Energy-Efficient Joint Offloading and Resource Allocation for Deadline-Constrained Tasks in Multi-Access Edge Computing

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

***Abstract*—This paper addresses the deadline-constrained task offloading and resource allocation problem in multi-access edge computing. We aim to determine where each task is offloaded and processed, as well as corresponding communication and computation resource allocations, to maximize the total saved energy for IoT devices, while considering task deadline and system resource constraints. Especially, our system allows each task to be offloaded to one of its accessible access points (APs) and processed on a server that is not co-located with its offloading AP. We formulate this problem as an Integer Nonlinear Programming problem and show it is NP-Hard. To address this problem, we propose a Graph-Matching-based Approximation Algorithm (GMA), the first approximation algorithm of its kind. GMA leverages linear relaxation, tripartite graph construction, and a Linear Programming rounding technique. We prove that GMA is a $\frac{1-\alpha}{2+\epsilon}$-approximation algorithm, where ϵ is a small positive value, and α ($0 \leq \alpha < 1$) is a system parameter that ensures the resource allocated to any task by an AP or a server cannot exceed α times its resource capacity. Experiments show that, in practice, GMA's energy saving achieves 97% of the optimal value on average.**

***Index Terms*—multi-access edge computing, task offloading and resource allocation, deadline-constrained workload**

I. Introduction

Recent advances in hardware, software, and communication technologies—such as ultra-reliable low-latency communication of 5G and low-power wide-area networks—have paved the way for Internet of Things (IoT) to become the next technological frontier [1]. Emerging IoT applications, including object detection and decision-making in autonomous driving [2], are becoming increasingly computation-intensive due to the rapid evolution of Artificial Intelligence (AI) technologies. These applications pose substantial deployment challenges for battery-powered and resource-constrained IoT devices, particularly those with stringent latency requirements [3].

To address these challenges, Multi-access Edge Computing (MEC) has emerged as a promising paradigm for supporting computation-intensive and time-sensitive IoT applications. In MEC, end devices (i.e., IoT devices) can offload computation-intensive tasks to nearby Access Points (APs) via wireless networks. These tasks are subsequently forwarded to some edge server through a wired backhaul network for processing. Unlike conventional cloud computing, MEC deploys servers in close proximity to end devices, significantly reducing communication latency and enabling prompt responses to latency-sensitive tasks. While offloading tasks to MEC servers conserves energy on end devices, it also introduces communication latency and additional energy consumption associated with the offloading process. Moreover, communication and computation resources at APs and servers are limited. Therefore, an effective strategy for task mapping (to APs and servers) and resource allocation (for offloading and processing) is essential for tasks with hard deadlines.

Numerous studies have investigated deadline-constrained task offloading and resource allocation problems. Some studies assume that each task must be offloaded to a fixed AP and focus solely on mapping tasks to servers [4]–[13]. Others consider a model where each task is processed by the server co-located with the AP it is offloaded to, thereby concentrating only on task-to-AP mapping [14]–[18]. In the former case, bandwidth limitations at a single AP may lead to congestion during task offloading, while in the latter, the lack of flexibility in backhaul task forwarding can result in highly imbalanced server workloads. Although some studies have explored task mapping to both APs and servers [19], this area remains underexplored. Furthermore, existing research on deadline-constrained task offloading and resource allocation typically relies on exponential-time exact algorithms for optimal solutions [8], [16], or polynomial-time heuristic algorithms that lack performance guarantees [4]–[7], [9]–[15], [17], [18]. Consequently, the domain of polynomial-time approximation algorithms—heuristics with provable performance guarantees—remains largely unexplored. Approximation algorithms not only improve computational efficiency compared to exponential-time methods but also offer performance guarantees absent in typical heuristics.

This paper addresses the deadline-constrained task offloading and resource allocation problem in MEC, aiming to max-

This work was supported in part by the MoE Tier-2 grant MOE-T2EP20221-0006, and in part by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme.

2325-1301/25/$31.00 ©2025 IEEE
DOI 10.1109/RTCSA66114.2025.00016

imize the total energy savings of end devices while satisfying task deadlines and system resource constraints. In our model, each task can be offloaded to one of several accessible APs (as opposed to a fixed AP). Thus, we must *determine the AP to which each task is offloaded*, along with the associated communication resource allocation. Additionally, each task can be processed on a server that is not necessarily co-located with its offloading AP. Therefore, we must also *determine the server on which each task is processed after offloading*, and allocate the corresponding computation resources. Moreover, to enhance the flexibility and efficiency of the offloading strategy, our model incorporates dynamic offloading power control, which allows each end device to adjust its transmission power during the offloading process. We refer to this Deadline-constrained Task offloading and Resource allocation Problem as DTRP, and formulate it as an Integer Nonlinear Programming (INLP) problem. The Maximum Weight 3-Dimensional Matching (MW3DM) problem can be reduced to a special case of DTRP, in which each job consume full capacity of the AP and server it is mapped to. MW3DM is NP-Hard [20], implying that DTRP is also NP-hard.

Furthermore, we propose the first polynomial-time approximation algorithm for DTRP with provable performance guarantees, termed the Graph Matching-based Approximation Algorithm (GMA). GMA consists of three main steps: 1) We discretize resource allocations for tasks and formulate a Linear Programming (LP) relaxation of the original problem. 2) Based on the LP solution, we create one or more AP/server nodes for each AP/server and construct a weighted tripartite graph connecting tasks, AP nodes, and server nodes. 3) We apply an LP rounding method to derive a matching in the tripartite graph, which is mapped to a feasible solution of DTRP.

The main contributions of this paper are as follows:

- We investigate the DTRP in MEC with both communication and computation contentions, aiming to maximize the total saved energy for end devices. This involves determining task mappings to both APs and servers, the resource allocations for task offloading and processing, and the power for task offloading. We formulate DTRP as a INLP problem and prove it is NP-Hard.
- Based on a novel technique to transform DTRP to a weighted tripartite graph matching problem, we propose the first polynomial-time approximation algorithm, GMA, for DTRP. We prove that GMA is a $\frac{1-\alpha}{2+\epsilon}$-approximation algorithm for DTRP (the objective obtained by GMA is at least $\frac{1-\alpha}{2+\epsilon}$ of the optimal objective of DTRP), where ϵ is a small positive value, and α ($0 \leq \alpha < 1$) is a system parameter indicating that any AP or server cannot allocate more than α times its resource capacity to any single task.
- We experimentally evaluate GMA and compare it with two existing heuristic algorithms [19] for DTRP. Results show that the energy savings obtained by GMA is 97% of the optimal value on average, while outperforming its own theoretical bound and the two heuristic algorithms by 56%, 22% and 7% on average, respectively. GMA outperforms the heuristic algorithms, even while those algorithms do not provide any performance guarantees.

