



Journal of Interdisciplinary Science & Technology

AI-Driven Energy-Efficient 6G Smart City Networks: Traffic Prediction, Deep Reinforcement Learning, and Zero-Touch Control

Rahul Rajawat^{1*}, Anmol Budhewar², Sai Balsaheb Gagare³, Abhijeet Shantosh Salve⁴, Rohit Dillip Cholke⁵, Tanuj Kisan Patil⁶

^{1,2} Assistant Professor, Department of Computer Engineering, Sandip Institute of Technology & Research Centre Nashik, India.

³⁻⁶ Department of Computer Engineering, Sandip Institute of Technology & Research Centre Nashik, India

rahul.rajawat@sitrc.org, anmolsbudhewar@gmail.com, saigagare9@gmail.com, santoshsalve6803@gmail.com,
rohitcholke94@gmail.com, patiltanuj581@gmail.com

Corresponding author- Tanuj Patil

DOI: 10.5281/zenodo.23344464

Abstract: Sixth-generation (6G) communication is expected to support dense Internet of Things deployments, connected vehicles, immersive services, intelligent transport, remote health-care, and other data-intensive smart-city functions. The same ultra-dense radio access infrastructure, massive multiple-input multiple-output arrays, high-frequency radios, and edge-computing capacity that enable these services can substantially increase network energy demand. Static sleep schedules and reactive load balancing cannot follow rapid spatial-temporal variations in urban traffic without either wasting power or degrading quality of service (QoS). This paper reviews energy-aware 6G research and presents a simulation-oriented architecture that combines a hybrid convolutional neural network-long short-term memory (CNN-LSTM) traffic forecaster with a proximal policy optimization (PPO) deep reinforcement learning agent. The forecaster estimates localized short-term load, while the agent selects base-station sleep states, radio-resource allocations, beam and bandwidth policies, and edge-offloading actions. A software-defined networking control plane translates these decisions into zero-touch commands, and a closed feedback loop returns power and QoS measurements as rewards or penalties. The proposed evaluation uses NS-3, Python, Gymnasium, and Stable-Baselines3 to compare the learning policy with static and reactive baselines. Metrics include total energy, energy per bit, latency, throughput, block error rate, reliability, switching frequency, and policy stability. No uncollected performance result is claimed. The work provides a reproducible research plan for reducing the operational and carbon cost of 6G infrastructure while protecting critical smart-city services.

Keywords: Internet of Things, edge computing, fog computing, multi-access edge computing, edge-cloud continuum, service orchestration, task placement, machine learning, 6G.

I. INTRODUCTION

The Internet of Things (IoT) connects sensing, communication, and computing resources across applications such as smart environments, healthcare, manufacturing, transportation, and autonomous systems. A large fraction of IoT workloads require timely processing of continuously generated data rather than only long-term storage. Cloud computing remains attractive because centralized data centres provide elastic compute, storage, and management capabilities. However, a cloud-centric design can introduce communication overhead and delay when data must travel from a resource-constrained device to a remote processing site. The edge-computing literature therefore motivates placing selected processing functions closer to data sources, especially for latency-sensitive applications [1], [4]. Fog computing extends this idea by providing an intermediate layer between end devices and centralized cloud resources, while MEC

places computing capabilities within or close to mobile access networks. These paradigms should not be viewed as mutually exclusive alternatives. The cloud continuum concept treats heterogeneous compute locations as a connected environment in which workload placement can vary according to service requirements and available resources [2], [3], [6]. Recent surveys further show that the IoT-edge-cloud continuum introduces coupled challenges involving task offloading, resource allocation, security, scalability, and interoperability [11], [13]. Within such an environment, service orchestration becomes a decision problem rather than merely a deployment problem. A service may need low latency and fast local response at one moment, but more processing capacity or long-term analytics at another. Orchestration mechanisms consequently use combinations of application requirements, network conditions, resource availability, workload characteristics, and optimization objectives to select a placement or execution strategy [5], [9], [10]. Existing work

includes rule-based policies, optimization and mathematical programming, heuristic placement, and learning-based task offloading. Each approach exposes a different trade-off between interpretability, adaptability, computational overhead, and data requirements [8], [14], [15]. The emergence of 6G strengthens this requirement for adaptive orchestration. Recent 6G surveys describe an architecture in which wireless connectivity, artificial intelligence, and distributed intelligence are increasingly intertwined, while heterogeneous edge and cloud resources support demanding IoT services [12], [16]. 6G should therefore be considered not simply as a faster communication layer, but as an environment in which orchestration decisions may need to incorporate richer context, dynamic resource states, and stronger resilience requirements.

