Mobility-Aware Edge Service Scheduling with Request Heterogeneity and Server Load Balancing

Guobing Zou[1], Yile Wang[1], Song Yang[1], Shengye Pang[1(✉)], Yanglan Gan[2(✉)], and Bofeng Zhang[3]

[1] School of Computer Engineering and Science, Shanghai University, Shanghai, China
{gbzou,22721500,yangsong,pangsy}@shu.edu.cn
[2] School of Computer Science and Technology, Donghua University, Shanghai 201620, China
ylgan@dhu.edu.cn
[3] School of Computer and Information Engineering, Shanghai Polytechnic University, Shanghai 201209, China

Abstract. The service request scheduling problem in Mobile Edge Computing (MEC) often overlooks user mobility, request heterogeneity, and server load distributions, resulting in increased latency and energy consumption. To address these challenges, we introduce the Mobility-Aware Edge Service Scheduling with Request Heterogeneity and Server Load Balancing (MESS-HL) problem, considering user mobility, data volume changes of service requests, and load balancing. By proving and solving the $\mathcal{NP}$-hard MESS-HL problem, we propose a novel heuristic simulated annealing algorithm, MHLSA. It integrates user mobility trajectories, heterogeneous requests, and server load status as heuristic information into both initialization and perturbation to find an approximately optimum solution. Comprehensive experiments on two real-world datasets show that MHLSA achieves an average improvement of 31.2% in response time, 62.2% in energy consumption, and 48.9% in load balancing over existing methods.

Keywords: Mobile Edge Computing · Edge Service Scheduling · User Mobility · Request Heterogeneity · Server Load Balancing

1 Introduction

With the rapid development of 5G and Artificial Intelligence, the demand for computationally intensive and low-latency sensitive applications, such as real-time video analysis, autonomous driving, and virtual/augmented reality, has significantly increased [1]. Traditional cloud computing, relying on distant centralized data centers, often suffers from high latency and inefficiency [2]. Mobile Edge Computing (MEC) has emerged as a promising extension of cloud computing, which addresses this issue by deploying resources closer to end users, significantly reducing response time [3]. Despite its advantages, MEC faces critical challenges. First, resource constraints limit the computing capabilities of edge servers compared to cloud data centers [4]. Second, coverage limitations restrict user-server interactions within specific geographical areas [5].

D.-S. Huang et al. (Eds.): ICIC 2025, CCIS 2575, pp. 485–496, 2025.
https://doi.org/10.1007/978-981-95-0017-8_40

Although service scheduling in MEC has been extensively studied, existing approaches suffer from three major limitations. First, user mobility is often overlooked, most approaches assume static user locations at request time, ignoring mobility during execution. If the user moves out of the executing server's coverage area, it will lead to extra delays and resources to respond with the result. Second, request heterogeneity is ignored, upstream and downstream data volumes can differ significantly across edge services. Failing to account for this leads to inefficient scheduling and increased latency. Third, load balancing is often under-addressed, many approaches pursue high resource utilization but neglect load distribution, causing some servers to be overloaded while others remain underutilized, which may degrade edge system's responsiveness.

To tackle these challenges, we formulate the Mobility-Aware Edge Service Scheduling with Request Heterogeneity and Server Load Balancing (MESS-HL) problem, which explicitly incorporates user mobility, request heterogeneity of edge services, and server load balancing into consideration. Based on the transformation from a MESS-HL problem to a multi-objective optimization problem, we prove its $\mathcal{NP}$-hardness and propose a heuristic approach based on Simulated Annealing (SA) to find an approximately optimum scheduling solution, which is called Mobility, Heterogeneous Requests and Load Balancing-aware SA (MHLSA), which enhances the performance of scheduling edge services for better satisfying users' diverse service demands.

In summary, our main contributions are listed as follows:

- We formulate a novel edge service scheduling problem called MESS-HL by considering three key factors for better reflecting the situations in real applications. Also, we model it as a multi-objective optimization problem that is proven to be $\mathcal{NP}$-hard.
- We propose an improved simulated annealing algorithm named MHLSA to solve the $\mathcal{NP}$-hard MESS- HL problem, which enhances the standard SA by incorporating user mobility, heterogeneous requests of edge services, and edge server load status as heuristic information into initialization and perturbation operations.
- Our proposed method MHLSA outperforms existing approaches across all metrics in two real-world datasets. Experimental results demonstrate the superiority and generality of MHLSA for handling edge service scheduling.

