Performance Evaluation of Reinforcement Learning and Greedy Algorithms for Time Negotiation in Intent-Based Networking

Manh Cuong NGUYEN, Nagham HACHEM and Éric RENAULT
LIGM, Univ. Gustave Eiffel, CNRS, ESIEE Paris, 93162 Marne-la-Vallee, France
{firstname.lastname}@esiee.fr

***Abstract*—Intent-Based Networking (IBN) allows network users to articulate high-level operational objectives without grappling with low-level configuration complexities. However, effectively translating these abstract intents into concrete resource allocations under strict Quality of Service (QoS) constraints remains a significant scheduling challenge. This paper presents a time-slotted intent scheduling framework that introduces a flexible time-negotiation layer, facilitating controlled service delays to mitigate network congestion. We model and evaluate three distinct algorithmic paradigms: a rigid baseline scheduler, a myopic negotiation-aware greedy scheduler, and an adaptive Deep Reinforcement Learning (DRL) framework using Q-Learning. Evaluated against a large-scale real-world intent dataset, our results highlight a critical architectural trade-off: while DRL provides up to an 8.6% improvement in intent acceptance rates over the greedy approach, the greedy heuristic executes up to 1,500 times faster, establishing its viability for ultra-low latency control loops.**

***Index Terms*—Intent-Based Networking, Resource Scheduling, Deep Reinforcement Learning, Greedy Heuristics, Time Negotiation, QoS.**

I. Introduction

Next-generation networks, including 5G/6G, Software-Defined Networking (SDN), and Network Function Virtualization (NFV), are moving rapidly toward highly dynamic, service-oriented architectures. The cornerstone of modern network automation lies in decoupling high-level user objectives from low-level device configurations. Intent-Based Networking (IBN) [1] has emerged as a key paradigm in this space, enabling users to specify *what* network behaviors or outcomes they require rather than *how* those conditions should be provisioned [2].

Generally, user objectives within an IBN engine are ingested through structured primitives, natural language inputs, or standardized policy templates. The core intent engine parses these inputs into machine-readable network rules. Crucially, the system uses closed-loop verification to monitor real-time telemetry and validate that the operational state of the network continuously matches the declared intent [3]. If variations or anomalies occur, autonomous corrective loops are executed to maintain service compliance.

Despite these automation benefits, satisfying highly dynamic intents under strict resource limits and mixed QoS parameters creates complex scheduling challenges. Standard admission control mechanisms often manage resource allocations rigidly; requests that cannot be fulfilled immediately at their arrival time are discarded, leading to resource fragmentation and high rejection rates.

To address this challenge, this paper investigates the performance of temporal negotiation within IBN resource allocation, comparing traditional heuristics with intelligent, learning-based optimization. We formulate the temporal scheduling challenge as a multi-constraint problem over discrete time slots. We then compare a traditional Greedy Heuristic Scheduler against an Intelligent Reinforcement Learning (RL) Agent. Our analysis maps out the optimization gains and computational trade-offs of both approaches, offering a guide for developing modern, autonomous IBN architectures.

To fully appreciate the motivation behind this study, it is essential to understand the structural lifecycle of an "Intent" within modern architectures. Unlike static policy configurations, an intent represents an operational contract between the user and the network infrastructure. This lifecycle traditionally relies on a four-stage closed-loop automation engine: Translation, Activation, Telemetry Verification, and Optimization.

The true paradigm shift of IBN lies in its ability to support abstraction and bi-directional negotiation. When network congestion peaks, a rigid system drops incoming intents, violating business continuity. By introducing a temporal negotiation framework, the abstract nature of the intent is exploited: the user defines a maximum tolerable delay (Δ_{max}), providing the orchestration layer with an essential degrees of freedom to dynamically shift or smooth out traffic peaks. Exploring this negotiation vector is what fundamentally bridges the gap between raw hardware constraints and business-level intent requirements.

