SvedbergOpen
DISSEMINATION OF KNOWLEDGE
International Journal of Artificial Intelligence and Machine Learning
Publisher's Home Page: https://www.svedbergopen.com/
Research Paper Open Access

AI-Driven Traffic Engineering For Large-Scale Wide Area Networks Using Reinforcement Learning

Chetan Padshala

Texas A&M University, Kingsville ORCID: 0009-0002-5966-4103

Abstract
Wide area networks that connect large enterprise estates face traffic-optimization pressures that static routing protocols were never designed to absorb, including volatile demand, link degradation, and application-specific quality-of-service requirements. Conventional traffic engineering, built on fixed link metrics in OSPF and BGP or on preconfigured policy in early software-defined WANs, adapts slowly and leaves capacity unevenly used. This paper proposes a reinforcement learning framework that treats wide area traffic engineering as a sequential decision problem and drives path selection through a centralized software-defined controller. The network state is assembled from streaming telemetry, flow records, and quality-of-service counters. An agent maps this state to rerouting actions based on a reward that favors throughput while penalizing latency and loss. The controller enforces the selected actions on edge devices within a bounded action space. The framework was evaluated on an emulated dual-data-center topology with fifteen branch nodes and injected congestion. Measured against a static baseline, the learned policy reduced latency by approximately 25 to 30 percent, lowered packet loss by around 20 percent, and raised throughput by roughly 15 percent while balancing load across parallel paths. Results suggest that constrained reinforcement learning is a practical route to adaptive, operator-supervised traffic engineering at enterprise scale.

Keywords: Reinforcement Learning, Traffic Engineering, Wide Area Networks, SD-WAN, Deep Q-Network, Network Telemetry, Quality of Service, Autonomous Networking.

1. Introduction

Enterprise wide area networks have grown in both scale and volatility as cloud adoption, remote work, and real-time applications reshape traffic patterns. The routing protocols that still underpin most of these networks, principally OSPF and BGP, select paths from fixed administrative metrics and cannot react to congestion or degradation as it develops. The result is uneven bandwidth use, elevated latency on hot links, and degraded application performance during transient events.

Centralized control offered a partial remedy. Software-defined networking separated the control plane from forwarding and made global, programmatic policy possible (Kreutz et al., 2014), and software-defined WAN carried that idea to the branch edge. Most deployments, however, still execute preconfigured rules rather than decisions informed by live network state (Troia et al., 2020). The maturation of reinforcement learning has opened an alternative where an agent improves a control policy through interaction with its environment, making traffic engineering adaptive and self-optimizing (Xiao et al., 2021).

Reinforcement learning for networking has repeatedly shown promise in simulation, but three obstacles have slowed its use in production wide area networks. Learned policies are often trained on small single-domain topologies that do not reflect enterprise scale. Operators are reluctant to cede unrestricted control to an opaque agent. And the integration path between a learning agent and existing controller and device infrastructure is rarely specified. These concerns are practical rather than theoretical, and they shape the design presented here.

A reinforcement learning framework for wide-area traffic engineering is proposed and evaluated with those obstacles in view. The contribution is threefold. Traffic engineering is cast as a Markov decision process whose state is drawn from operational telemetry rather than synthetic inputs. The agent operates within a bounded action space and issues its decisions through a software-defined controller so that automated rerouting remains under operator supervision. The framework is evaluated on an emulated dual-data-center enterprise topology under injected congestion, and the measured effect on latency, loss, throughput, and link utilization is reported. The remainder of the paper reviews related work, describes the architecture and the learning model, presents the experimental setup and results, discusses challenges, and concludes with future directions.

2. Related Work

Reinforcement learning provides the formal basis for the approach. The framework of value functions and temporal-difference learning is established in the standard treatment of the field (Sutton & Barto, 2018), and the combination of deep neural networks with value learning produced agents able to act from high-dimensional states (Mnih et al., 2015). Later algorithmic advances broadened the toolkit through asynchronous actor-critic methods (Mnih et al., 2016), stable policy-gradient optimization (Schulman et al., 2017), and continuous-action control (Lillicrap et al., 2015). These methods supply the learning machinery that the present work applies rather than extends.