2 Related Work

In recent years, edge service scheduling in MEC has garnered significant attentions. Minimizing latency is a fundamental objective in edge service scheduling. Cang et al. [6] tackled the issue of asynchronous service request arrivals in MEC by employing the Benders decomposition method, effectively improving scheduling efficiency through problem decomposition. Lou et al. [7] proposed the DCDS method to optimize dependent task scheduling under stringent deadlines, integrating long-term impact assessments of immediate scheduling decisions. However, these studies largely overlook the impact of user mobility, as users move across the coverage areas of different edge servers, response results are transferred among multiple servers, introducing additional delays.

Given the resource constraints of edge servers, resource consumption optimization is important in edge service scheduling. Ma et al. [8] introduced WiDaS, a dynamic scheduling algorithm that optimizes response time within a fixed resource budget. Jiang

et al. [9] applied a multi-task resource scheduling approach based on reinforcement learning, targeting energy minimization. However, these approaches have not considered the request heterogeneity, which may trigger additional costs like bandwidth.

Maintaining workload distribution is crucial to preventing performance degradation of edge servers. While many studies focus on maximizing resource utilization, which can lead to partial servers being overburdened while others remain underutilized [10]. Though some studies acknowledge this issue, their approaches have limitations. Dong et al. [11] proposed a Lyapunov-based online matching algorithm to balance system utility and workload distribution, yet it overlooked the heterogeneity of requests.

3 Problem Formulation

Given a set of m edge severs $\mathbb{ES} = \{es_1, es_2, ..., es_m\}$ and n edge users $\mathbb{U} = \{u_1, u_2, ..., u_n\}$ in a particular area, each user submits one edge service defined below.

Definition 1 (Edge Service). Given an edge user u, u's edge service s encompasses a specific type of request r $\in$ {UGD, ULD, UED}.

We classify request types based on edge service (ES) and its corresponding response result (RR) data volume changes: (1) **UGD Request**, like image analysis, where large input image data generates compact textual results; (2) **ULD Request**, like search engines, where lightweight queries return rich multimedia content; and (3) **UED Request**, where input and output sizes are approximately balanced, like file synchronization.

Definition 2 (Coverage Constraint). To determine whether a user u_i can interact with the edge server es_j at time t, the distance d_{ij} between them at the moment should be no more than es_j's coverage radius R_{es_j}, which is depicted as $d_{i,j}^t \leq R\left(es_j\right)$.

Definition 3 (Resources Constraint). Assuming that there have been a set of edge services $D(es_i) = \{s_{es_i}^1, s_{es_i}^2, \dots\}$ on es_i, if es_i wants to start executing the edge service s_j, it must satisfy the conditions in Eq. (1):

$$\begin{aligned} C_j + \sum\nolimits_{s_{es_i}^k \in D_{es_j}} C_k \leq C_{es} \\ M_j + \sum\nolimits_{s_{es_i}^k \in D_{es_j}} M_k \leq M_{es_i} \end{aligned} \tag{1}$$

where C_j and M_j are the computational and memory resources required by request s_j. C_{es_i} and M_{es_i} refer to the total computational and memory resources of edge server es_i.

Definition 4 (Mobility Trajectory). The user's mobility trajectories M_u is the sequence of all positions during the interaction between the user and the edge system.

In our experiments, we assume that all approaches can obtain users' mobility trajectories at any time. To reflect real-world scenarios, in Sect. 5.4, we conduct a detailed

discussion about the impact of prediction deviations on our approach, using a simulation approach inspired by the methodology presented in [12], as shown below:

$$\mathcal{M}_{\text{err}} = \mathcal{F}\left(P, \delta_x, \delta_y\right) \tag{2}$$

where P denotes the probability of accurate trajectory prediction, δ_x and δ_y represent the position offsets in the x and y directions, which both follow a Gaussian distribution.

Definition 5 (Response Time). The response time T consists of three components:

$$T = t^{exe} + t^{wait} + t^{tr} \tag{3}$$

where t^{exe} is the execution time required by the server. If a server lacks sufficient resources when the ES arrives, the request must wait before execution, denoted as t^{wait}. The mobility of users may lead to transfer delay t^{tr}. It is described separately as below.

