Verifying Behavior of Reinforcement Learning Agents for Network Slice Admission Control

Jean Pierre Asdikian*(1), Alaa Amro*(2), Louma Mehyeddine(2), Carlos Natalino (3), Ihab Sbeity(2), Guido Maier(1), Paolo Monti, Sebastian Troia(1), Omran Ayoub(4)

(1) Politecnico di Milano, Italy, (2) Lebanese University, Lebanon, (3) Chalmers University of Technology, Sweden, (4) University of Applied Sciences and Arts of Southern Switzerland, Switzerland

***Abstract*—Reinforcement Learning (RL) has emerged as a powerful tool for automating complex network management tasks, yet its lack of transparency and black-box nature hinder trust and adoption in operational environments. In this work, we focus on explaining the behavior of an RL agent applied to the problem of network slice admission control. We present a framework that integrates three key components: a Deep Reinforcement Learning (DRL) agent for admission control, an Integer Linear Programming (ILP) model for network slice embedding, and an explanation module for interpreting the DRL agent's policies, namely Shapley Value Explainable Reinforcement Learning (SVERL). Our analysis aims gives particular attention to cases where the RL agent rejects admitting a network slice request despite sufficient network capacity to provision it, and investigates whether explanations can be used to verify and validate the agent's behavior prior to deployment approval. Experimental results reveal that the agent's decisions are primarily influenced by substrate network conditions such as congestion, rather than by the intrinsic characteristics of slice requests. While this conservative policy prevents overload, it also leads to overly cautious rejections. Importantly, the proposed explanation framework provides operators with actionable insights to scrutinize, validate, and refine RL-driven policies before operational deployment.**

***Index Terms*—Reinforcement Learning, Admission Control.**

I. Introduction

Reinforcement Learning (RL) offers a powerful approach to automating various network resource allocation tasks including network slicing, where efficient allocation of bandwidth and compute resources is vital to meet diverse Quality of Service (QoS) demands [1], [2]. By interacting with the network environment and receiving reward feedback, RL agents learn adaptive policies that enable more intelligent allocation strategies than static or model-based methods [3], [4].

One key application of RL in network management is Admission Control (AC), the task of deciding whether a Network Slice Request (NSR) should be admitted or rejected under current and anticipated network conditions [1], [5]. AC regulates network load by verifying available bandwidth, buffers, and existing QoS commitments, ensuring that ongoing sessions are not degraded by new arrivals. Traditional AC methods, including heuristic and Integer Linear Programming (ILP)-based approaches, often suffer from static assumptions and limited foresight. In contrast, RL-based AC mechanisms can learn policies that balance immediate network states with long-term resource sustainability, enabling more adaptive and proactive admission strategies [6]–[9]. This leads to more informed admission decisions and a more efficient structuring of the overall resource allocation landscape.

Despite their strong performance, RL-based approaches face a critical limitation, characterized the lack of transparency in their decision-making [10]–[12]. In the context of AC, agents may learn policies that optimize long-term utility, such as rejecting NSRs to preserve resources for future ones, but the rationale behind individual decisions often remains opaque. For instance, rejecting an NSR despite sufficient capacity or adopting a non-intuitive allocation strategy that improves performance but lacks immediate clarity.

This black-box nature poses challenges for network operators, who may need to verify RL behavior, i.e., assess whether the learned policies and resulting actions are consistent with operational objectives, prior to deployment and therefore require interpretable actions to maintain trust and diagnose unexpected behaviors [13]. The first step toward addressing these challenges is to extract explanations that accurately characterize the model's behavior under specific circumstances, enabling operators to assess not only its networking performance but also the soundness of its decision-making. Explainable Artificial Intelligence (XAI), which is a subfield of Artificial Intelligence (AI) concerned with providing such transparency (described later in more detail), offers the means to this.

In this work, we make an attempt to interpret RL models for AC by developing a Deep Reinforcement Learning (DRL)-based AC agent equipped with Shapley Value Explainable Reinforcement Learning (SVERL). Our goal is to extract meaningful insights into the agent's behavior, with a particular focus on counter-intuitive decisions such as rejecting slice requests despite sufficient available resources in the Substrate Network (SN), a behavior we term proactive rejection. To this end, we design a framework that combines a DRL agent for admission control, an ILP model to benchmark embedding feasibility, and SVERL to generate post-hoc explanations of the agent's decisions. Our experimental results show that the agent relies predominantly on substrate network conditions, such as congestion, utilization, and topology, while assigning little importance to slice-specific requirements. This yields a conservative policy that successfully prevents overload but also

*Jean Pierre Asdikian and Alaa Amro contributed equally to this work.

