Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

ARTICLE TYPE

Path-based Deep Reinforcement Learning for On-board Routing in Satellite Constellation Networks

Manuel M. H. Roth[1] | **Thomas Jerkovits**[1] | **Anupama Hegde**[1] | **Thomas Delamotte**[2] | **Andreas Knopp**[2]

[1]Institute of Communications and Navigation, German Aerospace Center (DLR), Germany

[2]Chair of Signal Processing, University of the Bundeswehr Munich, Germany

Correspondence
Corresponding author Manuel M. H. Roth, Oberpfaffenhofen, 82234 Wessling, Germany.
Email: manuel.roth@dlr.de

Present address
This is sample for present address text this is sample for present address text.

Abstract

Efficient usage of available network resources is a crucial factor for broadband services in interconnected satellite constellations. To meet required quality of service standards under heavy network loads, it is essential to optimize traffic distribution among the inter-satellite links. To address this challenge, we propose an adaptive traffic engineering framework based on deep reinforcement learning. Our approach employs a path-based decision-making strategy, using a centralized agent to distribute incoming flow requests on a set of candidate paths. This method approximates optimal solutions to the multi-commodity flow problem with relatively low computational complexity, making it suitable for in-space network control despite on-board processing limitations. The performance of the proposed scheme is evaluated against state-of-the-art rule-based benchmarks in various scenarios. We quantify the impact on performance of different candidate-path sets and traffic patterns. Overall, the proposed solution presents a viable approach for optimizing flow distribution in satellite constellation networks, suitable for the integration into the controller logic of software-defined networks.

KEYWORDS
routing, traffic engineering, deep reinforcement learning, low earth orbit, satellite constellation networks, on-board network management

1 | INTRODUCTION

In recent years Low Earth Orbit (LEO) Satellite Constellation Networks (SCNs) have experienced increasing interest, in particular in the context of broadband traffic services[1]. SCN designs typically rely on Intersatellite Links (ISLs) to create mesh networks, reducing the required number of ground stations and increasing service coverage[1,2]. Satellites thus act as routing devices, which require configuration for efficient data transmission through the resulting space-borne network. However, depending on the routing approach, the traffic patterns resulting from the User Terminals (UTs) and Gateway stations (GWs) distributions on ground can cause bottlenecks. So, even if high-bandwidth ISLs are considered, congestion can decrease Quality of Experience (QoE) of users[2].

Therefore, various routing protocols tailored to the unique characteristics of Non-Terrestrial Networks (NTNs) have been proposed. For instance, 6G standardization aims to seamlessly integrate the space segment with terrestrial services[3]. To handle the increasing number of users and broadband traffic volume, novel routing approaches specifically tailored to SCNs become thus more relevant. Given that 6G aims to be an AI-native network of networks, suitable routing frameworks are of interest.

A key paradigm in this context is Software Defined Networking (SDN). In short, it offers improved flexibility through software and virtualization, scalability, fast reconfiguration, increased resiliency, and service-aware Quality of Service (QoS) handling[4,5]. However, due to the physical size of the constellation networks, significant end-to-end propagation delays of more than 100 ms are possible. Thus, distributed SDN-based approaches using in-space decision-making have been proposed to enable higher reactivity and local decision-making.

In order to maximize throughput by finding optimal routing configurations, the underlying Multi-Commodity Flow Problem (MCFP) has to be solved. It formalizes the distribution of routes for a set of commodities with varied source-destination pairs and associated demands. The corresponding computation is complex, the integer MCFP is NP-hard in

general[6,7]. Given the limited on-board processing, finding optimal solutions for in-space routing is thus impractical. As flows typically arrive concurrently and subsequently, making meaningful step-wise choices is advantageous.

We have previously proposed a Deep Reinforcement Learning (DRL)-based approach to approximate solutions to the MCFP with reduced complexity[8]. The approach is based on a locally centralized control unit which distributes incoming flows on sets of candidate paths. By maximizing expected future rewards, DRL can effectively handle the overlapping flow allocation sub-problems. Therefore, it represents a well-suited method to enable a proactive traffic distribution that anticipates potential bottlenecks. Unlike greedy approaches or heuristics, DRL can learn from experience, adapt to changing conditions, and discover non-obvious traffic patterns.

In this work, we extend and refine our previous architecture, demonstrating its enhanced performance across diverse scenarios. Key improvements include a novel method for gathering candidate paths and a more compact observation space representation, resulting in improved outcomes with smaller neural networks. Furthermore, we investigate the suitability of this approach to various heat maps and larger network topologies. The main contributions of this work are:

- We extend our previous path-based DRL approach[8] by introducing a new, simplified observation space based purely on aggregated path characteristics (e.g., minimum, maximum, average, variance, normalized path length), as well as global edge statistics (see Section 5.4). This observation space is entirely independent of the underlying graph topology, reducing complexity and improving generalization.
- We propose a new strategy for candidate path selection using four fixed path types: (i) shortest path (hop count), (ii) least-congested path (minimizing total utilization), and (iii–iv) two softmin-based selections with different variance scaling factors (see Section 5.3). This method outperforms the adaptive candidate selection used in our previous work[8].
- We conduct a comprehensive evaluation of the proposed approach across diverse network scenarios, including different traffic heat maps and larger network topologies. The results show that our method is not only competitive with state-of-the-art DRL and heuristic baselines but also resilient to shifts in traffic distribution and scalable to larger graphs.

The work is structured as follows. In Section 2, we summarize related works in the field of routing in satellite networks. The considered reference system and its relevant characteristics are provided in Section 3. The DRL background and the theoretical framework are presented in Section 4. Subsequently, we present the proposed architecture in Section 5. In Section 6, the simulator environment, comparative quantitative analyses, as well as comparisons to benchmark solutions are provided. The strengths and limitations of the approach as well as the implications for further developments are discussed in Section 7. Finally, the paper is concluded in Section 8.

2 | RELATED WORKS

For routing in wireless communications, Machine Learning (ML)-based approaches have been explored[9], often facilitated by SDN frameworks[10]. In the context of SCNs, DRL techniques have shown promise for network control. Various DRL-based routing schemes have been proposed, including hop-based Deep Q-Network (DQN) methods[11,12] and approaches approximating the MCFP[13]. However, the need to learn to generate stable, loop-free paths can pose problems in terms of scalability and robustness.

In contrast, our work focuses on a path-based approach, building upon a previously proposed method that distributes flows on candidate paths to approximate MCFP solutions[8]. While prior research focused on demonstrating the feasibility of the approach, this work concentrates on refining the design and evaluating its performance in diverse scenarios.

Other approaches, such as rule-based Linear Programming (LP) and Dynamic Programming (DP)[14], often require significant computational resources[15], making them infeasible for space networks with limited on-board processing power. To address this, various heuristics[14,16] and hierarchical architectures[17,18] have been proposed to reduce complexity. In general, system-specific assumption or pre-processing steps are oftentimes used to facilitate DRL-based solutions.

Various pre-processing concepts have been investigated, including Graph Neural Networks (GNNs), which aim to extract structural features of the network for improved decision-making[19]. These embeddings are typically passed to a DRL agent as part of its observation space[20]. However, such methods remain tightly coupled to the graph structure and often require significant compute overhead for training and inference. Moreover, they are primarily suited to static or fully known networks, aligning with the assumptions of the offline MCFP.

In this work, we target online or ad-hoc routing scenarios, where demands are not known in advance and decisions must be made with local or partial information. Here, the MCFP serves only as an upper bound, not as an achievable solution. Our approach avoids expensive GNN-based feature extraction by using a simplified observation space that is independent of the graph topology. By focusing on candidate paths and learning to allocate flows efficiently, we demonstrate a scalable, adaptable, and lightweight alternative well-suited for real-world space networks.