Definition 6 (Transfer Delay). Transfer delay includes the upload of the ES and the return of the RR, which depends on hop counts between servers and affected by a bandwidth factor ϕ reflecting real-world network fluctuations, as shown below:

$$\begin{aligned} t^{tr} &= t^{tr}_{up} + t^{tr}_{down} \\ t^{tr}_{up} &= Tr^{time}_{up} \cdot hop_{u,e} \cdot \phi_{up} \\ t^{tr}_{down} &= Tr^{time}_{down} \cdot hop_{e,d} \cdot \phi_{down} \end{aligned} \tag{4}$$

where Tr^{time} indicates the ideal time for an ES and its RR to transfer between edge servers over a single hop. The bandwidth factor ϕ is defined in Eq. (5):

$$\begin{aligned} \phi_{up} &= f\left(\frac{d_{u,e}}{hop_{u,e}}\right) \\ \phi_{down} &= f\left(\frac{d_{e,d}}{hop_{e,d}}\right) \\ f(x) &= 1 + \frac{\varepsilon}{1+e^x} \end{aligned} \tag{5}$$

where d is the Euclidean distance between two servers. ε is sampled from (−0.1,0.1).

Definition 7 (Communication Cost). We define the total resources consumed by the transfer of an edge service and its corresponding response result among multiple edge servers as the Communication Cost (CC), which is expressed in Eq. (6):

$$\begin{aligned} CC^{tr} &= CC^{tr}_{up} + CC^{tr}_{down} \\ CC^{tr}_{up} &= Tr^{bd}_{up} \cdot hop_{u,e} \cdot \phi_{up} \\ CC^{tr}_{down} &= Tr^{bd}_{down} \cdot hop_{e,d} \cdot \phi_{down} \end{aligned} \tag{6}$$

where CC^{tr}_{up} and CC^{tr}_{down} represent the resources consumed by the transfer of the ES before execution and the RR after execution, respectively. Tr^{bd} is the ideal resource consumption required to transfer the ES or the RR between edge servers in one hop.

Definition 8 (Server Load Status). The load status of an *es* is defined in Eq. (7):

$$\begin{aligned} es_{load} &= es_{load}^{exe} + \theta \cdot es_{load}^{wait} \\ es_{load}^{exe} &= \frac{C_{exe}}{C} + \frac{M_{exe}}{M} \\ es_{load}^{wait} &= \frac{C_{wait}}{C} + \frac{M_{wait}}{M} \end{aligned} \tag{7}$$

where es_{load}^{exe} and es_{load}^{wait} represent the load of requests on *es* executing and waiting, respectively. Under the same conditions, we aim to keep the backlogs on the *es* as minimal as possible. Therefore, we multiply es_{load}^{wait} by a coefficient θ greater than 1.

Definition 9 (MESS-HL Problem). The MESS-HL problem can be defined as a five tuple MESS-HL $= \langle \mathbb{U}, \mathbb{ES}, \mathbb{S}, \mathbb{M}, \mathbb{L} \rangle$, where $\mathbb{U}$ is a set of users, $\mathbb{ES}$ is a set of edge servers, $\mathbb{S}$ is a set of edge services submitted by edge users, $\mathbb{M}$ is the trajectories corresponding to each user u, $\mathbb{L}$ denotes the load status corresponding to each edge server *es*.

4 Approach

4.1 MESS-HL Problem Optimization Modeling

We evaluate the scheduling performance using three key metrics: average response time (*ART*), total communication cost (*TCC*), and load balancing (*LB*). Specifically, *ART* is the mean latency experienced by users and is computed as $ART = \frac{1}{|\mathbb{U}|} \sum T$, where T is the response time for each service request based on Eq. (3). *TCC* quantifies the total communication overhead and is defined based on Eq. (6) as $TCC = \sum CC^{tr}$, with CC^{tr} representing the per-request transfer cost. *LB* assesses load balancing of the edge system and is calculated as the variance of es_{load} in Eq. (7).

