Journal of Information & Optimization Sciences
ISSN 0252-2667 (Print), ISSN 2169-0103 (Online)
Vol. 47 (2026), No. 4, pp. 1591–1601
DOI : 10.47974/JIOS-2243

A mathematical congestion-aware Dijkstra optimization model for adaptive routing in CCDN

Rohit Kumar Gupta[†]
Department of Computer Science & Engineering
Malaviya National Institute of Technology Jaipur
Jaipur 302017
Rajasthan
India

and

Department of Information Technology
Manipal University Jaipur
Jaipur 303007
Rajasthan
India

Arka Prokash Mazumdar *
Department of Computer Science & Engineering
Malaviya National Institute of Technology Jaipur
Jaipur 302017
Rajasthan
India

Abstract

Today's content delivery over the Internet via Cloud-based Content Delivery Networks (CCDNs) must address critical challenges such as dynamic traffic, congestion, inefficient routing, and latency variations, which are not effectively handled by the traditional Dijkstra's algorithm due to its static cost assignments. To overcome these challenges, a congestion-aware collaborative Dijkstra optimization approach guided by the Proximal Policy Optimization (PPO) method is proposed. This PPO-Guided Dijkstra integrates graph-theoretic path computation with optimization-driven cost adjustments by considering real-time congestion data. Simulation-based assessments across various network conditions

[†] *E-mail:* 2018rcp9091@mnit.ac.in;
rohitkumar.gupta@jaipur.manipal.edu
* *E-mail:* apmazumdar.cse@mnit.ac.in (Corresponding Author)

© Taru Publications

indicate that the proposed model achieves a 33% reduction in latency, a 6.7% improvement in resilience under node failure scenarios, and significantly faster convergence in comparison to conventional and non-collaborative routing strategies. These findings demonstrate the scalable, congestion-aware routing solution for next-generation CCDNs.

Subject Classification: *Primary 68M10, Secondary 68T05, 94C15.*

Keywords: *Cloud-based CDN, Mathematical decision models, PPO, Dijkstra, Congestion-aware routing.*

1. Introduction

Replica servers (RSs) are placed at different geographical locations to reduce latency and network traffic on the Internet by using Content Delivery Networks (CDNs). Initially, these RSs are provisioned with static infrastructures, which are inefficient in utilizing resources and not scalable in dynamic scenarios. To deal with this, CCDNs [1] are proposed, which use leased servers to provision adaptive cloud resources dynamically and provide better flexibility and scalability in resource provisioning. However, these dynamic RSs also face different challenges during time-varying traffic conditions, fluctuating user demand, and the selection of an optimal number of RSs [2], [14], [17]. During these dynamic conditions, content delivery to end users with minimal latency by dynamically selecting optimal paths between RSs and requesting nodes in CCDNs is one of the primary challenges of the Internet [3]. This demanded the routing process to operate under highly dynamic network conditions, characterized by unpredictable traffic patterns and temporally varying resource provisioning on RSs, rendering static routing algorithms, such as Dijkstra's shortest path, inadequate for performance optimization. Therefore, improving content delivery efficiency in CCDNs has gradually depended on innovations in adaptive routing by enhancing hybrid optimization frameworks [6], [20] and reinforcement learning-based approaches [8], [10], [18].

1.1 *Mathematical Formulation of the CCDN Routing Problem*

The routing problem in CCDNs can be formally represented as a weighted directed graph.

$$G = (V, E) \tag{1}$$

Here, $V = \{RS_1, \ldots, RS_s\}$ represents a set of s nodes as replica servers configured for the cloud. E is the set of edges, where each edge is defined by $e(u, v)$, with an associated cost C_{uv} based on latency, bandwidth, and congestion, where $u, v \in V$. The different users $U_s = \{u_{1s}, u_{2s}, \ldots, u_{ks}\}$ under the $RS_s \in V$ place content requests $R_{ks} = \{r_{1ks}, r_{2ks}, \ldots, r_{nks}\}$ generated by the user $u_{ks} \in U_s$. The objective is to find optimal paths to minimize end-to-end latency and congestion, ensure that load is evenly distributed, and that routing decisions adapt in real time. All of this must be achieved with minimal computational or communication overhead.