Applying this machinery to routing and traffic engineering has become an active area of research. Critical-flow rerouting with reinforcement learning improved traffic engineering objectives without recomputing every path (Zhang et al., 2020). Intelligent routing agents for software-defined networks have shown resilience to changing environments (Casas-Velasco et al., 2020), and their deep-learning successors have shown ability to generalize across traffic matrices (Casas-Velasco et al., 2021). Efficient routing for traffic engineering has since been demonstrated with deep reinforcement learning in software-defined settings (Pei et al., 2024), and deep reinforcement learning has been applied specifically to traffic engineering in SD-WAN (Troia et al., 2020). Comparable ideas appear in optical networks, where routing, modulation, and spectrum assignment were learned end to end (Chen, et al., 2019). A survey of the area maps these efforts and their evaluation practices (Xiao et al., 2021).

Learning has also been used to characterize rather than steer traffic. Deep-learning methods for monitoring and analysis identify patterns in flow data (Abbasi et al., 2021). The evolution of statistical traffic prediction to machine learning has been extensively documented (for example, Lohrasbinasab et al., 2022, and Chen et al., 2021). More broadly, the application of machine learning to wireless and communication networks has been interpreted as an application of a network optimization theme (Sun et al., 2019; Chen et al., 2019a; Mao et al., 2018). Congestion control has been addressed with multi-agent reinforcement learning over multi-path transport (He et al., 2021), and intent-based networking has been proposed as the operational frame within which such autonomy should sit (Leivadeas & Falkner, 2022).

Two gaps persist across this body of work. Evaluation is usually confined to small or single-domain topologies, and the question of how much authority a learned agent should hold in a production network is seldom treated as a design variable. The framework described next addresses both by bounding the action space and routing decisions through an existing controller.

3. Proposed Architecture

The framework is organized into four cooperating layers that map onto the planes of a software-defined WAN. Keeping the layers explicit clarifies where learning occurs and where operational control is retained. Table 1 summarizes the layers and their functions.

Table 1. Layers of the proposed reinforcement learning traffic-engineering framework.

Layer	Function	Representative components
Data collection	Assemble network state from live measurement	NetFlow/sFlow, SNMP, streaming telemetry
AI engine	Map state to a rerouting action under a learned policy	RL agent (value-based), reward from QoS metrics
Control plane	Translate the chosen action into enforceable policy	SD-WAN / SDN controller
Data plane	Apply routing changes at the edge	SD-WAN edge devices, headend gateways

The data collection layer gathers latency, packet loss, bandwidth utilization, and jitter from flow records and streaming telemetry. These measurements form the observation that the agent receives each decision step. The AI engine holds the learned policy and selects an action, which may shift a subset of flows to an alternative path, invoke a load-balancing mechanism, or adjust dynamic routing configuration. The control plane, realized by a software-defined controller, validates and enforces the action, and the data plane applies the change on edge devices. Figure 1 shows the closed loop formed by these layers.

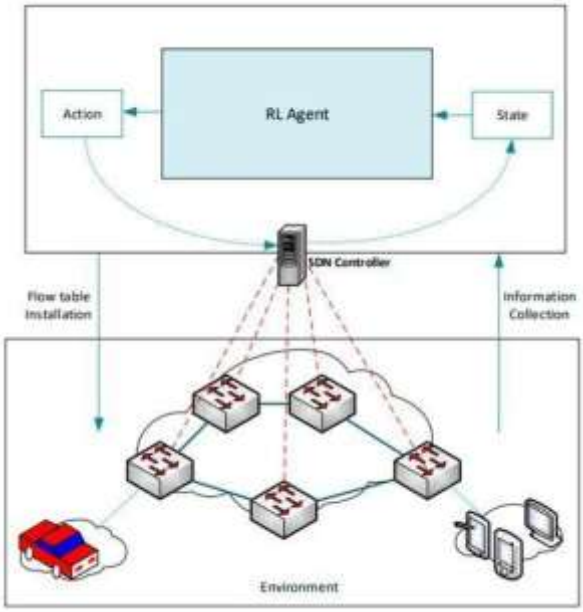

Fig. 1. Closed-loop reinforcement learning control cycle: telemetry-derived state, agent action, controller enforcement, and reward from observed quality-of-service metrics.