978-3-903176-75-1 ©2025 IFIP

leads to unnecessarily cautious rejections of feasible requests. Importantly, the explanation framework makes these decision drivers explicit, providing operators with actionable insights to scrutinize, validate, and refine RL-driven admission control policies prior to deployment.

II. Background and Related Work

A. Background on SVERL

A widely used XAI framework is *Shapley Additive Explanations (SHAP)* [14], which builds on cooperative game theory. In this approach, input features are treated as players and the model's prediction as the payoff, with SHAP attributing the contribution of each feature to the final decision in a fair and interpretable way. Formally, SHAP computes the contribution of each feature by considering all possible subsets of features and measuring the marginal effect of adding feature to each subset. Within XAI, eXplainable Reinforcement Learning (XRL) has emerged as a specialized subfield dedicated to interpreting the decisions of RL agents. Applying SHAP directly to RL is non-trivial because, unlike supervised learning models that output a single prediction, an RL agent produces a policy, i.e., a probability distribution over possible actions. Thus, Shapley values cannot be directly attributed to individual actions. To address this, SVERL [15] introduces a characteristic value function tailored to policies. Given a policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ that outputs action probabilities, the conditioned policy is $v^\pi(C) := \pi_C(a|s) = \sum_{s' \in \mathcal{S}} p^\pi(s'|s_C)\pi(a|s')$

Here, $p^\pi(s'|s_C)$ is the probability of being in state s' given that only a subset C of features is observed, and the agent follows policy π. The characteristic value function $v^\pi(C)$ defines the conditioned policy by computing the expected policy followed by the agent, i.e., the expected probability of taking each action a when only a subset C of features is available. It captures how the policy over states shifts when only partial observations are available, accounting for correlations between features and their influence on the resulting action distribution. This extends SHAP to the RL setting. As a result, SVERL offers a more-reliable approach to explain the decision-making behavior of RL agents.

B. Related Work

Several works have addressed the problem of AC using RL to optimize different objectives such as maximizing revenue and minimizing service delays [1], [5], [6], [16]–[20]. In [1], authors propose a multi-agent DRL to jointly solve the problems of network slicing and slice AC which proved to learn more effectively the dynamics of slice request traffics by leveraging reward shaping. In [17], authors compare optimization methods with RL techniques. Their results indicate that Deep-Q Network (DQN) achieved the highest cumulative rewards, outperforming State-Action-Reward-State-Action (SARSA), Expected SARSA, and Q-Learning. In [18], authors compare threshold-based and RL-based methods to minimize blocking probability of new user equipment in wireless 5G networks, showing that Q-Learning and Deep Q-Learning policies outperform those of threshold-based policies. The work in [19] propose an optimal AC policy based on DRL algorithm and memetic algorithm which can efficiently handle the load balancing problem without affecting the QoS parameters. Moreover, in [16], authors employed model-free RL and DRL using Q-learning for the problem of AC, maximizing the revenue from provisioned slices while avoiding overloading resources. Collectively, the aforementioned works have showcased the effectiveness of RL techniques in tackling the AC problem, often outperforming classical or heuristic approaches through their ability to adaptively learn optimal policies in complex and dynamic network environments.

Recently, works have shifted their focus to explaining RL models in the context of communication networks. For instance, [21] propose XRL-SHAP-Cache, an XRL approach for edge caching in CDNs. By integrating Deep-SHAP with a D3QN-based caching scheme, they enhance interpretability while maintaining strong performance in cache hit ratio, QoS, and space utilization. In [22], authors propose an explanation-guided XRL framework to enhance performance and interpretability of DRL-based resource allocation for 6G RAN slicing. Results show that the proposed framework improves both slice-level QoS and resource efficiency, outperforming conventional DRL approaches. The work in [2] presents SYMBXRL, an explainer for DRL that uses first-order logic to yield human-interpretable, rule-based rationales and guide action selection. Experiments on network slicing and massive MIMO report improved policy performance alongside interpretability. Moreover, [23] enhances transparency of RL agents using Decision Trees (DTs) as surrogate models. The proposed method achieves 94% accuracy in mimicking RL while improving explainability through graphical DT representations. The work in [24], authors mimic the RL agent for the problem of Routing, Modulation and Spectrum Assignment (RMSA) in optical networks using an XGB model and then apply SHAP to extract explanations from trained RL agents.