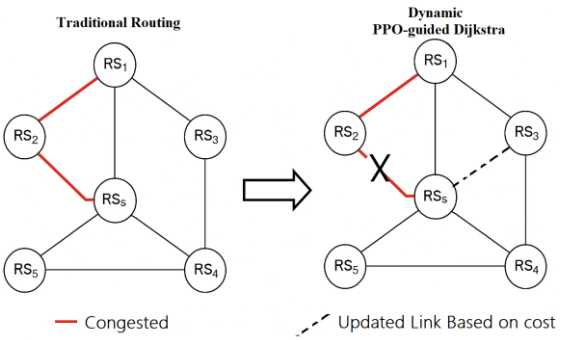

Figure 1
CCDN Network Routing Comparison

A comparison between two traditional routing protocols, namely Dijkstra's algorithm and dynamic lightweight hybrid routing, is illustrated in Figure 1. In traditional routing, congested paths between RS_1 and RSs via RS_2 are selected based on static costs without adapting to time-varying network conditions. Dynamic approaches identify congestion, discard overloaded links, and reroute traffic through alternative paths (e.g., $RS_1 \rightarrow RS_3 \rightarrow RSs$) by updating links based on dynamically learned edge costs. This highlights the hybrid model's ability to adapt in real time and improve routing efficiency.

A standard Dijkstra solution selects, for each request originating at node *s* to the destination *d*, such that the selected path (π) is minimized in terms of the total path cost**.** This results in the optimal path (π*) as shown in Eq. *(2)*.

$$\pi^* = \arg\min_{\pi: s \rightarrow d} \sum_{(u,v) \in \pi} (C_{uv}^{base}) \quad (2)$$

Each edge *(u, v)* $\in E$ has a static base cost C_{uv}^{base}. However, the cost for the minimum path does not account for time-varying congestion. Therefore, a lightweight collaborative routing method is proposed that combines the adaptive decision-making capabilities of PPO [7] with the efficiency of Dijkstra's shortest path algorithm. PPO-Guided Dijkstra adjusts edge costs in real time according to the RS's load, carefully redirecting traffic to prevent congestion with low overhead [19].

The remainder of this article is structured as follows: Section 2 reviews related work, Section 3 details the proposed Collaborative PPO-guided Dijkstra routing approach, Section 4 presents the experimental setup, followed by the results and analysis in Section 5, and Section 6 concludes with future directions.

2. Related Work

Early approaches to CDN optimization focused on traditional routing and resource management strategies. The article [2] proposed an efficient routing path planning and bandwidth guarantees method, and another article [3] is an advancement by introducing a two-stage stochastic optimization model for CDN planning, accounting for traffic uncertainty through a mixed physical-virtual

NFV-based infrastructure. However, this work lacks consideration for stochastic or uncertain traffic demands, which are ever more relevant in large dynamic CCDNs.

Integrating deep reinforcement learning (DRL) into CCDN, applied dynamic content placement, ensures adaptability but experiences high training time and convergence sensitivity [4]. A collaborative RL model (CRL-DT) for routing in CCDNs, which improved distributed management but required robustness to immediate traffic pattern changes, is proposed in [5]. While deep learning with RL is proposed in [9] for content caching and delivery, scalability and deployment complexity remain open issues. In article [11], a model was introduced using Hierarchical Probability Routing (HPR) for better resource utilization, but its performance under highly dynamic environments was not evaluated.

In the closest domains, enhanced PPO was applied to robotic navigation with the Dijkstra algorithm, but these works are limited in direct applicability to CCDN use cases [12], [13]. An actor-critic-based adaptive routing and caching in knowledge-defined networks is proposed in the article [15], yet it is challenged in generalizing across heterogeneous network topologies. Another work [16] addressed reliability in time-sensitive networks using RL and PPO, though the solution's latency overhead under extreme congestion was not thoroughly tested.

