Local Ratio based Real-time Job Offloading and Resource Allocation in Mobile Edge Computing

Chuanchao Gao, Arvind Easwaran
gaoc0008@e.ntu.edu.sg,arvinde@ntu.edu.sg
College of Computing and Data Science
Energy Research Institute @ NTU, Interdisciplinary Graduate Programme
Nanyang Technological University
Singapore

Abstract

Mobile Edge Computing (MEC) has emerged as a promising paradigm enabling vehicles to handle computation-intensive and time-sensitive applications for intelligent transportation. Due to the limited resources in MEC, effective resource management is crucial for improving system performance. While existing studies mostly focus on the job offloading problem and assume that job resource demands are fixed and given apriori, the joint consideration of job offloading (selecting the edge server for each job) and resource allocation (determining the bandwidth and computation resources for offloading and processing) remains underexplored. This paper addresses the joint problem for deadline-constrained jobs in MEC with both communication and computation resource constraints, aiming to maximize the total utility gained from jobs. To tackle this problem, we propose an approximation algorithm, `IDAssign`, with an approximation bound of $\frac{1}{6}$, and experimentally evaluate the performance of `IDAssign` by comparing it to state-of-the-art heuristics using a real-world taxi trace and object detection applications.

CCS Concepts

• **Theory of computation → Scheduling algorithms**.

Keywords

Mobile Edge Computing, Deadline-constrained Job Offloading and Resource Allocation, Approximation Algorithm

ACM Reference Format:
Chuanchao Gao, Arvind Easwaran. 2025. Local Ratio based Real-time Job Offloading and Resource Allocation in Mobile Edge Computing . In *The 4th International Workshop on Real-time and IntelliGent Edge computing (RAGE '25), May 6–9, 2025, Irvine, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3722567.3727843

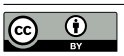
1 Introduction

Internet of Vehicles (IoV), a derivative technology of the Internet of Things (IoT), plays a pivotal role in realizing intelligent transportation systems by connecting vehicles and infrastructure to enable real-time service support.IoV utilizes advanced wireless communication technologies such as ultra-reliable low-latency communication of 5G to facilitate intelligent applications like smart traffic control, autonomous driving, and real-time navigation. These applications are generally computation-intensive and highly sensitive to latency, posing significant challenges for onboard vehicular systems with limited computing capabilities.

Mobile Edge Computing (MEC) has emerged as a promising paradigm enabling vehicles to support computation-intensive and time-critical applications in IoV. In MEC, vehicles can offload jobs to roadside edge servers (ES) that provide communication (bandwidth) and computation resources for instant processing. Deploying ES close to vehicles significantly reduces communication latency compared to traditional cloud computing, enabling prompt responses to vehicular requests. However, the bandwidth and computation resources of ES are limited, necessitating efficient strategies for *job offloading* (selecting the ES for each job) and *resource allocation* (determining the bandwidth and computation resource allocations for offloading and processing) in MEC.

Jointly optimizing job offloading and resource allocation enables MEC systems to dynamically allocate resources based on availability, improving overall resource utilization. For instance, an MEC system with abundant computational resources but limited bandwidth can allocate more computing power while reducing bandwidth per job, thereby accommodating more jobs while maintaining the required quality of service. While numerous studies have investigated the Deadline-constrained job Offloading Problem (DOP) in MEC under the assumption of fixed resource allocation per job [7, 8, 11, 18, 19], the joint optimization of job offloading and resource allocation [3, 5, 6, 9, 13, 16, 17], referred to as the Deadline-constrained job Offloading and resource Allocation Problem (DOAP), remains underexplored.

DOP reduces to the two-dimensional Generalized Assignment Problem (2DGAP) when considering both bandwidth and computational resource contention (e.g., [8]) and to the Generalized Assignment Problem (GAP, known to be NP-hard [4]) when only computational constraints are considered (e.g., [7, 8, 11, 18, 19]). DOAP is a more generalized version of DOP that additionally considers resource allocation, making it inherently NP-hard as well. Obtaining an exact solution for DOP or DOAP (e.g., [3, 16]) requires exponential time. Consequently, most existing studies focus on polynomial-time heuristic algorithms. Among these, approximation algorithms stand out as a special class of heuristics that provide worst-case performance guarantees while maintaining low computational complexity. Prior studies that propose approximation algorithms (e.g., [7, 8, 11, 17–19]) primarily focus on the DOP variant, formulating their problems as GAP by considering only computational constraints with fixed resource allocations. Although



RAGE '25, Irvine, CA, USA

ACM ISBN /2025/05
https://doi.org/10.1145/3722567.3727843

Wang *et al.* [17] addressed a DOAP and proposed an approximation algorithm with a probabilistic guarantee, their approach predefines resource allocation by manually specifying the offloading and processing times for each job. As a result, their approximation algorithm effectively applies only to DOP rather than the more general DOAP. To the best of our knowledge, no approximation algorithm with constant approximation ratio has been proposed for DOAP.

DOAP constraints can be categorized into two types. The first type ensures that allocated resources and offloading decisions satisfy job-specific requirements, such as deadlines and ES accessibility. The second type enforces system-wide resource constraints, ensuring that the total allocated resources at each ES do not exceed its capacity. We define an *assignment instance* of a job as a combination of (i) an offloading ES and (ii) allocated bandwidth and computational resources. Given an assignment instance, we can immediately verify whether it satisfies all first-type constraints and determine its contribution to the objective function. By focusing only on feasible assignment instances (i.e., those satisfying the first-type constraints), we only need to focus on the system-wide resource constraints in DOAP. Differences in the mathematical formulation of DOAP across various MEC applications primarily stem from variations in the first-type constraints and optimization objectives. Modeling DOAP based on assignment instances provides a uniform framework applicable to various scenarios.

