Journal of Integrated Science, Technology and Management, 2025; 1(2):1-8

Journal of Integrated Science, Technology and Management

Research Article

Constrained Multi-Objective Deep Reinforcement Learning for Resource Allocation in Edge–Cloud IoT Networks

Mirzayev Bobur Soyibjonovich[1*], Kashaf Shafeeq[2]

[1]*Head of the Department of Finance, Alfraganus University, Tashkent, Uzbekistan*
[2]*Researcher*

ARTICLE INFO

Article history:
Received: 14-03-2025
Revised: 22-06-2025
Accepted: 08-09-2025
Published: 15-10-2025

Keywords:
Internet of Things, edge–cloud computing, deep reinforcement learning, multi-objective optimization, task offloading, energy efficiency, quality of service.

DOI: 10.66592/krt582mx

ABSTRACT

The rapid proliferation of Internet of Things (IoT) applications has created heterogeneous, bursty and deadline-sensitive workloads that cannot be managed efficiently through static edge–cloud allocation. This study proposes a constrained multi-objective deep reinforcement learning (CMO-DRL) architecture for jointly selecting task placement and computational capacity across IoT devices, edge servers and cloud infrastructure. The design combines an actor–critic scheduler, normalized latency–energy–cost objectives, deadline constraints, action masking and a Pareto-aware decision layer. A mathematical formulation, implementation workflow, evaluation protocol and reproducible illustrative scenario are presented. The numerical examples demonstrate how a candidate policy could be evaluated against first-come-first-served, edge-first and latency-only reinforcement learning baselines, but are not claimed as outcomes of an executed experiment. The framework emphasizes feasibility, interpretable trade-offs, scalable orchestration and reproducible validation under variable traffic intensity.

INTRODUCTION

Internet of Things networks connect sensing, actuation, communication and computation across environments such as smart manufacturing, transportation, healthcare and environmental monitoring. Devices continuously generate telemetry, images, events and control requests whose value depends on timely processing. Their computational requirements differ dramatically: a temperature reading may need a simple threshold check, whereas a machine-vision stream requires sustained acceleration and memory. These differences make a single scheduling rule inadequate. Resource allocation must respond to changing task urgency, network conditions and available processing capacity while respecting energy budgets and service-level objectives.

Cloud computing provides elasticity, centralized management and access to powerful accelerators. However, routing every request to a remote data center introduces network round-trip delays, consumes bandwidth and may increase dependence on unstable backhaul links. Edge computing places compute closer to data sources, potentially improving responsiveness and reducing upstream traffic. Edge servers nevertheless have finite processors, memory and energy envelopes. An edge-only strategy can therefore create queues and deadline misses when many devices simultaneously submit computationally intensive jobs. A collaborative edge–cloud architecture must determine when proximity outweighs capacity and when remote execution is preferable.

Workload dynamics make the allocation problem particularly difficult. Arrival rates vary across hours, users and physical locations; task sizes and deadlines change unpredictably; wireless links experience fading and contention; and background workloads consume resources without warning. Decisions made for one task affect queue lengths and the performance of later tasks. Greedy scheduling can reduce immediate response time but create congestion in subsequent intervals. Conversely, a policy that conserves edge capacity may send excessive traffic to the

***Corresponding Author:** Mirzayev Bobur Soyibjonovich
Address: Head of the Department of Finance, Alfraganus University, Tashkent, Uzbekistan
Email ✉ b.mirzayev@afu.uz **ORCID:** 0009-0002-6741-4606

Relevant conflicts of interest/financial disclosures: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

© 2025, Mirzayev Bobur Soyibjonovich and Kashaf Shafeeq, This is an open access journal, and articles are distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 License, which allows others to remix, tweak, and build upon the work non-commercially, as long as appropriate credit is given and the new creations are licensed under the identical terms.

cloud. Effective orchestration therefore requires decisions that account for future system state rather than optimizing each request in isolation.

Energy consumption is a second central consideration. IoT devices frequently operate under battery limitations, while edge sites may face thermal restrictions, energy tariffs or carbon-intensity constraints. Local processing saves transmission energy for some workloads but can require prolonged CPU activity on a low-power device. Offloading may lower device energy while raising communication and server-side consumption. A comprehensive evaluation must distinguish device energy, edge energy and cloud energy, and must specify whether the objective is total system energy or a narrower operational boundary. Without a consistent accounting boundary, energy comparisons may be misleading.

