Retrieval-Augmented Memory for Multi-Agent Deep Reinforcement Learning in LEO Satellite Routing

by

Xin Ming Li

A research paper
presented to the University of Waterloo
in fulfillment of the
research paper requirement for the degree of
Master of Mathematics
in
Computational Mathematics

Waterloo, Ontario, Canada, 2026

© Xin Ming Li 2026

Author's Declaration

I hereby declare that I am the sole author of this research paper. This is a true copy of the research paper, including any required final revisions, as accepted by my examiners.

I understand that my research paper may be made electronically available to the public.

Abstract

Efficient packet routing in low-Earth orbit (LEO) satellite constellations is complicated by a continuously changing topology, time-varying traffic and link conditions, and the difficulty of maintaining a current global view of the network at any single satellite. Multi-agent deep reinforcement learning (MADRL) offers a natural framework, with each satellite acting as an independent decision-maker under partial observability. However, existing MADRL methods decide from the current observation and their trained weights, so they cannot reuse past experience, even though orbital periodicity makes similar network situations recur. This research paper introduces RAMA, a retrieval-augmented MADRL framework with approximate information state (AIS) that gives each satellite a bounded memory of past situations, the actions taken in them and the delays their packets eventually experienced. At each decision, a Transformer trained with AIS-based auxiliary objectives reads the most similar memory entries together with the satellite's current observation. The routing problem is formulated as a decentralized partially observable Markov decision process with delayed feedback, and a new proposition establishes conditions under which RAMA's recent-history window and memory bank together satisfy the AIS properties.

RAMA is compared with the double deep Q-network (DDQN) routing method it extends and with minimum-hop shortest path. On the 140-satellite Kepler constellation with frozen weights, RAMA has 13.4% lower mean latency and 66.7% lower 99^{th}-percentile latency than DDQN, mostly from taking fewer long detours. Updated online, DDQN degrades while RAMA stays stable, and since frozen DDQN is also stable, the degradation comes from the online updates. Shortest path, which uses the full, current topology, keeps the lower mean, median and 99^{th}-percentile latency, but RAMA comes within 2.9% of its mean from local observations and has the lower 90^{th}-percentile latency. On the 1,584-satellite Starlink shell 1, RAMA's mean latency is 93.2% lower than DDQN's and falls by a further 25.5% with online updates, but with frozen weights it is still 2.04 times shortest path's. RAMA's advantage over DDQN therefore grows with constellation size.

Acknowledgments

I would like to thank my supervisor, Prof. Peng Hu, for his guidance and support throughout this project.

I would also like to thank Ken Cao and Mahyar Tajeri for their help and for many useful discussions.

I acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), funding reference number RGPIN-2022-03364.

Table of Contents

Author's Declaration ii

Abstract iii

Acknowledgments iv

List of Figures viii

List of Tables ix

List of Abbreviations x

1 Introduction 1

1.1 Low-Earth Orbit Constellations 2

1.2 Components of Packet Latency 3

1.3 The Case for Memory in Satellite Routing 3

1.4 Approximate Information States 6

1.5 Contributions 7

2 Background 8

2.1 Related Work 8

2.1.1 Routing in LEO Satellite Constellations 8

2.1.2 Memory Under Partial Observability . . . 10
2.1.3 Approximate Information State Theory . . . 11
2.1.4 Multi-Agent Reinforcement Learning Methods . . . 11
2.1.5 Positioning of This Work . . . 12

3 Methods **14**
3.1 System Model and Simulation Environment . . . 14
3.2 Problem Formulation with Delayed Feedback . . . 15
3.2.1 Action Space . . . 15
3.2.2 Local Observations . . . 16
3.2.3 The Delayed Feedback Channel . . . 16
3.2.4 Rewards and Objective . . . 18
3.3 RAMA Agent . . . 18
3.3.1 RAMA Architecture . . . 18
3.3.2 The Episodic Memory Bank . . . 21
3.3.3 Retrieval and the Bank-Update Operator . . . 22
3.3.4 Training . . . 22
3.3.5 Decision and Write Procedure . . . 23
3.4 Theoretical Analysis . . . 24
3.4.1 The Value-Approximation Bound . . . 25
3.4.2 The Policy-Suboptimality Bound . . . 26
3.4.3 The Discounted Infinite-Horizon Case . . . 26
3.4.4 RAMA's Compressed State Is an Approximate Information State . . . 27
3.5 Deployment Regimes . . . 28

4 Performance Evaluation **30**
4.1 Experimental Setup . . . 30
4.2 Baselines . . . 33

4.3 Comparisons Performed . . . 33
4.4 Two Corrections to the Simulation Environment . . . 34
4.5 Routing Performance . . . 35
4.5.1 Weight Stability . . . 35
4.5.2 RAMA Against the DDQN Predecessor . . . 38
4.5.3 RAMA Against Shortest Path . . . 38
4.6 Tracing the Slowest Packets . . . 41
4.6.1 RAMA's Slowest Packets Share a Destination . . . 41
4.6.2 DDQN's Slowest Packets Wander . . . 42
4.7 A Larger Constellation . . . 43
4.7.1 Scalability . . . 44
4.8 Long Paths and Mean Latency . . . 45

5 Discussion **47**
5.1 Where the Comparisons Place RAMA . . . 47
5.2 Trade-Offs . . . 48
5.3 Costs . . . 49

6 Conclusion **50**
6.1 Future Work . . . 51

References **53**

APPENDICES **59**

A Supplementary Material **60**
A.1 Integral Probability Metrics . . . 60
A.2 Approximate Information States . . . 61
A.3 Deferred Proofs . . . 61
A.4 Agent Hyperparameters . . . 64

List of Figures

1.1 Latency and Path Length over Time for DDQN and Dijkstra on Kepler . . 5

3.1 RAMA's Decision Network at One Satellite 19

4.1 Latency and Path Length over Time for Frozen Conditions on Kepler . . . 37

4.2 Complementary Cumulative Distribution of Latency 40

List of Tables

3.1 Input Tokens of the RAMA Network 20

4.1 Constellation Parameters 30
4.2 Simulation Configuration on Kepler 31
4.3 Routing Performance on Kepler 36
4.4 Per-Hop Delay and Latency by Path Length on Kepler 39
4.5 Aggregate Results on Starlink Shell 1 43
4.6 Mean Latency Split Between Long and Other Paths 46

A.1 Agent Hyperparameters 66

List of Abbreviations

AIS Approximate Information State

DDQN Double Deep Q-Network

Dec-POMDP Decentralized Partially Observable Markov Decision Process

DRL Deep Reinforcement Learning

IPM Integral Probability Metric

KSAT Kongsberg Satellite Services

LEO Low-Earth Orbit

LSTM Long Short-Term Memory

MADRL Multi-Agent Deep Reinforcement Learning

MARL Multi-Agent Reinforcement Learning

RAMA Retrieval-Augmented Multi-Agent Routing with Episodic Memory

ReLU Rectified Linear Unit

Chapter 1

Introduction

Low-Earth orbit (LEO) satellite constellations can bring high-quality Internet access to rural and remote areas [1]. To do so, the data packets need to travel through a network of moving satellites, which raises the problem of finding suitable paths through a continuously changing topology. The problem is shaped by constraints not present in terrestrial networking. A LEO satellite, considered as a node in a network, moves at roughly 7.6 km/s, the link topology reconfigures continuously as satellites move relative to one another and to the ground, and it is difficult for a single node to maintain a current view of the global network state. Dijkstra's algorithm is a traditional approach to routing formulated as a shortest-path problem [2], but it needs a global view of the network topology, which becomes costly to obtain as the number of satellites grows, and a topology computed in advance cannot anticipate unplanned link or satellite failures. Learning-based routing, in which an agent is trained to make routing decisions from either local or global observations, has consequently emerged as an active research direction [3–5]. Beyond routing, Hu [6] was the first to discuss the use of artificial intelligence for operational tasks in proliferated satellite networks, and proposed SatAIOps, an approach covering their full life cycle.

This research paper develops and analyzes RAMA, a retrieval-augmented multi-agent deep reinforcement learning (MADRL) framework with approximate information state (AIS), whose defining feature is a per-satellite episodic memory. Each agent has its own bounded bank of previously encountered situations, the actions taken in them and the delivery outcomes those actions eventually produced, and a small Transformer reads the entries retrieved from it together with the current observation, a window of the satellite's recent decisions and its neighbors' state. The central premise is that orbital mechanics make the satellite routing problem repetitive when observed over long periods of time. Constellation geometry is periodic and traffic demand follows a daily pattern, so a routing

agent that can recall how a similar situation resolved in the past holds a structural advantage over one that reacts only to the present. The paper places this idea on a formal footing using the AIS framework [7, 8], which bounds the loss of optimality incurred by acting on a compressed summary of history in terms of that summary's reward errors and prediction errors.

1.1 Low-Earth Orbit Constellations

Satellites are grouped by orbital altitude, and the choice of altitude implies a trade-off between how much of the Earth one satellite can see, how long a signal takes to reach it and how much it costs to send the satellite into orbit. Lower orbits imply lower latency and cheaper launch, build and maintenance at the expense of coverage per satellite [2].

LEO satellites span altitudes from about 200 km to 2,000 km above the surface [9]. A satellite near 550 km completes one revolution in roughly 96 minutes and travels at about 7.6 km/s [10], so the one-way path to the ground takes light only a few milliseconds, which permits smaller antennas and lower transmit power. However, a LEO satellite sees only a small patch of the surface and stays in view of a given user for a few minutes at most before a handover is required, so continuous service demands a large number of satellites and constant reassignment of users between them.

Higher orbits, from medium-Earth orbits between 8,000 km and 20,000 km up to the geostationary orbit at 35,786 km [9], need far fewer satellites for global coverage and keep each satellite in view for hours, but at a one-way propagation delay of roughly 27 ms to 120 ms. Satellites in these orbits can also be networked together across layers, and Hu [11] proposed a multi-layer scheme that relays telecommand and telemetry messages through LEO, medium-Earth-orbit and geostationary satellites. Hu [12] also proposed Cross-Layer Descent, a cross-layer routing approach for multi-orbit satellite networks, which improved resilience and latency over classical schemes in simulation. This paper considers routing within a single LEO layer.

A constellation is a coordinated group of satellites operated as a single system. Kepler had launched 33 satellites as of January 2026, when it deployed the first tranche of its optical data relay constellation [13]. Its original design, announced in 2016 and authorized by the U.S. Federal Communications Commission in 2018, is a 140-satellite polar constellation between 500 km and 650 km, spread across seven planes with 20 satellites each [14, 15], and that is the main constellation geometry used in this paper. Starlink is by far the largest, having passed 11,000 satellites in orbit [16]. Its primary shell, Starlink

shell 1, has 1,584 satellites in 72 planes at 550 km, inclined at 53°, and it is the second constellation geometry used in this paper.

The experiments in this paper use the Kepler geometry for the full evaluation and repeat it on Starlink shell 1. With 140 planned satellites, Kepler is large enough that multi-hop routing decisions are non-trivial, yet small enough that repeated simulation-based training remains feasible. Kepler's explicit focus on inter-satellite communication also makes it a natural benchmark for routing algorithms. Starlink shell 1 has more than ten times as many satellites at a much lower inclination, so it tests whether the results carry over to a larger constellation, although its simulation cost is far higher.

1.2 Components of Packet Latency

The primary performance metric in this work is end-to-end latency, the total delay a packet experiences between creation at its source ground station and arrival at its destination. On each hop, this delay decomposes into three components. Queueing latency is the time a packet waits in a satellite's buffer before transmission begins, and under congestion, it dominates all other components. Transmission latency is the time needed to put the packet's bits onto the link, scaling with packet size. Propagation latency is the time the signal needs to travel from one end to another at the speed of light. It is entirely dependent on a link's physical length, but over a full path it depends on route length, so poor routing inflates it through unnecessary detours.

Routing therefore influences latency through two channels, path geometry and load placement. A routing policy that reacts only to recent local congestion can control neither channel well when congestion develops too quickly. This observation motivates the memory-based approach developed in this paper.

1.3 The Case for Memory in Satellite Routing

The starting point for this work is an empirical observation about the behavior of an existing learned routing system. Lozano-Cuadra *et al.* [17] developed a decentralized MADRL routing framework in which every satellite runs a double deep Q-network (DDQN) policy over local observations, and released an open-source simulator implementing it. Their results show the learned policy performing competitively with Dijkstra's shortest-path algorithm under moderate traffic.

Figure 1.1 shows what that competitiveness looks like when it is tracked over time at full ground-segment load. Lozano-Cuadra *et al.* [3] do evaluate eight active ground stations, but plot that case over time only for the 4 s offline training phase, report the online phase as a latency distribution, and use a scenario with only two ground stations, Málaga and Los Angeles, for their time-resolved analysis of satellite movement. The runs shown here keep all eight ground stations active and follow the result across one simulated hour at realistic satellite speed. Shortest-path routing holds close to its overall mean of 6.80 hops throughout, rising by 3.9% across the hour, which bounds how much of any drift constellation geometry and traffic load can explain, since every policy sees the identical network at the identical times.

DDQN online is the baseline in its online phase, in which each satellite runs its own copy of the trained network and continues local training from its own replay buffer, with exploration disabled. It averages 186 ms and 27.04 hops over the run, against 45 ms and 6.80 hops for Dijkstra's shortest path. The shape matters more than the average. It holds with the other two for the first few hundred seconds, then separates, reaching roughly 0.48 s and 66 hops near $t = 1{,}450$ s, and it never comes back. For the rest of the hour it stays between about 0.10 and 0.46 s, with further peaks near 1,750 s and 2,250 s, and mean latency and hop count follow this pattern together.

DDQN frozen is the control that identifies the mechanism, and it was not part of the original observation. It is the same checkpoint, the same traffic and the same topology, with the training routine switched off. It stays with Dijkstra's shortest path, a little above it and rising at the same rate. Comparing the first and last 300 s of the run, its mean latency goes from 52.0 to 57.8 ms, a rise of 11.1%, against 11.4% for Dijkstra's shortest path over the same two windows, and its mean path length goes from 8.11 to 8.90 hops. The degradation of DDQN online is therefore a consequence of updating weights during deployment and not of the policy's memorylessness, because a policy with no memory at all holds its ground once its weights are fixed.

The drift therefore does not show the need for memory. It had originally been read as evidence for two separate claims, however, and both still hold without it.

The first is that the baseline's learning signal is shortsighted, and this remains true independently of Figure 1.1. The reward is assembled only from observations at the moment of the decision, namely the packet's wait in the deciding satellite's queue, the slant-range progress the hop makes toward the destination and a penalty when a packet revisits a satellite it has already passed through. A series of hops can reduce slant range and find an empty queue while the path as a whole lengthens, and the delay the packet eventually accumulates never enters the reward. An agent whose learning signal carries the realized

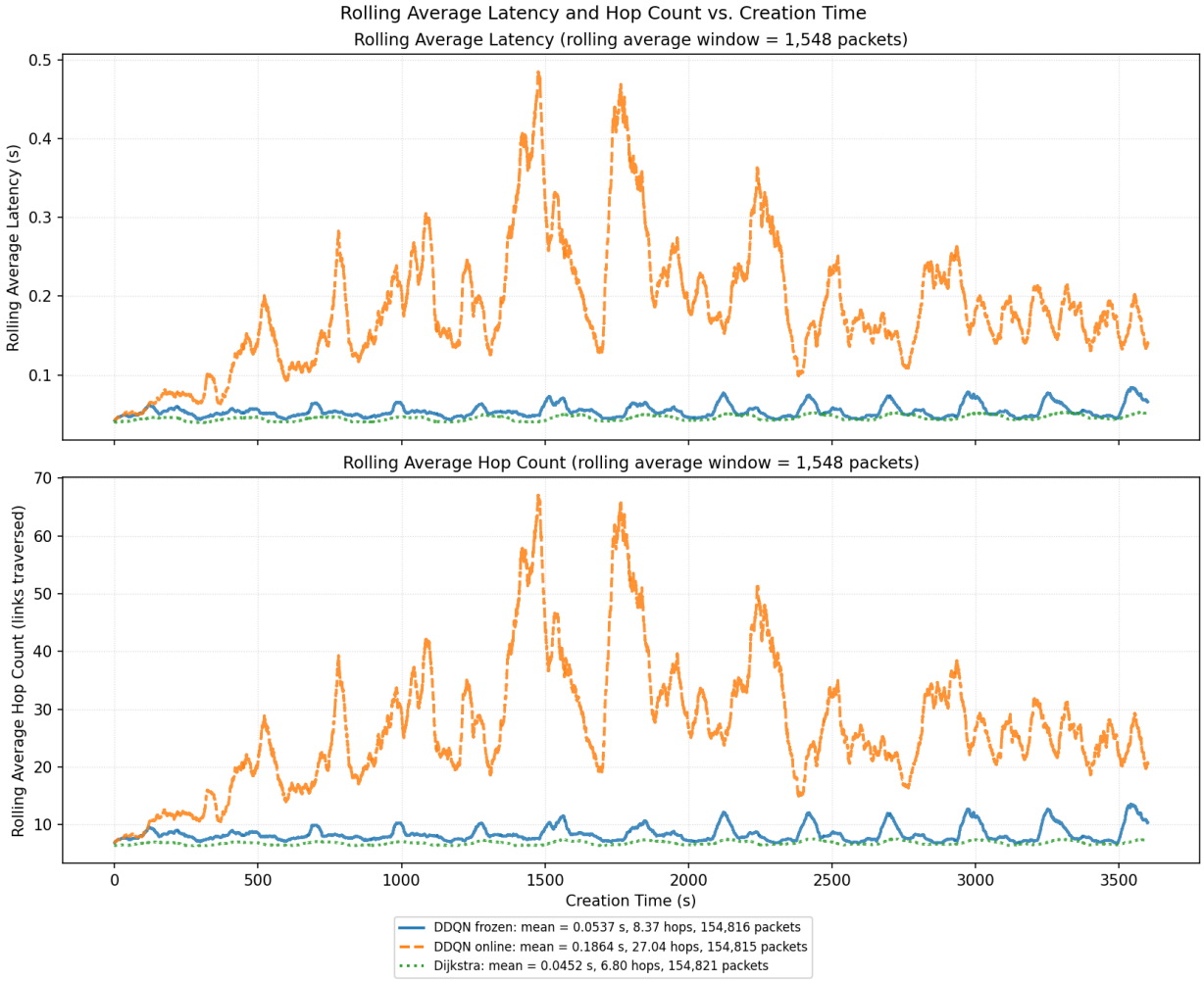


Figure 1.1: Rolling-average end-to-end latency (top) and rolling-average path length (bottom) against packet creation time on Kepler, over 3,600 s with eight ground stations, on one shared time axis and one smoothing window of 1,548 packets. DDQN online continues on-board training during the run, DDQN frozen is the same checkpoint with training switched off, and Dijkstra's shortest path uses the identical topology and traffic.

downstream delay is being told something the baseline is never told, whether or not that changes the drift.