A design decision distinguishes this architecture from unrestricted automation. The agent is granted a bounded action space and cannot make arbitrary configuration changes; its authority is limited to a defined set of rerouting actions that a network operator has sanctioned in advance. Decisions are issued to the controller rather than pushed directly to devices, which preserves a supervision and audit point. It reflects the operational realities that enterprises adopt automation gradually and demand behavior that is predictable and can be reversed.

4. Reinforcement Learning Model

Traffic engineering can be modeled as a Markov decision process. This is done by defining the state space, the action space, the transition function, the reward, and the discount factor given in the tuple of Equation (1) (Sutton & Barto, 2018).

M = (S, A, P, R, gamma) (1)

At step t, the state is composed of the network conditions, mainly link utilization, delay and a congestion indicator from telemetry. The action selects a rerouting decision for one or more traffic flows. The reward, given in Equation (2), rewards delivered throughput and penalizes latency and packet loss, with non-negative weights that encode operator priorities.

R_t = gamma_w * Theta_t - alpha * D_t - beta * Lambda_t (2)

Here Theta_t denotes throughput, D_t delay, and Lambda_t packet loss, while alpha, beta, and gamma_w are weighting factors. The learning objective is to find a policy that maximizes the expected cumulative discounted reward, as in Equation (3).

J(pi) = E [sum_t gamma^t R_t] (3)

A value-based agent estimates the action-value function and improves it through the temporal-difference update in Equation (4), which underlies the deep Q-network family used for high-dimensional state (Mnih et al., 2015).

Q(s,a) <- Q(s,a) + eta [r + gamma max_a' Q(s',a') - Q(s,a)] (4)

Because the raw metrics differ in scale, each state feature is normalized to a common range before it enters the reward and the network input, as in Equation (5). Link utilization, a central term in both the state and the reward, is defined in Equation (6) as offered load relative to capacity.

x_hat = (x - x_min) / (x_max - x_min) (5)

U_l = load_l / capacity_l (6)

The weighting factors in Equation (2) are treated as operator-set parameters rather than learned quantities, which keeps the objective interpretable and lets an operator bias the policy toward latency-sensitive or throughput-sensitive behavior. Table 2 lists the model elements and the parameters used in the evaluation.

Table 2. Reinforcement learning model elements and evaluation parameters.

Element	Definition	Setting in this study
State S_t	Link utilization, delay, congestion indicators	Normalized per Eq. (5)
Action A_t	Rerouting decision (path shift, load balance)	Bounded, operator-sanctioned set
Reward R_t	Throughput gain minus latency and loss penalties	Eq. (2)
Weights	Operator priority factors	alpha, beta, gamma_w set by operator
Discount gamma	Weighting of future reward	0 < gamma < 1

5. Experimental Setup

The framework was evaluated on an emulated network built in EVE-NG, with the learning agent implemented in Python using standard deep-learning libraries and traffic generated by iPerf. The topology reproduced the structure of an enterprise WAN in miniature: two data centers, each with dual headend devices, and fifteen branch nodes connected through dual internet transports with multiple paths between nodes. Table 3 records the environment.

Table 3. Emulation environment and evaluation configuration.

Component	Choice
Emulation platform	EVE-NG
Agent implementation	Python with TensorFlow / PyTorch
Traffic generation	iPerf
Topology	Dual data center, dual headend per site
Branch nodes	15, each with dual internet transports
Telemetry	NetFlow and QoS counters
Baseline	Static routing / manual intervention

Congestion scenarios were produced by injecting latency, jitter, and load onto the SD-WAN tunnels while iPerf drove traffic from the fifteen nodes through a headend device, saturating the headend link. Telemetry was fed back to the agent and compared with a previously recorded healthy baseline so that deviations identified the affected interfaces. When defined trigger criteria were met, the agent was permitted to shift roughly half of the affected connections to the secondary headend, issuing the instruction to the controller, which enforced the change on the devices. The agent's authority was deliberately restricted to this sanctioned action rather than unrestricted reconfiguration. Figure 2 shows the evaluation topology.