II. Related Work

While research into intent compilation, translation, and policy verification has advanced significantly, real-time resource scheduling under variable traffic load remains an operational bottleneck.

Traditional admission control frameworks typically rely on static parameters, where demands are either accepted instantly or rejected permanently [6]. This approach causes significant temporal fragmentation under bursty traffic conditions. In

979-8-3195-0167-7/26/$31.00 ©2026 IEEE

network slicing, elasticity mechanisms have been proposed to scale bandwidth or compute resources on demand [7]. However, these approaches usually operate under the assumption of fixed service start times, omitting temporal negotiation as an architectural optimization vector.

Recently, machine learning has been introduced to IBN environments to manage long-term resource orchestrations [8]. Deep Q-Networks (DQN) and Proximal Policy Optimization (PPO) algorithms have been applied to map intent requests directly to underlying virtual infrastructure [9]. These machine learning solutions handle complex, non-linear environments well, but often require significant computational power and long training periods.

This paper bridges the gap between rigid, fast heuristics and complex, learning-driven engines. By introducing controlled service shifts within a bounded temporal window, we analyze exactly when a computationally efficient greedy negotiation heuristic is sufficient, and when the overhead of a full Reinforcement Learning deployment is justified.

III. System Model

A. Intent Definition

Let $\mathcal{I} = \{I_1, I_2, \ldots, I_N\}$ be the set of incoming network intents managed by the IBN engine. Each individual intent $I_i \in \mathcal{I}$ is formally modeled as a multi-attribute tuple:

$$I_i = \langle t_a^i, d^i, b^i, \ell^i, \Delta_{max}^i \rangle \tag{1}$$

where t_a^i denotes the discrete arrival time slot, d^i represents the required service duration (in consecutive time slots), b^i is the requested bandwidth, ℓ^i signifies the normalized latency impact, and Δ_{max}^i defines the maximum tolerable negotiation delay specified by the user.

B. Resource Model and Optimization Constraints

The network infrastructure operates over a bounded scheduling horizon divided into discrete time slots $t \in \{1, 2, \ldots, T\}$. At any given time slot t, the cumulative resource allocations must not exceed the maximum physical capacities of the system.

To formalize these constraints, we introduce a binary decision variable $x_i(t) \in \{0, 1\}$, which indicates whether intent I_i is actively provisioned at time slot t:

$$x_i(t) = \begin{cases} 1, & \text{if } t \in [s^i, s^i + d^i - 1] \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where s^i is the actual allocated start time slot of intent I_i. Consequently, the time-varying system capacity is governed by the following strict upper bounds:

$$\sum_{I_i \in \mathcal{I}} x_i(t) \cdot b^i \leq BW_{max}, \quad \forall t \in \{1, \ldots, T\} \tag{3}$$

$$\sum_{I_i \in \mathcal{I}} x_i(t) \cdot \ell^i \leq LAT_{max}, \quad \forall t \in \{1, \ldots, T\} \tag{4}$$

where BW_{max} and LAT_{max} represent the total system capacity for bandwidth and latency burden, respectively.

C. Latency Impact Modeling

To seamlessly handle heterogenous QoS requirements, latency constraints expressed in milliseconds (L_{req}^i) are mapped into a normalized latency burden ℓ^i via:

$$\ell^i = \frac{L_{max}}{L_{req}^i} \tag{5}$$

where L_{max} is a constant representing the upper bound of tolerable network delay. This inverse formulation ensures that mission-critical, low-latency intents incur a higher resource cost, prioritizing them during competitive scheduling phases.

IV. Scheduling Algorithms

A. Baseline Scheduler

The baseline scheduler acts as a rigid, non-negotiable admission control model. It attempts to provision each incoming intent I_i strictly at its declared arrival time, setting $s^i = t_a^i$. Intent I_i is successfully accepted if and only if:

$$s^i + d^i - 1 \leq T \tag{6}$$

and the capacity constraints defined in (3) and (4) are satisfied throughout the entire interval $[t_a^i, t_a^i + d^i - 1]$. If a resource deficit occurs at any slot within this window, the intent is instantly rejected.

