Secure and Cost-Efficient Microservice Placement in MEC Through Network-Aware Adaptation

Secure and Cost-Efficient Microservice Placement in MEC Through Network-Aware Adaptation

FENG HE, University of Bologna, Bologna, Italy
ARMIR BUJARI, University of Bologna, Bologna, Italy
PAOLO BELLAVISTA, University of Bologna, Bologna, Italy
PEISONG LI, Xidian University, Xi'an, China
ZIREN XIAO, Loughborough University, Loughborough, United Kingdom of Great Britain and Northern Ireland
NICOLÒ ROMANDINI, University of Bologna, Bologna, Italy
YU QIU, South China University of Technology, Guangzhou, China

Mobile Edge Computing (MEC) facilitates low-latency service delivery by bringing computation to the network edge, while microservice architectures enhance system flexibility and scalability through modular application decomposition. Their integration allows applications to dynamically scale and efficiently isolate faults in response to fluctuating demand. However, the heterogeneous and dynamic nature of edge environments poses significant challenges to the cost efficiency, security, and privacy of such deployments. Unlike prior work that handle security and privacy as constraints or overlook AN dynamics, we characterize both AN selection and service placement as mutually dependent decision layers within a unified control structure. Our proposal balances the computational overheads incurred by security and privacy verification against deployment costs, preventing aggressive cost-reduction efforts from compromising system protections. Considering the unpredictability of run-time and network conditions, we propose an online algorithm for progressive decision-making. The deployment problem is decomposed into one-slot optimization sub-problems, each NP-hard, for which we develop a greedy-based heuristic algorithm. Experiments across diverse loads, request patterns, and security/privacy thresholds show that the proposed algorithm consistently achieves the lowest average deployment cost, with cost reductions of at least 41.67% compared with the baselines, while maintaining comparable or higher request acceptance ratios and generally lower, well-balanced edge-server workloads.

CCS Concepts: • **Networks** → **Network Services**; • **Security and Privacy** → **Security Services**.

Additional Key Words and Phrases: Service Management, Mobile Edge Computing, Security, Privacy, Resource Availability, Access Network Selection

1 Introduction

The widespread deployment of 5G and the Internet of Things (IoT) has driven the development of resource-intensive applications such as autonomous driving, smart cities, medical diagnosis, virtual/augmented reality (VR/AR) and online games [4, 27, 33, 34]. These applications are increasingly latency-sensitive and computationally demanding. Although cloud computing scales computation according to demand, its centralized resource

Authors' Contact Information: Feng He, University of Bologna, Bologna, Emilia-Romagna, Italy; e-mail: feng.he2@unibo.it; Armir Bujari, University of Bologna, Bologna, Emilia-Romagna, Italy; e-mail: armir.bujari@unibo.it; Paolo Bellavista, University of Bologna, Bologna, Emilia-Romagna, Italy; e-mail: paolo.bellavista@unibo.it; Peisong Li, Xidian University, Xi'an, Shanxi, China; e-mail: lipeisong@xidian.edu.cn; Ziren Xiao, Loughborough University, Loughborough, England, United Kingdom of Great Britain and Northern Ireland; e-mail: z.xiao@lboro.ac.uk; Nicolò Romandini, University of Bologna, Bologna, Emilia-Romagna, Italy; e-mail: nicolo.romandini@unibo.it; Yu Qiu, South China University of Technology, Guangzhou, Guangdong, China; e-mail: qyu_edu@163.com.

ACM 1557-6051/2026/5-ART
https://doi.org/10.1145/3815782

provisioning model often results in high energy consumption and resource inefficiencies, making it less suitable for applications embodying stringent latency and bandwidth requirements. To address these challenges, multi-access edge computing (MEC) has emerged as a complementary paradigm, allowing the deployment of computational resources also at the network edge, thus improving Quality of Service (QoS) in various application scenarios [13].

MEC represents a significant evolution of the network architecture and offers potential for innovation in service paradigms, provided that application- and service-level QoS requirements can be consistently guaranteed throughout their operational lifetime. Microservice-oriented architectures, which decompose applications into loosely coupled independent service modules, could fully capture the benefits of MEC, delivering services to end-users without reliance on backbone transmissions, while allowing each functional module to be developed, deployed, and scaled independently. This approach has the potential to improve overall resource efficiency and scalability, assuming the presence of intelligent resource management approaches [5].

Although desirable, the variability of user requests and the resource constraints of edge servers impose challenging requirements on microservice deployment. A service-management technique should balance users' need for high-quality service and network operators' need to budget the resource quota required to provide that service. Consequently, research on effective microservice deployment and management strategies, capable of jointly considering different aspects of the problem, is paramount. In addition, security and privacy of the deployment and user data should be taken into consideration: deploying low-security microservices on critical infrastructure nodes may limit the availability of suitable deployment locations for subsequent microservices with higher security requirements, thereby impacting overall system efficiency and security. Here, we take a general stance on security and privacy in relation to microservice placement, defining security as the assessment of security capabilities across multiple servers, ensuring that tasks are assigned to trustworthy computing nodes in accordance with their security requirements, while privacy concerns the protection of sensitive user data, i.e. geographic location, during task scheduling and deployment.

A plethora of studies have extensively and successfully explored aspects of QoS optimization in edge scenarios, such as minimizing latency, maximizing throughput, and improving cost efficiency [11, 12, 26, 35, 44], however, they exhibit two main limitations. Firstly, most research proposals have overlooked the uncertainty and fluctuating resource demand and network parameters in their algorithms, despite user behavior and network topology that are dynamically evolving and often difficult to fully capture. We argue that algorithmic approaches capable of dynamically adjusting under uncertain input to ensure stability in unpredictable environments are paramount. Secondly, many studies have emphasized optimizing service placement to improve QoS but have overlooked the impact of Access Network (AN) selection on QoS. In addition, ensuring the security and privacy of microservice deployments, especially in multi-domain multi-stakeholder environments, is critical. As services grow more personalized and tasks are dynamically distributed, even if optimal QoS is achieved, overall system effectiveness can be severely undermined by risks such as malicious attacks on the hosting infrastructure or application/service data tampering.

For the sake of clarity, and without loss of generality, we consider a general system model where different (decentralized) ANs, equipped with computational capability, that is, edge nodes co-located with the AN, deliver services to end-users. Figure 1 illustrates a representative scenario, where a MEC operator supports privacy-sensitive microservice applications such as privacy-preserving video analytics for public-space monitoring (e.g., in a campus or shopping mall). End users (e.g., security staff using mobile devices or fixed cameras) offload their processing to nearby edge servers to avoid sending raw data to a distant cloud. Each request corresponds to a single linear microservice chain (e.g., frame pre-processing, anonymization, and deep neural network–based analysis), and in the setting considered in this paper all microservices in a chain are co-located on the same edge server. The MEC network comprises nine edge servers (*ES*) and five users; each edge server deployment consists of an AN and a local cloud. When $User_1$ issues a service request to be served via the appropriate microservice chain,

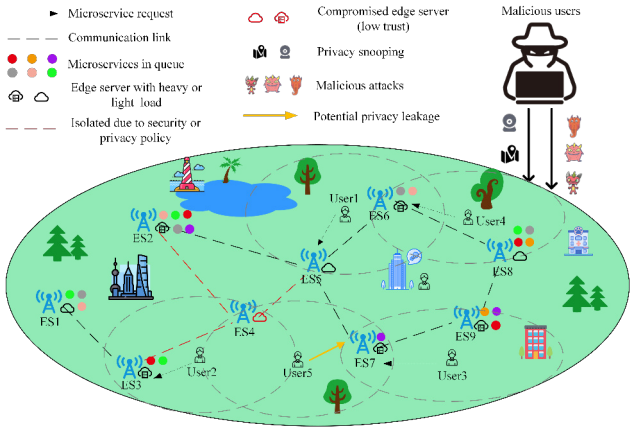


Fig. 1. An example illustrating AN selection and security/privacy threats in a MEC network. ES_4 represents a compromised edge server (e.g., subject to internal attacks such as selective packet dropping, data tampering or denial-of-service), and is therefore assigned a low security level S_j. $User_2$ must avoid ES_4 and requests are re-routed and serviced by ES_3, at the cost of longer queues and higher cost. $User_5$ is rejected because ES_4 is isolated and there is a potential privacy leakage between $User_5$ and ES_7 (e.g., adversaries inferring $User_5$'s position from channel observations along that link). Meanwhile, isolating ES_4 also disconnects the subnet $\{ES_1, ES_3\}$ from the rest of the network, which may trigger cascading failures on the remaining servers.

it can connect to either ES_5 or ES_6 within its transmission range. Although geographically closer to ES_6, $User_1$ prioritizes ES_5 to benefit from a shorter queuing delay. Similarly, $User_2 \sim User_5$ can select ES_4, ES_7, ES_6, and ES_7, respectively, based on the same rationale. However, security and privacy constraints significantly complicate these decisions. For instance, if ES_4 is compromised by internal attacks, $User_2$ must avoid it and reroute requests to ES_3, accepting higher costs to ensure security. Simultaneously, $User_5$ might be rejected if the link to ES_7 poses a privacy leakage risk. Crucially, isolating such compromised nodes, e.g., ES_4, can disconnect subnets, e.g., ES_1 and ES_3, from the rest of the network. This disconnection may trigger cascading failures, as tasks originally assigned to ES_4 are redistributed to remaining servers, potentially overloading them and leading to system-wide degradation. This example illustrates how AN selection, security/privacy constraints, and cascading failures interact in our problem setting and motivates the need for a network-aware, security- and privacy-preserving microservice placement strategy. Although some studies have addressed security and privacy aspects [28, 30, 32, 40, 41, 43], comprehensive approaches integrating both dimensions remain scarce. Therefore, to provide reliable service delivery, microservice deployment strategies must carefully integrate security and privacy considerations.

To this end, we formulate an Online Microservice Placement Problem (OMPP) that, to the best of our knowledge, is the first work to address the inherent trade-off between minimizing the average deployment cost and guaranteeing security and privacy in MEC. Specifically, our approach explicitly balances the computational overheads incurred by rigorous security and privacy verification against the fundamental deployment costs, preventing aggressive cost-reduction efforts from compromising system protections. Importantly, we account for the impact of AN selection, e.g., the AN with the fewest queued requests, and strike a tradeoff: instead of handling AN selection and service placement as independent functions, OMPP jointly decides which AN serces each request select and on which edge server its microservice chain should be deployed. These coupled decisions are taken online under in an evolving network environment, that is our approach performs a network-aware adaptation considering AN status and the fluctuating set of trusted servers. These characteristics raise several

intertwined challenges: (*i*) how to orchestrate AN selection in a congestion-aware and cost-efficient way; (*ii*) how to incorporate security and privacy mechanism into server selection without exploding algorithm overhead; and (*iii*) how to design an online decision-making mechanism that balances cost, security, and privacy, while remaining resilient to cascading node isolation and stochastic request arrivals.

To addresses above challenges, we develop a network-aware online algorithm and cost model that jointly account for AN dynamics, security- and privacy-induced logical connectivity, and compute-side deployment overheads. Given the long-term optimization challenge and the difficulty of predicting future network conditions, we decompose the OMPP into a sequence of 1-slot Microservice Placement Problems (1sMPP), each proven to be NP-hard. To solve each 1sMPP efficiently, we propose a novel greedy-based heuristic algorithm, shown to outperform traditional algorithms. In particular, extensive experiments based on a real-world dataset demonstrate that the proposed approach outperforms all baselines in terms of the request acceptance ratio, the average deployment costs, and edge server load when considering potential security and privacy threats.

The remainder of the paper is organized as follows: Section II surveys prior works in the field, positioning our work regarding existing microservice deployment schemes. Section III introduces our unified model that embeds security, privacy, and AN selection into a single cost optimization problem. Section IV builds on this formulation to present an online solution, which decomposes the problem into per-slot 1sMPP subproblems and solving each with a greedy-based heuristic integrated into this online algorithm. Section V presents the experimental results, while Section VI draws the conclusions and delineates future work.