Latency, energy and monetary cost are often conflicting. Increasing processor frequency can shorten execution time while raising dynamic power consumption. Reserving more edge capacity improves deadline adherence but can leave servers underutilized. Cloud execution may improve throughput but incur data transfer charges and longer network paths. Consequently, the scheduling task is naturally multi-objective. A useful method should expose the trade-offs rather than assuming that one scalar performance metric fully represents user requirements. Priority classes, application deadlines and resource scarcity can also change the relative importance of objectives over time.

Traditional resource-management approaches include first-come-first-served scheduling, earliest-deadline-first, least-loaded assignment, greedy offloading, integer programming and metaheuristic search. These methods remain valuable benchmarks because they are interpretable and often computationally inexpensive. However, fixed heuristics cannot always capture long-horizon interactions among workload queues, channel conditions and server states. Exact optimization may be computationally prohibitive for large combinatorial action spaces or may require forecasts that are unavailable during real-time execution. Hybrid solutions can combine fast learned decisions with hard feasibility checks.

Reinforcement learning offers a framework for sequential decision-making under uncertainty. An agent observes resource utilization, task attributes and network measurements; selects an allocation; and receives feedback reflecting service quality and operating cost. Over repeated interactions, it can learn how present actions affect later congestion and rewards. Deep neural networks extend this approach to high-dimensional observations, making them attractive for heterogeneous edge–cloud settings. Nevertheless, naïve reinforcement learning can violate capacity constraints during exploration, overfit to a simulated workload, or learn undesirable policies from poorly scaled reward functions.

This paper therefore develops a constrained multi-objective deep reinforcement learning design that separates feasibility from preference optimization. An action mask excludes placements lacking sufficient memory, compute quota or connectivity. A learned policy proposes eligible placement and resource-share decisions, while a supervisory controller enforces deadlines and admission rules. Objective normalization avoids allowing a numerically large metric to dominate merely because of its units. A Pareto-aware selection stage can then expose candidate policies corresponding to different operational priorities.

The proposed architecture is intended for centralized or hierarchical orchestration. IoT devices submit task metadata to a nearby gateway, edge nodes publish capacity and queue measurements, and the scheduler observes summarized cloud availability and link conditions. The controller returns a placement decision and a compute allocation that is subsequently enforced by the execution platform. Monitoring closes the feedback loop by reporting actual execution time, resource use and deadline status. Such a design can be implemented with container orchestration and telemetry systems, although its effectiveness must be validated with measured workloads.

The contributions of the present work are a formal multi-objective resource allocation model; a constraint-aware actor–critic scheduling mechanism; a step-by-step flowchart and training protocol; and a transparent numerical illustration with comparative tables and charts. Unlike studies that report simulation values without an executable experiment, the present manuscript explicitly distinguishes design hypotheses from demonstrated results. The proposed method should be judged through repeated seeded experiments, strong baselines, sensitivity studies and statistical uncertainty estimates before any performance advantage is asserted. The remainder reviews the literature, develops the model, specifies evaluation methods and discusses expected limitations.

Related Work

Resource allocation in distributed computing has historically been framed as a scheduling and queue-management problem. Classic heuristics prioritize arrival order, deadline or server load, whereas mathematical programming expresses placement as a constrained assignment. Their simplicity and transparency make them useful operational defaults. However, assumptions of known task duration, stable capacity or independent requests are frequently violated in edge–cloud IoT deployments. Dynamic workload variability has therefore motivated adaptive optimization and learning-based methods that estimate the value of decisions from observed system behavior.

Mao and colleagues introduced DeepRM as an early example of applying deep reinforcement learning to resource management. Their formulation showed that scheduling tasks with multiple resource demands can be approached as a sequential learning problem rather than solely as a handcrafted heuristic [1]. This work helped

establish the relevance of state representation, action design and reward shaping to systems management. Yet data-center scheduling differs from edge–cloud IoT operation because offloading introduces wireless variability, device energy constraints and geographically distributed compute resources.

Lei and coauthors investigated multi-user resource control in IoT edge computing, modeling coupled computation and communication decisions with a Markov decision process and deep learning-based value approximation [2]. Their study highlighted how contention among users can make local optimization inadequate. It also illustrated the curse of dimensionality as the number of devices grows. These observations motivate compressed state descriptions and hierarchical controllers in which global resource summaries are combined with local feasibility decisions.

Liu, Yu and Gao examined multi-agent reinforcement learning for computation offloading in IoT edge networks [3]. Their stochastic-game perspective recognizes that multiple devices compete for shared radio and processing resources. Decentralized learning can reduce the burden on a single coordinator and may improve robustness when local agents have incomplete information. However, independent learners can face nonstationary training conditions because each agent changes its policy while others are learning. A centralized training phase with coordinated constraints may provide a more stable starting point.