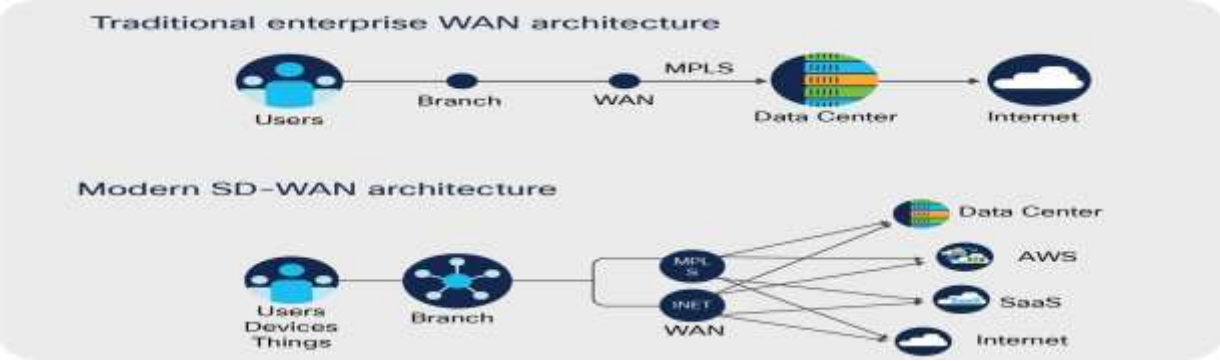

Fig. 2. Emulated dual-data-center evaluation topology with dual headends and fifteen dual-homed branch nodes.

6. Results and Analysis

The learned policy was compared with the static baseline under the injected congestion scenarios. The rerouting decision was triggered and enforced successfully, and the observed effect on the quality-of-service metrics is reported below as primary experimental data from this study. Figure 3 summarizes the improvement over the baseline, and Table 4 records the same results with the measured direction of change.

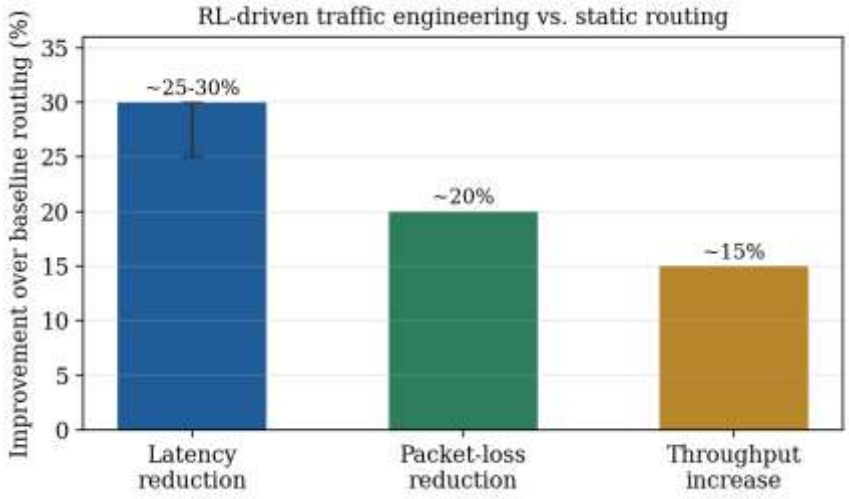

Fig. 3. Measured improvement of the reinforcement learning policy over static routing across latency, packet loss, and throughput (primary experimental data, this study).

Table 4. Measured performance of the reinforcement learning policy relative to the static baseline (primary experimental data, this study).