While these studies advance XRL in various networking applications, the explainability of RL agents for the problem of network slice AC remains largely unexplored. Our work addresses this gap by focusing on XRL for slice AC. In particular, we introduce a framework that combines an RL agent, an ILP model for benchmarking feasibility, and SVERL as an explainer, with a unique emphasis on counter-intuitive rejection cases. To the best of our knowledge, this is the first study to systematically analyze and interpret proactive rejections in slice admission, providing both methodological contributions and practical insights for operators.

III. Interpreting DRL for Admission Control

A. Objectives and Research Questions

Our work focuses on explaining the decision-making process of DRL agents used in AC for network slice allocation. Specifically, the problem of network slice AC can be formulated as follows. Given a set of incoming NSR, each with specific resource demands (virtual node CPU utilization capacity and virtual link capacity) and the current state of the network including available resources and active slices, decide

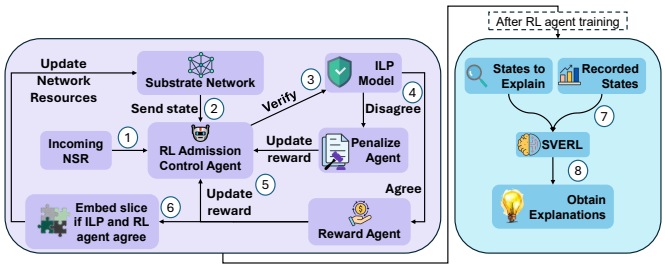

Fig. 1: XRL Pipeline using SVERL

whether to admit or reject each request, with the objective of maximizing long-term resource utilization. If a slice is admitted, its virtual nodes and links must be efficiently mapped onto the SN without violating resource constraints. Our study is guided by three research questions (RQs): *(RQ1) What features most influence the agent's decisions?*; *(RQ2) Why does the agent reject requests that appear feasible?* and *(RQ3) Can explanations of the RL agent's behavior be used to verify and validate its actions prior to deployment approval?*

B. Framework

Our proposed framework integrates three core components: A DRL agent for admission control, ILP model for slice embedding, and an XAI module (or, explainer), as illustrated in Fig. 1. The task of the DRL agent is to make dynamic slice AC decisions. Specifically, when an NSR arrives, the DRL agent evaluates the current state of the network and the resource demands of the incoming slice and takes a decision on whether to admit or deny the NSR. Upon admitting a request, the ILP model validates the feasibility of embedding requests optimally. We adopt an ILP-based Virtual Network Embedding (VNE) solver attempts to find an optimal mapping of the slice onto the network by solving a resource allocation problem that minimizes resource usage while satisfying constraints. Successful embeddings update the SN state to reflect new resource allocations. Finally, the explainer provides interpretable insights into the decision-making process of the DRL agent.

C. Methodology

To realize the described framework, we adopt an A2C RL agent that formulates slice admission as a binary decision problem (admit or reject). The agent's state representation combines real-time SN status with NSR characteristics at a given time t, sampled iteratively from batches of training slices. For each state, we define the following features: *occupied cpu*, *occupied bw*, *free cpu*, *free bw*, *links embedded*, *nodes embedded*, *number of nodes*, *number of links*, *slice cpu request* and *slice bw request*. We additionally include standard degree, average degree, average path length and standard path length features, elaborated in [25]. The reward function, carefully calibrated through sensitivity analysis, rewards successful slice embeddings proportionally to their revenue-to-cost ratio, penalizes invalid acceptance attempts where ILP embedding fails, and incentivizes correct rejections. The reward function is shaped around the feedback loop from the ILP. When both the agent's admission control decision (**AC=True**) and the ILP's validation of the virtual network embedding (**VNE=True**) agree, the agent receives a reward equal to the ratio $(\text{revenue}/\text{cost})_{\text{NSRi}}$. If the agent admits a request (**AC=True**) but the ILP finds it infeasible (**VNE=False**), it is penalized with a negative reward α (between -1 and 0). Conversely, when the agent rejects a request that the ILP could have provisioned (**AC=False**, **VNE=True**), the reward is zero. Finally, if both the agent and the ILP agree on rejection (**AC=False**, **VNE=False**), the agent receives a positive reward β (between 0 and 1).