The second is that the network topology and its traffic load are repetitive, so the same local situation recurs at predictable intervals across a deployment. A policy that conditions only on the present cannot use that structure, and a policy whose weights are frozen for stability, which Figure 1.1 shows is the safer way to deploy one, has no other channel through which past experience could reach a present decision. Freezing the weights makes the case for an explicit memory stronger rather than weaker, because it removes the alternative.

Recurrent architectures such as long short-term memory (LSTM) networks [18, 19] address the shortsightedness but not the recurrence, since in practice they are trained by backpropagation through a short window of recent steps, the last ten in the original deep recurrent Q-network [19], which gives them no means to carry a specific routing outcome across the full deployment. The recurrence structure of the problem calls for episodic memory, an explicit and queryable bank of past experiences.

1.4 Approximate Information States

Giving an agent memory raises a theoretical concern. When an agent acts on a compressed summary of its history instead of the full history, how much performance can be lost? The AIS framework of Subramanian *et al.* [7,8] answers this question in a precise form. An AIS is a compressed state that satisfies two properties. It suffices to predict the immediate reward approximately (property AP1), and it suffices to predict its own evolution approximately (property AP2). For any compression satisfying these properties with errors (ε, δ), the framework yields explicit bounds on both the error of value functions computed from the compressed state and the suboptimality of policies that act greedily on it.

The framework is attractive here for two reasons. It is prescriptive, since the two properties translate directly into auxiliary training losses. It is also independent of how the compression is computed, which allows a retrieval-augmented memory bank to be brought within the same analysis provided its update dynamics satisfy the required structure. Establishing that RAMA's compressed state satisfies this structure is one of the main theoretical contributions of this paper.

1.5 Contributions

This paper makes three contributions.

1. **The RAMA architecture.** Each satellite maintains a bounded, outcome-indexed episodic memory bank whose entries are written in hindsight from delayed feedback, retrieved by similarity at decision time, and read by a small Transformer alongside a history window and neighbor tokens, which is trained with AIS auxiliary losses. Every design choice is motivated by the theory or by the deployment constraints. Those choices are hindsight outcome labels, a slowly updated key encoder, a deterministic bank-update operator and permutation-invariant treatment of retrieved memories.

2. **The AIS analysis.** The routing problem is formulated as a decentralized partially observable Markov decision process (Dec-POMDP) [20] that extends the partially observable formulation of Lozano-Cuadra *et al.* [3] with a delayed feedback channel, so that each satellite eventually observes the realized downstream delay of packets it forwarded. Since the feedback is part of the observation process, the problem remains a legitimate Dec-POMDP and the AIS framework applies without modification. The value-approximation and policy-suboptimality bounds of Subramanian *et al.* [8] for AIS-based agents are then restated with full proofs. A proposition establishes conditions under which RAMA's pair of recent-history window and memory bank satisfies the AIS properties, so the bounds transfer to the memory-augmented agent.

3. **A 2×2 evaluation.** RAMA is evaluated against the DDQN routing method and Dijkstra's shortest path under matched traffic, with the learned agents' weights either frozen or updated online, on the Kepler constellation and on Starlink shell 1.

Chapter 2

Background

This chapter situates the paper within four bodies of literature and states the gap this work fills. Those literatures are learned routing for satellite constellations, memory mechanisms in reinforcement learning, AIS theory and multi-agent training paradigms.

2.1 Related Work

2.1.1 Routing in LEO Satellite Constellations

Existing LEO routing methods can be broadly divided into shortest-path approaches that rely on explicit network models and deep reinforcement learning (DRL)-based approaches that learn forwarding policies from interaction with the environment. While DRL methods improve adaptability, historical information is typically encoded in recurrent states, topology histories or continually updated network parameters. As a result, successful forwarding experiences cannot be explicitly retrieved when similar network conditions recur, a limitation that matters in LEO constellations, where orbital geometry and traffic patterns recur.

Before learning-based methods, constellation routing was addressed with graph algorithms over snapshot or time-expanded topologies. Dijkstra's algorithm [21] is the classical choice among them, and the survey of Çelikbilek *et al.* [2] covers the broader optimization landscape for LEO networks. Snapshot methods are optimal when global state is known, but maintaining global state in real time becomes costly as constellations grow, and static link costs cannot express stochastic queueing behavior.

The line of work this paper builds on directly is that of Lozano-Cuadra *et al.* They released an event-based simulator of LEO packet routing [17] that models packet creation, queueing, transmission and orbital motion for a configurable constellation, and supports shortest-path, tabular Q-routing and MADRL policies. Their MADRL routing method [3] models each satellite as an agent that selects the next hop using its position, the destination and neighboring queue states. It formulates decentralized routing as a partially observable problem in which every satellite runs a DDQN policy, trained first offline against a global model and then continually online on board. This method is the baseline here. Its formulation and reward structure are retained, and RAMA's observation extends the baseline's with direction scores, as Chapter 3 states formally. The important point for positioning is that the resulting policy is reactive. It decides from the current observation alone, with no mechanism for recalling past network conditions, and it adapts online only by updating its weights.

Several groups have since enriched the per-agent model. Chen *et al.* [4] proposed Transformer-MIX, a routing framework based on QMIX [22] for a 66-satellite Iridium-like constellation, in which each agent combines a gated recurrent unit with self-attention over its current observation, previous action and agent identifier. Its mixing network generates the weights that combine the agents' values during centralized training with a Transformer-based mechanism taken from TransfQMix [23]. Lyu *et al.* [5] proposed a constrained multi-agent reinforcement learning dynamic routing (CMADR) algorithm for integrated satellite-terrestrial networks, in which satellites and ground stations are jointly optimized to minimize packet delay subject to energy and packet-loss constraints using a Lagrangian max-min formulation with centralized reward and cost critics, and evaluated it on OneWeb and Telesat. He *et al.* [24] handled ultra-dense constellations with asynchronous, event-driven decisions and a risk-aware objective. In GNN-ASSSP, He *et al.* [10] instead computed end-to-end routes from a network-wide view by predicting link availability from recent topology snapshots with a spatiotemporal Transformer and learning congestion-aware edge weights for shortest-path search using a graph attention network. Transformer-MIX and CMADR choose joint actions in synchronized time steps, which per-packet, event-driven routing does not provide, and GNN-ASSSP relies on a network-wide view. A parallel thread applies large language models to network control, to tune the hyperparameters of DRL agents [25] or as agentic operations frameworks [26]. These target orchestration rather than per-packet forwarding.

Although these methods exploit historical information, they do so through recurrent hidden states [4], topology histories [10] or online parameter updates [3]. None maintains a per-satellite memory of forwarding decisions together with their realized end-to-end delivery outcomes.

2.1.2 Memory Under Partial Observability

Memory mechanisms in reinforcement learning generally fall into two categories, parametric memory, in which past information is compressed into network parameters or recurrent hidden states, and non-parametric memory, in which individual experiences are explicitly stored and later retrieved. Both have been used to improve decision making under partial observability and non-stationarity. The classical way to handle partial observability is to keep a belief state, which is a probability distribution over what the hidden state could be given everything the agent has seen so far. A belief state contains all the information needed to act optimally, but computing it requires a model of the environment and becomes impractical for large problems [20]. In practice, DRL methods replace the belief state with a learned recurrent summary. Hausknecht and Stone [19] proposed deep recurrent Q-networks, which replace a fully connected layer of a deep Q-network [27] with an LSTM layer [18] so that history is summarized in a hidden state trained over short unrolled sequences. Such recurrent summaries extend the effective observation window but degrade over long horizons and cannot recall a specific, temporally distant event on demand.

Non-parametric approaches address exactly that limitation by keeping individual experiences explicitly. Blundell *et al.* [28] proposed model-free episodic control, which stores the best return observed for each state-action pair and estimates values from nearby stored experiences. Pritzel *et al.* [29] extended this idea through neural episodic control, which learns state embeddings and stores them in differentiable memory structures indexed by action. In both methods, retrieved memories directly produce value estimates, and both were developed for single-agent games. More recently, the retrieval-augmented generation paradigm from language modeling [30] has been imported into reinforcement learning, where agents retrieve from much larger external stores. Humphreys *et al.* [31] conditioned a Go-playing agent on similar states retrieved from a large corpus of expert games, and Schmied *et al.* [32] proposed the Retrieval-Augmented Decision Transformer, which retrieves sub-trajectories from previously collected experience and incorporates them into in-context decision making. These two works are the closest architectural precedents for RAMA and demonstrate the value of explicit experience retrieval, but their stores are large and serve single-agent tasks outside networking, so they are not designed for decentralized packet routing in large-scale satellite networks. RAMA's store is small, private to each satellite, written as its packets are delivered or dropped, and read by the network together with the current observation. To the best of the author's knowledge, no prior work provides multi-agent packet routing with retrieval-augmented episodic memory, and none of the episodic memory literature connects the memory mechanism to AIS guarantees.

2.1.3 Approximate Information State Theory

Subramanian and Mahajan [7] defined an AIS as a compression of history that approximately predicts the immediate reward and its own evolution, as an alternative to belief states for partially observed control, and bounded the error of the dynamic program built on it. In the journal version, Subramanian, Sinha, Seraj and Mahajan [8] developed the framework in full generality. They measured prediction error with integral probability metrics (IPMs), proved bounds on the value error and the suboptimality of AIS-greedy policies, and used the errors as reinforcement learning losses in an algorithm that learns the compression and the policy jointly. The framework subsumes belief quantization and ε-sufficient statistics [33] as special cases.

Kao and Subramanian [34] extended the construction to multi-agent settings through the information common to all agents. They split histories into common and private components, compressed them separately, and showed that a dynamic program over the compressed representations is near-optimal, with an optimality gap expressed in the compression error parameters. Their result is what licenses the use of AIS-style compression by decentralized agents trained under centralized information. To the best of the author's knowledge, the framework has not been applied to packet routing or combined with a non-parametric memory.

2.1.4 Multi-Agent Reinforcement Learning Methods

Cooperative multi-agent reinforcement learning (MARL) under partial observability is commonly trained with centralized training and decentralized execution, in which a centralized learner may access global information during training while executed policies depend only on local observations [20]. The two dominant families of this paradigm are value decomposition and centralized critics. The first is represented by QMIX [22], which learns per-agent utilities combined by a monotonic mixing network and is the scheme adopted by Transformer-MIX [4]. The second is represented by multi-agent proximal policy optimization (MAPPO) [35], which trains decentralized actors against a centralized value function. Both introduce auxiliary networks that exist only at training time and whose fidelity limits the learned policies.

Group relative policy optimization (GRPO), introduced for language-model fine-tuning [36], removes the learned critic altogether. Multiple responses are sampled for the same prompt, and each response's advantage is computed relative to the empirical statistics of its own group. Liu *et al.* [37] lifted this recipe to cooperative multi-agent problems as multi-agent GRPO (MAGRPO), sampling groups of joint rollouts for the same

task instance and updating all agents with group-relative advantages, and demonstrated it on large-language-model collaboration benchmarks. Sun *et al.* [38] surveyed the broader intersection between language models and MARL. The routing setting is well suited to this recipe, because the simulator can replay the same traffic scenario under different policy randomness, which is the group structure GRPO requires. That direction is not pursued in this paper.

One further line deserves mention for positioning. Pavel *et al.* [39] studied resource-aware knowledge distillation in MARL with KD-MARL, compressing a centralized teacher into lightweight per-agent students. Since KD-MARL is close prior art for the distillation mechanism, distillation is given no load-bearing role in this paper. It appears only as an optional deployment-time compression step in Chapter 6, and the novelty claims rest elsewhere. Zhao *et al.* [40] studied order-independent Transformer encoders for multi-agent problems, and the permutation-equivariance property they relied on reappears in Chapter 3 as the justification for leaving retrieved memories without positional encodings. Outside networking, multi-agent pathfinding under local observability includes PRIMAL [41].

2.1.5 Positioning of This Work

The gap this work occupies lies at the intersection of the first three literatures above. Existing satellite-routing agents have no episodic memory, and existing AIS theory has never been asked to accommodate a non-parametric, growing memory store. RAMA differs from prior routing and memory-based approaches in four respects. First, each satellite maintains a small, private memory of its own forwarding experiences, labeled retrospectively with the realized delivery outcomes and retrieved by similarity search through a slowly updated key encoder. Second, retrieved memories are combined with current observations, recent routing history and neighboring state information through a Transformer-based decision network rather than directly producing value estimates. Third, the policy is trained using auxiliary objectives from the AIS framework, which, to the best of the author's knowledge, has not previously been applied to packet routing or combined with retrieval-based memory. Finally, the memory remains active after training, allowing deployed agents to continue recording and retrieving experiences even when policy weights are frozen, whereas the baseline adapts online only by updating its weights. To the best of the author's knowledge, no prior LEO routing framework combines decentralized packet forwarding, per-satellite retrieval-based episodic memory and AIS-based representation learning within a unified MARL architecture.

The claim to novelty rests on the first two contributions of Section 1.5. The first is the RAMA architecture built from these choices, a retrieval-augmented, outcome-indexed

episodic memory operated per satellite in a multi-agent routing system. The second is the AIS analysis showing that the resulting compressed state satisfies the AIS properties, so the value-approximation and policy-suboptimality bounds transfer to it. Retrieval (Section 2.1.2) and AIS (Section 2.1.3) each exist on their own, but to the best of the author's knowledge their combination, and their application to satellite routing, do not appear in prior work. This paper evaluates RAMA as a whole and does not isolate the contribution of any one of these design choices.

Chapter 3

Methods

3.1 System Model and Simulation Environment

The physical model follows the open-source simulator of Lozano-Cuadra *et al.* [17] and its companion formulation [3]. This section restates the elements needed later and records the conventions used throughout.

The constellation consists of N satellites evenly distributed over O circular orbital planes, and satellites are indexed $i \in [N]$. Each satellite carries one antenna for ground-to-satellite links, two roll-axis antennas for intra-plane inter-satellite links with the fore and aft neighbors, and two pitch-axis antennas for inter-plane inter-satellite links, giving at most four inter-satellite neighbors. Each satellite maintains one first-in first-out transmit buffer per neighbor, and packets arriving at a full buffer are dropped. Ground stations connect users on the ground to the constellation, each through a link to its nearest satellite, and every packet travels from a source ground station to a destination ground station across inter-satellite links. Each ground station's demand derives from the population within 500 km of it, and it aggregates user data into blocks of B bits generated by a Poisson process and sends equal traffic to every other ground station [3]. The data rate $R(i, j)$ of every link, inter-satellite and ground alike, is the bandwidth multiplied by the highest tabulated spectral efficiency that the received signal-to-noise ratio supports [3]. The source tabulates the spectral efficiencies of the second-generation Digital Video Broadcasting by Satellite standard, and the released simulator uses the larger table of that standard's extensions. Table 4.2 records the radio parameters.

A packet of B bits crossing the link from satellite i to satellite j incurs the latency

$$T(i,j) = t_q(i) + \frac{B}{R(i,j)} + \frac{\|ij\|}{c}, \tag{3.1}$$

where $t_q(i)$ is its wait time in satellite i's queue, $R(i,j)$ is the link rate, $\|ij\|$ is the slant range between satellites i and j, and c is the speed of light. The three terms are the queueing, transmission and propagation components introduced in Section 1.2. The end-to-end latency T_p of packet p is the sum of (3.1) over every link it traverses.

Table 4.2 gives the concrete values used for the reported Kepler runs, covering constellation geometry, ground-station set, simulated duration and topology update interval.

The simulator advances the constellation, rebuilds the topology graph and rebuilds each satellite's transmit-buffer processes at a fixed interval, `movementTime`, which Table 4.2 gives. Total orbital motion over a run is the product of the simulated duration and the per-second orbital advance, so it is independent of this interval, which controls only how finely the motion is discretized.

3.2 Problem Formulation with Delayed Feedback

Routing is formulated as a Dec-POMDP. The agents $\mathcal{I} = [N]$ are the satellites, the hidden global state S_t comprises all satellite positions, buffer contents, in-flight packets and link availabilities, and each agent observes only a local slice of it. The formulation below keeps the action and reward design of Lozano-Cuadra *et al.* [3], extends their observation with the direction in which the satellites involved are moving, and adds a delayed feedback channel.

3.2.1 Action Space

For a packet p at satellite i making its n^{th} hop, with d the satellite linked to p's destination ground station, the agent on i selects the next hop $a_{i,n} \in \mathcal{A}_i = \{\mathrm{U}_i, \mathrm{D}_i, \mathrm{R}_i, \mathrm{L}_i\}$ of p. The neighbors of satellite i above and below it in its orbital plane are U_i and D_i, respectively. Similarly, the neighbors of i to its right and left in the adjacent planes are R_i and L_i. The permitted set of actions $\mathcal{A}_{i,n}$ is the subset of the action space $\mathcal{A}_i$ containing the neighbors of i except, where possible, the one p came from, and excluding any neighbor that p has visited five times or more. This is a loop guard to prevent cycling, and if it excludes every

neighbor, the least-visited linked neighbor is selected. If $a_{i,n} \notin \mathcal{A}_{i,n}$, then $a_{i,n}$ is removed from $\mathcal{A}_i$ and the agent makes another decision. If d is a neighbor of i, then the agent is bypassed and p is routed to d. Once p reaches d it is delivered over the ground link to its destination ground station [3].