3 | REFERENCE SYSTEM

As a baseline for the considered ML-based method, we assume an SDN-based network control architecture for SCNs. The proposed approach is integrated within the routing logic of the controller, enabling

informed decision-making. While this investigation focuses on subsequent flow allocations, the approach can also be modified to apply proactive routing configurations.

For SCNs in particular, given their physical size, distributed in-space control has been proposed to enhance reactivity and scalability[5,2]. For this investigation we adopt an architecture where the network is subdivided into domains (or clusters) of 48 satellites with dedicated controllers. We focus on intra-domain routing, resulting in an environment consisting of 48 nodes. To assess performance in larger networks, we also consider a domain comprising 288 satellites. This is representative of a complete exemplary constellation. A 48-node domain in this reference constellation is illustrated in Figure 1.

Space segment characteristic	Example constellation
Number of satellites	288
Number of planes	12
Number of satellites per plane	24
Satellite altitude [km]	780
Orbital inclination [°]	86.4
Cross-seam planes spacing [°]	30
Co-rotating planes spacing [°]	15
Phase offset co-rotating planes [°]	7.5
Inter-plane ISL shut-down latitude [°]	80
Number of ISLs per satellite	4

TABLE 1 Summary of an exemplary polar constellation consisting of 288 satellites.

3.1 | Network Connectivity

We assume that each satellite of the constellation possesses four ISLs: two intra-plane ISLs and two inter-plane ISLs. The bandwidth and availability of all ISLs is considered equal in this work. Based on this setup, a grid network emerges. We also omit potential seams due to counter-rotating planes and inter-plane ISL shutdown latitudes.

Notably, an inherent path diversity is characteristic for larger SCNs. For multi-hop transmissions, multiple viable paths of comparable end-to-end characteristics are oftentimes available. This enables an efficient distribution of incoming flows based on an understanding of underlying traffic patterns.

The ground segment impacts the intended performance evaluation only indirectly. Although the heat maps used in this investigation correspond to systems consisting of both UTs and GWs, we focus exclusively on the sampled flow requests.

3.2 | 6G Functional Background

The deployment of 5G-RAN and 5G-core functions on ground and on-board satellites offers flexibility in designing centralized and distributed routing architectures for multi-orbit constellations. A key factor in this design is the choice of functional split[22,23], which determines the data path and computation of routing and forwarding functions in 6G-NTN networks[3]. The functional split refers to the division of responsibilities between different network entities, such as the user equipment, radio access network, and core network.

Within the standard-compliant functional split Option 2, the routing-related functionalities can be performed on a single entity or on multiple distributed entities on the ground. This makes it suitable for ML-based routing using centralized learning and decentralized control, where the central entity(-ies) have a global knowledge of the SCN. So in this functional split, a central entity can collect data from multiple satellites and use ML algorithms to optimize routing decisions.

The implementation of ML-based models for traffic engineering enables adaptive load balancing and improved utilization of the link resources. Additionally, variants of functional split Option 2[24] are being investigated to enable on-board routing, allowing traffic flows to remain within the space segment. This can help achieve lower end-to-end latency. For instance, the enhanced reactivity can be critical for supporting emergency communication services directly via satellite.

Given this background, our proposed approach is well-positioned to be of relevance to the development of 6G NTN integration. Its ability to distribute flows efficiently can help achieve the high-throughput requirements of future satellite networks.

3.3 | Multi-Commodity Flow Problem

In order to propose paths resulting in routing configurations which enable the highest possible throughput, approaches which solve or approximate solutions to the MCFP have to be formulated. The MCFP provides a framework for modeling and solving the network flow distribution when multiple traffic demands (commodities) must be routed through a network with limited capacity.

If integer flows are considered, as in the Integer Multi-Commodity Flow Problem (I-MCFP), the problem is NP-complete[7]. The Fractional Multi-Commodity Flow Problem (F-MCFP), on the other hand, can be solved in polynomial time using LP[7]. Nevertheless, for increasingly large networks, the problem size grows rapidly. Therefore, approximate solutions may be considered for large networks since complete LP solutions are costly.

As the overall objective is to support as many incoming flows as possible without saturating links, we focus on a minimization of the maximum link utilization. The *utilization* $u(i,j)$ is the total flow on the edge (i,j) divided by its capacity $c(i,j)$:

$$u(i,j) = \frac{f(i,j)}{c(i,j)} \tag{1}$$

There are other objectives which can be considered, such as minimizing the average link utilization, but we will focus on avoiding saturation as packet drops are particularly costly in SCNs.

We omit a detailed description of the MCFP and its associated linear programming constraints for static networks, as these have been covered in our prior work[8]. Instead we focus directly on dynamic networks,

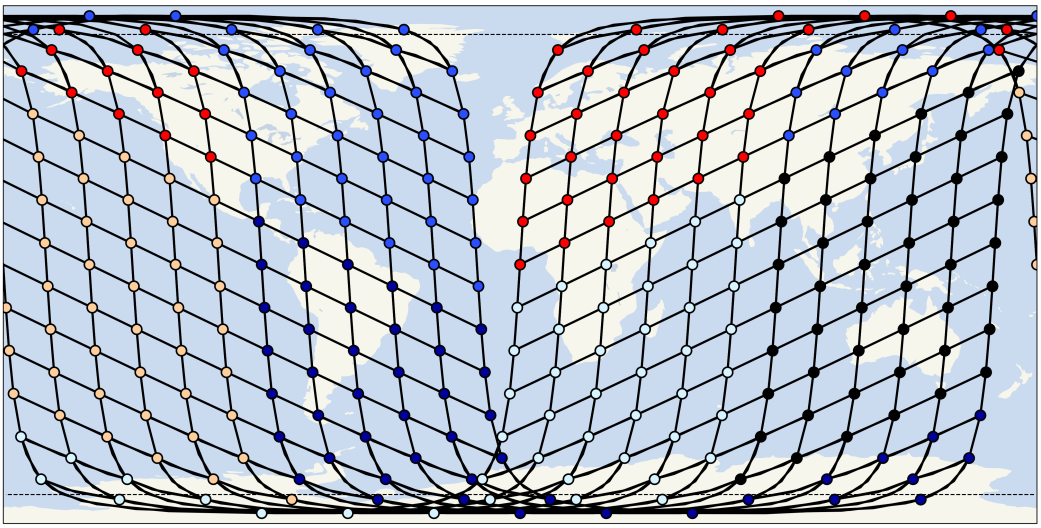

FIGURE 1 Reference satellite constellation network. A polar constellation of 288 satellites is assumed. The red area highlights a potential SDN domain of the network in the context of a distributed in-space architecture. Each domain consists of 48 satellites[21,2].

where the set of flows is subject to a temporal order. Accordingly, the graph is expanded over time steps t to model these dynamics. Thus, flows on edges depend on t:

$$f_t^k(i,j) \quad \text{for } t \in T \tag{2}$$

Similarly, the link capacity $u_t(i,j)$ is now time-dependent. For certain scenarios, the link capacities also change over time, hence we use $c_t(i,j)$.

$$u_t(i,j) = \frac{f_t(i,j)}{c_t(i,j)} = \frac{\sum_{(i,j)\in E} f_t^k(i,j)}{c_t(i,j)} \tag{3}$$

There are two options to formulate the objective function of minimizing the maximum link utilization over time. First, by extending the problem over all time steps. This represents the general approach of minimizing in all time steps:

$$\min \max_t \max_{(i,j)\in\mathcal{E}} u_t(i,j) \tag{4}$$

Or, secondly, by minimizing the time-averaged maximum link utilization over the time horizon T:

$$\min \frac{1}{T} \sum_{t=1}^{T} \max_{(i,j)\in\mathcal{E}} u_t(i,j) \tag{5}$$

As the network states are directly impacted by the route selection, any path selection effects the subsequent routing decisions. So, if we want to optimize the overall performance over time, the sub-problems of individual routing decisions overlap. For such sequential decision-making, dynamic programming is a suitable approach. To find solutions, good knowledge of the network and its state evolution is required.