The A2C DRL agent is complemented with a Long Short-Term Memory (LSTM) front-end network. The choice of LSTM reflects the temporal nature of AC: as slice requests arrive sequentially, the substrate state evolves over time, and decisions depend not only on the current feature vector but also on recent history. The LSTM captures these sequential patterns by maintaining a hidden state, enabling differentiation between transient resource fluctuations and sustained congestion. This temporal encoding is the complemented by the dense layers, which then abstract the representation into higher-level features before branching into the actor and critic heads. Both actor and critic share the same LSTM network.

We adopt a node–link ILP formulation to jointly embed virtual nodes and links while ensuring mapping constraints are met. Specifically, we generate slice requests with different resource requirements throughout our experiments with the objective function of minimizing the number of SN links utilized per NSR, defined in equation 1. We use binary assignment variables $\mu_{vp} \in \{0,1\}$, where $\mu_{vp} = 1$ if and only if virtual node v is mapped to substrate node p, and binary routing variables $u_{ij}^{ab} \in \{0,1\}$, where $u_{ij}^{ab} = 1$ if and only if virtual link (i,j) is carried over substrate link (a,b). We also use the weights $C_v \geq 0$ and $B_{ij} \geq 0$ that quantify, respectively, the cost or demand associated with placing virtual node v.

$$\min \sum_{v \in N_V} C_v \, \mu_{vp} + \sum_{(i,j) \in L_V} B_{ij} \, u_{ij}^{ab} \tag{1}$$

The ILP model enforces the constraint that each virtual node must be mapped to exactly one substrate node with sufficient CPU capacity. Virtual links are embedded onto substrate paths whose physical links meet bandwidth constraints. A multi-commodity flow formulation ensures link continuity and capacity feasibility. Additional constraints limit the number of virtual nodes per substrate node to satisfy redundancy or isolation requirements. Ultimately, the ILP model seeks to reduce the overall consumption of critical resources, characterized by the sum of CPU and bandwidth (*bw*) capacity, across the allocated substrate nodes and links. As for the explainer, since DRL policies represented by neural networks are not directly interpretable, we execute the trained agent over multiple episodes to gather the distribution of states encountered under a stable policy, saving entire environment states to enable feature masking and environment resets for reproducibility. Using these states, we extract action probabilities from the actor network and value predictions from the critic network

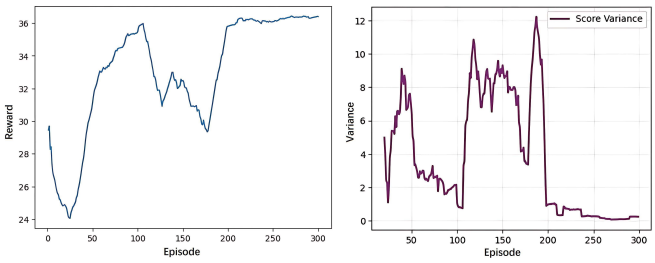

Fig. 2: RL Convergence plot and variance analysis

to build explicit policy and value mappings. Explanations are then generated by tracking the states visited by the DRL agent during and after training, with a focus on analyzing proactive rejection cases[1].

