resilience* is maintained by continuously monitoring subsystems to detect and correct faults, while Co-MATD3 reinforces this through a variance-triggered consensus rule that ensures reliable, fault-tolerant operation:

$$a_{\text{output}} = \begin{cases} \text{clip}\left(\frac{1}{n} \sum_i A_i, -1, 1\right), & \text{if } \max_j (\text{Var}_j(A)) > 0.1, \\ A, & \text{otherwise.} \end{cases} \quad (4.3)$$

When the action variance exceeds the threshold indicating instability or potential agent faults, the system replaces individual actions with their mean, thereby minimizing $\mathbb{E}[\|A_i - a_{\text{output}}\|^2]$. This follows from the fact that the mean action $\bar{A}$ yields the minimum-variance aggregate response across agents and the clipping operator keeps actions within safe limits, preventing instability.

- *Energy Efficiency* in the high-level comes from the stable and resilient design of Co-MATD3. The dual-critic and delayed updates prevent unstable actions and reduce extra communication, while the consensus mechanism detects inefficient nodes and balances tasks, leading to smoother control and lower energy use:

$$\mathbb{E}[E_{\text{total}}(t)] \propto \underbrace{\sum_i \mathbb{E}[\|\Delta a_i(t)\|^2]}_{\text{Stability term}} + \underbrace{\lambda \text{Var}(A(t))}_{\text{Resilience term}}. \quad (4.4)$$

Where stable actions (low $\sum_i \mathbb{E}[\|\Delta a_i(t)\|^2]$) reduce unnecessary device switching, and resilient coordination (low $\lambda \text{Var}(A(t))$) prevents energy-wasting failures or overcorrections.

4.3.2 Co-MATD3 Learning for Coordination

Building upon the previously defined global objectives of scalability, stability, resilience, and energy efficiency, the high-level coordination in **Hi-MAD** is achieved through Co-MATD3, formalizing these objectives within a unified reward function R_G . This reward-driven design enables smooth, synchronized, and energy-efficient decision-making, extending MATD3 into a coordination-aware, resilience-enhanced, and energy-optimized learning framework for large-scale IoT systems, expressed as:

$$R_G(t) = R_S(t) + R_R(t) + R_E(t), \text{ where:} \quad (4.5)$$

- $R_S(t) = -\eta_S \sum_i \|\Delta a_i(t)\|^2$, $\Delta a_i(t) = a_i(t) - a_i(t-1)$, the reward related to the stability of the system in 4.2.
- $R_R(t) = -\eta_R \text{Var}(A(t)) = -\eta_R \frac{1}{n} \sum_i \|A_i(t) - \bar{A}(t)\|^2$, the reward related to the resilience computed by the variance in 4.3
- $R_E(t) = -\eta_E (\sum_i \|\Delta a_i(t)\|^2 + \lambda \text{Var}(A(t)))$, the reward related to the energy efficiency in 4.4.

Within the hierarchical architecture of **Hi-MAD**, Co-MATD3 functions as the essential coordination backbone for the mid-level agents, and this is particularly crucial when managing the complexities of intelligent and energy-efficient smart building control. As detailed in Algorithm 1 (lines 1-3), the process begins by initializing the core components, which include the environment itself, the actor and critic networks, their corresponding target networks, and the replay buffer.

As the algorithm progresses through each episode and individual time step (lines 4-6), the agents observe the current state of the environment and proceed with action selection, during which the actor networks generate noisy actions to encourage exploration (lines 7-8). To ensure the system remains resilient to extreme or destabilizing behaviors (lines 9-11), the algorithm actively monitors action variance; if this variance exceeds a predetermined safety threshold, the erratic actions are immediately replaced by their clipped average, thereby preserving coordination across the multi-agent system. Once these regulated actions are executed within the environment, the agents obtain the subsequent states and calculate composite rewards based on stability, resilience, and energy efficiency, all of which are subsequently stored in the replay buffer for future learning (lines 12-14). From this buffer, a mini-batch of experiences is sampled (lines 15-16) to compute target values, utilizing policy smoothing and Double-Q learning techniques to stabilize the training process (lines 17-19). Following this, the twin critic networks are updated by applying gradient descent to minimize their loss (lines 20-21). At specified intervals (lines 22-27), the actor networks are updated using a specialized, coordination-aware gradient that intelligently balances deterministic performance with cooperative multi-agent objectives, which is then immediately followed by soft updates to the target networks to ensure gradual learning (lines 28-29). Finally, the current state is refreshed for the next iteration (line 30), allowing the Co-MATD3 framework to consistently deliver stable, highly coordinated, and energy-efficient multi-agent learning throughout the entire operational lifecycle.

Algorithm 1 Co-MATD3 for Coordination

```

1: Input: Environment state  $S$  ▷ Global observation of the environment
2: Output: Coordinated mid-level continuous actions  $\Sigma$ 
3: Initialize: Actor  $\pi_\phi$ , twin critics  $Q_{\theta_1}, Q_{\theta_2}$ , target networks  $Q_{\theta'_1}, Q_{\theta'_2}$ ,  $\pi_{\phi'}$ , replay buffer  $\mathcal{R}$  ▷ TD3-style actor-critic architecture
4: for episode = 1 to MaxEpisodes do
5:   Observe initial state  $S$  ▷ Reset environment at episode start
6:   for time step  $t = 1$  to  $T$  do
7:     Action selection
8:      $A = \{\alpha_i = \pi_\phi(s_i) + \text{noise} \mid \forall i\}$  ▷ Decentralized actions with exploration noise
9:     Resilience check: variance-triggered consensus
10:    Compute  $\text{Var}_j(A)$  for each action dimension  $j$  ▷ Measure disagreement among mid-level agents Equation 4.3
11:     $A_{\text{output}} = \begin{cases} \text{clip}(\frac{1}{n} \sum_{i=1}^n A_i, -1, 1), & \text{if } \max_j \text{Var}_j(A) > 0.1, \\ A, & \text{otherwise.} \end{cases}$  ▷ Apply consensus action when coordination degrades
12:    Environment transition
13:     $(S', R_G) = \text{ExecuteActions}(A_{\text{output}})$  ▷ Execute actions and observe next state
14:    Global reward computation
15:     $R_G = R_S + R_R + R_E$  ▷ Stability, resilience, and energy-aware reward Equation 4.5
16:    Store  $(S, A_{\text{output}}, R_G, S')$  in buffer  $\mathcal{R}$  ▷ Off-policy experience replay
17:    Sample mini-batch
18:     $(S_b, A_b, R_b, S'_b) = \text{Sample}(\mathcal{R})$ 
19:    Target policy smoothing and Double-Q target
20:     $\alpha'_b = \pi_{\phi'}(S'_b) + \text{clipped\_noise}$  ▷ TD3 noise smoothing to reduce overestimation
21:     $y_b = R_b + \gamma \min(Q_{\theta'_1}(S'_b, \alpha'_b), Q_{\theta'_2}(S'_b, \alpha'_b))$  ▷ Double critic target for bias reduction Equation 4.2
22:    Critic update
23:     $L_Q = (Q_{\theta_1}(S_b, A_b) - y_b)^2 + (Q_{\theta_2}(S_b, A_b) - y_b)^2$ 
24:    Update  $\theta_1, \theta_2$  via gradient descent ▷ Minimize Bellman error
25:    if  $t \bmod \text{policy\_delay} = 0$  then
26:      Actor update with coordination term
27:       $\nabla_{\theta_i} J(\theta_i) = \mathbb{E}[\nabla_{\alpha_i} Q_i(s, \alpha_i, \alpha_{-i}) \nabla_{\theta_i} \mu_i(s_i) + \lambda \nabla_{\theta_i} \Omega(\alpha_i, \alpha_{-i})]$  ▷ Coordination-aware policy gradient Equation 4.1
28:       $\phi \leftarrow \phi + \eta \nabla_{\phi} J(\phi)$  ▷ Delayed actor update improves stability
29:      Soft update of target networks
30:       $\{\theta'_1, \theta'_2, \phi'\} \leftarrow \tau \{\theta_1, \theta_2, \phi\} + (1 - \tau) \{\theta'_1, \theta'_2, \phi'\}$  ▷ Polyak averaging
31:    end if
32:     $S \leftarrow S'$  ▷ Advance environment
33:  end for
34: end for

```

4.4 Middle-Level Optimization

The Mid-Level layer orchestrates the composition and delivery of IoT services through two core mechanisms: *context-based discovery* and *QoS-based selection*. These mechanisms enable dynamic adaptation to environmental conditions and contribute to minimizing energy consumption in real time. This layer is built upon a CTDE paradigm, which ensures both scalability and operational autonomy. Agents are trained collectively to learn optimal policies but function independently during execution. To realize this architecture, we propose three specialized multi-agent learning models. *MAPPO-Att+KMeans* for Context Discovery using Multi-Agent Proximal Policy Optimization with Attention and K-Means clustering, that enhances standard MAPPO with a learned adaptive context, attention mechanism and clustering, enabling robust and context-sensitive IoT service discovery in dynamic environments. These additions allow to discover agents based on their contextual relevance to the environment. *MAPPO-QoS+MO* for QoS Selection using Multi-Agent Proximal Policy Optimization with Multi-Objective Optimization incorporates explicit QoS awareness into the state, action space, reward design, and convergence criteria. By enforcing QoS constraints and optimizing service quality, it transforms MAPPO into an efficient, domain-specific solution for QoS-driven IoT service selection in order to filter the active agents based on service-specific QoS metrics. Finally, *MADDPG-GNN* for Composition with Multi-Agent Deep Deterministic Policy Gradient and Graph Networks extends standard MADDPG by embedding explicit cooperative semantics integrating graph-based relational learning, service-level state and reward modeling, and

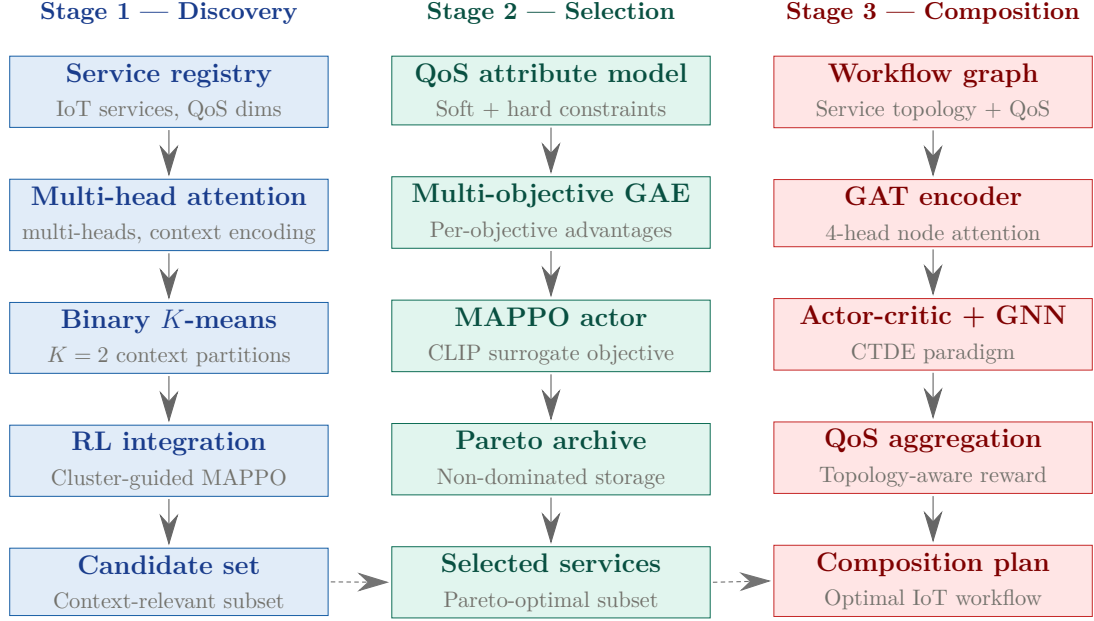


Figure 4.2: The **GAMMA** Three-Stages Architecture

composition-oriented coordination mechanisms. It facilitates decentralized coordination in continuous action spaces, optimizing both collaboration and trade-offs between energy efficiency and QoS objectives.

Figure 4.2 illustrates the three-stage pipeline architecture of the **GAMMA** framework. In Stage 1 (Discovery), a *Multi-Head Attention* mechanism encodes contextual relevance over a service registry, whose embeddings are partitioned by *Binary K-Means* ($K = 2$) clustering and fed into a cluster-guided MAPPO policy to produce a context-relevant *Candidate Set*. This set is forwarded to Stage 2 (Selection), where a *Multi-Objective GAE* computes per-objective advantage estimates under soft and hard QoS constraints, guiding a MAPPO actor optimized via a clipped surrogate objective; non-dominated solutions are retained in a *Pareto Archive*, yielding a Pareto-optimal *Selected Services* subset. Finally, Stage 3 (Composition) constructs a *Workflow Graph* over the selected services, applies a 4-head *GAT Encoder* to aggregate topology-aware node embeddings, and optimizes an *Actor-Critic + GNN* policy under the CTDE paradigm with a topology-aware QoS aggregation reward, producing a *Composition Plan* representing the optimal deployable IoT workflow.

4.4.1 Context-aware Service Discovery

Definition 1 formalizes this hierarchical optimization process as a transformation of a full Multi-Agent Markov Decision Process (MDP), denoted by E , into a reduced and task-specific MDP, E_Q as represented in Figure 4.3, through a two-stage agent selection mechanism. Initially, *Context Discovery* via CD-MAPPO identifies a relevant subset of agents suited to the environmental context.

This is followed by *QoS-Based Selection* via QS-MAPPO, which further narrows down the agent set based on real-time QoS criteria. The final subset operates under the Co-MADDPG framework, enabling efficient and adaptive decentralized execution.

Definition 1 (Full MPD). *The original environment is modeled as a MAMDP $E = \langle N, S, \Sigma, T, R \rangle$ where:*

- $N = \{a_1, a_2, \dots, a_N\}$: Set of all agents representing all the IoT services,
- S : State space of the environment,
- $\Sigma = \{\alpha_1, \alpha_2, \dots, \alpha_N\}$: Joint action space,
- $T(s' | s, \alpha)$: State transition function,
- $R = \{R_1, R_2, \dots, R_N\}$: Individual reward functions.

Each agent a_i aims to learn a policy $\pi_i(\alpha_i | s)$ that maximizes its expected cumulative reward $J_i = \mathbb{E} [\sum_{t=0}^{\infty} \gamma^t R_t^i]$ where $\gamma \in [0, 1)$ is the discount factor that balances immediate and future rewards. To quantify the scale of this MDP, we compute the following structural components:

1. Total number of agents n , which directly impacts the coordination overhead and policy learning complexity.
2. Aggregate action space size A_{total} , as the sum of individual agent action spaces.
3. Joint state dimensionality S_{joint} , derived from concatenating individual observation vectors.
4. Approximate transition space size $\mathcal{T}$, estimated using $|\mathcal{T}| \sim |\mathcal{S}_{\text{joint}}| \cdot |\mathcal{A}_{\text{total}}| \cdot n$ which captures the exponential growth in environment dynamics relative to agent count.

These metrics serve as the baseline for evaluating reductions achieved in subsequent filtering phases, particularly under a centralized training scheme where agents observe a joint state, act jointly, and share a global reward signal. The reduction process occurs in two stages: *Context Discovery*, where CD-MAPPO identifies agents that are contextually relevant to the environment, and *QoS-Based Selection*, which QS-MAPPO filters agents based on predefined QoS metrics. This ensures that only agents that meet QoS constraints are retained in the final MDP, improving both efficiency and scalability.

In large-scale IoT environments, raw QoS attributes alone are insufficient to capture the latent contextual relationships among heterogeneous services. We therefore introduce a *multi-head self-attention encoder* that transforms the original service feature space into a context-enriched embedding space, enabling downstream agents to reason over inter-service dependencies.

Multi-Head Self-Attention Mechanism

Rather than relying on a single attention function, we employ H parallel attention heads to capture diverse contextual patterns simultaneously. Each head $h \in \{1, \dots, H\}$ maintains its own learnable projection matrices: $W_Q^h \in \mathbb{R}^{d \times d_k}$, $W_K^h \in \mathbb{R}^{d \times d_k}$, $W_V^h \in \mathbb{R}^{d \times d_v}$,

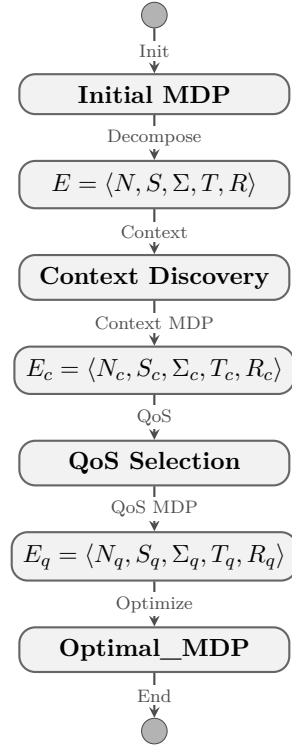


Figure 4.3: MDP Reduction Steps.

with $d_k = d_v = d/H$ to maintain the total parameter count. Each head computes:

$$\text{head}_h = \text{Attention}(\mathbf{X}W_Q^h, \mathbf{X}W_K^h, \mathbf{X}W_V^h) \in \mathbb{R}^{N \times d_v}. \quad (4.6)$$

The H head outputs are concatenated and linearly projected through an output matrix $W_O \in \mathbb{R}^{d \times d}$:

$$\text{MHA}(\mathbf{X}) = \text{Concat}(\text{head}_1, \dots, \text{head}_H)W_O \in \mathbb{R}^{N \times d}.$$

To facilitate gradient flow and preserve the original service features, we incorporate a residual connection followed by layer normalisation: $\mathbf{Z} = \text{LayerNorm}(\text{MHA}(\mathbf{X}) + \mathbf{X}) \in \mathbb{R}^{N \times d}$, where $\mathbf{z}_i \in \mathbb{R}^d$ denotes the *context-aware embedding* of service i . Each $\mathbf{z}_i$ now encodes not only the intrinsic QoS profile of service i but also its contextual relationships with all other services in the registry. The layer normalization operates across the feature dimension for each service independently: $\text{LayerNorm}(\mathbf{z}_i) = \frac{\mathbf{z}_i - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}} \odot \boldsymbol{\gamma} + \boldsymbol{\beta}$, with per-service mean $\mu_i = \frac{1}{d} \sum_{k=1}^d z_{ik}$, variance $\sigma_i^2 = \frac{1}{d} \sum_{k=1}^d (z_{ik} - \mu_i)^2$, and learnable affine parameters $\boldsymbol{\gamma}, \boldsymbol{\beta} \in \mathbb{R}^d$.

Context-Based Service Partitioning via Binary K -Means Clustering

After obtaining the context-aware embeddings $\{\mathbf{z}_i\}_{i=1}^N$, the discovery stage must partition the service registry into two disjoint subsets: *context-verified* services $\mathcal{S}^+$ that satisfy the contextual requirements of the current IoT task, and *non-verified* services $\mathcal{S}^-$ that do not. We formulate this as a binary K -Means clustering problem with $K = 2$. We first reduce each context-aware embedding to a scalar *context score* that measures the degree of contextual fitness. A learnable projection followed by a sigmoid activation is applied: $c_i = \sigma(\mathbf{w}_c^\top \mathbf{z}_i + b_c) \in (0, 1)$, $\forall i \in \{1, \dots, N\}$, where $\mathbf{w}_c \in \mathbb{R}^d$ and $b_c \in \mathbb{R}$ are learnable parameters, and $\sigma(\cdot)$ is the sigmoid function. The context score c_i encapsulates the attention-weighted contextual fitness of service i into a single discriminative value.

Given the set of scalar context scores $\mathcal{C} = \{c_1, c_2, \dots, c_N\}$, we apply K -Means with $K = 2$ to obtain two clusters. The objective is to minimise the within-cluster sum of squared errors (SSE):

$$\min_{\mu^+, \mu^-} \underbrace{\sum_{i \in \mathcal{S}^+} (c_i - \mu^+)^2}_{\text{context-verified cluster}} + \underbrace{\sum_{i \in \mathcal{S}^-} (c_i - \mu^-)^2}_{\text{non-verified cluster}}, \quad (4.7)$$

where μ^+ and μ^- are the centroids of $\mathcal{S}^+$ and $\mathcal{S}^-$, respectively.

Integration with the MAPPO Model

Only services in $\mathcal{S}^+$ are forwarded to the downstream MAPPO agents for reinforcement learning-based discovery. This pre-filtering yields two key advantages:

1. **Reduced action space.** The RL agents operate over $|\mathcal{S}^+| \ll N$ services, accelerating convergence and reducing sample complexity.
2. **Quality assurance.** Non-verified services in $\mathcal{S}^-$ are excluded before any costly QoS evaluation, preventing low-quality candidates from corrupting the reward signal.

At each training episode, the attention parameters $\{W_Q^h, W_K^h, W_V^h\}_{h=1}^H$, the output projection W_O , and the context projection $(\mathbf{w}_c, b_c)$ are jointly updated by minimizing a binary cross-entropy loss over the clustering assignments:

$$\mathcal{L}_{\text{cls}} = -\frac{1}{N} \sum_{i=1}^N \left[y_i \log c_i + (1 - y_i) \log(1 - c_i) \right], \quad (4.8)$$

where $y_i = \mathbb{I}[i \in \mathcal{S}^+]$ is the binary cluster label derived from the K -Means assignment. This end-to-end training ensures that the attention encoder progressively learns to produce embeddings that are maximally discriminative for context-based service partitioning.

The context relevance function is defined as:

$$C(a_i, s) = \exp \left(- \sum_{j \in O_i} w_j \cdot |s_j - s_{j, \text{opt}, i}| \right) \quad (4.9)$$

The function $C(a_i, s)$ decays exponentially with the aggregated weighted deviation between the observed state and the agent's optimal state. When $s_j \approx s_{j, \text{opt}, i}$, $C(a_i, s) \rightarrow 1$, indicating low intervention need. Else, if the deviation is large, $C(a_i, s) \rightarrow 0$, signaling a high need for agent activation. For activation purposes, we define the *relevance score* (or activation confidence) as:

$$R(a_i, s) = 1 - C(a_i, s)$$

This value $R(a_i, s) \in [0, 1]$ reflects priority for effective intervention. A higher value corresponds to greater deviation and a higher likelihood that agent a_i should be activated to take corrective action. MAPPO in service discovery aims to balance task objectives, contextual relevance, and multi-agent coordination. Thus, we design an affective reward function guiding the learning process to ensure adaptive decision-making defined as:

$$R_{\text{total}} = R_{\text{task}} + \alpha \cdot C(a, s) + \beta \cdot \text{Load} + \gamma \cdot \text{Energy} + \delta \cdot \text{Rate}$$

where:

- $R_{\text{task}}(s_t, a_t) = \lambda_1 \cdot \mathbb{I}_{\text{success}} + \lambda_2 \cdot \frac{d_{\min}(s_0) - d_{\min}(s_t)}{d_{\min}(s_0)} - \lambda_3 \cdot t$ models success/failure reward of service discovery with progressive shaping, $\mathbb{I}_{\text{success}}$ be a binary indicator (1 if the service is found), $d_{\min}(s_t)$ the minimum distance to the target service at state s_t , and t the number of time steps elapsed.
- $C(a, s)$ is the context relevance calculated with Eq. 4.9.
- $L = 1 - \frac{L_a}{\text{Capacity}}$ models the load balancing to avoid overloading servers/agents with the current resource utilization L_a and the maximum available resources Capacity .
- $\text{Energy} = 1 - \frac{\text{Energy}_{\text{used}}}{\text{Energy}_{\text{max}}} \in [0, 1]$ discourages energy-wasting behaviors: agents are rewarded proportionally to the fraction of energy saved relative to the maximum budget, incentivizing minimal consumption while maintaining service quality.
- $\text{Rate}(s_t, a_t) = \mu_1 \cdot -\text{Overlap}(a_t) + \mu_2 \cdot \text{Coverage}(a_t) + \mu_3 \cdot \sum_{i,j} \text{sim}(a_i, a_j) \cdot \mathbb{I}[\text{services } a_i, a_j \text{ are complementary}]$ with $\text{Overlap}(a_t) = -\sum_{i \neq j} \mathbb{I}[a_i = a_j]$ which penalizes redundant actions across agents by summing over all pairs $(i \neq j)$ where the same action is taken, using the indicator function $\mathbb{I}[\cdot]$.

$$\text{Coverage}(a_t) = \frac{\# \text{unique services discovered}}{\text{Total agents}}$$

Algorithm 2 MAPPO-Att+KMeans for Context Discovery

Require: Agents N , QoS attributes J , clusters K_{cl} , initial thresholds $\{s_{j,opt,i}^{(0)}\}$, weights $\{w_j\}$, horizon T , PPO epochs M ,
 $\gamma, \lambda, \epsilon, \alpha_{lr}, \eta, \tau, \eta_\mu, \lambda_{decay}, \Delta_{max}^{(0)}, \alpha_c, \beta_L, \gamma_E, \delta_B$
Ensure: Policies $\{\pi_{\theta_i}\}$, critic V_ϕ , thresholds $\{s_{j,opt,i}\}$

- 1: Init $\{\theta_i\}, \phi, \{W_Q, W_K, W_V\}$; centroids $\{\mu_k\}_{k=1}^{K_{cl}}$ via k -means++; $ep \leftarrow 0$
- 2: **repeat**
- 3: $ep \leftarrow ep + 1$; $\mathcal{D} \leftarrow \emptyset$ $\triangleright$ On-policy: clear buffer
- 4: **for** $t = 0, \dots, T-1$ **do**
- 5: $\mathbf{h}_{i,t} \leftarrow \sum_{j=1}^N \text{softmax}(\frac{(W_Q \mathbf{o}_{i,t})^\top W_K \mathbf{o}_{j,t}}{\sqrt{d_k}}) W_V \mathbf{o}_{j,t}$ $\triangleright$ Multi-agent attention
- 6: $p_{ik} \leftarrow \text{softmax}(-\|\mathbf{h}_{i,t} - \mu_k\|^2 / \tau)$; $C_i \leftarrow \sum_k p_{ik} \mu_k^\top \mathbf{h}_{i,t}$ $\triangleright$ Soft KMeans context
- 7: Sample $a_{i,t} \sim \pi_{\theta_i}(\cdot | \mathbf{o}_{i,t}, \mathbf{h}_{i,t})$ for each agent i
- 8: $R_t \leftarrow R_{task,t} + \alpha_c C_i - \beta_L L_t - \gamma_E \mathcal{E}_t - \delta_B B_t$
- 9: Store $(s_t, \mathbf{o}_t, \mathbf{a}_t, \{\log \pi_{\theta_i}^{old}\}, R_t, s_{t+1}, V_\phi(s_t))$ in $\mathcal{D}$
- 10: **end for**
- 11: **GAE:** $\hat{A}_T \leftarrow 0$; **for** $t = T-1 \dots 0$: $\delta_t \leftarrow R_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$, $\hat{A}_t \leftarrow \delta_t + \gamma \lambda \hat{A}_{t+1}$, $\hat{V}_t \leftarrow \hat{A}_t + V_\phi(s_t)$
- 12: **for** each agent i , each QoS j **do** $\triangleright$ Context threshold update
- 13: $\bar{C}_i \leftarrow \frac{1}{T} \sum_t C_i$; $s_{j,opt,i} \leftarrow \text{clip}(s_{j,opt,i} + \eta w_j \text{sign}(s_j - s_{j,opt,i}) \bar{C}_i, s_{j,opt,i}^{(0)} \pm \Delta_{max})$
- 14: **end for**
- 15: **for** $k = 1, \dots, K_{cl}$ **do** $\triangleright$ EMA centroid update
- 16: $\mu_k \leftarrow (1 - \eta_\mu) \mu_k + \eta_\mu \text{mean}_{(i,t): \arg \max_{k'} p_{ik'} = k} \mathbf{h}_{i,t}$
- 17: **end for**
- 18: **for** $m = 1, \dots, M$ **do** $\triangleright$ PPO-clip update
- 19: $r_i \leftarrow \pi_{\theta_i} / \pi_{\theta_i}^{old}$; $\theta_i \leftarrow \theta_i - \alpha_{lr} \nabla_{\theta_i} [-\frac{1}{T} \sum_t \min(r_i \hat{A}_t, \text{clip}(r_i, 1 \pm \epsilon) \hat{A}_t)] \quad \forall i$
- 20: $\phi \leftarrow \phi - \alpha_{lr} \nabla_\phi [\frac{1}{T} \sum_t (V_\phi(s_t) - \hat{V}_t)^2]$
- 21: **end for**
- 22: $\eta \leftarrow \lambda_{decay} \eta$; $\Delta_{max} \leftarrow \Delta_{max}^{(0)} (1 - ep / K_{max})$
- 23: **until** $ep \geq K_{max}$ **or** $\bar{\mathcal{D}}_{KL}(\pi^{(ep)} \| \pi^{(ep-1)}) < \epsilon_\pi$ **or** $|\Delta \bar{R}| < \epsilon_R$ **or** $\max_{i,j} |\Delta s_{j,opt,i}| < \epsilon_s$

quantifies how agents are exploring or covering distinct services.

$$(\text{Synergy}(a_t) = \sum_{i,j} \text{sim}(a_i, a_j) \cdot \mathbb{I}[\text{services } a_i, a_j \text{ are complementary}])$$

quantifies how well two services work together.

Algorithm 2 details the MAPPO-Context Discovery procedure with Attention and K-Means clustering. The framework is initialized with agent policies $\{\pi_{\theta_i}\}$, critic V_ϕ , attention projection matrices $\{W_Q, W_K, W_V\}$, and K -means++ centroids $\{\mu_k\}$ (Line 2). At each time step t of the on-policy rollout (Lines 4-10), a multi-head attention mechanism aggregates inter-agent observations into contextual embeddings $\mathbf{h}_{i,t}$ (Line 5), which are then soft-assigned to cluster centroids via temperature-scaled softmax to compute the context relevance score C_i (Line 6). Each agent samples an action conditioned on both its local observation and the attention embedding (Line 7), and receives a composite reward combining task performance, context relevance, load, energy, and bandwidth penalties (Line 8). After rollout completion, Generalized Advantage Estimation (GAE) computes advantage targets $\hat{A}_t$ and value targets $\hat{V}_t$ for the full trajectory (Line 11). Context-aware QoS thresholds $s_{j,opt,i}$ are then updated via a weighted gradient rule clipped to a decaying perturbation budget Δ_{max} (Lines 12-14), while cluster centroids are refined through Exponential Moving Average (EMA) over the assigned embeddings (Lines 15-17). Finally, M epochs of PPO-clip policy updates are applied to all agent actors and the shared critic (Lines 18-21), followed by learning rate decay and Δ_{max} annealing (Line 22). Training terminates when the maximum episode count K_{max} is reached or when KL divergence, reward improvement, and threshold shift all fall below their respective convergence tolerances ϵ_π , ϵ_R , and ϵ_s (Line 23).