Paper Organization. Section II surveys related work, and Section III specifies the system model and optimization problem. Section IV presents our algorithmic solution and derives its theoretical guarantee. Section V presents the experiment results, and Section VI concludes the paper.

II. Literature Review

Due to the promising potential of MEC, the joint task offloading and resource allocation problem has received considerable attention in recent research. For a comprehensive understanding of this field, readers are referred to relevant surveys [3], [21]–[23]. Research efforts in this domain are generally categorized into deadline-constrained problems and deadline-free problems (which typically focus on minimizing response time). In this section, we review the state-of-the-art algorithms developed for deadline-constrained task offloading and resource allocation in MEC.

The joint task offloading and resource allocation problem becomes particularly challenging when considering both bandwidth contention in wireless networks and computation resource contention at edge servers. Some studies simplify the problem by considering only computation resource contention [4]–[7], [14], [15]. To tackle these simplified scenarios, researchers [4]–[6], [14] have proposed approaches that decompose the original problem into two subproblems—task mapping and resource allocation—and iteratively solve them until convergence. Others have applied reinforcement learning [7], [15]. While these methods exhibit polynomial-time complexity, they lack theoretical performance guarantees. Moreover, these studies omit bandwidth contention in MEC.

Other works explicitly address deadline-constrained task offloading and resource allocation while considering both communication and computation contentions [8]–[13], [16]–[19]. Some adopt exact (optimal) methods such as the branch-and-bound algorithm [16] or Benders decomposition [8], [16], but their exponential time complexity limits practicality in real-world systems. To improve scalability, several studies have proposed heuristic algorithms. For instance, decomposition-based methods are used to iteratively solve task mapping and resource allocation subproblems [9], [17], [18], while deep reinforcement learning approaches are adopted in [10], [11]. Other works explore meta-heuristics, including migrating birds optimization [12] and ant colony optimization [13], or employ greedy strategies based on task deadlines [19]. Although these heuristic methods offer lower computational complexity, they do not provide any theoretical performance guarantees.

Most existing studies, except [19], make restrictive assumptions—either fixing the offloading AP for each task [4]–[13], or requiring task execution on the server co-located with the offloading AP [14]–[18]—limiting flexibility and efficiency of task execution in practical MEC deployments. We address these limitations by considering a more general model that jointly optimizes task mapping to both APs and servers, resource allocation for offloading and processing, and dynamic offloading power control under communication

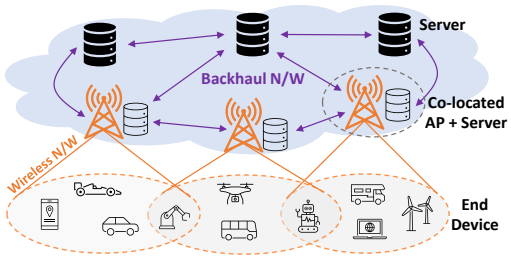

Fig. 1. A typical MEC as considered in this work

and computation contentions. We further propose the first polynomial-time approximation algorithm for DTRP with provable performance guarantees.

III. System Model and Problem Formulation

A. MEC Architecture

An MEC comprises end devices, APs, and servers (Fig. 1). Tasks generated by end devices can be offloaded to nearby APs through wireless networks, and further forwarded to different servers via the wired backhaul network for processing. We denote the set of tasks as $\mathcal{I}$, where $|\mathcal{I}| = I$ ($|\cdot|$ returns the number of items in a set). Each task $i \in \mathcal{I}$ is associated with four parameters: s_i, η_i, and d_i. Here, s_i represents the input size of i measured in Megabits (Mb). η_i denotes the number of CPU cycles required to process 1 bit of input for i, which can be obtained by profiling task execution [24]. d_i denotes the end-to-end deadline of i specified in seconds. In this paper, we consider non-splittable tasks, and each task is either fully processed locally or fully processed on some remote server.

We denote the set of APs as $\mathcal{J}$, where $|\mathcal{J}| = J$. To better balance AP workloads, *each end device can offload its tasks to one of its nearby APs*, and we denote the accessible set of APs for a task i as $\mathcal{J}_i \subseteq \mathcal{J}$. *In real-world systems, resource allocations are typical discrete.* We use $\bar{b}$ to denote the bandwidth unit, measured in MegaHertz (MHz), and b_j to denote the bandwidth capacity for each AP $j \in \mathcal{J}$, measured in number of bandwidth units. Let $\bar{p}$ be the power unit, measured in Watts, and p_{max} be the largest number of power units that end devices can use for task offloading. We denote the set of servers as $\mathcal{K}$, and denote $|\mathcal{K}| = K$. We denote the computation resource unit as $\bar{c}$, specified in CPU cycles/s, and the computation resource capacity of each server $k \in \mathcal{K}$ as c_k, specified in number of computation resource units. To ensure fairness in resource allocation, we introduce a user-defined system parameter α ($0 \leq \alpha < 1$), referred to as the resource allocation bound. This parameter ensures that the resource allocated to any single task by an AP or a server cannot exceed α times its resource capacity. In practice, similar constraints are employed by major cloud service providers to cap the computational resources assigned to individual tasks [25]–[27], and by multi-antenna APs to limit the bandwidth allocated to a single device due to hardware limitations [28]. In this paper, we focus on the problem with the same α for both communication and computation resources. We argue that our proposed solution can also be applied to the problem where the α for communication resource and the α for computation resource are different.