Given a MESS-HL problem, we use the weighted sum method to convert the original multi-objective optimization problem into a single-objective optimization problem:

$$\min(\alpha \cdot ART + \beta \cdot TCC + \gamma \cdot LB) \tag{8}$$

s.t.

$$0 < \alpha, \beta, \gamma < 1 \tag{9}$$

$$\sum_{j=1}^{|\mathrm{ES}|} x_{i,j} = 1, \forall i \in \{1, 2, ..., n\}, j \in \{1, 2, ..., m\} \tag{10}$$

$$d_{i,j}^{t} y_{i,j}^{t} \leq R\left(es_j\right), \ \forall_i \in \{1, \ 2, \ ..., \ n\}, \ j \in \{1, \ 2 \ , ..., m\} \tag{11}$$

where $x_{i,j}$ and $y_{i,j}^{t}$ are binary variables indicating that:

$$x_{ij} = \begin{cases} 1, \text{ if } s_i \text{ is executed on } es_j \\ 0, \qquad \text{otherwise} \end{cases} \tag{12}$$

$$y_{i,j} = \begin{cases} 1, & \text{if } u_i \text{ can interact with } es_j \text{ at time } t \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

In the objective function (8), we aim to minimize the average response time and total communication costs of all edge services while considering edge system load balancing. Equation (10) imposes the constraint that each edge service must be executed and can only be executed by a single edge server. Equation (11) imposes the constraint that a user can only interact with edge servers which cover him.

4.2 NP-Hardness Proving of MESS-HL Problem

Definition 10 (CFLP). The CFLP can be formulated as follows:

$$min \sum_{i \in F} h_i y_i + \sum_{i \in F} \sum_{j \in D} Cost_{i,j} x_{i,j} \tag{14}$$

s.t.

$$\sum\nolimits_{i \in F} x_{i,j} = 1, \ \forall j \in D \tag{15}$$

$$\sum\nolimits_{j \in D} d_j x_{i,j} \leq c_i y_i, \ \forall i \in F \tag{16}$$

$$x_{i,j}, y_i \in \{0, 1\}, \forall i \in F, \forall j \in D \tag{17}$$

where y_i denotes whether facility i is open, $x_{i,j}$ indicates whether demand d_j is fulfilled by facility i, and $Cost_{i,j}$ represents the cost of serving demand j from facility i. Furthermore, c_i corresponds to the capacity of facility i.

Theorem 1. The MESS-HL problem is $\mathcal{NP}$-hard.

Proof. We make the following assumptions to construct a simplified MESS-HL:

- We focus solely on latency and ignore other two parts in Eq. (8).
- There is only one type of resource, C in the problem.
- The total available resources across all edge servers exceed the total resource demand of all submitted edge services from users.
- Each edge service begins execution immediately upon arrival.
- All users are stationary during their interaction with the edge environment.

With these assumptions, we can obtain a response time matrix $\left[\mathcal{R}_{\rangle,|}\right]_{n,m}$, where n and m represent the number of edge services and edge servers, respectively. Let the resources of all edge servers form the set $\mathbb{C}$. As there is no cost to run an edge server in the MESS-HL Problem, the first part of the CFLP objective, $\sum_{i \in F} h_i y_i$ in Eq. (14) can be neglected. Under these conditions, the objective in Eq. (14) can be mapped to the objective in Eq. (8), the constraint in Eq. (15) can be mapped to the constraint in Eq. (10), the constraint in Eq. (16) can be mapped to the constraint in Eq. (11), and

the constraint in Eq. (17) can be mapped to the constraints in Eqs. (12) and (13). Thus, given a CFLP instance $(Cost, D, C, F)$, the corresponding simplified MESS-HL problem instance $(\mathcal{R}, \mathbb{S}, \mathbb{C}, \mathbb{ES})$ can be constructed. Since CFLP is $\mathcal{NP}$-hard, the simplified MESS-HL problem, through the mapping, is also $\mathcal{NP}$-hard. Therefore, the original MESS-HL problem is more complex than the CFLP, which has proven its $\mathcal{NP}$-hard.

4.3 Approximately Optimum Algorithm for MESS-HL Problem

The pseudo-code in Algorithm 1 presents the overall structure of MHLSA. Compared to the traditional SA, our approach incorporates heuristic information during the initialization and perturbation phases. The details are elaborated as follows:

Algorithm 1. MHLSA

Input: A set of edge users $\mathbb{U}$ and a set of edge servers $\mathbb{ES}$.

Output: An approximately optimum solution Θ.