Metric	Change vs. baseline	Direction
Latency	~ 25 to 30% reduction	Improved
Packet loss	~ 20% reduction	Improved
Throughput	~ 15% increase	Improved
Link utilization	Balanced across parallel paths	Improved

The policy responded to congestion faster than static routing or manual intervention, redistributing load before the saturated headend link degraded application traffic further. Balanced utilization across parallel paths indicates that the agent exploited redundant capacity that fixed metrics leave idle. Positioned against related work, the results are consistent in direction with reinforcement learning traffic-engineering studies in software-defined and SD-WAN settings, while the present evaluation emphasizes a constrained, controller-mediated action path rather than unrestricted control. Table 5 places the approach alongside representative prior methods.

Table 5. Positioning relative to representative reinforcement learning routing and traffic-engineering studies.

Study	Setting	Distinguishing emphasis
Zhang et al. (2020)	SDN, simulation	Critical-flow rerouting
Casas-Velasco et al. (2021)	SDN, simulation	Deep RL routing across matrices
Troia et al. (2020)	SD-WAN, emulation	Deep RL traffic engineering
Pei et al. (2024)	SDN, simulation	Efficient DRL routing
This study	Enterprise SD-WAN, emulation	Bounded action space, controller-mediated, operator-supervised

7. Discussion and Challenges

The results support the view that reinforcement learning can improve wide area traffic engineering, but the more useful finding concerns how that improvement is obtained. Restricting the agent to a sanctioned action set and routing its decisions through the controller produced measurable gains without surrendering operational control, which addresses the trust barrier that has kept learned agents out of production. The design treats operator authority as a first-class parameter rather than an afterthought.

Several challenges qualify the findings. Training a reinforcement learning agent to a dependable policy takes time and representative data, and a policy learned in emulation may transfer imperfectly to production traffic. Scaling from fifteen emulated nodes to an estate of thousands of branches raises questions of state dimensionality and decision latency that this study does not resolve. Integration with legacy equipment that predates programmable control remains an obstacle, and the safety and convergence of learned policies under rare conditions require guarantees that current methods provide only partially. These limitations bound the claims: the evaluation is an emulation of enterprise structure rather than a production trial, the scenario class is narrow, and the reported figures are specific to the tested conditions.

8. Conclusion and Future Directions

This paper framed wide area traffic engineering as a reinforcement learning problem and proposed a framework in which a telemetry-driven agent selects rerouting actions from a bounded set and enforces them through a software-defined controller. On an emulated dual-data-center enterprise topology, the learned policy reduced latency by approximately 25 to 30 percent, lowered packet loss by around 20 percent, and raised throughput by roughly 15 percent while balancing load across parallel paths, all under retained operator supervision. The central lesson is that constrained, controller-mediated learning offers a credible adoption path for autonomous traffic engineering in enterprises that cannot accept unrestricted automation. Future work will pair the control agent with traffic prediction so that rerouting anticipates congestion rather than reacting to it; extend the single agent toward multi-agent coordination for larger estates; represent network state with graph-based models to improve generalization; and situate the agent within an intent-based operational framework so that its authority is expressed as declarative policy. Validation on a production network, with a progressively widened action space backed by safety guarantees, is the necessary next step.

References

1. Abbasi, M., Shahraki, A., & Taherkordi, A. (2021). Deep learning for network traffic monitoring and analysis (NTMA): A survey. Computer Communications, 170, 19–41. https://doi.org/10.1016/j.comcom.2021.01.021
2. Casas-Velasco, D. M., Caicedo Rendón, O. M., & Da Fonseca, N. L. S. (2020). Intelligent routing based on reinforcement learning for software-defined networking. IEEE Transactions on Network and Service Management, 18(1), 870–881. https://doi.org/10.1109/TNSM.2020.3036911
3. Casas-Velasco, D. M., Caicedo Rendón, O. M., & da Fonseca, N. L. S. (2021). DRSIR: A deep reinforcement learning approach for routing in software-defined networking. IEEE Transactions on Network and Service Management, 19(4), 4807–4820. https://doi.org/10.1109/TNSM.2021.3132491
4. Chen, M., Challita, U., Saad, W., Yin, C., & Debbah, M. (2019). Artificial neural networks-based machine learning for wireless networks: A tutorial. IEEE Communications Surveys & Tutorials, 21(4), 3039–3071. https://doi.org/10.1109/COMST.2019.2926625