Cui and colleagues studied deep reinforcement learning-based resource allocation for content distribution in IoT–edge–cloud environments [4]. Their work combined caching and routing choices, showing that the allocation problem extends beyond CPU cycles to include network and storage resources. This distinction matters because a server with spare processors can still be unsuitable when its link is congested or necessary content is unavailable. Joint orchestration should therefore represent compute, memory, bandwidth and transfer requirements rather than using CPU utilization alone.

Ullah and coauthors formulated edge–cloud task offloading as a Markov decision process and used Double Deep Q-Network learning to address time-varying resources and task constraints [5]. The reported study supports the use of adaptive policies for offloading and rejection decisions. Double Q-learning can reduce overestimation in value-based methods, although discretizing resource shares may produce a large action space. Actor–critic methods with hybrid discrete–continuous decisions provide an alternative when placement and allocated compute fractions must be chosen jointly.

Zhang and colleagues explicitly considered multi-objective edge–cloud scheduling using deep reinforcement learning and optimization [6]. Their formulation included task completion time, economic cost, energy and reliability. This is particularly relevant to the present proposal because optimizing a single metric can degrade other dimensions of service quality. Their hybrid design suggests that reinforcement learning can guide search while an optimization layer refines candidate schedules. The present manuscript builds on this general direction but emphasizes capacity masking, deadline constraints and transparent reporting of objective weights.

Xu and colleagues investigated combining deep reinforcement learning with edge computing for real-time monitoring and control in industrial IoT environments [7]. Their architecture illustrates the importance of placing lightweight inference near time-critical processes and coordinating with cloud resources. Real-time control tasks impose a more demanding evaluation standard than batch jobs: average latency alone is insufficient when rare tail delays can destabilize operations. Future studies should consequently measure percentile response times, deadline misses and behavior during sudden workload spikes.

Recent multi-objective work has explored adaptive edge–cloud workflow optimization and IoT scheduling across energy, latency and service-level objectives [8,9]. Such studies indicate growing interest in jointly balancing heterogeneous quality-of-service targets. Nevertheless, direct comparisons can be difficult when papers use different simulation tools, workload distributions, energy models, baseline implementations and reporting conventions. An apparent percentage improvement may be attributable to the test scenario rather than the policy architecture. Reproducible traces and common evaluation boundaries are therefore essential.

The literature reveals several persistent gaps. Many approaches use fixed reward weights even when task urgency changes, and some treat hard capacity restrictions as soft penalties. Other studies report average delay but omit tail latency, training overhead, inference cost or fairness across devices. A rigorous evaluation should report feasible-action rates, SLA violations, resource utilization, operating cost and variability across independent random seeds. The proposed CMO-DRL framework responds to these needs through explicit feasibility filtering, normalized objectives, an adaptable preference vector and a structured evaluation plan. Its claimed contribution is a testable integration of these elements, not an experimentally established superiority.

PROPOSED METHODOLOGY

Architecture and problem definition

Let $I=\{1,\ldots,N\}$ be incoming IoT tasks and $J=\{0,\ldots,M\}$ the available execution locations, including device-local, edge and cloud resources. Each task i has input size d_i, required CPU cycles c_i, memory m_i, arrival time a_i, deadline δ_i and priority p_i. Each node j has available processing rate f_j, memory capacity M_j and queue workload q_j. The binary variable x_{ij} indicates task placement and y_{ij} represents the assigned fraction of processing capacity. The scheduler

makes placement decisions in discrete time windows while monitoring capacity and link quality.

Latency, energy and cost models

$$T_{ij} = T^{tx}_{ij} + T^{q}_{ij} + c_i/(y_{ij} f_j) + T^{ret}_{ij} \quad (1)$$

Equation (1) decomposes completion latency into upload, queueing, execution and return-transfer components. Local execution has zero network upload and return delay; cloud execution incorporates backhaul delay. Transmission time can be estimated as d_i/R_{ij}, where R_{ij} is the measured effective link rate.

$$E_{ij} = P^{tx}_i T^{tx}_{ij} + \kappa_j c_i (y_{ij} f_j)^2 + P^{idle}_j T^{q}_{ij} \quad (2)$$

Equation (2) is an illustrative energy model that combines communication, dynamic computation and idle/queue energy. The coefficient κ_j must be calibrated to hardware measurements; server-side and device-side energy boundaries should be reported separately.

$$C_{ij} = \pi^{cpu}_j c_i + \pi^{bw}_j d_i + \pi^{stor}_j m_i \quad (3)$$

Equation (3) estimates execution, data transfer and memory reservation charges using node-specific prices. A measured billing model may replace this linear approximation.