B. Problem Formulation

Local Computing: The local processing time of task i, t_i^l, is given by $t_i^l = s_i\eta_i/f_i$, where f_i is the computation resource capacity available for processing task i locally, specified in CPU cycles/s. In this paper, we assume tasks can meet their deadlines when they are processed locally. The power per CPU cycle for local processing of tasks is given by $p_i^l = \varrho f_i^2$ [29], which is widely adopted in the literature. ϱ is the energy consumption coefficient, depending on the chip architecture. Therefore, the energy consumption for local processing of i, E_i^l, is given as:

$$E_i^l = p_i^l \cdot s_i\eta_i = \varrho f_i^2 s_i\eta_i \tag{1}$$

Task Offloading: We assume that Orthogonal Frequency Division Multiplexing (OFDM) technology is used in the wireless network. OFDM divides the network into multiple orthogonal sub-channels, which can minimize interference between tasks during offloading [30]. Suppose task i is offloaded to AP j. Based on Shannon's theorem [31], the task offloading rate, r_{ij}, can be described as:

$$r_{ij} = b_{ij} \cdot \bar{b} \cdot \log_2(1 + p_{ij}^o \cdot \bar{p} \cdot G_{ij}/\sigma^2) \tag{2}$$

b_{ij} is the allocated bandwidth units to task i, G_{ij} is the channel power gain for offloading task i, and σ is the average noise power. p_{ij}^o is the power units used for offloading i, which cannot exceed p_{max}. We assume G_{ij} and σ are given for each task i and each AP $j \in \mathcal{J}_i$. Thus, the offloading time of task i, t_{ij}^o, is given as

$$t_{ij}^o = \frac{s_i}{r_{ij}} = \frac{s_i}{b_{ij} \cdot \bar{b} \cdot \log_2(1 + p_{ij}^o \cdot \bar{p} \cdot G_{ij}/\sigma^2)}. \tag{3}$$

The consumed offloading energy, E_{ij}^o, can be computed as:

$$E_{ij}^o = p_{ij}^o \cdot \bar{p} \cdot t_{ij}^o = \frac{p_{ij}^o \cdot \bar{p} \cdot s_i}{b_{ij} \cdot \bar{b} \cdot \log_2(1 + p_{ij}^o \cdot \bar{p} \cdot G_{ij}/\sigma^2)}. \tag{4}$$

We use E_{ij} to denote the *saved energy* for end devices when task i is offloaded to AP j given by:

$$E_{ij} = E_i^l - E_{ij}^o \tag{5}$$

Task Forwarding: Allowing tasks to be forwarded to different servers after being offloaded has advantages in balancing server workloads and mitigating wireless network coverage limitations. We consider a wired *backhaul network* that connects APs and servers and has enough capacity to support data transmission with no communication contention[1]. Additionally, we consider the backhaul network is enabled with Software Defined Network technology [24], providing monitor-based latency information among APs and servers. Thus, we assume a constant data transmission delay between

[1]The bandwidth capacity of the IEEE 802.11n Wi-Fi protocol is 120 MHz [32], and that of a wired optical transmission system is 4.5 THz [33].

a given pair of AP and server in the backhaul network [24], i.e., the allocated bandwidth to each task in the wired backhaul network depends on its data size. Let δ_{jk} be the delay between AP j and server k, where $\delta_{jk} = \delta_{kj}$. When AP j and server k are co-located (as in Fig. 1), $\delta_{jk} = 0$.

Server Computing: The total CPU cycles required to process task i is $s_i\eta_i$, and therefore the processing time of task i on server $k \in \mathcal{K}$, t^p_{ik}, is given by $t^p_{ik} = s_i \cdot \eta_i/(c_{ik} \cdot \bar{c})$, where c_{ik} is the allocated computation resource units to task i.

We use binary variables $x_{ij} \in \mathbf{x}$ and $y_{ik} \in \mathbf{y}$ to denote the offloading and processing decisions of task i, respectively. Specifically, $x_{ij} = 1$ if task i is to be offloaded to AP $j \in \mathcal{J}_i$; otherwise, $x_{ij} = 0$. Similarly, $y_{ik} = 1$ if and only if task i is to be processed on server $k \in \mathcal{K}$. Moreover, we use the integer variables $b_{ij} \in \mathbf{b}$ and $p^o_{ij} \in \mathbf{p}$ to denote the bandwidth units to be allocated to task i by AP j and the corresponding allocated offloading power units, respectively. We also use the integer variable $c_{ik} \in \mathbf{c}$ to denote the computation resource units to be allocated to task i by server k. Here, the bold notation is used to denote sets of variables (i.e., $\mathbf{x}$ denotes $\{x_{ij}|\forall i, \forall j\}$). Additionally, we use OPT_P to denote the optimal objective value of a defined problem P. For convenience, we summarize major notations used in this paper in Table I.

In this paper, we aim to find a task mapping $\langle \mathbf{x}, \mathbf{y} \rangle$ and resource allocation $\langle \mathbf{b}, \mathbf{c}, \mathbf{p} \rangle$ solution such that the total saved energy of end devices can be maximized, while satisfying the system resource and task deadline constraints. We refer to this Deadline-constrained Task offloading and Resource allocation Problem as DTRP and define it as follows.

$$(\mathsf{DTRP}) \quad \max \textstyle\sum_{i\in\mathcal{I}, j\in\mathcal{J}, k\in\mathcal{K}} x_{ij} y_{ik} E_{ij} \tag{6a}$$

$$\text{subject to:} \quad \textstyle\sum_{j\in\mathcal{J}_i} x_{ij} t^o_{ij} + \sum_{j\in\mathcal{J}_i, k\in\mathcal{K}} x_{ij} y_{ik} \delta_{jk} + \sum_{k\in\mathcal{K}} y_{ik} t^p_{ik} \le d_i, \forall i \in \mathcal{I} \tag{6b}$$

$$\textstyle\sum_{j\in\mathcal{J}_i} x_{ij} \le 1, \forall i \in \mathcal{I} \tag{6c}$$

$$\textstyle\sum_{j\in\mathcal{J}\setminus\mathcal{J}_i} x_{ij} = 0, \forall i \in \mathcal{I} \tag{6d}$$

$$\textstyle\sum_{k\in\mathcal{K}} y_{ik} \le 1, \forall i \in \mathcal{I} \tag{6e}$$

$$\textstyle\sum_{i\in\mathcal{I}} b_{ij} \le b_j, \forall j \in \mathcal{J} \tag{6f}$$

$$\textstyle\sum_{i\in\mathcal{I}} c_{ik} \le c_k, \forall k \in \mathcal{K} \tag{6g}$$

$$b_{ij} \le \alpha \cdot b_j, c_{ik} \le \alpha \cdot c_k, \ \forall i \in \mathcal{I}, \forall j \in \mathcal{J}, \forall k \in \mathcal{K} \tag{6h}$$

$$p^o_{ij} \le p_{max}, \ \forall i \in \mathcal{I}, \forall j \in \mathcal{J} \tag{6i}$$

$$x_{ij}, y_{ik} \in \{0,1\}, p^o_{ij}, b_{ij}, c_{ik} \in \mathbb{Z}_{\ge 0}, \forall i \in \mathcal{I}, \forall j \in \mathcal{J}, \forall k \in \mathcal{K} \tag{6j}$$