2 Background and Related Work

In recent years, significant progress has been made in the study of microservice deployment in MEC environments [9, 23, 38, 44]. Centralized optimization methods typically formulate deployment as integer linear or mixed integer problems. For example, Wang *et al* [36] proposed a two-layer service placement mechanism based on integer linear programming (ILP) and heuristic algorithms to reduce deployment costs while maintaining latency constraints. Zeng *et al* [44] addressed microservice dependency and reuse through a layer pulling mechanism based on a transformed linearized model. However, these works operate in static settings with known traffic demands, and typically optimize a single performance metric (e.g., cost, latency, or energy) under fixed network parameters. Their cost models do not account for the deployment-level security and privacy overhead, nor AN selection and inherent trade-offs between such overhead and placement cost.

To address system scalability and coordination, distributed and cooperative methods have also been explored. For example, Chen *et al* [9] introduced a collaborative service placement model using parallel Gibbs sampling in dense small-cell networks, considering spatial coupling and base station self-interest. Similarity, the authors in [23] proposes a Distributed Fog Service Placement (DFSP) algorithm based on iterative combinatorial auctions to address the microservice resource allocation problem in fog computing environments, aiming to optimize energy consumption and communication cost. Although these methods enhance deployment flexibility, they rely on prior knowledge of workload patterns and assume a fixed network topology. Consequently, they are unable to react to short-term variations in access conditions or sudden changes in security policies. Moreover, deployment cost is treated as a purely resource-oriented metric, failing to capture the trade-offs between deployment costs and the overheads of security and privacy.

Methods for deploying microservices without prior knowledge of future arrivals have been intensively studied, achieving several results in recent years. In this context, the authors in [11] studied the microservice deployment problem with users moving continuously in MEC. They propose an integrated approach combining mobility prediction and service composition for microservice predeployment, aiming to enhance the success rate of multi-user requests and responses while reducing the resource costs. He *et al.* [18] investigated the microservice deployment problem in MEC by proposing an online algorithm with the aim of minimizing the response time of

Table 1. SYNOPSIS SUMMARIZING THE RELATED WORK

Work	Optimization objective	Techniques	Resources	Security	Privacy	Decision Mode
[36]	CPU, memory, bandwidth and number of ESPs	Multi objective optimization	~	✗	✗	Online
[44]	Total cost including deployment and operation cost	Randomized Rounding-based algorithm (RR-MDLP)	~	✗	✗	Offline
[9]	Total utility	Cooperative Service Placement (CSP)	~	✗	✗	Online
[23]	Energy consumption and communication costs	Game-theoretic approximation method	✓	✗	✗	Online
[11]	Response rate and resource cost rate	Approach combining mobility prediction and service composition	~	✗	✗	Offline
[18]	Response time	Online algorithm	✓	✗	✗	Online
[3, 8, 29, 35, 45, 47]	QoS	ML-based methods (RL, LA, GCN, DRL, GNN)	✗	✗	✗	Offline
[30]	Number of security policy violations	Policy-as-Code Framework	~	✓	✗	Online
[28]	security risk and load imbalance	SmCPA	~	✓	✗	Offline
[43]	Efficiency and accuracy	Novel designed architectural	✗	✓	✗	Offline
[1]	Threat detection accuracy and Latency	Fed-TH (federated deep learning model)	~	✓	✓	Offline
[41]	Overall ECU performance	SPEA2, TOPSIS and MCDM	✓	~	✓	Offline
[32]	number of requests	FL and Greedy Algorithm	✓	✗	✓	Offline
Our work	Average deployment cost	Online heuristic for OMPP	~	✓	✓	Online

Note: (i) Resources include CPU, memory, and storage.
(ii) ✗, ✓, and ~ denote full, no, and partial consideration, for each aspect.

the services. However, these online schemes focus exclusively on QoS and resource efficiency, without modeling security and privacy requirements, or the additional cost induced by these considerations. Furthermore, the AN selection is overly simplified by considering the nearest AN candidate or by considering static routes rather than being treated as part of a joint decision space.

Recently, data-driven approaches, such as reinforcement learning and graph-based models, have been widely studied for microservice deployment in MEC [3, 8, 29, 35, 45, 47]. Although these methods perform well in controlled environments, they usually optimize QoS-oriented metrics such as latency or throughput under simplified and often quasi-static assumptions. They require extensive offline training (often involving thousands of iterations), careful reward design, and even retraining when traffic patterns (e.g., bursty or steady) or system constraints change. In addition, existing ML-based schemes seldom model deployment-level security and privacy requirements, and they rarely incorporate AN selection or changes in logical server availability (e.g., node isolation caused by security policies) into their decision models. As a result, these methods may suffer from outdated policies, unstable convergence, or prohibitive retraining overheads when facing non-stationary request arrivals and node isolation driven by security and privacy.

Security-aware microservice deployment has been explored in few recent studies [1, 28, 30, 40, 43]. Pallewatta *et al.* [30] introduced a Policy-as-Code framework for enforcing secure microservice orchestration in multi-domain edge environments. Li *et al.* [28] focused on mitigating co-residency attacks in containerized clouds through a placement strategy driven by service invocation criticality. Zdun *et al.* [43] addressed the challenge of building secure microservice systems by proposing a method that improves efficiency and accuracy by automatically detecting and evaluating architectural designs. The authors in [1] applied a federated learning model to tackle network threats in Industrial Cyber-Physical Systems. Although these methods address security at a system level, they target static deployment contexts where security policies are enforced independently of fine-grained placement and access decisions. Consequently, they primarily prioritize the improvement of security properties, such as policy compliance and threat detection accuracy. However, they overlook the trade-off between deployment costs and the overheads of security and privacy in dynamic environments, and fail to explicitly model the cost incurred by satisfying security and privacy requirements.

Privacy-aware placement has received even less attention [32, 41]. Xu *et al.* [41] designed a trust-based placement algorithm to protect the data of IoT devices during edge deployment, using multi-objective optimization to balance performance and confidentiality. Qian *et al.* [32] introduced a privacy-aware placement scheme for mobile edge cloud services that safeguards user identity during access. However, both works target offline or batch placement scenarios with relatively stable service sets and network parameters. They emphasize privacy modeling (e.g., conflict graphs or federated preference learning), but neither captures AN congestion nor the joint optimization of deployment cost and privacy-induced overhead under stochastic request arrivals.

Unlike previous works, our study is designed from the ground up to address real-time, secure microservice deployment through network-aware adaptation under request uncertainty. We propose an online decision-making algorithm that jointly optimizes AN selection and edge resource allocation, and models security and privacy levels at the server side. In our system, security and privacy thresholds determine a node's eligibility to serve a given request. Crucially, these thresholds may also isolate compromised nodes from the network. This can trigger logical connectivity changes, resource fragmentation, and even cascading failures when previously central servers become unavailable. We pursue a network-aware approach, implying that AN selection and microservice placement decisions are driven by the dynamic network-side state of the MEC system, rather than relying solely on static server capacities. This state includes the time-varying set of reachable ANs $\phi_{r_v}(t)$, AN queue lengths Q_i, request arrival patterns, and the logical server availability determined by security and privacy thresholds (via the auxiliary graph $\mathbb{G}^{'}$). A comparative summary of representative methods and our work, in terms of optimization

Table 2. KEY SYMBOLS AND THEIR DESCRIPTIONS

Symbol	Description
$\mathbb{V}$	The set of edge servers
$\mathbb{A}$	The set of access networks (ANs)
$\mathbb{N}$	The set of user devices
$\phi_{r_v}(t)$	The set of available APs near the user sending the request $r_v \in \mathbb{R}(t)$ at each time slot t
R_j	The CPU processing capacity of edge server $j \in \mathbb{V}$
$R_j(v)$	The residual computing capacity at edge server j before it receives the request r_v
A_i	The ingress capacity of each AN i
$x_{r_v i}(t)$	The variable showing whether the AN i is selected for the request r_v at time slot t
$y_{r_v j}(t)$	The variable showing whether the server j is selected by the request r_v at time slot t
$x(t)$	The vector of the decision variable $x_{r_v i}(t)$
$y(t)$	The vector of the decision variable $y_{r_v j}(t)$
a_{r_v}	The ingress resource demand for AN of request r_v
s_{r_v}	The resource demand for server of request r_v
$Dist(\cdot,\cdot)$	The approximation distance between any two objects
$\mathbb{C}_{r_v}$	The total deployment cost for request r_v
S_j	The security value of the edge server j
P_{jr_v}	The privacy value between requests r_v and the edge server j
S^{thr}	The security threshold of the system
P^{thr}	The privacy threshold of the system
c_{r_v}	The total consumped resources of request r_v on the server j
Q_i	The total number of queuing requests on the AN i

objectives, techniques, decision mode and whether resources, security and privacy are explicitly modeled, is provided in Table 1.

3 System Model and Problem Formulation

3.1 System Model

We model a heterogeneous MEC-based system as an undirected graph $\mathbb{G} = \{\mathbb{V}, \mathbb{E}\}$, where $\mathbb{V} = \{1, 2, 3, \cdots, |\mathbb{V}|\}$ denotes the set of edge servers (servers) deployed at the network edge for microservice execution. Each edge server is connected to one AN, e.g., a base station or access point, that serves as entry point for user requests, handling radio access functions and queuing. When a user request arrives at AN i, it is transmitted through a high-speed link to its co-located edge server j for further computation [20, 27]. For clarity, we denote the set of ANs as $\mathbb{A}$.

The physical links between any two edge servers are represented by $\mathbb{E} = \{e_{uw} | u, w \in \mathbb{V}\}$. Each physical link in $\mathbb{E}$ has sufficient bandwidth capacity, a reasonable assumption considering that networked systems typically communicate via optical fiber, which offers high bandwidth and low transmission power consumption [20, 26]. Additionally, each AN i has an ingress capacity $A_i > 0$, abstracting the maximum radio/access capacity it can provide to incoming requests, while each edge server j possesses a CPU processing capacity $R_j > 0$ (e.g., in MHz) and a security level S_j. The value S_j is periodically computed by a security management function that aggregates communication trust and data trust metrics, as detailed in Section 3.4.1. In this work, we focus on the CPU processing capacity of edge servers and other resources such as memory and storage are assumed to be provisioned sufficiently.

Furthermore, we assume that the MEC platform running on edge servers has been virtualized using container-based lightweight virtualization technology, ensuring efficient resource sharing and allocation [7]. We target MEC-empowered microservice applications that generate lightweight control or sensing updates but require computation-intensive processing at the edge, such as privacy-preserving video analytics, immersive media rendering, or anomaly detection. These applications are implemented as sequential processing pipelines (microservice chains). In our model, all microservices belonging to one application chain are co-located and deployed together on the same edge server as a single containerized instance. Thus, each user request corresponds to one microservice-chain instance with aggregated CPU demand and security and privacy requirements, and the placement decision determines the hosting edge server. In such scenarios, the communication overhead over high-speed backhaul is negligible compared with the dominant costs of server-side computation and security/privacy mechanisms, and is therefore ignored in our cost model. Figure 1 illustrates the MEC network considering instances of security and privacy risks.

Without loss of generality, to better capture the dynamic functionality of the system, we assume that the system operates in discrete time slots over an extended period. The timeline is discretized into $t \in T = \{1, 2, 3, ...T\}$. The most relevant symbols and their descriptions are provided in Table 2. Building on this network model, the rest of this section introduces the request, security, privacy, and cost models that together define a single optimization problem, namely OMPP.

3.2 User Request and AN Selection Model

Microservice-based applications with complex call relationships can be modeled as a directed acyclic graph (DAG) $\mathbb{D} = \{\mathbb{M}, \mathbb{C}\}$, where $\mathbb{M}$ represents the set of microservices and $\mathbb{C}$ represents the directed invocation dependencies between them. In this paper, we focus on the important class of applications whose microservices form a single linear call chain (sequential composition). For such applications, each user request triggers one instance of the microservice chain, which we treat as a single deployable unit, i.e., the chain instance defined in Section 3.1. For more general DAG-structured applications, standard flow decomposition techniques [2] can be used to decompose $\mathbb{D}$ into several linear chains; extending our model to jointly place multiple interacting chains derived from a DAG is left for future work.