3.2.2 Local Observations

Building on the observation of Lozano-Cuadra *et al.* [3], the RAMA agent on satellite i observes the 31-dimensional vector

$$\begin{aligned} o_n &= \left[v_d, v_{\mathrm{R}_i}, v_{\mathrm{L}_i}, \mathbf{b}_{\mathrm{U}_i}, \mathbf{b}_{\mathrm{D}_i}, \mathbf{b}_{\mathrm{R}_i}, \mathbf{b}_{\mathrm{L}_i}, \bar{\phi}_i, \bar{\lambda}_i, \bar{\phi}_d, \bar{\lambda}_d\right], \\ \mathbf{b}_j &= \left[C_{j,\mathrm{U}_j}, C_{j,\mathrm{D}_j}, C_{j,\mathrm{R}_j}, C_{j,\mathrm{L}_j}, \bar{\phi}_j, \bar{\lambda}_j\right], \quad j \in \mathcal{A}_i, \end{aligned} \tag{3.2}$$

where $\mathbf{b}_j$ holds the position of neighbor j and the congestion scores $C_{j,k}$ of j's queues toward j's neighbor k. Positions are latitudes ϕ and longitudes λ in whole degrees, normalized as $\bar{\phi} = (\phi + 90)/\sigma$ and $\bar{\lambda} = (\lambda + 180)/\sigma$ with $\sigma = 20$. The direction score of satellite j is $v_j = -\sin\psi_j$, where ψ_j is j's angle within its orbital plane measured from the plane's northernmost point, so v_j is positive while j moves north and negative while it moves south. The congestion score[1] of satellite j's queue for j's neighbor k is

$$C_{j,k} = \min\left(C^*, \left\lfloor C^* \frac{\log_{10}(q_t + 1)}{\log_{10} q_{\max}} \right\rfloor\right), \tag{3.3}$$

where $C^* = 10$ is the maximum congestion value, q_t is the queue length in packets at time t, and $q_{\max}$ is a normalizing queue length [3]. A queue toward a missing satellite is treated as infinitely long and scores C^*, so a missing neighbor has all four congestion scores at C^* and its coordinates at 0 [3].

3.2.3 The Delayed Feedback Channel

The other addition to the baseline formulation is a feedback field in each observation. When agent i forwards a packet p, the downstream cost of that decision is the delay the packet actually accumulates from i onward until delivery, and that quantity is unknown

[1] Lozano-Cuadra *et al.* [3] wrote this formula with max instead of min. With max, every queue with length less than $q_{\max}$ would score C^*, so no queue shorter than $q_{\max}$ could be differentiated from a full one. The released code caps the score at C^*, which matches the min form used here.

when the decision is made. It becomes known only when the packet is delivered or dropped, and the realized outcome of the decision is

$$g = \begin{cases} t_{\text{del}} - t_{\text{dec}}, & \text{if the packet is delivered,} \\ 0.6 \text{ s}, & \text{if it is dropped,} \end{cases} \tag{3.4}$$

where t_{dec} is the decision time and t_{del} is the delivery time. In a deployed constellation the outcome would have to be reported back to i, for example over telemetry piggybacked on acknowledgment traffic. Delayed feedback of this kind is inherent to large constellations, and the related asynchrony of routing decisions has been modeled explicitly in recent work [24]. In the simulator the report reaches i at the moment the packet is delivered or dropped.

Formally, the observation of agent i at its decision at time t is extended to $y_t^i = (o_t^i, F_t^i)$, where o_t^i is the observation (3.2) for the packet being routed and F_t^i is the (possibly empty) set of outcome reports delivered to i since its previous decision. Each report is a tuple

$$f = (\kappa, a, g, \tau), \tag{3.5}$$

where κ and a are the key of the observation (Section 3.3.3) and the action that agent i recorded at the earlier decision, g is its outcome (3.4), and τ is the time at which the packet was delivered or dropped. Since the outcome is generated by the environment as a function of the underlying state trajectory, F_t^i is part of agent i's observation, like o_t^i. Following the usual convention in reinforcement learning, the rewards of the agent's past decisions are taken to be observed by it, and they fill RAMA's history tokens (Section 3.3.1). The current time, which sets the staleness of retrieved entries (Section 3.3.3), is also known to the agent. Both are taken to be part of y_t^i. The process remains a legitimate Dec-POMDP, and every result that holds for Dec-POMDP observation histories applies to y_t^i without modification, the AIS machinery of Section 3.4 included. Packets still in flight when an episode ends simply never generate reports, which is harmless.

The two indices in this chapter count different things. The decision index n of Section 3.2.1 counts the hops of a packet, whereas t orders the decisions of one satellite in time. A decision of agent i at time t on a packet making its n^{th} hop has $o_t^i = o_n$.

3.2.4 Rewards and Objective

The reward for an agent's decision $a_{i,n}$ to forward p from i to j is

$$r_n = w_1\big(1 - 10^{\,t_q(i)}\big) + w_2\,\frac{\|id\| - \|jd\| - \|ij\|/w_4}{d_{\max}} + w_3\big(r_l\,\mathbb{I}[j \in \mathcal{P}] + r_d\,\mathbb{I}[j = d] + r_u\,\mathbb{I}[a_{i,n} \notin \mathcal{A}_{i,n}]\big), \tag{3.6}$$

where $\mathbb{I}[\cdot]$ is the indicator, $t_q(i)$ is in seconds, $d_{\max}$ is the longest inter-satellite link, and $\mathcal{P}$ is the set of satellites the packet visited before i. The weights are $w_1 = w_2 = 20$, $w_3 = 1$ and $w_4 = 5$. The term $r_l = -10$ penalizes a revisit, $r_d = 50$ rewards reaching d, and $r_u = -10$ penalizes an attempted hop outside $\mathcal{A}_{i,n}$. These are the values in the released code, which the runs here use, while the source reports -5 for both penalties [3]. The first term penalizes the packet's wait in i's queue, and the second rewards the reduction in slant range to d that the hop achieves.

The per-step reward is a surrogate for a cooperative criterion it does not express directly, the episode-level negative mean end-to-end delay over the set $\mathcal{D}$ of packets delivered in the episode,

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[-\frac{1}{|\mathcal{D}|}\sum_{p \in \mathcal{D}} T_p\right], \tag{3.7}$$

optionally augmented with a penalty per undelivered packet. The value-based learners of this paper are trained on the per-step reward (3.6).

3.3 RAMA Agent

RAMA equips each satellite with a bounded episodic memory whose entries are indexed by situation and labeled by outcome, and with a small Transformer that reads the retrieved entries together with the current observation, a window of the satellite's recent decisions and its neighbors' state.

3.3.1 RAMA Architecture

Figure 3.1 shows RAMA's network architecture.

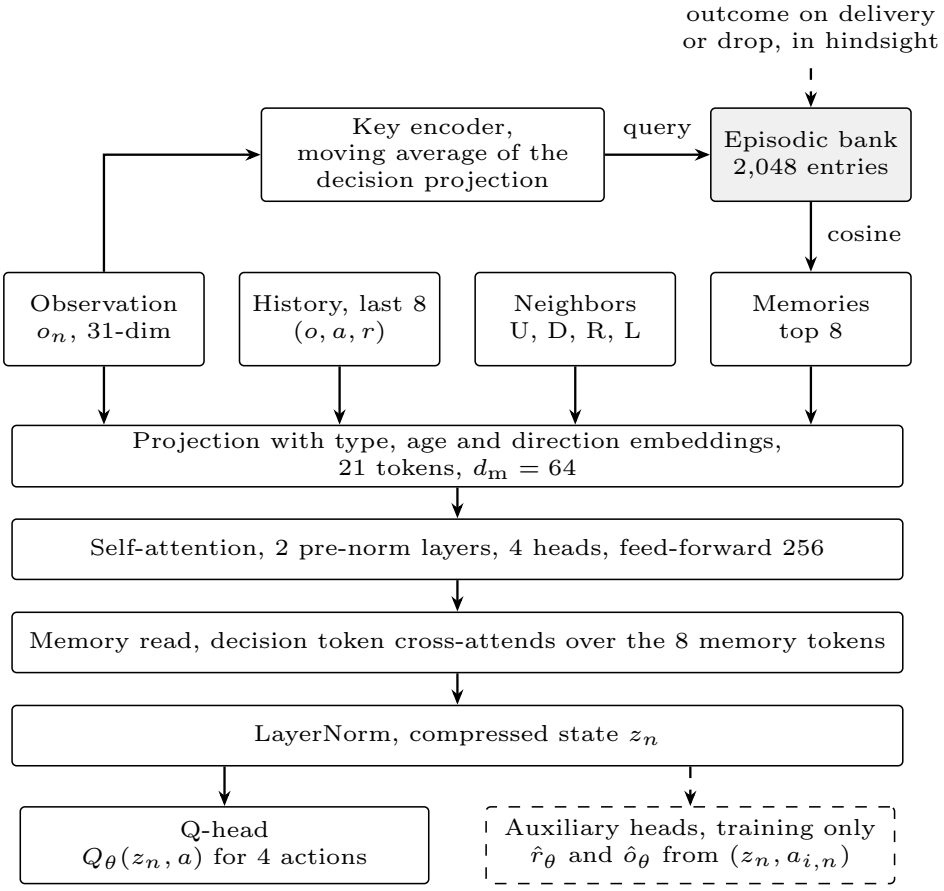


Figure 3.1: RAMA's decision network at one satellite. Solid arrows show the forward pass at decision time. Dashed arrows show the two paths outside it, the delayed write into the bank and the auxiliary heads.

At each decision the agent on satellite i forms 21 tokens of width $d_{\mathrm{m}} = 64$,

$$
\begin{aligned}
x^{\mathrm{dec}} &= W^{\mathrm{dec}} o_n + e_1, \\
x^{\mathrm{his}}_\ell &= W^{\mathrm{his}}\big[o^{(\ell)}; \mathbf{a}^{(\ell)}; \tanh(r^{(\ell)}/50)\big] + e_2 + e^{\mathrm{age}}_\ell, \\
x^{\mathrm{nei}}_j &= W^{\mathrm{nei}}\big[\mathbf{b}_j; \beta_j\big] + e_3 + e^{\mathrm{dir}}_j, \\
x^{\mathrm{mem}}_m &= W^{\mathrm{mem}}\big[\kappa_m; \mathbf{a}_m; \tanh(g_m/0.1); \varsigma_m; s_m\big] + e_4,
\end{aligned}
\tag{3.8}
$$

where W^{dec}, W^{his}, W^{nei} and W^{mem} are learned affine maps of the concatenated inputs, $\mathbf{a}$ is a one-hot action, and $e_1, \ldots, e_4$, e^{age}_ℓ and e^{dir}_j are learned type, age and direction embeddings.

History token x_ℓ^{his}, $\ell = 1, \ldots, 8$, holds the observation $o^{(\ell)}$, action $\mathbf{a}^{(\ell)}$ and reward $r^{(\ell)}$ of i's ℓ^{th} most recent decision. Neighbor token x_j^{nei} carries the flag β_j, which is 1 if neighbor j is linked. Satellite i retrieves from its episodic bank the set $\mathcal{M}_n$ of the eight entries whose keys are most similar to the key of o_n. Memory token x_m^{mem}, $m \in \mathcal{M}_n$, represents one of these entries, where κ_m is the key of the observation at the past decision the entry records, $\mathbf{a}_m$ is the action chosen at that decision, g_m is the realized delay to the destination, ς_m measures the time since the entry was written, and s_m is the cosine similarity between κ_m and the key of o_n. These are defined formally in (3.12) to (3.13), and Table 3.1 summarizes the token layout.

Table 3.1: Input tokens of the RAMA network, all projected to width $d_{\mathrm{m}} = 64$ by (3.8). History and memory slots are masked while a satellite has made fewer than eight decisions or its bank holds fewer than eight entries.

Tokens	Content	Count	Added embedding
Decision	current observation o_n	1	type
History	observation, action and reward of one of the satellite's recent decisions	8	type and age
Neighbor	neighbor state $\mathbf{b}_j$ and link flag β_j	4	type and direction
Memory	key, action, outcome, staleness and similarity of one retrieved entry	8	type only

Two Transformer blocks [42] in the pre-layer-norm arrangement [43] update the token matrix $X \in \mathbb{R}^{21 \times d_{\mathrm{m}}}$, whose rows are the tokens of (3.8),

$$\begin{aligned} X &\leftarrow X + \mathrm{Attention}\big(\mathrm{LayerNorm}(X)\big), \\ X &\leftarrow X + \mathrm{FeedForward}\big(\mathrm{LayerNorm}(X)\big), \\ \mathrm{head} &= \mathrm{softmax}\big(QK^\top/\sqrt{16} + M\big)\,V, \end{aligned} \tag{3.9}$$

where LayerNorm is layer normalization, Attention is multi-head attention with four heads, Q, K and V are a head's learned query, key and value projections of $\mathrm{LayerNorm}(X)$ to width 16, the mask M is $-\infty$ on empty history and memory slots, and FeedForward is a feed-forward network of width 256 with Gaussian error linear unit activation. After the second Transformer block, $\tilde{x}^{\mathrm{dec}}$ and $\tilde{X}^{\mathrm{mem}}$ denote the decision token and the eight memory tokens. The blocks in (3.9) mix all 21 tokens. The decision token then attends to the memory tokens only, which gives the retrieved entries a direct route into the compressed state

$$z_n = \mathrm{LayerNorm}\Big(\tilde{x}^{\mathrm{dec}} + \mathrm{Attention}\big(\mathrm{LayerNorm}(\tilde{x}^{\mathrm{dec}}), \mathrm{LayerNorm}(\tilde{X}^{\mathrm{mem}})\big)\Big), \tag{3.10}$$

where this attention step takes its queries from $\mathrm{LayerNorm}(\tilde{x}^{\mathrm{dec}})$ and its keys and values from $\mathrm{LayerNorm}(\tilde{X}^{\mathrm{mem}})$. While i's bank is empty there is nothing to attend to, so the attention term is dropped and $z_n = \mathrm{LayerNorm}(\tilde{x}^{\mathrm{dec}})$. Memory tokens carry no age or direction embedding, because their order is an artifact of similarity ranking. The memory read is therefore invariant to the order of the retrieved set, which attention without positional encodings provides [40].

A linear head on the compressed state $z_n \in \mathbb{R}^{d_{\mathrm{m}}}$ gives the action values, and RAMA acts greedily within the permitted set, so it never incurs r_u,

$$Q_\theta(z_n, a) = \mathbf{u}_a^\top z_n + u_a^0, \quad a_{i,n} = \arg\max_{a \in \mathcal{A}_{i,n}} Q_\theta(z_n, a), \tag{3.11}$$

where $\mathbf{u}_a$ and u_a^0 are the head's weights and bias for action a, and θ denotes the network's 145,124 trainable parameters, a size deliberately kept at on-board scale.

3.3.2 The Episodic Memory Bank

Each satellite i keeps its own bank of keys κ_m, actions a_m, outcomes g_m and write times τ_m,

$$\mathcal{B}^i = \{(\kappa_m, a_m, g_m, \tau_m)\}_{m=1}^{|\mathcal{B}^i|}, \qquad |\mathcal{B}^i| \le 2{,}048, \tag{3.12}$$

and the oldest entry is overwritten when $\mathcal{B}^i$ is full. Each entry is written into the deciding satellite's bank on delivery or drop of the packet, from the outcome report (3.5), so its outcome is g of (3.4) and its write time is the time of delivery or drop. Where the analysis needs the bank at agent i's decision at time t, it is written $\mathcal{B}_t^i$.

The defining design choice is the outcome label. The entry is not labeled with the instantaneous reward (3.6), since that quantity already sits in the replay buffer and teaches the memory nothing the value function does not already see. Instead, g_m is the realized delay from the decision to delivery, which is the long-horizon information a reactive policy lacks, because the memory records not only what the situation looked like but how it ended.

The write path is consequently delayed. At decision time, the agent attaches the key $\kappa(o_n)$ of its observation and the action $a_{i,n}$ to the packet, which carries them onward. When the packet is delivered or dropped, the completed entry is written into the bank. Hindsight labeling of this kind, which writes experience only once its consequence is known, is what distinguishes the bank from a replay buffer, and the delayed feedback channel of Section 3.2.3 is the formal device that makes it well defined inside the Dec-POMDP.

3.3.3 Retrieval and the Bank-Update Operator

Keys and queries are normalized outputs of an affine map W^{key} that has the same shape as W^{dec} but is not trained by gradient descent. Instead, after every gradient step it moves a fraction $\eta = 0.01$ of the way toward the current W^{dec},

$$\kappa(o) = \frac{W^{\text{key}} o}{\|W^{\text{key}} o\|_2}, \qquad W^{\text{key}} \leftarrow (1-\eta)\, W^{\text{key}} + \eta\, W^{\text{dec}}, \tag{3.13}$$

where the update averages the two maps' weights and biases, so W^{key} is a slowly moving average of W^{dec} and keys written to the bank earlier stay comparable with queries computed now. Retrieval returns the set $\mathcal{M}_n$ of the eight entries with the largest $s_m = \kappa_m^\top \kappa(o_n)$, which is the cosine similarity between κ_m and the current query $\kappa(o_n)$ because both have unit length. Each retrieved entry enters (3.8) with staleness $\varsigma_m = \min\{t_{\text{now}} - \tau_m,\, 1\}$, where t_{now} is the current time and all times are in seconds. Retrieval is a single small matrix product, and its cost has not been measured in isolation.

The slowly moving key encoder plays the role a target network plays in value learning [27]. If keys were computed by the trained projection W^{dec} itself, every gradient step would silently relocate every stored memory in embedding space and retrieval would degrade as training progressed, which is the non-stationary-memory problem. Stored keys are never re-encoded, so the slow average is what keeps old and new entries comparable during training, and in the frozen regime of Section 3.5 W^{key} does not move at all.

The bank evolves by the operator

$$\mathcal{B}^i_{t+1} = \Phi\big(\mathcal{B}^i_t, F^i_{t+1}\big) = \text{latest}_{2048}\left(\mathcal{B}^i_t \cup F^i_{t+1}\right), \tag{3.14}$$

where each outcome report in F^i_{t+1} becomes an entry, and $\text{latest}_K(S)$ keeps the K entries of S with the latest write times τ_m, or all of S when $|S| \le K$. The update is deterministic, so the next bank is fixed by the current bank and the newly arrived reports, with no random eviction or sampling. This deterministic property is used in Section 3.4.

The retrieved entries are consumed by the network through attention rather than by any hand-coded rule such as imitating the retrieved action with the lowest g_m. The network learns how much weight each memory deserves, including learning to ignore stale or misleading ones.