IV. Experimental Results and Interpretations

A. Experimental Settings

The network scenario we consider for our experimental setup is based on the Metro-Haul architecture [26]. As for the employed models, the actor-critic network consists of an LSTM layer with 200 units followed by two dense layers of 1024 and 512 units, and two output heads (actor and critic). We then consider 15 physical nodes, with each node CPU capacity randomly distributed between 50 and 80 units. Link capacity between the nodes is between 30 and 50 units. Each slice request is of type Ultra-Reliable Low Latency (URLLC), needing 3 nodes to be embedded. We consider 50 slices that are being embedded sequentially per episode, with each slice requiring between 5 to 20 CPU units per node and 5 to 10 BW units per link. The slices remain embedded throughout the entire episode. The RL agent learning rate is 0.001, with a discount factor of 0.85. It also includes cosine annealing with a factor of 0.01 to aid with the model convergence and help avoid local minima. The reward function is defined as follows: when the DRL agent admits an NSR and provisioning is feasible according to the ILP, the reward equals the NSR's revenue-to-cost ratio (ranging between 0 and 1). If the agent admits an NSR but provisioning is not feasible, it is penalized with -0.4. If the agent rejects an NSR despite feasible provisioning, the reward is 0. Finally, if the agent rejects an NSR and provisioning is indeed not feasible, it receives 0.4. The simulation was run for a total of 300 episodes, until convergence was reached after a specific number of episodes.

B. Experimental Results and Interpretations

RL Agent Convergence: We start with the DRL convergence plot. Figure 2 shows the DRL agent convergence plot, with an episode window of 20 (on the left). At first, the agent exhibits oscillations, characterized by the fluctuations in its reward in the first 200 episodes before stabilizing around the 300th episode where the training stopped. The convergence analysis (on the right) window confirms this stability, where we observe a spike in the reward variance around the 200 episode mark before converging and slowly plateauing near zero in the final episodes. This behavior indicates that the agent has sufficiently converged, making it suitable for state extraction, where each of the 14 previously described features is properly represented.

[1]Note that we further convert SVERL's tensor-based Shapley value computations into SHAP-compatible formats for visualization.

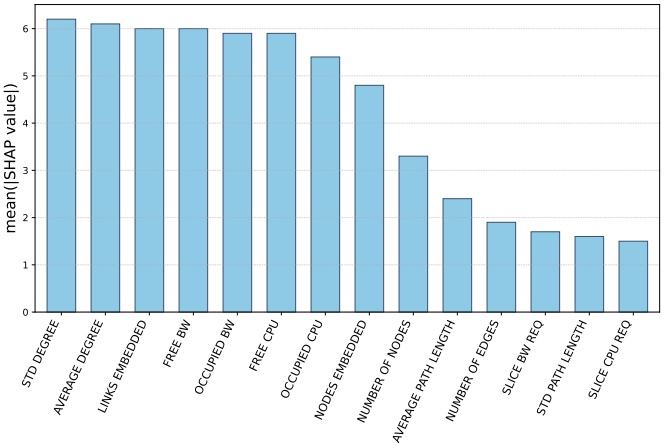

Fig. 3: Average feature impact on DRL agent's decisions

DRL Agent Performance: After its training, we analyze the DRL agent's performance by checking the number of times the agent agreed with the ILP to embed a slice, and the number of times it disagreed with it. This gives us four cases to account for: *Reject-Reject*, *Reject-Accept (Proactive rejection)*, *Accept-Reject*, *Accept-Accept*. In our scenario, 42% of all slices have been embedded successfully, while 29% of them were rejected due to both RL agent and ILP rejecting the embedded due to true lack of resources, and another 29% rejected due to proactive rejection. At first, the agent had no problem embedding the slices, but as the network became more congested, the agent became more selective in its choice of slice embedding even when the ILP suggested otherwise. Additionally, in all of our simulation runs, not a single instance of *Accept-Reject* occurred, indicating that the hard capacity constraints were always respected.

Overall Feature Importance: We now turn our attention to feature importance as computed through our proposed SVERL-based framework, focusing specifically on the *proactive rejection* cases, i.e., situations where the DRL agent rejects admitting a new NSR, while the ILP is able to identify a feasible provisioning solution. It is worth noting that in real operational settings the ILP would not even be queried once the agent rejects a request. However, in our analysis, the ILP is always prompted in order to determine whether a feasible solution exists, thereby enabling a deeper understanding of the DRL agent's decision-making process.

Figure 3 reports the mean absolute SHAP values across all Reject–Accept cases. The results reveal a clear trend: the most influential factors are features describing the *network state*, rather than those characterizing the NSR itself. For example, *slice bw req* and *slice cpu req* consistently rank among the least important features. In contrast, metrics such as *std/average degree, links embedded, free/occupied bw, and free/occupied cpu* dominate the importance ranking, while request intensity and path-length statistics exhibit relatively limited influence.