Constraint (6b) guarantees that each offloaded task can be completed within its deadline. Constraints (6c)∼(6e) ensure that a task i can only be offloaded to at most one accessible AP and processed on at most one server. Moreover, constraints (6f) and (6g) ensure that the total resource allocated to all tasks by an AP or a server cannot exceed its capacity. Finally, constraints (6h) and (6i) guarantee that the resource allocation bound and offloading power bound are satisfied. DTRP is an INLP problem due to the nonlinear objective function (6a) and task deadline constraint (6b). Next, we show that DTRP is NP-Hard in the following lemma.

Lemma 1. DTRP *is NP-Hard.*

TABLE I
NOTATION (KEY PARAMETERS AND VARIABLES)

symb.	definition
$\mathcal{I}$	task set, where $\|\mathcal{I}\| = I$. Each task $i \in \mathcal{I}$ is associated with input size s_i, CPU demand per bit data η_i, CPU cycle allocation rate for local processing f_i, and deadline d_i
$\mathcal{J}$	AP set, where $j \in \mathcal{J}$ denotes an AP, and $\|\mathcal{J}\| = J$
$\mathcal{J}_i$	set of APs to which task i can be offloaded ($\mathcal{J}_i \subseteq \mathcal{J}$)
$\mathcal{K}$	server set, where $k \in \mathcal{K}$ denotes a server, and $\|\mathcal{K}\| = K$
δ_{jk}	latency between AP j and server k in the backhaul network
b_j	total number of bandwidth units ($\bar{b}$) of AP $j \in \mathcal{J}$
c_k	total number of computation resource units ($\bar{c}$) of server $k \in \mathcal{K}$
α	resource allocation bound of all APs and servers
t^o_{ij}	offloading time of task i to AP j
t^p_{ik}	processing time of task i on server k
p_{max}	max offloading power units ($\bar{p}$) that can be used for any job
E^l_i	energy consumed for processing task i locally
E^o_{ij}	energy consumed for offloading task i to AP j
E_{ij}	energy saved by offloading task i to AP j for remote processing
φ	resource discretization constant; $B_m = \varphi^m$, $C_n = \varphi^n$
x_{ij}	binary var., if task i is offloaded to AP j; $\mathbf{x} = \{x_{ij} \mid \forall i, \forall j\}$
y_{ik}	binary var., if task i is processed on server k; $\mathbf{y} = \{y_{ik} \mid \forall i, \forall k\}$
b_{ij}	integer var., bandwidth units allocated to task i by AP j; $\mathbf{b} = \{b_{ij} \mid \forall i, \forall j\}$
c_{ik}	integer var., computation resource units allocated to task i by server k; $\mathbf{c} = \{c_{ik} \mid \forall i, \forall k\}$
p^o_{ij}	integer var., allocated power units used to offload task i to AP j; $\mathbf{p} = \{p^o_{ij} \mid \forall i, \forall j\}$

Proof. The decision version of DTRP can be defined as follows: *given an instance of* DTRP *and a target energy saving E, does there exist a feasible solution $\langle \mathbf{x}, \mathbf{y}, \mathbf{b}, \mathbf{c}, \mathbf{p} \rangle$ such that the total energy saving is at least E?* A given solution for the decision version of DTRP can be represented in polynomial size and verified in polynomial time with respect to the input size of DTRP. Therefore, DTRP is an NP problem.

Next, we show that DTRP is NP-Hard through a reduction from the MW3DM problem, known to be NP-Hard [20].

Given an instance of MW3DM: three disjoint sets $\mathcal{V}_1$, $\mathcal{V}_2$, and $\mathcal{V}_3$, a set of weighted triplets $\mathcal{T} \subseteq \mathcal{V}_1 \times \mathcal{V}_2 \times \mathcal{V}_3$, each with weight $u(v_1, v_2, v_3)$, and the goal is to find a maximum weight matching such that each element appears in at most one triplet. We construct a special case of DTRP as follows:

- Let each job i in DTRP correspond to an element in $\mathcal{V}_1$, each AP j in DTRP correspond to an element in $\mathcal{V}_2$, and each server k in DTRP correspond to an element in $\mathcal{V}_3$.
- For every triplet $(i, j, k) \in \mathcal{T}$, define a feasible job mapping in DTRP where job i is offloaded to AP j and processed on server k.
- Assume that each job consumes the full capacity of the AP and server it is mapped to (i.e., resource contention constraints enforce that no AP or server can be shared across jobs).
- Let the energy saving of mapping job i to AP j and server k, with resource allocations b_j and c_k, equal $u(v_1, v_2, v_3)$.

Then, the goal of maximizing the total energy saving in this special case of DTRP is equivalent to solving the MW3DM problem. Since MW3DM is NP-Hard, this special case of DTRP is NP-Hard. Therefore, the general DTRP problem, which generalizes this instance, is also NP-Hard. □

IV. GMA Approximation Algorithm

In MEC, tasks, APs, and servers exist as distinct and independent groups, motivating us to tackle DTRP using graph-matching algorithms for MW3DM. However, directly applying graph matching algorithms may lead to inefficient resource allocation in MEC, as each AP or server can only accommodate one task according to the definition of graph matching (Appendix A). To address this inefficiency, this section introduces a Graph-Matching-based Approximation Algorithm (GMA) for DTRP, the first polynomial-time approximation algorithm for DTRP.