The objective of this paper is to provide a structured and task-centric review of service orchestration in IoT edge–fog–cloud systems and to derive a practical reference framework for intelligent placement. In contrast to reviews that primarily classify computing architectures or offloading algorithms, this paper emphasizes the connection between service requirements, decision inputs, orchestration mechanism, and evaluation metrics. The framework described later is intended as a reference design; it is not presented as an experimentally validated result in this review manuscript.

The remainder of the paper is organized as follows. Section II describes the review methodology and research questions. Section III summarizes the evolution of cloud, edge, fog, MEC, and continuum models. Section IV synthesizes orchestration approaches and decision factors. Section V compares representative literature. Section VI presents a task-centric reference framework. Section VII discusses evaluation, open challenges, and 6G considerations. Section VIII concludes the paper.

II. REVIEW METHODOLOGY

This paper follows a structured narrative-review approach. The purpose is to synthesize representative peer-reviewed work rather than claim an exhaustive systematic review with statistical screening. Literature was identified through major scholarly publisher and indexing ecosystems, including IEEE Xplore, ACM Digital Library, Springer Nature, and Elsevier/ScienceDirect, with additional bibliographic verification where necessary. The review emphasizes work published from 2015 through 2026 so that foundational concepts and recent orchestration trends are considered together.

A. Research Questions

Five questions guide the synthesis:

- 1) RQ1: How have cloud, edge, fog, MEC, and edge–cloud continuum architectures evolved for IoT workloads?
- 2) RQ2: Which service orchestration and task-placement approaches are used across these architectures?
- 3) RQ3: Which runtime factors are used to make placement decisions, and how are they combined?

- 4) RQ4: Which metrics, baselines, tools, and evaluation settings are commonly used to assess orchestration methods?
- 5) RQ5: How do 6G requirements change the design space for IoT service orchestration?

B. Scope and Selection Criteria

The review corpus prioritizes work that addresses at least one of the following: IoT computing at the edge or fog; MEC; cloud-to-things or edge–cloud continuum architecture; application placement or service orchestration; task offloading and resource allocation; learning-based decision support; or 6G-oriented distributed intelligence. Conceptual papers, surveys, and technical studies are included when they contribute directly to one of the research questions. Sources unrelated to distributed IoT computing, non-technical material, duplicates, and papers without a meaningful connection to orchestration or placement are outside the intended scope.

RQ	Focus	Evidence synthesized
RQ1	Architectural Evolution	Cloud, edge, fog, MEC, continuum
RQ2	Orchestration methods	Rules, optimization, heuristics, ML/DL, hybrid methods
RQ3	Decision Context	Latency, load, task type, energy, cost, reliability, privacy
RQ4	Evaluation	Latency, load, task type, energy, cost, reliability, privacy
RQ5	6G impact	AI-native networking, heterogeneity, resilience, scale, mobility

Table 1: Research Questions and Review Focus

C. Synthesis Procedure

The selected literature is synthesized along four dimensions:

- (i) where computation can execute, (ii) how placement is decided, (iii) what information is used at decision time, and (iv) how performance is evaluated. This decision-oriented organization is used to avoid a paper-by-paper summary and instead expose common design patterns and trade-offs.

III. ARCHITECTURAL EVOLUTION OF IOT COMPUTING

A. Cloud-Centric IoT

Cloud computing centralizes significant processing and storage in data centres. Its key strengths are scalable resources, mature service models, and simplified centralized management. For IoT, however, continuous device-to-cloud data transfer can increase bandwidth demand and expose delay-sensitive services to network conditions. The growth of edge computing was motivated in part by these limitations [1], [4].

B. Edge Computing

Edge computing moves computation closer to data sources, using gateways, local servers, access points, or nearby infrastructure. The principal benefit is reduced communication distance for latency-sensitive tasks. The trade-off is that edge resources are typically more heterogeneous and constrained

than centralized cloud resources, making resource management and placement more difficult [6], [11].