We denote the state variable s_t describing the network state at time t. In addition, we consider the matrix of flows $\vec{F}_t$ which contains all flows $f_t^k(i,j)$ at time t. As before the decision variables are $f_t^k(i,j)$. Moreover, a transition function between time steps is required, which updates the current state based on the flows.

$$s_{t+1} = \Phi(s_t, f_t) \tag{6}$$

To formulate a suitable policy, the value function V_t is used to evaluate potential next states. This function considers the resulting cost, represented by a cost function C_t for the current moment in time, as well as the impact on the value of future states. So, the value function of the current state is based on the value function of subsequent states. Consequently, the computation relies on solving recursive sub-problems. Leveraging the Bellman equation enables a formulation of an optimal policy:

$$V_t(s_t) = \min_{f_t} \left[C_t(s_t, f_t) + V_{t+1}(s_{t+1})\right] \tag{7}$$

Given our objective function, the immediate cost function C_t is:

$$C_t(s_t, f_t) = \max_{(i,j)\in\mathcal{E}} u_t(i,j) \tag{8}$$

So, the for the complete time horizon from time t to T, the optimal value function representing the minimal total cost is:

$$V_t(s_t) = \min_{f_t,\ldots,f_T} \left(\max_{t'\geq t} \left[\max_{(i,j)\in\mathcal{E}} u_t(i,j) \right] \right) \tag{9}$$

This DP formulation implicitly assumes several simplifications. It does not include delay functions, queuing models, and most importantly stochastic traffic models. Therefore, formulating and solving DP for the considered SCN scenario is difficult[2].

In addition, using DP results in high computational complexity due to the state of dimensionality[15]. The state space grows exponentially with the number of input parameters. So, for large state spaces DP approaches may not be applicable in practice. Given the amount of

parameters, heuristics are typically considered to make the problem manageable[16].

3.4 | Approximating Least Congested Path Computation

To approximate the least congested path, we reformulate the classical min-max path selection problem as a differentiable min-sum formulation. For each edge $(i,j) \in \mathcal{E}$ with current utilization $u(i,j) \in [0,1]$, we define a non-linear edge weight:

$$w_\alpha(i,j) = \exp(\alpha \cdot u(i,j)),$$

where $\alpha > 0$ is a scaling parameter that controls the sharpness of the approximation.

This formulation is inspired by the *log-sum-exp* identity, which provides a smooth approximation of the maximum:

$$\log\left(\sum_i \exp(\alpha \cdot x_i)\right) \approx \max_i x_i \quad \text{as } \alpha \to \infty, \tag{10}$$

a well-known technique in optimization and machine learning [25].

Using this transformation, the cost of a path $p \subset \mathcal{E}$ becomes:

$$C(p) = \sum_{(i,j)\in p} \exp(\alpha \cdot u(i,j)),$$

and minimizing this cost approximates minimizing the maximum link utilization along the path:

$$\log C(p) \approx \max_{(i,j)\in p} u(i,j).$$

We use this formulation to compute alternative candidate paths with varying sensitivity to congestion. These paths are included in the set of routing options provided to the DRL agent, as described in Section 5.3.

4 | DEEP REINFORCEMENT LEARNING FRAMEWORK

The application of DRL techniques has gained significant attention in recent years, particularly in solving complex problems that are challenging for traditional heuristic methods and DP approaches[15,9]. For routing and network management in SCNs, DRL is a promising solution, as it enables low-complexity decision-making based on dynamically evolving network states.

However, the action space definition oftentimes poses a problem in the context of routing optimization. Given that variable action spaces are difficult to learn from, a common approach is to define the actions as the potential next hops of each node[26]. Such hop-based methods require an additional learning the pathfinding problem, which can introduce unnecessary complexity[12]. To circumvent this challenge, we formulate an approach which uses rule-based pathfinding algorithms to generate sets of candidate paths. This allows the DRL agent to focus on the primary objective of approximating the MCFP solution.

Our approach is based on the assumption that an optimal solution can be approximated by distributing incoming flows across a limited set of such pre-computed candidate paths. By learning the underlying traffic characteristics, the DRL agent is supposed to anticipate potential bottlenecks and make informed decisions, enabling the support of higher network loads and improved network performance. To this end, a suitable framework is required, which can effectively distribute flows on multiple candidate paths.

4.1 | Soft Actor-Critic

In order to comply with the path-based action space requirement and stochastic policies, we have proposed a Soft Actor Critic (SAC)[27,28] architecture[8]. SAC represents an off-policy actor-critic method which simultaneously maximizes the expected return and entropy. This promotes exploration and prevents premature convergence to sub-optimal policies. By learning a stochastic policy, value functions, and a temperature parameter, SAC balances the trade-off between exploration and exploitation. So, agents are able to navigate complex decision-making spaces. The approach is particularly useful in scenarios where convergence to sub-optimal policies is a concern, as it encourages the agent to continue exploring and learning, rather than settling on a potentially inferior solution.

It consists of an actor (policy) network, parameterized by θ_a, and two critic (Q-value) networks, parameterized by ϕ_{Q1} and ϕ_{Q2}. While the original SAC formulation[27] applied a V-critic network ϕ_V, the utilized implementation relies on the now more commonly used twin Q-critic approach[29]. This approach can help reduce overestimation of Q-values, improving stability.

The temperature parameter $\alpha_\mathcal{H}$ adjusts the weight of the entropy term. It thus controls the trade-off between reward maximization (exploitation) and entropy maximization (exploration). When the policy entropy is below the target, $\alpha_\mathcal{H}$ increases to encourage exploration. Inversely, it can also decrease when entropy is too high. $\bar{\mathcal{H}}$ denotes the target entropy, which is set to $-|\mathcal{A}|$, so the negative cardinality of the action space. The value is tuned automatically using the following objective:

$$\mathcal{J}(\alpha_\mathcal{H}) = \mathbb{E}_{a\sim\pi_{\theta_a}(\cdot|s)}\left[-\alpha_\mathcal{H} \log \pi_{\theta_a}(a|s) - \alpha_\mathcal{H}\bar{\mathcal{H}}\right] \tag{11}$$

The objective functions $J_{Q1}(\phi_{Q1})$ and $J_{Q2}(\phi_{Q2})$ for updating the soft Q-function parameters correspond to the minimization of the soft Bellman residual. So, they are a measure of the difference between the current estimate of the action-value function and its updated value based on the Bellman equation:

$$\mathcal{J}_{Q1}(\phi_{Q1}) = \mathop{\mathbb{E}}_{(s_t,a_t,r_t,s_{t+1})\sim\mathcal{D}}\left[\frac{1}{2}\left(Q(s_t,a_t;\phi_{Q1}) - \hat{Q}(s_t,a_t;\phi_{Q1})\right)^2\right] \tag{12}$$

$$\mathcal{J}_{Q2}(\phi_{Q2}) = \mathop{\mathbb{E}}_{(s_t,a_t,r_t,s_{t+1})\sim\mathcal{D}}\left[\frac{1}{2}\left(Q(s_t,a_t;\phi_{Q2}) - \hat{Q}(s_t,a_t;\phi_{Q2})\right)^2\right] \tag{13}$$