B. Negotiation-Based Greedy Scheduler

To mitigate premature rejections, the negotiation-based heuristic introduces a bounded temporal search window. If immediate allocation at t_a^i fails due to transient congestion, the scheduler iteratively evaluates alternative feasible start slots s^i within the user-defined boundaries:

$$s^i \in [t_a^i, \min(t_a^i + \Delta_{max}^i, T - d^i + 1)] \tag{7}$$

The greedy heuristic selects the earliest available time slot that satisfies the baseline resource checks, effectively shifting demands to smooth out traffic bursts.

C. Tabular Q-Learning Scheduler

To benchmark the greedy heuristic against an autonomous learning agent, we model the temporal scheduling task as a finite Markov Decision Process (MDP) solved via a tabular Q-Learning framework.

The environment state space is dynamically tracked using a sparse matrix approach, where the state key S is defined by a low-dimensional tuple:

$$S = \langle t_a^i, \mathcal{R}_{state}, b^i, d^i \rangle \tag{8}$$

where $\mathcal{R}_{state} \in \{0, 1, 2, 3, 4\}$ represents a discrete, smoothed indicator of the remaining average bandwidth across the scheduling window to accelerate convergence.

The action space A corresponds to the discrete temporal shift (delay slots) selected by the agent, bounded by the intent's tolerance: $A \in \{0, 1, \ldots, \Delta_{max}^i\}$. To maximize system throughput and reward resource efficiency, the reward function

R is explicitly aligned with the physical simulation environment as follows:

$$R = \begin{cases} \Omega_{base} + \left(\frac{b^i \cdot d^i}{BW_{max}}\right), & \text{if scheduled successfully} \\ -\Omega_{penalty}, & \text{if capacity violation or rejection} \end{cases} \tag{9}$$

where Ω_{base} is a positive baseline reward for successful placement, scaled up by the intent's resource volume footprint, and $\Omega_{penalty}$ is a strict negative penalty enforced when the agent selects an overloaded or out-of-bounds slot.

The Q-values are updated iteratively across multiple training episodes using the temporal difference Bellman equation:

$$Q(S,A) \leftarrow Q(S,A) + \alpha \left[R + \gamma \max_{A'} Q(S',A') - Q(S,A)\right] \tag{10}$$

where α is the learning rate, γ is the discount factor, and S' represents the subsequent state vector triggered by the arrival of the next sequential intent I_{i+1}. Exploration is maintained using an ϵ-greedy policy.