CD-MAPPO is trained to optimize the context selection policy $\pi_C(N_C | S) = \prod_{a_i \in N_C} \pi_i(\alpha_i | S)$, that determines the subset of agents N_C , resulting a reduced multi-agent MDP E_C defined in Definition 2.

Definition 2 (Context-Based MDP). A context-based MDP is $E_C = \langle N_C, S_C, \Sigma_C, T_C, R_C \rangle$ where:

- $N_C = \{a_i \in N \mid C(a_i) \geq \theta_C\} \subseteq N$ is the subset of selected agents representing services that satisfy a predefined context threshold θ_C .
- S_C represent the context state space where the mid-level agent maintains a learned context vector capturing contextual preferences: $\mathbf{c}_i^t = (c_{1,i}^t, c_{2,i}^t, \dots, c_{J,i}^t)$.
- Σ_C represent the action spaces correspond to the services selected by agent a_i at time t .
- T_C defines the transition function specific to the selected context defined by Equation 4.9.
- R_C is the adjusted reward function, reflecting context-driven decision-making and scales with context relevance, defined as: $R_C = R_{task} + \beta C(a_i, s)$, values near 1 indicate strong matches, around 0.5 partial relevance, and near 0 poor or mismatched contexts, guiding the agent to optimal service choices.

4.4.2 QoS-based Service Selection

Given the set of candidate services V_C discovered in the previous pipeline stage, the service selection problem consists of identifying the subset that jointly optimizes multiple QoS objectives while satisfying minimum quality guarantees. We formulate this as a multi-agent multi-objective optimization problem and solve it using MAPPO-QoS+MO, a multi-objective extension of Multi-Agent Proximal Policy Optimization with adaptive preference learning to produce a more reduced QoS-based MDP given in Definition 3.

Definition 3 (QoS-Based MDP). A QoS-based MDP is $E_Q = \langle N_Q, S_Q, \Sigma_Q, T_Q, R_Q \rangle$ where:

- $N_{Qos} \subseteq N_C$ is the final subset of agents representing services that satisfy QoS constraints. Agents meeting the QoS threshold θ_{Qos} are selected: $N_{Qos} = \{a_i \in N_C \mid Qos(a_i) \geq \theta_{Qos}\}$,
- S_{Qos} represents the state space satisfying the QoS,
- Σ_{Qos} represents the action spaces of the selected agents.
- T_{Qos} defines state transitions under QoS constraints computed by Equation 4.10.
- R_{Qos} is the reward tailored to QoS optimization, that yields a scalar, bounded, and interpretable reward in which higher values directly correspond to improved overall service quality. As a result, it facilitates efficient credit assignment and enhances learning stability within QS-MAPPO, computed by :

$$R_{Qos} = R_{task} + \sum_{j=1}^k w_j QoS_j(a_i).$$

QoS Attribute Model

Each candidate service $v_i \in V_C$ exposes a vector of m QoS attributes $\mathbf{q}_i = [q_i(1), q_i(2), \dots, q_i(m)]$, where each attribute j represents a measurable quality dimension (e.g., response time, availability, throughput, reliability, cost). We partition the QoS attribute index set into two disjoint subsets:

- **Hard-constraint attributes** J_{hard} : attributes for which services must meet a strict minimum threshold $\theta_{\min}(j)$.
- **Soft-optimization attributes** J_{soft} : attributes that are optimized in a best-effort manner through weighted utility aggregation.

For each soft attribute $j \in J_{\text{soft}}$, we define a normalization function $f_j : \mathbb{R} \rightarrow [0, 1]$ that maps raw QoS values to a common utility scale. For benefit attributes (higher is better), we use: $f_j(q_i(j)) = \frac{q_i(j) - q_j^{\min}}{q_j^{\max} - q_j^{\min}}$ and for cost attributes (lower is better): $f_j(q_i(j)) = \frac{q_j^{\max} - q_i(j)}{q_j^{\max} - q_j^{\min}}$ where $q_j^{\min}$ and $q_j^{\max}$ denote the minimum and maximum observed values of attribute j across V_C .

Before the reinforcement learning agents operate, we apply a two-filtering mechanism to prune the service space to a feasible subset $V_S \subseteq V_C$. The first stage enforces hard constraints via a conjunction over all critical attributes, and the second stage evaluates a weighted utility score over soft attributes. A service v_i is admitted into V_S if and only if it satisfies both conditions simultaneously:

$$V_S = \left\{ v_i \in V_C \mid \begin{array}{l} \bigwedge_{j \in J_{\text{hard}}} (q_i(j) \geq \theta_{\min}(j)) \\ \bigwedge \sum_{j \in J_{\text{soft}}} w_j \cdot f_j(q_i(j)) \geq \theta_{\text{utility}} \end{array} \right\} \quad (4.10)$$

where w_j denotes the importance weight assigned to soft attribute j ($\sum_{j \in J_{\text{soft}}} w_j = 1$), and $\theta_{\text{utility}} \in [0, 1]$ is the minimum acceptable aggregated utility score. This filtering step significantly reduces the action space for the downstream multi-agent selection, improving both convergence speed and solution quality.

Rather than collapsing all QoS dimensions into a single scalar a priori, we maintain separate reward signals for each objective throughout the learning process. At time step t , the per-objective reward for dimension j aggregates the normalized QoS contributions of all selected services:

$$r_j^t = \sum_{i \in V_S} f_j(q_i^t(j)), \quad \forall j \in \{1, \dots, m\} \quad (4.11)$$

These per-objective rewards are scalarized into a single training signal using the adaptive preference vector λ^k augmented with quadratic constraint violation penalties:

$$R^t = \underbrace{\sum_{j=1}^m \lambda_j^k r_j^t}_{\text{Preference-weighted utility}} - \underbrace{\sum_{j=1}^m \kappa_j \max(0, \theta_{\min}(j) - \bar{q}_j^t)^2}_{\text{Constraint violation penalty}} \quad (4.12)$$

where $\bar{q}_j^t$ is the mean QoS value of attribute j across selected services at time t , and $\kappa_j > 0$ are penalty coefficients that control the severity of threshold violations. The quadratic penalty ensures smooth gradients near the constraint boundary, guiding the policy toward feasible regions while allowing temporary violations during early exploration.

Multi-Objective Generalized Advantage Estimation

To enable stable policy optimization with reduced variance, we employ per-objective critics $\{V_{\psi_j}\}_{j=1}^m$, each estimating the expected return for a specific QoS dimension. For each objective j , we first compute temporal difference (TD) residuals: $\delta_{j,t} = r_j^t + \gamma V_{\psi_j}(s^{t+1}) - V_{\psi_j}(s^t)$, where $\gamma \in [0, 1)$ is the discount factor. The Generalized Advantage Estimation for objective j at time step t is then: $\hat{A}_{j,t} = \sum_{\ell=0}^{T-1-t} (\gamma \lambda_{\text{gae}})^\ell \delta_{j,t+\ell}$, where $\lambda_{\text{gae}} \in [0, 1]$ is the GAE smoothing parameter that controls the bias-variance trade-off. The multi-objective advantages are aggregated using the current preference weights: $\hat{A}_t = \sum_{j=1}^m \lambda_j^k \hat{A}_{j,t}$. This aggregated advantage drives the PPO clipped surrogate objective. Each per-objective critic is trained independently via regression on target returns: $\hat{R}_{j,t} = \hat{A}_{j,t} + V_{\psi_j}(s^t)$, $\mathcal{L}_{\text{critic}}^{(j)} = \mathbb{E}_t \left[\|V_{\psi_j}(s^t) - \hat{R}_{j,t}\|^2 \right]$.

Pareto Archive and Convergence

Throughout training, we maintain a Pareto archive $\mathcal{P}$ that stores non-dominated solution vectors. At the end of each episode e , the return vector:

$$\mathbf{G}^e = \left[\sum_{t=0}^{T-1} \gamma^t r_1^t, \dots, \sum_{t=0}^{T-1} \gamma^t r_m^t \right] \quad (4.13)$$

is added to $\mathcal{P}$ if it is not dominated by any existing solution. A solution $\mathbf{G}^a$ dominates $\mathbf{G}^b$ if $G_j^a \geq G_j^b$ for all j and $G_j^a > G_j^b$ for at least one j .

Training terminates when dual convergence criteria are met: the hypervolume indicator of the Pareto front stabilizes and all constraint violations fall below acceptable levels:

$$|HV(\mathcal{P})^k - HV(\mathcal{P})^{k-1}| < \epsilon \quad \wedge \quad \max_j v_j < \epsilon_v, \quad (4.14)$$

where $HV(\mathcal{P})$ measures the volume of the objective space dominated by $\mathcal{P}$ relative to a reference point, ϵ is the hypervolume convergence tolerance, and ϵ_v is the maximum acceptable constraint violation. The first condition ensures that the Pareto front has stabilized (no further trade-off improvements), while the second condition guarantees that all QoS thresholds are satisfied in expectation.

Algorithm 3 details the MAPPO-QoS+MO for multi-objective QoS-aware service selection. The framework is initialized with a shared actor π_ϕ , per-objective critics $\{V_{\psi_j}\}$, an empty Pareto archive $\mathcal{P}$, and uniform preference weights $\lambda^1 = \mathbf{1}/m$ (Line 3). At the beginning of each epoch, hard and soft QoS constraints are jointly enforced to filter the candidate set V_C into a feasible selection pool V_S (Line 5). During the inner roll-out (Lines 7–14), each agent observes a state augmented with its QoS profile and the current preference vector λ^k (Line 9), executes a joint action, and receives per-objective rewards r_j^t aggregated via objective-wise aggregators $\mathcal{A}_j$, alongside a penalty term for hard constraint violations (Lines 11–12). The cumulative per-objective return vector $\mathbf{G}^e$ of each episode is evaluated for Pareto dominance and, if non-dominated, inserted into $\mathcal{P}$ with crowding-distance pruning to capacity P_{max} (Lines 16–19). Multi-Objective GAE is then computed in a backward recurrence for each objective j independently (Lines 21–27), producing per-objective advantage estimates $\hat{A}_{j,t}$ and value targets $\hat{R}_{j,t}$, which are aggregated into a single scalar advantage $\hat{A}_t$ via the current preference weights and normalized (Line 28). The PPO update phase (Lines 30–33) optimizes the shared

Algorithm 3 MAPPO-QoS+MO Selection.

```

1: Input:  $V_C; \{\theta_{\min}(j), \kappa_j, f_j, \mathcal{A}_j\}_{j=1}^m$ ; soft weights  $\{w_j\}$ , threshold  $\theta_{\text{utility}}$ ; hyperparameters  $\gamma, \lambda_{\text{gae}}, \epsilon_{\text{clip}}, \beta, \eta_\phi, \eta_\psi, \eta, P_{\max}$ 
2: Output:  $\pi_\phi, \{V_{\psi_j}\}_{j=1}^m$ , Pareto archive  $\mathcal{P}$ 
3: Init. shared  $\pi_\phi$ , critics  $\{V_{\psi_j}\}$ ,  $\mathcal{P} \leftarrow \emptyset$ ,  $\lambda^1 \leftarrow 1/m$ ,  $HV^0 \leftarrow 0$  ▷ Uniform initial preference; empty archive
4: for epoch  $k = 1, \dots, K$  do
5:    $V_S \leftarrow \{v_i \in V_C : \bigwedge_{j \in J_{\text{hard}}} q_i(j) \geq \theta_{\min}(j) \wedge \sum_{j \in J_{\text{soft}}} w_j f_j(q_i(j)) \geq \theta_{\text{utility}}\}$  ▷ Feasibility pre-filter (hard QoS + soft utility)
6:    $\mathcal{B} \leftarrow \emptyset$  ▷ Reset rollout buffer
7:   for episode  $e = 1, \dots, M$  do
8:     for  $t = 0, \dots, T-1$  do
9:        $o_i^t \leftarrow [s^t, \text{QoS}(v_i), \lambda^k]; a_i^t \sim \pi_\phi(\cdot | o_i^t), \forall i \in V_S$  ▷ Preference-conditioned observation; sample actions
10:      Execute  $a^t$ ; observe  $s^{t+1}, d^t, \{q_i^t(j)\}$  ▷ Step the environment
11:       $\bar{q}_j^t \leftarrow \mathcal{A}_j(\{q_i^t(j)\})$ ,  $r_j^t \leftarrow \sum_i f_j(q_i^t(j))$  ▷ Per-objective aggregation and vector reward
12:       $R^t \leftarrow \sum_j \lambda_j^k r_j^t - \sum_j \kappa_j [\theta_{\min}(j) - \bar{q}_j^t]_+^2$  ▷ Scalarized reward with quadratic constraint penalty
13:      Store  $(s^t, a^t, \log \pi_{\phi_{\text{old}}}, R^t, \{r_j^t\}, s^{t+1}, d^t)$  in  $\mathcal{B}$  ▷ Cache transition for the update phase
14:     end for
15:      $G^e \leftarrow [\sum_t \gamma^t r_j^t]_{j=1}^m$  ▷ Discounted per-objective episodic return vector
16:     if  $G^e$  non-dominated in  $\mathcal{P}$  then
17:        $\mathcal{P} \leftarrow (\mathcal{P} \setminus \{G \prec G^e\}) \cup \{G^e\}$ ; prune to  $P_{\max}$  by crowding ▷ Archive insertion; drop dominated, cap size
18:     end if
19:     end for
20:     % Multi-objective GAE (backward recurrence)
21:     for  $j = 1, \dots, m$  do
22:        $\hat{A}_{j,T} \leftarrow 0$ ; for  $t = T-1$  downto 0: ▷ Per-objective advantage estimation
23:          $\delta_{j,t} \leftarrow r_j^t + \gamma(1-d^t)V_{\psi_j}(s^{t+1}) - V_{\psi_j}(s^t)$  ▷ TD residual
24:          $\hat{A}_{j,t} \leftarrow \delta_{j,t} + \gamma\lambda_{\text{gae}}(1-d^t)\hat{A}_{j,t+1}$  ▷ GAE accumulation
25:          $\hat{R}_{j,t} \leftarrow \hat{A}_{j,t} + V_{\psi_j}(s^t)$  ▷ Per-objective value target
26:       end for
27:        $\hat{A}_t \leftarrow \sum_j \lambda_j^k \hat{A}_{j,t}$ ; normalize  $\hat{A}_t$  ▷ Scalarize advantages under current preference
28:     % PPO updates
29:     for  $u = 1, \dots, U$  do
30:       Sample minibatch;  $\rho_t \leftarrow \pi_\phi(a_i^t | o_i^t) / \pi_{\phi_{\text{old}}}(a_i^t | o_i^t)$  ▷ Importance ratio
31:        $\phi \leftarrow \phi + \eta_\phi \nabla_\phi [\mathcal{L}^{\text{CLIP}}(\rho_t, \hat{A}_t; \epsilon_{\text{clip}}) + \beta \mathcal{H}[\pi_\phi]]$  ▷ Clipped actor update with entropy bonus
32:        $\psi_j \leftarrow \psi_j - \eta_\psi \nabla_{\psi_j} (V_{\psi_j}(s^t) - \hat{R}_{j,t})^2, \forall j$  ▷ Per-objective critic regression
33:     end for
34:     % Adaptive preference update
35:      $v_j \leftarrow \mathbb{E}[\theta_{\min}(j) - \bar{q}_j]_+, \forall j; V \leftarrow \sum_{j'} v_{j'}$  ▷ Mean residual violation per objective
36:      $\bar{v}_j \leftarrow \lambda_j^k$  if  $V < \epsilon_0$  else  $v_j/V$  ▷ Violation-proportional target (fallback if all satisfied)
37:      $\lambda^{k+1} \leftarrow (1-\eta)\lambda^k + \eta\bar{v}$ ; renormalize ▷ Smoothed preference shift toward violated objectives
38:      $HV^k \leftarrow HV(\mathcal{P}; \mathbf{z}_{\text{ref}})$  ▷ Hypervolume of current archive
39:     if  $k \geq 2$  and  $|HV^k - HV^{k-1}| < \epsilon$  and  $\max_j v_j < \epsilon_v$  then break ▷ Stop on HV plateau + near-feasibility
40:   end if
41: end for

```

actor via the clipped surrogate objective augmented with an entropy bonus $\beta \mathcal{H}[\pi_\phi]$, while each per-objective critic V_{ψ_j} is updated independently by minimizing its squared value error. Finally, the preference vector λ^{k+1} is adaptively reweighted toward objectives with the largest hard-constraint violations (Lines 35–37), and the hypervolume indicator HV^k is recomputed over $\mathcal{P}$ (Line 38). Training terminates early when both the hypervolume improvement and the maximum constraint violation fall below their respective tolerances ϵ and ϵ_v (Line 39).

After selecting the optimal agent subset modeled as a reduced MDP E_Q , we train them using Co-MADDPG which enables agents to learn coordinated policies while maintaining decentralized execution. The centralized critic provides a global perspective for learning, while individual actors focus on their optimal strategies. This approach effectively balances inter-agent dependencies, leading to an optimized service composition.

4.4.3 Optimization of Service Composition

After discovery and selection based on context constraints and QoS, the composition phase employs a unified MADDPG-GNN algorithm that integrates graph-based relational learning with cooperative multi-agent reinforcement learning to address QoS-aware ser-

vice composition in IoT environments. The model captures inter-service dependencies through GAT and leverages these representations to guide decentralized decision-making under a centralized training paradigm.

Graph Attention Encoder

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$ denote the service dependency graph, where $\mathcal{V} = \{v_1, \dots, v_N\}$ is the set of N candidate services, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the edge set, and $\mathbf{A} \in \mathbb{R}^{N \times N}$ is the weighted adjacency matrix. $A_{ij} = \text{sim}(\hat{x}_i, \hat{x}_j) \cdot \mathbb{I}[\text{sim}(\hat{x}_i, \hat{x}_j) \geq \delta]$. Each of the H heads (per-head dimension $d_h = d_{\text{out}}/H$) applies the shared projection $h_i = Wx_i$, then computes additive attention scores

$$e_{ij}^{(k)} = \text{LeakyReLU} \left(\mathbf{a}_{\text{src}}^{(k)\top} \mathbf{h}_i^{(k)} + \mathbf{a}_{\text{dst}}^{(k)\top} \mathbf{h}_j^{(k)} \right), \quad (4.15)$$

masked to graph neighbors and normalized via softmax:

$$\alpha_{ij}^{(k)} = \frac{\exp(e_{ij}^{(k)})}{\sum_{l \in \mathcal{N}(i)} \exp(e_{il}^{(k)})}, \quad j \in \mathcal{N}(i). \quad (4.16)$$

The head output and multi-head concatenation are

$$\tilde{h}_i^{(k)} = \sum_{j \in \mathcal{N}(i)} \alpha_{ij}^{(k)} h_j^{(k)}, \quad h'_i = \left\|_{k=1}^H \tilde{h}_i^{(k)} \right\|. \quad (4.17)$$

Stacking L such layers with layer normalization and ELU activation,

$$h_i^{(\ell)} = \text{ELU} \left(\text{LayerNorm}(\text{GATLayer}^{(\ell)}(h_i^{(\ell-1)}, A)) \right), \quad (4.18)$$

yields per-node relational embeddings $z_i = h_i^{(L)} \in R^{d_z}$ and a global graph embedding:

$$z_{\mathcal{G}} = \text{MLP}_{\text{read}} \left(\frac{1}{N} \sum_{i=1}^N z_i \right) \in R^{d_z}. \quad (4.19)$$

The mean embedding $\bar{z} = \frac{1}{N} \sum_j z_j$ and $z_{\mathcal{G}}$ serve as relational context fed to both actors and critic.

QoS Aggregation and Composition Reward

Let $Q_{\text{sel}} \in R^{M \times C}$ stack the QoS vectors of the M selected services. The composite value $\hat{q}_c$ for attribute c follows attribute-dependent aggregation:

$$\hat{q}_c = \begin{cases} \sum_{k=1}^M q_{k,c} & c \in \{\text{RT}, \text{Lat}\} \text{ (sequential)} \\ \max_k q_{k,c} & c \in \{\text{RT}, \text{Lat}\} \text{ (parallel)} \\ \prod_{k=1}^M \text{clip}(q_{k,c}/100, 0.01, 1) \cdot 100 & c \in \{\text{Avail}, \text{Rel}\} \\ \min_k q_{k,c} & c = \text{Thru} \\ \frac{1}{M} \sum_{k=1}^M q_{k,c} & \text{otherwise} \end{cases} \quad (4.20)$$

These rules reflect standard composition semantics: additive accumulation of delays on sequential paths, independent-failure probability multiplication for reliability metrics,

and pipeline-bounded throughput. Pool-level bounds $q_c^{\min}, q_c^{\max}$ are precomputed, and each composite value is mapped to $[0, 1]$ with direction-aware scaling:

$$\bar{q}_c = \text{clip} \left(\begin{cases} 1 - \frac{\hat{q}_c - q_c^{\min}}{q_c^{\max} - q_c^{\min} + \varepsilon} & c \in \{\text{RT}, \text{Lat}\} \\ \frac{\hat{q}_c - q_c^{\min}}{q_c^{\max} - q_c^{\min} + \varepsilon} & \text{otherwise} \end{cases}, 0, 1 \right). \quad (4.21)$$

The scalar composition reward is the weighted sum

$$r = \sum_{c=1}^C w_c \bar{q}_c, \quad w_c \geq 0, \sum_c w_c = 1, \quad (4.22)$$

with (w_c) configured to emphasize availability (0.20), response time, reliability, and latency (0.15 each), throughput and successability (0.10 each), and compliance, best-practices, documentation (0.05 each). This shared reward drives all agents cooperatively through the centralized critic, which performs implicit credit assignment via the joint action gradient.

MADDPG Actor-Critic with GNN Integration

Algorithm 4 describes the MADDPG-GNN for IoT service composition. The algorithm is initialized by constructing a service graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, A)$ from the QoS similarity matrix, instantiating the GAT encoder f_ψ , actor networks $\{\mu_{\theta_i}\}$, centralized critic Q_ϕ , and their respective target copies, along with an empty replay buffer $\mathcal{D}$ and initial exploration noise $\sigma \leftarrow \sigma_0$ (Lines 4-7). At each environment step, actions and service slot assignments are selected via masked greedy argmax with additive Gaussian noise (Lines 11-12), and a composition reward r^t is computed from the QoS attributes of the assigned services (Lines 13-14), with the resulting transition tuple stored in $\mathcal{D}$ (Line 16). Once the buffer reaches the minimum batch size B , a minibatch is sampled and GAT embeddings Z , z_g , and $\bar{z}$ are recomputed from the current graph (Lines 19-21). The centralized critic is updated by minimizing the Bellman residual $\mathcal{L}_{\text{critic}}$ against bootstrapped targets $y^{(b)}$ constructed from target actor and critic networks (Lines 23-27), with the GAT encoder parameters ψ updated jointly through the same loss (Line 28). Each actor θ_i is then updated via the deterministic policy gradient applied to $\mathcal{L}_{\text{actor}}^{(i)}$, with gradients selectively detached for all agents $j \neq i$ to isolate per-agent learning (Lines 30-35). Finally, target networks are smoothed via Polyak averaging and the exploration noise is geometrically decayed (Lines 36-39).

4.4.4 Complexity Analysis

To address practical deployability concerns, we analyze the computational cost of GAMMA, and discusses how it enables execution in realistic IoT/edge settings. Let N denote the size of the service pool, d the embedding dimension, H the number of attention heads, $|\mathcal{E}|$ the number of edges in the service graph, and M the number of agents. The per-step cost of each stage is:

- *Stage 1 (MAPPO + multi-head self-attention + soft K-Means)*: relies on the time execution of MAPPO $\mathcal{O}(N^2d)$ that depends on the number of services N and the

Algorithm 4 MADDPG-GNN for Service Composition

```

1: Inputs Service pool QoS  $Q \in R^{N \times 9}$ , config:  $\gamma, \tau, \sigma_0, \sigma_{\min}, \lambda_{\text{decay}}$ 
2: Outputs Trained actors  $\{\mu_{\theta_i}\}_{i=1}^M$ , critic  $Q_\phi$ , GAT encoder  $f_\psi$ 
3: Initialize:
4:   Build service graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E}, A)$  from  $Q$  using QoS similarity ▷ Edges from pairwise QoS similarity
5:   Initialize GAT encoder  $f_\psi$ , actors  $\{\mu_{\theta_i}\}$ , critic  $Q_\phi$  ▷ Shared encoder, per-agent actors, centralized critic
6:   Copy to targets:  $\theta'_i \leftarrow \theta_i, \phi' \leftarrow \phi$  ▷ Target networks for stable bootstrapping
7:   Initialize replay buffer  $\mathcal{D} \leftarrow \emptyset$ , noise  $\sigma \leftarrow \sigma_0$  ▷ Off-policy memory; initial exploration scale
8: for episode = 1, ...,  $E_{\max}$  do
9:   Observe initial states  $\{s_i^0\}_{i=1}^M$ 
10:  for step  $t = 0, \dots, T - 1$  do
11:    Action Selection
12:     $I^t, a^t \leftarrow \text{SelectActions}(\{s_i^t\}, \{\mu_{\theta_i}\}, f_\psi, \sigma, \text{explore}=\text{True})$  ▷ Selected service indices + actions with exploration noise
13:    Reward Computation
14:     $r^t, \hat{q}^t \leftarrow \text{CompositionReward}(Q[I^t])$  ▷ Aggregate QoS of the chosen composition
15:    Observe next states  $\{s_i^{t+1}\}$ , done flag  $d^t$ 
16:    Store  $(\{s_i^t\}, a^t, r^t, \{s_i^{t+1}\}, d^t)$  in  $\mathcal{D}$  ▷ Cache transition
17:    if  $|\mathcal{D}| \geq B$  then ▷ Begin learning once buffer is warm
18:      Sample minibatch  $\mathcal{B} = \{(s^{(b)}, a^{(b)}, r^{(b)}, s'^{(b)}, d^{(b)})\}_{b=1}^B$  from  $\mathcal{D}$  ▷ Uniform random sampling
19:      Recompute GAT embeddings
20:       $Z, z_{\mathcal{G}} \leftarrow f_\psi(X, A); \quad \bar{z} \leftarrow \frac{1}{N} \sum_j z_j$  ▷ Node embeddings, graph summary, mean-pooled context
21:      Update Critic
22:      for  $b = 1, \dots, B$  do
23:         $a_i'^{(b)} \leftarrow \mu_{\theta_i'}(s_i'^{(b)}, \bar{z}, z_{\mathcal{G}})$  for each agent  $i$  ▷ Target actions
24:         $y^{(b)} \leftarrow r^{(b)} + \gamma(1 - d^{(b)})Q_{\phi'}(s_{1:M}'^{(b)}, a_{1:M}'^{(b)}, \bar{z}_{1:M}, z_{\mathcal{G}})$  ▷ TD target from target critic
25:      end for
26:       $\mathcal{L}_{\text{critic}} \leftarrow \frac{1}{B} \sum_{b=1}^B (Q_\phi(s_{1:M}^{(b)}, a_{1:M}^{(b)}, \bar{z}_{1:M}, z_{\mathcal{G}}) - y^{(b)})^2$  ▷ Centralized MSE Bellman loss
27:      Update  $\phi, \psi$  by minimizing  $\mathcal{L}_{\text{critic}}$  ▷ Joint critic + GAT update
28:      Update Actors
29:      for agent  $i = 1, \dots, M$  do
30:        Recompute  $Z, z_{\mathcal{G}} \leftarrow f_\psi(X, A)$  ▷ Fresh embeddings for the policy gradient
31:         $\hat{a}_j \leftarrow \begin{cases} \mu_{\theta_j}(s_j, \bar{z}, z_{\mathcal{G}}) & j = i \\ \mu_{\theta_j}(s_j, \bar{z}, z_{\mathcal{G}})_{\text{detach}} & j \neq i \end{cases}$  ▷ Differentiate only through agent  $i$ ; freeze others
32:         $\mathcal{L}_{\text{actor}}^{(i)} \leftarrow -\frac{1}{B} \sum_{b=1}^B Q_\phi(s_{1:M}^{(b)}, \hat{a}_{1:M}^{(b)}, \bar{z}_{1:M}, z_{\mathcal{G}})$  ▷ Deterministic policy gradient objective
33:        Update  $\theta_i$  by minimizing  $\mathcal{L}_{\text{actor}}^{(i)}$ 
34:      end for
35:      Soft Target Update
36:       $\theta'_i \leftarrow \tau\theta_i + (1 - \tau)\theta_i'$  for all  $i = 1, \dots, M$  ▷ Polyak averaging of actor targets
37:       $\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$  ▷ Polyak averaging of critic target
38:      Noise Decay
39:       $\sigma \leftarrow \max(\sigma_{\min}, \sigma \cdot \lambda_{\text{decay}})$  ▷ Anneal exploration toward floor
40:    end if
41:  end for
42: end for

```

embedded dimension d , multi-head self-attention $\mathcal{O}(NdH)$ and NdH stands, and soft K -Means $\mathcal{O}(NK_{cl})$ where K_{cl} which results to $\mathcal{O}(N^2d + NdH + NK_{cl})$.

- *Stage 2 (MAPPO-QoS+MO)*: $\mathcal{O}(|V_S| m + m |\theta_\pi|)$ per update, with $|V_S|$ the sub-dataset of services satisfying the QoS hard and soft constraints and m objectives; the m per-objective critics scale linearly in m .
- *Stage 3 (MADDPG-GNN)*: $\mathcal{O}(LH|\mathcal{E}|d_h + M|\theta_\mu| + |\theta_Q|)$ per update, where the sparse GAT encoder (edge threshold τ_{edge}) keeps $|\mathcal{E}| \ll N^2$.

The use of sparse attention and the upstream filtering ($N \rightarrow |V_C| \rightarrow |V_S|$) keeps overall complexity tractable even for large registries.

Memory complexity. The full GAMMA pipeline requires under 1 GB of memory in our setup: attention encoder ($\approx 35\text{k}$ parameters), three per-objective critics ($\approx 100\text{k}$ each), GAT encoder ($\approx 12\text{k}$), six MADDPG actors ($\approx 50\text{k}$ each), and the centralized critic ($\approx 200\text{k}$), plus a replay buffer of 10^5 transitions (≈ 80 MB). This is well aligned within the capacity of commodity fog nodes (e.g., industrial gateways with 4–8 GB RAM). Table 4.1 summarizes the computational profile of each stage.

Table 4.1: Computational profile of GAMMA on QWS2.