GMA (Algorithm 1) consists of three main steps. First, we perform a resource allocation discretization for reducing the number of resource allocation options and formulate a new LP problem based on DTRP (line 1). Second, based on the LP solution, we construct two bipartite graphs (Appendix B) that establish task-to-AP mappings and task-to-server mappings, respectively. Then, we utilize sub-algorithm WTGConstruct to merge these two bipartite graphs into a weighted tripartite graph (Appendix C) whose each edge represents a task-AP-server mapping (line 2). Third, we employ an LP rounding method to obtain a matching of the weighted tripartite graph (line 3). This matching is then mapped to a feasible solution of DTRP (lines 4–7). Later in this section, we will provide a detailed proof of the theoretical approximation bound for GMA.

Algorithm 1: Graph-Matching-based Approximation (GMA)

1 Discretize task resource allocations and formulate an LP problem, RDP, based on DTRP. Let $\tilde{\mathbf{z}}$ be an optimal fractional solution to RDP;
2 Construct a weighted tripartite graph $\mathcal{H}$ based on $\tilde{\mathbf{z}}$ using WTGConstruct;
3 Formulate a relaxed maximum weighted 3-dimensional matching problem, 3DM, based on $\mathcal{H}$, and apply the kDMA to obtain a (integral) matching $\mathbf{M_z}$ of $\mathcal{H}$;
4 **for each** $M_{\mathbf{z}}(v_i, w_{jr}, w_{ks}) = 1$ **do**
5 $\quad x_{ij} \leftarrow 1, y_{ik} \leftarrow 1$;
6 $\quad b_{ij} \leftarrow b(v_i, w_{jr}, w_{ks}), c_{ik} \leftarrow c(v_i, w_{jr}, w_{ks})$;
7 $\quad$ Calculate p^o_{ij} based on Eqs. (4) and (7);

A. Resource Allocation Discretization and LP Formulation

The nonlinear objective (6a) and constraint (6b) make DTRP challenging to solve, even when the integer variables are relaxed to continuous ones. Since both job mapping and resource allocation variables are discrete, we can enumerate all possible combinations of task mappings and resource allocations, and compute the corresponding saved energy for each. We then focus exclusively on those combinations that satisfy deadline constraints and yield positive energy savings. This approach allows us to eliminate the nonlinear task deadline constraint (6b), while also linearizing the objective (6a), as detailed later in this subsection. However, the number of resource allocation options grows linearly with the value of b_j and c_k, leading to an exponential increase in the total number of combinations as the input size of b_j and c_k grows. To mitigate this issue, we first apply an additional discretization to the resource allocation variables b_{ij} and c_{ik}, thereby reducing the number of candidate options. The remainder of this subsection presents the discretization procedure and the corresponding LP formulation. Later in Theorem 1 (Subsection IV-C), we further show that the optimality gap introduced by this discretization can be effectively bounded.

We define a *discretization constant*, φ, where $\varphi > 1$. For each AP $j \in \mathcal{J}$, let $\pi_j = \lceil \log_\varphi(\alpha b_j) \rceil$. The discretized bandwidth allocations are defined as $\{B_0, B_1, ..., B_{\pi_j}\}$, where $B_m = \lfloor \varphi^m \rfloor$ for $m = 0, ..., \pi_j - 1$, and $B_m = \lfloor \alpha b_j \rfloor$ for $m = \pi_j$. Here, we *use the logarithmic operation to ensure that π_j (i.e., bandwidth allocation options) is polynomial in the size of input b_j*. Similarly, for each server $k \in \mathcal{K}$, let $\lambda_k = \lceil \log_\varphi(\alpha c_k) \rceil$. The discretized computation resource allocations can be defined as $\{C_0, C_1, ..., C_{\lambda_k}\}$, where $C_n = \lfloor \varphi^n \rfloor$, for $n = 0, ..., \lambda_k - 1$, and $C_n = \lfloor \alpha c_k \rfloor$, for $n = \lambda_k$.

We denote a task mapping and resource allocation combination as $\langle i, j, m, k, n \rangle$, representing that task i is offloaded to AP j with bandwidth allocation B_m and processed on server k with computation resource allocation C_n. Given a combination $\langle i, j, m, k, n \rangle$, we can compute the maximum allowable time for task offloading based on deadline constraint (6b), i.e.,

$$t^o_{ij} = \tau_i - \delta_{jk} - s_i \eta_i / C_n. \tag{7}$$

If $t^o_{ij} > 0$, let t^o_{ij} be the task offloading time. Based on t^o_{ij} and bandwidth allocation B_m, we can compute the offloading power p^o_{ij} based on (3), and determine energy E^o_{ij} based on Eq. (4). Then, we compute the saved energy associated with $\langle i, j, m, k, n \rangle$ based on Eq. (5), and we denote it as E_{ijmkn}. We claim that *a combination $\langle i, j, m, k, n \rangle$ is **feasible** if the computed $t^o_{ij} > 0$, $p^o_{ij} \le p_{max}$, and $E_{ijmkn} > 0$.* According to Eq. (7), the following proposition can be obtained.

Proposition 1. *A feasible combination $\langle i, j, m, k, n \rangle$ satisfies the deadline constraint* (6b) *of* DTRP *for task i.*

Once resource allocations are discretized, we can enumerate all $\langle i, j, m, k, n \rangle$. Let $U = \max_{j \in \mathcal{J}} \pi_j$ and $V = \max_{k \in \mathcal{K}} \lambda_k$. The total number of combinations is $IJUKV$. Let $\mathcal{N}$ denote the set of all feasible $\langle i, j, m, k, n \rangle$. The time complexity to obtain the set $\mathcal{N}$ is $\mathcal{O}(IJUKV)$, which is polynomial in the input size of DTRP. We define a new binary variable $z_{ijmkn} \in \mathbf{z}$ for all $\langle i, j, m, k, n \rangle$, and $z_{ijmkn} = 1$ if and only if $\langle i, j, m, k, n \rangle \in \mathcal{N}$ is selected in a task offloading and resource allocation solution. We relax z_{ijmkn} into a continuous variable of range $[0, 1]$ and define an LP problem, denoted as RDP, based on DTRP. We formulate RDP as follows [2].