In this paper, we study a DOAP in MEC with both bandwidth and computation resource constraints under the IoV scenario, aiming to maximize the total utility gained from offloading jobs; we refer to this specific DOAP as problem **P**. We first formulate **P** as an Integer Linear Programming (ILP) problem by enumerating all possible assignment instances. Then, we introduce an approximation algorithm for **P**, denoted as `IDAssign`, based on a local ratio framework [1]. We prove that `IDAssign` is a $\frac{1}{6}$-approximation algorithm for **P**, i.e., the total utility obtained by `IDAssign` is no less than $\frac{1}{6}$ of the optimal utility of **P**. Since **P** is formulated with the uniform framework, the approximation ratio of `IDAssign` is applicable across various DOAP applications. Finally, we evaluate `IDAssign` using profiled data from object detection applications and real-world taxi traces [15]. While providing a theoretical guarantee, `IDAssign` demonstrates performance comparable to all evaluated baseline algorithms in the literature [3, 5, 6, 9, 13].

2 System Model and Problem Formulation

In this section, we first introduce the MEC architecture, followed by a detailed formulation of DOAP in the presented MEC framework using its generalized form.

2.1 System Architecture

As illustrated in Fig.1(a), we consider an MEC comprising a set of roadside ES providing services to moving vehicles. The notations used in the paper are summarized in Table1. Let $\mathcal{M}$ denote the set of M ES, with $m_k \in \mathcal{M}$ representing the k-th ES. Each ES includes an access point (providing bandwidth for job offloading) and a server (providing computation resource for job processing). The computation resource capacity of each ES is measured in computing units, which depend on the specific server hardware configuration. For example, using NVIDIA's multi-instance GPU technology [12],

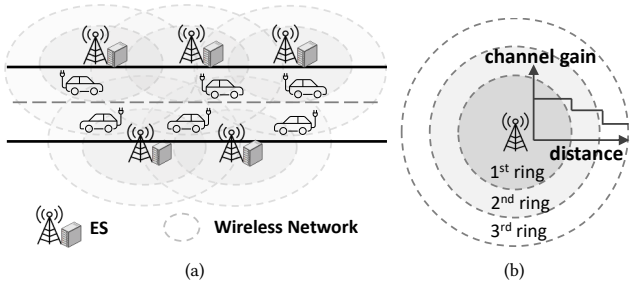


Figure 1: (a) An example of mobile edge computing for IoV; (b) Partition the communication range of a wireless network into three network rings;

a GPU-enabled ES can partition its computation resources into multiple computing units, each comprising a predefined amount of memory, cache, and compute cores. Let C_k be the number of computing units available at m_k, and let $C = \max\{C_k \mid m_k \in \mathcal{M}\}$.

We consider a 5G network in MEC, where the Orthogonal Frequency Division Multiple Access (OFDMA) is used as the network access method for vehicles. In an OFDMA-enabled network, the smallest bandwidth allocation unit available to users is referred to as a resource unit or bandwidth unit (BU) [20]. Each user may be assigned one or more BU for data offloading. The size (or bandwidth) of a BU is measured in megahertz (MHz) and is determined by the specific configuration of the 5G network. Let B_k denote the total number of BUs available at m_k, with β_k representing the size of each BU. Define $B = \max\{B_k \mid m_k \in \mathcal{M}\}$. The wireless network of each m_k has a limited effective communication range, and a vehicle can only offload a job to m_k when it is within this range. Since the channel gain in wireless networks diminishes as the distance between a vehicle and an ES increases [21], we divide each ES's effective range into several *network rings* such that the channel gain remains unchanged (from a practical viewpoint) within each network ring (Fig. 1(b)). Let m_{kr} be the r-th network ring of m_k.

Let $\mathcal{N}$ represent the set of N jobs to be processed. The j-th job, $n_j \in \mathcal{N}$, has an input data size θ_j (measured in megabytes, MB) and a deadline Δ_j (measured in seconds). When a job n_j is generated, its vehicle's geographical location determines the set of network rings covering the vehicle and the corresponding channel gains. Let $\mathcal{R}_j$ be the set of network rings covering n_j's vehicle at the time of its release, where $|\mathcal{R}_j| \leq M$. Based on the vehicle's moving speed and direction, the duration t^v_{jkr} for which the vehicle remains within the coverage of network ring $m_{kr} \in \mathcal{R}_j$ can be estimated. If m_{kr} is selected for offloading n_j, the processing of n_j must be completed before n_j's vehicle leaves the coverage of m_{kr}. Furthermore, if n_j is processed on m_k using c computing units, let t^p_{jkc} represent the processing time for n_j.

Definition 1 (Assignment Instance). *An assignment instance of job n_j is defined as $\ell \triangleq \langle m_{\ell r}, b_\ell, c_\ell \rangle$, which represents the scenario where n_j is offloaded to ES m_ℓ when n_j's vehicle is covered by network ring $m_{\ell r} \in \mathcal{R}_j$ and n_j is allocated b_ℓ bandwidth units and c_ℓ computing units.*