3.3.4 Training

RAMA and DDQN are trained with decaying random exploration [3] using a single network shared by all satellites, so the index i is dropped below. During training, the loop guard

introduced in Section 3.2.1 is not applied. The n^{th} RAMA decision gives a transition $(z_n, a_n, r_n, z_{n+1}, \chi_n)$, where $\chi_n = 1$ if the hop reaches d. The transition is stored in a replay buffer when the packet's next decision is made or the packet reaches d. RAMA is trained using double Q-learning [44] with unavailable neighbors masked out,

$$y_n = r_n + \gamma(1 - \chi_n)\, Q_{\theta^-}\Big(z^-_{n+1}, \arg\max_{a \in \mathcal{A}^{\text{link}}_{n+1}} Q_\theta(z_{n+1}, a)\Big), \tag{3.15}$$

where $\gamma = 0.99$ is the discount factor, and z_{n+1} and z^-_{n+1} are computed by the networks with parameters θ and θ^-, respectively. The set $\mathcal{A}^{\text{link}}_{n+1}$ holds the linked neighbors at the next hop, and the target network's parameters θ^- are copied from θ every 1,000 routing decisions. Two auxiliary heads predict the reward $\hat{r}_\theta$ and the next observation $\hat{o}_\theta$ from z_n and the action, and each gradient step minimizes over a minibatch of 32 replayed transitions

$$\mathcal{L}(\theta) = \mathbb{E}\Big[h\big(Q_\theta(z_n, a_n) - y_n\big) + \lambda_r\big(\hat{r}_\theta - r_n\big)^2 + \frac{\lambda_o}{31}\big\|\hat{o}_\theta - o_{n+1}\big\|_2^2\Big], \tag{3.16}$$

where h is the Huber loss and $\lambda_r = \lambda_o = 0.5$. Parameters are updated with Adam [45] at a learning rate of 10^{-3}, gradients are clipped at a norm of 1.0, and Table A.1 lists the remaining optimization settings.

The two auxiliary heads exist because of the theory. They are the objects that properties (AP1) and (AP2b) of Definition A.1 constrain, and the reward and observation terms of (3.16) follow the loss construction of Subramanian *et al.* [8]. In the notation of that framework, the observation head represents a predicted distribution $\hat{P}^y_\theta(\cdot \mid z_n, a_n)$ over the next observation through its mean $\hat{o}_\theta$. The observation term is then the maximum-mean-discrepancy surrogate $(\hat{o}_\theta - 2o_{n+1})^\top \hat{o}_\theta$ of [8], up to the factor $\lambda_o/31$ and the term $\|o_{n+1}\|_2^2$, which does not depend on θ. Minimizing the reward and observation terms is aimed at the (ε, δ) of Section 3.4, although the observation term predicts the packet's next observation and not the satellite's own next extended observation, so it is a proxy for δ_y and not an estimate of it.

3.3.5 Decision and Write Procedure

Algorithm 1 summarizes one decision of the agent on satellite i, combining the decision path with the delayed write path.

Algorithm 1 RAMA decision at satellite i, with the delayed write path.

Input: packet p at its n^{th} hop, with d not a neighbor of i, observation o_n, and outcome reports F received since the previous decision
for each outcome report $f = (\kappa, a, g, \tau) \in F$, in order of arrival **do** ▷ delayed write path
    $\mathcal{B}^i \leftarrow \Phi(\mathcal{B}^i, \{f\})$ ▷ overwrite the oldest entry when full
end for
$\mathcal{M}_n \leftarrow$ the eight entries of $\mathcal{B}^i$ with the largest $\kappa_m^\top \kappa(o_n)$ ▷ cosine-similarity retrieval
$z_n \leftarrow$ the network of (3.8) to (3.10) on o_n, the history window, the neighbor tokens and $\mathcal{M}_n$
$a_{i,n} \leftarrow \arg\max_{a \in \mathcal{A}_{i,n}} Q_\theta(z_n, a)$ ▷ ϵ-greedy and without the loop guard in training
attach $(\kappa(o_n), a_{i,n})$ to p ▷ completed when the report arrives
forward p along $a_{i,n}$

3.4 Theoretical Analysis

This section establishes what acting on RAMA's compressed state costs in principle. Throughout, fix a single agent facing a partially observed environment with histories H_t, bounded rewards and an $\{(\varepsilon_t, \delta_t)\}$-AIS generator $(\hat{\sigma}_t, \hat{P}_t, \hat{r}_t)$ in the sense of Definition A.1, with IPM function class $\mathcal{F}$.

Assumption 3.1. *The function class $\mathcal{F}$ is convex, symmetric ($f \in \mathcal{F} \Rightarrow -f \in \mathcal{F}$), and contains the zero function. Consequently $\mathcal{F}$ is star-shaped about zero, so that if $f \in \mathcal{F}$ and $c \geq 1$ then $f/c \in \mathcal{F}$.*

Assumption 3.1 holds for the three classes named in Section A.1, the total-variation class of functions bounded by one, the Wasserstein class of 1-Lipschitz functions vanishing at a fixed point and unit balls of reproducing-kernel Hilbert spaces [8].

Define the history-based dynamic program by $V_{T+1} \equiv 0$ and, for $t = T, \ldots, 1$,

$$Q_t(h_t, a_t) = \mathbb{E}[R_t \mid h_t, a_t] + \mathbb{E}[V_{t+1}(H_{t+1}) \mid h_t, a_t], \qquad V_t(h_t) = \max_{a \in \mathcal{A}} Q_t(h_t, a), \tag{3.17}$$

and the AIS dynamic program on the compressed spaces by $\hat{V}_{T+1} \equiv 0$ and

$$\hat{Q}_t(\hat{z}, a) = \hat{r}_t(\hat{z}, a) + \int_{\hat{\mathcal{Z}}_{t+1}} \hat{V}_{t+1}(\hat{z}')\, \hat{P}_t(d\hat{z}' \mid \hat{z}, a), \qquad \hat{V}_t(\hat{z}) = \max_{a \in \mathcal{A}} \hat{Q}_t(\hat{z}, a). \tag{3.18}$$

The question is how far (3.18) can drift from (3.17).

3.4.1 The Value-Approximation Bound

The proofs repeatedly need to convert an IPM bound between two distributions into a bound on the difference of expectations of a function that need not belong to $\mathcal{F}$. The Minkowski functional (A.2) is the conversion factor.

Lemma 3.2 (IPM–gauge inequality)**.** *Let $\mu, \nu \in \Delta(\mathcal{X})$ and let f be measurable with $\rho_{\mathcal{F}}(f) < \infty$. Then*

$$\left| \int_{\mathcal{X}} f \, d\mu - \int_{\mathcal{X}} f \, d\nu \right| \leq \rho_{\mathcal{F}}(f) \, d_{\mathcal{F}}(\mu, \nu). \tag{3.19}$$

Proof. Write $\rho = \rho_{\mathcal{F}}(f)$ and $I = \left| \int f \, d\mu - \int f \, d\nu \right|$. Let $c > \rho$ be arbitrary. By definition of the infimum there exists $c' \in [\rho, c)$ with $f/c' \in \mathcal{F}$, and since $c/c' > 1$, star-shapedness (Assumption 3.1) gives $f/c = (c'/c)(f/c') \in \mathcal{F}$. Hence, by the definition (A.1) of the IPM,

$$I = c \left| \int \frac{f}{c} \, d\mu - \int \frac{f}{c} \, d\nu \right| \leq c \, d_{\mathcal{F}}(\mu, \nu).$$

Taking the infimum over $c > \rho$ yields $I \leq \rho \, d_{\mathcal{F}}(\mu, \nu)$ when $\rho > 0$. If $\rho = 0$, the display holds for every $c > 0$, so $I \leq c \, d_{\mathcal{F}}(\mu, \nu)$ for all $c > 0$ and therefore $I = 0$, which again matches (3.19). □

Theorem 3.3 (Value approximation)**.** *Define $\alpha_{T+1} = 0$ and, for $t = T, \ldots, 1$,*

$$\alpha_t = \varepsilon_t + \rho_{\mathcal{F}}(\hat{V}_{t+1}) \, \delta_t + \alpha_{t+1} = \sum_{s=t}^{T} \Big(\varepsilon_s + \rho_{\mathcal{F}}(\hat{V}_{s+1}) \, \delta_s \Big). \tag{3.20}$$

Then for every t, every realization h_t of H_t and every $a_t \in \mathcal{A}$,

$$\left| Q_t(h_t, a_t) - \hat{Q}_t(\hat{\sigma}_t(h_t), a_t) \right| \leq \alpha_t \quad \text{and} \quad \left| V_t(h_t) - \hat{V}_t(\hat{\sigma}_t(h_t)) \right| \leq \alpha_t. \tag{3.21}$$

The proof proceeds by backward induction on t, using Lemma 3.2 to control the kernel term, and is given in full in Appendix A.3.

The bound has the right qualitative shape for routing. Errors accumulate additively over the remaining horizon, the reward-modeling error ε and the self-prediction error δ enter separately, and δ is weighted by how "irregular" the downstream compressed value function is, as measured by $\rho_{\mathcal{F}}(\hat{V}_{t+1})$. For the total-variation class, $\rho_{\mathcal{F}}(\hat{V}) = \sup |\hat{V}|$, which is finite whenever rewards are bounded.

3.4.2 The Policy-Suboptimality Bound

The practically important consequence is a guarantee for the policy an agent actually executes, namely the one that is greedy with respect to the AIS dynamic program.

Theorem 3.4 (Policy suboptimality)**.** *Let $\hat{\pi} = (\hat{\pi}_1, \dots, \hat{\pi}_T)$ be the AIS-greedy policy, $\hat{\pi}_t(h_t) \in \arg\max_a \hat{Q}_t(\hat{\sigma}_t(h_t), a)$, and let $V_t^{\hat{\pi}}$ denote its value. Then for every t and every realization h_t,*

$$V_t(h_t) - V_t^{\hat{\pi}}(h_t) \le 2\alpha_t, \tag{3.22}$$

with α_t as in (3.20).

The proof, also by backward induction, bounds the gap between the AIS value and the executed policy's value and then applies Theorem 3.3. It is given in full in Appendix A.3.

3.4.3 The Discounted Infinite-Horizon Case

Routing episodes in the simulator are long relative to per-hop timescales, so the stationary discounted criterion is the natural idealization. Suppose the AIS generator is time-homogeneous, $(\hat{\sigma}, \hat{P}, \hat{r})$ with uniform constants (ε, δ), rewards are bounded, and $\gamma \in [0, 1)$. Let $\hat{V}$ be the unique fixed point of the AIS Bellman operator

$$(\mathcal{T}\hat{V})(\hat{z}) = \max_a \left[\hat{r}(\hat{z}, a) + \gamma \int \hat{V}(\hat{z}')\, \hat{P}(d\hat{z}' \mid \hat{z}, a) \right], \tag{3.23}$$

which exists because $\mathcal{T}$ is a γ-contraction in the supremum norm.

Corollary 3.5 (Infinite-horizon bounds)**.** *Under the conditions above, for every history h,*

$$\left| V(h) - \hat{V}(\hat{\sigma}(h)) \right| \le \frac{\varepsilon + \gamma\, \rho_{\mathcal{F}}(\hat{V})\, \delta}{1 - \gamma}, \qquad V(h) - V^{\hat{\pi}}(h) \le \frac{2\big(\varepsilon + \gamma\, \rho_{\mathcal{F}}(\hat{V})\, \delta\big)}{1 - \gamma}, \tag{3.24}$$

where $\hat{\pi}$ is the stationary AIS-greedy policy.

The policy bound in (3.24) is the one stated in [8, Thm. 27]. The proof replays the finite-horizon argument as a fixed-point recursion in the supremum norm, and is given in Appendix A.3.

3.4.4 RAMA's Compressed State Is an Approximate Information State

The theorems above apply to any compression satisfying Definition A.1. The contribution of this subsection is to verify the definition for the compression RAMA actually computes. That is not obvious, because the compression routes through a growing, non-parametric memory bank, an object outside the settings in which AIS theory has previously been instantiated.

The analysis is carried out for a single agent in the frozen regime of Section 3.5. All network parameters θ, including the key encoder W^{key}, are frozen, while the memory bank continues to operate. The behavior of the other satellites is handled by assumption rather than by analysis. The analysis follows satellite i through its own decisions in time, indexed by t as in Section 3.2.3, and a decision of i at time t on a packet making its n^{th} hop has $z_t^i = z_n$.

Assumption 3.6 (Stationary co-agents)**.** *Every satellite other than agent i executes a policy that is fixed for the duration of the analysis.*

Under Assumption 3.6 the co-agents form part of a time-homogeneous environment, so from agent i's perspective the process is a stationary partially observed one and the single-agent framework applies unchanged. The assumption is satisfied exactly in the frozen regime, where every agent's weights are frozen, and it is violated during joint training.

Define the agent's internal state as the pair

$$\tilde{s}_t^i = \left(w_t^i,\ \mathcal{B}_t^i\right), \tag{3.25}$$

where the window w_t^i holds the current extended observation y_t^i together with the observations, actions and rewards of the agent's eight most recent decisions, and $\mathcal{B}_t^i$ is the memory bank. The internal state lives in the time-invariant space $\tilde{\mathcal{Z}} = \mathcal{W} \times \mathbb{B}$, where $\mathcal{W}$ is the space of such windows and $\mathbb{B}$ the space of banks with at most 2,048 entries. Windows have a fixed length and banks hold at most 2,048 entries of fixed width, so every element of $\tilde{\mathcal{Z}}$ can be written as a padded array of fixed size, and $\tilde{\mathcal{Z}}$ is a subset of a Euclidean space as Definition A.1 requires. The compressed state (3.10) is a deterministic readout $z_t^i = \zeta_\theta(\tilde{s}_t^i)$, because under a frozen W^{key} the retrieved set and the staleness of its entries are deterministic functions of the bank and the current extended observation. It therefore suffices to establish the AIS properties for $\tilde{s}_t^i$, with the reward and prediction heads composed with the readout.

Assumption 3.7 (Reward head accuracy)**.** *There exists $\varepsilon \geq 0$ such that, for all reachable histories h_t and actions a, $\big|\mathbb{E}[R_t \mid h_t, a] - \hat{r}_\theta\big(\zeta_\theta(\tilde{s}_t), a\big)\big| \leq \varepsilon$, where $\tilde{s}_t$ is the internal state generated by h_t.*

Assumption 3.8 (Observation-prediction accuracy)**.** *There exists $\delta_y \geq 0$ such that, for all reachable h_t and a, $d_{\mathcal{F}}\Big(\mathbb{P}(Y_{t+1} \in \cdot \mid h_t, a),\ \hat{P}^y_\theta(\cdot \mid \zeta_\theta(\tilde{s}_t), a)\Big) \leq \delta_y$.*

Assumptions 3.7 and 3.8 are the empirical content of the construction, and the reward and observation terms of (3.16) are the training signals aimed at them. Both constants are stated as suprema over reachable histories. The structural conditions, by contrast, hold exactly by construction, which is the point of the next result.

Proposition 3.9 (Memory-augmented AIS)**.** *Consider the frozen regime, take $\mathcal{F}$ to be the total-variation function class, and let Assumptions 3.6, 3.7 and 3.8 hold. Then the internal-state process $\tilde{s}^i_t$ of (3.25), with reward function $\hat{r}_\theta \circ \zeta_\theta$ and the update kernel induced by the observation-prediction head, is an (ε, δ)-AIS generator with $\delta \leq \delta_y$. Consequently Theorems 3.3 and 3.4 and Corollary 3.5 hold for the AIS dynamic program on RAMA's internal state with these constants.*

Proof sketch. Well-definedness follows by induction from $\mathcal{B}_{t+1} = \Phi(\mathcal{B}_t, F_{t+1})$ with Φ deterministic, starting from the bank that training left behind. Property (AP2a) then holds exactly, because the update $\hat{\varphi}\big((w, \mathcal{B}), y', a\big) = \big(\mathrm{shift}(w, y', a),\ \Phi(\mathcal{B}, F(y'))\big)$ is a deterministic function of the previous internal state, the new observation and the action. This is precisely where the determinism of Φ, engineered in Section 3.3.3, is consumed. Given (AP2a), the next internal state is the deterministic image of the next observation, and for the total-variation class pushforwards under measurable maps are non-expansive, so the observation-prediction error of Assumption 3.8 transfers to state prediction with $\delta \leq \delta_y$. Finally, (AP1) is Assumption 3.7 verbatim, since $z_t = \zeta_\theta(\tilde{s}_t)$ is a deterministic readout. The full proof is given in Appendix A.3.

3.5 Deployment Regimes

The theory of Section 3.4 assumes frozen network weights, and the main focus of Chapter 4 measures what the agents do without further learning, so the evaluation uses a frozen regime. In the frozen regime all network parameters, including RAMA's key encoder, are fixed, actions are greedy and no gradient step is taken. RAMA's banks start from the state training left them in and stay live, retrieving and recording outcomes, so frozen RAMA

still accumulates experience without learning, whereas frozen DDQN does not change at all.

The evaluation of Chapter 4 also considers an online regime, in which each satellite starts from the same offline-trained network and continues gradient updates from its own replay buffer. This is to mirror DDQN's online phase introduced in [3]. In this case, all RAMA agents start with the same banks as well. Both regimes are considered for RAMA and DDQN on Kepler and Starlink shell 1, as a 2×2 design over method and weight stability, alongside shortest-path routing.

Comparing frozen RAMA with frozen DDQN compares the two methods as a whole, including their observations and training procedures, and it is the comparison used for any claim about RAMA relative to DDQN. Comparing the frozen and online regimes of the same agent isolates the effect of weight stability, that is, of continuing gradient updates during deployment.

Chapter 4

Performance Evaluation

4.1 Experimental Setup

All experiments use the simulator of Lozano-Cuadra *et al.* [17] with the two constellations of Table 4.1, both implemented in the simulator. The simulator is extended with the RAMA agents and the direction scores. Kepler, introduced in Section 1.1, carries the full evaluation of this chapter, and Section 4.7 repeats it in both regimes on the larger Starlink shell 1. Table 4.2 records every configured value for the Kepler runs, orbital geometry and link budget included, and is the authoritative reference for what was actually run. The agent hyperparameters are in Appendix A.4, Table A.1.

Table 4.1: Parameters of the two constellations used in the experiments.

	Kepler	Starlink shell 1
Satellites	140	1,584
Planes	7	72
Altitude	600 km	550 km
Inclination	98.6°	53°
Walker pattern	Star	Delta

Each learned method is evaluated in the two regimes of Section 3.5, which together form a 2×2 design over method and weight stability. Frozen RAMA is the deployment regime of Section 3.5 and is the only condition the theory of Section 3.4 covers. The remaining three

Table 4.2: Simulation configuration for all five reported conditions on Kepler.