C. Fog Computing

Fog computing introduces an intermediate, distributed computing layer between endpoints and the cloud. Fog nodes can aggregate data, provide intermediate analytics and storage, and support location-aware or delay-sensitive applications [2], [7]. Fog is particularly relevant when local edge devices lack sufficient capacity but sending all workloads to the cloud is undesirable.

D. Multi-Access Edge Computing

MEC extends edge computing capabilities toward access networks and mobile environments. It is relevant to

applications with mobility and low-latency requirements and has become an important building block for IoT task offloading research [6], [13].

E. Edge-Fog-Cloud Continuum

The continuum perspective connects local edge resources, intermediate fog or MEC resources, and centralized cloud resources into one logical computing environment. Moreschini et al. emphasize the importance of a common definition of the cloud continuum, while Ullah et al. provide a taxonomy of orchestration in the broader cloud-to-things continuum [3], [5]. The practical consequence is that orchestration can choose among heterogeneous execution targets instead of assuming a single fixed destination.

Paradigm	Typical location	Primary strength	Typical limitation	Role in orchestration
Cloud	Central data centre	Large compute/storage capacity	Higher communication distance; possible backhaul load	Destination for compute-intensive or delay-tolerant workloads
Edge	Near devices/gateways	Low latency and local response	Limited and heterogeneous resources	Preferred for urgent or latency-critical tasks
Fog	Intermediate regional layer	Aggregation and distributed processing	Additional management complexity	Useful when edge capacity is insufficient but cloud-only processing is undesirable
MEC	Access-network edge	Mobility support and proximity	Resource and placement constraints across distributed nodes	Supports low-latency mobile workloads and dynamic offloading
Continuum	Integrated edge-fog/MEC-cloud	Flexible placement across tiers	Requires cross-layer orchestration	Provides the decision space for task placement

Table 2: Comparison of IoT Computing Paradigms

IV. SERVICE ORCHESTRATION: A TASK-CENTRIC SYNTHESIS

Service orchestration can be interpreted as the runtime process of selecting resources, deploying or routing services, and maintaining them as network and workload conditions change. In a continuum, orchestration has to operate across heterogeneous execution tiers rather than within a single data centre. Ullah et al. distinguish infrastructure-level and application-level orchestration, while Chiang et al. survey management and orchestration mechanisms for edge computing in IoT [5], [6].

A. Rule-Based Approaches

Rule-based policies use predefined thresholds or priority classes. They are simple, interpretable, and fast, making them suitable for safety-critical events or deterministic service requirements. Their weakness is that fixed thresholds may perform poorly when the environment changes. A rule such as “urgent event to edge” remains useful, but a rule that always sends every non-urgent task to one tier cannot fully exploit dynamic resource conditions.

B. Optimization and Heuristic Approaches

Optimization-based methods formulate placement as an objective involving delay, cost, resource utilization, energy, migration, or quality of service. Such methods can provide principled decisions but may become computationally expensive as the number of nodes, tasks, or constraints grows. For example, research on dynamic service provisioning in the

edge-cloud continuum considers service delay, resource constraints, migration, bandwidth, and computation costs jointly [10]. Application-placement surveys also document scalability challenges when large search spaces are explored exhaustively [8].

C. Machine-Learning and Deep-Learning Approaches

Learning-based approaches use historical or simulated observations to map system states to placement or offloading actions. Recent reviews show substantial interest in ML/DL for task offloading and resource allocation, especially where system states are dynamic or optimization models are difficult to solve repeatedly [14], [15]. The main advantages are adaptability and the ability to capture nonlinear relationships. The trade-offs include training-data requirements, distribution shift, explainability, and the need for carefully defined labels or reward functions.

D. Hybrid Approaches

A hybrid strategy combines deterministic handling of a small set of urgent or constrained cases with a learned or optimized decision process for the remaining workload. This pattern is attractive for IoT orchestration because safety-critical tasks often require predictable behavior, while routine workloads can benefit from adaptation. The framework proposed in Section VI follows this design principle without claiming that a specific hybrid policy is universally optimal.