We assume that the captured request $\mathbb{R}(t) = \{r_1, r_2, r_3, \cdots r_v, \}$ consisting of a set of requests from user devices arrived in order without any advanced information in the time slot t, where $1 \leq v \leq |\mathbb{R}(t)|$. Each request r_v has the following attributes: $< L_{r_v}(t), S_{r_v}(t), P_{r_v}(t), a_{r_v}(t)$ and $s_{r_v}(t) >$. Here, $L_{r_v}(t)$ denotes the real location of the user device. It is noteworthy to point out that while microservice placement is inherently location-aware, no explicit user location data is shared. Instead, the system passively infers user location through network-side techniques such as cell association and signal profiling. Decisions are made at a zone or cell level, inherently preserving user location privacy. $S_{r_v}(t)$ and $P_{r_v}(t)$ represent the security and privacy demands of r_v, respectively. Their values are specified by the service type and data sensitivity. Requests that handle safety-critical control or sensitive personal information are assigned higher security and privacy demands, while less sensitive services use lower ones.

$a_{r_v}(t)$ indicate the ingress resource demand of r_v for the AN, whereas $s_{r_v}(t)$ denotes CPU processing demand of r_v at the edge server, measured in the same units as R_j. In particular, there may be several requests for the same type of microservice instances in each $\mathbb{R}(t)$. We use $\mathbb{R}(t)$ to denote the batch of independent microservice-chain requests that arrive during time slot t without any advance information on future arrivals. Here, sequentially arriving refers only to the temporal arrival order of these requests and does not imply additional functional dependencies among different requests. In each time slot t, for each $r_v \in \mathbb{R}(t)$, there are several available ANs, denoted by $\phi_{r_v}(t)$, which can communicate with the user device sending r_v.

In many current MEC deployments, AN selection and service placement are often decided by separate control functions. In contrast, in our work these two aspects are modeled as coupled decision variables. Each request r_v first attaches to one neighboring AN in its candidate set $\phi_{r_v}(t)$ for radio access and is then forwarded to an edge server for execution. Thus, the selected AN determines the ingress point through which the request enters the MEC system and, together with the auxiliary graph $\mathbb{G}'$, constrains the set of feasible edge servers for placement. For each time slot t, the system determines which AN will be selected. Define $x_{r_v i}(t)$ as a binary variable to denote the dynamic AN selection for r_v in time slot t. Specifically, $x_{r_v i}(t) = 1$ if r_v selects the AN $i \in \phi_{r_v}(t)$, and $x_{r_v i}(t) = 0$ otherwise. To minimize deployment cost, the system will always select the AN with the fewest queued requests to access the network. The specific selection criterion for ANs, based on minimizing deployment cost, will be analyzed in Section 4.2 after the cost function definition in Section 3.5. Note that, each request can only select one AN for network access in each time slot t. Thus, we have:

$$\sum_{i \in \phi_{r_v}(t)} x_{r_v i}(t) = 1, \quad \forall r_v \in \mathbb{R}(t), 1 \le v \le |\mathbb{R}(t)| \tag{1}$$

$$x_{r_v i} \in \{0, 1\}, \quad \forall i \in \phi_{r_v}(t), \forall r_v \in \mathbb{R}(t), 1 \le v \le |\mathbb{R}(t)| \tag{2}$$

At each time slot t, for each AN i, the total resource demand of requests via i can't exceed its total resource capacity A_i.

$$\sum_{r_v \in \mathbb{R}(t)} \sum_{v=1}^{|\mathbb{R}(t)|} x_{r_v i}(t) a_{r_v}(t) \le A_i, \quad \forall i \in \phi_{r_v}(t) \tag{3}$$

3.3 Microservice Deployment and Selection Model

At every time slot t, let $y_{r_v j}(t)$ represent the dynamic server selection decision. Specifically, $y_{r_v j}(t) = 1$ if r_v is deployed on the server $j \in \mathbb{V}$; otherwise, $y_{r_v j}(t) = 0$. For each request, $y_{r_v j}(t) = 0$ is chosen among servers that are reachable from its selected AN in the auxiliary graph $\mathbb{G}'$, so AN selection and server placement are jointly optimized rather than treated as independent decisions. Similar to the above AN selection model, we give the following constraints:

$$\sum_{j\in\mathbb{V}} y_{r_v j}(t) = 1, \quad \forall r_v \in \mathbb{R}(t), 1 \le v \le |\mathbb{R}(t)| \tag{4}$$

$$\sum_{r_v\in\mathbb{R}(t)} \sum_{v=1}^{|\mathbb{R}(t)|} y_{r_v j}(t) s_{r_v}(t) \le R_j, \quad \forall j \in \mathbb{V} \tag{5}$$

$$\sum_{r_v\in\mathbb{R}(t)} \sum_{v=1}^{|\mathbb{R}(t)|} y_{r_v j}(t) \ge 0, \quad \forall j \in \mathbb{V} \tag{6}$$

$$y_{r_v j} \in \{0,1\}, \quad \forall r_v \in \mathbb{R}(t), 1 \le v \le |\mathbb{R}(t)|, j \in \mathbb{V} \tag{7}$$

The first constraint ensures that each request is satisfied by an existing service deployed on the edge computing fabric. The second constraint indicates that the total resource deployed on a server cannot exceed its resource capacity. The third constraint allows multiple requests to be served by a single server.

3.4 Security and Privacy Model

In this subsection, we present the security and privacy model used in our formulation. Security is evaluated through a trust-based mechanism, in which each edge server j is assigned a trust value $S_j \in [0,1]$ that captures how confidently the MEC system believes that the server behaves correctly. This trust value is inferred from past behavioral observations of communication and data, following the trust-modeling approaches in [15, 19, 21] and detailed in Section 3.4.1. For each microservice request r_v, we similarly associate security and privacy demand $S_{r_v}(t)$ and $P_{r_v}(t)$, which encode how stringent the protection requirements are for that request. For privacy, we primarily focus on safeguarding the user's location information, which is a critical aspect of privacy in contemporary MEC research. The resulting server-side trust scores S_j and request-side demands $S_{r_v}(t)$, $P_{r_v}(t)$ are then used both in the placement constraints and in the cost model to capture the additional cost induced by stronger security and privacy guarantees.

3.4.1 Security Model. In this paper, we evaluate the security of the proposed system through communication trust and data trust, since the abnormality of the communication status and the data packages are the primary factors reflecting the performance of malicious attacks [15, 19, 21]. Our threat model focuses on internal misbehaving edge servers, which may launch attacks such as selective packet dropping, falsifying microservice data, or denial-of-service, similar to the attack model considered in [15, 21]. These behaviours are reflected in the communication and data trust used to compute the server-level security value S_j. This abstraction aligns with established trust-based security modeling adopted in distributed wireless sensor networks [15, 21], where security levels derived from monitoring data are typically used to quantify node trustworthiness. Consequently, in our placement model, we treat S_j simply as the normalized output of such an underlying trust-management subsystem.

As noted in [15, 21, 39], the purpose of communication trust evaluation is to establish and maintain trust relationships between servers by assessing the behavior of servers, thereby enhancing the security and robustness of the system. The value of communication trust is determined based on their communication trust, which can be calculated as follows [15, 21].

$$T_{j,com} = b + \frac{u}{2} \tag{8}$$

where b, d and u represent belief, disbelief and uncertainty, respectively, with $b, d, u \in [0,1]$ and $b + d + u = 1$. Furthermore, $b = \frac{s}{s+f+1}$ and $u = \frac{1}{s+f+1}$, where s and f indicate the total number of successful and failed

communication attempts, respectively. These parameters can be efficiently computed using the method proposed by [31].

Regarding data trust, the evaluation focuses on the consistency of security-relevant monitoring data (e.g., the number of abnormal packets or intrusion alerts) associated with each edge server, which in turn affects the trust of the server that created and processed the data. Following the approaches in [15, 21], we assume that the monitored data for server j follow a probability density function $f(\cdot)$ with mean μ; for count-based metrics, a Poisson distribution is a common and convenient approximation [21], but other unimodal distributions could be used without changing the structure of the model. The data trust value of server j is then defined as:

$$T_{j,data} = 2(0.5 - \int_{\mu}^{v_d} f(h)dh) = 2\int_{v_d}^{\infty} f(h)dh \tag{9}$$

where v_d denotes the observed data value for server j. Intuitively, if v_d stays close to the mean μ, $T_{j,data}$ is close to 1; large deviations from μ lead to smaller values of $T_{j,data}$, indicating lower data trust [21].

From the above discussion, the overall security level of an edge server j is obtained by aggregating communication and data trust as follows:

$$S_j = w_{com}T_{j,com} + w_{data}T_{j,data} \tag{10}$$

where $w_{com} \in [0, 1]$, $w_{data} \in [0, 1]$ and $w_{com} + w_{data} = 1$. This aggregated value S_j is used as the server-level security metric and is directly incorporated into the feasibility constraints and cost model of the OMPP problem. Notably, the security level S_j is not a generic quality score, but is explicitly derived from security-relevant evidence: abnormal communication behaviour (e.g., selective packet dropping, DoS) and inconsistencies in security monitoring data (e.g., intrusion alerts, abnormal packet statistics). Higher S_j corresponds to servers that have behaved benignly under the above threat model, whereas compromised or misbehaving servers accumulate lower trust values and are gradually excluded by the security constraints.

3.4.2 Privacy Model. In MEC systems, tasks are usually delegated to edge servers to reduce latency and save costs, but this delegation can raise privacy concerns. Specifically, the quality of the wireless channel is closely correlated with the distance between the user and the server. By analyzing the system usage pattern, it is possible to infer the channel status and user location [16, 17, 46]. In particular, when communicating with multiple MEC servers, a user's location can be pinpointed accurately. In practice, privacy can be quantified by measuring the coherence between the actual location of a mobile node and the location observed by the peer nodes [46]. Therefore, we propose the following privacy model. For any user, its location privacy can be defined as follows.

$$P_{m,n,p} = \frac{Dist(m, p)}{Dist(m, n)} \tag{11}$$

where $Dist(\cdot, \cdot)$ denotes the approximation distance between two objects, m, n and p represent the mobile user, peer node and proxy node, respectively. This notion of location-privacy risk corresponds to the threatened link between $User_5$ and ES_7 in Fig. 1.

3.5 Cost Model

The traditional cost model is insufficient to capture the complexities introduced by the requirements of our proposed system. Specifically, it overlooks several critical overheads. First, ensuring security and privacy involves the use of specialized mechanisms and resources, thereby incurring additional computational and operational costs. Second, when multiple requests are queued in the AN cloud, queueing delays introduce additional overhead.

Third, in the event that a server is subject to a security or privacy breach, it must be physically isolated to contain potential damage. This isolation can trigger cascading failures across the edge network, as tasks originally assigned to the compromised server must be redistributed to other servers, potentially causing them to become overloaded. In light of these observations, we propose modeling the computational cost of servers in an exponential way, as suggested in [6, 14], to more accurately capture the dynamic nature of resource utilization.

Notably, as mentioned in 3.1, the communication between user devices and ANs, as well as among interconnected edge servers over high-speed optical links, incurs only a trivial overhead in the overall resource consumption compared with computation-intensive microservice execution. Consequently, the cost model in (12)–(13) focuses on capturing compute-side overheads, rather than explicitly modeling per-link transmission latency or bandwidth usage. Network connectivity is instead reflected through the limited transmission range of user devices, e.g. the neighboring ANs $\phi_{r_v}(t)$of r_v, and candidate servers selection based on security and privacy, e.g. the construction of the auxiliary graph $\mathbb{G}'$, as detailed in Sections 3.2-3.4.