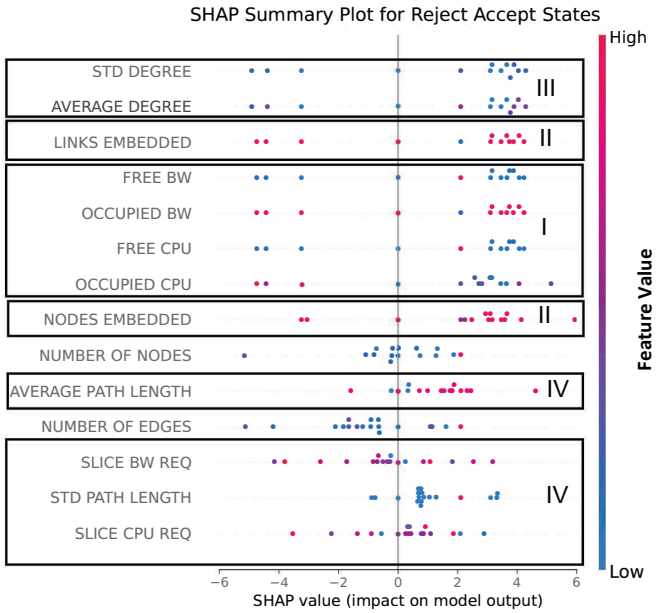

Fig. 4: SHAP Summary plot of feature impact on RL agent decision in AC

This suggests that the DRL agent places substantially more emphasis on assessing the current condition of the network than on the individual requirements of the slice. A plausible explanation is that NSRs in our setting do not exhibit dramatic variability in their resource demands; as a result, the agent learns that the feasibility of admitting a slice depends less on marginal differences in request characteristics and more on the overall availability and distribution of network resources. Put differently, since NSRs are relatively homogeneous in their specifications, the agent develops a policy that prioritizes evaluating network state features as the decisive factors, effectively making them central to its rejection decisions.

Feature Influence on Agent's Decision: We now shift our focus to analyzing the influence of individual features on the agent's decisions. As discussed earlier, we extract explanations and visualize them using a SHAP-like representation. Figure 4 presents the SHAP summary plot corresponding to the explanations obtained from the agent's proactive rejection decisions. The summary plot can be interpreted as follows: each feature is listed on the vertical axis, ordered by its overall importance, while the horizontal axis reports the SHAP value, which reflects the feature's contribution to the prediction. Each point corresponds to a single decision instance, with its position indicating the magnitude and direction of the feature's effect on the rejection outcome. The color encodes the actual value of the feature (e.g., from low in blue to high in red), thus providing insights not only into whether a feature is important, but also into how different value ranges drive the agent toward rejecting or accepting a request.

In inset I, we examine the influence of the features *free cpu*, *free bw*, *occupied cpu* and *occupied bw*. We first see that most of the decisions are characterized with consistently low values of available capacity (blue points *free cpu* and *free bw*) and high values of network utilization (red points of *occupied cpu* and *occupied bw*). The explanations show that, in the most of the cases, low values of *free cpu* and *free bw* have positive SHAP value, meaning they push towards rejecting the NSR. Conversely, high values of *occupied cpu* and *occupied bw* push towards rejecting the NSR. Together, these patterns indicate that the RL agent primarily grounds its rejection policy in detecting signs of network congestion, where CPU and bandwidth resources are already heavily consumed. This behavior aligns with the expected operational logic of avoiding overloading the substrate network. However, a few outlier cases reveal negative SHAP values even under resource scarcity, suggesting that in specific situations the agent identifies alternative feasible allocations or exhibits sensitivity to other contextual features. These exceptions highlight the complexity of the learned policy and suggest opportunities for further refinement to ensure consistent alignment with resource-aware decision-making.

In inset II, the features *nodes embedded* and *links embedded* consistently take on high values in the analyzed states, as shown by the predominance of red points. This indicates that, at the time of decision-making in these analyzed cases, the substrate network is already hosting a substantial number of active slices, resulting in significant utilization of both nodes and links. Interestingly, despite their consistently high values, these features exhibit a mix of positive and negative SHAP contributions. This means that while a high level of embedded nodes or links often pushes the agent towards rejecting an NSR, the effect is not uniform: in some contexts, the same condition instead supports acceptance. This variability suggests that the influence of these features is highly context-dependent, interacting with other network state indicators such as available CPU and bandwidth. In other words, the number of embedded resources alone is not a decisive factor; what matters is how this utilization aligns with the residual capacity of the network. This highlights that the RL agent has learned a nuanced policy, relying not on absolute utilization levels in isolation, but on their interplay with other features to balance resource efficiency and service admission.