Despite existing literature approaches based on RL-based routing, like CRL-DT for content routing in CCDNs, a research gap remains in further enhancing their scalability and adaptability to achieve more efficient content delivery under dynamic network conditions.

3. Congestion-Aware Adaptive Routing: Collaborative PPO-Guided Dijkstra

This section presents Collaborative PPO-guided Dijkstra, a novel hybrid routing technique for low-overhead, adaptive content delivery in CCDNs. This approach combines PPO-inspired adaptive penalties into Dijkstra's edge costs to enable dynamic, congestion-aware routing without neural network training or continuous policy propagation between nodes.

3.1 *Heuristic Penalty System:*

The Heuristic Penalty System is defined in terms of the load factor $(\pounds_u)$ and penalty multiplier $\mathfrak{p}(\pounds_u)$, which depends on the normalized queue length or load at node u.

$$\pounds_u = load\_factor(u) \in [0.1,\ 0.9] \tag{3}$$

Here, we define the dynamic cost of edge $(u, v) \in E$ in terms of $\pounds_u$ as:

$$C_{uv}(t) = C_{uv}^{base} \times (1 + \mathfrak{p}(\pounds_u) \times \pounds_u) \tag{4}$$

Here, the penalty multiplier $\mathfrak{p}(\pounds_u)$ is determined using a simple heuristic: if the ℓu on a node exceeds the threshold $(\Theta_{\pounds_u})$, the multiplier is set to m_h; otherwise, when the multiplier is set to m_l. Thus, the updated $C_{uv}(t)$ is defined as:

$$C_{uv}(t) = \begin{cases} C_{uv}^{base} \times (1 + \mathrm{m}_h \times \pounds_u), & \pounds_u > \Theta_{\pounds_u} \\ C_{uv}^{base} \times (1 + \mathrm{m}_l \times \pounds_u), & \pounds_u \leq \Theta_{\pounds_u} \end{cases} \quad (6)$$

3.2 *PPO for Adaptive Penalty Selection:*

In place of the two penalty levels as shown in Eq. 5 as used by the heuristic penalty system, a lightweight PPO policy (π_θ) can learn the optimal multiplier (ƥ) as a function of the local state, PPO-based formulation is included in [7], [13], [15] for different scenarios. The PPO's clipped surrogate objective is defined in Eq. 7 to calculate the average at timestamp *t*.

$$L^{CLIP}(\theta) = E_t[\min(r_t(\theta)\,\hat{\mathrm{A}}_t, clip(r_t(\theta), 1 - \varepsilon, 1 + \varepsilon)\,\hat{\mathrm{A}}_t)] \quad (7)$$

where:

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{old}}(a_t \mid s_t)}\,\hat{\mathrm{A}}_t \approx \sum_{k=0}^{T-t} \gamma^k\, r_{t+k} - V_{\theta_{old}}(s_t)$$

Here, the different parameters are defined according to the adaptive routing policy. The state $s_u(t)$ is defined in terms of $\{\pounds_u, \Delta\pounds_u, C_{uv}^{base}\}_{v \in \mathcal{N}(u)}$ as local features and action $a_u(t) \in \mathbb{R}^+$ is used for the chosen penalty multiplier $ƥ_u$, where $\mathbb{R}^+$ denotes that the action space is restricted to positive real values**.** Reward *r(t)* represents the negative of end-to-end latency, and a minor penalty on excessive rerouting. The function $clip(r_t(\theta), 1 - \varepsilon, 1 + \varepsilon)$ keeps the value of $r_t(\theta)$ to the range [1−ϵ,1+ϵ], where ϵ is a hyperparameter, usually about 0.2. PPO uses this for computing the advantage function ($\hat{\mathrm{A}}_t$), a measure of how much better the actual action performed is compared to the average expectation, where $V_\theta(s_t)$ estimates the expected return from s_t under the π_θ and $\gamma \in [0,1)$ is the discount factor. Once trained, the policy produces a node-specific penalty multiplier:

$$ƥ_u = \pi_\theta(\{\pounds_u, \Delta\pounds_u, C_{uv}^{base}\}_{v \in \mathcal{N}(u)}) \quad (8)$$