Let $R_j(v)$ be the residual resource on the server j before receiving the request r_v. Initially, $R_j(1) = R_j$. After receiving r_v, it is updated as $R_j(v) = R_j(v-1) - c_{j_{v-1}}$, where $c_{j_{v-1}}$ is the total computing resources consumed. Hence, for any r_v deployed on server j in time slot t, we define a comprehensive cost model that incorporates (i) the cost to enforce security and privacy measures, (ii) the cost due to queueing delays, and (iii) the deployment cost on servers. This cost model aims to provide a holistic representation of computing resources.

$$\omega_j(v) = R_j(\alpha^{1-R_j(v)/R_j} - 1) \tag{12}$$

$$c_{j_v} = S_j P_{jr_v} \cdot s_{r_v}(t) y_{r_v j}(t) + a_{r_v}(t) Q_i x_{r_v i}(t) \tag{13}$$

where $1 - R_j(v)/R_j$ is expressed as the utilization ratio of the server j, which quantifies the proportion of the server resources that are currently allocated. The parameter $\alpha > 1$ is a turning parameter indicating the sensitivity of servers to changes in workload. A higher α leads to a higher deployment cost, as indicated by Equation (12). Its increase incurs a more conservative system behavior, i.e. the system is more likely to reject the request [27]. Based on this, the average deployment cost for requests in $\mathbb{R}$ in the time slot t is calculated as follows:

$$\mathbb{C}_{r_v}(x(t), y(t)) = \frac{\sum_{j \in \mathbb{V}} \sum_{r_v \in \mathbb{R}(t)} \omega_j(v)}{|\mathbb{R}(t)|} \tag{14}$$
$$\text{s.t.}(1)(2)(3)(4)(5)(6)(7)(12)(13)$$

3.6 Problem Formulation

Combining the above elements, each request is characterized by its AN-side ingress demand, computational demand, and security/privacy requirements, while each server is characterized by its CPU capacity and security and privacy levels. The auxiliary graph encodes which servers are available under the current security and privacy thresholds, and the comprehensive cost model captures the resulting deployment, queuing, security and privacy overhead. On top of this unified model, we now formulate the overall Online Microservice Placement Problem (OMPP).

OMPP in a resource-constrained MEC system aims to optimally assign requests with distinct security and privacy demands to ANs and servers over a period T such that the average deployment cost is minimized while ensuring security and privacy, subject to the resource capacity of servers as well as the level of security and privacy of servers. To this end, the OMPP problem is formulated with the following optimization objective:

$$\min_{(X,Y)} \sum_{t=1}^{T} \mathbb{C}_{r_v}(x(t), y(t)) \tag{15}$$

$$\text{s.t.}(1)(2)(3)(4)(5)(6)(7)(12)(13)$$

$$S_j >= S_{r_v}, \quad \forall r_v \in \mathbb{R}(t), \forall j \in \mathbb{V} \tag{16}$$

$$P_{jr_v} >= P_{r_v} \quad \forall r_v \in \mathbb{R}(t), \forall j \in \mathbb{V} \tag{17}$$

$$S_j >= S_j^{thr}, \quad \forall j \in \mathbb{V} \tag{18}$$

$$P_{jr_v} >= P^{thr}, \quad \forall r_v \in \mathbb{R}(t), \forall j \in \mathbb{V} \tag{19}$$

Constraints (16) and (17) represent that the security and privacy of each received request are met. Constraints (18) and (19) indicate that each selected server meets the lowest security and privacy requirement.

This problem poses a long-term challenge due to the inherent unpredictability of future network conditions. Long-term strategies typically require iterative refinement to adapt to dynamic environments. Li et al. [26] addressed this by proposing an approach that decomposes long-term optimization into a series of discrete one-time tasks, enabling more focused and efficient problem solving. Inspired by their approach, we develop our original secure and privacy-preserving online algorithm that transforms our OMPP into a sequence of 1-slot microservice placement problems (1sMPP). The 1sMPP is defined as follows:

Problem Definition: Given a heterogeneous MEC network $\mathbb{G} = \{\mathbb{V}, \mathbb{E}\}$, a set of identical time slots t over a long time period $t \in T = \{1, 2, \cdots, |T|\}$ and a set of t-slot microservice request chains $\mathbb{R}(t)s$, each $\mathbb{R}(t) = \{r_1, r_2, \cdots, r_v\}$ with a microservice request chain in the time slot t. Each r_v is characterized by five attributes $< L_{r_v}(t), S_{r_v}(t), P_{r_v}(t), a_{r_v}(t), s_{r_v}(t) >$. *the 1-slot microservice placement problem* (1sMPP) jointly optimizes ANs selection and service placement for each request in $\mathbb{R}(t)$, with the aim of minimizing the average deployment cost while meeting the attribute requirements of each request, subject to different resource capacities of servers.

We then formulate the 1sMPP problem with the optimization objective according to Equation (15):

$$\min_{(X,Y)} \mathbb{C}_{r_v}(x(t), y(t)) \tag{20}$$

$$\text{s.t.}(1)(2)(3)(4)(5)(6)(7)(12)(13)(16)(17)(18)$$

Theorem 3.1. *The 1sMPP is NP-hard.*

Proof: We prove the NP-hardness of the 1sMPP by induction from the Generalized Assignment Problem (GAP), a classic combinatorial optimization problem, defined as follows: given a set of items $m = \{1, 2, 3...|m|\}$ and packages $n = \{1, 2, 3...|n|\}$, each package i has a capacity b_i. Assigning item j to package i incurs a cost c_{ji} and requires a resource r_{ji}. Let x_{ji} be the decision variable, where $x_{ji} = 1$ if the item j is assigned to packages i, and 0 otherwise. The objective is to assign each item to exactly one package without exceeding any package's capacity and the GAP can be formulated as follows:

$$
\begin{aligned}
&\text{MIN} \sum_{j=1}^{|m|} \sum_{i=1}^{|n|} c_{ji} x_{ji} \\
&\text{s.t.} \quad \sum_{j=1}^{|m|} r_{ji} x_{ji} \leq b_i, \quad i = 1, 2, 3 \ldots |n| \\
&\sum_{i=1}^{|n|} x_{ji} = 1, \quad j = 1, 2, 3 \ldots |m| \\
&x_{ji} \in \{0, 1\}, \quad j = 1, 2, 3 \ldots |m|, i = 1, 2, 3 \ldots |n|
\end{aligned} \tag{21}
$$

The first constraint ensures that the total consumed capacity of assigning items to packages does not exceed its capacity, while the second constraint ensures that each item is assigned to exactly one package.

We now consider a system with M edge servers, each with capacity C_p ($1 \leq p \leq M$), and F types of requests, each requiring a_f ($1 \leq f \leq F$). Each request is deployed once, and the deployment cost C_z depends on the residual capacity of the server, the level of security / privacy, and the associated AN. The goal is to assign all requests to edge servers while minimizing the average deployment cost.

This formulation aligns with GAP in structure and constraints. Since GAP is NP-hard [10], 1sMPP is NP-hard.

4 Algorithm Design

Building on the OMPP formulation in Section 3, this section presents a unified algorithmic solution rather than multiple independent schemes. First, we adopt a online optimization approach that decomposes the long-term OMPP into a sequence of per-slot 1-slot Microservice Placement Problems (1sMPP), each capturing the joint AN selection and microservice deployment decisions at one time slot. We then prove that 1sMPP is NP-hard and design a greedy-based heuristic (Algorithm 2) to efficiently approximate its solution, which is wrapped by an online controller (Algorithm 1) that updates AN selection and placement over time.

4.1 Online Algorithm for the long-term optimization problem

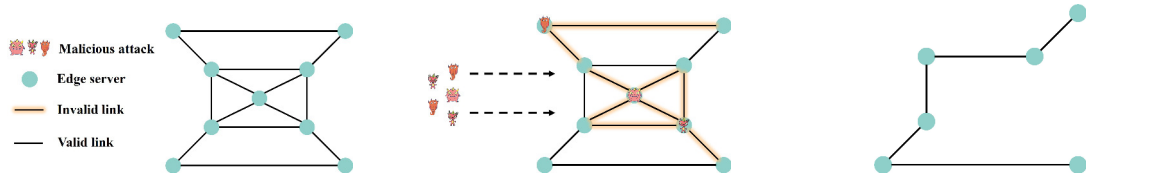
Considering the tradeoff of cost between deployment, AN selection and ensure security and privacy, an online algorithm is proposed. By jointly optimizing ANs selection and microservice deployment, its objective is to minimize the average cost while meeting security and privacy. As these decision variables are coupled in Problem (14), we jointly optimize them at each time slot t using Algorithm 2. Before presenting the main steps, we describe the construction of an auxiliary graph $\mathbb{G}' = \{\mathbb{V}', \mathbb{E}'\}$, derived from the original graph $\mathbb{G} = \{\mathbb{V}, \mathbb{E}\}$.

Given a connected graph $\mathbb{G}$ representing the MEC network and the security threshold S^{thr} of edge servers, $\mathbb{G}'$ can be constructed as follows: calculate the security value S_j for each server j using equations (8) ~ (10). If $S_j < S^{thr}$, server j and its adjacent edges are removed from $\mathbb{G}$. The remaining servers and links form the sets $\mathbb{V}' \subseteq \mathbb{V}$ and $\mathbb{E}' \subseteq \mathbb{E}$, resulting in $\mathbb{G}' = \{\mathbb{V}', \mathbb{E}'\}$. An illustrative example is shown in Figure 2.

Now, we provide a brief overview of Algorithm 1. It proceeds iteratively by addressing a sequence of one-slot optimizations over time. Initially, $x(t)$ and $y(t)$ are set as 0. In each time slot t, it first calculates the security and privacy value of all servers in $\mathbb{G}$. Using the method previously described, $\mathbb{G}'$ is constructed. Subsequently, $x(t)$ and $y(t)$ are calculated using Algorithm 2.

If $y(t) \neq y(t-1)$, the algorithm adopts the new AN selection strategy $x(t)$ and the deployment strategy $y(t)$; otherwise, it retains the previous reconfiguration. Similarly, the AN reselection strategy occurs if $x(t) \neq x(t-1)$. This process repeats until all time slots are processed.

The detailed steps are presented in Algorithm 1.



(a) The initial network. (b) A network with security and privacy risks. (c) A pure network (the obtained auxiliary graph).

Fig. 2. An example illustrating the construction of an auxiliary graph.

Algorithm 1: Online Algorithm.

Input:
$\mathbb{G} = \{\mathbb{V}, \mathbb{E}\}$, $\mathbb{R}$s, S^{thr}, P^{thr}, S_j, P_j, $x(t)$, $y(t)$.
Output:
$x(t)$, $y(t)$.
Initialize: $x(0) \leftarrow 0$ and $y(0) \leftarrow 0$.
$t = 1$.
while $t \leq T$ **do**
  Calculate the security value of each edge server in $\mathbb{G} = \{\mathbb{V}, \mathbb{E}\}$.
  Construct an auxiliary graph $\mathbb{G}' = \{\mathbb{V}', \mathbb{E}'\}$ based on the method proposed in Figure 2.
  Obtain the vector $x(t)$ and $y(t)$ by Algorithm 2.
  if $y(t) = y(t-1)$ **then**
    if $x(t) \neq x(t-1)$ **then**
      Use the new AN selection vector $x(t)$.
    end if
  else if $y(t) \neq y(t-1)$ **then**
    Use the new AN selection vector $x(t)$ and microservice deployment vector $y(t)$.
  end if
  $t = t + 1$.
end while

4.2 Greedy-Based Heuristic for our One Time-Slot Problem

As previously proven, our one-time-slot problem (1sMPP) is NP-hard, which makes global optimization infeasible for large-scale deployments. To efficiently minimize the average deployment cost, while satisfying security, privacy, and resource requirements, each request should be assigned to the server that provides the minimum deployment cost while meeting its specific requirements. According to Formulas (12)–(13), the deployment cost for placing request r_v on server j decreases as the residual capacity $R_j(v)$ of j increases and the number of queuing requests Q_i on AN i. Consequently, the algorithm prioritizes to select ANs with fewer queued requests and assigns each request to the server with the largest remaining resource capacity. This strategy promotes balanced resource utilization and cost efficiency while ensuring that all feasibility constraints are satisfied.

Next, we provide a brief introduction to the algorithm 2. The greedy heuristic executed once per time slot to handle the set of newly arriving requests $\mathbb{R}(t)$. This algorithm runs iteratively to select the optimal AN and server for each request $r_v \in \mathbb{R}(t)$. Within each iteration, all ANs $\phi_{r_v}(t)$ are sorted by the number of requests queued in

increasing order. The AN with the least queued requests, whose resource demand does not exceed its capacity, is then selected (step 2-3). This method enforces feasibility while favoring less congested ingress points, thereby reducing queue backlog and total cost. Subsequently, r_v is deployed on a specific server by finding the server j with the minimum deployment cost $min_{j\in\mathbb{V}}\{COST_{vj}\}$, where $COST_{vj}$ denotes the cost for deploying r_v on server j. The detailed steps about how to select optimal server j for each request r_v is described as follows (steps 6 -18):

Firstly, all servers are sorted in descending order of residual resources. For each server j, its residual resource capacity, security, and privacy values are computed. If three constraints are met for deploying r_v on j, the deployment cost is calculated using (12). This process continues until all sorted candidates servers are evaluated (steps 6 -12). After evaluating all servers, a set of $COST_{v_j}$s is obtained and r_v is deployed on the server with the minimum cost, i.e., $min_{j\in\mathbb{V}}\{COST_{v_j}\}$. Finally, update the selected server's residual capacity accordingly (steps 13 -18).