$$\text{(RDP)} \quad \max \textstyle\sum_{\langle i,j,m,k,n \rangle \in \mathcal{N}} z_{ijmkn} \cdot E_{ijmkn} \tag{8a}$$

$$\text{subject to:} \quad \textstyle\sum_{j,m,k,n} z_{ijmkn} \le 1, \ \forall i \in \mathcal{I} \tag{8b}$$

$$\textstyle\sum_{i,m,k,n} z_{ijmkn} B_m \le (1 - \alpha) \cdot b_j, \ \forall j \in \mathcal{J} \tag{8c}$$

$$\textstyle\sum_{i,j,m,n} z_{ijmkn} C_n \le (1 - \alpha) \cdot c_k, \ \forall k \in \mathcal{K} \tag{8d}$$

[2] In equations, we use $\sum_{i,j,m,k,n}$ as a shorthand notation for $\sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{m=0}^{\pi_j} \sum_{k=1}^{K} \sum_{n=0}^{\lambda_k}$.

$$z_{ijmkn} \geq 0,\ \forall \langle i,j,m,k,n\rangle \in \mathcal{N} \quad (8e)$$

In RDP, we only consider feasible combinations, so the deadline and offloading power requirements for each task are implicitly satisfied. After relaxing $\mathbf{z}$ into a continuous variable, each fractional value of z_{ijmkn} represents a portion of task i. Eq. (8b) ensures that the total portions of each task i do not exceed 1. In the following bipartite graph construction, the AP nodes defined for each AP need at most αb_j **additional** bandwidth resources to accommodate their connected task nodes in a matching (as shown in the proof of Lemma 3). To ensure that the final graph matching result does not violate the bandwidth constraint (6f) of DTRP, we modify the bandwidth constraint in RDP as constraint (8c) (i.e., reserve αb_j bandwidth for bipartite graph construction). The same interpretation applies to the computation resource constraint (8d) of RDP. The following lemma establishes a relation between the optimal solutions of DTRP and RDP.

Lemma 2. $\frac{1-\alpha}{\varphi} OPT_{\mathsf{DTRP}} \leq OPT_{\mathsf{RDP}}$.

Proof. Assume that $\{\mathbf{x},\mathbf{y},\mathbf{b},\mathbf{c},\mathbf{p}\}$ is a feasible solution for DTRP. Suppose an offloaded task i is offloaded to AP j ($x_{ij}=1$) with bandwidth allocation b_{ij} and offloading power p^o_{ij}, and processed on server k ($y_{ik}=1$) with computation resource allocation c_{ik}. Suppose the saved energy by processing i on the server is E_{ij}. The deadline of task i must be met. Besides, suppose $B_{m-1} < b_{ij} \leq B_m$ and $C_{n-1} < c_{ik} \leq C_n$ for some m and n. As $B_{\pi_j} = \alpha b_j$ and $C_{\lambda_k} = \alpha c_k$, such m and n always exist. We then set $z_{ijmkn} = \frac{1-\alpha}{\varphi}$, and repeat this assignment for each offloaded task.

Next, we show that for each $z_{ijmkn} = \frac{1-\alpha}{\varphi}$, the corresponding combination $\langle i,j,m,k,n\rangle$ is always feasible. Since $C_n \geq c_{ik}$, allocating C_n computation resource will result in a shorter processing time compared with allocating c_{ik} computation resource; thus, the allowable time for task offloading t^o_{ij} computed in Eq. (7) is longer. Since $B_m \geq b_{ij}$, based on Eq. (3), a longer offloading time and bandwidth allocation result in a smaller offloading power, i.e., the offloading power p_{ijmkn} associated with $\langle i,j,m,k,n\rangle$ is no greater than p^o_{ij}. According to Eq. (4), a larger bandwidth allocation and smaller offloading power lead to a smaller offloading power consumption. This leads to a greater saved energy for $\langle i,j,m,k,n\rangle$, i.e., $E_{ijmkn} \geq E_{ij}$. Since $t^o_{ij} > 0$, $p_{ijmkn} \leq p^o_{ij} \leq p_{max}$, and $E_{ijmkn} \geq E_{ij} > 0$, the combination $\langle i,j,m,k,n\rangle$ is feasible. Besides, we have the following inequality.

$$\textstyle\sum_{i,j,k} \frac{1-\alpha}{\varphi} x_{ij} y_{ik} E_{ij} \leq \sum_{\langle i,j,m,k,n\rangle \in \mathcal{N}} z_{ijmkn} E_{ijmkn}.$$

Moreover, for each offloaded task i, we construct only one $\langle i,j,m,k,n\rangle$ for it, which ensures the resulting $\mathbf{z}$ will satisfy constraint (8b) in RDP. Further, because $B_{m-1} < b_{ij}$, $B_m < b_{ij}\varphi$. Thus, for each $i \in \mathcal{I}$ and each $j \in \mathcal{J}$, we have

$$\textstyle\sum_{m,k,n} z_{ijmkn} B_m < \frac{1-\alpha}{\varphi} x_{ij} b_{ij} \varphi = (1-\alpha) x_{ij} b_{ij}.$$

According to constraint (6f) in DTRP, $\sum_{i\in\mathcal{I}} x_{ij} b_{ij} \leq b_j$. Thus, we conclude that $\sum_{i,m,k,n} z_{ijmkn} B_m \leq (1-\alpha) b_j,\ \forall j \in \mathcal{J}$, and the resulting $\mathbf{z}$ satisfies constraint (8c) in RDP. Similar arguments can be used to prove that the resulting $\mathbf{z}$ also satisfies constraint (8d) in RDP. Thus, given a feasible solution for DTRP with an objective value P, we can always construct a feasible solution for RDP with an objective value no less than $\frac{1-\alpha}{\varphi} P$. In particular, we can construct a feasible solution for RDP from the optimal solution of DTRP, with an objective value $\frac{1-\alpha}{\varphi} OPT_{\mathsf{DTRP}}$. □