E. Decision Factors

Across the reviewed literature, placement decisions commonly depend on several recurring factors:

- Service urgency and latency target: whether a task has a strict response deadline.
- Task characteristics: computational intensity, data volume, execution duration, and task type.
- Network state: communication latency, bandwidth availability, congestion, or link reliability.
- Resource state: CPU/memory utilization, queue length, accelerator availability, and residual capacity.
- Energy, cost, and mobility: relevant where devices are battery constrained or workloads/users move.
- Security and privacy: relevant when sensitive data cannot be freely moved across execution tiers.

A useful abstraction is to represent placement of task i among candidate tiers by a binary decision variable $x_{i,k}$, where $x_{i,k} = 1$ when task i is assigned to tier k . A generic multi-objective formulation can be written as

$$\min J_i = \alpha L_i + \beta C_i + \gamma U_i + \delta E_i + \epsilon R_i, \quad (1)$$

where L_i denotes latency, C_i a computation/communication cost term, U_i a resource-load term, E_i energy, and R_i a

reliability or risk penalty. The coefficients are application-dependent. The equation is used here as a conceptual synthesis rather than as a claimed optimization model of a particular prior study.

V. REPRESENTATIVE LITERATURE AND RESEARCH GAP

Table 3 summarizes representative work that directly informs the task-centric view adopted in this paper. Foundational research established the motivation for moving processing toward the network edge; later surveys formalized continuum architectures and orchestration taxonomies; more recent work increasingly focuses on dynamic offloading, ML/DL decision support, and 6G-aware distributed intelligence [1], [3], [5], [14]–[16].

Work	Scope	Approach	Main contribution	Relevance / limitation for the present review
Moreschini et al. (2022) [3]	Cloud continuum	Systematic mapping	Clarifies the cloud-continuum concept	Strong architectural foundation; does not center the review on runtime task decisions
Kong et al. (2023) [4]	Edge-driven IoT	Broad survey	Reviews architectures and enabling technologies	Broad IoT/edge coverage; orchestration decision factors are not the primary organizing lens
Ullah et al. (2023) [5]	Cloud-to-things orchestration	Taxonomy and survey	Classifies orchestration by levels and functions	Important orchestration taxonomy; present review emphasizes task-level placement context
Chiang et al. (2023) [6]	Edge/MEC management	Comprehensive survey	Reviews scheduling, resource management, and orchestration	Strong edge-management coverage; 6G-oriented task workflow is less central
Gkonis et al. (2023) [11]	IoT-edge-cloud continuum	Survey	Covers status, challenges, use cases, and technologies	Broad continuum view; present review focuses more tightly on orchestration decisions
Wang et al. (2024) [13]	IoT task offloading in MEC	Review	Classifies mechanisms, parameters, evaluation, and open issues	Detailed offloading analysis; broader service orchestration across tiers remains complementary
Dong et al. (2024) [14]	MEC offloading	Survey	Organizes offloading objectives and strategies	Useful for offloading methods; present review integrates placement with urgency and system context
Chhabra et al. (2025) [15]	DL-based offloading	State-of-the-art review	Surveys ML/DL for task offloading and resource allocation	Strong learning-centric focus; present review keeps rule, optimization, and hybrid methods in the same framework
Shahid et al. (2026) [17]	IoT service orchestration with 6G	Review	Compares distributed computing paradigms and orchestration in 6G	Closely related recent survey; present review differentiates itself through a task-centric decision lens and reference workflow linking inputs, mechanism, and metrics
This review	IoT service orchestration	Structured task-centric synthesis	Connects service requirements, decision inputs, orchestration mechanisms, and evaluation practice	Derives a practical reference framework for hybrid priority + learning-assisted placement

Table 3: Representative Literature and Positioning of This Review

A. Research Gap Identified from the Synthesis

The literature does not support a single universally applicable orchestration strategy. Instead, several recurring gaps emerge. First, architecture surveys often describe the continuum without mapping specific service requirements to runtime placement decisions. Second, offloading surveys frequently focus on mathematical objectives or algorithm classes, while operational safeguards for urgent IoT events are treated

separately. Third, learning-based methods can improve adaptability but depend on representative training data and runtime observability. Fourth, evaluation practices vary substantially across studies, making direct comparison difficult when datasets, simulators, baselines, and metrics differ [13]–[15].

A recent 2026 review has further reinforced the importance of comparing service orchestration across local-edge, fog, MEC, and cloud layers in a 6G context [17]. This makes it particularly important for the present manuscript not to duplicate a general architectural survey. The distinctive focus adopted here is therefore the task-centric decision chain: service requirement → observable context → orchestration mechanism → execution tier → measurable outcome. This lens also provides a direct bridge from literature synthesis to prototype design.