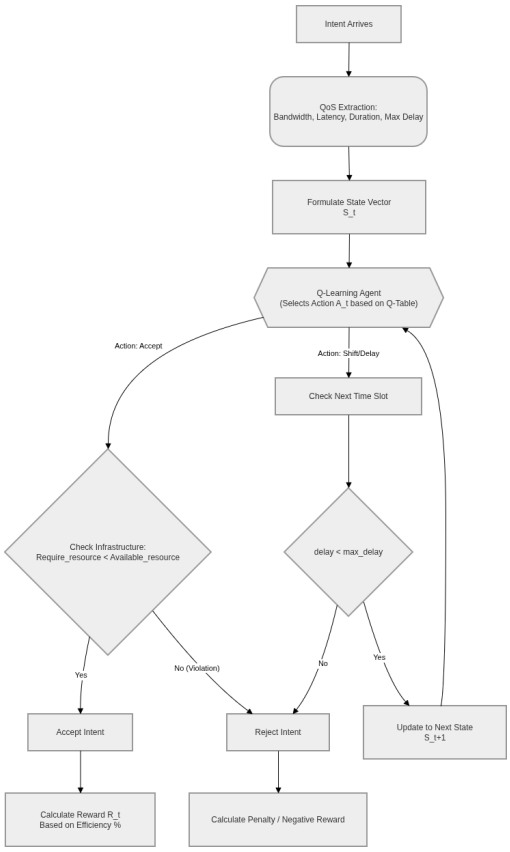


Fig. 1. Reinforcement Learning time negotiation scheduling method

D. Reinforcement Learning (RL) Scheduler

To benchmark the heuristic approaches against an intelligent optimization agent, we implement a learning-based scheduler modeled as a Markov Decision Process (MDP) (Fig. 1). The state space tracks the time-varying availability of bandwidth ($BW(t)$) and latency capabilities ($LAT(t)$) across the scheduling window. The action space defines whether to accept, reject, or temporally shift an incoming intent within its acceptable delay threshold (Δ_{max}). The reward function is formulated to maximize system throughput while explicitly penalizing QoS violations and excessive negotiation delays:

$$R = \omega_1 \cdot \mathbb{I}_{\text{accept}} - \omega_2 \cdot \Delta_s \tag{11}$$

where $\mathbb{I}_{\text{accept}}$ is an indicator variable denoting a successfully scheduled intent, $\Delta_s = s - t_a$ represents the negotiated delay penalty, and ω_1, ω_2 are weighting coefficients. Q-learning is employed to train the scheduling agent, allowing it to dynamically learn traffic smoothing patterns and anticipate optimal slot placements.

V. Simulation Results

A. Dataset Description and Simulation Framework

To evaluate the performance of the proposed scheduling paradigms under realistic network conditions, we utilize a large-scale real-world intent dataset, namely the *Sichuan Business Intent Dataset* [5]. The raw intent data is stored in a line-delimited JSON format, mapping structured intent primitives through Subject-Predicate-Object (SPO) triples. A regular-expression-based parser is engineered to programmatically extract vital Quality of Service (QoS) requirements—specifically target bandwidth (expressed in Mbps) and maximum latency constraints (expressed in ms)—from the unstructured text values.

For the simulation configuration, the extracted latency metrics are converted into normalized latency burdens to prioritize mission-critical, low-latency applications. The simulation generates a dense workload consisting of $N = 200$ dynamic intents arriving stochastically across a time-slotted horizon. To ensure statistical rigour and eliminate stochastic variance, each configuration is evaluated over $M = 30$ independent simulation runs to compute mean values accompanied by 95% confidence intervals (with a critical value of 1.96).

This analysis provides a comprehensive matrix comparison of three distinct algorithms—**Baseline**, **Greedy**, and **Q-Learning**—evaluated across two performance metrics: **Acceptance Rate (%)** and **System Efficiency (%)**. The evaluation is performed over two independent system parameters: **Time Slot Size** (x-axis, ranging from 10 to 30) and **Max Delay Tolerance** ($\Delta_{\max}$, y-axis, ranging from 2 to 10).

1. Acceptance Rate Analysis (%)

The top row of blue-gradient heatmaps displays the percentage of task acceptance.

- **Global Maxima: Q-Learning** achieves the highest overall acceptance capability, peaking at 26.1% under the most relaxed constraints (Time Slot Size $= 30$, $\Delta_{\max} = 10$). The **Greedy** algorithm follows closely with a peak of 24.7%, while the **Baseline** algorithm severely underperforms, capping out at a maximum of 17.1%.
- **Sensitivity to Time Slot Size:** All three algorithms exhibit a strong positive correlation with the time slot size.

As the slot size expands from 10 to 30, the Acceptance Rate (%) increases monotonically across almost all rows.