Stage	Parameters	Peak RAM	Episodes to converge	Wall-clock
Stage 1	$\approx 75\text{k}$	$\approx 220\text{ MB}$	≈ 500	$\approx 6\text{ min}$
Stage 2	$\approx 420\text{k}$	$\approx 310\text{ MB}$	$\approx 30\text{ (of 60)}$	$\approx 8\text{ min}$
Stage 3	$\approx 510\text{k}$	$\approx 460\text{ MB}$	$\approx 200\text{ (of 2000)}$	$\approx 14\text{ min}$
Total	$\approx 1.0\text{ M}$	$< 1\text{ GB}$	—	$\approx 28\text{ min}$

Deployment feasibility. GAMMA is explicitly designed for edge/fog deployment through the CTDE paradigm: the centralized critic and GAT encoder are required only during training (typically executed offline on a fog or cloud node), whereas at inference time each agent runs only its local actor network conditioned on its observation and the GAT-aggregated neighborhood embedding. The inference cost per decision is therefore reduced to a single forward pass through a lightweight MLP (≤ 128 hidden units, $\leq 50\text{k}$ parameters per agent), executable in sub-millisecond latency on commodity edge hardware. The service graph is rebuilt incrementally per episode, and attention coefficients are re-estimated only over the local neighborhood $\mathcal{N}_i$, which scales linearly with the average node degree rather than with N .

4.5 Low-Level Interaction

Low-level agents act as interconnected decision-makers that learn decentralized, lightweight policies from local observations and shared rewards to optimize both individual and global system performance [19]. A low-level agent is modeled as $A_i = \langle S_i, O_i, \Sigma, T_i, R_i, \pi_{\theta_i} \rangle$, where:

- S_i : The state space comprises local environmental variables that define the system’s current conditions, guiding agent decision-making and optimizing service composition.
- O_i : Observations refer to the subset of the state space S_i that is accessible to an agent, providing partial information to guide its decision-making and interactions within the system.
- Σ : The action space defines the set of possible actions an agent can execute, determining its interactions and influence within the system.
- $T_i(s' | s, \alpha, R_{\text{shared}})$: The transition function defines how an agent’s actions modify the environment by mapping the current state s and chosen action to a new state $s' = s + \Delta s(\alpha, R_{\text{shared}})$ - Δs is the state update caused by the agent’s action and the shared reward R_{shared} effect- shaping system dynamics over time. Additionally, the shared reward influences the new state by reinforcing cooperative behaviors and aligning agent decisions with overall system objectives.
- $R_i(s, \alpha)$: The reward function integrates both individual and shared rewards, assigning incentives or penalties based on actions to guide policy optimization, ensuring agents achieve both their own objectives and overall system efficiency $R_i = R_{\text{individual}} + \lambda R_{\text{shared}}$ where:

- $R_{\text{individual}}$: Local reward for optimizing device-level performance.

- R_{shared} : Global reward for ensuring system-wide efficiency.
- λ : Weight balancing local and shared objectives.
- π_{θ_i} : The policy function, parameterized by Double Deep Q-Network (DDQN) neural networks, maps an agent’s observed state to an optimal action by estimating Q-values, enabling efficient decision-making through value-based reinforcement learning. $\pi_{\theta_i} : S_i \rightarrow \Sigma$ where θ_i is the parameters of the trained DDQN Q-network.

Centralized Critic for Joint Learning. Each agent’s critic network evaluates joint actions, ensuring that learning occurs in the context of the decisions of other agents. The centralized critic enables agents to capture inter-agent dependencies, improving overall coordination. The Q-value function for each agent is updated using a centralized critic that considers global information:

$$Q_i(s, \alpha_1, \dots, \alpha_N) = \mathbb{E} [R_i + \gamma Q_i(s', \mu_1(s'), \dots, \mu_N(s'))], \text{ where:}$$

- $Q_i(s, \alpha_1, \dots, \alpha_N)$ represents the Q-value for agent a_i , conditioned on all agents’ actions.
- R_i is the immediate reward received by agent a_i .
- γ is the discount factor for future rewards.
- s' is the next state, and $\mu_i(s')$ is the next action according to the agent’s policy.

Decentralized Execution. After training, agents perform actions independently using their trained actor networks. At execution time, agents no longer require access to global information, as each operates based on its learned policy. The deterministic policy for each agent is given by $a_i = \mu_{\theta_i}(s_i)$ where:

- μ_{θ_i} is the actor-network of agent a_i , parameterized by $\theta_i \leftarrow \theta_i^{\text{actor}} \cdot \nabla_{\theta_i} J(\theta_i)$.
- s_i represents the agent’s local observation of the state.

Composition Through Policy Updates. To ensure that agents refine their policies while considering compositions with other agents, each agent updates its actor-network using a policy gradient method to maximize its expected return. The gradient is computed as $\nabla_{\theta_i} J(\theta_i) = \mathbb{E} [\nabla_{a_i} Q_{\phi_i}(s, a_1, \dots, a_N) \cdot \nabla_{\theta_i} \pi_{\theta_i}(s_i)]$, where:

- $J(\theta_i)$ is the objective function for agent i (expected cumulative reward).
- Q_{ϕ_i} is the centralized critic evaluating joint actions.
- $\pi_{\theta_i}(s_i)$ is the actor network generating agent i ’s action.

Stabilized Learning with Target Networks. To enhance stability, target networks are updated using soft updates rather than direct weight replacements, preventing instability from rapidly changing Q-value estimates. The soft update rule is $\theta_i^{\text{target}} \leftarrow \tau \cdot \theta_i + (1 - \tau) \cdot \theta_i^{\text{target}}$ and $\phi_i^{\text{target}} \leftarrow \tau \cdot \phi_i + (1 - \tau) \cdot \phi_i^{\text{target}}$, where: τ is a small update rate (e.g., $\tau = 0.01$), ensuring gradual parameter adjustments and $\theta_i^{\text{target}}, \phi_i^{\text{target}}$ are the parameters of the target actor and critic networks.

Shared Reward Policy Learning for Interactive Agents

Interactive IoT agents must learn policies that enable them to independently optimize their objectives while effectively coordinating with other agents. The shared reward promotes cooperative local policies, ensuring consistent service parameters and balanced energy usage across IoT nodes. This coherence stabilizes the input environment for the mid-level Co-MADDPG agents responsible for service composition. As a result, Co-MADDPG achieves smoother convergence, reduced coordination conflicts, and more resilient orchestration under dynamic conditions. The Q-function update rule in DDQN incorporates a shared reward component:

$$Q_{\theta_i}(s_i, \alpha_i) = Q_{\theta_i}(s_i, \alpha_i) + \beta [(R_{\text{individual}} + \lambda R_{\text{shared}}) + \gamma Q_{\theta'_i}(s', \alpha') - Q_{\theta_i}(s_i, \alpha_i)], \text{ where:} \quad (4.23)$$

- $Q_{\theta_i}(s_i, \alpha_i)$: The Q-value estimate for state-action pair (s_i, α_i) .
- $R_{\text{individual}} + \lambda R_{\text{shared}}$: The reward received after executing action α_i .
- $Q_{\theta'_i}(s', \alpha')$: The target Q-value from the **target network**, stabilizing updates.
- γ and β : Discount factor showing future reward importance and the learning rate.
- s' and α' represent the next state and the next selected action, respectively.

The agent selects the best action using the DDQN model by evaluating the Q-values for all possible actions and choosing the one that maximizes the expected cumulative reward: $\alpha_i = \arg \max_{\alpha} Q_{\theta_i}(s_i, \alpha)$ according to Equation 4.23, which modifies the agent's state according to the transition function $T_i(s' | s, \alpha, R_{\text{shared}})$.

Training Process for Shared Reward Learning.

The Q-value function quantifies the expected cumulative reward an agent can obtain by selecting an action α_i in a given state s_i , following an optimal policy, adapting dynamically to their actions and the evolving environment.

Algorithm 5 describes the learning procedure of the low-level DDQN agents with shared reward coupling. The algorithm starts by specifying the inputs and outputs (lines 1–2), followed by the initialization of the online Q-network, target Q-network, and replay buffer for each agent to ensure stable off-policy learning and reduced temporal correlation (lines 3–8). The main hyperparameters governing exploration, learning dynamics, reward coupling, and target updates are then defined (line 9). For each training episode, agents first observe their initial local states (lines 10–13). At every time step, each agent selects a discrete action using an ϵ -greedy policy to balance exploration and exploitation based on its current Q-function (lines 14–22). The selected actions are executed in the environment, producing the next state along with both individual and shared reward components that capture local performance and global coordination objectives; these transitions are stored in the replay buffer (line 23). Each agent then samples a mini-batch of past experiences to perform off-policy learning (line 25), computes the greedy next action using the online network as required by Double Q-learning (line 26), and updates its Q-values according to Equation 4.23, which integrates individual and shared rewards into a coupled temporal-difference update (line 27). To maintain learning stability, the target network is softly updated toward the online network (line 28). Finally, the exploration rate is gradually decayed to favor exploitation as training progresses (line 30).

Algorithm 5 DDQN for Low-Level Agent Learning with Shared Reward

```

1: Input: current environment state  $s$ 
2: Output: discrete action maximizing  $Q_{\theta_i}(s, a)$ 
3: Initialization:
4: for each agent  $i$  do
5:   Initialize online Q-network  $Q_{\theta_i}(s, a)$  ▷ Approximates optimal action-value function
6:   Initialize target Q-network  $Q_{\theta'_i}(s, a)$  with  $\theta'_i \leftarrow \theta_i$  ▷ Stabilizes TD learning
7:   Initialize replay buffer  $\mathcal{D}_i$  ▷ Breaks temporal correlations
8: end for
9: Set hyperparameters: exploration rate  $\epsilon$ , learning rate  $\beta$ , discount factor  $\gamma$ , reward coupling factor  $\lambda$ , soft-update factor  $\tau$ 
10: for episode = 1, . . . ,  $M$  do
11:   for each agent  $i$  do
12:     Observe initial state  $S_{i,0}$  ▷ Initial state
13:   end for
14:   for time step  $t = 0, \dots, T - 1$  do
15:     for each agent  $i$  do
16:       Select action  $\alpha_i$  using  $\epsilon$ -greedy policy ▷ Balances exploration and exploitation

$$\alpha_i = \begin{cases} \text{random action,} & \text{w.p. } \epsilon, \\ \arg \max_{\alpha} Q_{\theta_i}(S_{i,t}, \alpha), & \text{otherwise} \end{cases}$$

17:     end for
18:     for each agent  $i$  do
19:       Execute  $\alpha_i$  in the environment
20:       Observe next state  $S_{i,t+1}$  ▷ state transition
21:       Observe  $R_{\text{individual}}$  and  $R_{\text{shared}}$  ▷ Local performance and global coordination signals
22:       Store  $(S_{i,t}, \alpha_i, R_{\text{individual}}, R_{\text{shared}}, S_{i,t+1})$  in  $\mathcal{D}_i$  ▷ Experience tuple
23:     end for
24:     for each agent  $i$  do
25:       Sample mini-batch from  $\mathcal{D}_i$  ▷ Off-policy learning
26:       Compute greedy next action using online network ▷ Double Q-learning action selection

$$\alpha'_j = \arg \max_{\alpha'} Q_{\theta_i}(s'_j, \alpha')$$

27:       Q-value update using Equation (4.23): ▷ TD update with individual-shared reward coupling

$$Q_{\theta_i}(s_j, \alpha_j) \leftarrow Q_{\theta_i}(s_j, \alpha_j) + \beta \left[ (R_{\text{ind},j} + \lambda R_{\text{sh},j}) + \gamma Q_{\theta'_i}(s'_j, \alpha'_j) - Q_{\theta_i}(s_j, \alpha_j) \right]$$

28:       Soft-update target network ▷ Ensures slow and stable target drift

$$\theta'_i \leftarrow \tau \theta_i + (1 - \tau) \theta'_i$$

29:     end for
30:     Decay exploration rate ▷ Gradual transition to exploitation

$$\epsilon \leftarrow \max(\epsilon_{\min}, \epsilon \cdot \text{decay})$$

31:   end for
32: end for

```

4.6 Conclusion

Hi-MAD and GAMMA share an identical three-stage pipeline for context discovery, QoS-aware selection, and service composition, yet they diverge significantly in their agent coordination strategies. Hi-MAD, organizes agents hierarchically by decision scope, using an environmental reduction process. GAMMA, on the other hand, ensure a dynamically structured service interaction graph. It fulfills the shared pipeline using multi-head attention with binary K-means clustering, a multi-objective MAPPO, and a GAT-coupled MADDPG framework.

Chapter 5

Formal Verification & Experimental Validation

5.1 Introduction

The rapid evolution of IoT ecosystems has given rise to architectures that are inherently complex, highly distributed, and adaptive—especially when integrated with MADRL and GAT. Because these advanced systems operate with a high degree of autonomy, constantly engage in dynamic interactions, and rely on partial observability, validating their behavior presents a significant challenge. While traditional testing and simulation methods remain practically useful, they inevitably fall short of guaranteeing absolute correctness. They are frequently hindered by state-space explosion, inherent nondeterminism, and the unpredictable nature of emergent behaviors.

To comprehensively address these limitations and demonstrate the robustness of our proposed approaches, this section is divided into two synergistic parts. First, we employ formal verification to establish a mathematically rigorous foundation that guarantees system correctness. Second, we present an experimental validation to practically demonstrate the real-world performance and efficacy of our models. Together, these dual pillars—mathematical proof and empirical evidence—powerfully highlight the strength and reliability of the methodologies introduced in this thesis.

The remainder of this chapter is organized as follow: Section 5.2 describe the IoT system model verification. Section 5.3 describe the GAMMA framework verification. Section 5.4 describe the MDP reduction process verification. Section 5.5 and 5.6 describe the experimental validation of HiMAD and GAMMA.

5.2 IoT System Architecture Verification

To complement the empirical evaluation, we developed a formal model of the Hi-MAD system structure and its operational lifecycle in *Event-B* and verified it mechanically with the Rodin platform. The objective is threefold: to establish, by machine-checked proof, that the architecture defined in Chapter 3, the tuple $IoT = \langle D, N, C, MAD, SVC \rangle$ is *internally consistent*; that the layered decomposition is introduced by *behavior-preserving refinement*; and that a collection of domain *safety properties* holds in every reachable system state. We adopt a correct-by-construction refinement discipline; the resulting model is considerably larger in scope, jointly covering devices, edge/fog nodes, the communica-

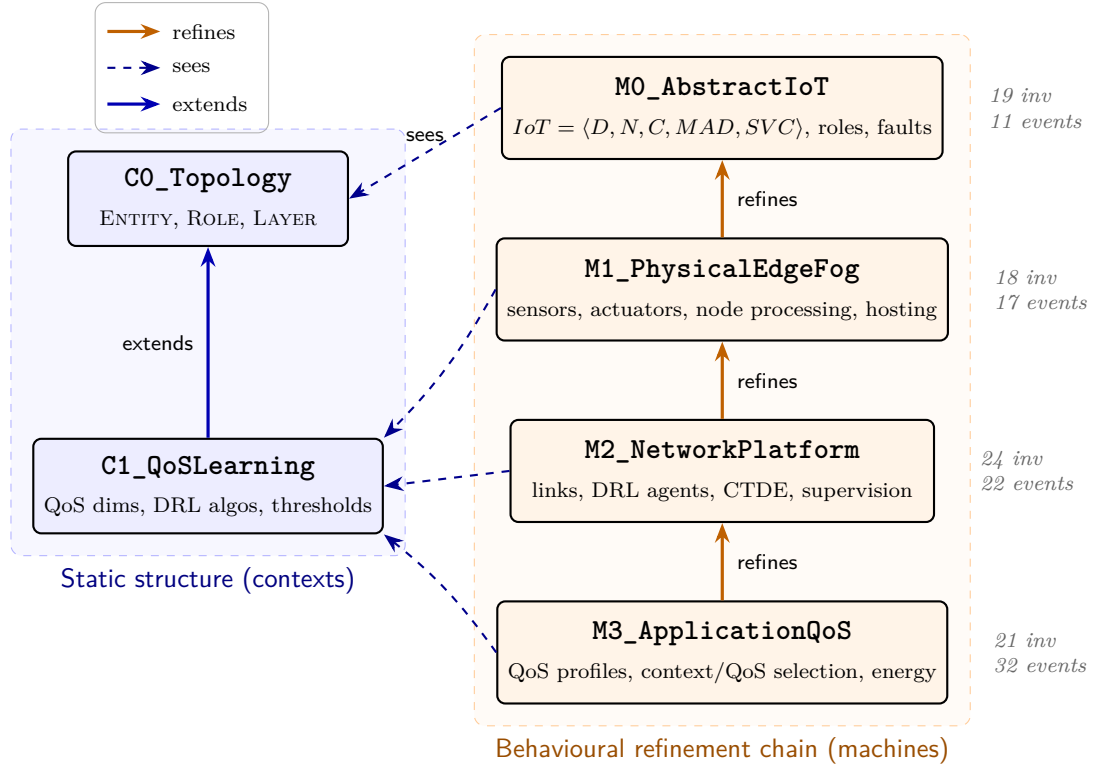


Figure 5.1: Event-B refinement architecture of the Hi-MAD system model.

tion network, the hierarchical multi-agent platform, and QoS-aware services ¹.

5.2.1 Event-B Architecture and Modeling Approach

The development is organized as two *contexts* (the static, immutable domain) and a chain of four *machines* (the dynamic behavior), related by the three structural mechanisms of Event-B: **EXTENDS** (context-to-context, a monotone accumulation of the static theory), **SEES** (machine-to-context, read-only use of the static domain), and **REFINES** (machine-to-machine, a behavioral simulation discharged by proof obligations). Figure 5.1 shows the resulting architecture.

We use *superposition refinement*: each machine adds the state and events of one architectural layer on top of the previous one *without removing or re-typing* the variables it inherits. Consequently the gluing invariants relating successive machines are identities, and every refinement step is data-preserving by construction; the refinement obligations then reduce to showing that each concrete event simulates its abstract counterpart and respects the frame of the abstract model. Two further decisions shape the model. First, the heterogeneous entities of the system are drawn from a single carrier set **ENTITY**, *partitioned* into devices, nodes, agents, sensors and actuators; this yields uniform membership reasoning and a single finiteness assumption $\text{finite}(\text{ENTITY})$, which is what the cardinality- and removal-related obligations depend on. Second, **SERVICE** is kept as a separate carrier set, since services are composed and requested rather than physically deployed like entities.

¹The complete Rodin project, together with the ProB/VisB animation, is available in the project repository [Github link](#)

CONTEXT C0_Topology

SETS

ENTITY, SERVICE, ROLE, LAYER

CONSTANTS

DEVICE, NODE, AGENT, SENSOR,
ACTUATOR, LOW, MID, HIGH,
PHYS_LAYER, EDGE_FOG_LAYER,
NET_LAYER, PLAT_LAYER, APP_LAYER

AXIOMS

axm0_1 : partition(ENTITY, DEVICE, NODE, AGENT,
SENSOR, ACTUATOR)
axm0_2 : partition(ROLE, {LOW}, {MID}, {HIGH})
axm0_3 : partition(LAYER, {PHYS_LAYER},
{EDGE_FOG_LAYER}, {NET_LAYER},
{PLAT_LAYER}, {APP_LAYER})
axm0_4 : DEVICE $\neq \emptyset$
axm0_5 : NODE $\neq \emptyset$
axm0_6 : AGENT $\neq \emptyset$
axm0_7 : SENSOR $\neq \emptyset$
axm0_8 : ACTUATOR $\neq \emptyset$
axm0_9 : finite(ENTITY)

END

MACHINE M0_AbstractIoT

SEES C0_Topology

VARIABLES

active_devices, active_nodes, active_services, active_agents,
device_layer, node_layer, agent_layer, svc_layer, agent_role,
sys_ok, fault_set

INVARIANTS

inv0_1 : active_devices $\subseteq$ DEVICE
inv0_5 : device_layer $\in$ active_devices $\rightarrow$ LAYER
inv0_12 : $\forall d \cdot d \in \text{active_devices} \Rightarrow \text{device_layer}(d) = \text{PHYS_LAYER}$
inv0_16 : $\forall a1, a2 \cdot (a1 \in \text{active_agents} \wedge \text{agent_role}(a1) = \text{HIGH} \wedge$
 $a2 \in \text{active_agents} \wedge \text{agent_role}(a2) = \text{HIGH}) \Rightarrow a1 = a2$
inv0_17 : sys_ok = TRUE $\Leftrightarrow$ fault_set = $\emptyset$
inv0_18 : $\forall a \cdot a \in \text{active_agents} \wedge \text{agent_role}(a) = \text{HIGH} \Rightarrow a \notin \text{fault_set}$
inv0_19 : finite(fault_set)

EVENTS

INITIALISATION

Device_Register

Agent_Register_HIGH

Service_Deploy

END

5.2.2 The static domain: contexts

C0_Topology fixes the topological vocabulary. It declares the carrier sets ENTITY, SERVICE, ROLE and LAYER, the partition of ENTITY into DEVICE, NODE, AGENT, SENSOR, ACTUATOR, the three hierarchical roles {LOW, MID, HIGH}, and the five architectural layers introduced. (PHYS, EDGE_FOG, NET, PLAT, APP). Non-emptiness of each entity kind and *finite*(ENTITY) complete the context (4 sets, 13 constants, 9 axioms).

C1_QoSLearning EXTENDS C0_Topology and adds the learning- and quality-related theory: the enumerations QOS_DIM (latency, energy, reliability, throughput, availability), COMP_TYPE (atomic/composite), DRL_ALGO (MATD3, MAPPO, MADDPG, DDQN), MGT_FUNC (monitoring, decision-making, adaptation) and LINK_PROTO (MQTT, CoAP, HTTP, gRPC); and the numeric parameters that the dynamics are bounded by—QOS_MIN = 90, CONTEXT_THRESHOLD = 70, ENERGY_MAX = 1000, MAX_AGENTS = 500 and VARIANCE_THRESH = 30, the last being the threshold of the variance-triggered consensus rule of Co-MATD3 (5 sets, 23 constants, 10 axioms).

5.2.3 The behavioral model: the refinement chain

Each machine introduces the state and operations of one layer and re-establishes the global invariants over the enlarged state.

M0_AbstractIoT (SEES C0) is the abstract root. It maintains the active populations *active_devices*, *active_nodes*, *active_agents*, *active_services*, their layer maps, the role map *agent_role*, the boolean health flag *sys_ok* and the *fault_set*, and provides the lifecycle events: registration and removal of each entity kind, fault injection, fault recovery, and health update. The global safety properties (layering, single coordinator, fault-health consistency) are stated here.

M1_PhysicalEdgeFog (SEES C1, REFINES M0) adds the physical and edge-fog detail: sensors and actuators attributed to devices, their readings and commands, the node processing pipeline (*node_input_load*, *node_output_ready*, *node_processing*) and service hosting (*svc_host*, *node_hosts_svc*).

M2_NetworkPlatform (REFINES M1) introduces the network and platform layers: the

Table 5.1: Domain safety properties of the Hi-MAD model.

ID	Property	Invariant(s)	Machine
SP1	Every active entity resides in its designated architectural layer (device→physical, node→edge-fog, agent→platform, service→application).	inv0_12–inv0_15	M0
SP2	At most one high-level (global coordinator) agent exists at any time.	inv0_16	M0
SP3	A high-level coordinator never belongs to the fault set (coordinator robustness).	inv0_18	M0
SP4	The system-health flag is true <i>iff</i> the fault set is empty (health consistency).	inv0_17	M0
SP5	Composition referential integrity: every active service is hosted by a single active node and is recorded in that node’s hosted-service set.	inv1_11, inv1_15, inv1_18	M1
SP6	No dangling device references: sensors and actuators map only to active devices.	inv1_12, inv1_13	M1
SP7	Pipeline safety: a node reports output-ready only when its input load has been cleared.	inv1_17	M1
SP8	Network well-formedness: communication links connect only active entities and contain no self-loops.	inv2_1, inv2_11	M2
SP9	Role-consistent control: each agent’s DRL algorithm and supervisor are fixed by its role (HIGH→MATD3, unsupervised; MID→MAPPO/-MADDPG, supervised by HIGH; LOW→DDQN, supervised by MID), and a HIGH agent carries all management functions.	inv2_12–inv2_14, inv2_17–inv2_19, inv2_23	M2
SP10	QoS and energy bounds: every completed service meets the minimum aggregate QoS ($\geq \text{QOS_MIN}$) and total system energy never exceeds the budget ($\leq \text{ENERGY_MAX}$).	inv3_15, inv3_17	M3

communication relation $\text{comm_links} \subseteq (D \cup N \cup A) \times (D \cup N \cup A)$ with its protocols, the per-agent DRL algorithm and management-function assignment, the supervision relation agent_supervisor , the binding of low-level agents to nodes, monitoring sets, the bounded performance metric and the training_mode flag.

M3_ApplicationQoS (REFINES M2) completes the model with the application layer: per-dimension service quality (svc_qos), composition type and the dependency graph (svc_deps), the two-stage agent selection ($\text{context_selected} \supseteq \text{qos_selected}$), energy accounting (node_energy , total_energy), the request lifecycle (pending_requests , $\text{completed_requests}$) and the resilience accounting of the consensus rule (faulty_tasks , recovered_tasks , action_variance).

5.2.4 Safety Properties

The invariants fall into two groups. The first is a layer of *well-formedness* conditions that type the state—subset constraints on the active populations ($\text{active_devices} \subseteq \text{DEVICE}$, ...) and total/partial-function typings of every attribute map ($\text{agent_role} \in \text{active_agents} \rightarrow \text{ROLE}$, $\text{svc_qos} \in \text{active_services} \times \text{QOS_DIM} \rightarrow \mathbb{N}$, ...). The second group, listed in Table 5.1, are the **domain safety properties** SP1–SP10 that the model is designed to guarantee; each is a machine-checked invariant established at the level indicated.

Several auxiliary invariants reinforce these guarantees, including the disjointness of pending and completed requests ($\text{pending_requests} \cap \text{completed_requests} = \emptyset$, inv3_20), the irreflexivity of the service dependency graph (inv3_18), the recovery bound $\text{recovered_tasks} \leq \text{faulty_tasks}$ (inv3_19), and the containment $\text{qos_selected} \subseteq \text{context_selected}$ (inv3_7) that mirrors the two-stage Context-Discovery/QoS-Selection filtering of the mid level.

5.2.5 Verification by proof obligations

From the model, Rodin generates and tracks the standard families of proof obligations (POs):

- **Well-definedness** (WD); every function application, cardinality and division is only used where it makes sense (argument in the domain, set finite, divisor non-zero).

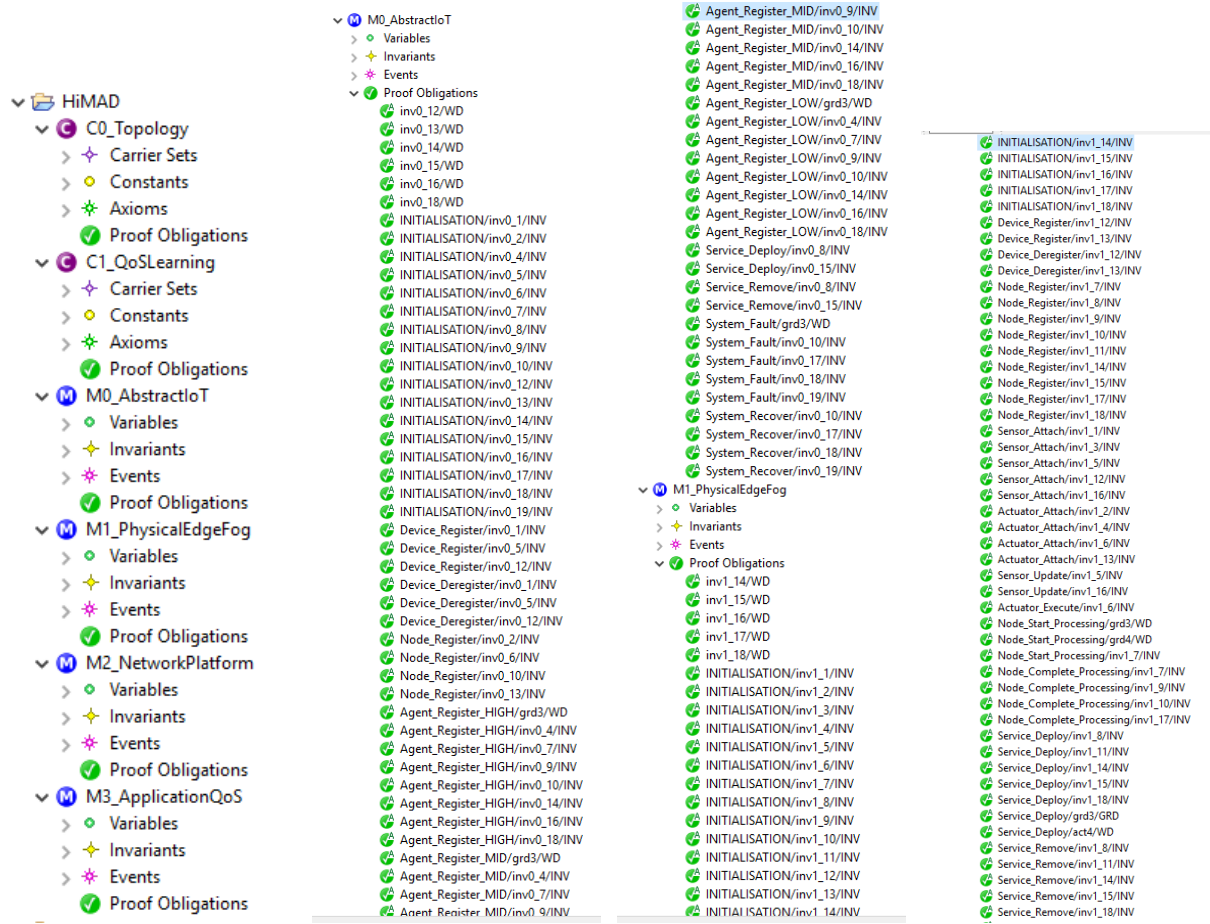
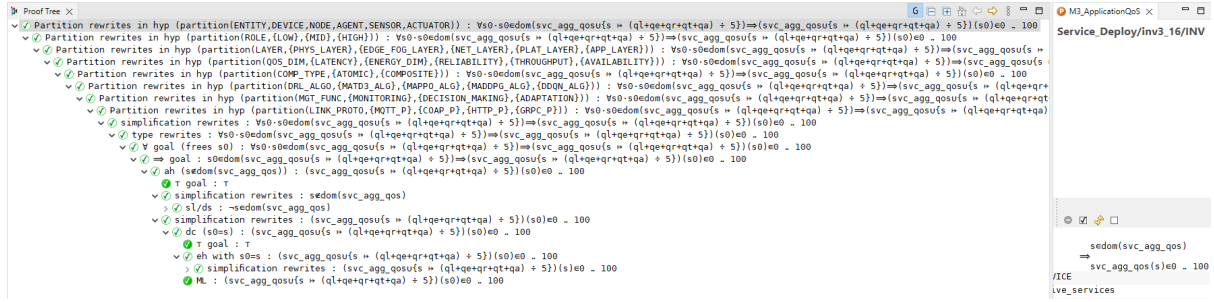


Figure 5.2: Proof Obligations Discharged

- **Invariant preservation (INV)**; every event keeps every invariant true: if it held before the event, it still holds afterwards.
- **Guard strengthening (GRD)**; a refined event may fire only when its abstract counterpart could also fire, so it is never less restrictive.
- **Simulation (SIM)**; a refined event's effect on the shared variables matches what the abstract event would have done.
- **Feasibility (FIS)**; every nondeterministic action has at least one legal value to choose, so the model can never get stuck.
- **Equality/frame (EQL)**; a refined event leaves untouched every abstract variable that its abstract counterpart does not change.