Suppose job n_j is assigned based on assignment instance $\ell \triangleq \langle m_{\ell r}, b_\ell, c_\ell \rangle$ (Definition 1). Based on Shannon's theorem [14], the

data offloading rate is given by $\eta_j = b_\ell \cdot \beta_\ell \log_2(1 + p_j \cdot h_{\ell r}/\sigma^2)$, where $b_\ell \cdot \beta_\ell$ is the allocated bandwidth, p_j is the offloading power of n_j's vehicle, $h_{\ell r}$ is the channel gain in network ring $m_{\ell r}$, and σ is the noise spectral density. The offloading time t_j^o is given by $t_j^o = \theta_j/\eta_j$. Given computation resource allocation c_ℓ, the processing time of n_j is $t_j^p = d_{j\ell c_\ell}^p$. Given that the data size of results is typically much smaller than job inputs, and the bandwidth capacity of the downlink is generally larger than that of the uplink in 5G networks, we disregard the result return time, as done in [10]. Let $t_j = t_j^o + t_j^p$ denote the completion time of n_j. Each offloaded job n_j offers a utility u_j depending on t_j, which is defined as follows.

$$u_j(t_j) = \begin{cases} U_j & \text{if } t_j \le \Delta_j \\ U_j \cdot \Psi_j(t_j) & \text{if } \Delta_j < t_j \le \gamma_j \Delta_j \\ 0 & \text{otherwise} \end{cases}. \quad (1)$$

$\gamma_j \ge 1$ represents the tolerance factor for deadline violations of job n_j. The full utility offered by n_j when completed before its deadline Δ_j is denoted by U_j. Let $u(\ell)$ be the utility obtained from n_j when it is assigned according to ℓ, as defined in Eq. (1). Ψ_j is a utility-reduction function for deadline violations that maps to the range $[0, 1]$. This function Ψ_j can take various forms (e.g., linear, nonlinear, or step functions), ensuring that $u_j(t_j)$ is flexible enough to model diverse practical scenarios. For example, $u_j(t_j)$ may represent a function related to energy consumption, completion time, or a combination of both factors.

In IoV, safety-critical jobs (e.g., object detection for autonomous driving) typically have hard deadlines that do not permit any deadline violations. Conversely, non-safety-critical jobs (e.g., image or video rendering) have soft deadlines, tolerating some deadline violations at the cost of reduced user experience when deadlines are missed. Therefore, this work considers jobs with both hard and soft deadlines. If n_j has a hard deadline, then $\gamma_j = 1$. If n_j has a soft deadline, $\gamma_j > 1$, and the utility derived from n_j decreases from U_j to 0 according to Ψ_j after its deadline is exceeded.

2.2 Problem Formulation

In this paper, we aim to develop a job offloading and resource allocation strategy that maximizes the utility gained from jobs while satisfying system resource constraints, ES access constraints, and job deadline constraints (Problem **P**). Next, we formulate **P** in its generalized form.

Definition 2 (Feasible Assignment Instance). *An assignment instance $\ell \triangleq \langle m_{\ell r}, b_\ell, c_\ell \rangle$ of job n_j is feasible if ℓ satisfies (i) the ES access constraint (i.e., $m_{\ell r} \in \mathcal{R}_j$), (ii) the deadline constraint (i.e., $t_j \le \min\{\gamma_j \Delta_j, t_{j\ell r}^o\}$), and (iii) the resource constraint (i.e., $b_\ell \le B_\ell, c_\ell \le C_\ell$).*

Given the finite bandwidth and computing resources of each ES, for each job $n_j \in \mathcal{N}$, we enumerate all its potential assignment instances by considering each $m_{kr} \in \mathcal{R}_j$ and all possible resource allocations. We then restrict our focus to feasible assignment instances (as defined in Definition 2). Henceforth, *unless explicitly stated otherwise, any reference to an assignment instance refers to a feasible assignment instance.* Let $\mathcal{L}$ denote the set of all feasible assignment instances for all jobs, $\mathcal{L}_j \subset \mathcal{L}$ the set of assignment instances for job n_j, and $\mathcal{E}_k \subset \mathcal{L}$ the set of assignment instances associated with ES m_k. For a given assignment instance ℓ, let $\mathcal{L}(\ell)$ represent the set of all assignment instances corresponding to the same job as ℓ, and let $\mathcal{E}(\ell)$ denote the set of all assignment instances associated with the same ES as ℓ. Since each job has at most M accessible network rings, B bandwidth allocation choices, and C computation resource options, the total number of assignment instances in $\mathcal{L}$ is at most $NMBC$. For any set of assignment instances $\mathcal{S}$, let $u(\mathcal{S}) = \sum_{\ell \in \mathcal{S}} u(\ell)$. For each assignment instance $\ell \in \mathcal{L}$, let $x_\ell \in 0, 1$ be the selection variable of ℓ, where $x_\ell = 1$ if and only if ℓ is chosen in the solution. Then, we formulate **P** as follows.

Table 1: Notation (Main Parameters and Variables)

Symb.	Definition
$\mathcal{M}$	the set of M ESs; $m_k \in \mathcal{M}$ denotes an ES
C_k	the total computing units of ES m_k; $C_k \le C$
B_k	the total bandwidth units of ES m_k; $B_k \le B$
m_{kr}	r-th network ring of ES m_k
$\mathcal{N}$	the set of N jobs; $n_j \in \mathcal{N}$ denotes a job
$\mathcal{R}_j$	the set of accessible network rings of job n_j; $\lvert\mathcal{R}_j\rvert \le M$
Δ_j	deadline of job n_j
γ_j	tolerance factor for deadline violation of job n_j; $\gamma_j \ge 1$
U_j	utility gained from job n_j when completed before Δ_j
t_j	completion time of job n_j
ℓ	$\ell \triangleq \langle m_{\ell r}, b_\ell, c_\ell \rangle$ is an assignment instance of a job
$u(\ell)$	utility gained from job n_j when assigned based on ℓ
$\mathcal{L}$	the set of all assignment instances of all jobs
$\mathcal{L}_j$	the set of all assignment instances of job n_j
$\mathcal{E}_k$	the set of all assignment instances mapped to ES m_k
$\mathcal{L}(\ell)$	set of assignment instances related to the same job as ℓ
$\mathcal{E}(\ell)$	set of assignment instances mapped to the same ES as ℓ
$\mathcal{L}_L$	the set of all light assignment instances in $\mathcal{L}$
$\mathcal{L}_H$	the set of all heavy assignment instances in $\mathcal{L}$
x_ℓ	binary selection variable of assignment instance ℓ