Here, $\mathcal{D}$ denotes the replay buffer. Lastly, the actor is updated to maximize the expected reward plus the entropy term. The objective of the actor network is to encourage the policy to take actions with high Q-values while maintaining diversity for exploration through entropy maximization:

$$\mathcal{J}_\pi(\theta_a) = \mathop{\mathbb{E}}_{(s_t,a_t,r_t,s_{t+1})\sim D}\Big[\mathbb{E}_{a_t\sim\pi_{\theta_a}(\cdot|s)} \big[\min(Q(s_t,a_t;\phi_{Q1}),Q(s_t,a_t;\phi_{Q2})) - \alpha_{\mathcal{H}}\log\pi(a_t|s_t;\theta_a)\big]\Big] \tag{14}$$

These objective functions highlight the self-regulating design of SAC, in terms of its exploration-exploitation trade-off. This balancing of a primary objective, i.e. reward maximization, with a regularization term which promotes diversity represents a formal relationship with the MCFP.

4.2 | Relationship to Multi-Commodity Flow Problem

Based on the presented objective functions, a formal relationship between the MCFP and the SAC framework can be established. This connection demonstrates the theoretical viability of SAC for the optimization problem at hand. While we derived why DRL-based approaches are well-suited for routing, this framework shares properties with throughput maximization under constraints, the QoS requirements.

The distinguishing feature of SAC is its entropy regularization term. It prevents the policy from becoming deterministic too quickly and encourages exploration of the state space. A similar entropy-regularization term can be introduced into the MCFP formulation to promote flow diversity.

In the context of throughput maximization in the MCFP, the objective function can be formulated as:

$$\mathcal{J}_{\text{MCFP}}(f^k) = \max \sum_{k\in K}\sum_{(i,j)\in\mathcal{E}} f^k(i,j) \tag{15}$$

Capacity and flow conservation constraints apply to this maximization. Using entropy regularization, a formal connection is visible. To control the importance of flow diversity, α_R is introduced, and we denote the flow-specific entropy term, $\mathcal{H}(f)$, by:

$$\mathcal{H}(f) = -\sum_{k\in K}\sum_{(i,j)\in\mathcal{E}} f^k(i,j)\log f^k(i,j) \tag{16}$$

Accordingly, the regularized MCFP objective function becomes:

$$\mathcal{J}_{\text{MCFP}}(f^k) = \max \sum_{k\in K}\sum_{(i,j)\in\mathcal{E}} f^k(i,j) - \alpha_R\mathcal{H}(f) \tag{17}$$

This formulation shares structural similarity with the SAC objective of the actor network, provided in Equation (14). In both cases, the primary objective, throughput or expected reward, is balanced with an entropy term that promotes diversity in solutions. The temperature parameter $\alpha_{\mathcal{H}}$ in SAC serves a similar role to α_R in the regularized MCFP, controlling the trade-off between the primary objective and solution diversity. As we have seen, the latter is an important aspect of proactive load balancing.

Overall, this formal relationship indicates that SAC has the potential to adequately approximate solutions to the MCFP. The automatic tuning of the exploration-exploitation trade-off is considered a significant advantage, as converging towards promising policies is non-trivial.

5 | PROPOSED ARCHITECTURE

Based on this SAC framework, we propose a suitable DRL architecture for the reference scenario. The components and their interaction is summarized in the diagram shown in Figure 2.

5.1 | Environment

The environment is defined over a grid network graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each edge $(i,j) \in \mathcal{E}$ has a capacity $c(i,j) > 0$ and a dynamic utilization $u(i,j) \in [0, c(i,j)]$. At each time step t, a new flow request must be routed through the network. Each request is a tuple (x_t, y_t, d_t), where $x_t \in \mathcal{V}$ is the source node, $y_t \in \mathcal{V}$ is the destination node, and $d_t \in [0,1]$ is the flow demand.

The source and destination nodes are sampled according to a fixed heat map distribution, while the demand d_t is drawn from a uniform distribution. For each request, a set of k candidate paths from x_t to y_t is pre-computed using efficient pathfinding algorithms (see Section 5.3). The agent then allocates the flow across these paths.

After allocation, link utilizations $u(i,j)$ are updated according to the assigned flow fractions. If the allocation would exceed the capacity on any edge, that portion of the flow is not routed.

Termination Condition: The episode terminates when the environment becomes saturated, i.e., when an incoming flow request cannot be routed on any of the available paths without violating link capacity constraints. This reflects a realistic failure mode in network operation due to resource exhaustion. The overall objective of the agent is to route as much demand as possible before such a failure occurs.

5.2 | Action Space

The objective of the agent is to distribute the incoming flow request among a set of pre-generated candidate paths. To achieve this, we define the action space as a k-dimensional vector over $\mathbb{R}_{\geq 0}$. Here, k represents the number of computed candidate paths.

Notably, in some cases, the set of candidate paths may contain duplicates, for instance when there are fewer than k viable paths available. However, the agent should be able to learn effective policies despite the presence of such overlapping actions. To ensure performance, a meaningful choice of k should be considered, taking into account the network

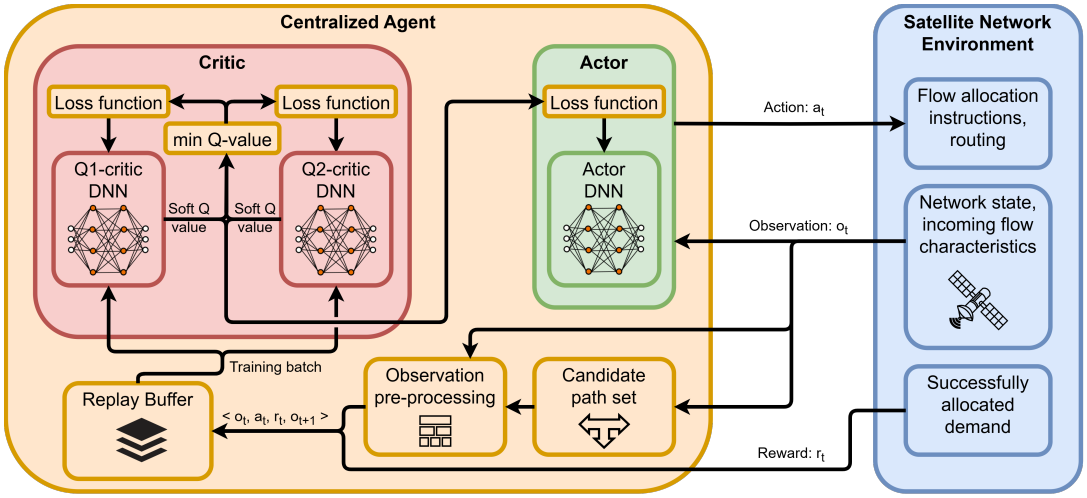

FIGURE 2 Diagram of proposed path-based routing architecture. A centralized agent using a soft actor-critic framework interacts with the satellite network environment.

topology and resulting sets of candidate paths. For the reference scenario, a constant value of $k = 4$ is assumed.

To guarantee that the agent's actions correspond to valid flow allocations, we normalize the action vector to ensure that its entries sum to 1. Such a normalization step enables the agent to output a probability distribution over the candidate paths, effectively dividing the flow proportionally among them based on the decision of the agent. For example, for $k = 4$ a potential normalized action vector of $\mathbf{a}' = [0.3, 0.0, 0.3, 0.4]$ is valid.