VI. REFERENCE FRAMEWORK FOR INTELLIGENT IOT ORCHESTRATION

A. Design Objectives

The reference framework is designed around five objectives: (i) minimize avoidable latency for urgent services, (ii) use available edge and cloud resources efficiently, (iii) make runtime decisions from observable context, (iv) retain an interpretable safety path for critical events, and (v) produce sufficient logs for later analysis and model improvement.

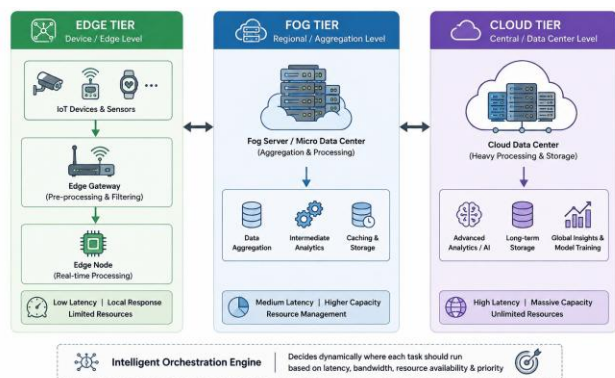


Figure 1: Reference three-tier edge–fog–cloud continuum used by the reviewed framework.

B. System Layers

The reference system contains six logical layers. The IoT layer generates sensor readings or application events. An MQTT-based communication layer provides lightweight message transport for constrained devices. An edge server performs ingestion and basic preprocessing. The orchestration layer contains a priority engine and a machine-learning decision model. The execution layer represents edge, fog/MEC, and

cloud targets. Finally, a monitoring and storage layer records decisions and performance measurements. These layers are consistent with common architectural patterns reported in the literature and with the prototype direction in the original project draft [5], [6], [11].

C. Hybrid Decision Workflow

The orchestration workflow is organized as follows:

- 1) A sensor or IoT application publishes a data item or event.
- 2) The edge ingestion service validates the message and forms a task descriptor containing task type, timestamp, and relevant telemetry.
- 3) The priority engine checks deterministic rules for urgent or policy-constrained events.
- 4) If a high-priority condition is triggered, the task follows the low-overhead edge path.
- 5) Otherwise, the ML decision layer uses current context such as network latency, edge load, and task type to estimate a suitable execution tier.
- 6) The orchestration manager routes the task and records the decision, outcome, and system state.
- 7) Logged observations can later be used for model evaluation and retraining, subject to data-quality and drift controls.

This architecture intentionally separates the safety path from the adaptive path. The goal is not to assume that machine learning should control every task, but to use learning where changing conditions make fixed rules less effective.

D. Illustrative Decision Logic

An implementation can use a simple policy such as

$$D_i = \text{EDGE if } P_i \geq \tau_P, \text{ otherwise } D_i = \text{ML}(z_i) \quad (2)$$

where P_i is the priority score of task i , τ_P is an application-defined threshold, and z_i is the runtime feature vector. The ML output can be a predicted execution tier or a probability distribution over candidate tiers. The exact model should be selected and reported according to the actual prototype implementation; the review does not prescribe a particular classifier.

VII. EVALUATION FRAMEWORK AND PRACTICAL VALIDATION

A recurring weakness in orchestration literature is that performance cannot be interpreted without understanding the baseline, workload, and metrics. The reference framework therefore recommends an evaluation protocol that can later be instantiated in a prototype without changing the conceptual research question.

A. Evaluation Scenarios

Five scenarios provide complementary stress conditions:

- Normal workload: steady sensor traffic with no exceptional events.
- Priority-event burst: sudden safety-related events that should trigger immediate edge handling.
- Network degradation: increased communication delay or reduced available bandwidth.

- Resource contention: multiple concurrent tasks competing for edge capacity.
- Mixed workload: simultaneous normal, urgent, network-stressed, and resource-constrained tasks.