After all requests in $\mathbb{R}(t)$ are processed, Algorithm 2 ends and outputs two decision variables, $x(t)$ and $y(t)$. This greedy scheme ensures per-request feasibility and adaptively selects ANs and servers based on real-time load information, enabling robust performance under dynamic conditions.

Algorithm 2: Greedy-Based Heuristic Algorithm.

Input:
$\mathbb{G}' = \{\mathbb{V}', \mathbb{E}'\}, \mathbb{R}(t), \phi_{r_v}(t), R_j, P_j, S_j\ x(t), y(t)$.
Output:
$x(t), y(t)$.
1: **for** each new arrived request r_v in $\mathbb{R}(t)$ **do**
2: Sort all neighboring ANs $\phi_{r_v}(t)$ by the queued number of requests in increasing order.
3: In the sorted ANs, select the AN with the fewest queued requests that does not violate the resource capacity constraints.
4: Choose the subnetwork G' containing the selected AN from the auxiliary graph $\mathbb{G}'$.
5: Sort all edge servers into $V(G')$ by their residual resource capacity in increasing order.
6: **for** each potential edge server v_j in the sorted edge servers. **do**
7: Let $COST_{vj}$ be the total deployment cost of r_v assigned to v_j.
8: Calculate the privacy value P_{jr_v}.
9: **if** $P_{r_v} <= P_{jr_v}$ and $c_{j_v} <= R_j(v)$ and $S_{r_v} <= S_j$ and $P_{thr} <= P_{r_v}$ **then**
10: Calculate the deployment cost based on equation (12).
11: **end if**
12: **end for**
13: **if** $COST_{vj}$ == NULL **then**
14: Reject r_v.
15: Continue.
16: **end if**
17: Deploy r_v on server v_j with minimum cost, that is, $min_{j\in\mathbb{V}}\{COST_{v_j}\}$.
18: Update the residual resource capacity of the edge server v_j.
19: **end for**
20: Calculate the average deployment cost based on equation (14).
21: **return** $x(t)$ and $y(t)$.

4.3 Performance Analysis

In the following, we analyze the performance of both Algorithm 1 and Algorithm 2, including the time complexity of both algorithms, the competitive ratio of Algorithm 1, and the suboptimality discussion of Algorithm 2.

Theorem 4.1. *The time complexity of Algorithm 2 is $O(|\mathbb{R}(t)|(|\phi_{r_v}(t)|^2 + |\mathbb{V}'|^2))$, where $|\mathbb{R}(t)|$ is the number of requests at time slot t, $|\phi_{r_v}(t)|$ is the number of available ANs and $|\mathbb{V}'|$ located near the device that issued request r_v is the number of available servers.*

Proof: Algorithm 2 generates a feasible solution by assigning each request $r_v \in \mathbb{R}(t)$ to its optimal AN and server to minimize the total cost. The algorithm operates iteratively. In each iteration (Steps 1–19), for every request r_v, it selects the optimal AN and server, which requires $O(|\mathbb{R}(t)|)$ operations.

This process involves two main computational steps. First, sorting all candidate ANs $\phi_{r_v}(t)$ located near the device that issued r_v incurs a time complexity of $O(|\phi_{r_v}(t)|^2)$. Second, in the inner loop (step 6-11), iteratively selecting the optimal server for placement has a complexity of $O(|\mathbb{V}'|^2)$. Considering that there are $|R|$ requests in total, the overall time complexity of Algorithm 2 can be expressed as: $O(|\mathbb{R}(t)|(|\phi_{r_v}(t)|^2 + |\mathbb{V}'|^2))$.

Theorem 4.2. *The time complexity of Algorithm 1 is $O(|T|(|\mathbb{V}|^2 + |\mathbb{R}|(|\phi_{r_v}(t)|^2 + |\mathbb{V}'|^2)))$.*

Proof: As seen in Algorithm 1, the loop of line 3∼15 exhibits a worst-case time complexity of $O(|T|)$. The construction of the auxiliary graph takes $O(|\mathbb{V}|^2)$. According to the theorem presented before, the time complexity of Algorithm 2 is $O(|\mathbb{R}|(|\phi_{r_v}(t)|^2+|\mathbb{V}'|^2))$. Consequently, the overall time complexity of Algorithm 1 is $O(|T|(|\mathbb{V}|^2+ |\mathbb{R}|(|\phi_{r_v}(t)|^2 + |\mathbb{V}'|^2)))$.

Theorem 4.3. *Suppose that $(x^*(t), y^*(t))$ represents the optimal solution of ANs and servers selection. Algorithm 1 can generate a optimal solution with the competitive ratio α, which can be expressed as:*

$$\alpha = max \frac{max \sum_{t=1}^{T} \mathbb{C}_{r_v}(x(t), y(t))}{min \sum_{t=1}^{T} \mathbb{C}_{r_v}(x(t), y(t))}$$

Proof: We can obtain this result by the following steps:

$$\begin{aligned} max \sum_{t=1}^{T} \mathbb{C}_{r_v}(x(t), y(t)) &\geq \alpha \sum_{t=1}^{T} \mathbb{C}_{r_v}(x^*(t), y^*(t)) \\ &\geq \alpha \cdot min \sum_{t=1}^{T} \mathbb{C}_{r_v}(x(t), y(t)) \end{aligned}$$

As seen above, we can obtain that the competitive ratio is $\alpha = max \frac{max \sum_{t=1}^{T} \mathbb{C}_{r_v}(x(t), y(t))}{min \sum_{t=1}^{T} \mathbb{C}_{r_v}(x(t), y(t))}$.

Given the NP-hardness of the 1sMPP problem established in Theorem 4.3, the proposed greedy-based heuristic (Algorithm 2) ensures feasibility at each time slot and obtains a valid deployment within polynomial time. Although deriving a strict approximation or competitive bound is intractable due to the stochastic and online nature of the problem, the algorithm leverages cost monotonicity and residual-resource awareness to iteratively approach a locally optimal allocation (see problem analysis in section 4.2). This design balances computational efficiency and solution quality, providing a efficient approach for large-scale dynamic scenarios.

5 Simulation and Analysis

In this section, we evaluate the superiority of the proposed algorithm by conducting a series of extensive experiments. Firstly, we present the experimental setting and these benchmark experiments in detail. Then, we investigate the impact of several significant metrics on the performance of the proposed algorithm.

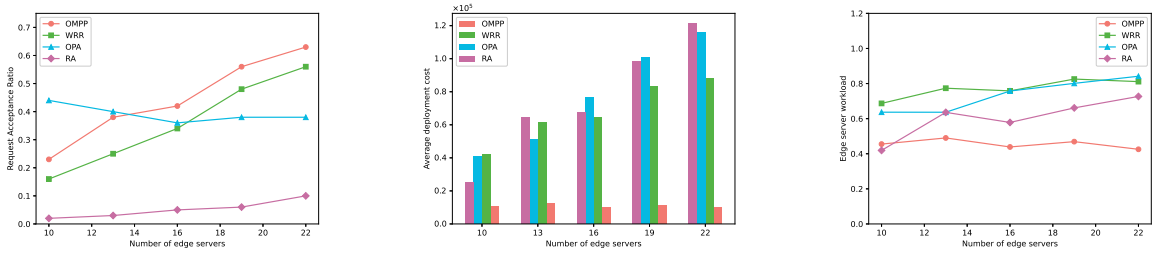
5.1 Experimental Setting

In this experiment, we consider heterogeneous MEC networks. The network topology is derived from real-world data, specifically the Melbourne Central Business District (CBD) network, which comprises 125 physical nodes. Each node co-located with an AN features a transmission range between 150m and 200m, as documented in [24]. In our setting, each such node provides radio access to nearby user devices, which offload microservice requests (e.g., IoT sensing reports, video frames, or other application data) for processing at the edge server. We assume that the physical links among edge servers remain stable and have sufficient bandwidth throughout the evaluation, which is consistent with optical link backhaul deployments in urban MEC environments. Therefore, the experiments focus on time-varying request traffic dynamics and logical connectivity changes induced by servers isolation based on security and privacy constraints rather than low-level physical link failures. To simulate a representative network scale, we initialize the system with 20 edge servers and 170 user requests, where each edge server is co-located with an AN. The capacity of each server is randomly assigned within the range of 3000MHz to 7000MHz, while the initial trust value S_j of each server is sampled uniformly from $[0, 1]$, emulating heterogeneous security postures resulting from different past behaviours (benign or malicious) as captured by the trust model described in Section 3.4.1. To ensure simplicity and generality in the modeling process, the CPU processing capacity R_j and the per-request computational demand s_{r_v} are expressed in megahertz (MHz). This CPU-based abstraction enables a unified modeling of computational demand and capacity across heterogeneous platforms and is consistent with how modern virtualized edge systems (e.g., containers, vCPUs) allocate CPU resources in practice [27]. Furthermore, the weighting factors, denoted w_{com} and w_{data}, are configured as 0.5. For each request, the capacity demand on servers is randomly distributed between 20 MHz and 300 MHz. The security demands of these requests range from 0.4 to 0.8, and privacy demands span a range from 0 to 6. To instantiate these two parameters, each request is first assigned to a discrete security and privacy level according to its service sensitivity (e.g., low, medium, high), and the exact values $S_{r_v}(t)$ and $P_{r_v}(t)$ are then sampled uniformly within the corresponding intervals. Other network parameters are defined as follows: the initial trust threshold is set to 0.4, the privacy threshold is established at 1.0, and the tuning parameter α is calculated as $\alpha = 2|V| + 2$, where $|V|$ represents the number of edge servers. These configuration settings for the above parameters and hyperparameters follow the settings commonly adopted in prior MEC and microservice placement studies [13, 15, 27, 36, 37, 42] rather than being tuned specifically for the proposed algorithm. All hyperparameters are kept fixed across the experiments unless otherwise stated, so that the observed performance differences can be attributed to the algorithmic design and online decision logic instead of parameter tuning.

Additionally, we assume that both the number of requests and the likelihood of generating a request vary across time slots. To model such temporal dynamics, we employ a time-varying Poisson process (also known as a non-homogeneous Poisson process), where the request arrival rate $\lambda(t)$ changes over time rather than remaining constant. When $\lambda(t)$ is stable, the process represents steady request arrivals; when $\lambda(t)$ fluctuates significantly across time slots, it captures bursty arrival patterns. In our simulation, $\lambda(t)$ is configured to vary between different time slots to emulate realistic dynamics in request traffic at the edge, e.g., time-varying request traffic intensity at edge servers. For the results presented in the subsequent figures, each value represents the average performance of the algorithms over 50 independent runs with identical parameter settings, i.e., number of servers and initial requests, security/privacy threshold and parameter ranges, while only the random seeds used to generate server/request attributes and request arrivals differ. Unless otherwise specified, all experiments adopt the default parameters and the steady request pattern.

5.2 Benchmark Algorithms

To evaluate the effectiveness and superiority of the proposed algorithm, we choose the following representative algorithms as baselines. For a fair comparison, all baseline algorithms were implemented under the same



(a) The request acceptance ratio. (b) The average deployment cost. (c) The workload of edge servers.