Multi-objective constrained optimization

$$\min_{x,y} [\, \bar{L}(x,y), \bar{E}(x,y), \bar{C}(x,y), V(x,y) \,] \quad (4)$$

$$\Sigma_j x_{ij} = 1;\ x_{ij} \in \{0,1\};\ \Sigma_i x_{ij} m_i \le M_j;\ \Sigma_i x_{ij} y_{ij} \le 1 \quad (5)$$

$$V = (1/N) \Sigma_i \mathbb{1}[T_i > \delta_i];\ U_j = \text{allocated CPU capacity / available CPU capacity} \quad (6)$$

The four objectives are average latency, energy, cost and deadline violation rate. Equation (5) ensures unique placement, memory feasibility and processor-share limits for each node in a scheduling window. The violation indicator in Equation (6) equals one when completion exceeds the task deadline. For workloads where rejection is allowed, an explicit reject action and rejection penalty must be added.

Constrained multi-objective reinforcement learning

The state s_t concatenates task size, deadline slack, priority, edge/cloud queue lengths, available CPU and memory, link bandwidth, energy estimates and price signals. The action a_t selects an eligible destination and a processor-share bucket. Infeasible destinations are removed with an action mask before policy sampling. The reward combines normalized objectives with a penalty for deadline misses; weights may be varied during training to approximate a family of trade-off policies.

$$r_t = -[w_l L_t/Lref + w_e E_t/Eref + w_c C_t/Cref + w_v V_t] - \lambda g_t \quad (7)$$

$$w_l + w_e + w_c + w_v = 1;\ w_k \ge 0;\ g_t = \max(0, V_t - Vmax) \quad (8)$$

Here reference scales Lref, Eref and Cref are fixed from training data, not recomputed from test results. The nonnegative multiplier λ penalizes violations above the allowed rate Vmax. A proximal policy optimization (PPO) actor–critic agent can learn masked actions, with an independent critic estimating discounted returns.

$$A_t = \Sigma_{k=0}^{K-1} (\gamma\lambda GAE)^k [r_{t+k} + \gamma V\phi(s_{t+k+1}) - V\phi(s_{t+k})] \quad (9)$$

$$L^{PPO}(\theta) = E[\min(\rho_t(\theta)A_t, clip(\rho_t(\theta),1-\varepsilon,1+\varepsilon)A_t)] \quad (10)$$

Equation (9) defines generalized advantage estimation and Equation (10) gives the clipped PPO surrogate, where ρ_t is the probability ratio between updated and behavior policies. Training uses multiple preference vectors; nondominated validation policies are retained as an approximate Pareto set. At deployment, the selected policy must satisfy operational SLA thresholds before preferences among feasible candidates are considered.

System workflow

Figure 1 presents the proposed constraint-aware resource allocation workflow.

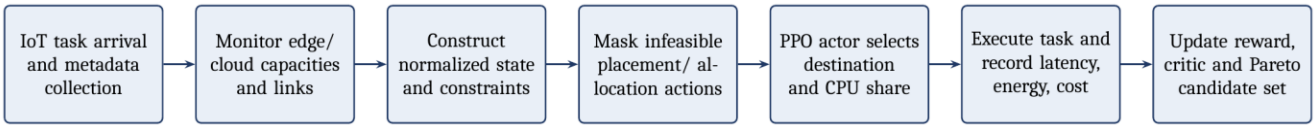


Figure 1: Proposed constraint-aware resource allocation workflow.

Experimental setup and validation protocol

A reproducible implementation may use Python, PyTorch for PPO, and an event-driven simulator such as EdgeCloudSim or a custom SimPy environment. An illustrative testbed comprises 200–1,000 devices, 10 edge servers and one cloud pool. Task arrivals follow low, medium and high-load traces; task CPU requirements and deadlines are sampled from prespecified distributions. A proper study should reserve disjoint training and test traces, use at least five independent seeds, measure wall-clock inference overhead and compare FCFS, edge-first, latency-only PPO and CMO-DRL under identical capacity constraints. Hyperparameters such as learning rate, discount factor, clipping ratio and preference weights must be tuned on validation traces only.