- **Sensitivity to Max Delay Tolerance ($\Delta_{\max}$):**
  - **Greedy and Q-Learning** are highly sensitive to delay parameters. When the delay budget is heavily restricted ($\Delta_{\max} = 2$), their acceptance rates collapse significantly (dropping to 11.9%–17.9% for Greedy and 12.7%–18.5% for Q-Learning). Relaxing $\Delta_{\max}$ allows both algorithms to dynamically optimize schedules and accept more tasks.
  - **Baseline** exhibits complete rigidity regarding $\Delta_{\max}$. At any given Time Slot Size, its acceptance rate remains virtually unchanged vertically (e.g., at Time Slot Size = 30, the rate only trivially fluctuates between 16.7% and 17.1%).

2. Efficiency Analysis (%)

The bottom row of orange/brown-gradient heatmaps evaluates system resource utilization and efficiency.

- **Global Dominance:** Both **Greedy** and **Q-Learning** vastly outperform the **Baseline** in resource efficiency. Under optimal configurations, Greedy and Q-Learning sustain efficiencies in the high 90% (peaking at 97.3% and 95.8% respectively). The Baseline algorithm, by contrast, fails to cross 68.4% and routinely drops below 60%.
- **The Inverse Scaling Trend:** Unlike the acceptance rate, system efficiency displays an inverse relationship with Time Slot Size:
  - For **Greedy** and **Q-Learning**, peak efficiencies are localized at smaller Time Slot Sizes ($10 \leq$ Size ≤ 20) when paired with high delay tolerance ($\Delta_{\max} \geq 6$). As the slot size reaches 30, efficiency drops down to the 60%–80% range.
  - For the **Baseline**, efficiency continuously degrades as the Time Slot Size moves from 10 to 30, bottoming out at 59.2%.
- **Impact of Strict Deadlines ($\Delta_{\max} = 2$):** When $\Delta_{\max} = 2$, both Greedy and Q-Learning experience severe efficiency drops at larger time slot sizes (63.4% and 65.5% respectively). The Baseline strategy remains predictably unimpacted by changes to $\Delta_{\max}$.

3. Algorithmic Trade-off Summary

Performance Rank (Acceptance & Efficiency):

$$Q\text{-Learning} \gtrsim \text{Greedy} \gg \text{Baseline}$$

TABLE I
ALGORITHMIC PROFILE SUMMARY

Algorithm	Peak Acceptance	Peak Efficiency
Baseline	17.1% (Poor)	68.4% (Low)
Greedy	24.7% (High)	97.3% (Excellent)
Q-Learning	26.1% (Optimal)	95.8% (Excellent)

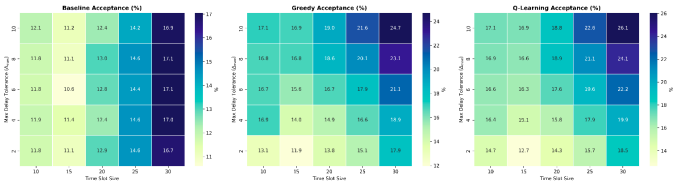


Fig. 2. Comparison of RL, baseline and greedy algorithms based on the slot size and delta max for the acceptance rate

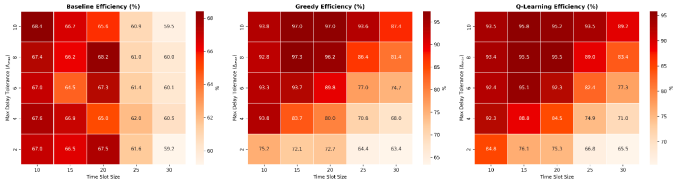


Fig. 3. Comparison of RL, baseline and greedy algorithms based on the slot size and delta max for the bandwidth efficiency

As shown in Table I, the system performance evolves significantly across the evaluated strategies. Notably, the **Peak Acceptance** metric improves progressively from the **Baseline** (17.1%) through the **Greedy** algorithm (24.7%), reaching an optimal 26.1% with **Q-Learning**. However, despite this positive trend, the overall acceptance rate remains relatively low, underscoring the necessity for further optimization in resource allocation strategies. Furthermore, while the **Greedy** approach yields a slightly higher **Peak Efficiency** (97.3%), it suffers from extreme volatility under varying constraint thresholds. **Q-Learning** strikes the best balance, delivering excellent efficiency (95.8%) alongside adaptive, stable constraint sensitivity to $\Delta_{\max}$.