The model was developed iteratively until all generated proof obligations were discharged. The overwhelming majority were proved automatically by Rodin's auto-provers (the Atelier-B provers PP/ML and the integrated SMT solvers Figure 5.3a). A small minority required interactive proof—case splitting, hypothesis addition and instantiation—and concerned, characteristically, the well-definedness of partial-function lookups, finiteness and cardinality, the bound on an integer-division expression, the cascade cleanup performed by entity-removal events (which must subtract the removed entity from every



(a) Proof Tree

Element Name	Total	Auto	Man.	Rev.	Und.
HiMAD	339	331	8	0	0
C0_Topology	0	0	0	0	0
C1_QoS Learning	0	0	0	0	0
M0_AbstractIoT	70	70	0	0	0
M1_PhysicalEdgeFog	68	68	0	0	0
M2_NetworkPlatform	107	103	4	0	0
M3_ApplicationQoS	94	90	4	0	0

(b) Proof Obligations Statistics

Figure 5.3: Proof Obligation in Rodin.

variable keyed on it to preserve referential integrity), and one frame correction where an abstract recovery event was wrongly modifying a retained performance variable. Table ?? reports the per-machine statistics.

Discharging these obligations establishes that the model is type-correct and internally consistent, that every safety property of Table 5.1 is inductively preserved by every event, and that each machine is a sound refinement of its predecessor as shown in Figure 5.2, 5.3b.

5.2.6 Discussion and Conclusion

The formal modeling of the Hi-MAD architecture successfully establishes a mathematically verifiable blueprint. By successfully discharging the INV and WD obligations, we certify the internal consistency of the design. Similarly, the GRD and SIM proofs confirm that each machine is a sound refinement of its predecessor, protecting abstract behavioral rules as deep architectural details are superposed. This Event-B model significantly advances prior formal verification baselines in IoT research. While earlier studies—such as the work by Toman et al.—focused predominantly on basic service interoperability, our formalization validates an entire hierarchical multi-agent structure, a dynamic QoS filtering pipeline, and strict energy/resilience parameters.

It is important to acknowledge the deliberate scope constraints of this formalization. The model evaluates the structural safety and logical correctness of the architecture, leaving the evaluation of numerical learning dynamics (such as DRL convergence speed or exact energy reduction percentages) to our subsequent experimental phase. Ultimately, this creates a deeply complementary validation framework: while empirical simulations demonstrate *how well* the system performs, the Event-B verification unequivocally proves that the coordination it performs is *always safe and well-formed*.

5.3 Environment Size Reduction (GAMMA) Verification

Beyond its empirical performance, the soundness of the context-aware discovery stage of GAMMA, which we term *Environment Size Reduction* (ESR)—is safety-critical: it decides which services are removed before any QoS evaluation or composition takes place. An over-eager filter that discards a feasible service can silently make a downstream composition infeasible, and no later stage can recover it. We therefore contribute a complete **correct-by-construction** formalization of ESR in Event-B, mechanically verified in the Rodin platform and model-checked with ProB. To our knowledge this is the first machine-checked guarantee that the search-space reduction in a MADRL-based service-composition pipeline is **sound** (it removes only non-compliant services), **complete** (it retains every compliant service), **composition-preserving** (it never destroys a feasible workflow), and **scalability-bounded**, while remaining fully discharged by automated provers.

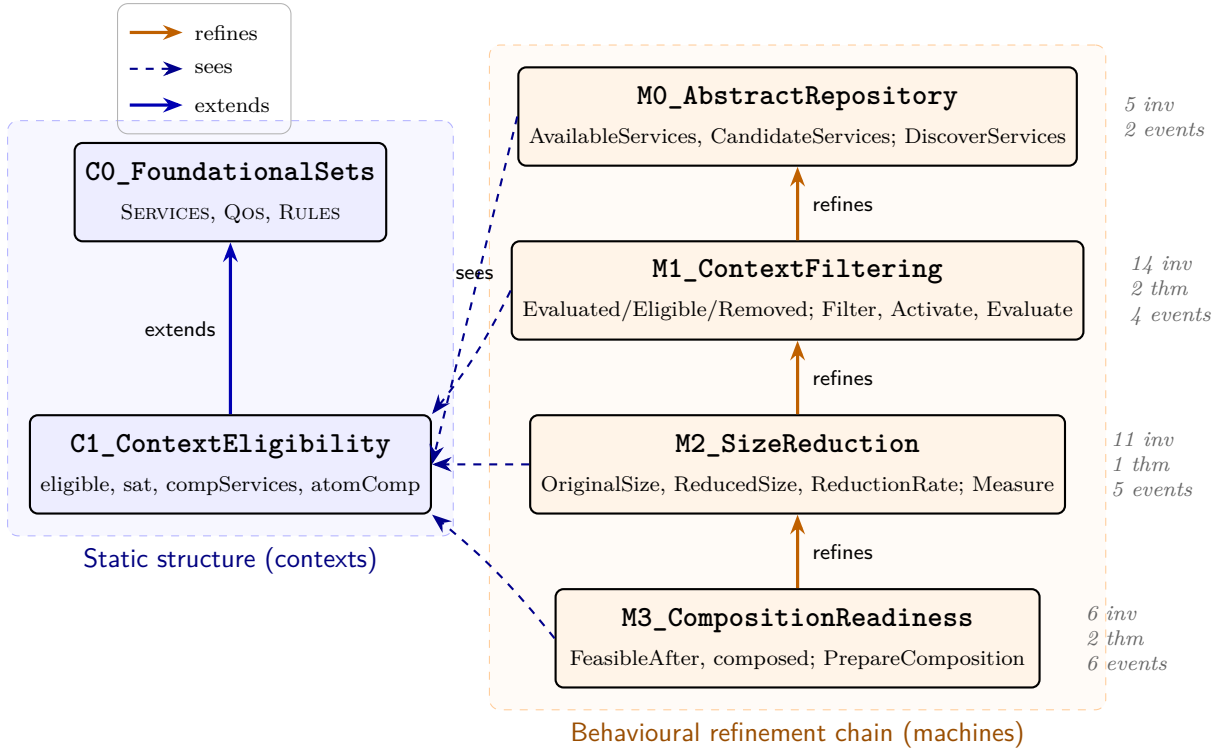


Figure 5.4: Event-B refinement architecture of the Environment Size Reduction (ESR) process

5.3.1 Event-B Architecture and Refinement Strategy

The specification follows the correct-by-construction discipline of Event-B: an abstract model is enriched through a chain of provably consistent refinements, so that every property established at one level is preserved at all subsequent levels. The architecture (Fig. 5.4) comprises two *contexts* fixing the static theory and four *machines* introducing behavior:

- **C0_FoundationalSets** declares the carrier sets (**SERVICES**, **QOS_ATTRIBUTES**, **RULES**, **COMPOSITIONS**), the finite non-empty service registry, and the nine QoS dimensions of the QWS dataset partitioned into benefit and cost attributes.
- **C1_ContextEligibility** (extending C0) defines the QoS value model, the per-attribute thresholds, the context rules, and—crucially—a **single declarative definition** of service eligibility against which every machine is later proven correct.
- **M0_AbstractRepository** models the abstract repository and a non-deterministic discovery step; **M1_ContextFiltering** refines it into context-based filtering; **M2_SizeReduction** adds the quantitative size-reduction metric; and **M3_CompositionReadiness** establishes composition readiness.

All four machines **see** C1_ContextEligibility, so they share one ground-truth notion of eligibility, and each machine **refines** its predecessor. This mirrors the three-stage GAMMA (discovery → selection → composition) under the CTDE paradigm, the program-counter variable *phase* encoding the stage ordering.

<pre> CONTEXT C0_FoundationalSets SETS SERVICES, CONTEXTS, RULES, QOS_ATTRIBUTES, COMPOSITIONS CONSTANTS Registry, N0, nQoS, RT, AV, SU, TH, REL, COMP, BP, LAT, DOC, benefitAttr, costAttr AXIOMS axm0_1 : Registry ⊆ SERVICES axm0_2 : finite(Registry) axm0_2b : finite(SERVICES) axm0_3 : Registry ≠ ∅ axm0_3b : SERVICES ≠ ∅ axm0_4 : partition(QOS_ATTRIBUTES, {RT}, {AV}, {SU}, {TH}, {REL}, {COMP}, {BP}, {LAT}, {DOC}) axm0_5 : nQoS = 9 axm0_6 : card(QOS_ATTRIBUTES) = nQoS axm0_7 : partition(QOS_ATTRIBUTES, benefitAttr, costAttr) axm0_8 : costAttr = {RT, LAT} axm0_9 : benefitAttr = {AV, SU, TH, REL, COMP, BP, DOC} axm0_10 : N0 = card(Registry) axm0_12 : COMPOSITIONS ≠ ∅ END </pre>	<pre> CONTEXT C1_ContextEligibility EXTENDS C0_FoundationalSets CONSTANTS qos, sopt, weight, ruleOf, sat, eligible, thetaUtility, utility, compServices, atomComp AXIOMS axm1_1 : qos ∈ Registry → (QOS_ATTRIBUTES → 0 .. 100) axm1_2 : sopt ∈ QOS_ATTRIBUTES → 0 .. 100 axm1_3 : weight ∈ QOS_ATTRIBUTES → 0 .. 100 axm1_4 : ruleOf ∈ QOS_ATTRIBUTES ⇔ RULES axm1_5 : sat ∈ RULES ⇔ Registry axm1_6 : ∀j, s · (j ∈ QOS_ATTRIBUTES ∧ s ∈ Registry ⇒ ((ruleOf(j) ↦ s) ∈ sat ⇔ qos(s)(j) ≥ sopt(j))) axm1_7 : utility ∈ Registry → 0 .. 100 axm1_8 : thetaUtility ∈ 0 .. 100 axm1_9 : thetaUtility = 35 axm1_10 : eligible ⊆ Registry axm1_11 : ∀s · (s ∈ eligible ⇔ (∀j · (j ∈ QOS_ATTRIBUTES ⇒ (ruleOf(j) ↦ s) ∈ sat))) axm1_12 : finite(eligible) axm1_13 : compServices ∈ COMPOSITIONS → P(Registry) axm1_14 : atomComp ∈ Registry → COMPOSITIONS axm1_15 : ∀v · (v ∈ Registry ⇒ compServices(atomComp(v)) = {v}) END </pre>
--	--

5.3.2 Mathematical Modeling of ESR

Let $R \subseteq \text{SERVICES}$ denote the finite, non-empty registry of $N_0 = |R|$ services, and J the set of nine QoS attributes. After min-max normalization and benefit-form flipping of cost attributes, each service v exposes a value map $q(v) : J \rightarrow [0, 100]$, and each attribute j carries an optimal threshold $s_{\text{opt}}(j)$. A context rule is satisfied iff the corresponding attribute meets its threshold; combined with a soft-utility screen $u(v) \geq \theta_u$, this yields the **eligibility** predicate, the formal ground truth of ESR:

$$\text{eligible} = \left\{ v \in R \mid \left(\forall j \in J : q(v)(j) \geq s_{\text{opt}}(j) \right) \wedge u(v) \geq \theta_u \right\}. \quad (5.1)$$

ESR is then modeled as a deterministic transition system whose state records the available, evaluated, retained (*EligibleServices*) and removed (*RemovedServices*) service sets, together with the size metrics and a composition oracle $compServices : \text{COMPOSITIONS} \rightarrow \mathbb{P}(\text{SERVICES})$. The pipeline executes five events in a fixed order:

ActivateContext $\rightarrow$ EvaluateRules $\rightarrow$ FilterServices $\rightarrow$ MeasureReduction $\rightarrow$ PrepareComposition.

The decisive event, *FilterServices*, commits the partition

$$EligibleServices = Evaluated \cap eligible, \quad RemovedServices = Evaluated \setminus eligible, \quad (5.2)$$

and *MeasureReduction* records the reduction rate. To keep the metric verifiable by automated provers we avoid division by a variable and instead pin the integer percentage $\rho = ReductionRate$ by its floor characterization

$$\rho \cdot N_0 \leq 100(N_0 - |EligibleServices|) < (\rho + 1) \cdot N_0, \quad \rho \in 0..100, \quad (5.3)$$

which is provably equivalent to $\rho = \lfloor 100(N_0 - |EligibleServices|)/N_0 \rfloor$.

5.3.3 Invariants and Refinement Soundness

Correctness is captured by gluing and safety invariants discharged at every refinement level. The central ones are: the subset safety $CandidateServices \subseteq AvailableServices$ (M0); soundness $EligibleServices \subseteq eligible$ and disjointness $EligibleServices \cap RemovedServices = \emptyset$ (M1); the exactness gluing $phase=3 \Rightarrow EligibleServices = Evaluated \cap eligible$ (M1); the bound $ReducedSize \leq OriginalSize$ with $\rho \in 0..100$ (M2); and the composition-feasibility invariants (M3). The refinement of the abstract discovery step by *FilterServices* uses the witness $cand = Evaluated \cap eligible$, so guard-strengthening and action-simulation obligations discharge directly from the soundness invariant.

5.3.4 Proof-Based Verification

The ESR process is thus established to be sound (Thm. 1), complete (Thm. 2), composition-correct (Thm. 3), live (Thm. 4), and scalability-preserving (Thm. 5)—all machine-checked, with 98/98 proof obligations discharged in Rodin and zero invariant violations across the full reachable state space in ProB. This elevates the search-space reduction of GAMMA from an empirically observed optimization to a formally guaranteed property, ensuring that scalability is achieved without ever sacrificing the feasibility of QoS-aware service composition.

Theorem 1 (Soundness). *In every reachable state, $EligibleServices \subseteq eligible$; i.e. every retained service satisfies all context rules and the soft-utility threshold of eq. (5.1).*

Proof sketch. *FilterServices* assigns $Evaluated \cap eligible$, and the inclusion is preserved by all later refinements.

Theorem 2 (Completeness). *Upon filtering, $Evaluated \cap eligible \subseteq EligibleServices$ and $RemovedServices \cap eligible = \emptyset$; no eligible service is ever discarded.*

Together, Theorems 1–2 give the **exactness** $EligibleServices = Evaluated \cap eligible$: ESR retains **precisely** the compliant services, neither more nor fewer.

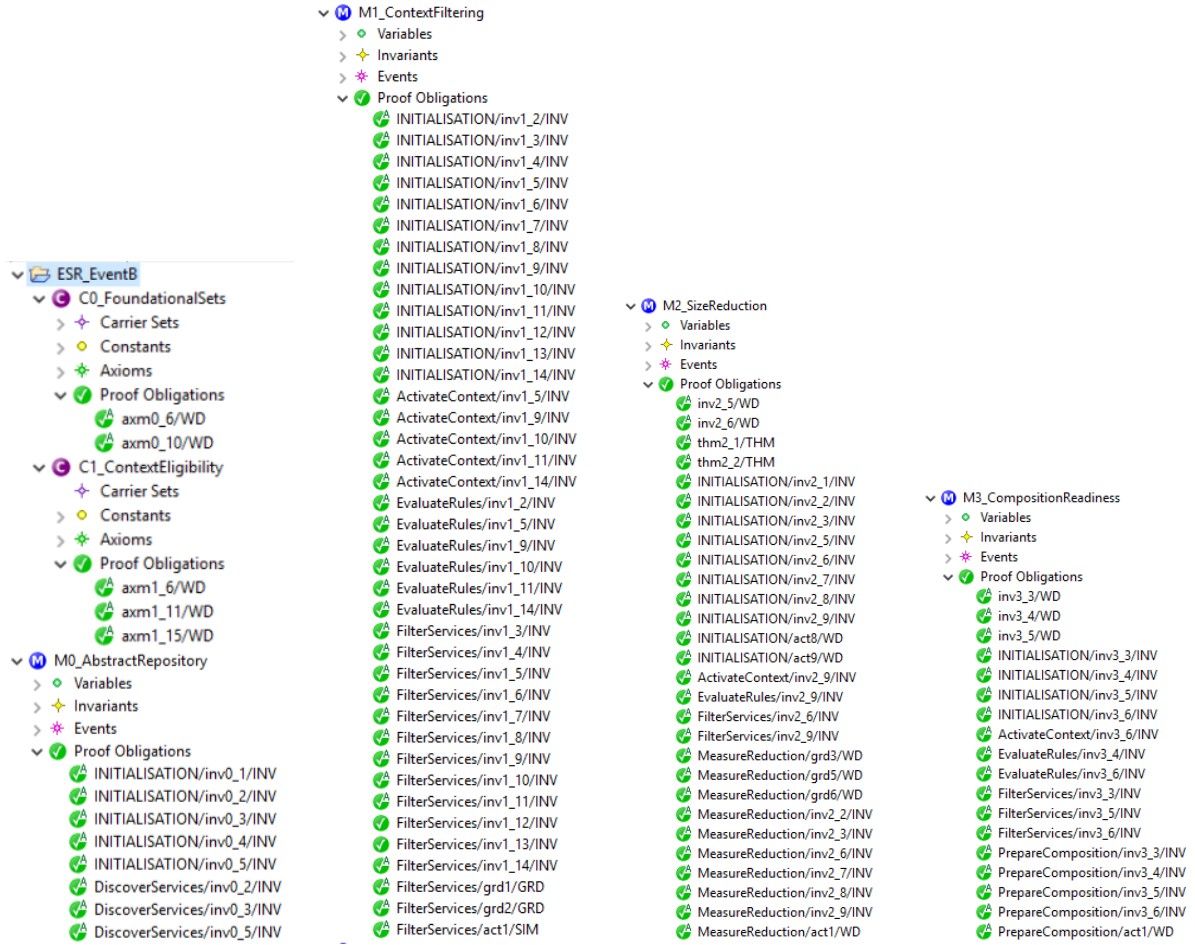


Figure 5.5: ESR Proof Obligations

Theorem 3 (Composition preservation). *For every composition c whose support lies in the eligible pool ($\text{compServices}(c) \subseteq \text{eligible}$ and $\subseteq \text{Evaluated}$), c remains feasible after reduction ($c \in \text{FeasibleAfter}$). Consequently, ESR never removes a service required by a feasible composition.*

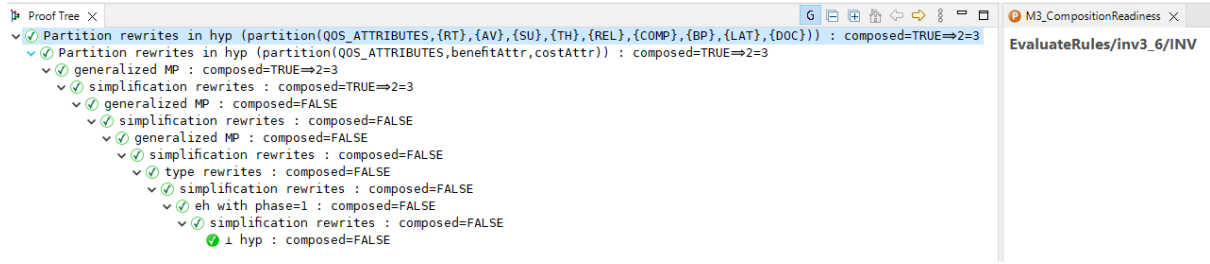
Proof sketch. By exactness, $\text{compServices}(c) \subseteq \text{Evaluated} \cap \text{eligible} = \text{EligibleServices}$, hence c satisfies the comprehension defining *FeasibleAfter*.

Theorem 4 (Liveness). *Whenever retained services exist, a feasible composition exists: for every $v \in \text{EligibleServices}$, its atomic composition $\text{atomComp}(v) \in \text{FeasibleAfter}$, hence $\text{EligibleServices} \neq \emptyset \Rightarrow \text{FeasibleAfter} \neq \emptyset$.*

Theorem 5 (Reduction bound and scalability). *The reduction metric is well defined and bounded, $0 \leq \text{ReductionRate} \leq 100$ with $\text{ReducedSize} \leq \text{OriginalSize}$, and the action space exposed to the downstream selection and composition stages shrinks monotonically, $|R| \geq |V_C| \geq |V_S|$.*

Theorem 5 provides the verification-backed justification for the complexity argument of GAMMA: because the $\mathcal{O}(N^2d)$ attention cost and the GAT encoder operate on *EligibleServices* rather than on R , the proven monotone shrink directly bounds their per-step cost.

The development generates proof obligations in Rodin, of which are discharged automatically (lasso + PP/ML/SMT) (Figure 5.6a and interactively, the reduction-rate bound



(a) Proof Tree

Element Name	Total	Auto	Man.	Rev.	Und.
ESR_EventB	98	96	2	0	0
C0_FoundationalSets	2	2	0	0	0
C1_ContextEligibility	3	3	0	0	0
M0_AbstractRepository	8	8	0	0	0
M1_ContextFiltering	39	37	2	0	0
M2_SizeReduction	28	28	0	0	0
M3_CompositionReadiness	18	18	0	0	0

(b) Proof Obligations Statistics

Figure 5.6: Proof Obligations in Rodin.

and the composition-liveness witness—so that all obligations are discharged (Figure 5.5). The verified guarantees are summarized by the following results in Figure 5.6b, each a machine-checked invariant or theorem of the model.

5.3.5 Model Checking with ProB

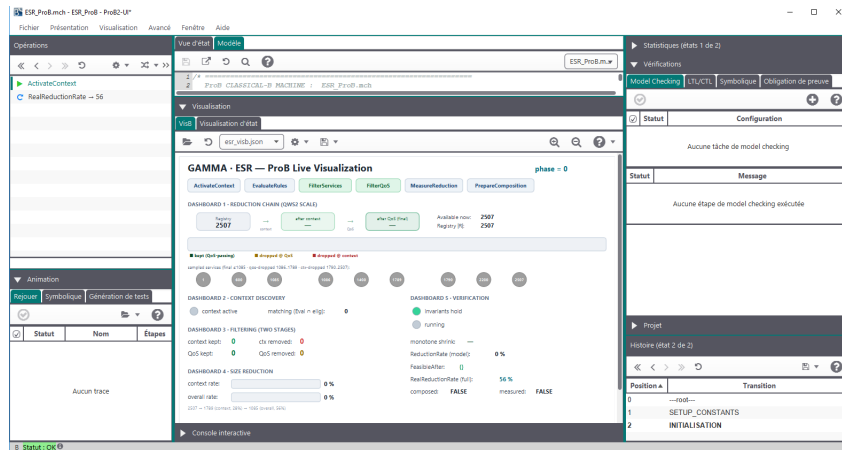


Figure 5.7: ProB Visualization

To complement the deductive proofs with exhaustive state-space exploration, the model is checked in ProB. The model checking reports **zero invariant violations** and full operation coverage, confirming the deductive results independently. The ProB animation executes the ESR pipeline as a single deterministic trace: each operation is guarded by the **phase** counter, so it becomes enabled only once the previous stage has

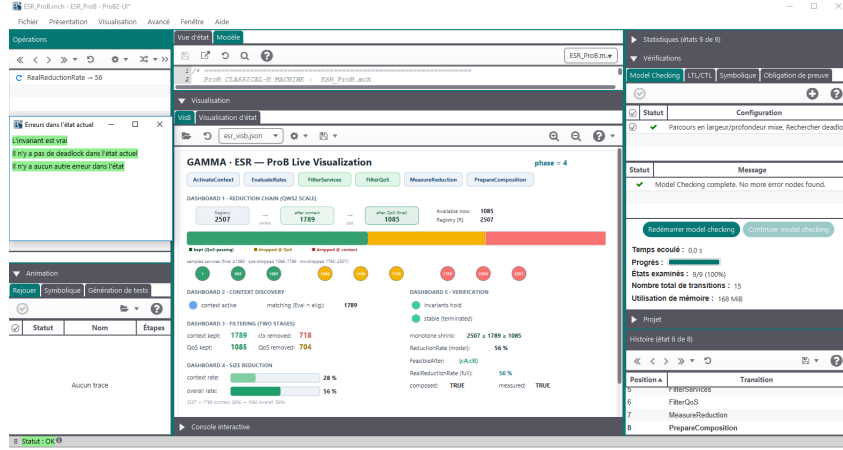


Figure 5.10: ProB Visualization

services whose attributes fail the contextual thresholds. This step establishes *soundness* (every retained service is context-valid) and *completeness* (no eligible service is discarded).

- **Step 4 — FilterQoS (QoS, 1789 → 1085).** *phase* moves 3 → 4 and the QoS partition is committed within the context-verified set: $QoSServices := EligibleServices \cap qosEligible$ (1085) and $QoSRemovedServices$ (704) Figure 5.8. QoS-aware filtering further narrows the pool to services that also meet the QoS requirements—the second reduction—and yields the nested chain $QoSServices \subseteq EligibleServices \subseteq Registry$.
- **Step 5 — MeasureReduction.** No sets change; the metrics are recorded and *measured* := TRUE: $ContextReducedSize = 1789$, $ReducedSize = 1085$, $ContextRate = 28\%$ and $ReductionRate = 56\%$ Figure 5.9. This quantifies the shrink at each stage and overall, and confirms the monotone-shrink invariant $ReducedSize \leq ContextReducedSize \leq OriginalSize$ ($1085 \leq 1789 \leq 2507$).
- **Step 6 — PrepareComposition.** *composed* := TRUE and $FeasibleAfter := \{c \mid compServices(c) \subseteq QoSServices\} = \{c_A, c_B\}$. The system checks which compositions remain feasible on the final pool; c_C is excluded because it requires a context-removed service. This demonstrates *composition preservation* and *liveness*: a feasible composition survives the 56% reduction.
- **Terminal state.** Once composed, no pipeline operation is enabled (*phase* = 4, *measured* and *composed* both TRUE); only the read-only **RealReductionRate** query remains firable, as a self-loop, so model checking reports **no deadlock** Figure 5.10. The pipeline has reached its designed fixed point: the environment is reduced $2507 \rightarrow 1789 \rightarrow 1085$, every invariant held at every step, and a feasible composition survived.

5.3.6 Discussion and Conclusion

The ESR process is thus established to be sound, complete, composition-correct, live, and scalability-preserving. All machine-checked, with all proof obligations discharged in Rodin and zero invariant violations across the full reachable state space in ProB. This

elevates the search-space reduction of GAMMA from an empirically observed optimization to a formally guaranteed property, ensuring that scalability is achieved without ever sacrificing the feasibility of QoS-aware service composition.

5.4 MDP Reduction Verification (HiMAD)

While the experimental evaluation in the next Section demonstrates the empirical performance of Hi-MAD, the structural correctness of its two-stage filtering mechanism warrants an independent, machine-checkable guarantee. To this end, we formalize the MDP reduction pipeline presented in (Definitions 1–3) in Event-B and mechanically verify its correctness using the Rodin platform. The verification is complementary to the deep reinforcement learning components: it does not address learning dynamics, convergence, or reward optimality, all of which remain empirical concerns. Rather, it establishes that the agent-filtering pipeline over which the DRL agents operate is structurally sound — every retained agent provably satisfies its admission criteria, the agent set decreases monotonically across filter stages, and action execution is restricted to validated agents.

5.4.1 Verification Architecture

We adopt the Event-B refinement methodology because the three-stage reduction $E \sqsupseteq E_C \sqsupseteq E_Q$ naturally corresponds to a refinement chain of formal machines. Event-B’s centralized treatment of state, events, and invariants enables compositional reasoning: each refinement step introduces new state and events while preserving the structural invariants of its abstraction, and the proof obligations generated at each step are mechanically discharged by the Rodin prover suite.

The verification scope covers the discrete, structural aspects of Hi-MAD: the agent universe, the threshold-based filtering predicates, the subset chain $N_Q \subseteq N_C \subseteq N$ underlying the cardinality reductions, and the safety of action execution under the filtered MDP. Continuous DRL operations — policy gradients, GAE estimation, replay-buffer dynamics — fall outside this scope, as classical refinement reasoning is unsuited to stochastic learning processes. The formal model is structured as one Event-B context (**C0_Agents**) shared across three machines (**M0_FullMDP**, **M1_ContextMDP**, **M2_QoSMDP**) connected by a refinement chain. Figure 5.11 illustrates the architecture and its correspondence with Definitions 1, 2 and 3.

Context C0_Agents The context fixes the universe of discourse. It declares three carrier sets, *AGENT* (all candidate IoT-service agents), *STATE* (global environment states), and *ACTION* (discrete action labels corresponding to Σ in Definition 1). It introduces five constants: the population $N \subseteq \text{AGENT}$, the scoring functions $\text{context_score} : \text{AGENT} \times \text{STATE} \rightarrow \mathbb{N}$ (formalising Eq. 6) and $\text{qos_score} : \text{AGENT} \rightarrow \mathbb{N}$ (formalising Eq. 7), and the thresholds $\theta_C, \theta_Q \in \mathbb{N}$. Real-valued scores in the original specification are encoded as scaled naturals (multiplication by a fixed constant followed by rounding), which is sound for threshold comparisons because the predicates of interest involve only $\geq$ relations. Seven axioms establish typing, non-emptiness ($N \neq \emptyset$), totality of the scoring functions, and finiteness of N (required for cardinality reasoning).

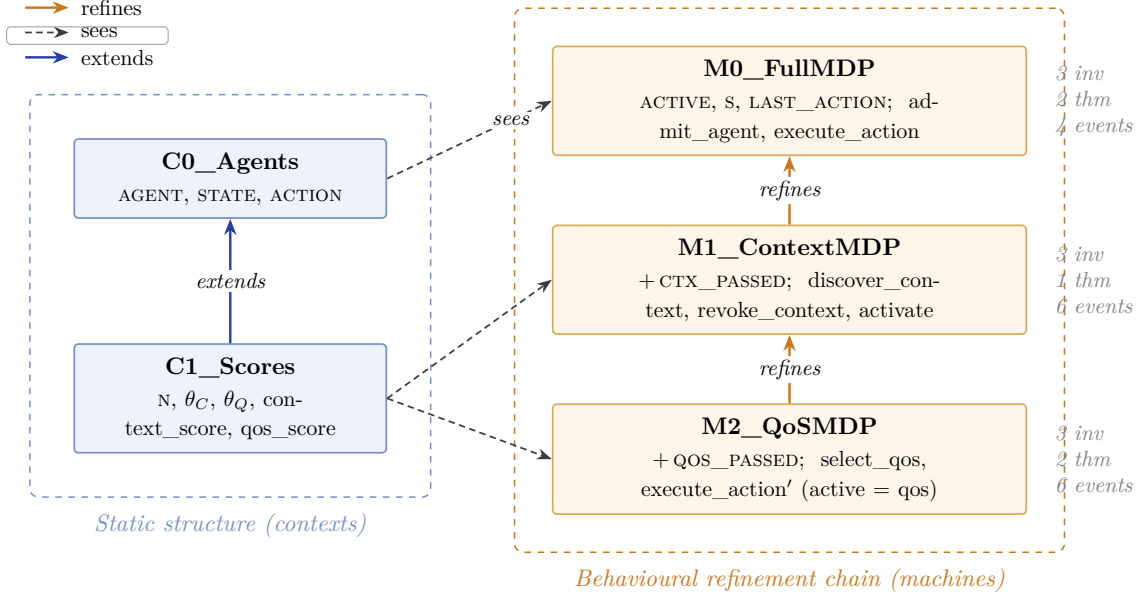


Figure 5.11: Event-B Architecture of MDP Reduction

Abstract Machine M0_FullMDP The abstract machine formalizes Definition 1. It declares three variables: the active agent set $active \subseteq N$, the global state $s \in STATE$, and the most recent joint action $last_action \in active \rightarrow ACTION$. Three asserted invariants establish typing and the well-formedness of $last_action$. Four events specify the abstract operational semantics: *INITIALISATION* (non-deterministically selects an initial active set), *admit_agent* (admits a new agent to $active$, refined in subsequent machines), *execute_action* (records an action by an active agent), and *transition* (updates the global state).