Set the parameters: Temperature T_0, cooling rate c, maximum iterations N_{iter}.
Initialize: Generate the initial solution Ω_{ini} using heuristic information.
Calculate the energy $f(\Theta)$.
For $i = 1$ to N_{iter} do
  Perturbation:
  Get the $Top-K$ load edge servers ES_K^{load} based on the current solution Θ.
  For $j = 1$ to K do
  Calculate the loads of servers ES_{Pr} which are within P-hops from es_j.
  Select servers with lighter loads from ES_{Pr}, represented by ES_{Pr}^L.
  Determine edge services S_{re} on es_j to reschedule based on probability p_e.
  Reschedule (S_{re}, ES_{Pr}^L)
  End for
  Get a new solution Ω_j, and Calculate the energy $f(\Omega_j)$.
  $\Delta f \leftarrow f(\Omega_j) - f(\Theta)$.
  if $\Delta f < 0$ then
    Set $\Theta \leftarrow \Omega_j$.
  else
    Set $\Theta \leftarrow \Omega_j$ with probability $\exp\left(-\frac{\Delta f}{T}\right)$.
  End if
  Update temperature: $T \leftarrow c \cdot T$.
End for

Initialization of MHLSA. To generate the initial solution Ω_{ini}, we first classify each heterogeneous request type. Then, leveraging user mobility trajectories and server load status, we assign each edge service to an appropriate execution server. Specifically, if s_i is UGD, we obtain the servers covering s_i's submission position, denoted as ES_{cov}^{sub}. Then, to balance response latency and system loads, we expand this set to include servers within a short-hop distance, forming ES_{sub}^{sh}. Next, we choose a subset $ES_{sub}^{sh\prime}$ from ES_{sub}^{sh}

that have sufficient resources to execute s_i, according to Eq. (1). If $ES_{sub}^{sh/}$ is empty, we assign s_i to the lowest load ES_{load}^{min} in ES_{sub}^{sh}. Otherwise, we randomly select a server ES_i^{exe} from $ES_{sub}^{sh/}$ to execute s_i. Similarly, if the request is *ULD*, we obtain the set of servers covering the request receipt position, ES_{cov}^{rec}, and apply the same strategy. For *UED*, we repeat the above operations for both ES_{cov}^{sub} and ES_{cov}^{rec}. After all edge services in $\mathbb{S}$ have selected their execution servers, we generate the initial solution Ω_{ini}.

Perturbation of MHLSA. To improve load balancing while minimizing the impact on latency and energy consumption, we first identify the top K load servers ES_K^{load}. Then, for each server es_j in ES_K^{load}, we calculate the loads of servers within P-hop, denoted by the set ES_{Pr}. We select servers with lighter loads from ES_{Pr}, represented by ES_{Pr}^L. Next, we choose a set of edge services S_{re} on es_j to reschedule based on the probability $p_e^{\omega_i} = \frac{1}{t_i^{exe}}$. This Selection enhances load balancing without significantly increasing delay or resource, since edge services with shorter execution time incur less delay and resource consumption during the transfer process. Each $s_i \in S_{re}$ is rescheduled to an appropriate execution server, using a heuristic-based operation, *Reschedule*.

The first step of *Reschedule* is to filter from ES_{Pr}^L a subset of servers with sufficient resources to execute S_{re}, denoted as ES_{suf}^L based on Eq. (1). If ES_{suf}^L is empty, the server with the lowest load in ES_{Pr}^L, denoted as ES_{min}^L, is selected as the new scheduling server ES_{re}. Otherwise, ES_{re} is determined based on the request type of s_i and the mobility trajectories of the corresponding edge user u_i. Specifically, for UGD requests, we identify the subset ES_{cov}^{sub} that covers the location where u_i submits s_i via coverage constraint, and select from ES_{suf}^L the server ES_{min}^{hop} with the shortest hop distance to ES_{cov}^{sub} as ES_{re}. A similar procedure is applied to ULD requests, but with focus on the set ES_{cov}^{rev}, which covers the location where u_i receives the response of s_i. For UED requests, the union ES_{cov}^{union} of ES_{cov}^{sub} and ES_{cov}^{rev} is considered. In all cases, if multiple servers satisfy the shortest hop criterion, the one with the lowest load is chosen as ES_{re}.

5 Experiments

5.1 Datasets and Experimental Setup

We conduct extensive experiments on two datasets widely used in edge computing:

- *Shanghai Telecom Dataset*[1]: It contains 3,233 base stations within Shanghai, China. It has been utilized as edge servers in numerous existing studies [13]. In our study, we select 100 servers located in the central area of the Shanghai municipality.
- *EUA Dataset*[2]: It comprises 125 base stations located in the Melbourne CBD area, Australia. It has been extensively used as edge serves such as [14].