Figures 2 and 3 highlight the impact of negotiation mechanisms on intent scheduling performance under varying slot sizes and delay tolerance constraints. Figures 3 and 4 reveal distinct structural patterns: Greedy and Q-Learning exhibit a smooth *diagonal wave front*, whereas the Baseline displays a rigid *vertical, erratic* profile. The diagonal gradient signifies a synergetic coupling where expanding both slot size and Δ_{max} exponentially broadens the optimization space, allowing flexible multi-period resource packing. Conversely, the Baseline's strict verticality proves its total insensitivity to Δ_{max} due to immediate scheduling demands, while its localized volatility reflects chaotic resource fragmentation governed purely by stochastic intent arrival orders.

As illustrated in Figure 4, the proposed Q-Learning (RL) framework exhibits a distinct evolutionary advantage as the maximum delay tolerance (Δ_{max}) increases. The larger slot size of 30 expands the scheduling action space, allowing the RL policy to optimize resources more effectively.

Specifically, for a slot size of 30, the RL algorithm consistently outperforms both the Baseline and Greedy heuristics across all key performance metrics:

- **Acceptance Rate:** The RL agent achieves a significant performance leap, scaling from approximately 18.5% at $\Delta_{max} = 2$ up to nearly 26% at $\Delta_{max} = 10$.

This linear progression demonstrates the agent's capacity to exploit wider time-negotiation windows. In contrast, in the smaller slot size scenario, performance plateaus abruptly around 16.5% after $\Delta_{max} = 4$.

- **Bandwidth Efficiency:** Mirroring the acceptance rate, bandwidth utilization for the RL policy climbs steadily from 65% to approximately 89% as Δ_{max} expands. This clear upward trend underscores that a larger slot size provides the RL algorithm with the structural flexibility needed to strategically pack and schedule multi-period demands without incurring severe resource fragmentation.

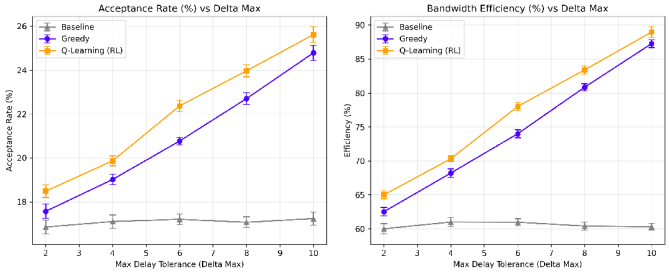


Fig. 4. Acceptance Rate improvement: Greedy vs Reinforcement Learning with time slot 30

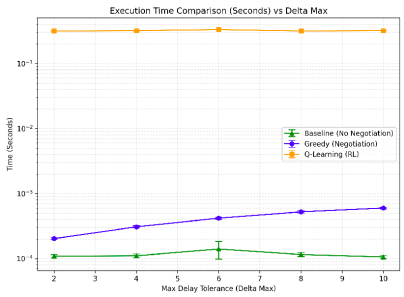


Fig. 5. Execution time: Greedy vs Reinforcement Learning with time slot 30

Figures 4 and 5 illustrate a classic engineering trade-off between optimization performance and computational efficiency with the slot size of 30. The figure 4 shows that the Reinforcement Learning (RL) approach consistently achieves a higher acceptance rate improvement than the Greedy algorithm across all delay tolerances, with its advantage widening up to roughly 8.6% at a Delta Max of 10. However, the figure 5 reveals a heavy time penalty: the logarithmic scale shows that the RL algorithm takes around 0.3 seconds to execute, making it roughly 500 to 1,500 times slower than the sub-millisecond Greedy method. Ultimately, while RL delivers superior system optimization, it requires a massive compromise in execution speed compared to the rapid but less optimal Greedy heuristic.