$$\begin{aligned} & (\mathbf{P}) \ \max \textstyle\sum_{\ell \in \mathcal{L}} u(\ell) \cdot x_\ell && (2a) \\ \text{subject to:} \quad & \textstyle\sum_{\ell \in \mathcal{E}_k} b_\ell \cdot x_\ell \le B_k, \forall m_k \in \mathcal{M} && (2b) \\ & \textstyle\sum_{\ell \in \mathcal{E}_k} c_\ell \cdot x_\ell \le C_k, \forall m_k \in \mathcal{M} && (2c) \\ & \textstyle\sum_{\ell \in \mathcal{L}_j} x_\ell \le 1, \forall n_j \in \mathcal{J} && (2d) \\ & x_\ell \in \{0, 1\}, \forall \ell \in \mathcal{L} && (2e) \end{aligned}$$

Eqs. (2b) and (2c) are the bandwidth and computation resource constraints for each ES, respectively. Eq. (2d) ensures that at most one assignment instance is selected for each job. Notably, since set $\mathcal{L}$ consists of only feasible assignment instances, the deadline and ES access constraints have been implicitly satisfied.

3 Instance Dividing Assignment Algorithm

In this section, we present an approximation algorithm for **P**, named the Instance Dividing Assignment Algorithm (`IDAssign`), which leverages a local ratio framework combined with an instance-dividing technique. Later in this section, we formally prove that `IDAssign` achieves a $\frac{1}{6}$-approximation ratio for **P**. To the best of our knowledge, `IDAssign` is the first algorithm that provides a constant approximation ratio for DOAP.

The detailed steps of `IDAssign` are outlined in Algorithm 1. For each assignment instance ℓ, let $\tilde{b}_\ell = \frac{b_\ell}{B_\ell}$ and $\tilde{c}_\ell = \frac{c_\ell}{C_\ell}$ represent

Algorithm 1: Instance Dividing Assignment Algorithm

1 IDAssign($\mathbf{w^i}$, $\mathcal{L}$):
2 Remove all instances ℓ from $\mathcal{L}$ with $w^i(\ell) \le 0$;
3 **if** $\mathcal{L} = \emptyset$ **then return** $\mathcal{S}^i = \emptyset$;
4 **if** $\mathcal{L}_L \neq \emptyset$ **then** $\ell^i \leftarrow \arg\min_{\ell \in \mathcal{L}_L}(\max\{\tilde{b}_\ell, \tilde{c}_\ell\})$ **else** $\ell^i \leftarrow \arg\min_{\ell \in \mathcal{L}_H}(\max\{\tilde{b}_\ell, \tilde{c}_\ell\})$;
5 Decompose utility vector $\mathbf{w^i}$ into $\mathbf{w_1^i}$ and $\mathbf{w_2^i}$ (i.e., $\mathbf{w^i} = \mathbf{w_1^i} + \mathbf{w_2^i}$) such that for any $\ell \in \mathcal{L}$,

$$w_1^i(\ell) = w^i(\ell^i) \cdot \begin{cases} 1 & \text{if } \ell \in \mathcal{L}(\ell^i) \\ (\tilde{b}_\ell + \tilde{c}_\ell) & \text{if } \ell \in \mathcal{E}(\ell^i) \setminus \mathcal{L}(\ell^i) \\ 0 & \text{otherwise} \end{cases};$$

6 $\mathcal{S}^{i+1} \leftarrow$ IDAssign($\mathbf{w_2^i}$, $\mathcal{L}$);
7 **if** $\mathcal{S}^{i+1} \cup \{\ell^i\}$ *is feasible* **then return** $\mathcal{S}^i \leftarrow \mathcal{S}^{i+1} \cup \{\ell^i\}$ **else return** $\mathcal{S}^i \leftarrow \mathcal{S}^{i+1}$;

the normalized bandwidth and computation resource allocation, respectively. We classify an assignment instance ℓ as a *light instance* if $\tilde{b}_\ell \le \frac{1}{2}$ and $\tilde{c}_\ell \le \frac{1}{2}$, and as a *heavy instance* otherwise. Let $\mathcal{L}_L$ denote the set of all light instances in $\mathcal{L}$, and $\mathcal{L}_H$ the set of all heavy instances in $\mathcal{L}$. The algorithm is then recursively called with the initial invocation IDAssign($\mathbf{u}, \mathcal{L}$), where $\mathbf{u}$ is a vector containing the utility values $u(\ell)$ for all $\ell \in \mathcal{L}$. Since the utility of each assignment instance is updated at each recursive layer, we denote the updated utility of ℓ at the i-th recursive layer as $w^i(\ell)$, where $u(\ell)$ represents the initial utility of ℓ.

In each recursive layer i, the algorithm selects an instance ℓ^i with the smallest $\max\{\tilde{b}_{\ell^i}, \tilde{c}_{\ell^i}\}$, prioritizing instances in $\mathcal{L}_L$ over those in $\mathcal{L}_H$ (line 4). The utility value $w^i(\ell)$ for each assignment instance $\ell \in \mathcal{L}$ is then decomposed into two components, $w_1^i(\ell)$ and $w_2^i(\ell)$ (line 5), where $w_2^i(\ell)$ represents the marginal gain corresponding to the choice of ℓ^i. The marginal utility vector $\mathbf{w_2^i}$ becomes the utility vector $\mathbf{w^{i+1}}$ for the next recursive layer (line 6). Based on the solution $\mathcal{S}^{i+1}$ returned from layer $(i+1)$, ℓ^i is added to the solution if it satisfies the feasibility constraints (2b)–(2d). Next, we prove the approximation ratio of IDAssign for $\mathbf{P}$ as follows.