Additionally, we generate users of varying numbers and simulate their mobility trajectories by referring to the *Shanghai Qiangsheng Taxi GPS trace dataset*[3].

[1] http://www.sguangwang.com/TelecomDataset.html.
[2] https://github.com/PhuLai/eua-dataset.
[3] https://sodachallenges.com/datasets/taxi-gps/

All experiments are conducted on a workstation equipped with two NVIDIA GeForce 1080Ti GPUs and an Intel Xeon Gold 6132 CPU running at 2.60 GHz.

5.2 Competing Approaches and Evaluation Metrics

To assess the performance of *MHLSA*, we compare it with eight existing approaches:

- ***Random***: It randomly selects a server to execute the user's edge service.

Table 1. Experimental results on SHT datasets with different edge services scales

Method	1000 Edge Services			1500 Edge Services			2000 Edge Services		
	ART	TCC	LB	ART	TCC	LB	ART	TCC	LB
Random	44.5	23130	0.67	47.9	35641	0.81	48.2	49625	1.12
Greedy	43.6	23945	0.07	54.1	36912	5.61	64.5	49137	25.08
ECFA	35.4	14961	0.06	37.6	21932	0.17	40.0	29462	1.17
CGWO	41.1	21468	0.89	45.0	35887	0.07	45.1	48645	0.59
AMO	44.0	24007	0.20	48.0	36505	0.49	48.2	48386	0.76
Sheu	43.4	23091	0.25	47.1	36204	0.61	46.0	47794	0.88
MHLSA-LB	31.9	12041	**0.01**	35.3	19054	**0.01**	36.5	25889	**0.01**
MHLSA-CC	**25.6**	**4256**	0.23	**28.8**	**6478**	0.15	**30.3**	**8362**	0.27
MHLSA	29.5	8580	0.20	31.6	12758	**0.01**	33.1	16936	**0.01**

- ***Greedy***: It selects the server with the most available resources.
- ***ECFA*** [15]: It introduces a novel probability-based mapping method and an efficient position update strategy. It selects servers by considering both latency and resource.
- ***CGWO*** [16]: It incorporates a new position update strategy for generating the next generation. It greedily selects servers where edge services can start executing earlier.
- ***AMO*** [17]: It proposes a new bio-inspired optimization approach called Ant Mating Optimization. It uses the uniform distribution method to generate initial solutions.
- ***Sheu*** [18]: It incorporates optimization-direction operations for generating a new solution. It prefers servers with lighter loads or faster executing rate.
- ***MHLSA-CC***: It is a variation of *MHLSA* that prioritizes selecting servers only from those covering the user's submission or receiving position.
- ***MHLSA-LB***: It is a variation of *MHLSA* that prioritizes selecting the server within more hops from the servers covering the user's submission or receiving position.

5.3 Experimental Results and Analyses

Comparisons of Edge Service Scheduling. Table 1 and Table 2 provide a comprehensive evaluation of our proposed *MHLSA* method against eight competing approaches

across different edge service scales on two real-world datasets. The results demonstrate that *MHLSA* and its two variants consistently achieve the best performance across all three metrics, though under varying edge servers and edge service scales.

Compared with the two baselines (*Random* and *Greedy*), *MHLSA* exhibits significantly superior performance across all three metrics, demonstrating the effectiveness of heuristic-driven scheduling over naive or greedy selection strategies. More importantly, taking *SHT 2000 ES dataset* as an example, *MHLSA* consistently outperforms existing state-of-the-art (SOTA) approaches. Firstly, *ECFA*, despite considering delay and resource, it neglects server loads, leading to a 99.1% higher *LB* than *MHLSA*. Secondly,

Table 2. Experimental results on EUA datasets with different edge services scales

Method	1000 Edge Services			1500 Edge Services			2000 Edge Services		
	ART	TCC	LB	ART	TCC	LB	ART	TCC	LB
Random	50.8	32831	0.35	53.1	51447	0.79	58.1	69789	1.12
Greedy	40.5	32397	0.03	42.7	50277	0.28	51.4	69644	8.79
ECFA	37.6	16826	0.06	40.3	38608	0.05	49.2	43453	2.87
CGWO	48.8	31593	0.86	48.8	50617	0.09	54.8	69914	0.72
AMO	50.8	33391	0.29	52.3	50950	0.44	58.2	70226	0.71
Sheu	49.5	32020	0.31	51.3	50779	0.73	54.7	68152	0.74
MHLSA-LB	34.8	13983	**0.01**	36.8	23953	**0.01**	41.7	32904	**0.01**
MHLSA-CC	**29.9**	**6102**	0.18	**31.5**	**10074**	0.14	**35.6**	**14303**	0.40
MHLSA	32.6	10449	0.20	33.6	16763	0.02	38.4	23035	0.02