B. Metrics

The core metrics should be defined before experimentation:

- 1) End-to-end latency: time from task generation to completion.
- 2) Decision quality: classification accuracy, F1-score, or another suitable measure for ML-based placement, where labeled ground truth is available.
- 3) Resource utilization: CPU and memory utilization at edge and cloud tiers.
- 4) Priority responsiveness: latency from arrival of an urgent event to initiation/completion of local handling.
- 5) Operational overhead: decision time, message overhead, and, where measurable, energy or monetary cost.

C. Baselines

At minimum, a rule-only baseline should be reported when evaluating a hybrid framework. Additional baselines may include cloud-only placement, edge-only placement, or another published method when it can be implemented fairly. The baseline must use the same workload and measurement procedure as the proposed approach. This is essential when interpreting whether adaptation actually improves performance.

D. Prototype Alignment

The project draft proposes a Python-oriented implementation with simulated IoT data, Mosquitto MQTT, Flask-based edge/orchestration logic, Scikit-learn for the ML component, SQLite or Firebase for persistence, and an HTML/CSS/JavaScript dashboard using Chart.js. These technologies are retained here as an implementation path rather than as a requirement of the review methodology. The final paper should report only components that are genuinely implemented and tested.

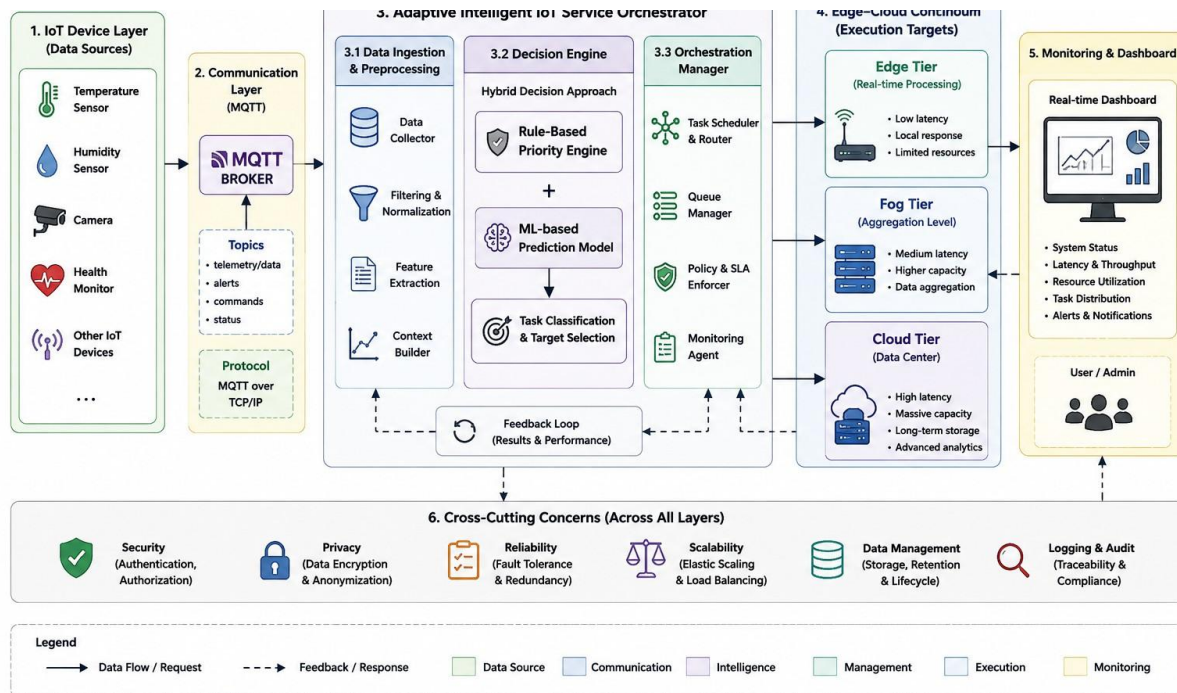


Figure 2: Task-centric reference architecture for the proposed Adaptive Intelligent IoT Service Orchestrator.

VIII. OPEN CHALLENGES AND 6G RESEARCH DIRECTIONS

A. Dynamic and Non-Stationary Workloads

Training data can become stale when traffic patterns, resource availability, or application behavior changes. Future orchestrators therefore need drift monitoring and periodic validation rather than assuming that one trained model remains optimal indefinitely. Recent reviews of learning-based offloading identify adaptability as both an opportunity and a challenge [14], [15].