Numerical Illustration and Discussion

IMPORTANT: All numerical values in Tables 1–2 and Figures 2–3 are constructed examples to illustrate reporting and arithmetic. No simulation has been executed for this manuscript; these figures are not empirical findings, confidence intervals or evidence that the proposed method outperforms prior work.

Illustrative baseline comparison

Table 1: Illustrative numerical comparison at a fixed nominal workload. The proposed example values imply a 38.5% reduction in latency relative to FCFS ((182−112)/182×100) and a 26.4% reduction in energy relative to FCFS ((14.8−10.9)/14.8×100). These percentages describe the constructed table only.

Method	*Latency (ms)*	*Energy (J/task)*	*Cost (¢/100 tasks)*	*Deadline miss (%)*	*Utilization (%)*
FCFS	182	14.8	9.4	15.0	64
Edge-first	151	13.2	8.6	11.8	76
Latency-only PPO	128	12.6	9.1	8.5	81
Proposed CMO-DRL	112	10.9	7.8	6.1	85

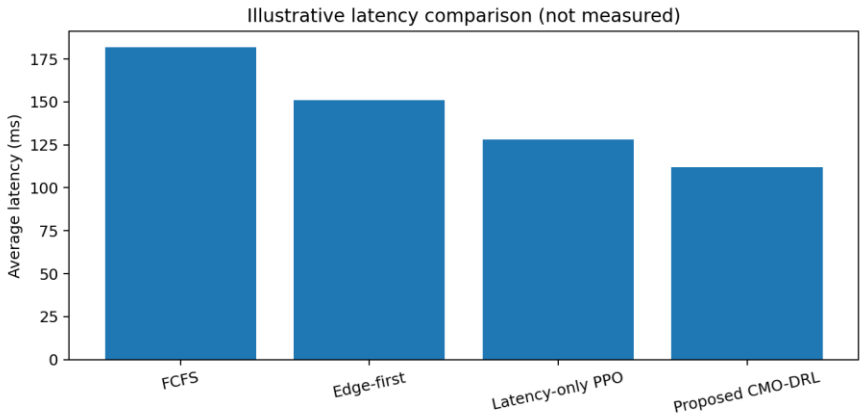


Figure 2: Illustrative average latency across four scheduling policies.

The illustrative latency chart represents the hypothesis that combining congestion awareness and multi-objective control can outperform fixed queueing rules. The latency-only PPO example reduces delay but retains higher energy and cost than the proposed multi-objective example. A real evaluation must test whether this pattern persists across independent workload traces and report dispersion rather than only point estimates.

Sensitivity to increasing task arrival intensity

Table 2: Hypothetical latency under five arrival rates; values are illustrative, not outputs from a simulator.

Arrival rate (tasks/min)	*FCFS latency (ms)*	*CMO-DRL latency (ms)*
200	90	76
400	120	92
600	162	112
800	221	151
1000	298	213

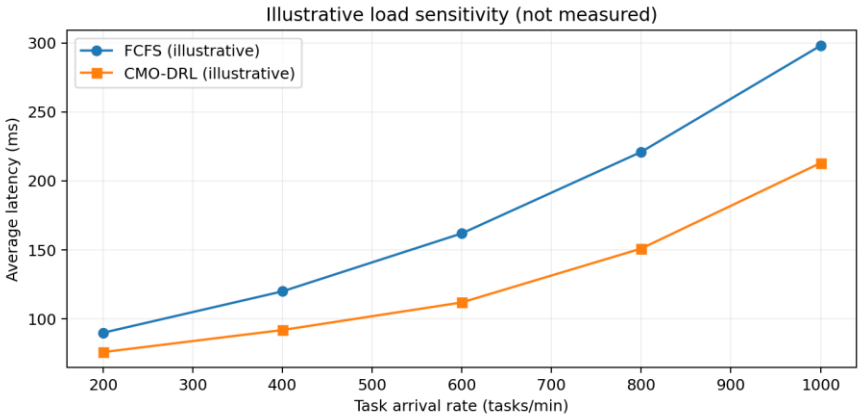


Figure 3: Hypothetical response of latency to rising task arrival intensity.

The constructed sensitivity scenario assumes that increasing arrivals raises queueing delay for every scheduler. The separation between curves illustrates a potential benefit of load-aware allocation, but the values cannot establish robustness. In an actual experiment, the evaluator should also plot 95th-percentile latency, violation rate, cloud transfer volume, energy consumption and policy decision time as a function of load. Ablation tests should remove masking, preference conditioning and SLA penalties individually to identify which components drive any measured gains.