Theorem 1. IDAssign *is a* $\frac{1}{6}$*-approximation algorithm for* $\mathbf{P}$.

Proof. Suppose Q is an optimal solution of $\mathbf{P}$. Let the innermost recursive layer of IDAssign be layer I, and the outermost be layer 1. We use simple induction to prove this lemma by showing that $w^i(\mathcal{S}^i) \ge \frac{1}{6} w^i(Q)$ for $i = I, I-1, ..., 1$ (since $\mathbf{w^1} = \mathbf{u}$).

Base Case ($i = I$): When $i = I$, $w^i(\ell) \le 0, \forall \ell \in \mathcal{L}$. The returned $\mathcal{S}^I = \emptyset$, so $w^I(\mathcal{S}^I) = 0 \ge w^I(Q) \ge \frac{1}{6} w^I(Q)$.

Inductive Step ($i < I$): When $i < I$, suppose $w^{i+1}(\mathcal{S}^{i+1}) \ge \frac{1}{6} w^{i+1}(Q)$. Since $\mathbf{w_2^i} = \mathbf{w^{i+1}}$, $w_2^i(\mathcal{S}^{i+1}) \ge \frac{1}{6} w_2^i(Q)$. Based on the utility decomposition in line 5, $w_1^i(\ell^i) = w^i(\ell^i)$, so $w_2^i(\ell^i)$=0; thus,

$$w_2^i(\mathcal{S}^i) = w_2^i(\mathcal{S}^{i+1}) \ge \tfrac{1}{6} w_2^i(Q). \tag{3}$$

In line 7, if ℓ^i can be added to $\mathcal{S}^i$, $w_1^i(\mathcal{S}^i) \ge w^i(\ell^i)$. Otherwise, either one $\ell \in \mathcal{L}(\ell^i)$ already exists in $\mathcal{S}^{i+1}$ that stops ℓ^i from being added to $\mathcal{S}^i$ (constraint (2d)) or the remaining resource in ES m_{ℓ^i} is not enough to accommodate ℓ^i (constraints (2b) and (2c)). In the former case, $w_1^i(\mathcal{S}^i) \ge w^i(\ell^i)$ due to line 5. In the latter case, let $\mathcal{A}^i = \mathcal{S}^{i+1} \cap \mathcal{E}(\ell^i)$, representing the instances that have been selected in $\mathcal{S}^{i+1}$ and have resource contention with ℓ^i. We consider the following two situations for the latter case: $\ell^i \in \mathcal{L}_H$ and $\ell^i \in \mathcal{L}_L$. If $\ell^i \in \mathcal{L}_H$, as we only consider instances in $\mathcal{L}_H$ when $\mathcal{L}_L = \emptyset$, the solution $\mathcal{S}^{i+1}$ contains only heavy instances; in this situation, there exists at least one heavy instance $\ell \in \mathcal{S}^{i+1}$ that prevents ℓ^i being added to the solution due to resource contention. Based on line 5 and the definition of heavy instances, $w_1^i(\mathcal{S}^i) \ge \frac{1}{2} w^i(\ell^i)$. If $\ell^i \in \mathcal{L}_L$, either $\sum_{\ell \in \mathcal{A}^i} \tilde{b}_\ell \ge \frac{1}{2}$ or $\sum_{\ell \in \mathcal{A}^i} \tilde{c}_\ell \ge \frac{1}{2}$ (i.e., at least $\frac{1}{2}$ of m_{ℓ^i}'s bandwidth or computation resource are allocated to $\mathcal{S}^{i+1}$); thus, $w_1^i(\mathcal{S}^i) \ge \frac{1}{2} w^i(\ell^i)$ due to line 5. Therefore, if instance ℓ^i cannot be added to $\mathcal{S}^i$, we have $w_1^i(\mathcal{S}^i) \ge \frac{1}{2} w^i(\ell^i)$. Besides, $w_1^i(Q)$ is maximized when Q contains one $\ell \in \mathcal{L}(\ell^i) \setminus \mathcal{E}(\ell^i)$ and all resources of ES m_{ℓ^i} has been allocated to $\ell' \in Q \cap \mathcal{E}(\ell^i) \setminus \mathcal{L}(\ell^i)$; in this case, $w_1^i(Q) = w^i(\ell^i) + 2w^i(\ell^i) = 3w^i(\ell^i)$. Hence, we have

$$w_1^i(\mathcal{S}^i) \ge \frac{1}{6} w_1^i(Q). \tag{4}$$

Given Eq. (3), Eq. (4), and $\mathbf{w^i} = \mathbf{w_1^i} + \mathbf{w_2^i}$, we have $w^i(\mathcal{S}^i) \ge \frac{1}{6} w^i(Q)$. Based on simple induction, $w^i(\mathcal{S}^i) \ge \frac{1}{6} w^i(Q)$ for $i = I, ..., 1$. □

Time Complexity. Since $\mathcal{L}$ contains at most $NMBC$ assignment instances, IDAssign has at most $NMBC$ recursive layers. In line 5, at most $NMBC$ assignment instances are considered for utility decomposition. Thus, the time complexity of IDAssign is $O((NMBC)^2)$.