B. Explainability and Safety

A placement decision for a safety-critical task may require a clear rationale. Hybrid policies can reserve explicit rules for hard constraints while allowing ML to optimize softer objectives. This division can make the decision process easier to validate than an entirely opaque controller.

C. Resource Heterogeneity and Scalability

As the continuum grows, an orchestrator may have to choose among large numbers of devices and nodes with different capabilities. Optimization approaches can face increasing search complexity, while learning approaches can require representative data from diverse states [8], [10]. Hierarchical

or decentralized orchestration is therefore a promising direction.

D. Security, Privacy, and Trust

Sensor data may contain sensitive information, and orchestration itself becomes a control point that can be attacked or manipulated. Security and privacy constraints should therefore be included in placement decisions rather than treated only as an external concern. The IEC-continuum literature identifies security, privacy, interoperability, and reliability as persistent challenges [7], [11].

E. 6G-Aware Orchestration

6G research increasingly links wireless networking with distributed AI and pervasive intelligence. In such systems, orchestration may need to incorporate richer network context, mobility, sensing, distributed learning, and resilience requirements [12], [16]. A future orchestrator should therefore evolve from simple edge-versus-cloud rules toward multi-objective policies that understand application intent and resource context across the continuum.

F. Future Research Agenda

Based on the reviewed literature, five directions deserve further attention: (1) standardized benchmark workloads and evaluation procedures for continuum orchestration; (2) explainable ML or hybrid control policies for safety-critical IoT; (3) federated or privacy-aware learning across heterogeneous edge nodes; (4) scalable hierarchical orchestration across many continuum tiers; and (5) experimentally validated 6G-aware orchestration under realistic mobility and network variability. The first four directions are already visible across current reviews and studies, while the fifth becomes increasingly relevant as 6G architectures mature [11], [15], [16].

IX. CONCLUSION

This paper presented a structured review of IoT service orchestration across the edge-fog-cloud continuum with a task-centric emphasis. The literature shows a clear evolution from centralized cloud processing toward heterogeneous computing environments in which tasks can execute close to data sources or in larger centralized resources according to their requirements. Service orchestration is consequently a multi-factor decision problem involving latency, resource state, workload characteristics, network conditions, energy, cost, security, and reliability.

The review synthesized representative rule-based, optimization-based, learning-based, and hybrid approaches and highlighted the trade-offs between predictability, adaptability, computational overhead, and data requirements. It also positioned a reference hybrid framework that uses a deterministic priority path for urgent tasks and learning-assisted placement for other workloads. This framework is intended as a practical bridge between literature findings and prototype development rather than as a claim of experimental superiority. Future validation should use common workloads and transparent baselines so that any measured benefits can be attributed to the orchestration mechanism rather than to differences in experimental setup.

ACKNOWLEDGMENT

The authors thank the Department of Computer Engineering, Sandip Institute of Technology and Research Centre, Nashik, Maharashtra, India, and the project guide for support and guidance during this study.