Threats to validity

Simulation outcomes depend strongly on workload generation, network-delay assumptions, power-model calibration and baseline tuning. Unrealistic fixed bandwidth or omitted data-transfer costs can systematically favor cloud placement. Inference overhead and failed migrations can erase nominal gains. Generalization should be tested on unseen task distributions, edge failures and bursty arrivals. Fairness across devices and privacy implications of centralized telemetry also require evaluation. Statistical testing should use repeated random seeds and uncertainty intervals; conclusions must be limited to the tested environment.

Conclusion

This paper presents a constrained multi-objective deep reinforcement learning framework for resource allocation in heterogeneous edge–cloud IoT networks. The proposed method combines deadline-aware state modeling, feasible-action masking, actor–critic scheduling, normalized latency–energy–cost objectives and Pareto-aware policy selection. Equations and a flowchart specify how tasks can be assigned while respecting compute and memory limits. The illustrative tables and charts show a transparent template for comparing policies and studying load sensitivity, but do not constitute measured experimental evidence. The principal research finding at this stage is a formally specified, testable design rather than a demonstrated performance improvement. Future work should implement the policy in a reproducible simulator and real edge testbed, publish code and traces, quantify inference overhead, test against strong optimization baselines and report statistical confidence intervals. Such validation is necessary before claiming improvements in latency, energy efficiency, cost or SLA adherence.

References

1. Khan, S. (2022). The use of artificial intelligence in ESG (environmental, social, and governance) investing: A study on AI models for sustainability analytics in asset and wealth management. International Journal of Multidisciplinary and Scientific Emerging Research, 10(1), 431–439. https://doi.org/10.15662/IJMSERH.2022.1001046
2. Rao, S. Madhusudhana, and Arpit Jain. "Advances in Malware Analysis and Detection in Cloud Computing Environments: A Review." International Journal of Safety & Security Engineering 14, no. 1 (2024).
3. Anchala, S. (2024). Deployment of zero trust security models in hybrid cloud infrastructures for minimizing lateral movement and strengthening access control through continuous verification mechanisms. International Journal of Innovative Research in Computer and Communication Engineering, 12(7), 9993–10002. https://doi.org/10.15680/IJIRCCE.2024.1207114
4. Jain, A. (2024, December). Energy-Efficient Signal Processing Architectures for IoT Networks Design. In 2024 International Conference on Emerging Technologies and Innovation for Sustainability (EmergIN) (pp. 362–367). IEEE.
5. Ahuja, R. (2019). A comparative study of traditional DevOps and DevSecOps: Examining the cultural, technical, and organizational shifts required to embed security into software engineering practices. International Journal of Innovative Research in Science, Engineering and Technology, 8(8), 8561–8569. https://doi.org/10.15680/IJIRSET.2019.0808096
6. Kumar, S., Sharma, K., Kumar, P. A., Jain, A., Bhagat, S. K., & Singh, P. (2024, September). An improved particle swarm approach for energy-aware location-aided routing in mobile ad-hoc network. In 2024 7th International Conference on Contemporary Computing and Informatics (IC3I) (Vol. 7, pp. 1119–1124). IEEE.
7. Bhaskara Raju Rallabandi. (2023). AGENTIC-AI xAPPS: O-RAN MULTI-AGENT CONTROL FOR ENTERPRISE POLICY, SCHEDULING & SPECTRUM. International Journal of Data Science and IoT Management System, 2(4), 54–58. https://doi.org/10.64751/ijdim.2023.v2.n4.pp54-58