Fig. 3. Comparison of the performance of different algorithms with different number of edge servers.

security and privacy constraints as the proposed approach. This ensures that performance differences arise from algorithmic efficiency rather than the absence of security and privacy mechanisms. Following is a brief introduction of these algorithms:

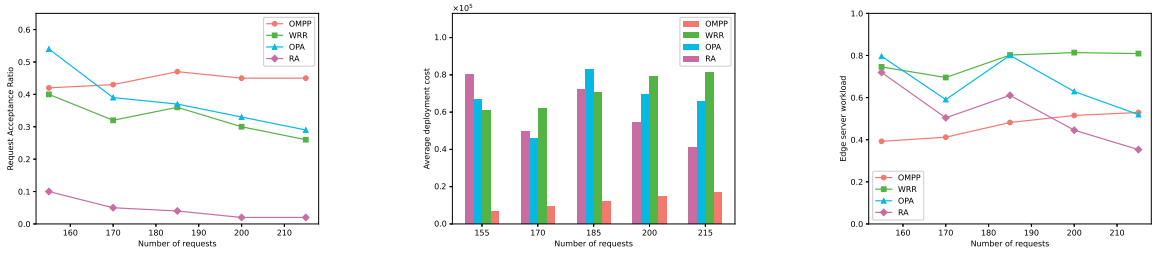
- *Random Algorithm (RA):* For each request, an AN will be randomly selected. Then, a server will also be randomly assigned.
- *Optimal Paths Algorithm (OPA) for All Requests [25]:* Inspired by the improved shortest-path routing strategy proposed in [25] for generic network environments, it iteratively selects the AN (server) for each request based on the shortest path between them using the Dijkstra algorithm.
- *Weighted Round Robin (WRR) Algorithm [22]:* Following the weighted round-robin scheduling mechanism in [22], which is designed for load balancing, it assigns a weight to each server and distributes requests proportionally to the weights. For easier comparison, we modified it to compute the weights of each edge server based on its resource capacity.

5.3 Experiment Results and Analysis

In what follows, we systematically evaluate the proposed algorithm along several complementary dimensions. Section 5.3.1 evaluates the impact of the number of edge servers (overall server capacity) on the overall performance of algorithms. Section 5.3.2 studies the algorithm's performance under stress by increasing the request load. Sections 5.3.3–5.3.4 analyze the impact of security and privacy thresholds and the resulting logical node isolation and cascading effects; Section 5.3.5 examines robustness under bursty versus steady request patterns; and Section 5.3.6 explores the advantage of the AN selection strategy. Together, these experiments assess both the overall behavior of the approach and the effectiveness of its key design components.

5.3.1 Impact of the number of edge servers. We now evaluate the impact of the number of edge servers (overall server capacity) on the performance of these algorithms against RA, OPA, and WRR, by varying the number of edge servers from 10 to 22 with an interval of 3. The results are presented in Fig. 3.

As shown in Fig. 3(a), the request acceptance ratio of all algorithms increases with the number of edge servers. This trend is expected, as a larger number of servers provides more processing capacity to handle incoming requests. When the number of edge servers is small, e.g., 10, OMPP performs worse than OPA but still outperforms WRR and RA. The motivation behind the trend can be explained by the higher traffic load per server under limited capacity, which leads to longer request queues for all algorithms. In such cases, the AN selection strategy of OMPP becomes less effective because most ANs are already heavily loaded. However, as illustrated in Fig. 3(b) and (c), under the same security and privacy thresholds, WRR and OPA incur the highest average costs and



(a) The request acceptance ratio. (b) The average deployment cost. (c) The workload of edge servers.

Fig. 4. Comparison of the performance of different algorithms with different number of newly arrival requests.

server loads. This is mainly due to their lack of adaptivity to network conditions: they fail to leverage the global system state to balance the load and select optimal servers. Additionally, although RA achieves a low server workload, only a few requests satisfying the security and privacy requirements are accepted. OMPP lies between these two extremes. Compared with the baselines, it increases the request acceptance ratio while significantly reducing costs and keeping the server workload close to that of RA simultaneously. This behavior is expected, as its well-designed resource allocation strategy not only balances the workload effectively but also ensures efficient utilization of available resources. These results further indicate that, with edge capacity extended, OMPP achieves a better trade-off: it can accommodate more requests that satisfy security and privacy requirements while avoiding the high costs and heavy workloads observed in other algorithms.

5.3.2 Impact of the number of requests. In this subsection, we analyze the impact of increasing request loads on the performance of different algorithms by varying the number of requests from 155 to 215 in increments of 15. The simulation results are presented in Fig. 4.

As shown in Fig. 4(a), as the number of requests increases, all algorithms are gradually pushed into a high-load status, but their behaviors differ significantly. The request acceptance ratio of OMPP first increases and then remains stable, indicating that when the number of requests exceeds 185, the number of requests is no longer the primary factor limiting OMPP's performance. In other words, security and privacy thresholds become the primary constraints. At the same time, OMPP demonstrates stronger robustness under heavy load, since its performance is always better than that of WRR, OPA, and RA. By contrast, the acceptance ratios of WRR and OPA steadily decline as the load grows, while RA remains at a low level. This implies that these baseline algorithms struggle under high-load conditions, as their resource allocation and AN selection strategies cannot effectively cope with stress scenarios. Figs. 4(b) and (c) further show that, under the same security and privacy thresholds, OMPP consistently achieves the lowest average cost and, in almost all configurations, it obtains the lowest server workloads. This is because OMPP is more network-aware, e.g., its AN selection strategy, and employs an effective resource allocation strategy that prevents individual nodes from becoming overloaded while improving overall resource utilization. In contrast, WRR and OPA incur higher costs and heavier workloads because they attempt to serve more requests without jointly accounting for AN queue length, residual resource capacity, and security/privacy levels in their decision process. On the other hand, RA maintains a relatively low cost and workload primarily because it accepts only a few requests that satisfy the security and privacy requirements. Overall, these results indicate that, as the number of requests increases, OMPP maintains a relatively high acceptance ratio while keeping cost and server workloads in almost all cases, thereby achieving a more favorable trade-off between cost and security/privacy than other algorithms.

5.3.3 Impact of the security threshold. The below experiments is used as stress tests of the security dimension: increasing the security threshold emulates scenarios where more servers become untrusted under our threat model, thereby shrinking the feasible deployment region and amplifying the cost impact of cascading failures. Note that the purpose of this experiment is not to re-evaluate the detection accuracy of the underlying trust mechanism itself, which has been extensively studied in prior work [30–32], but to quantify how different security levels and thresholds, which reflect varying degrees of malicious attacks in the system, affect the performance of the proposed algorithm. We vary the security threshold from 0.2 to 1 in increments of 0.2. The evaluation results, are presented in Fig. 5.

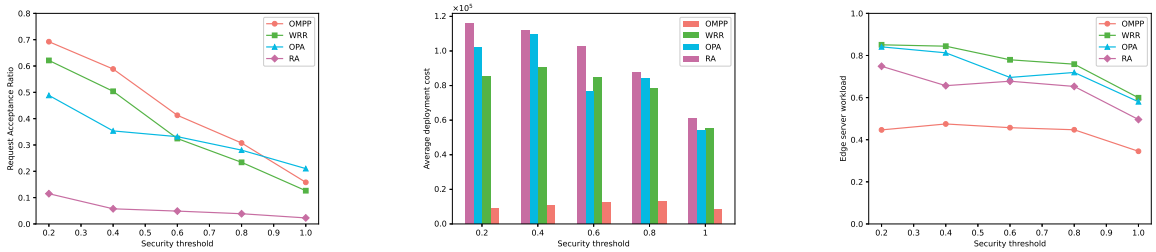
As illustrated in Fig. 5(a), the performance of all algorithms decreases as the security threshold increases. This decline occurs because a higher security threshold reduces the number of requests that meet the security requirements. Overall, OMPP consistently maintains the highest acceptance ratio across all security levels, while WRR and OPA drop more sharply and RA remains at a low level throughout. This behavior reflects the fact that higher security requirements shrink the feasible deployment space in the auxiliary graph, making it increasingly difficult for all algorithms to find feasible servers for incoming requests.

As shown in Figs. 5(b), for WRR, OPA, and RA, the average deployment cost decreases as the security threshold rises, primarily because fewer requests can be admitted and processed. In contrast, OMPP exhibits a distinct non-monotonic trend: its average cost initially increases when the threshold is raised from 0.2 to 0.8, and then gradually decreases as the threshold approaches 1.0. Crucially, this unique trend fundamentally demonstrates the trade-off governed by our proposed cost models in Eq. (12) and (13). According to Eq. (13), enforcing higher security requirements inevitably inflates the resource consumption multiplier ($S_j P_j \cdot s_{r_v}$) for individual requests. During the initial threshold increase (up to 0.8), several previously available servers are logically isolated and then incur cascading failure. To preserve compliant requests from being rejected, OMPP actively absorbs the additional security verification overheads and the cost of rerouting them to a smaller set of highly trusted servers. This temporary cost increase represents the necessary overhead to guarantee stringent security. However, as dictated by Eq. (12), aggressive placement under extreme security constraints would trigger an exponential explosion in deployment costs due to severe server resource depletion. Thus, when the threshold exceeds 0.8, OMPP adaptively rejects unfeasible requests to protect the system from cost explosions, leading to the subsequent cost decline. In stark contrast, baseline algorithms lack the capability to address this complex dynamic. WRR and OPA operate based on their static allocation strategies without considering the network dynamics and real-time resource utilization, leading them to incur unnecessarily high costs and overloaded servers, whereas RA merely achieves low costs by blindly rejecting most requests. Only OMPP successfully orchestrates the elasticity defined by Eqs. (12) and (13), validating that it effectively balances rigorous security verification overheads against fundamental deployment costs.

Throughout this process, as depicted in Fig. 5(c), OMPP maintains server workloads significantly lower and more balanced than WRR and OPA, benefiting from its joint consideration of AN queue lengths, residual resource capacity, and security requirement. The results show that different from RA, WRR and OPA, OMPP attains a more robust cost–security trade-off under increasing security requirements: it preserves a higher acceptance ratio and balanced workloads, while containing the additional cost caused by cascading node isolation.

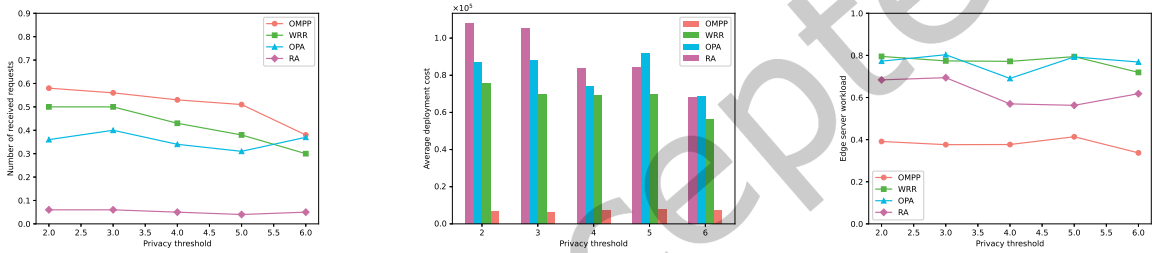
5.3.4 Impact of the privacy threshold. This subsection analyzes the impact of the privacy threshold on the overall performance of all algorithms. By gradually tightening the privacy requirement, we reduce the number of servers that can host privacy-sensitive requests, and observe how this affects each algorithm's performance. We vary privacy thresholds, ranging from 2 to 6 in increments of 1. The results, as shown in Fig. 6, illustrate the performance of each algorithm as the privacy threshold changes.

Fig. 6(a) reports the impact of increasing the privacy threshold on the request acceptance ratio. As the threshold grows gradually, the request acceptance ratio gradually decreases for all algorithms. OMPP consistently maintains



(a) The request acceptance ratio. (b) The average deployment cost. (c) The workload of edge servers.

Fig. 5. Comparison of the performance of different algorithms with different security threshold.



(a) The request acceptance ratio. (b) The average deployment cost. (c) The workload of edge servers.

Fig. 6. Comparison of the performance of different algorithms with different privacy threshold.

the highest acceptance ratio across the whole range of thresholds, whereas WRR and OPA have the similar decreasing trend and RA remains at a low level. The potential reasons for the above phenomenas is similar with Fig. 5(a).