4 Numerical Evaluation

In this section, we evaluate the performance of IDAssign. This algorithm is compared against three existing algorithms from the literature, focusing on utility performance and runtime efficiency. All experiments were performed on a desktop PC equipped with an Intel(R) Core(TM) i7-14700KF CPU and 64 GB of RAM.

4.1 Simulation Setup

A real taxi GPS data trace [15], collected on April 1, 2018 in Shanghai by Shanghai Qiangsheng Taxi Company, is used as the vehicle traffic data. We select a city area of 1km×1km in Shanghai and extract all taxi trajectories passing through this area over a period of 15 minutes during the evening peak. We consider a MEC with 20 ES evenly distributed along the roads. Each ES is mounted at a height of 10m, and its network is divided into 2 rings with coverage ranges of 0 ∼ 100m and 100 ∼ 200m. For each ES m_k, β_k is set to 2MHz (the smallest BU type in WiFi-6), and B_k is set to either 20 or 40. The data offloading rate η_j for one BU is set to 1.65 MBps for ring 1 and 1.15 MBps for ring 2. Each ES is equipped with a GPU randomly sampled from 5 types of GPUs (TITAN XP, TITAN RTX, RTX3090, RTX4060Ti, and A100). We set C_k of each ES m_k as 25.

In this experiment, we synthesize jobsets with varying resource utilizations and jobset sizes. The bandwidth utilization ru_b of a jobset is defined as the ratio of the total bandwidth demand of all jobs in a jobset to the total bandwidth available in the MEC [5], and the computation resource utilization ru_c of a jobset is analogously defined. We consider two resource utilization sampling ranges: **low** ([0.6, 0.9]) and **high** ([1.2, 1.5]), forming four different range combinations. Jobset size N ranges from 200 to 400 in steps of 40.

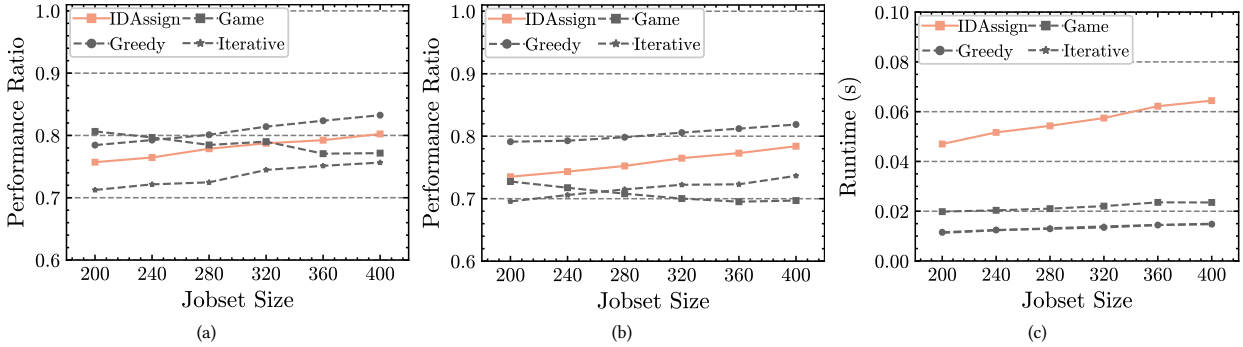


Figure 2: (a) average performance ratios of algorithms under the (low,low) range combination for different jobset sizes; (b) average performance ratios of algorithms under the (high,high) range combination for different jobset sizes; (c) average runtime of algorithms for different jobset sizes (the plots for `Greedy` and `Iterative` are overlapped);

For each jobset size, we sample 150 different (ru_b, ru_c) pairs from each range combination. In total, we synthesize 3600 jobsets.

Given a jobset size and the sampled (ru_b, ru_c), we synthesize a jobset as follows. The jobset release time is randomly sampled within the 15-minute period, and each job is randomly mapped to a taxi active at the release time to obtain the ring coverage $\mathcal{R}_j$. For each job n_j, we randomly sample its input θ_j from 45 images ranging from 0.15 to 0.63 MB, and its utility U_j from the range $[20, 60]$. Then, we randomly sample the application type of n_j from 9 GPU applications (resnet34/50/101, densenet121/169, and vgg11/13/16/19) for object detection. We use Stafford's Randfixedsum Algorithm [2] to distribute ru_b and ru_c to each individual job in the jobset in a uniformly random and unbiased manner, which is used to compute the job deadline. Next, we compute the maximum tolerable completion time $\gamma_j \Delta_j$ where we uniformly sample γ_j from the range $[1.8, 2.2]$ and define the function Ψ_j as a linear decreasing function when n_j has a soft deadline.

Baseline Algorithms. We compare `IDAssign` with three heuristic algorithms that are summarized from the literature on DOAP: `Greedy` [5], `Iterative` [3, 6, 9, 13], and `Game` [9]. Note that none of these baselines provide theoretical guarantees on the performance of their algorithms for problem **P**. `Greedy` defines $e_\ell = u(\ell)/(\frac{b_\ell}{B_\ell} \times \frac{c_\ell}{C_\ell})$ as the resource efficiency of $\ell \in \mathcal{L}$, and adds $\ell \in \mathcal{L}$ into the solution whenever feasible in a descending order of e_ℓ. `Iterative` decouples **P** into a job offloading subproblem and a resource allocation subproblem; then, solve these subproblems iteratively until convergence. `Game` defines **P** as a non-cooperative game, where each vehicle is a player; in each round, select the $\ell \in \mathcal{L}$ that contributes the largest increment on the total utility gained.

Metrics. We use Performance Ratio to measure the performance of all algorithms, defined as the ratio of the total utility obtained by an algorithm to the optimal utility for **P** (the optimal utility is obtained by solving **P** using the ILP solver Gurobi, with a timeout limit set to 10 minutes).