8. Khan, S. (2019). Sales force security compliance: An in-depth study of GDPR, HIPAA, and PCI-DSS enforcement in cloud-based CRM systems and their implications for global enterprises. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, 8(9), 2211–2219. https://doi.org/10.15662/IJAREEIE.2019.0809010
9. Gupta, Namit, Kunwar Singh Vaisla, Arpit Jain, Adesh Kumar, and Rajeev Kumar. "Performance Analysis of AODV Routing for Wireless Sensor Network in FPGA Hardware." Computer Systems Science & Engineering 40, no. 3 (2022).
10. Aggarwal, A. (2025). Artificial intelligence in criminal justice systems: An examination of predictive policing, automated surveillance, and ethical concerns in legal decision-making. International Journal of Multidisciplinary Research in Science, Engineering and Technology, 8(1). https://doi.org/10.15680/IJMRSET.2025.0801070
11. Jain, Jaishree, Arpit Jain, Saurabh Kumar Srivastava, Chaman Verma, Maria Simona Raboaca, and Zoltán Illés. "Improved security of e-healthcare images using hybridized robust zero-watermarking and hyper-chaotic system along with RSA." Mathematics 10, no. 7 (2022): 1071.
12. Mahajan, R. A., Dey, R., Khan, M., Mohd Su'ud, M., Alam, M. M., & Jadhav, P. (2025). Enhancing personalization in IoT-based health monitoring via generative AI and transfer learning. Egyptian Informatics Journal, 32, Article 100788. https://doi.org/10.1016/j.eij.2025.100788
13. Dwivedi, D. (2018). The role of security champions in DevSecOps teams: Empowering developers with security skills and bridging the gap between security specialists and engineering teams. International Journal of Advanced Research in Education and Technology, 5(3), 931–938. https://doi.org/10.15680/IJARETY.2018.0503016
14. Kumar, S., Jain, A., & Moparthi, N. R. (2022). Enhanced method of object tracing using extended Kalman filter via binary search algorithm.
15. Aggarwal, A. (2025). Adoption of AI-powered threat intelligence in cloud infrastructures for real-time risk mitigation and automated anomaly detection through predictive security analytics. Journal of Digital Security and Forensics, 2(1), 139–150. https://doi.org/10.29121/digisecforensics.v2.i1.2025.87
16. Bhaskara Raju Rallabandi. (2022). SYNC-ADAPTIVE RESOURCE SCHEDULING IN PRIVATE 5G FOR TIME-CRITICAL APPLICATIONS. International Journal of Data Science and IoT Management System, 1(2), 40–45. https://doi.org/10.64751/ijdim.2022.v1.n2.pp40-45
17. Anchala, S. (2020). Adoption of secure access service edge (SASE) in distributed enterprises for ensuring cloud application protection and network optimization through unified security frameworks. Turkish Online Journal of Qualitative Inquiry, 11(4), 2964–2973. https://doi.org/10.69980/60m26411
18. Harshitha, G., Kumar, S., Rani, S., & Jain, A. (2021, November). Cotton disease detection based on deep learning techniques. In 4th Smart Cities Symposium (SCS 2021) (Vol. 2021, pp. 496–501). IET.
19. Devulapalli, N. (2020). The evolution of database security from centralized architectures to distributed cloud-native systems: A comparative and historical perspective. Turkish Online Journal of Qualitative Inquiry, 11(4), 2954–2963. https://doi.org/10.69980/j1064z97
20. Mahadasu, S. S., & Rallabandi, B. R. (2023). Federated learning for privacy-preserving RAN optimization across multi-site enterprise 5G networks. International Journal of Engineering Science and Advanced Technology, 23(4), 53–57. https://doi.org/10.64771/ijesat.2023.v23.i04.pp3847
21. Ahuja, R. (2020). Adoption of secure API development practices in microservice architectures for preventing unauthorized access and data leakage through OAuth2 and input throttling. Journal of Critical Reviews, 7(9), 4063–4069. https://doi.org/10.69980/jcr.v7:i9.13404
22. Jain, Arpit, Adesh Kumar, and Sanjeev Sharma. "Comparative design and analysis of mesh, torus and ring NoC." Procedia Computer Science 48 (2015): 330–337.
23. Aggarwal, A. (2021). Utilization of continuous vulnerability scanning in DevSecOps pipelines for preemptive threat detection and secure code deployment through automated security checks. NeuroQuantology, 19(1), 427–442. https://doi.org/10.48047/nq.2021.19.1.NQ21055
24. Jain, Arpit, Nageswara Rao Moparthi, A. Swathi, Yogesh Kumar Sharma, Nitin Mittal, Ahmed Alhussen, Zamil S. Alzamil, and MohdAnul Haq. "Deep Learning-Based Mask Identification System Using ResNet Transfer Learning Architecture." Computer Systems Science & Engineering 48, no. 2 (2024): 341.
25. Bhaskara Raju Rallabandi. (2024). DESIGNING NTN–TERRESTRIAL 6G EXPERIMENTS FOR AGRICULTURE AND SMART GRIDS. American Journal of Management and IOT Medical Computing, 3(3), 27–32. https://doi.org/10.64751/ajmimc.2024.v3.n3.pp27-32
26. Mahajan, R. A., Dey, R., Khan, M., Kanabar, K., & Gupta, S. K. (2025). Real-time vehicular-to-vehicular (V2V) communication and applications for pedestrian safety in AI-optimized VANETs for autonomous vehicles. In H. Maryam, M. M. Malik, I. U. Khan, & S. K. Gupta (Eds.), AI-driven transportation systems: Real-time applications and related technologies (pp. 151–167). Springer. https://doi.org/10.1007/978-3-031-98349-8_8
27. Bhaskara Raju Rallabandi. (2023). CLOSED-LOOP DIGITAL-TWIN MANAGEMENT FOR HYBRID 5G-NTN NETWORKS. American Journal of Management and IOT Medical Computing, 2(3), 30–35. https://doi.org/10.64751/ajmimc.2023.v2.n3.pp30-35
28. Mahadasu, S. S. (2024). Closed-loop sustainability orchestration for energy-aware RAN, edge, and data-center workloads in 5G-Advanced networks. International Journal of Engineering Research and Science & Technology, 20(2), 1392–1398. https://doi.org/10.62643/ijerst.2024.v20.n2.4407
29. Khan, S. (2018). The role of zero-trust models in database security: Eliminating implicit trust and enforcing continuous verification in enterprise data access systems. International Journal of Research in Electronics and Computer Engineering, 6(1), 1622–1628.
30. Devulapalli, N. (2023). Database security and big data analytics: Protecting large-scale data warehouses, ensuring query privacy, and mitigating emerging threats in real-time analytics systems. International Journal on Recent and Innovation Trends in Computing and Communication, 11(7), 622–629. https://www.ijritcc.org/index.php/ijritcc/article/view/11939
31. Mahadasu, S. S. (2024). AI-native wireless digital twins for precision agriculture: Edge intelligence and predictive farm operations in 6G ecosystems. International Journal of Data Science and IoT Management System, 3(1), 65–72. https://doi.org/10.64751/ijdim.2024.v3.n1.1287
32. Ahuja, R. (2023). The role of artificial intelligence and machine learning in DevSecOps: Leveraging predictive analytics, automated threat detection, and anomaly identification for secure software delivery pipelines. International Journal on Recent and Innovation Trends in Computing and Communication, 11(2), 381–389. https://www.ijritcc.org/index.php/ijritcc/article/view/11946
33. Aggarwal, A. (2024). Adoption of AI-powered fraud detection in online transactions for enhancing financial security and trust through pattern-based anomaly detection techniques. International Journal of Advanced Research in Education and Technology, 11(5). https://doi.org/10.15680/IJARETY.2024.1105043
34. Mahadasu, S. S. (2022). Cross-domain service assurance for private 5G networks using streaming telemetry and dependency-aware event correlation. International Journal of Data Science and IoT Management System, 1(3), 51–56. https://doi.org/10.64751/ijdim.2022.v1.n3.1290
35. Kumar, Ashok, and Arpit Jain. "Image smog restoration using oblique gradient profile prior and energy minimization." Frontiers of Computer Science 15, no. 6 (2021): 156706.
36. Bhaskara Raju Rallabandi. (2024). 6G Cross-Layer Intent Engine: Automated Translation from Business Requirements to Network Policies and Real-Time Scheduling Controls. International Journal of Data Science and IoT Management System, 3(1), 44–48. https://doi.org/10.64751/ijdim.2024.v3.n1.pp44-48
37. Anchala, S. (2019). The evolution of DevSecOps in software supply chain security: Addressing dependencies, open-source vulnerabilities,