In inset III, the topology features *std degree* and *average degree* generally take low values, which are mostly associated with positive SHAP contributions. This indicates that the RL agent is more likely to reject slices when the substrate topology involves nodes with limited connectivity. From an operational perspective, this behavior is reasonable, as embedding in sparsely connected topologies can constrain routing flexibility and resource availability, increasing the likelihood of congestion or infeasibility.

Finally, inset IV examines the features corresponding to the individual NSR. The explanations reveal that a high *average path length* is consistently associated with positive SHAP values, indicating that longer routes strongly drive the agent towards rejecting the request. By contrast, *slice bw req* and *slice cpu req* exhibit no clear patterns and rank among the least important features. Their limited impact suggests that the agent does not base decisions directly on slice resource demands, but rather considers them only in combination with broader network state indicators. This reinforces the earlier finding

that rejection decisions are primarily governed by substrate conditions rather than by the intrinsic characteristics of the requests.

Overall, the explanation analysis reveals that the RL agent's rejection behavior is largely driven by indicators of network state such as resource utilization, congestion, and topology connectivity, rather than by the specific characteristics of the slice requests. This answers *RQ2*: the agent sometimes rejects requests that appear feasible to the ILP because it has learned to prioritize global substrate conditions over individual request demands. In particular, it perceives high utilization or unfavorable topological conditions as signals of potential risk, even if, in principle, a feasible allocation exists. While this conservative policy can prevent overload and maintain stability, it may also lead to unnecessarily rejecting viable requests, highlighting a trade-off between efficiency and caution. Addressing *RQ3*, the use of SHAP-based explanations proves valuable in verifying and validating the agent's learned policy. By making its decision drivers explicit, explanations allow operators to identify cases where the agent's reasoning aligns with operational logic (e.g., avoiding congested states) and where it diverges (e.g., outlier cases under resource scarcity). Thus, explanations provide a practical mechanism to scrutinize the agent's decisions prior to deployment, increasing confidence in its behavior and enabling corrective measures where its policy does not align with intended operational objectives.

V. Conclusion

In this work, we proposed a framework that combines a deep RL agent for slice admission control, an ILP model for feasibility benchmarking, and SVERL to explain the agent's decisions. Focusing on cases where the agent rejects requests despite sufficient capacity, we found that its policy is driven mainly by substrate conditions such as congestion and topology, with little regard for slice-specific requirements. This conservative strategy helps prevent overload but also leads to overly cautious rejections. By making these patterns explicit, SVERL provides operators with actionable insights to verify, validate, and refine RL-driven policies, supporting more reliable and trustworthy network automation.

Acknowledgment

This work was partially supported by Innosuisse, the Swiss Innovation Agency, through the innovation project SUSTAINET (No. 119.588 INT-ICT), carried out within the EUREKA Cluster CELTIC-NEXT under the project SUSTAINET-Advance, and by the SNS JU ECO-eNET project under GA 10113933.

References

[1] M. Sulaiman *et al.*, "Coordinated slicing and admission control using multi-agent deep reinforcement learning," *IEEE Transactions on Network and Service Management*, vol. 20, no. 2, pp. 1110–1124, 2023.

[2] "SYMBXRL: Symbolic Explainable Deep Reinforcement Learning for Mobile Networks — hdl.handle.net," https://hdl.handle.net/20.500.12761/1888, [Accessed 07-02-2025].

[3] J. W. Nevin *et al.*, "Techniques for applying reinforcement learning to routing and wavelength assignment problems in optical fiber communication networks," *Journal of Optical Communications and Networking*, vol. 14, no. 9, pp. 733–748, 2022.

[4] L. Xu, Y.-C. Huang, Y. Xue, and X. Hu, "Hierarchical reinforcement learning in multi-domain elastic optical networks to realize joint rmsa," *Journal of Lightwave Technology*, vol. 41, no. 8, pp. 2276–2288, 2023.