The corresponding trends in deployment costs are shown in Fig. 6(b). Under the same security settings, the average cost of all baselines generally decreases as the privacy threshold increases, because fewer requests can be successfully placed. However, OMPP exhibits a distinct resilience: its average cost experiences a slight initial growth before stabilizing and gradually decreasing at the highest thresholds. Similar to the analysis in Fig. 5(b), this trend reflects the cost–privacy trade-off governed by Eqs. (12) and (13). As the threshold grows, the privacy multiplier (P_j) in Eq. (13) increases, and more servers become logically unusable due to potential privacy leakages. To maintain a high acceptance ratio, OMPP reroutes compliant requests to a set of available privacy-preserving servers, willfully absorbing the additional deployment costs caused by node isolation which incurs cascading failure. This slight cost increase is the essential overhead to guarantee user privacy. However, to prevent the exponential cost explosion dictated by Eq. (12) under severe resource depletion, OMPP adaptively throttles placement when the privacy constraints become overly extreme, leading to the subsequent cost stabilization. Baseline algorithms, lacking such an adaptive mechanism, either suffer from inefficient allocations (WRR and OPA) or simply achieve low costs by blindly rejecting most privacy-sensitive requests (RA).

Throughout this process, as depicted in Fig. 6(c), OMPP consistently keeps edge server workloads more stable and lower than WRR, OPA, and RA, achieving improvements of at least 46.84%, 44.93%, and 26.31%, respectively.

This benefits from its well-designed joint resource allocation and network-aware strategy, i.e., the simultaneous selection of ANs with the shortest queue length and servers with sufficient residual capacity. Overall, these results demonstrate that OMPP successfully orchestrates the elasticity defined by Eqs. (12) and (13), offering a more advantageous cost–privacy trade-off than the baselines. Different from other algorithms which almost sacrifice one performance metric to derive a better performance of other metrics, OMPP sustains a significantly higher request acceptance ratio while effectively controlling both deployment cost and server workload as the privacy threshold increases.

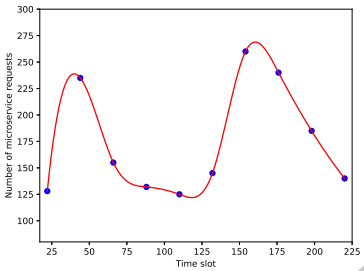
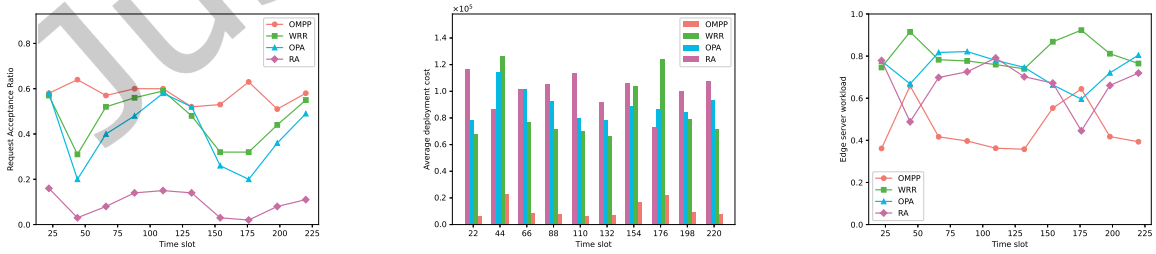


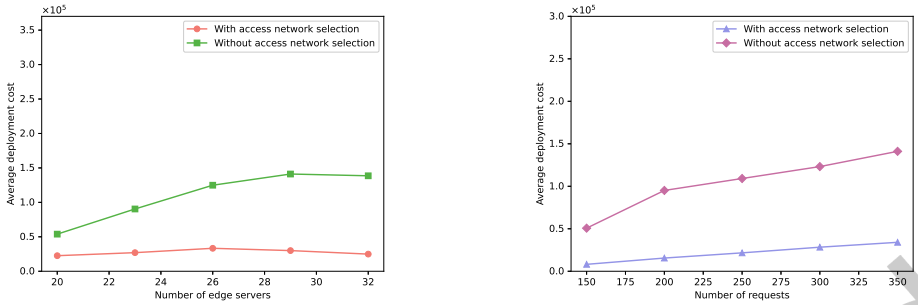
Fig. 7. Real trace-driven request arrival pattern.

5.3.5 Impact of different time slots. We analyze the impact of dynamically varying request arrivals over time by incorporating real-world workload characteristics derived from the Alibaba Cluster Trace (microservices-v2021). Specifically, the request arrival rate $\lambda(t)$ fluctuates across time slots to capture production-level traffic dynamics, where larger $\lambda(t)$ represent real-world bursty periods with dense request surges (e.g., peak operational hours), and smaller $\lambda(t)$ reflect steady, low-load intervals observed in the trace. As shown in Fig. 7, the number of newly arrived requests follows a representative dual-peak pattern extracted and appropriately scaled from the Alibaba trace. The workload curve features two distinct bursts of approximately 235 and 260 requests at the second and seventh time slots, respectively, interspersed with steady phases dropping to 125. To ensure the trace remains compatible with our experimental server capacity while preserving the non-stationary and bursty nature of the original production environment, we aggregated the raw arrival logs from multiple microservices into unified request volumes and calibrated them to fluctuate within the range of 125 to 260. Fig. 7 therefore



(a) The request acceptance ratio. (b) The average deployment cost. (c) The workload of edge servers.

Fig. 8. Performance comparison of different algorithms under the real trace-driven workload.



(a) The impact of network size on the average deployment cost.

(b) The impact of the number of newly arrival requests on the average deployment cost.

Fig. 9. Comparison of the performance of the proposed algorithm with or without AN selection strategy.

visualizes this arrival pattern used as the input for the performance comparison in Fig. 8. This range is calibrated against the server capacity configuration and the saturation behavior observed in Section 5.3.2 (Fig. 4), where the system already operates under high load when the number of requests reaches about 185 or more. By crossing this 185-request saturation threshold twice with varying intensities, the system experiences distinct transitions between stable operation and extreme over-load conditions. This experiment therefore demonstrates not only the system's adaptability under real workload-based bursty versus steady request patterns, but also the effect of varying load intensity on algorithmic performance in realistic, dynamic MEC scenarios. The corresponding results are presented in Fig. 8.

Fig. 8 reports the algorithmic performance under the real trace-driven dual-peak request pattern shown in Fig. 7. In Fig. 8(a), while the request acceptance ratios of baseline methods fluctuate heavily with the load and drop significantly during the two bursty peaks, OMPP not only consistently achieves the highest acceptance ratio but also exhibits a slight upward trend during these spikes. This shows OMPP's resilience under non-stationary real-world traffic. Conversely, WRR exhibits sharp drops during peak-load periods, OPA is highly vulnerable to traffic spikes, and RA fluctuates with a persistently low acceptance ratio. The corresponding average cost and server workloads in Figs. 8(b) and 8(c) show that WRR and OPA incur high costs and workloads during the trace's bursty stage. They admit more requests without considering AN queue lengths, residual capacity, and other constraints. On the other hand, RA shows erratic costs and moderate workloads primarily due to its randomized allocation strategy. By contrast, OMPP addresses the trace's dynamism by combining network-aware AN selection with cost-aware resource allocation to assign requests over less congested and sufficiently trusted servers. This strategy balances the surging workload, effectively preventing cost spikes suffered by baselines during peaks. Consequently, OMPP maintains low, balanced deployment costs while keeping a higher acceptance ratio in both bursty and steady periods. This demonstrates that OMPP achieves a better trade-off between cost, security, and privacy under realistic production-level stress.

Overall, across all experimental settings, RA attains low deployment cost and moderate workloads mainly because it admits only a few security- and privacy-compliant requests, whereas WRR and OPA maintain higher acceptance ratios at the expense of significantly higher cost and heavily loaded servers. Under the same security and privacy thresholds, OMPP consistently achieves a more advantageous trade-off by admitting more compliant requests while keeping the average cost and server workloads lower and more balanced than those of the baselines.

5.3.6 Impact of the access network selection strategy. As discussed earlier in this paper, a well-designed AN selection strategy can avoid network congestion and prevent inefficient resource allocation, thus reducing the total deployment cost. Now, we consider the following two cases to validate the superiority of the AN selection strategy, illustrated in Figs. 9(a) and 9(b). For comparison, we introduce an alternative algorithm, identical to OMPP except that it uses a nearest-AN selection strategy instead of the proposed strategy, which always selects ANs with the fewest queued requests. For the first case, we vary the network size from 20 to 32 in increments of 3 and fix the number of requests to 320 while the other parameters remain unchanged. For the second case, we vary the number of requests from 150 to 350 in increments of 50 and fix the number of edge servers as 25 while other parameters remain unchanged.

Figs. 9(a) and 9(b) illustrate the performance of the proposed algorithm with and without the AN selection strategy as the network size and the number of requests increase, respectively. Fig. 9(a) demonstrates the critical role of the AN selection strategy in reducing the deployment cost as the network size increases. In particular, the algorithm with the AN selection strategy always outperforms without the strategy. For example, when the network size reaches 20, the cost of the algorithm with the AN selection strategy is approximately 71.46% lower than that of the version without it. Furthermore, as the network size increases, the difference in the cost between them fluctuates more and keeps stable finally, with the AN selection algorithm demonstrating greater cost efficiency. This improvement occurs because of its optimized AN selection and resource allocation strategies, which effectively reduce server workloads.

Similarly, Fig. 9(b) reveals performance trends consistent with those observed in Fig. 9(a). The underlying reason is also similar. These results indicate the importance of a well-designed AN selection strategy in maintaining cost efficiency for deployment.

6 Limitations and Future Directions

While OMPP demonstrates robust performance, our current model relies on specific abstractions to maintain tractability. To clearly define the application scope of our proposed approach, we outline the primary limitations and their corresponding future research directions below.

First, regarding network dynamics, our work primarily focuses on the time-varying request arrivals and the logical isolation of nodes and links induced by security and privacy constraints. Building upon this, the model assumes that communication links (e.g., optical backhaul between edge servers) possess sufficient bandwidth capacity. This assumption somewhat idealizes the edge realism. In realistic MEC deployments, particularly those involving wireless backhaul or multi-hop relays, unpredictable link congestion, dynamic channel conditions, and resulting queuing delays are prevalent. Omitting the explicit modeling of these dynamic link conditions simplifies the communication overhead evaluations. As a future modeling extension, we plan to integrate stochastic queuing models and dynamic link capacity constraints to capture the impact of network fluctuations on microservice placement.

Second, our model utilizes CPU processing capacity as the singular metric for resource abstraction. Although this is a justified approach in computational-intensive scheduling, it overlooks other critical bottlenecks inherent to actual microservice deployments. In practical containerized environments, even when CPU resources are abundant, memory (RAM) exhaustion (e.g., Out-Of-Memory errors) or storage I/O constraints can inevitably lead to placement failures. Extending the resource allocation model to incorporate multi-dimensional resource constraints (e.g., jointly optimizing CPU, memory, and I/O) represents a crucial modeling extension for our future work to better reflect practical edge deployments.

Third, concerning the security model, we abstract complex malicious behaviors and node vulnerabilities into a unified trust score (S_j), which filters requests via predefined thresholds. In reality, specific cyberattacks (such as Distributed Denial-of-Service, selective packet dropping, or data tampering) not only degrade a node's

trustworthiness, but also physically perturb the network environment, altering the request arrival distributions and overall workload characteristics. Our algorithm assumes attack intensities and rates without establishing an explicit perturbation model that dynamically couples specific attack vectors with fluctuating workload profiles. To address this, a promising future direction is to design an explicit perturbation model and transition to a threat-model-driven approach, systematically evaluating algorithmic robustness under simulated cyberattacks.

Fourth, regarding algorithmic efficiency, while our evaluation provides a theoretical complexity analysis, the current study lacks an empirical runtime comparison with baseline algorithms. As a critical future step, conducting an empirical runtime validation on physical edge controllers would further strengthen the practical evaluation and quantify the actual execution time overhead introduced by the joint optimization process.

In summary, building upon the established foundation of our proposed approach, addressing these outlined directions through future theoretical and empirical extensions will further enhance its applicability in highly dynamic and realistic MEC environments.