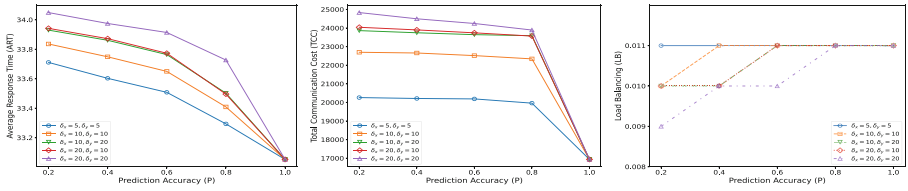
AMO, which employs uniform initialization without accounting for service and server heterogeneity, exhibits a 31.3% higher *ART*, a 65.0% higher *TCC*, and a 98.7% higher *LB*. Furthermore, *Sheu*, it begins with a random initialization, which inherently limits its ability to converge to an optimum solution. Consequently, Sheu underperforms MHLSA by 28.0% in ART, 64.6% in TCC, and 98.9% in LB. Lastly, compared with the most recent SOTA method, CGWO, MHLSA achieves significant improvements of 26.6% in *ART*, 65.0% in *TCC*, and 98.7% in *LB*. Overall, the consistent and substantial performance gains across multiple datasets demonstrate the robustness, generalizability, and applicability of *MHLSA* in real-world scenarios.

5.4 Performance Impact of Trajectory Prediction Deviations

Real-world scenarios inevitably involve trajectory prediction deviations. To better assess *MHLSA* under such conditions, we conduct an experiment analyzing when trajectory predictions deviate on the *2000 Edge Services SHT dataset* under different values of P, σ_x, and σ_y in Eq. (2). The results are shown in Fig. 1. The horizontal axis represents the probability of accurate predictions, where $P = 1$ indicates no deviations. The vertical

axis in each subplot represents the performance of three metrics: *ART*, *TCC*, and *LB,* respectively. Each curve corresponds to a different level of position offsets δ.

As shown in Fig. 1, when the prediction accuracy *P* decreases and the position offset δ increases, both the *ART* and *TCC* exhibit a growing trend. However, *ART* remains relatively stable, even under the worst-case scenario, it increases by only about 1 unit. This is due to *MHLSA*'s consideration of server loads, reducing the waiting latency, which highlights *MHLSA*'s ability to effectively mitigate the impact of prediction deviations on latency. Although *TCC* shows a more significant increase, *MHLSA* still outperforms existing methods even assuming that they have the perfect trajectory prediction, underscoring the robustness of our approach. Regarding *LB*, as shown in Fig. 1(c), it declines slightly as *P* decreases and δ increases. This is because the multi-objective.



(a) Impact on ART (b) Impact on TCC (c) Impact on LB

Fig. 1. The impact of trajectory prediction deviation on MHLSA

optimization design of our energy function, the relatively large fluctuation in the *TCC* drives the scheduling algorithm to prioritize improvements in *ART* and *LB*. Given that *ART* exhibits minimal variation, the algorithm may prefer solutions with decreased *LB* as the final scheduling outcome. Overall, *MHLSA* consistently maintains superior performance across all metrics despite trajectory prediction deviations.

6 Conclusion

In this paper, we introduce MESS-HL, a novel edge service scheduling problem that incorporates user mobility, request heterogeneity, and load balancing into consideration compared with conventional service scheduling problems. We formulate MESS-HL as a multi-objective optimization problem in MEC and prove its $\mathcal{NP}$-hardness. To address this challenge, we propose MHLSA, a heuristic simulated annealing-based algorithm designed to minimize average response time, communication cost, and maintain server load balancing. Experimental results demonstrate that MHLSA outperforms existing methods, achieving superior scheduling efficiency, and maintaining robust performance under trajectory prediction deviations. In future work, we aim to extend our approach to handle workflow-based scheduling with complex task dependencies.