Parameter	Value
Constellation	Kepler (Walker star)
Orbital planes O	7
Satellites per plane	20
Total satellites N	140
Altitude	600 km
Inclination	98.6°[†]
Minimum elevation angle	30°
Ground stations	8
Simulated duration	3,600 s
Packets created	154,823 per run
Packets delivered	154,815–154,821 per run
Topology update interval (`movementTime`)	1 s
Orbital movement speed multiplier (`ndeltas`)	1
Packet size	64,800 bits
Satellite buffer limit	1×10^6 bytes
Ground-station buffer limit	1×10^{11} bytes
Radio link budget, ground links and inter-satellite links	
Carrier frequency, downlink and uplink	20 GHz and 30 GHz
Carrier frequency, inter-satellite links	26 GHz[‡]
Channel bandwidth	500 MHz
Transmit power, all links	10 W[§]
Antenna diameter	0.26 m[§]
Antenna efficiency	0.55
Pointing loss	0.3 dB
Noise figure	2 dB
Noise temperature	290 K
Minimum admissible link rate	10 kbps
Spectral efficiency, all links	Modulation and coding table

[†] Simulator configuration value, not a published Kepler figure.

[‡] Corrected value, see Section 4.4.

[§] Released code, with 0.33 m at ground-station transmitters. The source gives ground stations 20 W and 0.33 m in both directions [3].

are controls. Shortest-path routing is the fifth condition and is stateless, so the distinction does not apply to it.

The eight ground stations are those of Lozano-Cuadra *et al.* [3], mostly taken from the Kongsberg Satellite Services (KSAT) ground station network, in Málaga, Los Angeles, Mauritius, Vardø, Nuuk, Nemea, the Azores and Bangalore. The selection is deliberately spread, and includes two high-latitude sites, Vardø and Nuuk, along with Los Angeles and Mauritius, which are 18,400 km apart, close to antipodal, and form the longest pair in the set. The number of packets a ground station sends is determined using a Poisson process assuming each person near the ground station generates 8,593 bytes of data per second, and its packets are divided evenly among the other ground stations. Traffic therefore follows population, and Bangalore and Málaga generate the most, 63.4% and 18.2% of all packets, while Nuuk generates the least, 13 packets over the hour. Packets are generated from a fixed seed over 3,600 s of simulated time, giving 154,823 created packets per run, so every Kepler run generates the same packets.

Two configuration choices deserve comment. First, the topology update interval `movementTime` is set to 1 s in every evaluation run, which changes only how finely the orbital motion is discretized (Section 3.1). Second, the frequency used by inter-satellite links is increased to 26 GHz, which slightly increases the maximum distance before the data rate drops to zero.

Mean queueing delay is below one microsecond in all five conditions and the largest single queueing event anywhere in the five runs is 0.27 ms, against a mean end-to-end latency of at least tens of milliseconds. Queueing therefore accounts for 0.0002% of measured latency, and the results below characterize routing under topological stress only. They say nothing about routing under congestion.

The two learned agents were trained offline with different procedures. RAMA was trained over a 50 s simulated episode with the topology updated every 0.1 s, and the DDQN baseline over a 4.0 s episode with the topology updated every second, both with the same eight ground stations and `ndeltas` $= 5805.44/20 = 290.27$. RAMA's training episode therefore covers 14,514 s of orbital motion, about 2.5 orbital periods of 5,805 s at 600 km, in 500 snapshots, while the baseline's covers 1,161 s, about one fifth of a period, in four. Each was then evaluated over 3,600 s at `ndeltas` $= 1$, which spans about 0.62 of one period. The same settings were used on Starlink shell 1, where the orbital period at 550 km is 5,739 s, so the spans are about 2.5 periods for RAMA's training, one fifth of a period for the baseline's and 0.63 of a period for evaluation.

A path is called long when it exceeds twice shortest path's median hop count on the same constellation, which is 14 hops on Kepler and 24 hops on Starlink shell 1.

4.2 Baselines

The first baseline is Dijkstra's shortest path algorithm [21] for finding a minimum-hop path to d. The path is computed at the source ground station over the current topology graph and carried by the packet, and it is recomputed from the exact, current topology of the whole constellation at every topology update, also for packets in transit. Shortest path is therefore an idealized reference with full global knowledge, which the simulator provides at no cost. Edge weights are unit, so the path minimizes hop count rather than latency, and Section 4.5.3 shows that the difference between those two objectives matters.

The second baseline is the published DDQN routing method for this simulator [3], a recent state-of-the-art method for decentralized LEO packet routing. It is adopted since it shares RAMA's fully decentralized, event-driven setting and simulator, so both methods face the same topology, traffic, action space and reward. It is a three-layer network Q'_ω of 2,116 parameters that uses only the current 28-dimensional observation $o'_n = \left[\mathbf{b}_{\mathrm{U}_i}, \mathbf{b}_{\mathrm{D}_i}, \mathbf{b}_{\mathrm{R}_i}, \mathbf{b}_{\mathrm{L}_i}, \bar{\phi}_i, \bar{\lambda}_i, \bar{\phi}_d, \bar{\lambda}_d\right]$, which is (3.2) without the direction scores. Since it decides from the current observation alone, using the same information as RAMA except the direction scores, it is the natural memoryless reference for RAMA. Its released training step minimizes

$$\mathcal{L}_B(\omega) = \frac{1}{4} \sum_{a \in \mathcal{A}_i} \left(Q'_\omega(o'_n, a) - \mathbb{I}[a = a_{i,n}]\, y_n^B \right)^2,$$
$$y_n^B = r_n + \gamma \max_{a'} Q'_{\omega^-}(o'_{n+1}, a'), \tag{4.1}$$

where ω^- are the parameters of a target network, copied from ω every 1,000 routing decisions. Despite its name, this released update bootstraps from the target network's own maximum and not from the double Q-learning target [44], and the indicator regresses the value of every action not taken toward zero. The name is kept to match Lozano-Cuadra *et al.* [3]. The DDQN is trained offline with its original procedure, which differs from RAMA's (Section 4.1), then evaluated both frozen and online.

4.3 Comparisons Performed

The five conditions are RAMA and the DDQN baseline, each frozen and online, and the shortest-path baseline, with both baselines as described in Section 4.2. All five route the same packets over the same constellation phase with the same topology update interval. The DDQN and shortest-path runs predate the RAMA runs by about five weeks, and

the simulator was extended in between with RAMA's agents and state builder, so the conditions share traffic, constellation and link configuration but not an identical build. Delivered packet counts agree to within six packets out of 154,821, so the offered load is matched. Both learned agents use the same action space and reward constants of Table A.1, and both operate under the same loop guard (Section 3.2.1), so neither is advantaged by it. They differ in observation, since only RAMA observes the direction scores, its recent decisions and its memory bank, and in training procedure (Section 4.1). RAMA's training episode covers 12.5 times the baseline's orbital motion in 125 times as many topology snapshots. Their learning updates also differ (Table A.1). RAMA samples minibatches of 32 from a replay buffer of 4,096 transitions and minimizes a Huber loss on a masked double Q-learning target with gradient clipping, while the baseline samples minibatches of 16 from a buffer of 50 transitions and minimizes the squared loss (4.1) without clipping. Any of these differences can contribute to the gaps reported below, including the online baseline's degradation, so the comparisons describe RAMA as a whole against the baseline as released and do not isolate the memory bank.

4.4 Two Corrections to the Simulation Environment

Two problems in the released simulator were corrected before any of the runs reported in this chapter, since both inflated measured latency by mechanisms unrelated to routing quality. The first is a loop guard that the released code implements but leaves disabled. With only a two-hop mask active, a packet that entered a cycle of three satellites or more stayed in it until orbital motion broke a link on the cycle. The guard is enabled at five visits per satellite per packet, with a terminating fallback that selects the least-visited linked neighbor, which gives the permitted set of Section 3.2.1 and applies identically to both learned agents. The second is that two of the three graph builders built inter-satellite links at the 20 GHz ground-downlink carrier frequency instead of 26 GHz. At 20 GHz, inter-plane links longer than 4,482 km were admitted with a data rate of exactly zero and stranded packets in their transmit buffers for tens of seconds, and before the correction this alone caused every shortest-path packet that took longer than one second. Setting the carrier to 26 GHz removes these zero-rate links. With both corrections in place, shortest path delivers every one of its 154,821 packets in under 104 ms, and queueing and transmission together account for about 1% of latency in every condition (Section 4.6).

4.5 Routing Performance

Table 4.3 is the headline comparison on Kepler. In both regimes, RAMA outperforms DDQN on every aggregate in Table 4.3 on which the two differ, and the advantage is concentrated among the slowest packets. Online RAMA has the lowest mean, median, 90^{th}- and 99^{th}-percentile latency of the four learned conditions. RAMA's 90^{th}- and 95^{th}-percentile latencies are also below shortest path's in both regimes, which is possible because shortest path minimizes hop count rather than latency. Shortest path remains best on every other latency aggregate, although frozen RAMA matches it with no packet above 0.5 s. Figure 4.1 shows how each condition evolves over the hour, and Figure 4.2 shows the whole latency distribution.

4.5.1 Weight Stability

The largest single effect in Table 4.3 is not architectural. The two DDQN conditions share a checkpoint, a traffic seed, a constellation phase, an observation builder and an action space. They differ only in whether the agent undergoes further training during evaluation. Freezing it reduces mean latency from 186.36 to 53.71 ms, mean path length from 27.04 to 8.37 hops, the 99^{th} percentile from 1,636.81 to 296.01 ms and the fraction of packets that revisit a satellite from 21.59% to 2.95%. The online condition's median path is 10 hops against the frozen condition's 7, its 99^{th} percentile 224 hops against 46, and its longest path 655 hops against 166, and it puts 36.26% of its packets past 14 hops against 3.83%, a factor of 9.5. The online condition puts 14,780 packets past half a second and 5,427 past a second. The frozen condition puts 354 packets past half a second and 3 past a second.

Figure 4.1 shows that the two behave differently. Frozen DDQN tracks shortest path throughout, its mean latency rising 11.1% from the first 300 s of the run to the last 300 s against shortest path's 11.4%. Online DDQN climbs from 63.27 ms in the first 300 s to a peak of 272.5 ms for packets created between 1,500 and 1,800 s, and then oscillates back to 170.41 ms in the last 300 s, which is not the smooth trajectory a geometric cause would produce. Its hop count pattern mirrors its latency, so what it accumulates is distance traveled and not time waiting.

RAMA does not show the same effect. Updated online with continuous training, it stays close to its frozen performance and is marginally better on most aggregates. Mean latency falls from 46.51 to 45.90 ms, mean path length from 7.81 to 7.58 hops, the 99^{th} percentile from 98.71 to 96.41 ms and the share of packets above 14 hops from 2.55% to 1.66%. Revisits rise slightly, from 0.85% to 0.95%, and 10 packets exceed half a second

Table 4.3: Routing performance on Kepler over 3,600 s with eight ground stations. Frozen holds all weights fixed, and online continues per-satellite training while routing. Best value in each row in bold, ties unbolded, single seed per condition.

	RAMA frozen	RAMA online	DDQN frozen	DDQN online	Dijkstra (min hop)
Packets delivered	154,821	154,819	154,816	154,815	154,821
Weights during run	frozen	updating	frozen	updating	n/a
End-to-end latency (ms)					
Mean	46.51	45.90	53.71	186.36	**45.20**
Median	43.94	43.78	45.87	67.83	**42.76**
90^{th} percentile	68.24	**67.63**	74.41	477.42	71.36
95^{th} percentile	74.71	**72.29**	86.51	827.78	75.06
99^{th} percentile	98.71	96.41	296.01	1,636.81	**81.86**
99.9^{th} percentile	292.75	280.31	615.30	2,632.22	**93.84**
Maximum	493.3	880.7	1,159.9	4,660.0	**103.0**
Standard deviation	24.47	22.39	50.39	320.40	**16.65**
Latency components, mean (ms)					
Queueing	< 0.001	< 0.001	< 0.001	< 0.001	< 0.001
Transmission	0.472	0.466	0.548	1.878	**0.464**
Propagation	46.04	45.44	53.16	184.48	**44.74**
Slow packets					
Packets > 0.5 s	0	10	354	14,780	0
Packets > 1 s	0	0	3	5,427	0
Packets > 10 s	0	0	0	0	0
Path length (hops)					
Mean	7.81	7.58	8.37	27.04	**6.80**
Median	7	7	7	10	7
90^{th} percentile	11	11	11	69	**9**
99^{th} percentile	16	16	46	224	**11**
Maximum	78	122	166	655	**13**
Fraction > 14 hops[§]	2.55%	1.66%	3.83%	36.26%	**0**%
Mean ms per hop	**5.90**	5.99	6.35	6.82	6.58
Packets revisiting a satellite	0.85%	0.95%	2.95%	21.59%	**0.002**%

[§] 14 hops is twice shortest path's median path length of 7 hops, which every condition except online DDQN shares. Hop counts are read from each packet's recorded path, as the number of links it traversed.

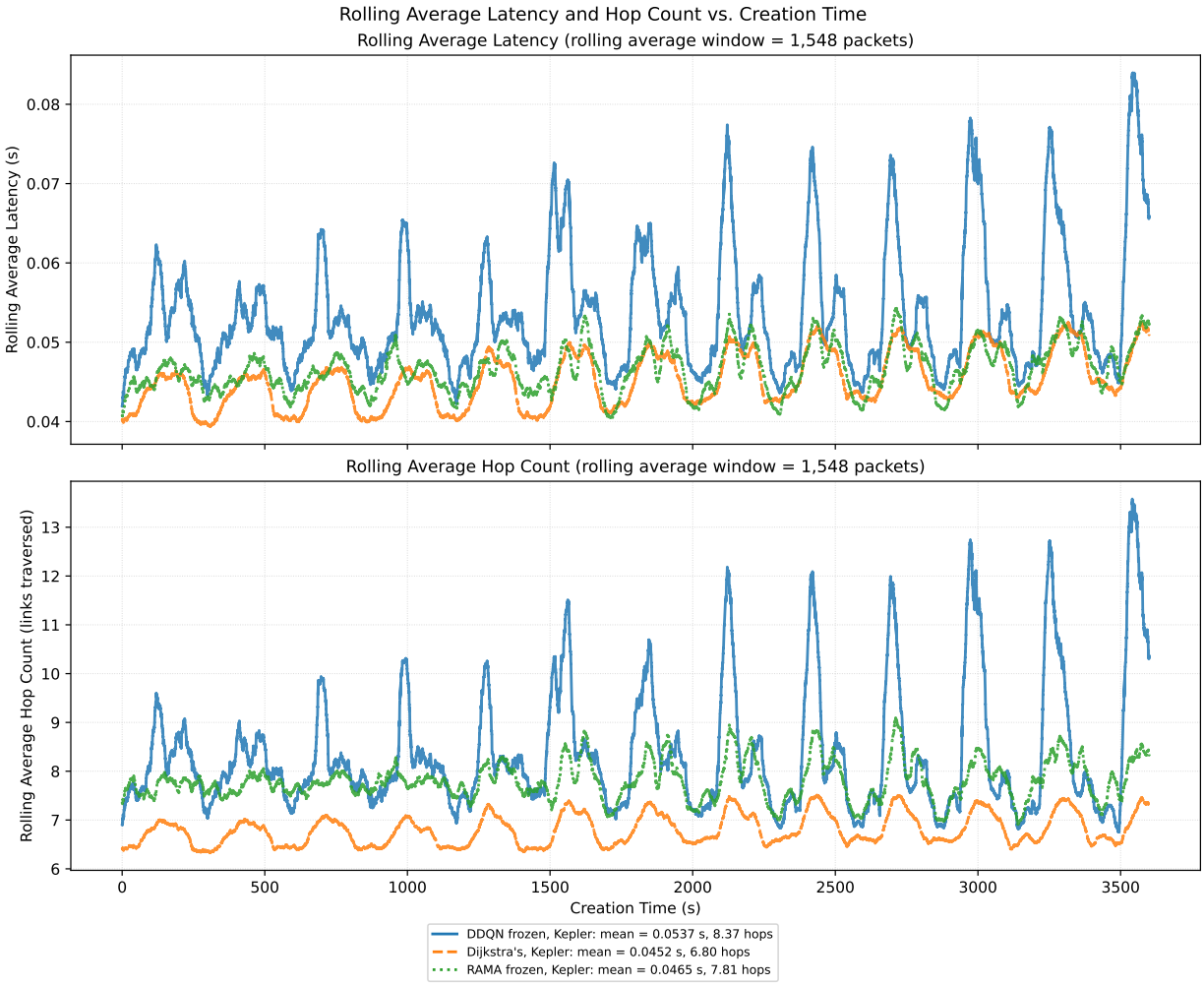


Figure 4.1: Rolling-average end-to-end latency (top) and rolling-average path length (bottom) against packet creation time on Kepler, over 3,600 s with eight ground stations, for the two frozen agents and Dijkstra on one shared time axis and one shared smoothing window. All three oscillate together on a period of a few hundred seconds, which follows the constellation's recurring major topology changes.

against none frozen. From the first 300 s of the run to the last, online RAMA's mean latency rises 10.3%, from 43.50 to 47.97 ms, against 11.4% for shortest path, so it does not degrade beyond what the geometry imposes.

Across frozen RAMA, online RAMA, frozen DDQN and shortest path, mean latency rises by 7.7% to 11.4% between packets created in the first and last 300 s of the run, so that drift is not caused by learning. Online DDQN's instead rises by 169.3%.

The frozen DDQN condition is therefore the reference used for any comparison between the two agents, and the remainder of this section uses it. The drift that motivated this work in Section 1.3 is also a property of online weight updating and not of memorylessness, since a memoryless policy with frozen weights drifts only as fast as the geometry does.

4.5.2 RAMA Against the DDQN Predecessor

RAMA improves on its predecessor in both of the baseline's modes. The improvement shows at every reported point of the latency distribution, and most of it comes from avoiding the baseline's long detours.

Against frozen DDQN, frozen RAMA is 13.4% lower in mean latency, 4.2% lower at the median, 8.3% lower at the 90^{th} percentile and 66.7% lower at the 99^{th}, a factor of 3.0. It takes 6.7% fewer hops and makes each hop 7.1% shorter, at 5.90 ms of propagation per hop against 6.35 ms. Revisit incidence is 0.85% against 2.95%, and no RAMA packet exceeds half a second against 354 for DDQN.

Frozen DDQN sends 3.83% of its packets past 14 hops against 2.55% for frozen RAMA, and every DDQN packet above its 99^{th}-percentile latency exceeds 14 hops. Section 4.8 shows that these long paths account for most of the difference between the two mean latencies. Avoiding these detours therefore gives RAMA most of its mean advantage over DDQN and its 66.7% lower 99^{th}-percentile latency.