5. Chen, X., Li, B., Proietti, R., Lu, H., Zhu, Z., & Yoo, S. J. B. (2019a). DeepRMSA: A deep reinforcement learning framework for routing, modulation and spectrum assignment in elastic optical networks. Journal of Lightwave Technology, 37(16), 4155–4163. https://doi.org/10.1109/JLT.2019.2923615
6. Chen, A., Law, J., & Aibin, M. (2021). A survey on traffic prediction techniques using artificial intelligence for communication networks. Telecom, 2(4), 518–535. https://doi.org/10.3390/telecom2040029
7. He, B., Wang, J., Qi, Q., Sun, H., Liao, J., Du, C., . . . Han, Z. (2021). DeepCC: Multi-agent deep reinforcement learning congestion control for multi-path TCP based on self-attention. IEEE Transactions on Network and Service Management, 18(4), 4770–4788. https://doi.org/10.1109/TNSM.2021.3093302
8. Kreutz, D., Ramos, F. M. V., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. (2014). Software-defined networking: A comprehensive survey. Proceedings of the IEEE, 103(1), 14–76. https://doi.org/10.1109/JPROC.2014.2371999
9. Leivadeas, A., & Falkner, M. (2022). A survey on intent-based networking. IEEE Communications Surveys & Tutorials, 25(1), 625–655. https://doi.org/10.1109/COMST.2022.3215919
10. Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., . . . Wierstra, D. (2015). Continuous control with deep reinforcement learning. arXiv. https://doi.org/10.48550/arXiv.1509.02971
11. Lohrasbinasab, I., Shahraki, A., Taherkordi, A., & Jurcut, A. D. (2022). From statistical-to machine learning-based network traffic prediction. Transactions on Emerging Telecommunications Technologies, 33(4), e4394. https://doi.org/10.1002/ett.4394
12. Mao, Q., Hu, F., & Hao, Q. (2018). Deep learning for intelligent wireless networks: A comprehensive survey. IEEE Communications Surveys & Tutorials, 20(4), 2595–2621. https://doi.org/10.1109/COMST.2018.2846401
13. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., . . . Hassabis, D. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529–533. https://doi.org/10.1038/nature14236
14. Mnih, V., Puigdomènech Badia, A., Mirza, M., Graves, A., Lillicrap, T., Harley, T., . . . Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In Proceedings of the 33rd International Conference on Machine Learning (pp. 1928–1937). PMLR. https://proceedings.mlr.press/v48/mniha16.html
15. Pei, X., Sun, P., Hu, Y., Li, D., Chen, B., & Tian, L. (2024). Enabling efficient routing for traffic engineering in SDN with deep reinforcement learning. Computer Networks, 241, 110220. https://doi.org/10.1016/j.comnet.2024.110220
16. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv. https://doi.org/10.48550/arXiv.1707.06347
17. Sun, Y., Peng, M., Zhou, Y., Huang, Y., & Mao, S. (2019). Application of machine learning in wireless networks: Key techniques and open issues. IEEE Communications Surveys & Tutorials, 21(4), 3072–3108. https://doi.org/10.1109/COMST.2019.2924243
18. Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction (2nd ed.). MIT Press. https://mitpress.mit.edu/9780262039246/reinforcement-learning/
19. Troia, S., Sapienza, F., Varé, L., & Maier, G. (2020). On deep reinforcement learning for traffic engineering in SD-WAN. IEEE Journal on Selected Areas in Communications, 39(7), 2198–2212. https://doi.org/10.1109/JSAC.2020.3041385
20. Xiao, Y., Liu, J., Wu, J., & Ansari, N. (2021). Leveraging deep reinforcement learning for traffic engineering: A survey. IEEE Communications Surveys & Tutorials, 23(4), 2064–2097. https://doi.org/10.1109/COMST.2021.3102580
21. Zhang, J., Ye, M., Guo, Z., Yen, C.-Y., & Chao, H. J. (2020). CFR-RL: Traffic engineering with reinforcement learning in SDN. IEEE Journal on Selected Areas in Communications, 38(10), 2249–2259. https://doi.org/10.1109/JSAC.2020.3000371