[5] M. Raeis *et al.*, "Reinforcement learning-based admission control in delay-sensitive service systems," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 2020, pp. 1–6.

[6] R. Li *et al.*, "Deep reinforcement learning for resource management in network slicing," *IEEE Access*, vol. 6, pp. 74 429–74 441, 2018.

[7] Y. Liu, J. Ding, and X. Liu, "A constrained reinforcement learning based approach for network slicing," in *2020 IEEE 28th International Conference on Network Protocols (ICNP)*. IEEE, 2020, pp. 1–6.

[8] A. K. Chabi Sika Boni *et al.*, "Oneshot deep reinforcement learning approach to network slicing for autonomous iot systems," *IEEE Internet of Things Journal*, vol. 11, no. 10, pp. 17 034–17 049, 2024.

[9] W. Liu *et al.*, "Reinforcement learning-based network slicing scheme for optimized ue-qos in future networks," *IEEE Transactions on Network and Service Management*, vol. 21, no. 3, pp. 3454–3464, 2024.

[10] G. A. Vouros, "Explainable deep reinforcement learning: state of the art and challenges," *ACM Computing Surveys*, vol. 55, no. 5, pp. 1–39, 2022.

[11] A. Heuillet, F. Couthouis, and N. Díaz-Rodríguez, "Explainability in deep reinforcement learning," *Knowledge-Based Systems*, vol. 214, p. 106685, 2021.

[12] M. Arana-Catania *et al.*, "Explainable reinforcement and causal learning for improving trust to 6g stakeholders," *IEEE Open Journal of the Communications Society*, pp. 1–1, 2025.

[13] F. Ruggeri, "Explainable reinforcement learning for mobile network optimization," Ph.D. dissertation, KTH Royal Institute of Technology, 2025.

[14] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf

[15] D. Beechey, T. M. Smith, and Ö. Şimşek, "Explaining reinforcement learning with shapley values," pp. 2003–2014, 2023.

[16] W. F. Villota-Jacome, O. M. C. Rendon, and N. L. S. da Fonseca, "Admission control for 5g core network slicing based on deep reinforcement learning," *IEEE Systems Journal*, vol. 16, no. 3, pp. 4686–4697, 2022.

[17] M. A. Haque and V. Kirova, "5g network slice admission control using optimization and reinforcement learning," in *IEEE Wireless Communications and Networking Conference (WCNC)*, 2022, pp. 854–859.

[18] Y. Raaijmakers, S. Mandelli, and M. Doll, "Reinforcement learning for admission control in 5g wireless networks," in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 1–6.

[19] N. R. Chauhan and S. P. Tripathi, "Optimal admission control policy based on memetic algorithm in distributed real time database system," *Wireless Personal Communications*, vol. 117, no. 2, pp. 1123–1141, 2021.

[20] B. Bakhshi *et al.*, "R-learning-based admission control for service federation in multi-domain 5g networks," in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 1–6.

[21] X. Xu *et al.*, "Xrl-shap-cache: an explainable reinforcement learning approach for intelligent edge service caching in content delivery networks," *Science China Information Sciences*, vol. 67, 06 2024.

[22] F. Rezazadeh, H. Chergui, and J. Mangues-Bafalluy, "Explanation-guided deep reinforcement learning for trustworthy 6g ran slicing," in *2023 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2023, pp. 1026–1031.

[23] A. Botta, R. Canonico, and A. Navarro, "Explainable reinforcement learning for network management via surrogate model," in *2024 IEEE 44th International Conference on Distributed Computing Systems Workshops (ICDCSW)*, 2024, pp. 35–40.

[24] O. Ayoub, C. Natalino, and P. Monti, "Towards explainable reinforcement learning in optical networks: The rmsa use case," in *2024 Optical Fiber Communications Conference and Exhibition (OFC)*. IEEE, 2024, pp. 1–3.

[25] S. Troia *et al.*, "Admission control and virtual network embedding in 5g networks: A deep reinforcement-learning approach," *IEEE Access*, vol. 10, pp. 15 860–15 875, 2022.

[26] R. Casellas *et al.*, "Metro-haul: Sdn control and orchestration of disaggregated optical networks with model-driven development," in *2018 20th International Conference on Transparent Optical Networks (ICTON)*, 2018, pp. 1–4.