4.2 Result Discussion

We first evaluated the average performance ratios of the algorithms across various jobset sizes under the range combinations (low, low) and (high, high) (Figs. 2(a) and 2(b)). The combination (low, low) indicates that both ru_b and ru_c of the jobsets are sampled from

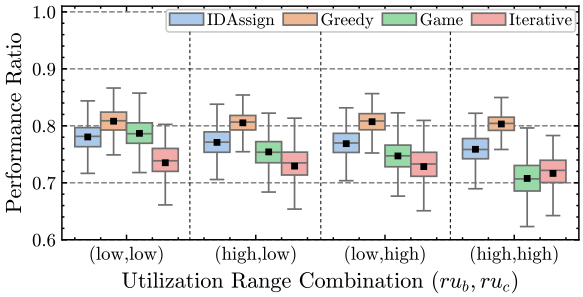


Figure 3: Performance ratios of algorithms under different resource utilization range combinations (ru_b, ru_c)

the **low** range. For both range combinations, `IDAssign` demonstrates improved performance as the jobset size increases, with better results observed under the (low, low) combination. This performance trend occurs because `IDAssign` prioritizes light instances (i.e., $\mathcal{L}_L$) over heavy instances (i.e., $\mathcal{L}_H$). When the jobset size is small or when the resource demand falls within the (high, high) range combination, the per-job resource demand increases, making heavy instances more favorable candidates in an optimal solution. However, due to the prioritization strategy in `IDAssign`, the utility values $w^i(\ell)$ of many heavy instances $\ell \in \mathcal{L}_H$ are frequently reduced to zero before being considered, thereby decreasing the likelihood of selecting heavy instances and ultimately diminishing the overall performance of `IDAssign`.

The average runtime of algorithms for different jobset sizes is illustrated in Fig. 2(c). While offering a theoretical guarantee and competitive practical performance, `IDAssign` achieves a runtime comparable to the baseline algorithms (`Greedy`, `Game`, and `Iterative`). *This makes* `IDAssign` *a preferable choice when both a theoretical performance guarantee and minimal computational overhead are required.* Furthermore, `IDAssign` exhibits nearly linear runtime scalability with respect to jobset size, confirming its practical suitability for handling large-scale jobsets.

The overall algorithm performance across all resource utilization range combinations (including (low,low), (low, high), (high, low) and (high,high)) in the MEC with 20 ES is presented in Fig. 3. `IDAssign` achieves an average performance ratio of 76.9%. Despite the *baseline algorithms lacking theoretical guarantees*, `IDAssign`

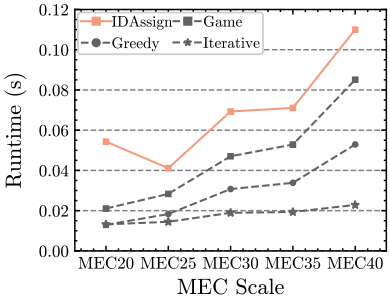


Figure 4: average runtime of algorithms for different MEC scales (with jobset size 280)

still demonstrates a practical performance comparable to `Greedy`, `Game`, and `Iterative`, underscoring its practical effectiveness. Remarkably, `IDAssign` surpasses its theoretical bound of $\frac{1}{6}$ by 60.2%, illustrating its ability to narrow the gap between theoretical guarantees and optimal solutions.

To evaluate the runtime scalability of `IDAssign` across different MEC scales, we consider four additional MEC with 25, 30, 35, and 40 ES, referred to as MEC25, MEC30, MEC35, and MEC40, respectively. The base configuration with 20 ES is denoted as MEC20. For each MEC, the jobset size is fixed at 280, and 150 jobsets are synthesized for each resource utilization range combination. The runtime performance across varying MEC scales is shown in Fig. 4. With a constant jobset size, the runtime of `IDAssign` exhibits linear growth with the number of ES. This scalability trend, together with observations from Fig. 2(c), demonstrates that `IDAssign` achieves efficient runtime scalability with respect to both jobset size and the number of ES in the MEC.

5 Conclusion

This paper addressed a joint job offloading and resource allocation problem, **P**, in MEC with both bandwidth and computation resource constraints, aiming to maximize the total utility gained from jobs. We proposed an approximation algorithm, `IDAssign`, for **P** with a theoretical bound of $\frac{1}{6}$. `IDAssign` provided the first constant approximation bound for **P**. Experimental results showed that `IDAssign` delivered superior practical performance and exhibited strong runtime scalability. Notably, despite the absence of theoretical guarantees for the baseline algorithms, `IDAssign` achieves comparable practical performance, underscoring its efficiency and effectiveness. In the future, we plan to explore the online deployment of resource allocation algorithms in MEC environments with dynamic network conditions.

Acknowledgments

This work was supported in part by the MoE Tier-2 grant MOE-T2EP20221-0006.

References

[1] Reuven Bar-Yehuda, Keren Bendel, Ari Freund, and Dror Rawitz. 2004. Local ratio: A unified framework for approximation algorithms. In Memoriam: Shimon Even 1935-2004. *ACM Comput. Surv.* 36, 4 (dec 2004), 422–463. doi:10.1145/1041680.1041683

[2] Paul Emberson, Roger Stafford, and Robert I Davis. 2010. Techniques for the synthesis of multiprocessor tasksets. In *proceedings 1st International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS 2010)*. 6–11.