and the prevention of software supply chain attacks. International Journal of Multidisciplinary and Scientific Emerging Research, 7(1), 415–422. https://doi.org/10.15662/IJMSERH.2019.0701047

38. Ahuja, R. (2020). Database security in smart cities: Protecting interconnected infrastructure data, ensuring citizen privacy, and managing large-scale IoT data streams. International Journal of Multidisciplinary and Scientific Emerging Research, 8(1), 410–418. https://doi.org/10.15662/IJMSERH.2020.0801047

39. Anchala, S. (2020). Cloud security challenges in supply chain systems: Addressing vendor risks, data integrity, and third-party service provider vulnerabilities. Journal of Critical Reviews, 7(2). https://doi.org/10.69980/jcr.v7:i2.13401

40. Devulapalli, N. (2018). Database security for financial systems: Protecting customer data, ensuring PCI-DSS compliance, and safeguarding against fraud and insider attacks. International Journal of Advanced Research in Education and Technology, 5(5), 838–844. https://doi.org/10.15680/IJARETY.2018.0505018

HOW TO CITE THIS ARTICLE: Mirzayev, B. S., & Shafeeq, K. (2025). Constrained Multi-Objective Deep Reinforcement Learning for Resource Allocation in Edge–Cloud IoT Networks. *Journal of Integrated Science, Technology and Management*, 1(2), 1-8.