REFERENCES

- [1] W. Shi and S. Dustdar, "The promise of edge computing," *Computer*, vol. 49, no. 5, pp. 78–81, May 2016, doi: 10.1109/MC.2016.145.
- [2] S. Yi, Z. Hao, Z. Qin, and Q. Li, "Fog computing: Platform and applications," in *Proc. 3rd IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*, Nov. 2015, pp. 73–78, doi: 10.1109/HOTWEB.2015.22.
- [3] S. Moreschini, F. Pecorelli, X. Li, S. Naz, D. Hästbacka, and D. Taibi, "Cloud continuum: The definition," *IEEE Access*, vol. 10, pp. 131876–131886, 2022, doi: 10.1109/ACCESS.2022.3229185.
- [4] L. Kong, J. Tan, J. Huang, G. Chen, S. Wang, X. Jin, P. Zeng, M. Khan, and S. K. Das, "Edge-computing-driven Internet of Things: A survey," *ACM Computing Surveys*, vol. 55, no. 8, Art. no. 174, pp. 1–41, 2023, doi: 10.1145/3555308.
- [5] A. Ullah, T. Kiss, J. Kovács, F. Tusa, J. Deslauriers, H. Dagdeviren, R. Arjun, and H. Hamzeh, "Orchestration in the cloud-to-things compute continuum: Taxonomy, survey and future directions," *Journal of Cloud Computing*, vol. 12, Art. no. 135, 2023, doi: 10.1186/s13677-023-00516-5.
- [6] Y. Chiang, Y. Zhang, H. Luo, T.-Y. Chen, G.-H. Chen, H.-T. Chen, Y.-J. Wang, H.-Y. Wei, and C.-T. Chou, "Management and orchestration of edge computing for IoT: A comprehensive survey," *IEEE Internet of Things Journal*, vol. 10, no. 16, pp. 14307–14331, 2023, doi: 10.1109/JIOT.2023.3245611.
- [7] L. F. Bittencourt, R. Immich, R. Sakellariou, N. Fonseca, E. Madeira, M. Curado, L. Villas, L. daSilva, C. Lee, and O. F. Rana, "The Internet of Things, fog and cloud continuum: Integration and challenges," *Internet of Things*, vols. 3–4, pp. 134–155, 2018, doi: 10.1016/j.iot.2018.09.005.
- [8] A. Brogi, S. Forti, C. Guerrero, and I. Lera, "How to place your apps in the fog: State of the art and open challenges," *Software: Practice and Experience*, vol. 50, no. 5, pp. 719–740, 2020, doi: 10.1002/spe.2766.
- [9] K. Fu, W. Zhang, Q. Chen, D. Zeng, and M. Guo, "Adaptive resource efficient microservice deployment in cloud-edge continuum," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 8, pp. 1825–1840, Aug. 2022, doi: 10.1109/TPDS.2021.3128037.
- [10] I. Cohen, C.-F. Chiasserini, P. Giaccone, and G. Scalosub, "Dynamic service provisioning in the edge-cloud continuum

with bounded resources,” *IEEE/ACM Transactions on Networking*, vol. 31, no. 6, pp. 3096–3111, Dec. 2023, doi: 10.1109/TNET.2023.3271674.

[11] P. Gkonis, A. Giannopoulos, P. Trakadas, X. Masip-Bruin, and F. D’Andria, “A survey on IoT-edge-cloud continuum systems: Status, challenges, use cases, and open issues,” *Future Internet*, vol. 15, no. 12, Art. no. 383, 2023, doi: 10.3390/fi15120383.

[12] F. Guo, F. R. Yu, H. Zhang, X. Li, H. Ji, and V. C. M. Leung, “Enabling massive IoT toward 6G: A comprehensive survey,” *IEEE Internet of Things Journal*, vol. 8, no. 15, pp. 11891–11915, Aug. 2021, doi: 10.1109/JIOT.2021.3063686.

[13] D. Wang, K. B. Abu Bakar, B. Isyaku, T. A. E. Eisa, and A. Abdelmaboud, “A comprehensive review on internet of things task offloading in multi-access edge computing,” *Heliyon*, vol. 10, no. 9, Art. no. e29916, 2024, doi: 10.1016/j.heliyon.2024.e29916.

[14] S. Dong, J. Tang, K. Abbas, R. Hou, J. Kamruzzaman, L. Rutkowski, and R. Buyya, “Task offloading strategies for mobile edge computing: A survey,” *Computer Networks*, vol. 254, Art. no. 110791, 2024, doi: 10.1016/j.comnet.2024.110791.

[15] G. S. Chhabra, G. N. V. Rajareddy, A. Mahapatra, S. S. Mangalampalli, K. S. Sahoo, D. Sethi, and K. Mishra, “Deep learning-centric task offloading in IoT-fog-cloud continuum: A state-of-the-art review, open research issues, and future directions,” *IEEE Access*, vol. 13, pp. 144241–144270, Aug. 2025, doi: 10.1109/ACCESS.2025.3599190.

[16] H. Pennanen, T. Hänninen, O. Tervo, A. Tölli, and M. Latva-Aho, “6G: The intelligent network of everything,” *IEEE Access*, vol. 13, pp. 1319–1421, 2025, doi: 10.1109/ACCESS.2024.3521579.

[17] H. F. Shahid, B. Akdemir, J. Islam, I. Ahmad, and E. Harjula, “IoT service orchestration in edge-cloud continuum with 6G: A review,” *IEEE Internet of Things Journal*, vol. 13, no. 10, pp. 20339–20358, 2026, doi: 10.1109/JIOT.2026.3663918.