Against the baseline in its online mode the margin is an order of magnitude, a 99^{th} percentile of 98.71 ms against 1,636.81 ms and a mean of 46.51 ms against 186.36 ms. Most of that larger margin is the weight-stability effect of Section 4.5.1 and not anything RAMA contributes, so the frozen comparison is the one to quote.

4.5.3 RAMA Against Shortest Path

Shortest path keeps a small lead over RAMA in mean latency, 45.20 ms against 46.51 ms. It routes on the full topology while RAMA sees only local observations, and RAMA closes

84.6% of the mean-latency gap between frozen DDQN and shortest path. Across the distribution the comparison splits. Figure 4.2(a) shows that frozen RAMA's latency is lower than shortest path's at every percentile from the 83.8$^{\text{th}}$ to the 95.4$^{\text{th}}$, which includes the 90$^{\text{th}}$, and higher at every percentile above, by 20.6% at the 99$^{\text{th}}$ and by a factor of 4.8 at the maximum. Updated online, RAMA's latency stays below shortest path's up to the 97.8$^{\text{th}}$ percentile.

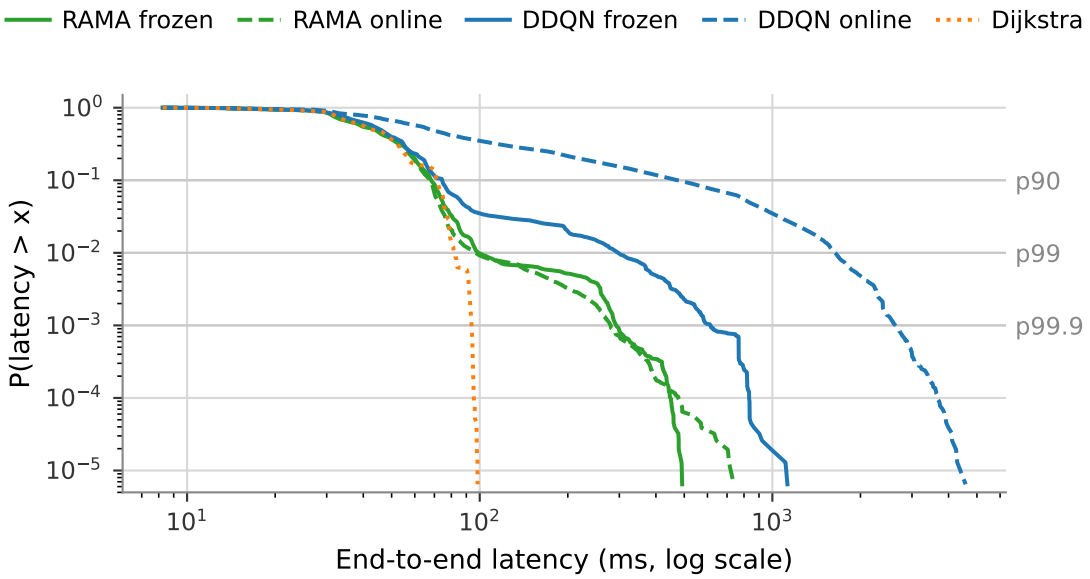
In the body of the distribution the margins are small in both directions. RAMA's median is 2.7% above shortest path's, 43.94 ms against 42.76 ms, and its mean is 2.9% above. Its 90$^{\text{th}}$ percentile is 4.4% below, 68.24 ms against 71.36 ms, and its 95$^{\text{th}}$ percentile is 0.5% below.

Table 4.4 gives the mechanism behind the crossover, and it turns on the objective the baseline actually optimizes. Minimizing hop count is not minimizing latency. On a long route the minimum-hop path is forced onto long inter-satellite links, whose propagation delay grows with range, and the saving in hop count does not pay for the extra distance. On routes of up to six hops the three frozen conditions are within 0.31 ms of each other per hop. From seven hops upward shortest path pays more per hop than RAMA, 6.59 ms against 6.17 ms at 7 to 8 hops, 7.35 ms against 5.92 ms at 9 to 10 hops and 7.02 ms against 5.77 ms at 11 to 13 hops. On the 9 to 10 hop routes that is a mean latency of 67.8 ms against 58.0 ms.

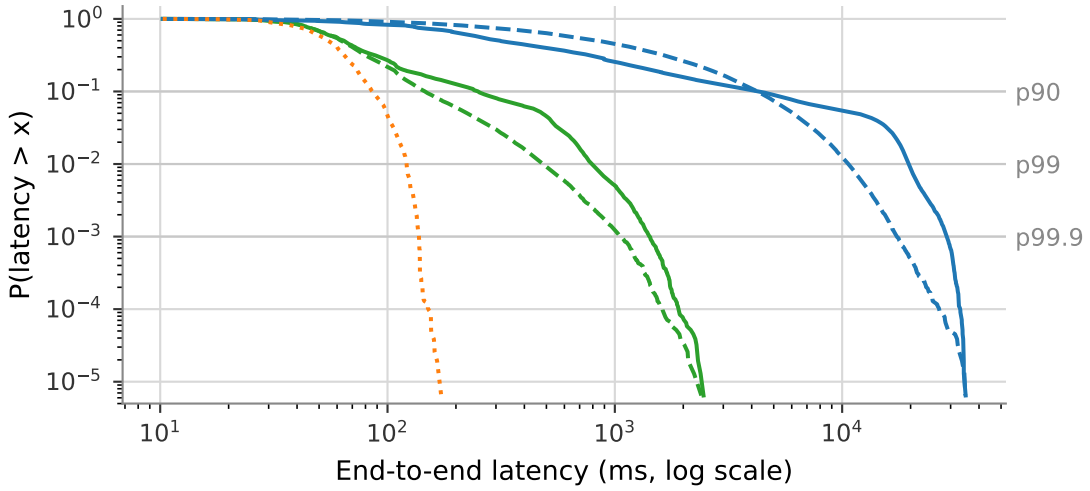
Table 4.4: Mean propagation delay per hop and mean end-to-end latency by path length on Kepler, for the three frozen conditions. All three are close on routes of up to six hops. On longer routes shortest path pays more per hop than RAMA, since minimizing hop count on a long route forces the use of long inter-plane links. Shortest path has no route longer than 13 hops.

	RAMA frozen		DDQN frozen		Dijkstra	
Path length (hops)	ms/hop	Mean (ms)	ms/hop	Mean (ms)	ms/hop	Mean (ms)
≤ 6	5.82	30.6	6.13	32.1	5.95	31.4
7 to 8	6.17	46.0	6.61	50.0	6.59	49.8
9 to 10	5.92	58.0	6.33	60.3	7.35	67.8
11 to 13	5.77	65.9	6.75	78.1	7.02	79.6
≥ 14	5.78	109.3	6.18	225.5	–	–

Aggregated over the run, RAMA takes 14.8% more hops than shortest path, 7.81 against 6.80, and makes each one 10.4% shorter, 5.90 ms against 6.58 ms. The two nearly cancel,



(a) Kepler



(b) Starlink shell 1

Figure 4.2: Complementary cumulative distribution of end-to-end latency, $P(\text{latency} > x)$ against x, for RAMA frozen, RAMA online, DDQN frozen, DDQN online and Dijkstra. Both axes are logarithmic with reference lines at the 90^{th}, 99^{th} and 99.9^{th} percentiles. On Kepler, frozen RAMA's curve lies below Dijkstra's from 62.1 ms to 75.2 ms and above it beyond.

and the means agree to within 2.9%. Per destination the trade is visible as a split instead of an average. RAMA is faster to the Azores by 4.1 ms, to Los Angeles by 4.1 ms, to Málaga by 2.5 ms and to Nemea by 1.7 ms. Shortest path is faster to the two high-latitude ground station, to Vardø by 10.4 ms and to Nuuk by 10.2 ms, and the two agree to within 0.6 ms at Mauritius and Bangalore.

All of RAMA's mean-latency deficit, and more, comes from the 2.55% of packets on which it exceeds 14 hops (Section 4.8). Long detours thus account for both RAMA's advantage over DDQN and its remaining gap to shortest path.

Reported as a single number, the comparison favors shortest path. It is lower on the mean, the median, the standard deviation and every percentile above the 95.4^{th}, and Figure 4.2(a) shows the gap opening across the whole upper end of the distribution and not only at the points the table samples. These runs therefore do not support presenting a learned policy as beating shortest-path routing outright, but since RAMA is lower between the 83.8^{th} and 95.4^{th} percentiles, they do not support the reading that it is simply the weaker method either. Section 5.1 states the narrower claim they do support.

4.6 Tracing the Slowest Packets

On Kepler, aggregate statistics establish that the conditions differ in their slowest packets. Tracing those packets individually establishes that they are slow for different reasons, a more useful result. Every delivered packet carries its arrival and departure timestamps at each hop, so the delay of each packet can be localized to a specific hop, satellite and link.

Slow packets in these runs are not caused by the environment. Of the three components of (3.1), queueing has effectively vanished. Mean queueing delay is under a microsecond in every condition and the largest queueing event in any of the five runs is 0.27 ms, so no slow packet anywhere is waiting. Propagation accounts for 98.97% to 98.99% of total latency in all five conditions, and for 98.94% to 99.00% of the latency of the packets above each condition's 99^{th} percentile. Every slow packet is slow because it traveled too far.

4.6.1 RAMA's Slowest Packets Share a Destination

Of the slowest 0.1% of frozen RAMA's packets, 149 of 155 are bound for Vardø. Vardø is the highest-latitude ground station in the set. It is also the destination on which shortest path beats RAMA by the largest margin, 36.91 ms against 47.28 ms in the mean, with

6.43 hops against 8.77. Nuuk, the other high-latitude ground station, is a close second, at 45.17 ms against 55.32 ms. RAMA's slowest packets and its worst destinations are the same phenomenon. Online RAMA shows the same pattern, with 123 of its slowest 155 packets bound for Vardø and 21 for Nuuk. These runs do not establish why the high-latitude ground stations are the exception. One explanation, not tested here, is that the satellite holding Vardø's downlink changes often, so a policy deciding hop by hop from local link state cannot know which of its near neighbors will still hold that downlink when the packet arrives, while shortest path is handed the current global graph.

Frozen RAMA's latency also grows across the hour, although its mean grows less than shortest path's. From the first 300 s of the run to the last 300 s, frozen RAMA's mean latency rises 7.7%, from 45.28 to 48.76 ms, against 11.4% for shortest path, while its 99$^{\text{th}}$ percentile rises from 95.63 to 116.79 ms and that of shortest path goes only from 81.54 to 85.46 ms.

4.6.2 DDQN's Slowest Packets Wander

Frozen DDQN puts 354 packets past half a second and 3 past a second, with a median of 81.5 hops among the packets past half a second. Its slowest 0.1% is spread across destinations instead of concentrated, 106 packets bound for Los Angeles, 24 for Nuuk and 23 for Vardø. The pattern is a policy that occasionally takes a long excursion on any long route, rather than one destination it cannot handle.

Online DDQN puts 9.5% of all packets past half a second and 3.5% past one second, and 21.6% of all its packets revisit a satellite at least once. Its mean path length is 27.04 hops against shortest path's 6.80, its median is 10 against 7, and its longest packet takes 655 hops, so the behavior is diffusion through the constellation and not a tight cycle. The loop guard is what bounds it. Five visits to each of 140 satellites allows 700 hops, and the longest path in the run is 655, so the bound is close to binding rather than comfortably slack. It is not a hard ceiling, since the terminating fallback may select a neighbor already at the limit when every direction is refused, which it did 6,134 times in this run. No packet circles indefinitely and none exceeds 4.66 s, but the guard cannot prevent a policy whose action values have collapsed from choosing an arbitrary direction at every hop.

Table 4.5: Aggregate results on Starlink shell 1 over all delivered packets. Latency in ms, best value in each row in bold. Revisits is the share of packets that visit a satellite more than once.

	RAMA		DDQN		
	Frozen	Online	Frozen	Online	Dijkstra
Mean latency	116.56	86.79	1,726.22	1,661.92	**57.13**
Median latency	64.29	62.56	312.79	842.15	**54.85**
90^{th} percentile latency	251.02	146.01	4,275.90	4,238.56	**85.71**
99^{th} percentile latency	791.48	484.55	19,642.82	10,651.59	**121.16**
Latency standard deviation	159.87	92.12	4,009.54	2,225.16	**21.85**
Latency above 0.5 s	7,027	1,439	61,386	98,321	**0**
Mean hops	40.72	30.24	483.08	391.12	**12.62**
Revisits	16.77%	12.30%	62.41%	84.71%	**0.11%**

4.7 A Larger Constellation

The experiments are repeated on Starlink shell 1, a larger LEO constellation of 1,584 satellites in 72 planes, with the same ground stations and traffic rates. Both RAMA and DDQN were retrained on this constellation, each with its Kepler training procedure. Table 4.5 gives the aggregate results.

With weights frozen, RAMA lowers mean latency by 93.2% and 99^{th}-percentile latency by 96.0% relative to DDQN. As on Kepler, RAMA takes far fewer long detours than DDQN, which exceeds 24 hops, twice shortest path's median of 12, on 89.52% of packets against 43.40% for RAMA.

Shortest path achieves the best value in every row. Frozen RAMA closes 96.4% of the mean-latency gap between frozen DDQN and shortest path, against 84.6% on Kepler, but its mean latency is still 2.04 times shortest path's. Long detours are also more common than on Kepler, where RAMA exceeds 14 hops on only 2.55% of packets.

Updated online, RAMA improves on its frozen performance, lowering mean latency by 25.5% and 99^{th}-percentile latency by 38.8%, mainly by shortening its paths above 24 hops from 73.7 to 48.4 hops on average. Those paths contribute 87.80 ms to frozen RAMA's mean latency and 58.71 ms to online RAMA's, so shortening them accounts for 29.09 ms of the 29.77 ms reduction. Online DDQN instead degrades over the run. Against frozen DDQN its aggregates are mixed, lower in the mean, the 90^{th} and 99^{th} percentiles and the standard deviation, but 2.69 times higher in the median and with more packets above

0.5 s. The mean latency of the other four conditions changes by −2.2% to +4.3% between packets created in the first and last 300 s of the run, while online DDQN's rises by 46.3%, from 1,338.50 ms to 1,957.74 ms. Most of its packets already take long paths when frozen, but as it updates, its revisits rise from 62.41% to 84.71%. As on Kepler, online RAMA has the lowest mean, median, 90$^{\text{th}}$- and 99$^{\text{th}}$-percentile latency of the four learned conditions. Figure 4.2(b) shows that RAMA's latency is higher than shortest path's at almost every percentile in both regimes, the exceptions covering only 0.18 percentile points frozen and 0.07 online.

4.7.1 Scalability

From 140 to 1,584 satellites, shortest path's mean latency grows by a factor of 1.26 and its mean hop count by 1.86. RAMA's mean latency grows by a factor of 2.51 and its hop count by 5.22 when frozen, and by 1.89 and 3.99 when updated online. DDQN's mean latency grows by a factor of 32.14 and its hop count by 57.71 when frozen, and by 8.92 and 14.46 when updated online. Its online factors are smaller only because online updates had already raised its Kepler mean latency to 3.47 times that of frozen DDQN, while on Starlink shell 1 both regimes reach similar mean latencies. RAMA's advantage over DDQN therefore widens on the larger constellation, while its gap to shortest path also grows. Frozen RAMA's 99$^{\text{th}}$-percentile latency grows faster than its mean, by a factor of 8.02 against 2.51, while shortest path's grows by a factor of only 1.48. This growth comes with more packets taking long detours and more revisits, 16.77% of frozen RAMA's packets on Starlink shell 1 against 0.85% on Kepler.

Part of this growth comes from the geometry of Starlink shell 1, where satellites moving north and satellites moving south pass over the same locations. Of frozen RAMA's packets, 23.4% first come within 2,000 km of d on a satellite moving north while d moves south, or south while d moves north. From that point, these packets need a median of 20 more hops to reach d, against four more hops for packets that arrive on a satellite moving in the same direction as d. Although they are less than a quarter of all packets, they account for 32.5% of frozen RAMA's hops. On the near-polar Kepler constellation, 15.5% of packets arrive this way, and they need a median of one more hop. Online updates therefore help RAMA most on Starlink shell 1, while on Kepler they lower its mean latency by only 1.3%, since frozen RAMA already takes few long detours there.

4.8 Long Paths and Mean Latency

Table 4.6 splits each run's packets into long paths and other paths, which decomposes its mean latency into the contribution of each category, the share of packets in it multiplied by their mean latency. The split applies to Starlink shell 1 as well as to Kepler. On Kepler, long paths contribute 3.37 ms to frozen RAMA's mean latency and 9.38 ms to frozen DDQN's, which accounts for 6.0 ms of the 7.2 ms difference between their means. Frozen DDQN's long paths average 39.0 hops, and 76.3% of them revisit a satellite, against 22.0 hops and 32.6% for frozen RAMA. On Starlink shell 1, long paths contribute 87.80 ms to frozen RAMA's mean latency and 1,721.05 ms to frozen DDQN's. This difference of 1,633.25 ms is larger than the 1,609.66 ms difference between their means, since the other paths contribute 23.59 ms more to RAMA's mean latency than to DDQN's, as far more of RAMA's packets take them. The same pattern holds in the online regime, where long paths account for 151.86 ms of the 140.46 ms difference on Kepler and 1,597.59 ms of the 1,575.13 ms difference on Starlink shell 1. Avoiding long detours thus gives RAMA most of its advantage over DDQN on Kepler with weights frozen, and all of it otherwise.

Long paths also account for RAMA's remaining gap to shortest path. On both constellations and in both regimes, the contribution of RAMA's long paths exceeds that of shortest path's long paths by more than the whole gap in mean latency, 3.37 ms against a gap of 1.31 ms for frozen RAMA on Kepler and 86.30 ms against 59.43 ms on Starlink shell 1.

Table 4.6: Mean latency split between long paths and other paths, in ms. A category's contribution is its share of packets multiplied by its mean latency, and the two contributions sum to the mean latency of Tables 4.3 and 4.5.