5.3 | Set of Candidate Paths

In order for the agent to learn effective routing policies, the design of the candidate path set is critical. These paths are pre-computed before the agent selects any action and form the available routing options for a given flow request. While the action space itself consists of continuous allocations over this set (representing how much of the flow is routed along each path), the quality and diversity of the candidate paths directly impact the agent's ability to learn meaningful decisions and observe varied reward outcomes.

An effective set of candidate paths must strike a balance between viability and discriminability. On one hand, it should include high-quality paths, such as those that minimize delay or congestion, to offer meaningful routing options. On the other hand, the paths should overlap minimally to ensure that the agent can distinguish their effects and learn from the resulting performance differences.

In our previous investigation[8], the candidate paths were selected using heuristics with varying degrees of overlap and cost formulations. This approach sometimes resulted in ambiguous behavior, as different paths could share similar characteristics or serve overlapping roles. Consequently, the agent could not reliably associate a specific allocation with a clearly defined routing objective.

In this work, we introduce a revised path selection method that improves both learning performance and interpretability. For every flow request, we construct four distinct paths based on the following criteria:

1. **Shortest Path**: path minimizing the number of hops, independent of link utilization.
2. **Least Congested Path**: path minimizing the sum of utilizations,
$$\sum_{(ij)\in p} u(i,j).$$
3. **Softmin Path** ($\alpha = 0.5$): path minimizing the sum of exponential weights,
$$w(i,j)_{0.5} = \exp(0.5 \cdot u(i,j)).$$
4. **Softmin Path** ($\alpha = 10$): path minimizing the sum of sharper weights,
$$w(i,j)_{10} = \exp(10 \cdot u(i,j)).$$

Each path is uniquely associated with a specific routing principle, allowing the agent to interpret each action dimension as targeting a distinct traffic behavior. This structure simplifies the learning task and leads to improved performance, as the agent can learn to associate traffic conditions with appropriate allocation strategies more effectively than with the more ambiguous sets used previously.

The candidate paths are recomputed at each step based on the current network state. The agent's action is a normalized vector over $[0, 1]^4$, specifying how the current demand is distributed across the four paths. This compact formulation supports flexible, adaptive behavior while maintaining semantic clarity for each component of the decision.

In the experimental results, we refer to this new scheme as **P1**, reflecting its use of four fixed and semantically distinct path types. For comparison, the earlier version based on heuristically generated paths with varying overlap and less clearly defined roles is referred to as **P2**.

5.4 | State Space

We use a compact, statistical representation of the network state, avoiding path-level structural encodings. For each of the k candidate paths, the following five features are included:

$$\min(u_p), \quad \max(u_p), \quad \text{avg}(u_p), \quad \text{var}(u_p), \quad \ell_p,$$

where u_p denotes the vector of link utilizations on the path p, and $\ell_p \in [0, 1]$ is the normalized path length (relative to the maximum possible length).

In addition, four global features are included over all network links:

$$\min(u), \quad \max(u), \quad \text{avg}(u), \quad \text{var}(u),$$

where $u \in [0, 1]^{|\mathcal{E}|}$ is the vector of current link utilizations across the graph.

Finally, the current flow demand $d_t \in [0, 1]$ is appended.

The total observation vector is real-valued, normalized, and of size

$$S = 5k + 4 + 1 = 5k + 5. \tag{18}$$

5.5 | Reward

The reward reflects the amount of demand successfully routed at each step. Let d_t be the demand of the current flow request. If the allocation is feasible (i.e., does not exceed any link capacities), the agent receives a reward equal to d_t. Otherwise, the reward is zero:

$$r_{t+1} = \begin{cases} d_t & \text{if the flow is successfully routed,} \\ 0 & \text{otherwise.} \end{cases} \tag{19}$$

This formulation encourages the agent to maximize the total routed demand over the episode. If no valid allocation exists, meaning that routing the flow would exceed link capacity on any path, the episode terminates. This reflects the saturation of the network and enforces early stopping once no further traffic can be accommodated.

Due to the step-wise nature of the task and the absence of long-term shaping rewards, a high discount factor is beneficial. It allows the agent to value long-term throughput and learn to avoid early network saturation.

Alternative reward formulations, including penalties for exceeding link capacity and exponential shaping functions, were evaluated but did not yield improved performance. The simple, demand-based reward proved most effective for encouraging stable and efficient routing behavior.

5.6 | Complexity

The complexity of the proposed approach after training can be broken down into two components: the complexity of calculating candidate paths, denoted as $\mathcal{TC}_{\text{paths}}$ and the complexity of model inference, denoted as $\mathcal{TC}_{\text{inference}}$. This relationship can be represented by:

$$\mathcal{TC}_{\text{deploy}} = \mathcal{TC}_{\text{paths}} + \mathcal{TC}_{\text{inference}} \tag{20}$$

The time complexity of the k-shortest paths algorithm can be used as a baseline for computing the candidate paths:

$$\mathcal{TC}_{\text{paths}} = O(|\mathcal{E}| + |\mathcal{V}| \log |\mathcal{V}| + k). \tag{21}$$

Here, k denotes the number of paths. $\mathcal{V}$ and $\mathcal{E}$ denote the sets of vertices and edges respectively, as before. Naturally, the path selection does not directly follow this algorithm. Instead, it is based on a mix of pathfinding algorithms which typically results in slightly higher complexity. Nevertheless, it is assumed to remain in a similar domain.

On the other hand, the inference process involves a forward pass through the actor network. The process consists primarily of matrix multiplications representing the layers of the trained neural network. Its complexity in time can be represented by:

$$\mathcal{TC}_{\text{inference}} = O\left(\sum_{\iota \in \mathcal{L}} N_n^{\iota-1} N_n^{\iota}\right) \tag{22}$$

In this context, $\mathcal{L}$ denotes the set of layers in the neural network, and N_n^{ι} represents the number of nodes or neurons in layer ι.

In contrast, the training complexity is significantly higher. It can be expressed as the product of the number of iterations, denoted as N_{its}, and the time complexity associated with forward and backward passes through both the actor network and two critic networks, denoted as $\mathcal{TC}_{\text{actor}}$ and $\mathcal{TC}_{\text{critic}}$ respectively:

$$\mathcal{TC}_{\text{training}} = O(N_{\text{its}} \left(\mathcal{TC}_{\text{actor}} + 2\mathcal{TC}_{\text{critic}}\right)) \tag{23}$$

6 | PERFORMANCE EVALUATION

To evaluate the effectiveness of the proposed DRL-based routing method, we conduct simulations on different network configurations under varying traffic loads. The agent's goal is to maximize the total amount of routed demand before the network reaches saturation. We compare its performance to baseline routing schemes and investigate the impact of different candidate path selection strategies.

6.1 | Simulator Environment

We implement the environment using the Python-based `Gymnasium` library[30]. The SAC algorithm is implemented using the `Stable-Baselines3` library[29]. Additionally, we utilize the `NetworkX` library[31] to represent and manipulate the underlying graph structure, as well as to perform pathfinding routines.