[3] Wenhao Fan, Yi Su, Jie Liu, Shenmeng Li, Wei Huang, Fan Wu, and Yuan'an Liu. 2023. Joint Task Offloading and Resource Allocation for Vehicular Edge Computing Based on V2I and V2V Modes. *IEEE Transactions on Intelligent Transportation Systems* 24, 4 (2023), 4277–4292. doi:10.1109/TITS.2022.3230430

[4] Lisa Fleischer, Michel X Goemans, Vahab S Mirrokni, and Maxim Sviridenko. 2006. Tight approximation algorithms for maximum general assignment problems. In *SODA*, Vol. 6. Citeseer, 611–620.

[5] Chuanchao Gao, Aryaman Shaan, and Arvind Easwaran. 2022. Deadline-constrained Multi-resource Task Mapping and Allocation for Edge-Cloud Systems. In *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*. 5037–5043. doi:10.1109/GLOBECOM48099.2022.10001137

[6] Shihong Hu and Guanghui Li. 2020. Dynamic Request Scheduling Optimization in Mobile Edge Computing for IoT Applications. *IEEE Internet of Things Journal* 7, 2 (2020), 1426–1437. doi:10.1109/JIOT.2019.2955311

[7] Jing Li, Weifa Liang, Yuchen Li, Zichuan Xu, Xiaohua Jia, and Song Guo. 2023. Throughput Maximization of Delay-Aware DNN Inference in Edge Computing by Exploring DNN Model Partitioning and Inference Parallelism. *IEEE Transactions on Mobile Computing* 22, 5 (2023), 3017–3030. doi:10.1109/TMC.2021.3125949

[8] Jing Li, Weifa Liang, Wenzheng Xu, Zichuan Xu, Xiaohua Jia, Wanlei Zhou, and Jin Zhao. 2022. Maximizing User Service Satisfaction for Delay-Sensitive IoT Applications in Edge Computing. *IEEE Transactions on Parallel and Distributed Systems* 33, 5 (2022), 1199–1212. doi:10.1109/TPDS.2021.3107137

[9] Qiuping Li, Junhui Zhao, and Yi Gong. 2019. Cooperative Computation Offloading and Resource Allocation for Mobile Edge Computing. In *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*. 1–6. doi:10.1109/ICCW.2019.8756684

[10] Jie Lin, Siqi Huang, Hanlin Zhang, Xinyu Yang, and Peng Zhao. 2023. A Deep-Reinforcement-Learning-Based Computation Offloading With Mobile Vehicles in Vehicular Edge Computing. *IEEE Internet of Things Journal* 10, 17 (2023), 15501–15514. doi:10.1109/JIOT.2023.3264281

[11] Yu Ma, Weifa Liang, Jing Li, Xiaohua Jia, and Song Guo. 2022. Mobility-Aware and Delay-Sensitive Service Provisioning in Mobile Edge-Cloud Networks. *IEEE Transactions on Mobile Computing* 21, 1 (2022), 196–210. doi:10.1109/TMC.2020.3006507

[12] Nvidia. 2024. NVIDIA Multi-Instance GPU User Guide. https://docs.nvidia.com/datacenter/tesla/mig-user-guide/index.html

[13] Qiao Qi, Xiaoming Chen, and Chau Yuen. 2024. Joint Offloading Selection and Resource Allocation for Integrated Localization and Computing in Edge-intelligent Networks. *IEEE Transactions on Vehicular Technology* (2024), 1–15. doi:10.1109/TVT.2024.3374705

[14] C. E. Shannon. 1948. A mathematical theory of communication. *The Bell System Technical Journal* 27, 3 (1948), 379–423. doi:10.1002/j.1538-7305.1948.tb01338.x

[15] SODA. 2018. Shanghai Qiangsheng Taxi GPS data trace (2018-04-01). https://github.com/hetianzhang/Edge-DataSet?tab=readme-ov-file#taxi-trajectory-data

[16] Thai T. Vu, Diep N. Nguyen, Dinh Thai Hoang, Eryk Dutkiewicz, and Thuy V. Nguyen. 2021. Optimal Energy Efficiency With Delay Constraints for Multi-Layer Cooperative Fog Computing Networks. *IEEE Transactions on Communications* 69, 6 (2021), 3911–3929. doi:10.1109/TCOMM.2021.3064333

[17] Haichuan Wang, Xi Chen, Hongli Xu, Jianchun Liu, and Liusheng Huang. 2019. Joint Job Offloading and Resource Allocation for Distributed Deep Learning in Edge Computing. In *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. 734–741. doi:10.1109/HPCC/SmartCity/DSS.2019.00109

[18] Zichuan Xu, Lin Wang, Weifa Liang, Qiufen Xia, Wenzheng Xu, Pan Zhou, and Omer F. Rana. 2024. Age-Aware Data Selection and Aggregator Placement for Timely Federated Continual Learning in Mobile Edge Computing. *IEEE Trans. Comput.* 73, 2 (2024), 466–480. doi:10.1109/TC.2023.3333213

[19] Qingyong Yang, Shu-Chuan Chu, Chia-Cheng Hu, Lingping Kong, and Jeng-Shyang Pan. 2024. A Task Offloading Method Based on User Satisfaction in C-RAN With Mobile Edge Computing. *IEEE Transactions on Mobile Computing* 23, 4 (2024), 3452–3465. doi:10.1109/TMC.2023.3275580

[20] Xia Zhou. 2021. Wi-Fi 6. https://support.huawei.com/enterprise/en/doc/EDOC1100201276

[21] Tongxin Zhu, Tuo Shi, Jianzhong Li, Zhipeng Cai, and Xun Zhou. 2019. Task Scheduling in Deadline-Aware Mobile Edge Computing Systems. *IEEE Internet of Things Journal* 6, 3 (2019), 4854–4866. doi:10.1109/JIOT.2018.2874954