VI. Conclusion and Future Horizons

This paper presented a time-slotted intent scheduling framework using a flexible time-negotiation layer. Rather than treating intent timeline goals as rigid operational constraints, our framework enables controlled temporal shifts within a user-defined threshold (Δ_{max}). This transformation mitigates resource fragmentation and reduces intent rejections during transient traffic spikes.

Our comparative analysis highlights a practical engineering trade-off for next-generation IBN design. While Deep Reinforcement Learning agents achieve the highest overall acceptance rates by predicting long-term traffic distributions, they introduce significant computational latency. Conversely, the greedy negotiation heuristic delivers substantial optimization gains while maintaining sub-millisecond execution times, making it well-suited for high-throughput, real-time network controllers.

Future extensions of this work will explore:

- **Multi-Attribute Trade-offs:** Expanding negotiation loops to balance multiple resource bounds simultaneously, such as trading lower instantaneous bandwidth for reduced latency or accelerated slot placement.
- **Dynamic Windowing:** Utilizing lightweight machine learning models to dynamically adjust Δ_{max} limits based on real-time traffic volatility rather than relying on static profiles.

VII. Acknowledgments

This work is supported by French government funding within the France 2030 framework through the INFLUENCE project.

References

[1] L. Velasco, M. Signorelli, O. González de Dios, C. Papagianni, R. Bifulco, J. J. Vegas Olmos, S. Pryor, G. Carrozzo, J. Schulz-Zander, M. Bennis, R. Martinez, F. Cugini, C. Salvadori, V. Lefebvre, L. Valcarenghi, and M. Ruiz, "End-to-End Intent-Based Networking," *IEEE Communications Magazine*, vol. 59, no. 10, pp. 106–112, Oct. 2021, doi: 10.1109/MCOM.101.2100141.

[2] A. Leivadeas and M. Falkner, "A Survey on Intent-Based Networking," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 625–655, 2023, doi: 10.1109/COMST.2022.3215919.

[3] E. Zeydan and Y. Turk, "Recent advances in intent-based networking: A survey," in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, 2020, pp. 1–5.

[4] D. Konidaris, P. Soumplis, A. Varvarigos, and P. Kokkinos, "Intent-Based Resource Allocation in Edge and Cloud Computing Using Reinforcement Learning," *Algorithms*, vol. 18, no. 10, Art. no. 627, 2025, doi: 10.3390/a18100627.

[5] J. Li, S. Zou, Y. Sun et al., "Business Intent and Network Slicing Correlation Dataset from Data-Driven Perspective," *Scientific Data*, vol. 12, no. 419, 2025. Available: https://doi.org/10.1038/s41597-025-04736-z

[6] D. Gross, J. F. Shortle, J. M. Thompson, and C. M. Harris, *Fundamentals of Queueing Theory*, 4th ed. Hoboken, NJ, USA: Wiley, 2008.

[7] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network Slicing in 5G: Survey and Challenges," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 94–100, May 2017.

[8] S. Alam and W.-C. Song, "Intent-Based Network Resource Orchestration in Space-Air-Ground Integrated Networks: A Graph Neural Networks and Deep Reinforcement Learning Approach," *IEEE Access*, vol. 12, pp. 185057–185077, 2024, doi:10.1109/ACCESS.2024.3507829.

[9] R. Kozlica, S. Wegenkittl, and S. Hiränder, "Deep Q-Learning versus Proximal Policy Optimization: Performance Comparison in a Material Sorting Task," in *2023 IEEE 32nd International Symposium on Industrial Electronics (ISIE)*, Helsinki, Finland, 2023, pp. 1–6, doi: 10.1109/ISIE51358.2023.10228056.