We consider two grid network configurations with $8 \times 6 = 48$ and $12 \times 24 = 288$ satellite nodes, respectively. The 48-node setup is supposed to represent the domain shown in Figure 1. Flows are generated

Hyperparameter	Value
Algorithm	SAC
Policy	MlpPolicy
State Space Size	$S = 5k + 5 = 25$ (see (18))
Policy Architecture	$[64, 64]$
Critic Architecture	$[64, 64]$
Action Space Size	$k = 4$
Activation Function	ReLU
Learning Rate	$1 \cdot 10^{-3}$
Buffer Size	1,000,000
Batch Size	512
Target Update Rate	0.005
Discount Factor	0.999
Entropy Coefficient	Automatic

TABLE 2 Hyperparameters of the SAC model using the simplified statistics-based observation space with $k = 4$ pre-computed paths.

by sampling source and destination nodes according to the heat map shown in Figure 3. The demand d_t is drawn uniformly from the interval $[0.1, 0.5]$.

At each time step, a new flow is sampled and a set of $k = 4$ candidate paths is computed based on the current network state, as described in Section 5.3. The agent observes path statistics and global utilization features as defined in Section 5.4, and selects a flow allocation accordingly.

Flow requests are continuously generated until the environment terminates due to saturation, i.e., when no feasible path allocation exists without exceeding link capacities (see Section 5.1). Thus, there is no fixed number of flows per episode. Instead, the agent's goal is to route as much total demand as possible before failure.

The used hyperparameters for training are summarized in Table 2.

To investigate how different traffic patterns influence learning and performance, we compare two heat map configurations for the 8×6 grid. The first heat map emphasizes the central region of the network, encouraging flows between nodes in the middle area. The second heat map concentrates traffic at the edges, particularly in the top and bottom rows. The heat maps are visualized in Figure 3 and Figure 4. These configurations are used to train and evaluate separate agents, allowing us to analyze generalization capabilities across spatially distinct traffic distributions.

6.2 | Benchmark Comparison

To evaluate the performance of the proposed DRL-based method, we compare it to several routing baselines in a Monte Carlo simulation setting. For each run, flow requests are generated until the network reaches saturation, and the total successfully routed demand is recorded. This setup reflects the primary optimization goal of maximizing throughput before link overflow occurs (see Section 5.5).

The considered methods are:

- **SPF**: Shortest-path-first, based on hop count.

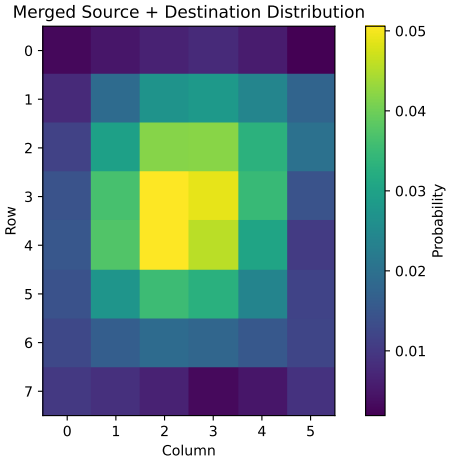

FIGURE 3 Centered heat map (around center) for the 8×6 grid. Brighter areas indicate higher probability of node selection for source and destination sampling.

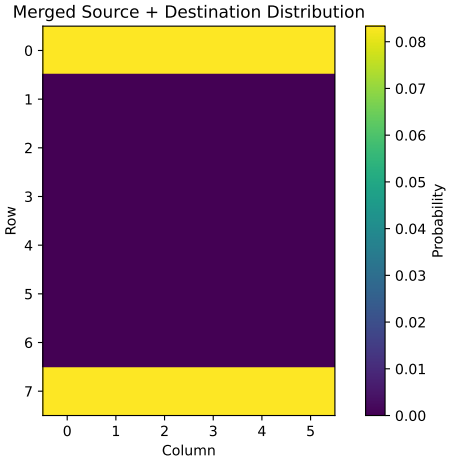

FIGURE 4 Edge-focussed heat map for the 8×6 grid.

- **LCF**: Least-congested-first, minimizing the sum of current link utilizations.
- **SM** $\alpha = 0.5$ and **SM** $\alpha = 10$: Softmin routing strategies based on exponential cost transformations (see Section 5.3).
- **EMP P1**: Equal multi-path routing using candidate path set P1.
- **SAC P1**: DRL agent trained using the candidate path set P1.
- **SAC P2**: DRL agent trained using the candidate path set P2.

Figure 5 shows the distribution of total routed demand over 10,000 randomly sampled flow sequences on the 8×6 grid. The SAC agent using the interpretable candidate path set (P1) consistently outperforms all baseline approaches, including the rule-based methods and the legacy DRL setup (P2). This highlights the effectiveness of structured path selection for enabling better routing decisions under high load.

To further compare both SAC agents during training, Figure 6 shows the evolution of the average reward over time. Both agents were trained on the 8×6 grid using a centered traffic heat map. SAC P1 consistently achieves higher reward throughout training. This may be due to the higher consistency in performance across its candidate paths, which can simplify the learning task. SAC P1 enforces semantic separation between paths (e.g., shortest, least-congested, softmin variants), which may introduce more variability in action outcomes. While P2 is explicitly designed to reduce path overlap, it can be more challenging to learn an effective allocation policy in such a structured setting. Nonetheless, P1 offers better interpretability and shows strong inference performance under varying conditions.

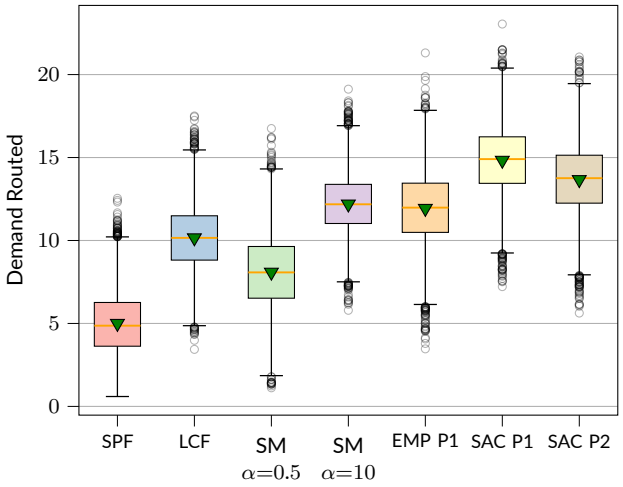

FIGURE 5 Monte Carlo simulation: comparison of total routed demand for 8×6 grid.

6.3 | Heat map Comparison

To evaluate the robustness of the trained policy under different traffic patterns, we compare the performance of two SAC agents trained on different heat maps, while testing both on the same edge-focused traffic scenario. The experiment is conducted on an 8×6 grid network.

During evaluation, source and destination nodes are sampled exclusively from the top-most and bottom-most rows of the grid, simulating traffic concentrated at the network edges. **"SAC P1 edge"** was trained on this same edge-focused heat map (see Figure 4), whereas **"SAC P1 uniform"** was trained on a uniform heat map, where all nodes are equally likely to act as source or destination. Both models use the **P1** (see Section 5.3) candidate path selection method, which combines shortest path, least-congested path, and two softmin-based path variants.