First Refinement M1_ContextMDP The first refinement formalizes Definition 2 by introducing context-aware filtering. It adds the variable $ctx_passed \subseteq N$, representing agents that have passed context discovery. Three new asserted invariants capture the central correctness properties at this level:

$$inv1 : ctx_passed \subseteq N \quad (5.4)$$

$$inv2 : active \subseteq ctx_passed \quad (5.5)$$

$$inv3 : \forall a \cdot a \in ctx_passed \implies context_score(a \mapsto s) \geq \theta_C \quad (5.6)$$

Three new events extend the abstract semantics: *discover_context* (admits an agent to ctx_passed if the context threshold is met), *revoke_context* (cascades removal when the threshold is no longer met), and *activate_agent* (refines *admit_agent* with the strengthened precondition $a \in ctx_passed$). The *transition* event is refined with a precondition forcing context-threshold preservation across state changes, making context drift explicit in the formal semantics.

Second Refinement M2_QoSMDP The second refinement formalizes Definition 3 by introducing QoS-based filtering on top of context filtering. It adds $qos_passed \subseteq ctx_passed$ and the gluing invariant $active = qos_passed$, binding the operational active set to the doubly-filtered set. Five asserted invariants and two derived theorems are introduced at

```

CONTEXT C0_Agents

SETS
AGENT, STATE, ACTION

CONSTANTS
N, context_score, qos_score, theta_C, theta_Q

AXIOMS

ax1 :  $N \subseteq \text{AGENT}$ 
ax2 :  $N \neq \emptyset$ 
ax3 :  $\text{context\_score} \in \text{AGENT} \times \text{STATE} \rightarrow \mathbb{N}$ 
ax4 :  $\text{qos\_score} \in \text{AGENT} \rightarrow \mathbb{N}$ 
ax5 :  $\text{theta\_C} \in \mathbb{N}$ 
ax6 :  $\text{theta\_Q} \in \mathbb{N}$ 
ax7 :  $\text{finite}(N)$ 

END

```

```

MACHINE M0_FullMDP
SEES C0_Agents

VARIABLES
active, s, last_action

INVARIANTS

inv1 :  $\text{active} \subseteq N$ 
inv2 :  $s \in \text{STATE}$ 
inv3 :  $\text{last\_action} \in \text{active} \mapsto \text{ACTION}$ 

EVENTS

INITIALISATION
execute_action
transition
admit_agent
END

```

this level. The new event *select_qos* refines *activate_agent*, requiring $\text{qos_score}(a) \geq \theta_Q$ as an additional precondition. The *execute_action* event is further refined to require both threshold conditions, formalizing the safety of action execution under the doubly-filtered MDP.

5.4.2 Verification Process

The verification establishes nine correctness properties across the two refinement machines, partitioned into asserted invariants (verified per-event) and derived theorems (verified once from existing invariants). The distinction reflects the role of each property: asserted invariants must be preserved by every event of the machine, providing strong operational guarantees; derived theorems are consequences of existing invariants, documenting useful summaries and corollaries.

Asserted Invariants Six asserted invariants capture properties the system must maintain across every state transition.

(I1) *Context filter well-formedness* — *M1_ContextMDP*, *inv1*.

$$\text{ctx_passed} \subseteq N$$

Every context-passed agent originates from the declared population.

(I2) *Active-context consistency* — *M1_ContextMDP*, *inv2*.

$$\text{active} \subseteq \text{ctx_passed}$$

Operationally active agents must have passed context discovery.

(I3) *Context threshold preservation* — *M1_ContextMDP*, *inv3*.

$$\forall a \cdot a \in \text{ctx_passed} \implies \text{context_score}(a \mapsto s) \geq \theta_C$$

The context filter is sound: every retained agent satisfies the context threshold under the current state.

(I4) *QoS threshold preservation* — *M2_QoSMDP*, *inv_qos_threshold*.

$$\forall a \cdot a \in \text{qos_passed} \implies \text{qos_score}(a) \geq \theta_Q$$

The QoS filter is sound: every QoS-passed agent satisfies the QoS threshold.

(I5) *Cross-filter consistency* — $M2_QoSMDP, inv_ctx_threshold$.

$$\forall a \cdot a \in qos_passed \implies context_score(a \mapsto s) \geq \theta_C$$

QoS-filtered agents simultaneously satisfy the context threshold, formalising the compositionality of the two-stage filter.

(I6) *Action-execution safety* — $M2_QoSMDP, inv_safe_execution$.

$$\forall a \cdot a \in \text{dom}(last_action) \implies qos_score(a) \geq \theta_Q \wedge context_score(a \mapsto s) \geq \theta_C$$

Any agent that has executed an action provably satisfies both filtering thresholds, establishing that the operational pipeline never permits action execution by an unvalidated agent.

Derived Theorems Three derived theorems are proven once from the asserted invariants and document key consequences of the filtering structure.

Theorem 1. *Subset chain (Definition 3)* — $M2_QoSMDP, thm_monotone$.

$$qos_passed \subseteq ctx_passed \wedge ctx_passed \subseteq N$$

The hierarchical inclusion $N_Q \subseteq N_C \subseteq N$ from Definition 3 is mechanically established.

Theorem 2. *Cardinality reduction* — $M2_QoSMDP, thm_card_reduction$.

$$\text{card}(qos_passed) \leq \text{card}(ctx_passed) \leq \text{card}(N)$$

Theorem 3. *Context filter soundness (summary)* — $M1_ContextMDP, thm_ctx_sound$.

$$\forall a \cdot a \in ctx_passed \implies a \in N \wedge context_score(a \mapsto s) \geq \theta_C$$

Combines (I1) and (I3) into a single soundness statement, used as a citable summary lemma.

Proof Obligation Discharge

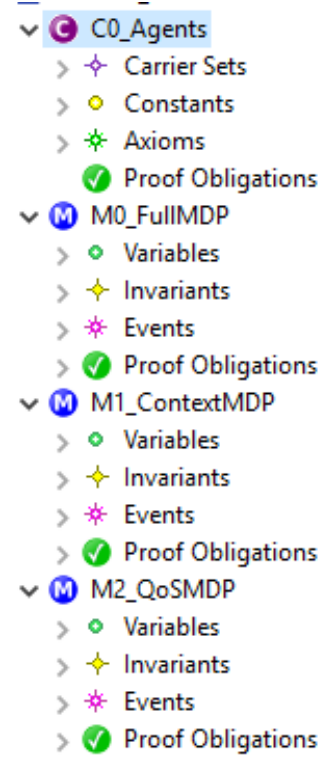
The Rodin platform automatically generates proof obligations from the model. Each invariant produces one preservation obligation per event that modifies a referenced variable; each derived theorem produces a single consistency obligation; each refinement link produces simulation, guard-strengthening, and witness obligations. Figure 5.12a summarizes the distribution and discharge status across the verification.

The proof obligations span four categories: (i) **well-definedness** obligations (/WD), automatically generated for every function application and discharged by the predicate prover given the totality axioms of *context_score* and *qos_score*; (ii) **invariant preservation** obligations (/INV), one per event-invariant pair; (iii) **refinement** obligations (/SIM, /GRD, /EQL), generated at refining events to verify simulation of the abstract event, guard strengthening, and gluing-invariant preservation; and (iv) **theorem** obligations (/THM), one per derived theorem.

The clean structure of the refinement chain — each level introducing a single filtering stage with strengthened guards — allows Rodin’s auto-tactic to discharge all generated obligations without manual intervention. This automation reflects the structural simplicity of the verification rather than a weakness of scope: every invariant is preserved by construction through the strengthened guards of the refining events, and every derived theorem follows directly from the asserted invariants via standard set-theoretic and arithmetic reasoning.

Element Name	Total	Auto	Man.	Rev.	Und.
HiMAD Verif...	54	54	0	0	0
C0_Agents	0	0	0	0	0
M0_FullMDP	7	7	0	0	0
M1_Context...	19	19	0	0	0
M2_QoSMDP	28	28	0	0	0

(a) Proof Obligations Statistics



(b) Proof Obligations Discharged

Figure 5.12: Proof Obligations in Rodin.

ProB Model Checking

The Event-B refinement proof was complemented by explicit-state model checking in ProB. Exhaustive exploration of the reachable state space for a representative six-agent instance (52 states, 187 transitions, fully explored) reported no invariant violations and no deadlocks, confirming that the two-stage filtering pipeline preserves all structural and safety invariants under every admissible operation sequence. Figure 5.13 shows the corresponding VisB visualization, in which the population contracts from six agents to three after context filtering and to two after QoS selection, with action execution confined to the doubly-filtered (active) set.

5.4.3 Discussion and Conclusion

The verification demonstrates that Hi-MAD’s MDP reduction pipeline is structurally sound: the agent set decreases monotonically across filter stages, every retained agent satisfies its admission criteria, and action execution is provably safe with respect to both filtering thresholds. The reduction is formally guaranteed (Theorem 2.), and the safety property (I6) ensures that the DRL agents operate exclusively over a validated subset of the agent universe.

Three caveats define the boundary of these guarantees. First, the verification is purely structural: it makes no claim about the convergence, optimality, or reward properties of the DRL policies, which remain empirical. Second, the encoding of real-valued scores as scaled naturals is sound for the predicates considered (involving only threshold comparisons), but does not extend to verifications requiring continuous arithmetic. Third, the

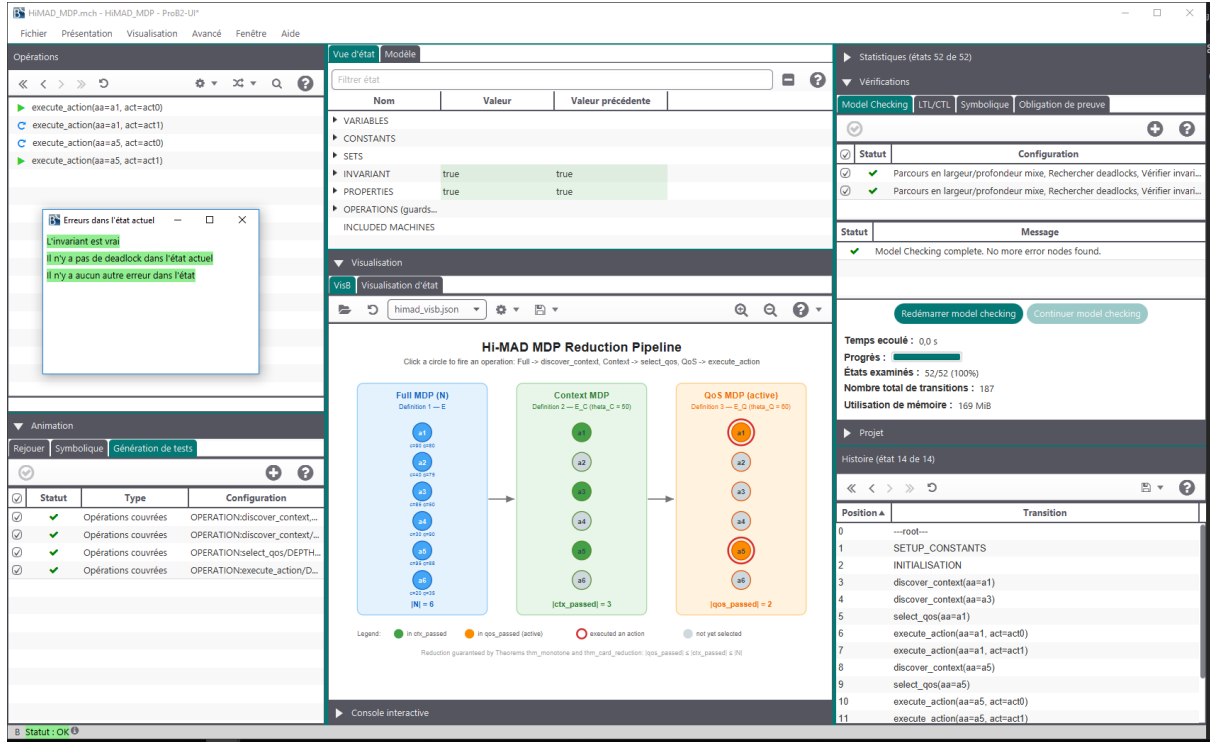


Figure 5.13: ProB Visualization

model abstracts the temporal evolution of the system to discrete events; liveness properties (e.g., guaranteed eventual admission of qualifying agents) would require an extension to Event-B with explicit fairness constraints, which is left for future work.

Within these boundaries, the verification provides a complementary layer of assurance to the empirical evaluation in following Section: the DRL adaptation operates over a search space whose structural correctness has been mechanically established, addressing reviewer concerns regarding the soundness of the two-stage filtering mechanism.

5.5 Experimental Validation of HiMAD

This section presents the experimental evaluation of the proposed **Hi-MAD** framework². First, we describe the simulation setup and the parameters configuration used to emulate realistic smart-building conditions. Then, the subsequent subsections analyze **Hi-MAD**'s performance across multiple criteria, including framework scalability, coordination efficiency, and energy-resilience optimization. Comparative results are then reported to highlight the advantages of **Hi-MAD** and the proposed multi-layer architecture. Finally, a discussion synthesizes key findings and insights, emphasizing how **Hi-MAD** achieves balanced trade-offs among stability, resilience, and energy efficiency in dynamic IoT environments.

²The implementation, the simulation environment, and the results are available on [Github link](#) and updated according the progress.

5.5.1 Simulation Environment and Parameters

To evaluate the effectiveness of the **Hi-MAD** framework, a realistic simulation environment was developed to emulate the operation of a smart office building equipped with interconnected IoT systems. The environment replicates dynamic real-world conditions — including occupancy variations, temperature changes, and fluctuating energy prices. Environment state is randomly generated to simulate unstable and dynamic condition, ensuring a comprehensive stress-test of the framework’s ability to maintain comfort, stability, and energy efficiency under unstable and unpredictable conditions.

Environment Setup

The experiments simulate a ten-zone intelligent office building equipped with heterogeneous IoT devices for climate control, lighting, and air quality management. Each zone hosts 50 autonomous IoT agents, a total of 500 distributed agents—coordinated through the hierarchical architecture. The building is assumed to span multiple floors and includes 10 rooms spread across various functional zones, such as offices, meeting rooms, corridors, restrooms, and open workspaces.

The agents are classified into six categories based on environmental control relevance. At the High-Level, a global coordinator based on Co-MATD3 manages long-term energy strategies and allocates budgets across zones, maintaining resilience and stability Equations 4.3 and 4.2. Its multi-dimensional reward function ensures global optimization of energy, comfort, and reliability objectives Equation 4.5. At the Mid-Level, specialized agents CD-MAPPO, QS-MAPPO, and Co-MADDPG—orchestrate zone-level operations after context discovery and QoS selection. They regulate comfort based on occupancy and environment, optimize subsystem energy use under constraints, and coordinate HVAC, lighting, and air-quality systems to sustain operational stability and occupant satisfaction. At the Low-Level, individual IoT devices such as thermostats, luminaries, and smart blinds are managed by DDQN agents. These agents adjust device parameters, adapt to occupancy changes, and minimize local energy use without compromising comfort—collectively contributing to system-wide efficiency. All simulations follow a centralized training with decentralized execution paradigm to ensure scalable training while preserving realistic agent autonomy during deployment, and was implemented in Python 3.10 using PyTorch 2.0, TensorFlow 2.11, Gymnasium, and PettingZoo, and executed on a workstation equipped with an Intel® Core™ i7-12700K CPU and 16 GB RAM.

Training Parameters

Each simulation runs for 5000 episodes, corresponding to roughly 6.5 hours wall-clock time on the test platform. To comprehensively evaluate the effectiveness of **Hi-MAD**, we employ a set of key performance indicators that assess thermal comfort maintenance, energy consumption, and the overall efficiency of agent-based decision-making. These metrics support an objective comparison between coordinated and non-coordinated approaches, highlighting the impact of **Hi-MAD** on system performance. Table 5.2 summarizes the key hyperparameters used across the four DDQN, CD-MAPPO, QS-MAPPO, Co-MADDPG, and Co-MATD3 algorithms, tuned to optimize learning performance, convergence rate, and policy robustness in the context of smart building coordination and control.

DDQN Algorithm		(CD/QS)-MAPPO Algorithm		Co-MADDPG Algorithm		Coo-MATD3 Algorithm	
Parameter	Value	Parameter	Value	Parameter	Value	Parameter	Value
Learning Rate (α)	0.001	GAE Lambda (λ)	0.95	Critic Learning Rate	0.0001	Actor Learning Rate	0.001
Discount Factor (γ)	0.99	Clipping Epsilon	0.2	Actor Learning Rate	0.0001	Discount Factor (γ)	0.99
Exploration Rate (ϵ)	1.0	Entropy Coefficient	0.01	Replay Buffer Size	1,000,000	Replay Buffer Size	10,000
Min Exploration (ϵ_{min})	0.01	Value Loss Coeff.	0.5	Noise Mean	0.0	Policy Noise	0.2
Exploration Decay	0.99	KL Divergence	0.01	Noise Deviation	0.2	Noise Clip	0.5
Replay Buffer Size	10,000	Batch Size	512	Batch Size	256	Batch Size	64
Batch Size	64	PPO Epochs	4	Soft Update (τ)	0.01	Critic Learning Rate	0.002
Target Update Freq.	1,000	Reward Scaling	True	Gradient Clipping	0.6	Target Network Update	0.005
Max Episodes	10,000	Update Frequency	2,048	Update Frequency	500	Policy Delay	2
Max Steps/Episode	500	Gradient Clipping	0.5	Policy Entropy Coeff.	0.02	Variance Threshold	0.1
Gradient Clipping	10	Reward Normalization	True	Global State Size	Full state	Resilience Clip Range	$[-1, 1]$

Table 5.2: Summary of hyperparameters for DRL algorithms used in training.

1. *Learning Rate (α) and Optimizer*: DDQN and Coo-MATD3 actors use $\alpha = 0.001$ for balanced convergence; Co-MADDPG and Coo-MATD3 critics use smaller α values (0.0001–0.002) to stabilize Q-value.
2. *Discount Factor (γ)*: A high discount factor $\gamma = 0.99$ ensures long-term reward optimization suited for building control dynamics.
3. *Exploration Parameters*: DDQN uses ϵ -greedy with decay from 1.0 to 0.01; Co-MADDPG and Coo-MATD3 rely on Gaussian noise ($\sigma = 0.2$) with clipped noise for Coo-MATD3 (limit 0.5).
4. *Replay Buffers and Batch Sizes*: CD- and QS-MAPPO use large buffers (1,000,000); DDQN and Coo-MATD3 use smaller buffers (10,000) with batch size 64 for responsive updates.
5. *Target Updates and Polyak Averaging (τ)*: Coo-MATD3 applies Polyak averaging with $\tau = 0.005$; Co-MADDPG performs soft updates with $\tau = 0.01$.
6. *Stability Enhancements*: Coo-MATD3 uses policy delay (actor every 2 critic steps) and clipped noise (0.5). Gradient clipping is applied in MAPPO (0.5–0.6) and DDQN (10).
7. *Multi-Agent Parameters*: CD- and QS-MAPPO use GAE with $\lambda = 0.95$, KL constraints (0.01), and entropy coefficients (0.01). Co-MADDPG includes full-state observability and entropy coefficient 0.02.
8. *Coo-MATD3 Resilience Mechanisms*: A variance-based filter triggers smoothing when variance > 0.1 with actions clipped in $[-1, 1]$ to ensure robust coordination.

Evaluation Metrics

To assess the effectiveness of the **Hi-MAD** framework, several evaluation metrics are defined to quantify system performance in terms of energy efficiency, stability, resilience, and user comfort.

- *Energy consumption* evaluates how well the hierarchical policy minimizes redundant actuator activity. Lower values indicate effective collaboration between high-level coordination and low-level execution: $E(t) = \sum_{i=1}^N P_i(t) \Delta t$.

- *QoS satisfaction* captures the functional outcome of mid-level decision modules (CD-MAPPO and QS-MAPPO). A high ratio means that discovered and composed services successfully meet user requirements under dynamic workloads: $Q(t) = \frac{N_{\text{tasks, completed}}(t)}{N_{\text{tasks, requested}}(t)} \times 100$.
- *Thermal-comfort deviation* links physical-world comfort with algorithmic efficiency. Maintaining comfort with minimal energy reflects the optimal trade-offs achieved by Co-MADDPG during service composition: $C(t) = |T_{\text{zone}}(t) - Opt|$.
- *Stability index* tied to the dual-critic and delayed-update mechanisms of Co-MATD3. Small $S(t)$ values imply smooth, non-oscillatory control actions and low switching losses, essential for long-term reliability: $S(t) = \sum_{i=1}^N |\Delta a_i(t)|^2$, $\Delta a_i(t) = a_i(t) - a_i(t-1)$.
- *Resilience score* quantifies recovery behavior under simulated node/link faults. Demonstrates how the variance-based consensus rule in Co-MATD3 enables self-healing coordination among mid- and low-level agents: $R(t) = \frac{N_{\text{tasks, recovered}}(t)}{N_{\text{tasks, faulty}}(t)} \times 100$.
- *Convergence rate* T_c : summarizes overall learning efficiency of the hierarchical system. hierarchical structured decomposition ensures faster convergence than flat DRL baselines because each layer learns sub-tasks independently yet cooperatively.

5.5.2 Hi-MAD Framework Evaluation

To validate the effectiveness of the proposed **Hi-MAD** framework, a comprehensive evaluation is performed to assess its performance across all hierarchical levels. The analysis focuses on how each layer meets its respective objectives using the previously defined evaluation metrics, providing a holistic understanding of the framework's scalability, coordination efficiency, energy optimization, and environmental stability.

Environmental Comfort Ensuring

The stability of environmental parameters is essential for maintaining occupant comfort and ensuring overall system efficiency in smart building environments. In the **Hi-MAD** framework, this stability is first supported by CD-MAPPO, which dynamically discovers and selects the most relevant services according to the current environmental context and occupancy conditions. The evaluation focuses on deviations of environmental parameters from their optimal comfort ranges, including temperature, light intensity, humidity, and air quality indicators (CO₂, VOC, PM_{2.5}). The adopted thresholds (see Table 5.3) follow indoor ergonomics and air-quality standards: temperature 20°C-26°C, lighting 300-500 lux, humidity 30-50%, CO₂ between 400-1000 ppm, VOC < 0.3 mg/m³, and PM_{2.5} < 12 µg/m³. These ranges ensure occupant comfort, health, and energy-efficient building operation.

Figure 5.14 illustrates the temporal evolution of key environmental parameters, including temperature, PM_{2.5} concentration, CO₂ levels, light intensity, humidity, and VOC concentration, in order to assess the system's ability to maintain indoor comfort and air quality within optimal bounds. In Figure 5.14a, temperature readings remain consistently within the range of 20°C to 25°C, indicating effective thermal regulation and compliance with standard indoor comfort guidelines. The PM_{2.5} concentration oscillates

Comfort Stability Metric	Optimal Range	Input Range
Temperature	20°C to 26°C	10°C to 40°C
Occupancy	Fewer than 50 occupants	0 to 150 occupants
Light Intensity	300 to 500 lux	0 to 1000 lux
Humidity	30% to 50%	10% to 90%
CO ₂ (Carbon Dioxide)	400 to 1000 ppm	300 to 2000 ppm
VOC (Volatile Organic Compounds)	$\leq 0.3 \text{ mg/m}^3$	0 to 2.0 mg/m^3
PM _{2.5} (Particulate Matter)	$\leq 12 \text{ }\mu\text{g/m}^3$	0 to $150 \text{ }\mu\text{g/m}^3$

Table 5.3: Environmental Performance Criteria for Comfort Stability Experiment.

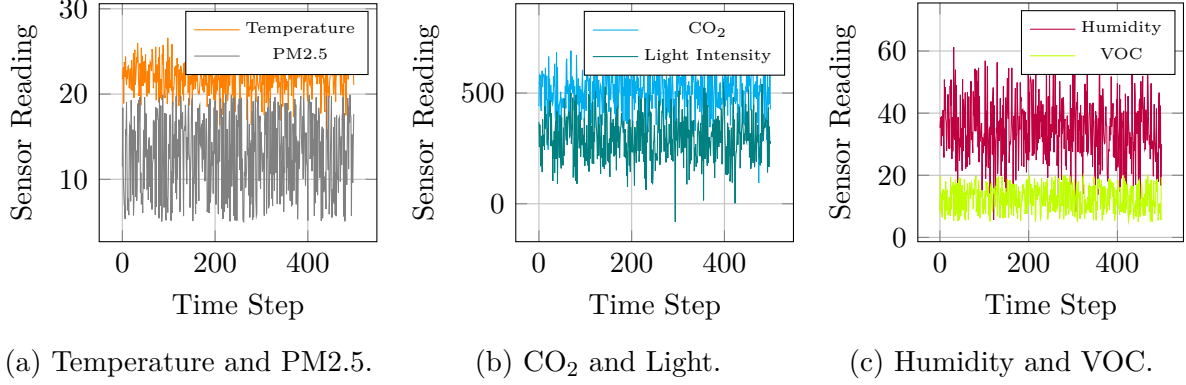


Figure 5.14: Environmental Sensor State

between $10 \text{ }\mu\text{g/m}^3$ and $20 \text{ }\mu\text{g/m}^3$, thereby confirming healthy air quality conditions. Figure Figure 5.14b shows CO₂ levels fluctuating between approximately 400 and 600 ppm, which is below the commonly accepted indoor comfort limit of 800 ppm. At the same time, light intensity stabilizes between 350 and 500 lux, ensuring adequate illumination for office-like environments while avoiding unnecessary energy consumption. In Figure 5.14c, relative humidity is maintained between 35% and 55%, aligning with recommended indoor comfort ranges. Additionally, VOC concentrations remain consistently low (below 20 units), demonstrating the effectiveness of the proposed ventilation and air purification mechanisms.

QoS Satisfaction

In this part we evaluate the QoS satisfaction achieved by the QS-MAPPO algorithm as illustrated on Figure 5.15a, the evolution of QoS satisfaction during the training. The results demonstrate that QoS satisfaction consistently remains within the acceptable range of 90%-98%, as indicated by the red dashed thresholds. Despite minor fluctuations inherent to dynamic service composition and environmental variations, the system rapidly converges to a stable operating regime where most episodes maintain QoS levels above 95%. This behavior highlights the robust adaptive capability of QS-MAPPO in maintaining service quality under changing workload and context conditions.

In Addition, Figure 5.28 presents a comparative analysis of QoS satisfaction across five different DRL models. The results indicate that QS-MAPPO outperforms all other models, achieving the highest QoS satisfaction of 95.3%, which is significantly higher than the second-best model, MAPPO, at 87.5%. This performance advantage underscores the effectiveness of QS-MAPPO in service selection and ensuring quality in dynamic IoT en-

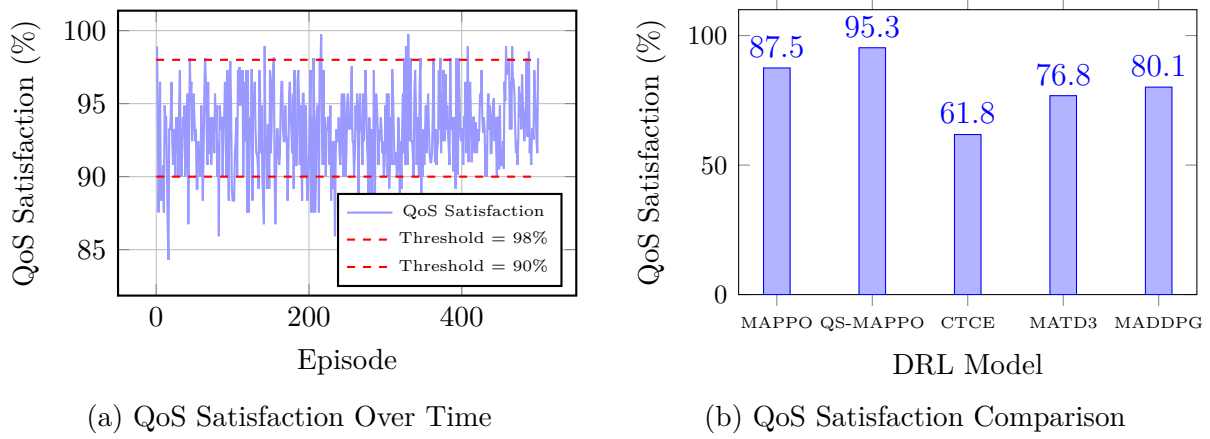


Figure 5.15: Quality of Service Evaluation.

vironments. The CTCE model, on the other hand, achieves the lowest QoS satisfaction at 61.8%, suggesting that its approach does not fully capture the complexity and coordination needs of multi-agent systems in real-world scenarios. The other models, MATD3 and MADDPG, also show respectable performance, with QoS satisfaction scores of 76.8% and 80.1%, respectively, indicating their ability to manage service quality but still falling short of QS-MAPPO's results.

As shown in Figure 5.16, the QS-MAPPO model achieves a consistently high accuracy in service selection across 50 episodes. On average, the number of correct selections per episode ranges between 100 and 115, while incorrect selections remain below 20, corresponding to an average accuracy of about 88-92%. During the first few episodes, incorrect decisions appear slightly higher (around 15-18%) due to exploration and policy adaptation, but they decrease progressively to below 10% after episode 10 as the policy stabilizes. This trend, further highlighted by the moving average overlay, demonstrates that QS-MAPPO efficiently learns to match services with contextual requirements and improves its decision-making precision over time. The consistently high number of correct selections also shows strong model generalization and adaptation.