Acknowledgments. This work was supported by National Natural Science Foundation of China (No. 62272290, 62172088).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Lu, S., Wu, J., Lu, P., Wang, N., Liu, H., Fang, J.: Qos-aware on-line service provisioning and updating in cost-efficient multi-tenant mobile edge computing. IEEE Trans. Serv. Comput. **17**(1), 113–126 (2024)
2. Chu, S., Gao, C., Xu, M., Ye, K., Xiao, Z., Xu, C.: Efficient multi-task computation offloading game for mobile edge computing. IEEE Trans. Serv. Comput. **17**(1), 30–46 (2024)
3. Hu, Z., Niu, J., Ren, T., Guizani, M.: Achieving fast environment adaptation of DRL-based computation offloading in mobile edge computing. IEEE Trans. Mob. Comput. **23**(5), 6347–6362 (2024)
4. Yang, Y., Wang, S.: EdgeOPT: a competitive algorithm for online parallel task scheduling with latency guarantee in mobile edge computing. IEEE Trans. Commun. **72**(11), 7077–7092 (2024)
5. Tong, Z., Ye, F., Mei, J., Liu, B., Li, K.: A novel task offloading algorithm based on an integrated trust mechanism in mobile edge computing. J. Parallel Distributed Comput. **169**, 185–198 (2022)
6. Cang, Y., et al.: Joint user scheduling and computing resource allocation optimization in asynchronous mobile edge computing networks. IEEE Trans. Commun. **72**(6), 3378–3392 (2024)
7. Lou, J., Tang, Z., Zhang, S., Jia, W., Zhao, W., Li, J.: Cost-effective scheduling for dependent tasks with tight deadline constraints in mobile edge computing. IEEE Trans. Mob. Comput. **22**(10), 5829–5845 (2023)
8. Ma, X., Zhou, A., Zhang, S., Li, Q., Liu, A.X., Wang, S.: Dynamic task scheduling in cloud-assisted mobile edge computing. IEEE Trans. Mob. Comput. **22**(4), 2116–2130 (2023)
9. Jiang, F., Peng, Y., Wang, K., Dong, L., Yang, K.: MARS: A DRL-based multi-task resource scheduling framework for UAV with IRS-assisted mobile edge computing system. IEEE Trans. Cloud Comput. **11**(4), 3700–3712 (2023)
10. Sharif, Z., Jung, L.T., Razzak, I., Alazab, M.: Adaptive and priority-based resource allocation for efficient resources utilization in mobile edge computing. IEEE Internet Things J. **10**(4), 3079–3093 (2023)
11. Dong, X., Di, Z., Wang, L., Yao, Q., Li, G., Shen, Y.: Load balancing of double queues and utility-workload tradeoff in heterogeneous mobile edge computing. IEEE Trans. Wirel. Commun. **22**(7), 4313–4326 (2023)
12. Chai, Y., Sapp, B., Bansal, M., Anguelov, D.: Multipath: multiple probabilistic anchor trajectory hypotheses for behavior prediction. In: Proceedings of the Conference on Robot Learning, pp. 86–99 (2020)
13. Li, Y., Zhou, A., Ma, X., Wang, S.: Profit-aware edge server placement. IEEE Internet Things J. **9**(1), 55–67 (2022)
14. He, Q., et al.: A game-theoretical approach for user allocation in edge computing environment. IEEE Trans. Parallel Distributed Syst. **31**(3), 515–529 (2020)
15. Yin, L., Sun, J., Zhou, J., Gu, Z., Li, K.: ECFA: an efficient convergent firefly algorithm for solving task scheduling problems in cloud-edge computing. IEEE Trans. Serv. Comput. **16**(5), 3280–3293 (2023)
16. Lian, Z., Shu, J., Zhang, Y., Sun, J.: Convergent grey wolf optimizer metaheuristics for scheduling crowdsourcing applications in mobile edge computing. IEEE Internet Things J. **11**(2), 1866–1879 (2024)
17. Ghanavati, S., Abawajy, J.H., Izadi, D.: An energy aware task scheduling model using ant-mating optimization in fog computing environment. IEEE Trans. Serv. Comput.Comput. **15**(4), 2007–2017 (2022)
18. Liu, H., Li, Y., Wang, S.: Request scheduling combined with load balancing in mobile-edge computing. IEEE Internet Things J. **9**(21), 20841–20852 (2022)