TABLE II
NOTATION USED IN SECTION IV

symb.	definition
π_j	$\pi_j = \lceil \log_\varphi(\alpha b_j) \rceil$, number of discrete bandwidth allocation options after discretization; where $U = \max_{j\in\mathcal{J}} \pi_j$
λ_k	$\lambda_k = \lceil \log_\varphi(\alpha c_k) \rceil$, number of discrete computational resource allocation options after discretization; where $V = \max_{k\in\mathcal{K}} \lambda_k$
z_{ijmkn}	relaxed selection var. of combination $\langle i,j,m,k,n\rangle$
E_{ijmkn}	energy saved associated with combination $\langle i,j,m,k,n\rangle$
$\tilde{z}_{ijmkn}$	$\tilde{z}_{ijmkn} \in \tilde{\mathbf{z}}$, optimal fractional solution of LP problem RDP
$\tilde{x}_{ijm}$	$\tilde{x}_{ijm} \in \tilde{\mathbf{x}}$, derived from Eq. (9) based on $\tilde{z}_{ijmkn}$
$\tilde{y}_{ikn}$	$\tilde{y}_{ikn} \in \tilde{\mathbf{y}}$, derived from Eq. (10) based on $\tilde{z}_{ijmkn}$
B_m	$m \in \{0,1,...,\pi_j\}$, discretized bandwidth allocation
C_n	$n \in \{0,1,...,\lambda_j\}$, discretized computation resource allocation
$\mathcal{B}_{\tilde{\mathbf{x}}}$	$\mathcal{B}_{\tilde{\mathbf{x}}} = (\mathcal{V}_{\tilde{\mathbf{x}}}, \mathcal{W}_{\tilde{\mathbf{x}}}, \mathcal{E}_{\tilde{\mathbf{x}}})$, bipartite graph constructed based on $\tilde{\mathbf{x}}$
$\mathcal{B}_{\tilde{\mathbf{y}}}$	$\mathcal{B}_{\tilde{\mathbf{y}}} = (\mathcal{V}_{\tilde{\mathbf{y}}}, \mathcal{W}_{\tilde{\mathbf{y}}}, \mathcal{E}_{\tilde{\mathbf{y}}})$, bipartite graph constructed based on $\tilde{\mathbf{y}}$
$b(e)$	bandwidth allocation associated with edge e in a graph
$c(e)$	computational res. allocation associated with edge e in a graph
$\tilde{\mathbf{x}}_j$	sorted list of $\tilde{x}_{ijm}$ defined in lines 4–5 of Algorithm 2
$\tilde{x}_{j,s}$	the s-th element in the sorted list $\tilde{\mathbf{x}}_j$
$\mathbf{F}_{\tilde{\mathbf{x}}}$	fractional matching of $\mathcal{B}_{\tilde{\mathbf{x}}}$ derived from $\tilde{\mathbf{x}}$
$M_{\mathbf{x}}$	a (integral) matching of $\mathcal{B}_{\tilde{\mathbf{x}}}$

B. Bipartite Graph Construction

RDP is an LP problem, efficiently solvable using algorithms such as the simplex algorithm or the ellipsoid algorithm. Leveraging an optimal solution for RDP enables us to establish one or more AP or server nodes for each respective AP or server. This facilitates the creation of a weighted tripartite graph, allowing multiple tasks to be potentially mapped to the same AP or server to increase resource allocation efficiency. Directly constructing the tripartite graph can be quite intricate; hence, we first utilize `BGConstruct` (Algorithm 2) to create two bipartite graphs illustrating the task-to-AP and task-to-server mappings, respectively. In the subsequent subsection, we will introduce how to merge these two bipartite graphs into a weighted tripartite graph.

Based on an optimal solution for RDP, denoted as $\tilde{\mathbf{z}}$, we define two variables $\tilde{x}_{ijm} \in \tilde{\mathbf{x}}$ and $\tilde{y}_{ikn} \in \tilde{\mathbf{y}}$ as follows.

$$\tilde{x}_{ijm} = \textstyle\sum_{k,n} \tilde{z}_{ijmkn}, \forall i \in \mathcal{I}, \forall j \in \mathcal{J}, m = 0,...,\pi_j \quad (9)$$

$$\tilde{y}_{ikn} = \textstyle\sum_{j,m} \tilde{z}_{ijmkn}, \forall i \in \mathcal{I}, \forall k \in \mathcal{K}, n = 0,...,\lambda_k \quad (10)$$

As $\tilde{x}_{ijm}$ and $\tilde{y}_{ikn}$ are both in the range $[0,1]$ due to constraint (8b), they represent a portion of task i's mapping and resource allocation. For example, $\tilde{x}_{ijm} = \frac{1}{2}$ indicates that half of task i is offloaded to AP j with a discretized bandwidth allocation B_m. Based on $\tilde{\mathbf{x}}$ ($\tilde{\mathbf{y}}$), we create one or more AP (server) nodes for each AP (server) and construct bipartite graph $\mathcal{B}_{\tilde{\mathbf{x}}}$ ($\mathcal{B}_{\tilde{\mathbf{y}}}$) connecting tasks and AP (server) nodes.

$\tilde{x}_{ijm}$	sort	$\mathcal{E}_{\tilde{\mathbf{x}}}$ create	b update	$F_{\tilde{\mathbf{x}}}$ update
$\tilde{x}_{1j8}=0.5$	$\tilde{x}_{j,1}$	(v_1, w_{j1})	$b(v_1, w_{j1}) = B_8$	$F_{\tilde{\mathbf{x}}}(v_1, w_{j1}) = 0.5$
$\tilde{x}_{2j7}=1$	$\tilde{x}_{j,2}$	(v_2, w_{j1}) (v_2, w_{j2})	$b(v_2, w_{j1}) = B_7$ $b(v_2, w_{j2}) = B_7$	$F_{\tilde{\mathbf{x}}}(v_2, w_{j1}) = 0.5$ $F_{\tilde{\mathbf{x}}}(v_2, w_{j2}) = 0.5$
$\tilde{x}_{3j6}=0.5$	$\tilde{x}_{j,3}$	(v_3, w_{j2})	$b(v_3, w_{j2}) = B_6$	$F_{\tilde{\mathbf{x}}}(v_3, w_{j2}) = 0.5$
$\tilde{x}_{4j5}=0.4$	$\tilde{x}_{j,4}$	(v_4, w_{j3})	$b(v_4, w_{j3}) = B_5$	$F_{\tilde{\mathbf{x}}}(v_4, w_{j3}) = 0.4$