	RAMA		DDQN		
	Frozen	Online	Frozen	Online	Dijkstra
(a) Kepler, long paths have more than 14 hops					
Long-path share	2.55%	1.66%	3.83%	36.26%	0%
Long: mean hops	22.0	25.5	39.0	60.5	–
Long: revisits	32.6%	54.7%	76.3%	59.0%	–
Long: mean latency	132.23	142.96	245.24	425.39	–
Long: contribution	3.37	2.38	9.38	154.24	0
Other: mean latency	44.27	44.26	46.09	50.39	45.20
Other: contribution	43.15	43.52	44.32	32.12	45.20
(b) Starlink shell 1, long paths have more than 24 hops					
Long-path share	43.40%	44.14%	89.52%	91.05%	1.31%
Long: mean hops	73.7	48.4	538.0	428.2	27.0
Long: revisits	38.2%	26.9%	69.7%	92.7%	0.9%
Long: mean latency	202.32	133.01	1,922.56	1,819.20	114.97
Long: contribution	87.80	58.71	1,721.05	1,656.30	1.50
Other: mean latency	50.81	50.27	49.33	62.80	56.37
Other: contribution	28.76	28.08	5.17	5.62	55.63

Chapter 5

Discussion

5.1 Where the Comparisons Place RAMA

Against the DDQN predecessor the result is clear, although smaller than it first appears. Under matched weight stability, frozen against frozen, RAMA is better at every reported point of the distribution, by 13.4% in the mean, 4.2% at the median, 8.3% at the 90$^{\text{th}}$ percentile and 66.7% at the 99$^{\text{th}}$. Most of its gain comes from avoiding the baseline's long detours. Against the baseline in its online mode the margin is an order of magnitude at the 99$^{\text{th}}$ percentile, but Section 4.5.1 attributes most of that to the baseline's own degradation rather than to anything RAMA does. Unlike DDQN, RAMA does not degrade when updated online, and on Starlink shell 1 its online updates shorten its long detours.

Against shortest path the result is a split and not a win. Frozen RAMA selects shorter links among paths of equal hop count, which gives it the lower latency between the 83.8$^{\text{th}}$ and 95.4$^{\text{th}}$ percentiles, by 4.4% at the 90$^{\text{th}}$ percentile and 0.5% at the 95$^{\text{th}}$. Above the 95.4$^{\text{th}}$ percentile it is slower, by 20.6% at the 99$^{\text{th}}$ percentile and a factor of 4.8 at the maximum. Which of those matters depends entirely on the traffic. For bulk transfer, where throughput is set by the typical packet, the body of the distribution is what counts, and there the two are within a few percent of each other, RAMA being 4.4% lower at the 90$^{\text{th}}$ percentile and 2.7% higher at the median. For anything with a deadline, such as voice, control traffic or clock synchronization, the upper percentiles are the specification and shortest path is the better choice. Reported as a single scalar on these runs, shortest path wins on the mean, the median and the variance, and any summary that omits this is misleading. Even so, RAMA comes within 2.9% of shortest path's mean latency from local

observations only, closing 84.6% of the gap between its predecessor and shortest path, and the residual gap comes from the 2.55% of packets on which it exceeds 14 hops.

That attribution bears directly on the motivation in Section 1.3. The path-length drift of Figure 1.1, which opened this paper, was hypothesized as a memory failure, on the grounds that a policy conditioned only on the present cannot notice its own accumulated decisions lengthening the paths it selects. The frozen-baseline control refutes that reading. The same memoryless policy, with the same observation, the same reward and the same action space, does not drift when its weights are held fixed. The learning drifts, not the policy it starts from. The motivation for memory therefore has to rest on the two arguments set out in Section 1.3, that the baseline's learning signal never sees realized downstream delay and that a frozen policy has no other channel through which past experience can reach a present decision, and not on the drift.

On Starlink shell 1 the ordering holds and RAMA's advantage over DDQN widens. Mean hop count grows on the larger constellation for every method, as expected, but much more for DDQN than for RAMA, from 8.37 to 483.08 hops for frozen DDQN against 7.81 to 40.72 for frozen RAMA and 6.80 to 12.62 for shortest path. Relative to shortest path, mean latency also grows more for DDQN, so RAMA's advantage over DDQN widens with constellation size, from 13.4% to 93.2% in the mean. Frozen RAMA closes 96.4% of the gap to shortest path, against 84.6% on Kepler, but its mean latency is still 2.04 times shortest path's, since long detours are far more common on this constellation, partly for the geometric reason Section 4.7.1 gives. The decision cost of each agent does not depend on the size of the constellation, since each satellite decides from a local observation of fixed size. As constellations grow, the cost of a constellation-wide view grows with them, which makes routing from local information more valuable, and among the decentralized methods compared, RAMA's performance holds up far better as the constellation grows.

5.2 Trade-Offs

Against shortest path, RAMA trades latency for the information it needs. Shortest path has an advantage that RAMA does not have, the exact, current topology of the whole constellation at every topology update. A deployed shortest-path scheme would have to gather this global state and recompute routes over a graph that grows with the constellation, which the simulator provides at no cost. RAMA instead routes from local observations, its own history and its memory bank, at a per-decision cost that does not depend on the size of the constellation. The cost of routing from local information is small on Kepler and larger

on Starlink shell 1, and on both constellations and in both regimes RAMA's long detours account for more than the whole gap (Section 4.8).

Against DDQN, RAMA trades computation for latency. At every decision, each satellite retrieves entries from its bank and runs a Transformer over tokens built from its current observation, its recent decisions, its neighbors and the retrieved entries. This costs more computation and memory than DDQN's feed-forward network on the current observation alone, as Section 5.3 details. Online updates add a further cost through gradient steps for both RAMA and DDQN, while frozen RAMA adapts through its bank without them.

5.3 Costs

RAMA is a far larger model than the agent it replaces. Its 145,124 parameters exceed the DDQN baseline's 2,116 by a factor of 69, and every routing decision additionally scores the current observation against a bank of up to 2,048 stored entries. That size carries three costs, and none of them is characterized by the runs reported here. Inference cost per decision matters since the network has to run on a satellite payload and not on ground hardware. Training cost matters since the offline phase has to be paid before any routing benefit is available. Storage cost matters since the bank occupies memory the spacecraft has to carry.

Storage is the cost that scales with the constellation. A bank of 2,048 keys of width $d_{\mathrm{m}} = 64$, with their actions, outcomes and write times, is modest for one satellite, but it is replicated across all 140 satellites of Kepler, across the 1,584 satellites of Starlink shell 1 and across thousands in a full Starlink constellation. Banks are private to their satellites, so the total grows linearly with constellation size and nothing is amortized. Section 6.1 raises memory sharing as one way to change that. How far the capacity can be reduced before routing quality degrades is not established.

Chapter 6

Conclusion

This research paper set out from an empirical failure mode of learned routing, in which path lengths in a LEO constellation drift upward across a sustained deployment while the policy has no mechanism to notice or correct the drift as it compounds. It asked whether a routing agent could be given an ability to remember. The answer developed here is RAMA, which gives each satellite a history window and a bounded, hindsight-labeled episodic memory that is retrieved by similarity and read by a small Transformer. It is analyzed within the AIS framework.

The formulation extends the partially observable Markov decision process of Lozano-Cuadra *et al.* [3] to a Dec-POMDP with a delayed feedback channel inside the observation process. The architecture's four design choices, hindsight outcome labels, a slowly updated key encoder, a deterministic bank update and permutation-invariant memory reads, each follow from a theoretical or deployment requirement. The central new result is that in the frozen regime the pair of recent-history window and memory bank is itself an AIS, so the value-approximation and policy-suboptimality bounds of Subramanian *et al.* [8] transfer to the memory-augmented agent, although their constants are not computable for a learned compression.

RAMA was evaluated on the Kepler constellation, with weights frozen and updated online, against the DDQN routing method it descends from and minimum-hop shortest path, and the evaluation was repeated on Starlink shell 1. None of the findings concerns congestion, which the offered load never produced. RAMA is the better of the two learned agents at every reported point of the latency distribution, and most of its gain comes from avoiding the baseline's long detours. Updated online it stays close to its frozen performance, whereas the baseline degrades. Shortest path remains the stronger method overall, with

the lower mean, median and upper percentiles. RAMA nonetheless comes within a few percent of its mean latency from local observations alone and is faster in the upper body of the distribution, since minimizing hop count is not minimizing latency, and its remaining gap to shortest path comes from its long detours.

Weight stability dominates the differences between methods on these runs. With its weights frozen the baseline does not drift, so the drift that opened this paper was never evidence for memory, and the frozen-weight requirement of Proposition 3.9 becomes a practical advantage. On the larger Starlink shell 1, RAMA's advantage over DDQN widens and online updates shorten its long detours, although shortest path's lead grows as well. Routing from local information becomes more valuable as constellations grow. Since RAMA and the DDQN baseline differ in more than the memory bank, these results describe RAMA as a whole and do not isolate the bank's contribution.

The bank also lets a deployed agent keep recording and retrieving outcomes without gradient updates, which other DRL routing methods can adopt where on-board training is costly, and it is most useful where similar network states recur, as with periodic orbits and daily traffic patterns. Tracing individual slow packets showed that the conditions fail by unrelated mechanisms, so no single scalar summarizes this problem well, and mean latency least of all.

6.1 Future Work

The long detours that remain on Starlink shell 1, many of which occur when a packet approaches its destination on a satellite moving opposite to d, are the main target for future work.

Heavier traffic is needed to test whether the memory helps when queues build up. Holding link capacity fixed and raising traffic will not produce queues at this margin, since the mesh runs at well under one percent utilization, so capacity has to come down as offered load goes up. Only in that regime does the congestion input to RAMA's observation carry information.

The memory bank is not tied to RAMA's network, and it can be added to other DRL agents such as Rainbow [46] to test whether the gains carry over.

Banks are currently private, yet neighboring satellites traverse the same geography minutes apart. Exchanging a few high-value memories over existing inter-satellite link telemetry could let one agent's experience pre-warm another's bank, at the cost of a staleness-weighting problem.

If on-board budgets require an even smaller network, the trained RAMA policy can serve as a teacher for a distilled student, in the manner of resource-aware distillation for MARL [39].

References

[1] Peng Hu. Closing the performance and management gaps with satellite Internet: Challenges, approaches, and future directions. In *Internet Architecture Board (IAB) Workshop on Barriers to Internet Access of Services (BIAS)*, January 2024. arXiv:2401.07842. 1

[2] Kaan Çelikbilek, Zainab Saleem, Ruben Morales Ferre, Jaan Praks, and Elena Simona Lohan. Survey on optimization methods for LEO-satellite-based networks with applications in future autonomous transportation. *Sensors*, 22(4):1421, 2022. 1, 2, 8

[3] Federico Lozano-Cuadra, Beatriz Soret, Israel Leyva-Mayorga, and Petar Popovski. Continual deep reinforcement learning for decentralized satellite routing. *IEEE Transactions on Communications*, 73(10):8996–9012, 2025. arXiv:2405.12308. 1, 4, 7, 9, 14, 15, 16, 18, 22, 29, 31, 32, 33, 50

[4] Xiao Chen, Zhe Ji, Sheng Wu, Haoge Jia, Ailing Xiao, and Chunxiao Jiang. A distributed routing algorithm for LEO satellite networks: A multiagent transformer-MIX learning approach. *IEEE Internet of Things Journal*, 12(11):15748–15763, 2025. 1, 9, 11

[5] Yifeng Lyu, Han Hu, Rongfei Fan, Zhi Liu, Jianping An, and Shiwen Mao. Dynamic routing for integrated satellite-terrestrial networks: A constrained multi-agent reinforcement learning approach. *IEEE Journal on Selected Areas in Communications*, 42(5):1204–1218, 2024. 1, 9

[6] Peng Hu. SatAIOps: Revamping the full life-cycle satellite network operations. In *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*, pages 1–5, 2023. 1

[7] Jayakumar Subramanian and Aditya Mahajan. Approximate information state for partially observed systems. In *Proceedings of the 58th IEEE Conference on Decision and Control (CDC)*, pages 1629–1636, 2019. 2, 6, 11

[8] Jayakumar Subramanian, Amit Sinha, Raihan Seraj, and Aditya Mahajan. Approximate information state for approximate planning and reinforcement learning in partially observed systems. *Journal of Machine Learning Research*, 23(12):1–83, 2022. 2, 6, 7, 11, 23, 24, 26, 50, 60, 61, 64

[9] International Telecommunication Union. WRS-22: Regulation of satellites in Earth's orbit, 2023. https://www.itu.int/hub/2023/01/satellite-regulation-leo-geo-wrs/. Accessed 2026. 2

[10] Wentao He, Huayi Li, and Shi Qiu. GNN-ASSSP: A study of an attention mechanism driven dynamic routing algorithm for LEO satellite networks. *Aerospace Science and Technology*, 178:112361, 2026. 2, 9

[11] Peng Hu. Enabling resilient and real-time network operations in space: A novel multi-layer satellite networking scheme. In *2022 IEEE Latin-American Conference on Communications (LATINCOM)*, pages 1–6, 2022. 2

[12] Peng Hu. A cross-layer descent approach for resilient network operations of proliferated LEO satellites. In *2023 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6, 2023. 2

[13] Kepler Communications Inc. Kepler successfully launches first tranche of optical relay satellites, 2026. https://kepler.space/kepler-successfully-launches-first-tranche-of-optical-relay-satellites/. Accessed August 2026. 2

[14] Federal Communications Commission. Order and declaratory ruling, Kepler Communications Inc. petition for declaratory ruling to grant access to the U.S. market for Kepler's NGSO FSS system, 2018. FCC 18-162, 33 FCC Rcd 11453. https://www.fcc.gov/document/fcc-grants-kepler-communications-access-us-market. Accessed August 2026. 2

[15] Marc Boucher. Details emerge on Kepler Communications and Telesat small satellite constellations, 2016. SpaceQ. https://spaceq.ca/details_emerge_on_kepler_communications_and_telesat_small_satellite_constellations/. Accessed August 2026. 2

[16] Jonathan McDowell. Jonathan's Space Report: Space statistics, Starlink, 2026. https://planet4589.org/space/con/star/stats.html. Accessed October 2026. 2

[17] Federico Lozano-Cuadra, Mathias D. Thorsager, Israel Leyva-Mayorga, and Beatriz Soret. An open source multi-agent deep reinforcement learning routing simulator for satellite networks. In *Proceedings of the SPAICE Conference on AI in and for Space*, 2024. arXiv:2407.11047. 3, 9, 14, 30

[18] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. 6, 10

[19] Matthew Hausknecht and Peter Stone. Deep recurrent Q-learning for partially observable MDPs. In *AAAI Fall Symposium on Sequential Decision Making for Intelligent Agents*, 2015. arXiv:1507.06527. 6, 10

[20] Frans A. Oliehoek and Christopher Amato. *A Concise Introduction to Decentralized POMDPs*. SpringerBriefs in Intelligent Systems. Springer, 2016. 7, 10, 11

[21] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959. 8, 33

[22] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 4295–4304, 2018. 9, 11

[23] Matteo Gallici, Mario Martin, and Ivan Masmitja. TransfQMix: Transformers for leveraging the graph structure of multi-agent reinforcement learning problems. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1679–1687, 2023. 9

[24] Ke He, Thang X. Vu, Le He, Lisheng Fan, Symeon Chatzinotas, and Björn Ottersten. Asynchronous risk-aware multi-agent packet routing for ultra-dense LEO satellite networks, 2025. arXiv:2510.27506. 9, 17

[25] Jiahui Li, Geng Sun, Zemin Sun, Jiacheng Wang, Yinqiu Liu, Ruichen Zhang, Dusit Niyato, and Shiwen Mao. LLM-guided DRL for multi-tier LEO satellite networks with hybrid FSO/RF links, 2025. arXiv:2505.11978. 9

[26] Wenyu Sun, Chenhua Sun, Yasheng Zhang, Zhan Yin, and Zhifeng Kang. SCNOC-Agentic: A network operation and control agentic for satellite communication systems. *Electronics*, 14(16):3320, 2025. 9

[27] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. 10, 22

[28] Charles Blundell, Benigno Uria, Alexander Pritzel, Yazhe Li, Avraham Ruderman, Joel Z. Leibo, Jack Rae, Daan Wierstra, and Demis Hassabis. Model-free episodic control, 2016. arXiv:1606.04460. 10

[29] Alexander Pritzel, Benigno Uria, Sriram Srinivasan, Adrià Puigdomènech Badia, Oriol Vinyals, Demis Hassabis, Daan Wierstra, and Charles Blundell. Neural episodic control. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 2827–2836, 2017. 10

[30] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 10

[31] Peter C. Humphreys, Arthur Guez, Olivier Tieleman, Laurent Sifre, Théophane Weber, and Timothy Lillicrap. Large-scale retrieval for reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 10

[32] Thomas Schmied, Fabian Paischer, Vihang Prakash Patil, Markus Hofmarcher, Razvan Pascanu, and Sepp Hochreiter. Retrieval-augmented decision transformer: External memory for in-context RL. In *Proceedings of the 4th Conference on Lifelong Learning Agents (CoLLAs)*, volume 330 of *Proceedings of Machine Learning Research*, pages 376–417. PMLR, 2026. 10

[33] Vincent François-Lavet, Guillaume Rabusseau, Joelle Pineau, Damien Ernst, and Raphael Fonteneau. On overfitting and asymptotic bias in batch reinforcement learning with partial observability. *Journal of Artificial Intelligence Research*, 65:1–30, 2019. 11

[34] Hsu Kao and Vijay Subramanian. Common information based approximate state representations in multi-agent reinforcement learning. In *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 151 of *Proceedings of Machine Learning Research*, pages 6947–6967. PMLR, 2022. 11

[35] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of PPO in cooperative multi-agent games. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 11

[36] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024. arXiv:2402.03300. 11

[37] Shuo Liu, Zeyu Liang, Xueguang Lyu, and Christopher Amato. LLM collaboration with multi-agent reinforcement learning. In *Proceedings of the 40th AAAI Conference on Artificial Intelligence*, volume 40, pages 32150–32158, 2026. 11

[38] Chuanneng Sun, Songjun Huang, and Dario Pompili. LLM-based multi-agent reinforcement learning: Current and future directions, 2024. arXiv:2405.11106. 12

[39] Monirul Islam Pavel, Siyi Hu, Muhammad Anwar Ma'sum, Mahardhika Pratama, Ryszard Kowalczyk, and Zehong Jimmy Cao. KD-MARL: Resource-aware knowledge distillation in multi-agent reinforcement learning, 2026. arXiv:2604.06691. 12, 52

[40] Zijian Zhao, Jing Gao, and Sen Li. Bridging MARL to SARL: An order-independent multi-agent transformer via latent consensus, 2026. arXiv:2604.13472. 12, 21

[41] Guillaume Sartoretti, Justin Kerr, Yunfei Shi, Glenn Wagner, T. K. Satish Kumar, Sven Koenig, and Howie Choset. PRIMAL: Pathfinding via reinforcement and imitation multi-agent learning. *IEEE Robotics and Automation Letters*, 4(3):2378–2385, 2019. 12

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 20

[43] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer architecture. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pages 10524–10533. PMLR, 2020. 20

[44] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double Q-learning. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pages 2094–2100, 2016. 23, 33

[45] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015. 23, 66

[46] Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, 2018. 51

APPENDICES

Appendix A

Supplementary Material

Throughout, $\Delta(X)$ denotes the set of probability measures on a measurable space X, uppercase letters denote random variables and lowercase letters their realizations, and $[N]$ denotes $\{1, \dots, N\}$. Time is discrete, $t \in \{1, \dots, T\}$, with T finite except where an infinite-horizon discounted criterion is stated explicitly.