7 Conclusion

In this paper, we investigated the problem of service provisioning in cost-efficient MEC, approaching it from the unique perspective of the antagonistic relationship between minimizing the average deployment cost and ensuring security and privacy guarantees. To cope with unpredictable network dynamics, we developed a network-aware online algorithm that jointly optimizes AN selection and microservice deployment while accounting for cascading failures induced by node isolation, as well as the demands of security and privacy. We proved that solving the resulting per-slot optimization problem to optimality is NP-hard and therefore proposed a Greedy-Based Heuristic Algorithm to efficiently approximate these joint AN selection and placement decisions. Extensive experiments on a real-world MEC topology with time-varying request loads and security/privacy thresholds demonstrate the superiority of the proposed algorithm over the baselines. Across all scenarios, our algorithm offers a more favorable trade-off between cost, security, and privacy, especially under high-load and cascading-failure conditions induced by node isolation.

References

[1] Mohamed Abdel-Basset, Hossam Hawash, and Karam Sallam. 2021. Federated threat-hunting approach for microservice-based industrial cyber-physical system. *IEEE Trans. Ind. Inform.* 18, 3 (2021), 1905–1917.

[2] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. 1988. Network flows. (1988).

[3] Işıl Karabey Aksakalli, Turgay Çelik, Ahmet Burak Can, et al. 2021. Deployment and communication patterns in microservice architectures: A systematic literature review. *J. Syst. Softw.* 180 (2021), 111014.

[4] Paolo Bellavista, Michele Brigadoi, Armir Bujari, Angelo Feraudo, and Andrea Garbugli. 2025. The Metaverse in Smart Industrial Environments: from Vision to Proof-of-Concept. In *2025 International Conference on Metaverse Computing, Networking and Applications (MetaCom)*. 340–347.

[5] Paolo Bellavista, Armir Bujari, Luca Foschini, Andrea Sabbioni, and Riccardo Venanzi. 2024. A MECApp-aware Lifecycle Management Approach in 5G Edge-Cloud Deployments. In *Proc. of International Conference on Computer Communications and Networks (ICCCN)*. 1–6.

[6] Dirk G. Cattrysse and Luk N. Van Wassenhove. 1992. A survey of algorithms for the generalized assignment problem. *Eur. J. Oper. Res.* 60, 3 (1992), 260–272.

[7] Saqib Rasool Chaudhry, Andrei Palade, Aqeel Kazmi, et al. 2020. Improved QoS at the edge using serverless computing to deploy virtual network functions. *IEEE Internet Things J.* 7, 10 (2020), 10673–10683.

[8] Lixing Chen, Yang Bai, Pan Zhou, et al. 2024. On Adaptive Edge Microservice Placement: A Reinforcement Learning Approach Endowed with Graph Comprehension. *IEEE Trans. Mob. Comput.* (2024).

[9] Lixing Chen, Cong Shen, Pan Zhou, et al. 2019. Collaborative service placement for edge computing in dense small cell networks. *IEEE Trans. Mob. Comput.* 20, 2 (2019), 377–390.

[10] Reuven Cohen, Liran Katzir, and Danny Raz. 2006. An efficient approximation for the generalized assignment problem. *Inf. Process. Lett.* 100, 4 (2006), 162–166.

[11] Jiale Deng, Bing Li, Jian Wang, and Yuqi Zhao. 2021. Microservice pre-deployment based on mobility prediction and service composition in edge. In *IEEE ICWS*. IEEE, 569–578.

[12] Shuiguang Deng, Zhengzhe Xiang, Javid Taheri, Mohammad Ali Khoshkholghi, et al. 2020. Optimal application deployment in resource constrained distributed edges. *IEEE Trans. Mob. Comput.* 20, 5 (2020), 1907–1923.
[13] E. Moro and I. Filippini. 2021. Joint management of compute and radio resources in mobile edge computing: A market equilibrium approach. *IEEE Trans. Mob. Comput.* 22, 2 (2021), 983–995.
[14] Xiuwen Fu, Haiqing Yao, and Yongsheng Yang. 2019. Modeling Cascading Failures for Wireless Sensor Networks With Node and Link Capacity. *IEEE Trans. Veh. Technol.* 68, 8 (2019), 7828–7840. doi:10.1109/TVT.2019.2925013
[15] Guangjie Han, Yu He, Jinfang Jiang, et al. 2019. A synergetic trust model based on SVM in underwater acoustic sensor networks. *IEEE Trans. Veh. Technol.* 68, 11 (2019), 11239–11247.
[16] Ting He, Ertugrul Necdet Ciftcioglu, Shiqiang Wang, et al. 2017. Location Privacy in Mobile Edge Clouds: A Chaff-Based Approach. *IEEE J. Sel. Areas Commun.* 35, 11 (2017), 2625–2636. doi:10.1109/JSAC.2017.2760179
[17] Xiaofan He, Richeng Jin, and Huaiyu Dai. 2019. Deep PDS-Learning for Privacy-Aware Offloading in MEC-Enabled IoT. *IEEE Internet Things J.* 6, 3 (2019), 4547–4555. doi:10.1109/JIOT.2018.2878718
[18] Xiang He, Zhiying Tu, Markus Wagner, et al. 2022. Online deployment algorithms for microservice systems with complex dependencies. *IEEE Trans. Cloud Comput.* 11, 2 (2022), 1746–1763.
[19] Yu He, Guangjie Han, Jinfang Jiang, et al. 2020. A trust update mechanism based on reinforcement learning in underwater acoustic sensor networks. *IEEE Trans. Mob. Comput.* 21, 3 (2020), 811–821.
[20] Meitian Huang, Weifa Liang, et al. 2019. Maximizing throughput of delay-sensitive NFV-enabled request admissions via virtualized network function placement. *IEEE Trans. Cloud Comput.* 9, 4 (2019), 1535–1548.
[21] Jinfang Jiang, Guangjie Han, Feng Wang, et al. 2014. An efficient distributed trust model for wireless sensor networks. *IEEE Trans. Parallel Distrib. Syst.* 26, 5 (2014), 1228–1237.
[22] Ajay Katangur, Somasheker Akkaladevi, and Sadiskumar Vivekanandhan. 2022. Priority weighted round robin algorithm for load balancing in the cloud. In *IEEE SmartCloud*. IEEE, 230–235.
[23] Paridhika Kayal and Jörg Liebeherr. 2019. Distributed service placement in fog computing: An iterative combinatorial auction approach. In *IEEE ICDCS*. IEEE, 2145–2156.
[24] Bo Li, Qiang He, Guangming Cui, et al. 2020. READ: Robustness-oriented edge application deployment in edge computing environment. *IEEE Trans. Serv. Comput.* 15, 3 (2020), 1746–1759.
[25] Hao Li, Xin Li, et al. 2021. Resource-Aware Service Function Chain Deployment in Cloud-Edge Environment. In *IEEE INFOCOM*. 1–6. doi:10.1109/INFOCOMWKSHPS51825.2021.9484555
[26] Jing Li, Song Guo, Weifa Liang, Jianping Wang, Quan Chen, Yue Zeng, Baoliu Ye, and Xiaohua Jia. 2023. Digital twin-enabled service provisioning in edge computing via continual learning. *IEEE Transactions on Mobile Computing* 23, 6 (2023), 7335–7350.
[27] Jing Li, Weifa Liang, Wenzheng Xu, et al. 2021. Maximizing user service satisfaction for delay-sensitive IoT applications in edge computing. *IEEE Trans. Parallel Distrib. Syst.* 33, 5 (2021), 1199–1212.
[28] Mingyang Li, Hongchao Hu, and Wenyan Liu. 2024. A Secure Container Placement Algorithm Based on Microservice Invocation Criticality. In *IEEE ICCECT*. IEEE, 234–240.
[29] Wenkai Lv, Pengfei Yang, Tianyang Zheng, et al. 2023. Graph-reinforcement-learning-based dependency-aware microservice deployment in edge computing. *IEEE Internet Things J.* 11, 1 (2023), 1604–1615.
[30] Samodha Pallewatta and Muhammad Ali Babar. 2024. Towards Secure Management of Edge-Cloud IoT Microservices using Policy as Code. *arXiv preprint arXiv:2406.18813* (2024).
[31] Evripidis Paraskevas, Tao Jiang, and John S. Baras. 2016. Trust-aware network utility optimization in multihop wireless networks with delay constraints. In *24th MED*. 593–598. doi:10.1109/MED.2016.7536054
[32] Yongfeng Qian, Long Hu, et al. 2019. Privacy-aware service placement for mobile edge computing via federated learning. *Inf. Sci.* 505 (2019), 562–570.
[33] Hamed Rahimi, Yvan Picaud, Kamal Deep Singh, et al. 2021. Design and simulation of a hybrid architecture for edge computing in 5G and beyond. *IEEE Trans. Comput.* 70, 8 (2021), 1213–1224.
[34] Pasika Ranaweera, Anca Jurcut, and Madhusanka Liyanage. 2021. MEC-enabled 5G use cases: a survey on security vulnerabilities and countermeasures. *ACM Comput. Surv.* 54, 9 (2021), 1–37.
[35] Kaustabha Ray, Ansuman Banerjee, and Nanjangud C Narendra. 2023. Learning-based microservice placement and migration for multi-access edge computing. *IEEE Trans. Netw. Serv. Manag.* 21, 2 (2023), 1969–1982.
[36] Leilei Wang, Xiaoheng Deng, Jinsong Gui, et al. 2023. Microservice-Oriented Service Placement for Mobile Edge Computing in Sustainable Internet of Vehicles. *IEEE Trans. Intell. Transp. Syst.* 24, 9 (2023), 10012–10026. doi:10.1109/TITS.2023.3274307
[37] Tian Wang, Guangxue Zhang, Anfeng Liu, et al. 2019. A Secure IoT Service Architecture With an Efficient Balance Dynamics Based on Cloud and Edge Computing. *IEEE Internet Things J.* 6, 3 (2019), 4831–4843. doi:10.1109/JIOT.2018.2870288
[38] Yimeng Wang, Cong Zhao, Shusen Yang, et al. 2020. MPCSM: Microservice placement for edge-cloud collaborative smart manufacturing. *IEEE Trans. Ind. Inform.* 17, 9 (2020), 5898–5908.

[39] Zhexiong Wei, Helen Tang, F Richard Yu, et al. 2014. Security enhancements for mobile ad hoc networks with trust management using uncertain reasoning. *IEEE Trans. Veh. Technol.* 63, 9 (2014), 4647–4658.
[40] Zhenyu Wen, Tao Lin, Renyu Yang, et al. 2019. GA-Par: Dependable microservice orchestration framework for geo-distributed clouds. *IEEE Trans. Parallel Distrib. Syst.* 31, 1 (2019), 129–143.
[41] Xiaolong Xu, Xihua Liu, Zhanyang Xu, et al. 2019. Trust-oriented IoT service placement for smart cities in edge computing. *IEEE Internet Things J.* 7, 5 (2019), 4084–4091.
[42] Zichuan Xu, Weifa Liang, Wenzheng Xu, et al. 2016. Efficient Algorithms for Capacitated Cloudlet Placements. *IEEE Trans. Parallel Distrib. Syst.* 27, 10 (2016), 2866–2880. doi:10.1109/TPDS.2015.2510638
[43] Uwe Zdun, Pierre-Jean Queval, Georg Simhandl, et al. 2023. Microservice security metrics for secure communication, identity management, and observability. *ACM Trans. Softw. Eng. Methodol.* 32, 1 (2023), 1–34.
[44] Deze Zeng, Hongmin Geng, Lin Gu, and Zhexiong Li. 2023. Layered structure aware dependent microservice placement toward cost efficient edge clouds. In *IEEE INFOCOM*. IEEE, 1–9.
[45] Yanlong Zhai, Tianhong Bao, Liehuang Zhu, et al. 2020. Toward reinforcement-learning-based service deployment of 5G mobile edge computing with request-aware scheduling. *IEEE Wirel. Commun.* 27, 1 (2020), 84–91.
[46] Ping Zhang, Mimoza Durresi, and Arjan Durresi. 2018. Mobile Privacy Protection Enhanced with Multi-access Edge Computing. In *IEEE AINA*. 724–731. doi:10.1109/AINA.2018.00109
[47] Qijun Zhu, Sichen Wang, Hualong Huang, et al. 2023. Deep-Reinforcement-Learning-Based Service Placement for Video Analysis in Edge Computing. In *IEEE ICCCBDA*. 355–359.

Received 3 July 2025; revised 28 March 2026; accepted 2 May 2026