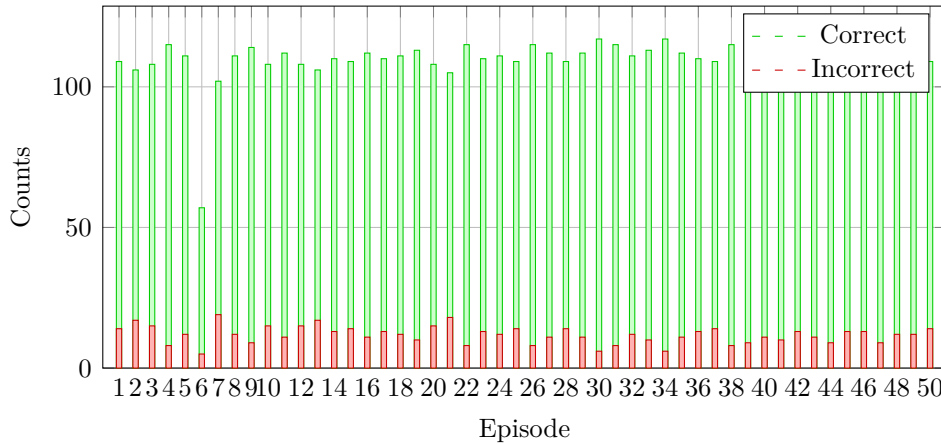


Figure 5.16: Correct vs Incorrect Selections per Episode.

Hi-MAD Energy Optimization

The goal of this evaluation is to measure how hierarchical agent filtering—through Context Discovery and QoS-based Selection—reduces MDP size, coordination complexity, and overall energy consumption. By pruning irrelevant or low-performing agents before coordination, the approach decreases computational overhead and enhances DRL policy learning efficiency. Additionally, the evaluation assesses energy minimization achieved through Co-MADDPG-based coordination with the shared reward mechanism used in DDQN, which together promote cooperative behavior and balanced energy usage across agents. The simulation involves 500 heterogeneous agents, each defined by a semantic context and a QoS score in the range $[0.2, 1.0]$.

Context and QoS-Based MDP Reduction. In the initial phase, we construct a baseline representation of the full MDP using all 500 agents. Each agent contributes a discrete action space and a high-dimensional observation space, Table 5.4 summarizes the MDP size reduction obtained through a hierarchical filtering pipeline applied to the initial 500-agent system. Each agent contributes one unit to both the state and action spaces, resulting in an initial state shape of $(500, 1)$ and action shape of $(500, 1)$. The filtering process includes two MAPPO-based phases: (1) Context Discovery, which retains only agents with relevant contextual rules (e.g., low temperature, high brightness, poor air quality); and (2) QoS-Based Selection, which further prunes agents with QoS scores below a threshold.

Metric	Full Environment	Context-Based MDP	QoS-Based MDP
Number of Agents	500	416	358
State Shape	$(500, 1)$	$(416, 1)$	$(358, 1)$
Action Shape	$(500, 1)$	$(416, 1)$	$(358, 1)$
Selected Agents Count	N/A	416	358
State Space Size	$2^{500} \approx 3.27 \times 10^{150}$	$2^{416} \approx 1.69 \times 10^{125}$	$2^{358} \approx 5.87 \times 10^{107}$
Action Space Size	2^{500}	2^{416}	2^{358}
Transition Space Size	$2^{1000} \approx 1.07 \times 10^{301}$	$2^{832} \approx 2.86 \times 10^{250}$	$2^{716} \approx 3.45 \times 10^{215}$
$\log_{10}(\text{Transition Space})$	≈ 301.03	≈ 250.46	≈ 215.54

Table 5.4: MDP Reduction via Context and QoS-Based Agents Filtering.

Table 5.5 presents a comparative analysis of energy consumption across different agent types under three configurations: Full Agent Deployment, Context-Based Filtering, and QoS-Based Filtering. The objective of this analysis is to assess the impact of intelligent filtering mechanisms on the overall energy efficiency of the multi-agent system in the smart building environment. In the full deployment configuration, all 500 agents operate simultaneously, leading to a total energy consumption of 352.25 kWh. This baseline scenario reflects an unoptimized setup, where all agents are active regardless of their contextual relevance or QoS constraints. By applying Context-Based filtering, the number of active agents is reduced to 416, yielding a significant energy reduction of 105.23 kWh, or **29.87%**. This demonstrates that contextual awareness—such as occupancy, time-of-day, or environmental thresholds—can effectively deactivate non-essential agents while still maintaining operational coverage. Further optimization is achieved through QoS-Based filtering, which considers service quality criteria (e.g., comfort thresholds, criticality of control actions). This results in only 385 agents being activated, with energy consumption dropping to 206.79 kWh, a total reduction of **41.29%** compared to the full deployment scenario.

Among all agent types, Temperature control agents consume the highest share of energy (2.5 kWh per agent), and their count drops significantly from 123 (Full) to 70

(QoS), reflecting the importance of intelligent agent activation to optimize power-hungry control functions. Similar downward trends are observed in other agent categories, such as Humidity, Light, and Air Quality, albeit with smaller absolute energy reductions due to their lower unit energy consumption.

Agent Type	Energy (kwh)	Agents (Full)	Total Energy (Full)	Agents (Context)	Total Energy (Context)	Agents (QoS)	Total Energy (QoS)
Temperature	2.5	123	312.5	85	212.5	70	175
Light	0.015	75	1.125	69	1.035	65	0.975
Humidity	0.5	75	37.5	65	32.5	60	30
Air Quality	0.005	225	1.125	197	0.98	193	0.082
Total	–	500	352.25	416	247.02	385	206.79
Energy-Reduction	–	N/A	N/A	16.8	29.87%	28,4	41.29%

Table 5.5: Energy consumption before and after Context and QoS-based filtering.

Energy Minimization Through Low-Agents The Co-MADDPG-based coordination agent learns a shared policy that harmonizes the decisions of all DDQN agents, reducing redundant or opposing actions and promoting smooth transitions toward optimal environmental states. This coordination leads to tighter adherence to comfort thresholds, improved air quality regulation, and more energy-efficient operation as shown in Figure 5.17 and proved in [19]. It illustrates the evolution of energy consumption over a series of training episodes within a multi-agent smart environment. The vertical axis represents energy cost (ranging approximately from 0 to 0.6), while the horizontal axis denotes the training episodes (from 0 to 2400).

- *Early episodes (0–200):* High variability and frequent energy spikes are observed, with values occasionally exceeding 0.5. This behavior reflects inefficient system control and a lack of coordination among agents during the initial learning phase.
- *Post-episode ~ 600 :* Energy usage stabilizes and consistently decreases, with most values falling in the 0.1–0.3 range and fewer pronounced peaks, indicating improved system behavior.

In contrast to the energy minimization observed in Figure 5.17, where the shared reward and the coordination promote stability and efficiency, Figure 5.18 instead illustrates the temporal evolution of energy consumption in a non-cooperative environment. The plot shows a dense distribution of energy values with frequent and irregular spikes, typically ranging between low and moderately high levels without any observable downward trend. The variability remains persistent across all time steps, indicating that the system does not learn or adapt toward improved efficiency. This pattern reflects where agents behave independently and fail to consider global system performance. Consequently, conflicting or redundant actions lead to sustained high energy consumption and prevent convergence toward an optimal regime.

System Resilience & Stability.

To improve stability and resilience in the hierarchical smart-building controller, a resilience-aware supervisory mechanism is deployed at the high-level Co-MATD3 agent. This mechanism, implemented via the `ensure_resilience` function, monitors the statistical

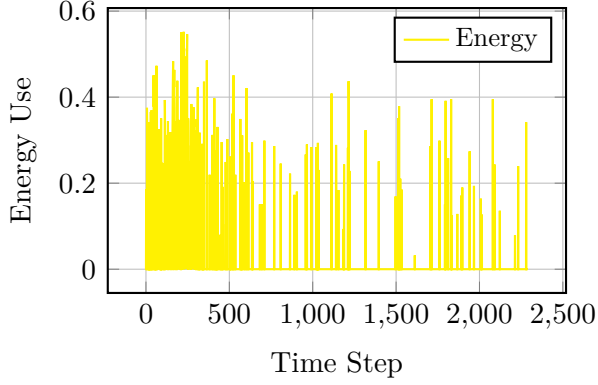


Figure 5.17: Energy Use Minimization.

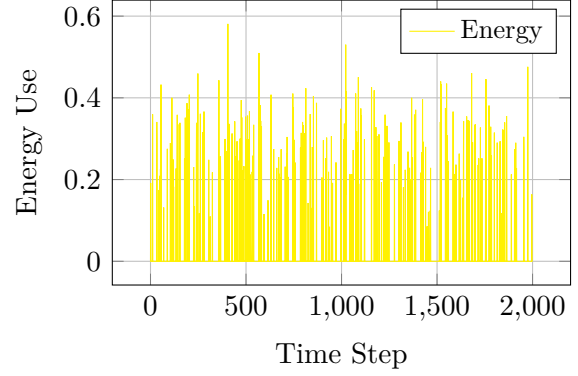


Figure 5.18: Energy Use Baseline

dispersion of mid-level agents' actions and adaptively enforces coordination when instability is detected defined in Equation 4.3, directly applying the resilience score $R(t)$ by driving faulty task recovery under simulated node and link faults.

Figure 5.19 shows the evolution of variance among mid-level agents' policies as an indicator of system resilience. High variance denotes uncoordinated actions, while low variance indicates synchronized, cooperative behavior aligned with high-level objectives. During the early phase (0-100 time steps), variance is high and oscillatory as agents explore their action spaces. Between 100-200 time steps, variance drops sharply as the Coo-MATD3 agent enforces coordination by reallocating tasks and correcting inefficient behaviors. After 200 time steps, variance stabilizes near zero, confirming consistent, energy-efficient cooperation and demonstrating **Hi-MAD**'s stable and resilient coordination across large-scale IoT environments.

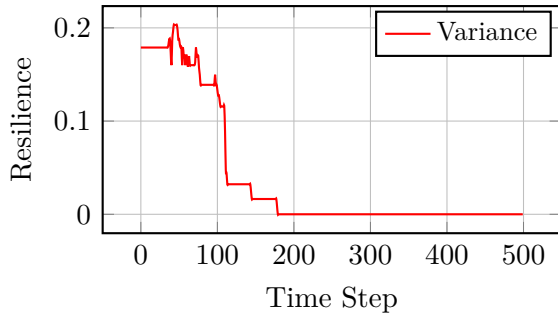


Figure 5.19: Mid-Agent Stability

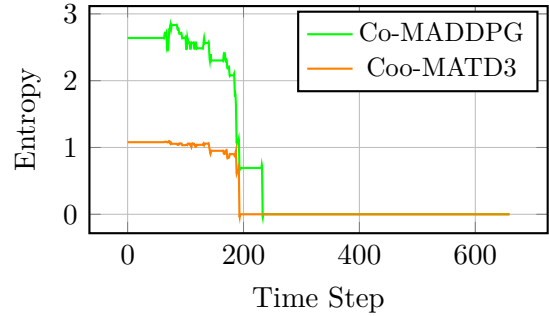


Figure 5.20: Policy Stability over Time

To evaluate the temporal stability of the high-level *Coo-MATD3* policy, we compute the Shannon entropy of its action distribution over time. Policy stability reflects the consistency and determinism of agent decisions after convergence defined in Equation 4.2. Entropy values are tracked throughout training, a decreasing trend indicates reduced uncertainty and increasing policy determinism, while convergence is characterized by entropy stabilizing at a low value, reflecting confident and consistent action selection.

Figure 5.20 shows the entropy evolution of the high-level *Coo-MATD3* and mid-level *Co-MADDPG* agents, reflecting the progression from exploration to policy convergence. Initially, *Co-MADDPG* maintains high entropy (~ 2.6), indicating extensive exploration of coordination strategies needed for adaptive control. By Time Step 200, entropy drops sharply to near 0.1, demonstrating rapid convergence toward a stable policy capable of

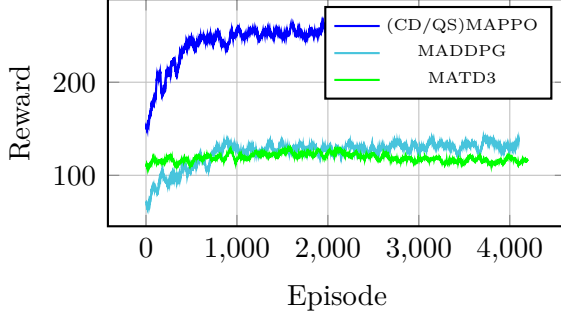


Figure 5.21: (CD/QS)-MAPPO Reward vs Other DRL Models

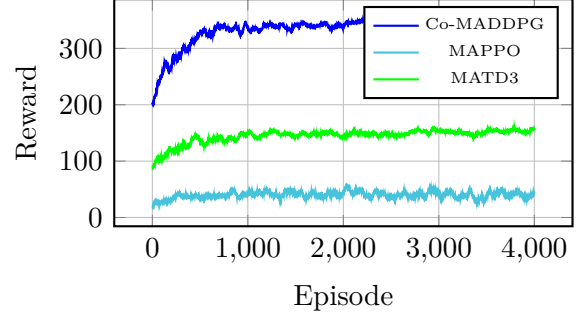


Figure 5.22: Co-MADDPG Reward vs Other DRL Models

maintaining coherent orchestration under dynamic conditions. Conversely, Co-MATD3 starts with moderate entropy (~ 1.0) and stabilizes around 0.2 within the same period, showing faster but narrower convergence focused on global optimization. The consistently lower entropy of Co-MADDPG and Co-MATD3 confirms superior long-term stability. Together, both layers achieve synchronized policy stabilization, ensuring coherent, energy-efficient, and resilient system performance across the hierarchical framework.

5.5.3 DRL Models Performance Evaluation

To evaluate the practical applicability of each DRL algorithm in the multi-agent smart building environment, we analyzed convergence time and reward evolution across training and testing phases. This assessment highlights how efficiently each algorithm stabilizes its policy and optimizes long-term rewards, revealing their comparative strengths and real-world suitability for dynamic and distributed IoT control.

Reward Evolution

Figure 5.21 shows that (CD/QS)-MAPPO achieves rapid convergence within approximately 600–800 episodes, reaching a steady-state cumulative reward of around 250, while MADDPG and MATD3 plateau at lower values of roughly 130 and 120, respectively, after nearly 1,500–2,000 episodes. This demonstrates that (CD/QS)-MAPPO learns nearly $2\times$ faster and achieves about 90% higher reward performance, confirming its superior policy optimization and stability under complex coordination tasks. In the other side, in Figure 5.22 Co-MADDPG achieves the highest peak reward of about 320 within 700–900 episodes, outperforming MAPPO (~ 170 reward) and MATD3 (~ 150 reward). Co-MADDPG thus provides more performance gain and converges nearly 50% faster than the baselines. Finally, Figure 5.23 shows that Co-MATD3 rapidly converges within approximately 600–800 episodes, achieving a peak cumulative reward of around 450, while MADDPG stabilizes near 230 and MAPPO around 150 after more than 2000 episodes. This represents roughly a 95% improvement in reward performance over MADDPG and a 200% gain over MAPPO. Moreover, Co-MATD3 exhibits smoother convergence with minimal oscillations.

Figure 5.24 shows that DDQN agents achieve strong and stable performance during testing, consistently obtaining positive rewards in about 90 of 105 episodes. Early episodes (e.g., 8, 16, 30) display occasional low or negative rewards (down to -3) due to exploration, but from episodes 35–80, rewards stabilize between 5 and 15, indicating

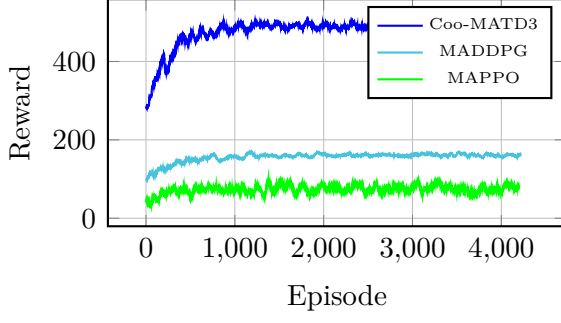


Figure 5.23: Co-MATD3 Evaluation vs Other DRL Models.

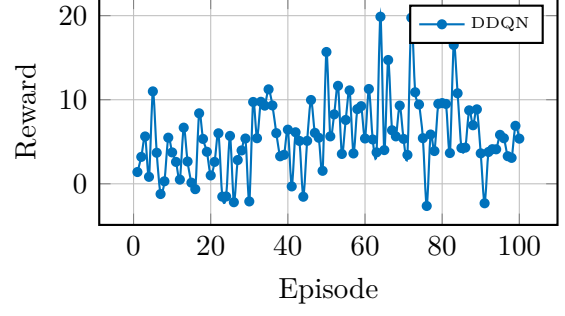


Figure 5.24: DDQN Reward Performance over Time.

policy convergence. Peaks around episodes 66 and 73 reach nearly 21, demonstrating optimal decision-making under favorable conditions. Under the coordination of the mid-level Co-MADDPG agent, the low layer agent achieves an average reward of 7.5, confirming robust learning, effective task distribution, and high reliability for real-time smart building control.

Convergence Time

The comparative analysis highlights distinct convergence behaviors across algorithms due to their learning architectures and coordination mechanisms. (CD/QS)-MAPPO achieved the fastest convergence at 600 episodes under the MAPPO setup, owing to its CTDE paradigm and efficient policy-gradient optimization, which promote rapid stabilization. Co-MADDPG followed closely, converging at 800 episodes under the MADDPG setup, benefiting from cooperative actor-critic updates that accelerate learning in dynamic multi-agent environments. Co-MATD3 reached convergence at 700 episodes within the MATD3 framework, balancing stability and exploration through twin critics and delayed updates that suppress policy oscillations.

5.5.4 Statistical Validation and Discussion

Overall, the obtained results confirm that **Hi-MAD** effectively maintains environmental comfort by keeping all critical parameters—thermal conditions, air quality, lighting, and humidity within optimal comfort ranges throughout system operation. This performance highlights the efficiency of coordinated multi-agent control in ensuring stable, healthy, and comfortable indoor environments.

The results also confirm that **Hi-MAD** achieves high selection precision with minimal error propagation, demonstrating its robustness and accuracy in dynamic IoT service orchestration while ensuring QoS, and that the filtering using CD-MAPPO and QS-MAPPO is not only reduces MDP complexity but also significantly improves energy efficiency by eliminating high-energy-consuming and less-informative agents, while retaining the most impactful subset for decision-making.

Therefore, the entropy reduction across both levels confirms the emergence of adaptive stability, ensuring that once optimal behaviors are learned, they are maintained even as the external conditions shift. This balance between early exploration and long-term stability underpins **Hi-MAD**'s capacity to deliver sustained coordination, efficient energy use, and reliable adaptation in complex, dynamic IoT ecosystems. The results under-

line the importance of algorithm and architecture choice in multi-agent DRL systems, with CD-MAPPO, QS-MAPPO, Co-MADDPG and Co-MATD3 proving most efficient for rapid convergence, making them well-suited for time-sensitive IoT control applications over their baseline counterparts, MAPPO, MADDPG and MATD3, in terms of both learning speed and reward stability.

Overall, the results demonstrate that this hierarchical architecture with policy-gradient and coordinated-learning ((CD/QS)MAPPO, Co-MADDPG, Co-MATD3) substantially reduce convergence time compared to traditional value-based methods, ensuring faster adaptation and computational efficiency in large-scale multi-agent systems. The **Hi-MAD** framework faces challenges such as high training costs and sensitivity to hyperparameter settings. Achieving the best performance requires significant computational resources and careful tuning of learning and coordination parameters for different environments.

5.6 Experimental Validation of GAMMA

This section presents the experimental evaluation of the proposed framework for QoS-aware service composition in large-scale IoT environments. The experiments are designed to assess the effectiveness of the complete workflow, including context-aware representation learning, multi-objective service selection using, and final composition through the **GAMMA** framework. Using the QWS2 dataset, we simulate realistic service composition scenarios and evaluate the framework’s ability to optimize multiple QoS attributes while capturing inter-service dependencies.

5.6.1 Experimental Settings

The experimental evaluation of **GAMMA** framework is carried out using the QWS Dataset 2.0 [7], which consists of 2507 real-world web services characterized by nine QoS attributes, including response time, availability, throughput, successability, reliability, compliance, best practices, latency, and documentation . These attributes provide a comprehensive basis for assessing both performance and reliability aspects of service composition. Simulations ran on the same hardware of the HiMAD experiments, providing the necessary performance for MADRL services scenarios. The software stack included Python 3.10, PyTorch 2.0, Stable-Baselines3, and Gymnasium, alongside NumPy and Matplotlib for data processing and visualization. All the experiments settings are detailed in tables 1, 2, and 3.

To ensure a fair and reproducible comparison, all baselines (MAPPO, QS-MAPPO [21], MATD3, MADDPG, CTCE, Genetic Algorithm, Particle Swarm Optimization , Ant Colony Optimization and QAOA [67]) and the proposed GAMMA framework were trained and evaluated under identical conditions on the QWS2 dataset, using the same train/-validation split, the same nine normalized QoS attributes, and identical hard- and soft-constraint thresholds ($Av. \geq 70\%$, $Suc. \geq 70\%$, $Rel. \geq 50\%$, $RT \leq 1000$ ms; $\theta_{utility} = 0.35$). All learning-based methods share the same actor/critic backbone (2 hidden layers, 128–256 units, ReLU/tanh activations), the same optimizer (Adam), discount factor ($\gamma = 0.99$), GAE parameter ($\lambda_{gae} = 0.95$), and gradient clipping norm (0.5).

5.6.2 GAMMA Framework Evaluation

The experimental evaluation uses the QWS Dataset 2.0 [8], where QoS attributes are normalized and realistic service composition tasks are generated with candidate service sets for each abstract task. The framework integrates a MAPPO multi-head self-attention encoder for context-aware representations, followed by MAPPO-QoS+MO for multi-objective service selection and a MADDPG-GNN model for dependency-aware service composition under a multi-agent setting. Performance is assessed using QoS-based metrics, convergence behavior, and computational efficiency, with scalability validated across varying task sizes and results averaged over multiple runs for robustness.

MAPPO-Att-KMeans Context Discovery Evaluation

The MAPPO-Att-KMeans for context discovery module is evaluated on the QWS2 dataset [6], comprising 2507 real web services characterized by 9 QoS attributes. All attributes are min-max normalized to $[0, 1]$, and cost-type attributes (e.g., response time, latency) are

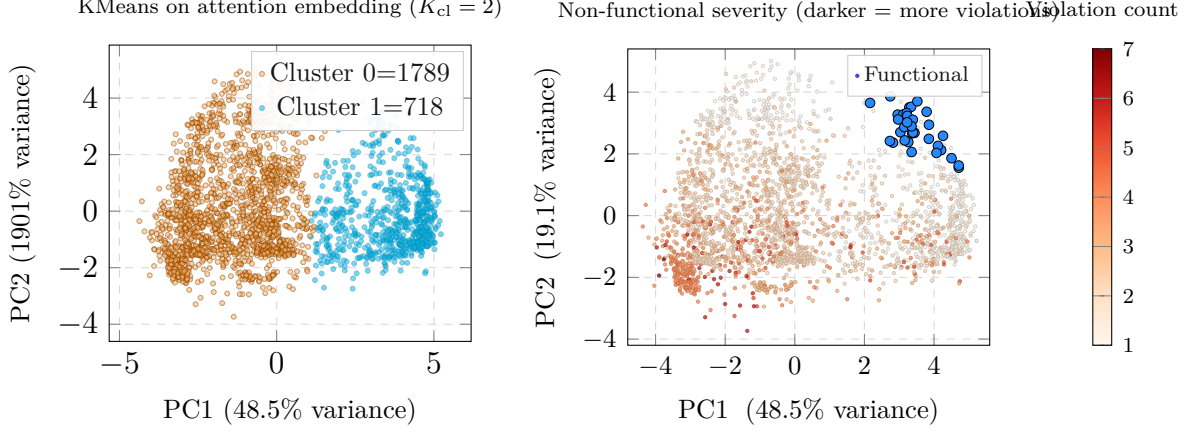


Figure 5.25: Service Distribution after to the context discovery step

transformed into benefit-type representations. This ensures that the functional-context predicate reduces to the monotonic condition: $\forall j : q_j \geq s_{j,\text{opt},i}$. When the MAPPO-Att-KMeans module processes a web service, it does not keep the service as a simple list of nine QoS numbers. Instead, the multi-head self-attention encoder transforms each service into a 32 number fingerprint which is an abstract representation in which services with similar quality profiles end up with similar fingerprints. The problem is that a 32-dimensional space cannot be drawn on paper. This is where *Principal Component Analysis* (PCA) comes in. PCA is a mathematical technique that finds, among all possible 2D views of a high-dimensional cloud of points, the one that preserves the most information. It does this by identifying the two directions along which the data varies the most:

- PC1 is the direction of largest variance in the 32-dimensional embedding space. It captures the single most important axis along which services differ from each other. In our case, PC1 alone preserves 48.5% of all the variation in the 32-dimensional fingerprints.
- PC2 is the direction of second-largest variance, perpendicular to PC1. It captures another 19.1% of the variation.

The KMeans-attention panel of Figure 5.25 demonstrates the efficiency of the module at three levels. First, the soft-KMeans head partitions the 2507 services into two spatially coherent clusters, Cluster 0 and Cluster 1. Cluster 1 can be read as the *candidate region* containing services with higher-quality QoS fingerprints). The alignment between the dominant embedding axis and the cluster boundary is a direct consequence of co-training the encoder and the cluster head under the shared cluster-context reward $\alpha_c C(a_i, s_t)$, confirming that the two modules converge to mutually consistent representations. Second, the partition aligns perfectly with the context predicate in one critical direction: all 34 functional services fall inside Cluster 1, and zero fall inside Cluster 0. This is the agreement between the unsupervised clustering and the supervised verification that the interpretation paragraph refers to —the two mechanisms independently identify the same elite region of the embedding space.

Based on this analysis, the context-verified sub-dataset V_c retained for subsequent processing corresponds to Cluster 0 ($V_c = 1789$ services, 71.4% of the original dataset), which is identified by the MAPPO-Att-KMeans module as the population of services

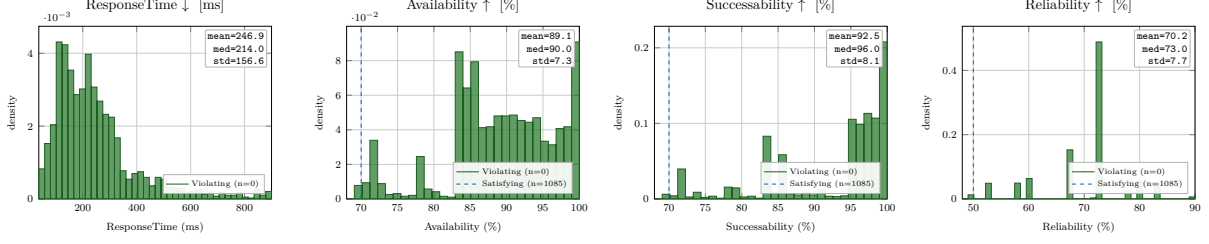


Figure 5.26: QoS hard-constraints distributions of the QoS-based sub-dataset (V_S , $n=1085$)

exhibiting QoS drift from the learned functional context and is therefore flagged for corrective adaptation—a prerequisite for ensuring the operational stability of the IoT environment in which these services are deployed.

Collectively, these observations demonstrate that the discovery module provides: (i) a geometrically separable functional subset suitable for subsequent QoS-aware selection (Stage 2), (ii) a structured neighborhood enabling sensitivity analysis under relaxed consensus criteria, and (iii) an interpretable ranking of non-functional services. These properties cannot be achieved using traditional percentile-based QoS filtering methods.

MAPPO QoS+MO Selection Evaluation

To evaluate the effectiveness of the MAPPO-QoS+MO selection module, the algorithm is trained on a context-relevant sub-dataset comprising $V_c = 1789$ services. Following the training phase, a refined QoS-based sub-dataset is generated. At this stage, the policy’s role is to evaluate composition-level QoS feasibility under both *hard* and *soft* constraint regimes. Each candidate service $v_i \in V_C$ is evaluated against the hard-threshold predicate $\bigwedge_{j \in J_{\text{hard}}} (q_i(j) \geq \theta_{\min}(j))$ on the four binding dimensions (Availability $\geq 70\%$, Successability $\geq 70\%$, Reliability $\geq 50\%$, ResponseTime ≤ 1000 ms), and is simultaneously scored by the soft-utility aggregate $U_i = \sum_{j \in J_{\text{soft}}} w_j f_j(q_i(j))$ on the five remaining dimensions (Throughput, Compliance, BestPractices, Documentation, Latency). Only services that *jointly* pass the hard-threshold test and exceed the soft-utility threshold $U_i \geq \theta_{\text{utility}} = 0.35$ are retained in the verified set V_S .

In our evaluation, this dual screening yields $V_S = 1085$ services, corresponding to a 39.4% reduction relative to V_C and a pass rate of 60.6%. The rejected complement $V_C \setminus V_S$, containing 704 services, comprises candidates that either fail at least one hard threshold, fall below the soft-utility cutoff, or both; these are preserved for diagnostic analysis but excluded from the MARL training loop.

As shown in Figure 5.26, the retained services strictly satisfy all hard QoS constraints with clear margins. The empirical distributions indicate mean values of response time ≈ 246.9 ms, availability $\approx 89.1\%$, successability $\approx 92.5\%$, and reliability $\approx 76.2\%$, all comfortably above their respective thresholds. Complementarily, Figure 5.27 confirms near-perfect compliance rates, with 100% satisfaction for availability (≥ 70), successability (≥ 70), reliability (≥ 50), and response time (≤ 1000 ms), and an overall soft-constraint satisfaction of 97.6%.

As illustrated in Figure 5.28, the filtered dataset enables superior learning performance across existing models. The proposed approach achieves the highest QoS satisfaction (99.5% for MAPPO-QoS+MO), outperforming MAPPO [19] (87.5%), QS-MAPPO [21] (95.3%), MATD3 (76.8%), MADDPG (80.1%), heuristics methods - GA (72%), PSO

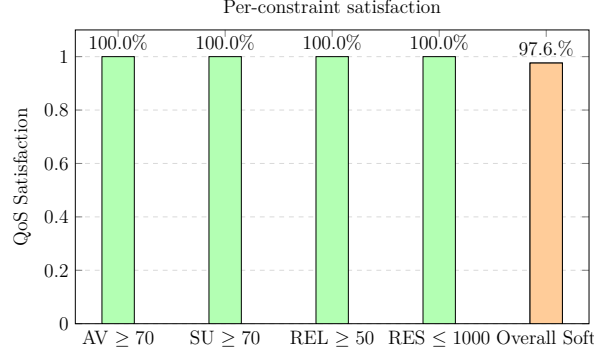


Figure 5.27: QoS Satisfaction of hard and soft constraints

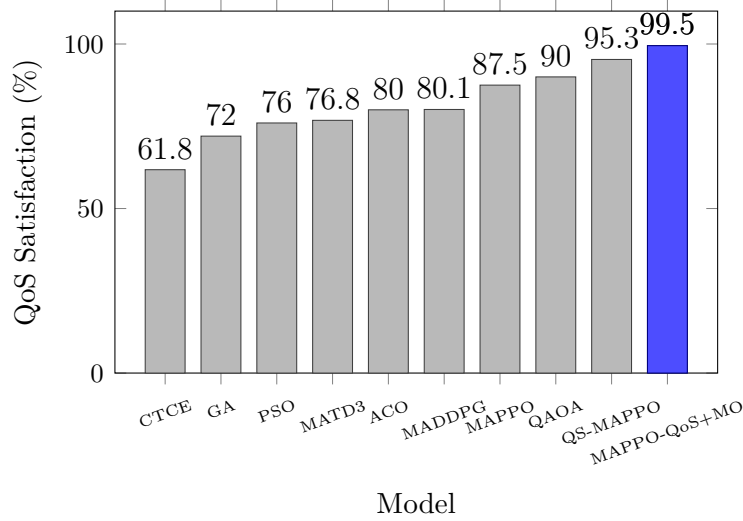


Figure 5.28: QoS Satisfaction Comparison

(76%), ACO (80%), QAOA (90%) [67]- and CTCE (61.8%), demonstrating that the screening process preserves high-quality candidates while facilitating more effective policy learning.