Then, the updated cost is*:*

$$C_{uv}(t) = C_{uv}^{base} \times (1 + ƥ_u \times \pounds_u) \quad (9)$$

All edge costs are updated dynamically by using Eq. (9), and Dijkstra's algorithm is executed periodically after the time interval (*t*) or for each incoming request set at node *u*. From this, the algorithm adapts to real-time network congestion conditions according to load. The optimal path from source node *s* to destination *d* at *t* is:

$$\pi^* = \arg\min_{\pi: s \to d} \sum_{(u,v) \in \pi} (C_{uv}(t)) \quad (10)$$

This maintains deterministic optimal-path computation within a dynamic cost environment that represents congestion.

Algorithm 1: Collaborative PPO-guided Dijkstra

Input: G = (V, E), nodes s, t

1. For each edge e(u, v) ∈ E: *u, v* ∈ *V*

a. Set $C_{uv}^{base} \leftarrow$ G[u][v]['C_{uv}']

b. Set $£_u \leftarrow$ G.nodes[u]['load']
c. If $£_u > \Theta_{£_u}$ then: Set $ß_u \leftarrow m_h$
else: Set $ß_u \leftarrow m_l$
d. Set $C_{uv}(t) \leftarrow C_{uv}^{base} \times (1 + ß_u \times £_u)$
e. Set G[u][v]['AD_Cost'] $\leftarrow C_{uv}(t)$
2. path ← Dijkstra(G, s, t, cost = 'AD_Cost')
3. Return path

3.3 *Collaborative PPO-Guided Dijkstra Working Algorithm:*

The Collaborative PPO-Guided Dijkstra algorithm (Algorithm 1) begins by using Dijkstra's algorithm with predetermined edge costs to determine the initial shortest path between the source node (s) and the target node (t). It continuously evaluates node load parameters such as CPU utilization, queue lengths, and traffic volume to identify congestion. PPO-inspired penalty scheme (subsection 4.2) modifies edge costs according to load. Finally, Dijkstra's algorithm is subsequently executed with the modified adaptive costs (AD_Cost) to determine a new, low-congestion route.

The system autonomously redirects traffic to a longer yet less congested route when a node on the first path becomes overloaded.

4. Experimental Setup

To evaluate the performance of the PPO-Guided Dijkstra against the baseline algorithms, two synthetic network topologies were constructed. First, 100 nodes are deployed in the network within a 20×20 area as shown in Figure 2, where nodes were connected if they were within a 5-unit Euclidean distance, simulating uniform transmission range. Edge costs were initially assigned based on longitudinal distances to serve as a baseline before adaptive penalties were applied. Second, to emulate real-world CCDNs, a Gaussian-distributed topology ($\mu = 10$, $\sigma = 3$) was created, clustering nodes toward the center to represent urban deployments with high server density, as illustrated in Figure 2.

Multiple load and failure scenarios were simulated to evaluate algorithm robustness in dynamic CCDN environments. The simulation setup performs evaluation on low and high load based on the value of the current load (cl=0.3 (low) or 0.8 (high)), which indicates 30% or 80% of nodes experienced moderate (load factors 0.6–0.7) or heavy (0.8–0.9) congestions, respectively. Different baseline algorithms are included: static Dijkstra-DT, RL-DT, CRL-DT [5], and SLA-RPM [2]. These algorithms are assessed using metrics such as delay cost, convergence time, and tree length, under varying failure rates. Which offer a complete picture of each algorithm's performance on lacking adaptability, adaptive but slow and computationally intensive, faster with coordination overhead, and rigid under dynamic violations.