A.1 Integral Probability Metrics

Let $(\mathcal{X}, \mathcal{G})$ be a measurable space and $\mathcal{F}$ a class of uniformly bounded measurable functions on it. The IPM induced by $\mathcal{F}$ between $\mu, \nu \in \Delta(\mathcal{X})$ is

$$d_{\mathcal{F}}(\mu, \nu) \coloneqq \sup_{f \in \mathcal{F}} \left| \int_{\mathcal{X}} f \, d\mu - \int_{\mathcal{X}} f \, d\nu \right|. \tag{A.1}$$

Bounded functions give total variation, 1-Lipschitz functions give the Wasserstein distance, and unit balls of reproducing-kernel Hilbert spaces give maximum mean discrepancy [8]. The 1-Lipschitz class is not uniformly bounded as it stands, but adding a constant to f leaves (A.1) unchanged, so it can be restricted to functions vanishing at a fixed point, which are bounded by the diameter of $\mathcal{X}$ when that diameter is finite. The companion quantity used in the bounds of Chapter 3 is the Minkowski functional of $\mathcal{F}$,

$$\rho_{\mathcal{F}}(f) \coloneqq \inf\{c > 0 : f/c \in \mathcal{F}\}, \tag{A.2}$$

which measures how far a given function f lies outside the class $\mathcal{F}$.

A.2 Approximate Information States

The following definition is due to Subramanian *et al.* [8], stated here for a single agent that never sees the state S_t and acts instead on its history $H_t = (Y_1, A_1, Y_2, A_2, \ldots, Y_t)$ of observations and actions, with history spaces $\{\mathcal{H}_t\}$ and observation space $\mathcal{Y}$.

Definition A.1 ($\{(\varepsilon_t, \delta_t)\}$-AIS generator)**.** Let $\{\hat{\mathcal{Z}}_t\}_{t=1}^T$ be a pre-specified collection of Banach spaces, $\mathcal{F}$ a function class for IPMs, and $\{(\varepsilon_t, \delta_t)\}_{t=1}^T$ positive reals. A collection of history compression functions $\{\hat{\sigma}_t : \mathcal{H}_t \to \hat{\mathcal{Z}}_t\}_{t=1}^T$, approximate update kernels $\{\hat{P}_t : \hat{\mathcal{Z}}_t \times \mathcal{A} \to \Delta(\hat{\mathcal{Z}}_{t+1})\}_{t=1}^T$ and reward approximation functions $\{\hat{r}_t : \hat{\mathcal{Z}}_t \times \mathcal{A} \to \mathbb{R}\}_{t=1}^T$ is an $\{(\varepsilon_t, \delta_t)\}$-AIS generator if the process $\hat{Z}_t = \hat{\sigma}_t(H_t)$ satisfies the following two properties.

(AP1) *Sufficient for approximate performance evaluation.* For all t, all realizations h_t and all actions a_t,

$$|\mathbb{E}[R_t \mid H_t = h_t, A_t = a_t] - \hat{r}_t(\hat{\sigma}_t(h_t), a_t)| \le \varepsilon_t. \tag{A.3}$$

(AP2) *Sufficient to predict itself approximately.* For all t, h_t, a_t, with $\mu_t(\cdot) = \mathbb{P}(\hat{Z}_{t+1} \in \cdot \mid H_t = h_t, A_t = a_t)$ and $\nu_t(\cdot) = \hat{P}_t(\cdot \mid \hat{\sigma}_t(h_t), a_t)$,

$$d_{\mathcal{F}}(\mu_t, \nu_t) \le \delta_t. \tag{A.4}$$

Property (AP2) has a sufficient condition in two parts that is easier to verify architecturally [8]. **(AP2a)** says the compressed state evolves in a state-like manner, meaning there exist measurable update functions $\hat{\varphi}_t$ with $\hat{\sigma}_{t+1}(h_{t+1}) = \hat{\varphi}_t(\hat{\sigma}_t(h_t), y_{t+1}, a_t)$, and **(AP2b)** says it approximately predicts the next observation under $d_{\mathcal{F}}$. Together they imply (AP2). That sufficient condition is central to Chapter 3, because (AP2a) is a purely structural condition that RAMA's memory is engineered to satisfy exactly, leaving only (AP1) and (AP2b) to carry approximation error.

A.3 Deferred Proofs

This section collects the proofs of the value-approximation bound of Theorem 3.3, the policy-suboptimality bound of Theorem 3.4, the discounted infinite-horizon bounds of Corollary 3.5 and the memory-augmented AIS result of Proposition 3.9, all stated in Section 3.4. The statements, the error recursion α_t of (3.20) and all notation are as in Chapter 3.

Proof of Theorem 3.3. Backward induction on t. For $t = T+1$ both sides are identically zero and (3.21) holds with $\alpha_{T+1} = 0$.

Assume the V-inequality holds at $t+1$. Fix h_t and a_t, write $\hat{z}_t = \hat{\sigma}_t(h_t)$, and decompose using (3.17) and (3.18):

$$\left|Q_t(h_t, a_t) - \hat{Q}_t(\hat{z}_t, a_t)\right| \le \underbrace{\left|\mathbb{E}[R_t \mid h_t, a_t] - \hat{r}_t(\hat{z}_t, a_t)\right|}_{\text{(I)}} + \underbrace{\left|\mathbb{E}[V_{t+1}(H_{t+1}) \mid h_t, a_t] - \int \hat{V}_{t+1}\, d\hat{P}_t(\cdot \mid \hat{z}_t, a_t)\right|}_{\text{(II)}}. \tag{A.5}$$

Term (I) $\le \varepsilon_t$ by property (AP1). For term (II), insert the compressed value along the true trajectory:

$$\begin{aligned}\text{(II)} \le{}& \left|\mathbb{E}\big[V_{t+1}(H_{t+1}) - \hat{V}_{t+1}(\hat{\sigma}_{t+1}(H_{t+1})) \mid h_t, a_t\big]\right| \\ &+ \left|\mathbb{E}\big[\hat{V}_{t+1}(\hat{Z}_{t+1}) \mid h_t, a_t\big] - \int \hat{V}_{t+1}\, d\hat{P}_t(\cdot \mid \hat{z}_t, a_t)\right|.\end{aligned} \tag{A.6}$$

The first term of (A.6) is at most α_{t+1} by the induction hypothesis applied inside the expectation. The second term equals $\left|\int \hat{V}_{t+1}\, d\mu_t - \int \hat{V}_{t+1}\, d\nu_t\right|$ for the measures μ_t, ν_t of property (AP2), and Lemma 3.2 together with (AP2) bounds it by $\rho_{\mathcal{F}}(\hat{V}_{t+1})\, d_{\mathcal{F}}(\mu_t, \nu_t) \le \rho_{\mathcal{F}}(\hat{V}_{t+1})\, \delta_t$. Combining, the right side of (A.5) is at most $\varepsilon_t + \rho_{\mathcal{F}}(\hat{V}_{t+1})\delta_t + \alpha_{t+1} = \alpha_t$, which is the Q-inequality at t.

For the V-inequality, use that the difference of maxima is bounded by the maximum difference:

$$\left|V_t(h_t) - \hat{V}_t(\hat{z}_t)\right| = \left|\max_a Q_t(h_t, a) - \max_a \hat{Q}_t(\hat{z}_t, a)\right| \le \max_a \left|Q_t(h_t, a) - \hat{Q}_t(\hat{z}_t, a)\right| \le \alpha_t. \quad \square$$

Proof of Theorem 3.4. It suffices to prove

$$\left|\hat{V}_t(\hat{\sigma}_t(h_t)) - V_t^{\hat{\pi}}(h_t)\right| \le \alpha_t \qquad \text{for all } t,\ h_t, \tag{A.7}$$

since then, writing $\hat{z}_t = \hat{\sigma}_t(h_t)$,

$$V_t(h_t) - V_t^{\hat{\pi}}(h_t) = \big(V_t(h_t) - \hat{V}_t(\hat{z}_t)\big) + \big(\hat{V}_t(\hat{z}_t) - V_t^{\hat{\pi}}(h_t)\big) \le \alpha_t + \alpha_t,$$

using Theorem 3.3 for the first bracket and (A.7) for the second.

Claim (A.7) is proved by backward induction, and holds trivially at $t = T+1$. Assume it at $t+1$, fix h_t, and let $\hat{a} = \hat{\pi}_t(h_t)$ be the greedy action, so that $\hat{V}_t(\hat{z}_t) = \hat{Q}_t(\hat{z}_t, \hat{a})$, while by definition of the policy value $V_t^{\hat{\pi}}(h_t) = \mathbb{E}[R_t \mid h_t, \hat{a}] + \mathbb{E}[V_{t+1}^{\hat{\pi}}(H_{t+1}) \mid h_t, \hat{a}]$. Subtracting,

$$\begin{aligned}\left|\hat{V}_t(\hat{z}_t) - V_t^{\hat{\pi}}(h_t)\right| \le{}& \left|\hat{r}_t(\hat{z}_t, \hat{a}) - \mathbb{E}[R_t \mid h_t, \hat{a}]\right| \\ &+ \left|\int \hat{V}_{t+1}\, d\hat{P}_t(\cdot \mid \hat{z}_t, \hat{a}) - \mathbb{E}\big[\hat{V}_{t+1}(\hat{\sigma}_{t+1}(H_{t+1})) \mid h_t, \hat{a}\big]\right| \\ &+ \left|\mathbb{E}\big[\hat{V}_{t+1}(\hat{\sigma}_{t+1}(H_{t+1})) - V_{t+1}^{\hat{\pi}}(H_{t+1}) \mid h_t, \hat{a}\big]\right|. \qquad \text{(A.8)}\end{aligned}$$

The first term of (A.8) is at most ε_t by (AP1). The second is the IPM comparison of ν_t against μ_t evaluated on $\hat{V}_{t+1}$, hence at most $\rho_{\mathcal{F}}(\hat{V}_{t+1})\,\delta_t$ by Lemma 3.2 and (AP2). The third is at most α_{t+1} by the induction hypothesis. Summing gives $\varepsilon_t + \rho_{\mathcal{F}}(\hat{V}_{t+1})\delta_t + \alpha_{t+1} = \alpha_t$, establishing (A.7) at t. □

Proof of Corollary 3.5. Let $\Delta = \sup_h \left|V(h) - \hat{V}(\hat{\sigma}(h))\right|$, which is finite since both value functions are bounded by $\sup|R|/(1-\gamma)$. For any h and a, the discounted analogues of (A.5) and (A.6) give

$$\left|Q(h,a) - \hat{Q}(\hat{\sigma}(h), a)\right| \le \varepsilon + \gamma\left(\Delta + \rho_{\mathcal{F}}(\hat{V})\,\delta\right),$$

using (AP1) for the reward, the definition of Δ for the trajectory term and Lemma 3.2 with (AP2) for the kernel term, where the discount γ multiplies the continuation. Taking maxima over a and then the supremum over h yields $\Delta \le \varepsilon + \gamma\rho_{\mathcal{F}}(\hat{V})\delta + \gamma\Delta$, and solving gives the first bound. The same one-step expansion applied to $\hat{V}(\hat{\sigma}(h)) - V^{\hat{\pi}}(h)$, exactly as in the proof of Theorem 3.4, shows $\sup_h |\hat{V}(\hat{\sigma}(h)) - V^{\hat{\pi}}(h)|$ satisfies the same recursion and hence the same bound, and the triangle inequality doubles it. □

Proof of Proposition 3.9. Three properties must be verified.

The compression is a well-defined function of the history. By induction on t, the window w_t holds the current extended observation together with the observations, actions and rewards of the agent's eight most recent decisions, all of which are entries of h_t, and the bank satisfies $\mathcal{B}_{t+1} = \Phi(\mathcal{B}_t, F_{t+1})$ by (3.14), where F_{t+1} is a field of the observation y_{t+1} contained in h_{t+1}, and Φ is deterministic. The initial bank $\mathcal{B}_1$ is the one training left behind, which is fixed before deployment begins. Hence $\tilde{s}_t = \tilde{\sigma}_t(h_t)$ for a measurable, deterministic $\tilde{\sigma}_t$.

(AP2a): the internal state evolves in a state-like manner, exactly. Define

$$\hat{\varphi}\big((w, \mathcal{B}),\, y',\, a\big) = \Big(\text{shift}(w, y', a),\ \Phi(\mathcal{B}, F(y'))\Big),$$

where shift moves the observation of the current decision, its action a and its reward, which y' carries, into the history, drops the oldest of the eight decisions, and makes y' the current extended observation, and $F(y')$ extracts the feedback field. Both coordinates are deterministic functions of the previous internal state, the new observation and the action, so $\tilde{\sigma}_{t+1}(h_{t+1}) = \hat{\varphi}(\tilde{\sigma}_t(h_t), y_{t+1}, a_t)$ holds with equality, and the structural error in (AP2a) is zero. This is where the determinism of Φ, engineered in Section 3.3.3, is consumed, because a stochastic or history-dependent eviction rule would break the identity.

(AP2a) + (AP2b) ⇒ (AP2). Fix h_t, a_t and let μ_t^y, ν_t^y be the true and predicted next-observation distributions of Assumption 3.8. The next internal state is the deterministic image $\tilde{s}_{t+1} = \hat{\varphi}(\tilde{s}_t, Y_{t+1}, a_t)$, so the true and predicted next-state distributions μ_t, ν_t appearing in (AP2) are the pushforwards of μ_t^y, ν_t^y under the measurable map $\Lambda(y') = \hat{\varphi}(\tilde{s}_t, y', a_t)$. For the total-variation class, the measurable functions bounded by one, pushforwards are non-expansive. For any f in the class on $\tilde{\mathcal{Z}}$, the composition $f \circ \Lambda$ is measurable and bounded by one, so it belongs to the class on the observation space and

$$\left|\int f\, d\mu_t - \int f\, d\nu_t\right| = \left|\int f \circ \Lambda\, d\mu_t^y - \int f \circ \Lambda\, d\nu_t^y\right| \le d_{\mathcal{F}}\big(\mu_t^y, \nu_t^y\big).$$

Taking the supremum over f gives $d_{\mathcal{F}}(\mu_t, \nu_t) \le d_{\mathcal{F}}(\mu_t^y, \nu_t^y) \le \delta_y$. Hence (AP2) holds with $\delta \le \delta_y$. For general IPM classes, including the maximum-mean-discrepancy class that the observation term of (3.16) targets, the same implication holds with $\delta \le \Gamma_{\mathcal{F}}(\hat{\varphi})\, \delta_y$, where $\Gamma_{\mathcal{F}}(\hat{\varphi})$ is the expansion constant of the update map, as in [8].

(AP1) is Assumption 3.7 verbatim, since $z_t = \zeta_\theta(\tilde{s}_t)$ is a deterministic readout.

All conditions of Definition A.1 are met by the process $\tilde{s}_t^i$ with constants (ε, δ_y), and the theorems apply with α_t computed from those constants. □

A.4 Agent Hyperparameters

Table A.1 lists the hyperparameters of both learned agents. The reward constants and the loop guard are identical across the two, while the observation and the training procedure differ, as Section 4.1 records. In the reward (3.6), $w_3 = 1$, since the shaping term enters the sum with no coefficient of its own. The loop penalty r_l and the arrival reward r_d

are added to the hop's reward as written. The unavailable-action penalty r_u, which only DDQN incurs because RAMA selects within the permitted set, is instead stored as its own transition from the state to itself, once per refused direction, rather than added to the reward of the hop eventually taken.

Table A.1: Agent hyperparameters, read from the source and from each run's recorded configuration.

	RAMA	DDQN baseline
Architecture		
Hidden width	$d_{\mathrm{m}} = 64$	2 layers of 32 (ReLU)
Attention heads, head width	4, 16	–
Self-attention blocks	2	–
Memory-read attention steps	1	–
Feed-forward width	256	–
Dropout	0	–
Observation dimension	31	28
Action space	4 (U, D, R, L)	4 (U, D, R, L)
Trainable parameters	145,124	2,116
Memory		
Bank capacity per satellite	2,048	–
Retrieved entries	8	–
History window, decisions	8	–
Outcome encoding scale	0.1 s	–
Staleness cap	1.0 s	–
Outcome recorded for a dropped packet	0.6 s	–
Optimization		
Optimizer, learning rate	Adam [45], 10^{-3}	Adam, 10^{-3}
Replay buffer	4,096	50
Minibatch size	32	16
Gradient-norm clip	1.0	–
AIS loss weights (λ_r, λ_o)	$(0.5, 0.5)$	–
Key-encoder moving-average rate η	0.01	–
Temporal-difference loss	Huber	squared error
Bootstrap target	double Q-learning, masked	maximum of target network
Stops bootstrapping at arrival	yes	no
Target-network update	hard, every 1,000	hard, every 1,000
Gradient step frequency	every 4 steps	every 4 steps
Shared		
Discount γ	0.99	0.99
Arrival reward r_d	$+50$	$+50$
Revisit penalty r_l (`againPenalty`)	-10	-10
Unavailable-action penalty r_u	-10	-10
Reward weights (w_1, w_2, w_3, w_4)	$(20, 20, 1, 5)$	$(20, 20, 1, 5)$
Loop guard, `maxVisits`	5	5
State builder	`getDeepStateDir`	`getDeepState`
Exploration at evaluation	greedy ($\epsilon = 0$)	greedy ($\epsilon = 0$)
Offline training episode	50 s	4.0 s
Topology update interval in training	0.1 s	1 s
Movement speed-up during training	5805.44/20	5805.44/20