Service Composition Optimization

We evaluate MADDPG-GNN on a subset of 1085 web services, all services in the pool satisfy the hard QoS constraints imposed upstream in the pipeline, so the composition stage receives a pre-verified candidate set. The MADDPG-GNN composer combines two components. A two-layer Graph Attention Network with four attention heads encodes each service as a 32-dimensional relational embedding. Nodes correspond to services, while edges are established between pairs whose cosine similarity exceeds 0.85 on z -score-normalized QoS features. A MADDPG module instantiates agents per workflow slot, each implemented as an actor network that outputs a preference vector over the 1085 candidate services, conditioned on its local state, the mean GAT node embedding, and the global graph embedding. A centralized critic evaluates the joint composition under the CTDE paradigm.

The radar chart in Figure 5.29 compares the mean normalized utility of services selected by the composition (V_S , solid green) against those not selected ($V_C \setminus V_S$, dashed red) across nine QoS attributes. Consistently, the green curve lies outside the red curve on ev-

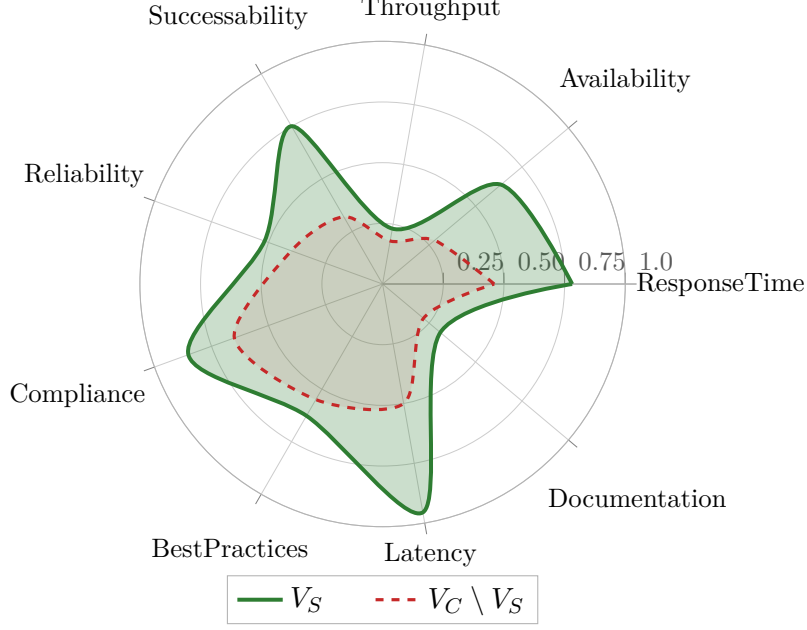


Figure 5.29: Mean normalized utility $\overline{f_j(q(j))}$

ery axis, confirming that the learned policy systematically selects higher-quality services across all dimensions. Notably, this dominance directly translates into a 100% composition success rate, indicating that the selected services are not only individually superior but also collectively reliable in satisfying all the context-rules and the QoS constraints without failure. The strong separation on attributes such as Availability, Reliability, Response Time and Successability highlights the key drivers behind this perfect success rate, as these dimensions are QoS hard constraints and critical for ensuring robust and executable compositions. This behavior aligns with their higher weights in the reward function and the energy model’s sensitivity.

Figure 5.30 compares the success rates of different algorithms for service composition. Traditional metaheuristic approaches such as GA (78%), PSO (82%), ACO (85%) and QAOA (95%) show moderate performance. In contrast, the proposed MADDPG-GNN approach achieves a perfect 100% success rate, clearly outperforming all baselines.

Energy Efficiency Minimization

Energy is not directly provided as a QoS attribute in the QWS2 dataset. Therefore, to evaluate energy minimization, we construct an energy model based on existing QoS metrics and assess how well the learned policy reduces energy while preserving QoS constraints. We define the energy consumption of a service E_i as:

$$E_i = \lambda \widehat{RT}_i \widehat{Thru}_i + \mu \widehat{Lat}_i + \nu \frac{Avail_i}{100} \frac{Rel_i}{100} + P_{base}, \quad (5.7)$$

with $\lambda=200$, $\mu=100$, $\nu=80$, $P_{base}=60$ W, and $\widehat{RT}_i, \widehat{Thru}_i, \widehat{Lat}_i \in [0, 1]$ min-max normalized against the full QWS2 bounds so that energies remain comparable across stages. The cumulative pool energy at stage k is then $E_{pool}^{(k)} = \sum_{s_i \in \mathcal{S}^{(k)}} E_i$.

First, the term $RT \times Thru$ represents the total computational workload. Response time reflects how long a request occupies system resources, while throughput indicates

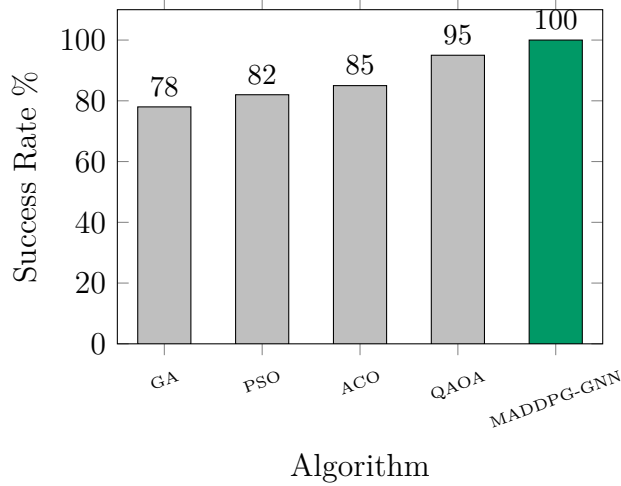


Figure 5.30: Success Rate of Service Composition

how many requests are processed. Their product therefore approximates CPU usage over time, which is a primary driver of energy consumption in data centers. Second, latency contributes to energy because higher delays often indicate inefficient resource usage or waiting times, which increase idle power consumption. Third, availability and reliability are included because maintaining high values typically requires redundancy, replication, and fault-tolerance mechanisms, all of which introduce additional energy overhead. Finally, P_{base} represents the static power consumption of the system (e.g., idle servers and infrastructure), which is always present regardless of workload.

Table 5.6: Filtering-induced size and energy reduction.

Stage	N	E_{pool} (W)	$\bar{E}$ (W)	Δ_{size}	Δ_{energy}
QWS2 (raw)	2,507	272,397.51	108.65	—	—
Context-Based	1,789	185,297.86	103.58	28.6 %	31.9 %
QoS-Based	1,085	105,914.63	97.62	56.7 %	61.1 %

Table 5.6 reports $E_{\text{pool}}^{(k)}$ at the three pipeline stages. $\bar{E}$ is the per-service mean energy. This table highlights the cumulative impact of a multi-stage filtering pipeline on reducing the size of the service pool as well as on their average and total energy consumption. We first observe an initial context-based filtering phase, which acts as a mechanism for roughly reducing the search space. It reduces the number of services from 2507 to 1789, a decrease of 28.6%. This step is already accompanied by a substantial decrease in the pool’s total energy consumption (from 272,397.51 W to 185,297.86 W), indicating that the elimination of context-irrelevant services also removes energy-intensive components. In a second step, filtering based on QoS constraints further refines the selection by incorporating stricter performance criteria. This step further reduces the number of services to 1085, corresponding to an overall reduction of 56.7% compared to the initial dataset. At the same time, total energy consumption drops to 105,914.63 W, representing an overall improvement of 61.1%. We also observe a gradual decrease in average energy per service (from 108.65 W to 97.62 W), suggesting that filtering affects not only the quantity but also the energy efficiency of the retained services.

The right figure of Figure 5.31 presents the energy efficiency metric, defined as

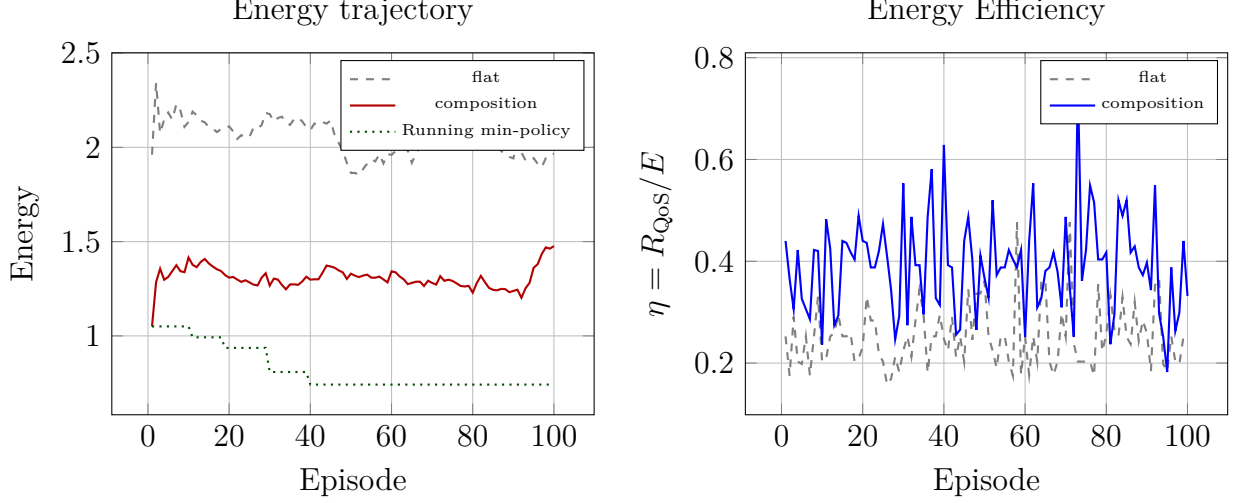


Figure 5.31: Energy Consumption

$\eta = \frac{R_{\text{QoS}}}{E}$, where the QoS reward is divided by the total energy consumption. The composition optimization policy (blue line) consistently achieves higher efficiency values than the flat policy, with several peaks reaching nearly 0.75, while the flat policy remains mostly between 0.2 and 0.35. This indicates that the MADDPG-GNN-based composition strategy not only reduces energy consumption but also maintains better service quality per unit of energy used. Although the running min-policy consumes less energy overall, it is not shown as the most efficient because lower energy alone does not guarantee high QoS. Therefore, the composition optimization policy demonstrates the best trade-off between performance and energy consumption, confirming the effectiveness of MADDPG-GNN in energy-aware optimization.

The left figure of Figure 5.31 shows the energy trajectory of three policies across 100 episodes: the flat policy, the composition optimization policy learned by MADDPG-GNN, and the running min-policy. The flat policy maintains the highest energy consumption, remaining around 2.0-2.2 units, which indicates inefficient resource utilization due to the absence of adaptive coordination. In contrast, the composition optimization policy significantly reduces energy consumption and stabilizes around 1.2-1.4 units, demonstrating that MADDPG-GNN successfully learns cooperative decisions that minimize unnecessary energy usage. Although the running min-policy achieves the lowest energy consumption, it may do so by overly restricting system operations, which can negatively affect service quality. This shows that the MADDPG-GNN composition policy provides a better balance between energy saving and system performance.

Figure 5.32 illustrates the reward evolution of the MADDPG-GNN algorithm across 200 training episodes, highlighting its learning stability and convergence capability. The blue curve represents the smoothed reward, which shows a steady upward trend from an initial value of approximately 0.47 to above 0.52 in later episodes, indicating continuous improvement in policy quality. The red dashed line marks 95% of the maximum reward, while the green vertical line at around episode 100 indicates the convergence point where the learning process becomes stable. This demonstrates the power of MADDPG-GNN in efficiently learning optimal cooperative strategies among multiple agents. By combining MADDPG with GNN, the model captures both agent interactions and system dependencies, enabling faster convergence and better coordination than traditional rein-

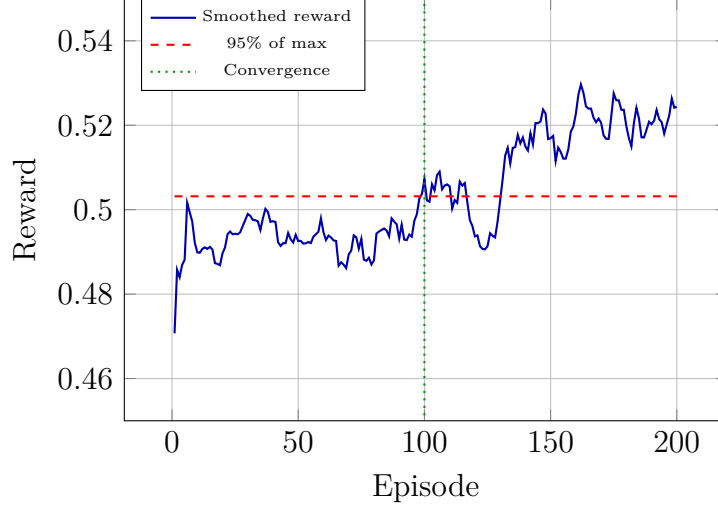


Figure 5.32: Reward Evolution and Convergence

forcement learning methods. The stable convergence after episode 100 proves that the algorithm successfully learns a high-quality policy with reduced training instability, making it highly effective for complex service composition optimization problems where both QoS and energy efficiency must be jointly optimized.

5.6.3 Experimental results discussion

MAPPO-Att+KMeans enhances service discovery by learning meaningful representations, organizing services into context-consistent clusters, and ranking non-functional properties along a clear severity scale. Together, these capabilities show that attention, clustering, and threshold optimization operate as an integrated system, outperforming traditional percentile-based QoS filtering. On the verified pool of services, MAPPO-QoS+MO jointly enforces the hard-threshold contract and the soft-utility preference, converging to a Pareto-optimal frontier of service compositions whose aggregated QoS profile dominates uniformly-random selection on all three objectives (ResponseTime, Throughput, Reliability) while satisfying every composition-level constraint by construction. The results show that the filtering pipeline is structurally effective: it significantly reduces the search space while improving the energy characteristics of the service pool, which is a favorable prerequisite for downstream optimization and composition mechanisms. Results demonstrate that the MADDPG-GNN-based composition optimization significantly improves system energy performance. While the flat policy exhibits the highest energy consumption and lowest efficiency, the learned composition policy reduces energy usage while maintaining superior QoS, resulting in the highest energy efficiency. Although the running min-policy achieves lower absolute energy consumption, it likely sacrifices service quality. Therefore, MADDPG-GNN successfully learns a balanced cooperative strategy that optimizes both energy expenditure and operational effectiveness. Results also indicate that integrating multi-agent deep reinforcement learning with graph neural networks enables more reliable and consistent compositions, effectively capturing complex dependencies between services and ensuring successful execution in all cases.

Three deployment factors merit explicit acknowledgment. *First*, training is performed offline; periodic retraining (e.g., daily or weekly) is sufficient for the time-scales of QoS drift observed in QWS-class registries. *Second*, the upstream filtering stages substantially

shrink the action space exposed to Stage 3 (from 2,507 to 1,085 services), which is the principal driver of the observed convergence speed. *Third*, for very large registries ($N \gg 10^4$), the $\mathcal{O}(N^2d)$ attention cost in Stage 1 becomes the dominant bottleneck; in such regimes, replacing full self-attention with a linear-attention variant would preserve the framework’s behavior while reducing complexity to $\mathcal{O}(Nd^2)$.

5.7 Conclusion

This chapter established the correctness and effectiveness of the proposed frameworks along two complementary axes: *formal verification*, which guarantees that the structural transformations underpinning our pipelines preserve the properties they are intended to preserve, and *experimental validation*, which demonstrates that these guarantees translate into measurable performance on a realistic service pool.

We formalized the IoT system architecture as an Event-B context-and-machine hierarchy and discharged its proof obligations, establishing an internally consistent, invariant-preserving substrate for the reduction proofs that follow. On this basis we verified the two reduction mechanisms that make large-scale composition tractable: GAMMA’s Environment Size Reduction, proven through a multi-machine refinement chain to contract the candidate set monotonically while preserving feasibility so that no admissible solution is discarded, and HiMAD’s MDP reduction, shown to yield a reduced decision process whose optimal policies remain sound with respect to the original formulation. The numerical reduction reported empirically in Table 5.4 ($500 \rightarrow 416 \rightarrow 358$) is formally guaranteed: the agent count is non-increasing across filter stages.

The formal guarantees were then corroborated empirically. The experimental validation of HiMAD confirmed that the verified MDP reduction preserves solution quality while substantially lowering the effective decision space, yielding the expected gains in convergence and composition quality on the benchmark dataset. The validation of GAMMA likewise demonstrated that the proven Environment Size Reduction delivers the anticipated contraction of the service pool without degrading the QoS of the resulting compositions, evidencing the practical efficiency that the formal proofs promise in principle.

Taken as a whole, this chapter closes the loop between specification and behavior: the properties asserted by the Event-B models are exactly those observed to hold under evaluation, so the performance reported here is attributable to provably correct reductions rather than to unverified heuristics. This dual guarantee—formal soundness paired with empirical efficiency—constitutes the principal contribution of the chapter and motivates the work that follows, where these verified components are integrated into the end-to-end composition framework and examined under broader operating conditions.

Chapter 6

Conclusion and Future Work

This thesis has introduced **Hi-MAD** and **GAMMA** a three-level hierarchical and graph attention multi-agent DRL framework for managing large-scale IoT services with resilience and efficiency. Its core contributions span the design of a hierarchical DRL architecture (global/mid/low levels), a context-aware QoS-driven coordination mechanism, and specialized layer-specific learning agents. Together, these advances enable adaptive service orchestration, high resilience to failures, and significant energy savings. In our smart-building case study, achieved over 99.5% QoS satisfaction, success rate of 100% of service composition, and reduced energy use by 40% compared to baselines. Nevertheless, practical deployment faces challenges: for instance, even hierarchies struggle at Internet-of-Things scale (thousands to millions of devices), and each agent has only partial observability of the system (a POMDP scenario). To address these, future work will incorporate transfer learning (to speed up training), federated multi-agent RL (to scale and preserve privacy), and adversarial learning for security (robust against evolving attacks). The following sections summarize the chapters’ contributions, discuss practical constraints, and outline a phased roadmap for future research.

6.1 Summary of Contributions

This work addresses key research questions about scalable, resilient IoT service management by decomposing the problem into hierarchical layers illustrated in Figure 6.1. Chapter 1 formulated the problem and defined the research goals (such as “How can a decentralized IoT control architecture ensure service QoS and resilience under dynamic conditions?”). Chapter 3 introduced the IoT-MAS architecture. The framework uses centralized training, decentralized execution to balance coordination with local autonomy. Chapter 4 developed a context-aware QoS coordination mechanism, a two-stage algorithm that discovers context (e.g. network conditions) and dynamically composes services to meet application constraints and detailed the layer-specific learning algorithms: each agent learns a policy tailored to its scope, with fast convergence and stability. Finally, Chapter 5 validates the contributions across Event-B formal verification and use case with real dataset experimentation: we demonstrated that the hierarchical model significantly improves system resilience (fault tolerance) and energy efficiency while satisfying QoS requirements (over 99.5% satisfied) compared to flat baselines. The methodological advances include integrating centralized training/decentralized execution (CTDE) in a multi-level IoT control loop, and incorporating high-level resilience objectives into the reward structure.

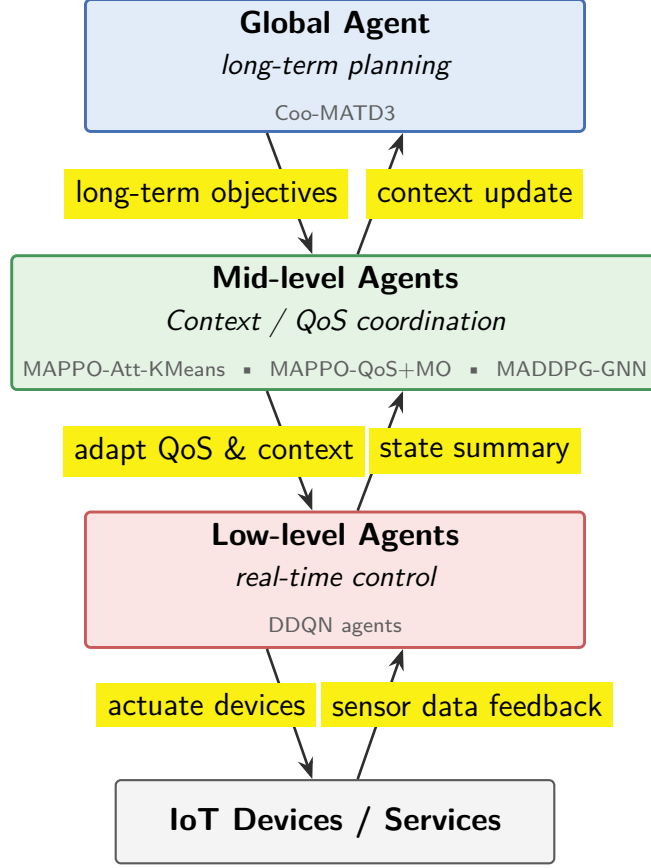


Figure 6.1: Hierarchical-Graph Control Flow

6.2 Limitations and Practical Considerations

While **Hi-MAD** and **GAMMA** demonstrate strong performance in simulation, several practical constraints must be acknowledged. Our experiments used up to 500 agents; real IoT deployments may involve orders of magnitude more endpoints. Beyond a point, simple hierarchical designs may need dynamic clustering or additional meta-coordination to manage complexity. Real-time demands and cost: Each layer’s DRL agents require significant training (computational cost) and may introduce latency. Fast inference is necessary for real-time IoT control, so computational efficiency (e.g. lightweight models or on-device acceleration) must be considered. The centralized training phase, while enabling coordination, also assumes reliable data collection and communication: in practice, transmitting experiences for CTDE can be bandwidth- and energy-intensive. Partial Observability: In most IoT scenarios, no single agent has full system knowledge. **Hi-MAD**’s design approximates a Partially Observable MDP: each agent acts on local observations (similar to other UAV/IoT studies) and must infer missing information. This may degrade optimality if system state is not properly estimated. Domain-specific factors: Our case study (smart building control) may not capture all IoT domains. For instance, industrial IoT might impose stricter safety or timing constraints; mobile IoT (like vehicles) adds topology changes. Adapting **Hi-MAD** to such domains may require custom observation models or safety constraints. Cost and infrastructure: Deploying and maintaining a multi-agent DRL system entails overhead (development, tuning, monitoring). Assurance of robust operation (e.g. against hardware failures) may require additional safeguards.

beyond the DRL policies. In summary, our approach advances IoT management theory and methods, but real-world use will involve addressing these scalability, observability, and infrastructure challenges.

Three deployment factors merit explicit acknowledgment. *First*, training is performed offline; periodic retraining (e.g., daily or weekly) is sufficient for the time-scales of QoS drift observed in QWS-class registries. *Second*, the upstream filtering stages substantially shrink the action space exposed to Stage 3 (from 2,507 to 1,085 services), which is the principal driver of the observed convergence speed. *Third*, for very large registries ($N \gg 10^4$), the $\mathcal{O}(N^2d)$ attention cost in Stage 1 becomes the dominant bottleneck; in such regimes, replacing full self-attention with a linear-attention variant would preserve the framework’s behavior while reducing complexity to $\mathcal{O}(Nd^2)$.

6.3 Future Work

To further advance this research, we envision several directions. **Transfer Learning:** Leveraging prior knowledge can greatly reduce training effort. For example, hierarchical transfer learning has achieved 78% fewer training episodes in a related IoT energy control task. We will explore policy transfer across similar IoT contexts (e.g. different buildings or time periods) so that new agents can quickly adapt without learning from scratch. This could involve transferring neural network weights between environments or using meta-RL techniques. **Federated Multi-Agent Reinforcement Learning:** Privacy and scalability can be addressed by enabling decentralized model training. In a federated setup, edge nodes (or mid-level controllers) would train local policies and share model updates rather than raw data. Such Federated MARL allows each agent to learn independently while contributing to a global model, this creates “a secure, distributed MARL paradigm” suited for large-scale, privacy-sensitive environments. Future work will develop FRL protocols for our approach, including secure aggregation and asynchronous updates, to deploy across distributed IoT networks (smart buildings, cities, etc.). **Robust Security and Adversarial Training:** IoT systems face active threats. We plan to integrate adversarial learning into our approach: for example, training against simulated “attackers” (malicious agents) or perturbations. Recent work shows that co-training a hierarchical defender against an evolving attacker yields policies that remain effective against novel (“zero-day”) threats. Adapting such adversarial frameworks will improve our approach’s resilience to cyber-physical attacks. **Large-Scale and Industrial Deployment:** Ultimately, our approach should be tested in real or emulated industrial IoT settings. Near-term phases will involve scaling experiments (e.g. thousands of simulated agents) and integrating with digital-twin platforms.

Table 1: Experimental settings for MAPPO-Context discovery on QWS2.

MAPPO-Context Discovery – Experimental Configuration	
Parameter	Value
<i>Dataset</i>	
Source QWS2 dataset	QWS2 dataset 2507
QoS attributes J	9
Normalization	Min-max $[0, 1]$ + benefit-form flip
<i>Multi-agent setup</i>	
Number of agents N	4
Action space (per agent)	Discrete $\{0 = \text{reject}, 1 = \text{accept}\}$
Max episodes $K_{\max}$	500
Buffer policy	On-policy (cleared each episode)
<i>Attention encoder</i>	
Type	Multi-head self-attention
Embedding dim d_{model}	32
Number of heads	4
Residual + LayerNorm	Yes
Actor / critic hidden units	128 (2 layers, ReLU)
<i>Soft KMeans</i>	
Number of clusters K_{cl}	2
Initialization	k -means++
Softmax temperature τ	1.0
Centroid EMA rate η_{μ}	0.05
<i>PPO / MAPPO optimization</i>	
Optimizer	Adam (joint actors + critic + encoder)
Learning rate α_{lr}	3×10^{-4}
PPO epochs M	4
Mini-batch size	32 steps
Clip ratio ϵ	0.2
Discount γ	0.99
GAE coefficient λ	0.95
Entropy / value coefficients	0.01 / 0.5
<i>Threshold update</i>	
Initial value $s_{j,\text{opt},i}^{(0)}$	0.5 (midrange)
Context learning rate η	10^{-2}
Decay λ_{decay}	0.995 / episode
Trust-region $\Delta_{\max}^{(0)}$	0.25 (linear shrink)
<i>Reward weights</i>	
Cluster-context bonus α_c	0.20
Latency penalty β_L	0.10
Energy penalty $\gamma_{\mathcal{E}}$	0.05
SLA-violation penalty δ_B	0.20
Hard SLA floor	10th percentile per attribute
<i>Convergence thresholds</i>	
Policy KL ϵ_{π}	10^{-3}
Critic drift ϵ_V	5×10^{-3}
Reward stability ϵ_R	10^{-3}
Threshold stability ϵ_s	5×10^{-4}
<i>Implementation</i>	
Framework	PyTorch 2.x
Language	Python 3.11
Consensus rule (verification)	Majority vote ($\geq \lceil N/2 \rceil$)

Table 2: Experimental settings for MAPPO-QoS+MO on QWS2.

MAPPO-QoS+MO – Experimental Configuration	
Symbol / Parameter	Value
<i>Dataset & constraints</i>	
Pre-qualified pool $ V_C^{(1)} $	1789
Verified set $ V_S $	1085
Hard constraints $\theta_{\min}(j)$	Av. $\geq 70\%$, Suc. $\geq 70\%$, Rel. $\geq 50\%$, RT ≤ 1000 ms
Soft weights w_j	(0.30, 0.20, 0.25, 0.15, 0.10)
Utility threshold θ_{utility}	0.35
<i>Multi-objective formulation</i>	
Number of objectives m	3 (RT, Thr, Rel)
Aggregators $\mathcal{A}_j$	sum, min, product
Penalty coefficient κ_j	2.0
<i>MAPPO architecture</i>	
Number of agents N	6
Hidden units	128
Hidden layers	2, tanh
Shared policy π_ϕ	parameter sharing
Centralized critics $\{V_{\psi_j}\}_{j=1}^m$	one per objective
<i>Training schedule</i>	
Epochs K	60
Episodes per epoch M	500
Horizon T	8
PPO iterations U	6
Minibatch size B	128
Policy LR η_ϕ	3×10^{-4}
Critic LR η_ψ	1×10^{-3}
Grad-norm clipping	0.5
Discount γ	0.99
GAE parameter λ_{gae}	0.95
PPO clip range ϵ_{clip}	0.2
Entropy coefficient β	0.01
<i>Multi-objective control</i>	
Preference update rate η	0.10
Archive capacity $P_{\max}$	60
Archive pruning	NSGA-II crowding distance
Zero-violation guard ϵ_0	10^{-6}
HV convergence tolerance ϵ	10^{-4}
Violation convergence ϵ_v	10^{-3}
<i>Reproducibility</i>	
Wall-clock per run	≈ 8 minutes

Table 3: Experimental settings for MADDPG-GNN composition on QWS2.