As shown in Figure 7, both SAC agents achieve comparable routing performance. This indicates that the learned policy generalizes well to new traffic distributions and is resilient to changes in the training

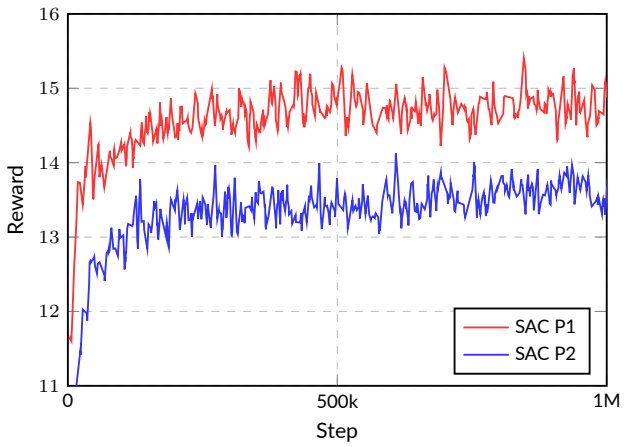

FIGURE 6 Training curves showing the average reward over time for SAC agents using candidate path sets P1 and P2 on the 8×6 grid with a centered traffic heat map. SAC P1 consistently achieves higher reward during training, suggesting that the diverse set of candidate paths may simplify the learning task.

heat map. The agent's ability to maintain performance under shifted traffic conditions supports its scalability and applicability to dynamic or unforeseen traffic scenarios.

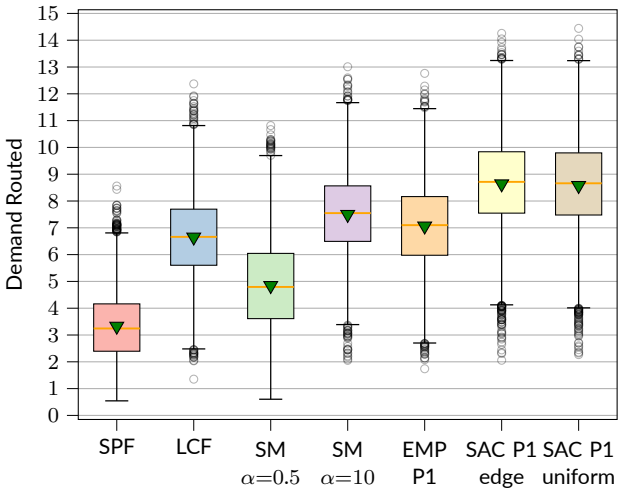

FIGURE 7 Monte Carlo simulation on an 8×6 grid with edge-focused traffic (top and bottom rows only). "SAC P1 edge" is trained on the same edge-only heat map (see Figure 4), "SAC P1 uniform" is trained on a uniform heat map.

6.4 | Larger Networks

While the main evaluation focused on routing within SCN sub-networks, the proposed DRL-based method is equally applicable to ground-based

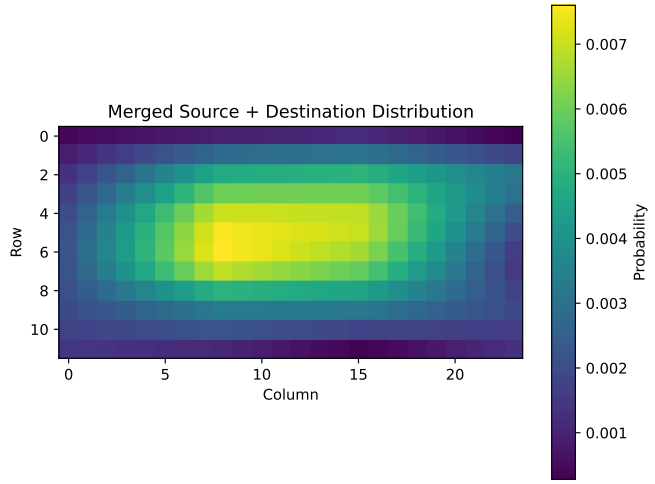

FIGURE 8 Traffic heat map used for the 24×12 grid. Brighter areas indicate higher probability of node selection for source and destination sampling.

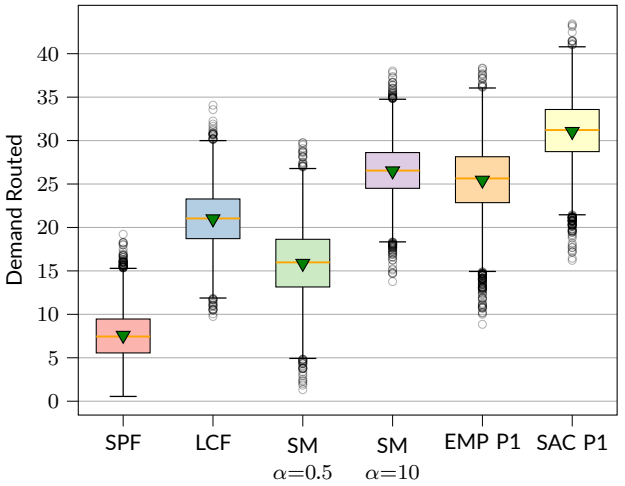

FIGURE 9 Monte Carlo simulation: comparison of total routed demand for 24×12 grid.

or centralized control of the full constellation. To assess its potential in larger-scale domains, we extend our evaluation to a 24×12 grid topology, comprising 288 nodes. The corresponding heat map is shown in Figure 8.

It is important to note that this configuration does not replicate the precise layout of a Walker-type or polar-orbit constellation, as the orientation and directionality of ISLs are simplified. However, the grid structure provides a relevant approximation of path diversity, scale, and congestion effects in large space-borne or hybrid networks.

Figure 9 shows the total routed demand until saturation for different baseline and DRL methods, using the same Monte Carlo sampling procedure as in the smaller network. The SAC agent trained with the interpretable candidate path set (P2) again achieves the highest throughput, confirming that the approach scales well with network size and maintains a performance advantage even under increased path complexity and capacity pressure.

7 | DISCUSSION AND IMPLICATIONS

In the following, we will summarize the observed strengths and limitations of the proposed approach and briefly discuss potential implications for applications in real-world systems.

7.1 | Strengths of Proposed Method

The proposed method exhibits several key strengths that make it an attractive solution for flow distribution in SCNs. Notably, the approach demonstrates flexibility in learning complex traffic patterns and distributing flows with foresight, allowing it to anticipate and mitigate potential bottlenecks. The proposed method consistently outperforms state-of-the-art methods, including the least-congested path approach.

Furthermore, the improved performance achieved by the method could not be replicated by mimicking its behavior with rule-based decision-making. This demonstrates that the agent is actually learning effective policies and not generic strategies.

In addition, the observed complexity of the approach, even during training, poses no significant computational challenges. Given that we were able to achieve improved performance with smaller models indicates that further reduction of processing needs may be possible. Moreover, the method seamlessly integrates into SDN frameworks, making it a viable and practical solution for real-world deployment.

7.2 | Limitations of Proposed Method

A key assumption underlying the approach is that distributing flows among a set of candidate paths will result in near-optimal performance. While the results show the capacity to route more demands, the given sets of paths are expected to represent a ceiling to the effectiveness of the approach. Naturally, as the analysis indicates, the potential improvement is also strongly scenario-dependent.

The learning of underlying traffic patterns is currently tied to static heat maps, which may not capture the full complexity of real-world traffic dynamics. Given the movement of the constellation and dynamically changing network topology, satellites may be faced with diverse traffic scenarios. More complex, time-dependent environments have to be considered. While the results demonstrate some degree of robustness and generalization, the effectiveness of the method may decrease for such environments and related models.

7.3 | Real World Application

Although the considered approach can be applied for ground-based control, the availability of additional computational resources in such scenarios may reduce the relevance of low-complexity models. Nevertheless, the demonstrated efficiency and scalability of the approach make it relevant for both in-space as well as on ground applications. Moreover, similar methods may be of interest for different networks with comparable characteristics.