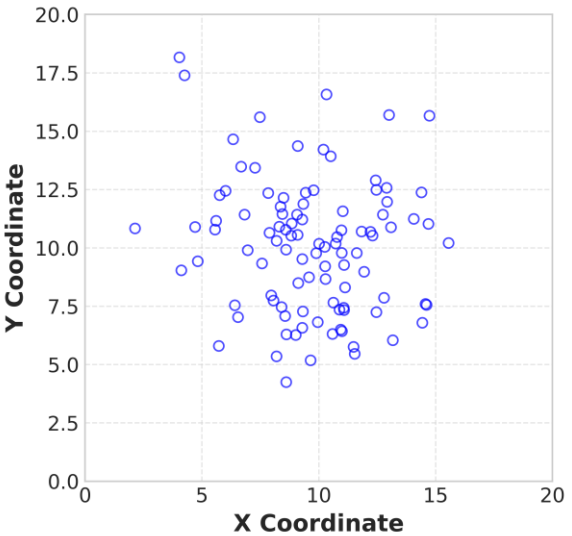

Figure 2
Node Distribution in Normal (Gaussian) Topology (μ=10, σ=3)

5. Results and Analysis

The outcomes of PPO-Guided Dijkstra's algorithm performance in network topology and both load scenarios against other algorithms demonstrate to minimize network latency, achieve quick convergence, and preserve resilience during node failures. Figure 3 and Figure 4 show that the algorithm can deftly avoid congested nodes and maintain efficient routing even as network load increases. Figure 3 shows that during cl = 0.3, the proposed algorithm drops the delay cost to 63ms and performs better than Dijkstra-DT and CRL-DT, with delay cost 80ms and 68ms, respectively, as shown in Figure 3.

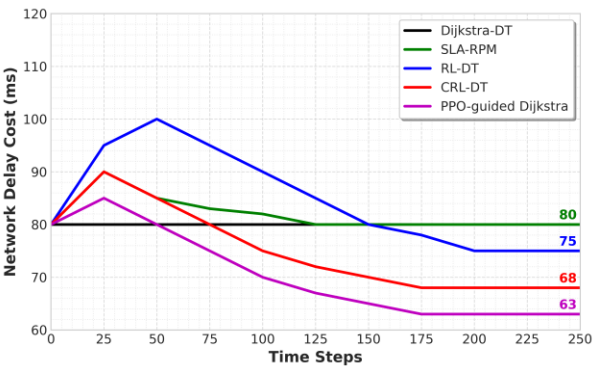

Figure 3
Network delay cost under low load

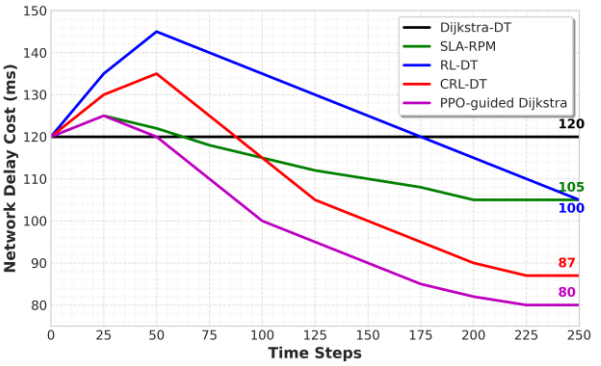

Figure 4
Network delay cost under high load

On the other hand, the proposed algorithm faces an end delay cost of 80 ms during high load cl = 0.8, which is 8% better than CRL-DT and 33% lower than Dijkstra-DT. During both scenarios, PPO-Guided Dijkstra consistently performed better than the standard algorithms in terms of the cost of network delay.

Figure 5 shows balanced path optimization while avoiding congestion during PPO-Guided Dijkstra. When the network load increased, the total tree length enlarged relatively from 25 to 39, but CRL-DT's tree length increased more consistently till 47. This shows that the proposed algorithm preserves efficiency and adaptability by avoiding congestion without causing undue path inflation. Figure 6 shows that the algorithm's resistance in situations including node failures. The delay cost is raised by 15.4% in Dijkstra-DT; however, only 6.7% in the proposed algorithm when 10, 15, or 20 nodes were randomly removed from the network. Because of its resilience, the network will continue to function appropriately even in the case of unexpected outages.