Category	Symbol / Parameter	Value
<i>Dataset & task</i>		
Candidate pool	V_C	1,085 verified services
QoS attributes	d_{qos}	9
Aggregation rules	$\mathcal{A}_j$	sum / product / min / mean
<i>Graph construction</i>		
Edge rule	–	Cosine similarity on z -scored QoS
Edge threshold	τ_{edge}	0.85
Attention variant	–	Sparse GAT
<i>GAT encoder</i>		
Input dimension	d_{in}	9
Hidden dimension	d_{h}	64
Output dimension	d_{out}	32
Number of layers	L_{GAT}	2
Attention heads	H	4
<i>MADDPG architecture</i>		
Actor network	μ_{θ_i}	3-layer MLP, 128 hidden, ReLU
Actor output activation	–	tanh
Critic network	Q_{ϕ_i}	3-layer MLP, 256 hidden, ReLU, CTDE
<i>Reward formulation</i>		
Reward function	R	$(1 - \lambda) R_{\text{QoS}} - \lambda \tilde{E}_{\text{workflow}}$
Energy weight	λ	0.30
Normalization	–	Min-max (energy), $[0, 1]$ per attr. (QoS)
<i>Training schedule</i>		
Episodes	K	2000
Batch size	B	64
Replay buffer	$ \mathcal{D} $	10^5 transitions
Actor learning rate	η_{θ}	1×10^{-4}
Critic learning rate	η_{ϕ}	1×10^{-3}
Discount factor	γ	0.95
Soft target update	τ	0.01
Exploration noise	σ	Gaussian, $\sigma_0=0.1$, decay 0.995, $\sigma_{\text{min}}=0.01$
Gradient clipping	–	0.5 (global norm)

Bibliography

- [1] Dezheen H Abdulazeez and Shavan K Askar. A novel offloading mechanism leveraging fuzzy logic and deep reinforcement learning to improve iot application performance in a three-layer architecture within the fog-cloud environment. *IEEE Access*, 2024.
- [2] Dezheen H Abdulazeez and Shavan K Askar. A novel offloading mechanism leveraging fuzzy logic and deep reinforcement learning to improve iot application performance in a three-layer architecture within the fog-cloud environment. *IEEE Access*, 2024.
- [3] Muhammad Adnan, Madiha Haider Syed, Adeel Anjum, and Semeen Rehman. A framework for privacy-preserving in iot using federated learning with differential privacy. *IEEE Access*, 2025.
- [4] Souhila Ait Hacène Ouhadda, Samia Chibani Sadouki, Achour Achroufene, and Abdelkamel Tari. A discrete adaptive lion optimization algorithm for qos-driven iot service composition with global constraints. *Journal of Network and Systems Management*, 32(2):34, 2024.
- [5] Abeer Z Al-Marridi, Amr Mohamed, and Aiman Erbad. Optimized blockchain-based healthcare framework empowered by mixed multi-agent reinforcement learning. *Journal of Network and Computer Applications*, 224:103834, 2024.
- [6] Eyhab Al-Masri and Qusay H Mahmoud. Discovering the best web service. In *Proceedings of the 16th international conference on World Wide Web*, pages 1257–1258, 2007.
- [7] Eyhab Al-Masri and Qusay H Mahmoud. Qos-based discovery and ranking of web services. In *2007 16th international conference on computer communications and networks*, pages 529–534. IEEE, 2007.
- [8] Eyhab Al-Masri and Qusay H Mahmoud. Investigating web services on the world wide web. In *Proceedings of the 17th international conference on World Wide Web*, pages 795–804, 2008.
- [9] Someah Alangari. An unsupervised machine learning algorithm for attack and anomaly detection in iot sensors. *Wireless Personal Communications*, 2024.
- [10] Ahmad Alferidi, Mohammed Alsolami, Badr Lami, and Sami Ben Slama. Ai-powered microgrid networks: Multi-agent deep reinforcement learning for optimized energy trading in interconnected systems. *Arabian Journal for Science and Engineering*, 50(8):6157–6179, 2025.

- [11] Guillen-Perez Antonio and Cano Maria-Dolores. Multi-agent deep reinforcement learning to manage connected autonomous vehicles at tomorrow's intersections. *IEEE Transactions on Vehicular Technology*, 71(7):7033–7043, 2022.
- [12] Fahim Arif, Nauman Ali Khan, Javed Iqbal, Faten Khalid Karim, Nisreen Innab, Samih M Mostafa, et al. Dqqs: Deep reinforcement learning based technique for enhancing security and performance in sdn-iot environments. *IEEE Access*, 2024.
- [13] KyeongDeok Baek and In-Young Ko. Multifedrl: Efficient training of service agents for heterogeneous internet of things environments. *IEEE Internet of Things Journal*, 2025.
- [14] Anjan Bandyopadhyay, Vagisha Mishra, Sujata Swain, Kalyan Chatterjee, Sweta Dey, Saurav Mallik, Amal Al-Rasheed, Mohamed Abbas, and Ben Othman Soufiene. Edgematch: A smart approach for scheduling iot-edge tasks with multiple criteria using game theory. *IEEE Access*, 2024.
- [15] Maggi Bansal, Inderveer Chana, and Siobhán Clarke. Urbanenqosplace: A deep reinforcement learning model for service placement of real-time smart city iot applications. *IEEE Transactions on Services Computing*, 16(4):3043–3060, 2022.
- [16] Arturo Barriga, José A Barriga, Miguel A Pérez-Toledano, and Pedro J Clemente. Model-driven development towards distributed intelligent systems. *ACM Transactions on Internet Technology*, pages 1–28, 2024.
- [17] Chaima Bejaoui, Fatma Achour, Nasreddine Hajlaoui, and Anis Jedidi. An iot semantic web service for objects identifying. In *2024 IEEE 27th International Symposium on Real-Time Distributed Computing (ISORC)*, pages 1–6. IEEE, 2024.
- [18] Fatima Zohra Bemrah, Azeddine Chikh, and Samir Ouchani. A comparative analysis of service management techniques in the internet of things. In *2025 International Conference on Intelligent Computer Systems, Data Science and Applications (IC2SDA)*, pages 1–8. IEEE, 2025.
- [19] Fatima Zohra Bemrah, Azeddine Chikh, and Samir Ouchani. A deep reinforcement learning-based multi-agent framework for dynamic optimization of qos in iot services. In *2025 28th International Symposium on Real-Time Distributed Computing (ISORC)*, pages 1–10. IEEE, 2025.
- [20] Fatima-Zohra Bemrah, Samir Ouchani, and Azeddine Chikh. A review on multi-agent deep reinforcement learning for iot: Techniques and applications. In *2025 IEEE/ACS 22nd International Conference on Computer Systems and Applications (AICCSA)*, pages 1–8. IEEE, 2025.
- [21] Fatima-Zohra Bemrah, Azeddine Chikh, and Samir Ouchani. Hi-mad: A hierarchical multi-agent drl framework for resilience and service optimization in iot systems. *Information and Software Technology*, page 108151, 2026.
- [22] Fatima Zohra Bemrah, Samir Ouchani, and Azeddine Chikh. Gamma: A madrl framework with graph attention and multi-objective optimization for qos aware iot services. *Journal of Systems Architecture*, page 103887, 2026.

- [23] Samir Berrani, Ali Yachir, Badis Djemaa, and Mohamed Aissani. Extended multi-agent system based service composition in the internet of things. In *2018 3rd International Conference on Pattern Analysis and Intelligent Systems (PAIS)*, pages 1–8. IEEE, 2018.
- [24] Marlom Bey, Pratyay Kuila, Banavath Balaji Naik, and Santanu Ghosh. Quantum-inspired particle swarm optimization for efficient iot service placement in edge computing systems. *Expert Systems with Applications*, 236:121270, 2024.
- [25] Manel Boutarfa, Ramdane Maamri, and Mohammed Nassim Lacheheb. Towards an approach for cloud service composition in multi-cloud environment based qos using deep q-learning. In *2022 International Conference on Advanced Aspects of Software Engineering (ICAASE)*, pages 1–7. IEEE, 2022.
- [26] Pavlos S Bouzinis, Panagiotis Radoglou-Grammatikis, Ioannis Makris, Thomas Lagkas, Vasileios Argyriou, Georgios Th Papadopoulos, Panagiotis Sarigiannidis, and George K Karagiannidis. Statavg: Mitigating data heterogeneity in federated learning for intrusion detection systems. *IEEE Transactions on Network and Service Management*, 2025.
- [27] Buqing Cao, Mi Peng, Lulu Zhang, Yueying Qing, Bing Tang, Guosheng Kang, and Jianxun Liu. Web service recommendation via integrating heterogeneous graph attention network representation and fibinet score prediction. *IEEE Transactions on Services Computing*, 16(5):3837–3850, 2023.
- [28] Mi Chen, Jalel Ben-Othman, and Lynda Mokdad. Greedy behavior detection with machine learning for lorawan network. *IEEE Transactions on Network and Service Management*, 21(3):2731–2740, 2024.
- [29] Yi Chen, Lianglun Cheng, and Tao Wang. Deep reinforcement learning for qos-aware iot service composition: The pd3qnd approach. In *2023 IEEE 14th International Conference on Software Engineering and Service Science (ICSESS)*, pages 38–41. IEEE, 2023.
- [30] Yourong Chen, Yang Zhang, Yubo Zhuang, Kelei Miao, Seyedamin Pouriyeh, and Meng Han. Efficient and secure blockchain consensus algorithm for heterogeneous industrial internet of things nodes based on double-dag. *IEEE Transactions on Industrial Informatics*, 20(4):6300–6312, 2024.
- [31] Hyebin Cho, Hyungsub Kim, JeeHyeon Na, Seung-Chan Lim, and Howon Lee. Multi-agent distributed dqn and transfer learning for energy-efficient power management in solar energy-harvested small-cell networks. *IEEE Internet of Things Journal*, 2025.
- [32] Fulvio Corno, Luigi De Russis, and Alberto Monge Roffarello. From users’ intentions to if-then rules in the internet of things. *ACM Transactions on Information Systems (TOIS)*, pages 1–33, 2021.
- [33] Monika Dangore, Deepali Bhatarkar, Kanchan M Bhale, Hemlata Makarand Jadhav, Vishal Kisan Borate, and Yogesh Kisan Mali. Applying random forest for iot systems in industrial environments. In *2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT)*. IEEE, 2024.

- [34] Ibrahim Dhaini, Soha Rawas, and Ali El-Zaart. Simulated annealing-based energy and traffic-aware virtual machine placement in fog computing. In *2025 5th International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET)*, pages 1–7. IEEE, 2025.
- [35] Jianbo Du, Haobo Yan, Aijing Sun, Chong Wang, Yuan Gao, Zhiquan Liu, and Xiaoli Chu. Fairness-guaranteed joint access control, 3-d uav trajectory planning, and task scheduling in agins. *IEEE Transactions on Vehicular Technology*, 2025.
- [36] Bangji Fan, Xinghua Liu, Gaoxi Xiao, Yu Kang, Dianhui Wang, and Peng Wang. Attention-based multiagent graph reinforcement learning for service restoration. *IEEE Transactions on Artificial Intelligence*, 5(5):2163–2178, 2023.
- [37] Bangji Fan, Xinghua Liu, Yuanzhe Wang, Gaoxi Xiao, Yu Kang, and Danwei Wang. Graph-based heterogeneous multiagent reinforcement learning for distribution system service restoration. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2026.
- [38] Sajad Faramarzi, Sepideh Javadi, Farshad Zeinali, Hosein Zarini, Mohammad Robat Mili, Mehdi Bennis, Yonghui Li, and Kai-Kit Wong. Meta reinforcement learning for resource allocation in aerial active-ris-assisted networks with rate-splitting multiple access. *IEEE Internet of Things Journal*, 2024.
- [39] Babak Farhadi, Parvaneh Asghari, Ebrahim Mahdipour, and Hamid Haj Seyyed Javadi. A novel recommendation-based framework for reconnecting and selecting the efficient friendship path in the heterogeneous social iot network. *Computer Networks*, 258:111016, 2025.
- [40] Amal Feriani and Ekram Hossain. Single and multi-agent deep reinforcement learning for ai-enabled wireless networks: A tutorial. *IEEE Communications Surveys & Tutorials*, 23(2):1226–1252, 2021.
- [41] Anastasios Giannopoulos, Ilias Paralikas, Sotirios Spantideas, and Panagiotis Trakadas. Cooler: Cooperative computation offloading in edge-cloud continuum under latency constraints via multi-agent deep reinforcement learning. In *2024 International Conference on Intelligent Computing, Communication, Networking and Services (ICCNIS)*, pages 9–16, 2024.
- [42] Ameni Harrabi, Hamdi Hassen, Khlil Ahmad Alsulbi, and Meftah Zouai. Iot environment based on multi agent system in smart home. In *International Conference on Advanced Information Networking and Applications*, pages 415–426. Springer, 2024.
- [43] Selman Hizal, Unal Cavusoglu, and Devrim Akgun. A novel deep learning-based intrusion detection system for iot ddos security. *Internet of Things*, 2024.
- [44] Mohammad Arif Hossain, Weiqi Liu, and Nirwan Ansari. Computation-efficient offloading and power control for mec in iot networks by meta-reinforcement learning. *IEEE Internet of Things Journal*, 11(9):16722–16730, 2024.
- [45] Zhongwei Huang, Fei Richard Yu, and Jun Cai. Knowledge-collaboration-based resource allocation in 6g iot: A graph attention rl approach. *IEEE Internet of Things Journal*, 11(22):36581–36595, 2024.

- [46] Yunwei Jin, Ruisi He, Bo Ai, Tong Wu, Haoxiang Zhang, Bincheng Liu, Yujian Li, and Jing Li. A novel geometry-based stochastic channel model in integrated sensing and communication scenarios. *IEEE Wireless Communications Letters*, 13(7):2018–2022, 2024. doi: 10.1109/LWC.2024.3401482.
- [47] Thandra Jithendra, Mohammad Zubair Khan, S Sharief Basha, Raja Das, A Divya, Chiranji Lal Chowdhary, Abdulrahman Alahmadi, and Ahmed H Alahmadi. A novel qos prediction model for web services based on an adaptive neuro-fuzzy inference system using coot optimization. *IEEE Access*, 2024.
- [48] Mina Emami Khansari and Saeed Sharifian. A scalable modified deep reinforcement learning algorithm for serverless iot microservice composition infrastructure in fog layer. *Future Generation Computer Systems*, pages 206–221, 2024.
- [49] Mina Emami Khansari and Saeed Sharifian. A deep reinforcement learning approach towards distributed function as a service (faas) based edge application orchestration in cloud-edge continuum. *Journal of Network and Computer Applications*, 233:104042, 2025.
- [50] Dong-Seong Kim, Ikechi Saviour Igboanusi, Love Allen Chijioke Ahakonye, and Goodness Oluchi Anyanwu. Proof-of-authority-and-association consensus algorithm for iot blockchain networks. In *2025 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–6. IEEE, 2025.
- [51] Han Li, Shunmei Meng, Jing Shang, Anqi Huang, and Zhicheng Cai. Value-based multi-agent deep reinforcement learning for collaborative computation offloading in internet of things networks. *Wireless Networks*, 30(8):6915–6928, 2024.
- [52] Yuanfeng Li, Qi Zhang, Haipeng Yao, Ran Gao, Xiangjun Xin, and Mohsen Guizani. Next-gen service function chain deployment: Combining multi-objective optimization with ai large language models. *IEEE Network*, 2025.
- [53] Chengfang Ling, Kai Peng, Shangguang Wang, Xiaolong Xu, and Victor CM Leung. A multi-agent drl-based computation offloading and resource allocation method with attention mechanism in mec-enabled iiot. *IEEE Transactions on Services Computing*, 2024.
- [54] Dingbang Liu, Fenghui Ren, Jun Yan, Guoxin Su, Wen Gu, and Shohei Kato. Scaling up multi-agent reinforcement learning: An extensive survey on scalability issues. *IEEE Access*, 2024.
- [55] Pengtao Liu, Kang An, Jing Lei, Yifu Sun, Wei Liu, and Symeon Chatzinotas. Computation rate maximization for scma-aided edge computing in iot networks: A multi-agent reinforcement learning approach. *IEEE Transactions on Wireless Communications*, 2024.
- [56] Xiao Liu and Yun Deng. A new qos-aware service discovery technique in the internet of things using whale optimization and genetic algorithms. *Journal of Engineering and Applied Science*, 71(1):4, 2024.

- [57] Yaju Liu, Siyuan Li, Xi Lin, Xiuzhen Chen, Gaolei Li, Yuchen Liu, Bolin Liao, and Jianhua Li. Qos-aware multi-aigc service orchestration at edges: An attention-diffusion-aided drl method. *IEEE Transactions on Cognitive Communications and Networking*, 11(2):1078–1090, 2025.
- [58] Tingyan Long, Yunni Xia, Mengchu Zhou, Jianqi Li, Yong Ma, and Yusuf Al-Turki. Fault-tolerant mobile service offloading in mobile edge computing. *IEEE Transactions on Automation Science and Engineering*, 2025.
- [59] Wenting Lu, Ayong Ye, Peixin Xiao, Yuanhuang Liu, Longjing Yang, Donglin Zhu, and Zhiquan Liu. Stones from other hills: Intrusion detection in statistical heterogeneous iot by self-labeled personalized federated learning. *IEEE Internet of Things Journal*, 2025.
- [60] Wenhao Ma and Hongzhen Xu. Skyline-enhanced deep reinforcement learning approach for energy-efficient and qos-guaranteed multi-cloud service composition. *Applied Sciences*, 13(11):6826, 2023.
- [61] Katarina Mandaric, Ana Keselj Dilberovic, and Gordan Jezic. A multi-agent system for service provisioning in an internet-of-things smart space based on user preferences. *Sensors*, 24(6):1764, 2024.
- [62] Bui Duc Manh, Chi-Hieu Nguyen, Dinh Thai Hoang, Diep N Nguyen, Ming Zeng, and Quoc-Viet Pham. Privacy-preserving cyberattack detection in blockchain-based iot systems using ai and homomorphic encryption. *IEEE Internet of Things Journal*, 2025.
- [63] Mahtab Masoori, Lata Narayanan, and Denis Pankratov. Renting servers in the cloud: Parameterized analysis of firstfit. In *Proceedings of the 25th International Conference on Distributed Computing and Networking*, pages 199–208, 2024.
- [64] Amin Mohajer, Abbas Mirzaei, Mostafa Darabi, and Xavier Fernando. Joint sla-aware task offloading and adaptive service orchestration with graph-attentive multi-agent reinforcement learning. *IEEE Transactions on Network and Service Management*, 2026.
- [65] Hajar Moudoud, Zakaria Abou El Houda, and Bouzian Brik. Reputation-aware scheduling for secure internet of drones: A federated multi-agent deep reinforcement learning approach. In *IEEE INFOCOM 2024-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2024.
- [66] Najmeh Nabavizadeh, Vahid Kayvanfar, and Majid Rafiee. A mixed integer linear programming model for quarantine-based home healthcare scheduling under uncertainty, healthcare analytics, volume 6, 2024, 100356, issn 2772-4425, 2024.
- [67] Rajganesht Nagarajan and Ramkumar Thirunavukarasu. Qos-aware quantum service composition using multi-objective optimization. *Quantum Machine Intelligence*, 8(1):24, 2026.
- [68] Masanobu Namiki, Yoshihiro Okawa, Natsuki Morita, Jun Kakuta, and Masatoshi Ogawa. Transfer learning with less negative transfer for multi-agent reinforcement learning: Application and evaluation in base station control. In *2024 IEEE 35th*

- [69] Vukan Ninkovic, Dejan Vukobratovic, Dragisa Miskovic, and Marco Zennaro. Com-split: A communication-aware split learning design for heterogeneous iot platforms. *IEEE Internet of Things Journal*, 2024.
- [70] Dhouha Ben Noureddine, Moez Krichen, Seifeddine Mechti, Tarik Nahhal, and Wilfried Yves Hamilton Adoni. An agent-based architecture using deep reinforcement learning for the intelligent internet of things applications. In *Advances on Smart and Soft Computing: Proceedings of ICACIn 2020*, pages 273–283. Springer, 2021.
- [71] Shengye Pang, Xinkui Zhao, Shuyi Yu, Jintao Chen, Shuiguang Deng, and Jianwei Yin. Trustpay: A dual-layer blockchain-based framework for trusted service transaction. *IEEE Transactions on Services Computing*, 2025.
- [72] Aditya Pathak, Irfan Al-Anbagi, and Howard J Hamilton. Blockchain-enhanced zero knowledge proof-based privacy-preserving mutual authentication for iot networks. *IEEE Access*, 2024.
- [73] Kulaea Taueveeve Pauu, Qianqian Pan, Jun Wu, Ali Kashif Bashir, Marwan Omar, et al. Irs-aided federated learning with dynamic differential privacy for uavs in emergency response. *IEEE Internet of Things Magazine*, 7(4):108–115, 2024.
- [74] Tao Qian, Mingyu Fang, Yongxu Zhu, Zeyu Liang, Yuxiong Huang, Qinran Hu, and Zaijun Wu. Asp-drl: A novel framework for unifying iots energy usage flexibilities characterized by neural networks and optimization models. *IEEE Internet of Things Journal*, 2024.
- [75] Pranab Sahoo, Ashutosh Tripathi, Sriparna Saha, and Samrat Mondal. Fedmrl: Data heterogeneity aware federated multi-agent deep reinforcement learning for medical imaging. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 640–649. Springer, 2024.
- [76] Adeb Salh, Mohammed A Alhartomi, Ghasan Ali Hussain, Hock Guan Goh, Nor Shahida M Shah, Saeed Alzahrani, Ruwaybih Alsulami, and Saad Alharbi. Enhancing energy-efficient power allocation for iot urllc through action space reduction in energy harvesting devices. *IEEE Access*, 2024.
- [77] Kruntarat Samngamnoi, Sutinun Pomsiri, Apipath Kamput, and Chanon Dechsupa. Ensuring iot controller reliability with colored generalized stochastic petri net. In *2024 28th International Computer Science and Engineering Conference (ICSEC)*, pages 1–6. IEEE, 2024.
- [78] Mohan Reddy Sareddy and Shakir Khan. Role-based access control, secure multi-party computation, and hierarchical identity-based encryption: Combining ai to improve mobile healthcare security. In *2024 International Conference on Emerging Research in Computational Science (ICERCS)*, pages 1–5. IEEE, 2024.
- [79] Shalini Sharma, Bhupendra Kumar Pathak, and Rajiv Kumar. Multi-objective service composition optimization in smart agriculture using fuzzy-evolutionary algorithm. In *Operations Research Forum*, page 43. Springer, 2024.

- [80] Shailendra Pratap Singh, Naweem Kumar, Gyanendra Kumar, Balamurugan Balusamy, Ali Kashif Bashir, and Maryam M Al Dabel. Enhancing quality of service in iot-wsn through edge-enabled multi-objective optimization. *IEEE Transactions on Consumer Electronics*, 2025.
- [81] P Sakthi Shunmuga Sundaram and K Vijayan. Neuro-fuzzy clustering and genetic optimization algorithm to enhance the quality of services in iot-enabled wireless sensor networks. In *2024 2nd International Conference on Networking and Communications (ICNWC)*, pages 1–7. IEEE, 2024.
- [82] Wei Tang, Chao Li, Jun Wu, and Qiuguo Zhu. Decentralized communication-maintained coordination for multi-robot exploration: Achieving connectivity and adaptability. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3417–3424. IEEE, 2024.
- [83] Zhengyi Tang, Yongbing Wu, Jinshui Wang, and Tianwei Ma. Iot service composition based on improved shuffled frog leaping algorithm. *Heliyon*, 10(7), 2024.
- [84] Sarah Hussein Toman, Lazhar Hamel, Zinah Hussein Toman, Mohamed Graiet, and Samir Ouchani. Formal modelling and verification of scalable service composition in iot environment. *Service Oriented Computing and Applications*, 17(3):213–231, 2023.
- [85] Asrin Vakili, Hamza Mohammed Ridha Al-Khafaji, Mehdi Darbandi, Arash Heidari, Nima Jafari Navimipour, and Mehmet Unal. A new service composition method in the cloud-based internet of things environment using a grey wolf optimization algorithm and mapreduce framework. *Concurrency and Computation: Practice and Experience*, 36(16):e8091, 2024.
- [86] Vineet Vishnoi, Ishan Budhiraja, Deepak Garg, Sahil Garg, Bong Jun Choi, and M Shamim Hossain. Multiagent deep reinforcement learning based energy efficient resource management scheme for ris assisted d2d users in 6g-aided smart cities environment. *Alexandria Engineering Journal*, 2025.
- [87] Janhvi Wadhwa, Aman Suthar, Deepanshi Joon, Prabhjot Kaur, et al. Enhanced weather forecasting with hybrid model of k means and dbscan. In *2024 Second International Conference on Advanced Computing & Communication Technologies (ICACCTech)*, pages 298–303. IEEE, 2024.
- [88] Chenxu Wang, Yonggang Jin, Cheng Hu, Youpeng Zhao, Zipeng Dai, Jian Zhao, Liuyu Xiang, Junge Zhang, and Zhaofeng He. Generalizable agent modeling for agent collaboration-competition adaptation with multi-retrieval and dynamic generation. *Neurocomputing*, page 130912, 2025.
- [89] Hongbing Wang, Mingzhu Gu, Qi Yu, Yong Tao, Jiajie Li, Huanhuan Fei, Jia Yan, Wei Zhao, and Tianjing Hong. Adaptive and large-scale service composition based on deep reinforcement learning. *Knowledge-Based Systems*, 180:75–90, 2019.
- [90] Li Wang, Yupeng Zhang, Yujing Hu, Weixun Wang, Chongjie Zhang, Yang Gao, Jianye Hao, Tangjie Lv, and Changjie Fan. Individual reward assisted multi-agent reinforcement learning. In *International conference on machine learning*, pages 23417–23432. PMLR, 2022.

- [91] Qi Wang, Jianhui Huang, and Yongjun Xu. Scheduling of real-time wireless flows: A comparative study of centralized and decentralized reinforcement learning approaches. *IEEE/ACM Transactions on Networking*, 2024.
- [92] Yihang Wang, Shengchao Su, and Yiwang Wang. Attention-augmented multi-agent collaboration for smart industrial internet of things task offloading. *Internet of Things*, 2025.
- [93] Yinfan Wang, Weihao Hu, Di Cao, Pengfei Zhao, Sayed Abulanwar, Zhe Chen, and Frede Blaabjerg. Local distribution voltage control using large-scale coordinated pv inverters: A novel multi-agent deep reinforcement learning-based approach. *IEEE Transactions on Smart Grid*, 2025.
- [94] Zhiyu Wang, Mohammad Goudarzi, Mingming Gong, and Rajkumar Buyya. Deep reinforcement learning-based scheduling for optimizing system load and response time in edge and fog computing environments. *Future Generation Computer Systems*, 152:55–69, 2024.
- [95] Ferry Wahyu Wibowo, Andi Sunyoto, Bayu Setiaji, et al. Analysis of round-robin and random scheduling algorithms on sensor node activation towards latency of wireless sensor network. In *2024 7th International Conference on Information and Communications Technology (ICOIACT)*, pages 304–309. IEEE, 2024.
- [96] Ziren Xiao, Peisong Li, Chang Liu, Honghao Gao, and Xinheng Wang. Macns: A generic graph neural network integrated deep reinforcement learning based multi-agent collaborative navigation system for dynamic trajectory planning. *Information Fusion*, 105:102250, 2024.
- [97] Chi Xu, Peifeng Zhang, Haibin Yu, and Yonghui Li. Dynamic blockchain-empowered trustworthy end-edge collaborative computing via rotating multi-agent drl. *IEEE Transactions on Wireless Communications*, 2025.
- [98] Kan Yi, Jin Yang, Shuangling Wang, Zhengtong Zhang, and Xiao Ren. Ppdrl: A pretraining-and-policy-based deep reinforcement learning approach for qos-aware service composition. *Security and Communication Networks*, 2022(1):8264423, 2022.
- [99] Jun Zeng, Juan Yao, Yang Yu, and Yingbo Wu. Multi-d3qn: A multi-strategy deep reinforcement learning for service composition in cloud manufacturing. In *International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pages 225–240. Springer, 2021.
- [100] Fan Zhang, Guangjie Han, Li Liu, Yu Zhang, Yan Peng, and Chao Li. Cooperative partial task offloading and resource allocation for iiot based on decentralized multiagent deep reinforcement learning. *IEEE Internet of Things Journal*, 11(3): 5526–5544, 2023.
- [101] Jing Zhang, Fei Shen, Liang Tang, Feng Yan, Fei Qin, and Chao Wang. A multi-agent reinforcement learning approach for dynamic offloading with partial information-sharing in iot networks. In *2023 IEEE 98th Vehicular Technology Conference (VTC2023-Fall)*, pages 1–5. IEEE, 2023.

- [102] Linchao Zhang, Lei Hang, Keke Zu, Yi Wang, and Kun Yang. Constructing next-generation iot security: Embedded smart contracts and multi-layer security protection. *IEEE Internet of Things Journal*, 2025.
- [103] Tianrong Zhang, Fan Wu, Zeyu Chen, and Senyang Chen. Optimization of edge cloud collaborative computing resource management for internet of vehicles based on multi-agent deep reinforcement learning. *IEEE Internet of Things Journal*, 2024.
- [104] Yawei Zhang, Lanxiang Chen, Yizhao Zhu, and Xiangqian Kong. Privacy-enhanced role-based access control for iot systems. *IEEE Internet of Things Journal*, 2025.
- [105] Huimin Zhuo, Lianglun Cheng, and Tao Wang. A deep reinforcement learning method for service composition in cyber-physical-social systems. In *2023 IEEE 14th International Conference on Software Engineering and Service Science (ICSESS)*, pages 47–54. IEEE, 2023.

Abstract

This thesis proposes a hierarchical multi-agent deep reinforcement learning (MADRL) framework for adaptive IoT service management in dynamic environments. The approach enables scalable coordination, QoS-aware service composition, and energy-efficient operation using a CTDE paradigm. Evaluations in a smart building scenario with up to 500 agents demonstrate fast convergence, 95.3% QoS satisfaction, and a 40% reduction in energy consumption compared to DRL baselines.

keywords: Internet of Things (IoT), Multi-Agent Systems, Deep Reinforcement Learning, Hierarchical Architecture, Service Composition, Quality of Service (QoS), Energy Efficiency, Resilience.

Résumé

Cette thèse propose un framework hiérarchique multi-agents basé sur le DRL pour la gestion adaptative des services IoT dans des environnements dynamiques. L'approche permet une coordination scalable, une composition de services sensible à la QoS et une optimisation énergétique via le paradigme CTDE. Les résultats, dans un scénario de bâtiment intelligent avec jusqu'à 500 agents, montrent une convergence rapide, 95,3 % de satisfaction QoS et une réduction de 40 % de la consommation énergétique.

mots-clés: Internet des Objets (IoT), systèmes multi-agents, apprentissage par renforcement profond, architecture hiérarchique, composition de services, qualité de service (QoS), efficacité énergétique, résilience.

مُلخَص

يقترح هذا العمل إطاراً هرمياً متعدد الوكلاء قائماً على التعلم العميق بالتعزيز لإدارة خدمات إنترنت الأشياء بشكل تكيفي في بيئات ديناميكية. يحقق النهج تنسيقاً قابلاً للتوسع، وتركيب خدمات يعتمد على جودة الخدمة، وكفاءة في استهلاك الطاقة باستخدام نموذج (CTDE). أظهرت النتائج في سيناريو مبنى ذكي يضم حتى ٥٠٠ وكيل تقارباً سريعاً، ونسبة رضا عن (QoS) بلغت ٩٥.٣٪، وتقليلاً بنسبة ٤٠٪ في استهلاك الطاقة.

إنترنت الأشياء ، الأنظمة متعددة الوكلاء، التعلم العميق بالتعزيز، البنية الهرمية، تركيب الخدمات، جودة الخدمة ، كفاءة الطاقة، المرونة.