Notably, this study demonstrates that significant improvements in decision-making can be achieved with relatively small observation spaces and models, highlighting the potential for efficient and effective optimization in satellite constellation networks. Still, the robustness of the model to handle real-world scenarios, including more complex environments with time-dependent network dynamics, remains to be evaluated in detail. Extensions of the method may also benefit from incorporating historical trends as inputs to improve the accuracy of traffic predictions. Despite these challenges, the generic nature of the method and inherent flexibility offer presents opportunities for further research and development.

8 | CONCLUSION

In this work, we extend the design of a SAC-based DRL approach for approximating solutions to the MCFP in SCNs, while maintaining relatively low computational complexity. Following a path-based decision-making strategy, a centralized agent effectively distributes incoming flow requests among a set of candidate paths. The approach is able to outperform rule-based state-of-the-art approaches in various scenarios. The provided analyses provide valuable insights into the architecture of the method, including the composition of path sets. Moreover, we demonstrate that the general performance improvement persists also in larger networks. Thus, while the method is primarily designed to integrate into the routing logic of in-orbit distributed SDN, it is also viable for global ground-based control entities. Overall, the combination of flexibility, adaptability, and performance makes the proposed method a promising solution for optimizing flow distribution in SCNs.

ACKNOWLEDGMENT

This work has been supported by the Space for 5G & 6G programme of the European Space Agency (ESA), activity code 5A.079, https://connectivity.esa.int/projects/aicoms. Responsibility for the contents of this publication rests with the authors.

REFERENCES

1. Butash T, Garland P, Evans B. Non-Geostationary Satellite Orbit Communications Satellite Constellations History. *International Journal of Satellite Communications and Networking*. 2021;39(1):1–5. doi: 10.1002/sat.1375
2. Roth M, Brandt H, Bischl H, Piñas DF, Acar G. IDLB: An SDN-based Load-balancing Routing Protocol for Autonomous Satellite Constellation Networks. 2025
3. 3GPP . Specification # 38.821. Tech. Rep. 3GPP TR 38.821, Technical Specification Group Radio Access Network; 2023.
4. Bertaux L, Medjiah S, Berthou P, et al. Software Defined Networking and Virtualization for Broadband Satellite Networks. *IEEE Communications Magazine*. 2015;53(3):54–60. doi: 10.1109/MCOM.2015.7060482
5. Bannour F, Souihi S, Mellouk A. Distributed SDN Control: Survey, Taxonomy, and Challenges. *IEEE Communications Surveys Tutorials*. 2018;20(1):333–354. doi: 10.1109/COMST.2017.2782482
6. Papadimitriou CH, Steiglitz K. *Combinatorial Optimization: Algorithms and Complexity*. Courier Corporation, 1998.
7. Larsson C. *Design of Modern Communication Networks: Methods and Applications*. Academic Press, 2014.
8. Roth MMH, Jerkovits T, Hegde A, Delamotte T, Knopp A. Deep Reinforcement Learning for Adaptive Traffic Engineering in Satellite Constellation Networks. In: *2025 12th Advanced Satellite Multimedia Systems Conference and the 18th Signal Processing for Space Communications Workshop (ASMS/SPSC)* 2025:1–8
9. Zhang C, Patras P, Haddadi H. Deep Learning in Mobile and Wireless Networking: A Survey. *IEEE Communications Surveys & Tutorials*. 2019;21(3):2224–2287. doi: 10.1109/COMST.2019.2904897
10. Xie J, Yu FR, Huang T, et al. A Survey of Machine Learning Techniques Applied to Software Defined Networking (SDN): Research Issues and Challenges. *IEEE Communications Surveys & Tutorials*. 2019;21(1):393–430. doi: 10.1109/COMST.2018.2866942
11. Zuo P, Wang C, Yao Z, Hou S, Jiang H. An Intelligent Routing Algorithm for LEO Satellites Based on Deep Reinforcement Learning. In: *2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall)* 2021:1–5
12. Hegde A, Roth M, Bischl H. Routing in Low Earth Orbit Satellite Networks: Constrained DQN-Based Approach. In: *2025 IEEE Wireless Communications and Networking Conference (WCNC)* 2025:1–6
13. Tsai KC, Fan L, Wang LC, Lent R, Han Z. Multi-Commodity Flow Routing for Large-Scale LEO Satellite Networks Using Deep Reinforcement Learning. In: *2022 IEEE Wireless Communications and Networking Conference (WCNC)* 2022:626–631
14. Salimifard K, Bigharaz S. The Multicommodity Network Flow Problem: State of the Art Classification, Applications, and Solution Methods. *Operational Research*. 2022;22(1):1–47. doi: 10.1007/s12351-020-00564-8
15. Sutton RS, Barto AG. *Reinforcement Learning: An Introduction*. MIT press, 2018.
16. Grislain P, Pelissier N, Lamothe F, et al. Rethinking LEO Constellations Routing with the Unsplittable Multi-Commodity Flows Problem. In: *2022 11th Advanced Satellite Multimedia Systems Conference and the 17th Signal Processing for Space Communications Workshop (ASMS/SPSC)* 2022:1–8
17. Ali RE, Erman B, Baştuğ E, Cilli B. Hierarchical Deep Double Q-Routing. Tech. Rep. arXiv:1910.04041, arXiv; 2020.
18. Cigliano A, Zampognaro F. A Machine Learning Approach for Routing in Satellite Mega-Constellations. In: *2020 International Symposium on Advanced Electrical and Communication Technologies (ISAECT)* 2020:1–6
19. Geyer F, Carle G. Learning and Generating Distributed Routing Protocols Using Graph-Based Deep Learning. In: *Proceedings of the 2018 Workshop on Big Data Analytics and Machine Learning for Data Communication Networks*Big-DAMA '18. Association for Computing Machinery 2018; New York, NY, USA:40–45
20. Wang H, Ran Y, Zhao L, Wang J, Luo J, Zhang T. GRouting: Dynamic Routing for LEO Satellite Networks with Graph-based

Deep Reinforcement Learning. In: *2021 4th International Conference on Hot Information-Centric Networking (HotICN)* 2021:123–128
21. Met Office . *Cartopy: A Cartographic Python Library with a Matplotlib Interface*. Exeter, Devon: 2010.
22. Campana R, Amatetti C, Vanelli-Coralli A. RAN Functional Splits in NTN: Architectures and Challenges. 2023
23. 3GPP . Specification # 23.501. Tech. Rep. 3GPP TS 23.501, Technical Specification Group Services and System Aspects; 2025.
24. 3GPP . Specification # 23.700-29. Tech. Rep. 3GPP TR 23.700-29, Technical Specification Group Services and System Aspects; 2024.
25. Bishop CM, Nasrabadi NM. *Pattern Recognition and Machine Learning*. 4. Springer, 2006.
26. Boyan J, Littman M. Packet Routing in Dynamically Changing Networks: A Reinforcement Learning Approach. In: *Advances in Neural Information Processing Systems*. 6. Morgan-Kaufmann 1993.
27. Haarnoja T, Zhou A, Abbeel P, Levine S. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. 2018
28. Haarnoja T, Zhou A, Hartikainen K, et al. Soft Actor-Critic Algorithms and Applications. 2019
29. Raffin A, Hill A, Gleave A, Kanervisto A, Ernestus M, Dormann N. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research*. 2021;22(268):1–8.
30. Towers M, Kwiatkowski A, Terry J, et al. Gymnasium: A Standard Interface for Reinforcement Learning Environments. 2024
31. Hagberg A, Swart PJ, Schult DA. Exploring Network Structure, Dynamics, and Function Using NetworkX. Tech. Rep. LA-UR-08-05495; LA-UR-08-5495, Los Alamos National Laboratory (LANL), Los Alamos, NM (United States); 2008.