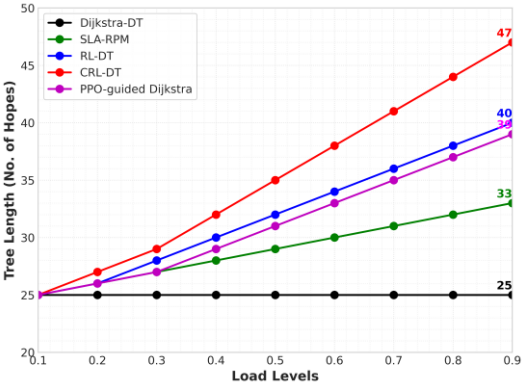

Figure 5
Tree Length Analysis Network

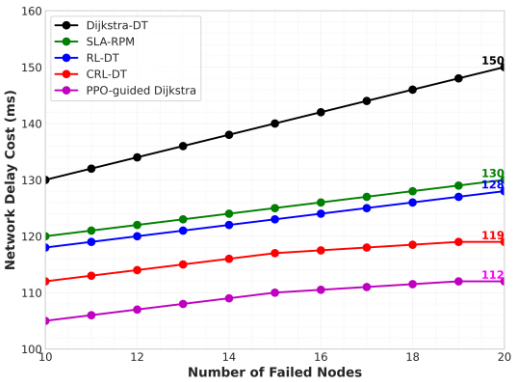

Figure 6
Delay cost in node failure

The above results of PPO-Guided Dijkstra adjusts edge costs locally based on node load, supporting fast redirecting around bottlenecks or failures without restructuring entire delivery trees. Unlike tree-based methods that may require subtree reconstruction on failure, it recalculates only affected routes using adaptive penalties.

6. Conclusion and Future Work

Conventional approaches are not appropriate to deal with high network traffic and load on RSs during content delivery, which demands delivering it with minimum latency. The standard Dijkstra path selection algorithm is constrained by static costs, while recent RL methods enforce poor convergence and real-time overhead. The proposed path-finding PPO-Guided Dijkstra approach shows exceptional performance during low and high network load conditions, achieving a 33% reduction in latency and sustaining resilience with only a 6.7% increase in time during node failures. Probable future developments of this work include concentrating on CCDN testing during real-time dynamic changes and focusing on complex difficulties like multiple failures.

References

[1] C. Papagianni, A. Leivadeas, and S. Papavassiliou, "A cloud-oriented content delivery network paradigm: Modeling and assessment," *IEEE Trans. Dependable Secure Comput.*, vol. 10, no. 5, pp. 287–300 (2013).

[2] Y.-W. Ma, J.-L. Chen, C.-C. Chang, A. Nakao, and S. Yamamoto, "A novel dynamic resource adjustment architecture for virtual tenant networks in SDN," *J. Syst. Softw.*, vol. 143, pp. 100–115 (2018).

[3] M. Mangili, J. Elias, F. Martignon, and A. Capone, "Optimal planning of virtual content delivery networks under uncertain traffic demands," *Comput. Netw.*, vol. 106, pp. 186–195 (2016).

[4] Y. Liu, D. Lu, G. Zhang, J. Tian, and W. Xu, "Q-learning based content placement method for dynamic cloud content delivery networks," *IEEE Access,* vol. 7, pp. 66384–66394 (2019).

[5] M. He, D. Lu, J. Tian, and G. Zhang, "Collaborative reinforcement learning based route planning for cloud content delivery networks," *IEEE Access*, vol. 9, pp. 30868–30880 (2021).

[6] M. Pooyandeh and I. Sohn, "Edge network optimization based on AI techniques: A survey," *Electronics,* vol. 10, no. 22, pp. 2830 (2021).

[7] W. Funika, P. Koperek, and J. Kitowski, "Automated cloud resources provisioning with the use of the proximal policy optimization," *J. Supercomput.,* vol. 79, no. 6, pp. 6674–6704 (2023).

[8] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications," *IEEE Trans. Cybern.,* vol. 50, no. 9, pp. 3826–3839 (Sep. 2020).

[9] A. Sadeghi, G. Wang, and G. B. Giannakis, "Deep reinforcement learning for adaptive caching in hierarchical content delivery networks," *IEEE Trans. Cogn. Commun. Netw.,* vol. 5, no. 4, pp. 1024–1033 (Dec. 2019).

[10] Q. Cai, C. Cui, Y. Xiong, W. Wang, Z. Xie, and M. Zhang, "A survey on deep reinforcement learning for data processing and analytics," *IEEE Trans. Knowl. Data Eng*., vol. 35, no. 5, pp. 4446–4465 (May 2023).

[11] M. Sasikumar, P. J. Jayarin, and F. S. F. Vinnarasi, "A hierarchical optimized resource utilization based content placement (HORCP) model for cloud content delivery networks (CDNs)," *J. Cloud Comput.,* vol. 12, no. 1, pp. 139 (2023).

[12] H. Taheri, S. R. Hosseini, and M. A. Nekoui, "Deep reinforcement learning with enhanced PPO for safe mobile robot navigation," arXiv preprint, arXiv:2405.16266 (2024).

[13] Li Keqin, Lipeng Liu, Jiajing Chen, Dezhi Yu, Xiaofan Zhou, Ming Li, Congyu Wang, and Zhao Li, "Research on reinforcement learning based warehouse robot navigation algorithm in complex warehouse layout," in Proc. 2024 6th Int. Conf. Artif. Intell. Comput. Appl. (ICAICA), pp. 296–301 (2024).

[14] R. K. Gupta and A. P. Mazumdar, "Cost-aware and time-varying request rate-based dynamic cache provisioning over CCDN," in Proc. IEEE Int. Conf. Adv. Netw. Telecommun. Syst. (ANTS), Jaipur, India, pp. 610–615 (2023).

[15] Y. Xiao, H. Yu, Y. Yang, Y. Wang, J. Liu, and N. Ansari, "Adaptive joint routing and caching in knowledge-defined networking: An actor-critic deep reinforcement learning approach," *IEEE Trans. Mobile Comput.*, vol. 24, no. 5, pp. 4118–4135 (May 2025).

[16] H. Cheng, L. Yang, Q. Zhang, and W. Zhu, "Reliable routing and scheduling in time-sensitive networks based on reinforcement learning," *IEEE Trans. Netw. Sci. Eng.* (2025).

[17] R. K. Gupta and A. P. Mazumdar, "Cost-balanced adaptive replica server cache management under time-varying user request density in CCDNs," in Proc. IEEE Int. Conf. Commun. Syst. Netw. Technol. (CSNT), Bhopal, India, pp. 696–702 (2025).

[18] V. R. Bolla, B. Vikas, S. Potluri, Y. Subbarayudu, G. Sucharitha, and N. Narisetty, "Scalable cloud-based reinforcement learning for multimedia data in cognitive neuroscience for secure healthcare analysis," *J. Inf. Optim. Sci.*, vol. 46, no. 6, pp. 1793–1801 (2025).

[19] N. Tuli, R. Phursule, S. Bansal, B. M. Nanche, S. Katoch, and A. Raina, "Investigating discrete structures in network algorithms for optimizing traffic flow and reducing congestion in urban areas," *J. Discrete Math. Sci. Cryptogr.*, vol. 29, no. 2-B, pp. 849–856 (2026).

[20] O. Aruna and A. Sharma, "An adaptive routing protocol in flying ad hoc networks," *J. Discrete Math. Sci. Cryptogr.*, vol. 25, no. 3, pp. 757–770 (2022).

Received December, 2025