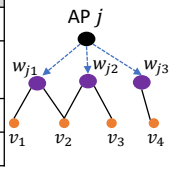

Fig. 2. Example of edge creation for AP j when $n_j > 1$: Given that $\sum \tilde{x}_{ijm} = 2.4$, we create three AP nodes for AP j (where i denotes the task node index and m indicates the bandwidth allocation level B_m). The values of $\tilde{x}_{ijm}$ are sorted in descending order of m, and we denote the s-th element in the sorted list as $\tilde{x}_{j,s}$. Edges between task nodes and AP nodes are then constructed based on this sorted list, with AP nodes ordered as w_{j1}, w_{j2}, w_{j3}. The first element, $\tilde{x}_{j,1}$, corresponds to $\tilde{x}_{1j8}$. Therefore, we create an edge (v_1, w_{j1}) between task node v_1 and AP node w_{j1}, with bandwidth allocation $b(v_1, w_{j1}) = B_8$ and edge fraction $F_{\tilde{\mathbf{x}}}(v_1, w_{j1}) = \tilde{x}_{1j8} = 0.5$. The second element, $\tilde{x}_{j,2}$, corresponds to $\tilde{x}_{2j7}$. Since $\tilde{x}_{1j8} + \tilde{x}_{j,2} > 1$, define two edges based on $\tilde{x}_{j,2}$, (v_2, w_{j1}) and (v_2, w_{j2}). Specifically, we ensure that the total fraction assigned to each AP node does not exceed 1. For example, $F_{\tilde{\mathbf{x}}}(v_1, w_{j1}) + F_{\tilde{\mathbf{x}}}(v_2, w_{j1}) = 1$, and the remaining portion is assigned as $F_{\tilde{\mathbf{x}}}(v_2, w_{j2}) = \tilde{x}_{1j8} + \tilde{x}_{j,2} - 1 = 0.5$. The edge construction process continues similarly for the remaining elements $\tilde{x}_{j,s}$ in the sorted list.

Here, we provide an illustration of constructing $\mathcal{B}_{\tilde{\mathbf{x}}}$ as an example. The detailed steps for constructing $\mathcal{B}_{\tilde{\mathbf{x}}}$ and the resulting fractional matching $\mathbf{F}_{\tilde{\mathbf{x}}}$ for $\mathcal{B}_{\tilde{\mathbf{x}}}$ are outlined in Algorithm 2. Let $\mathcal{B}_{\tilde{\mathbf{x}}} = (\mathcal{V}_{\tilde{\mathbf{x}}}, \mathcal{W}_{\tilde{\mathbf{x}}}, \mathcal{E}_{\tilde{\mathbf{x}}})$. $\mathcal{V}_{\tilde{\mathbf{x}}}$ is the set of *task nodes*, and $\mathcal{W}_{\tilde{\mathbf{x}}}$ is set of *AP nodes* (lines 1–2). By summing up task fractions mapped to AP j, we determine the number of AP nodes that are defined for AP j, which is denoted as n_j and given by

$$n_j \triangleq \left\lceil \textstyle\sum_{i,m} \tilde{x}_{ijm} \right\rceil, \forall j \in \mathcal{J}. \tag{11}$$

Besides, for each edge $e = (v_i, w_{jr}) \in \mathcal{E}_{\tilde{\mathbf{x}}}$, let $b(e) \in \mathbf{b}$ denotes its associated bandwidth allocation. For each AP j, we first sort all positive $\tilde{x}_{ijm}$, for $i \in \mathcal{I}, m \in \{0, 1, ..., \pi_j\}$, in non-increasing order of m, where a larger m represents a larger bandwidth allocation B_m. Let $\tilde{\mathbf{x}}_j$ denote this sorted list, and $\tilde{x}_{j,s}$ denote the s-th element in $\tilde{\mathbf{x}}_j$ (lines 4–5).

If $n_j = 1$, we create a single AP node w_{j1} for AP j and establish connections between this AP node and all the tasks with positive $\tilde{x}_{ijm}$ (lines 6–9).

If $n_j > 1$ (e.g., Fig. 2), we create n_j AP nodes for AP j. We then traverse the sorted list $\tilde{\mathbf{x}}_j$ from left to right, and link task nodes to nodes of AP j such that the sum of fractions ($F_{\tilde{\mathbf{x}}}(e)$) assigned to edges incident on each AP node is exactly 1 (line 13). *This ensures that each AP node can accommodate exactly one task node and prevents resource over-provisioning.* For s-element in $\tilde{\mathbf{x}}_j$, $\tilde{x}_{j,s}$, we first compute $\sum_{l=1}^{s-1} \tilde{x}_{j,l}$ to determine which AP node a task node should link to (line 11). Each $\tilde{x}_{j,s}$ can create one or two edges, depending on the value of $\sum_{l=1}^{s} \tilde{x}_{j,l}$ (lines 12–18).

The bipartite graph $\mathcal{B}_{\tilde{\mathbf{y}}} = (\mathcal{V}_{\tilde{\mathbf{y}}}, \mathcal{W}_{\tilde{\mathbf{y}}}, \mathcal{E}_{\tilde{\mathbf{y}}})$, $\mathbf{c}$, and a fractional matching $F_{\tilde{\mathbf{y}}}$ of $\mathcal{B}_{\tilde{\mathbf{y}}}$ can be obtained using similar steps. Here, $\mathcal{V}_{\tilde{\mathbf{y}}} = \{v_i : i = 1, ..., I\}$ is the set of task nodes, and $\mathcal{W}_{\tilde{\mathbf{y}}} = \{w_{ks} : k = 1, ..., K, s = 1, ..., n_k\}$ is the set of server nodes, where $n_k = \left\lceil \sum_{i,n} \tilde{y}_{ikn} \right\rceil$. Besides, $\mathbf{c}$ is the set of computation resource allocation associated with each edge in $\mathcal{E}_{\tilde{\mathbf{y}}}$.

Algorithm 2: `BGConstruct`

input : $\tilde{\mathbf{x}}$ (*obtained from Eq.*(9))
output: $\mathcal{B}_{\tilde{\mathbf{x}}} = (\mathcal{V}_{\tilde{\mathbf{x}}}, \mathcal{W}_{\tilde{\mathbf{x}}}, \mathcal{E}_{\tilde{\mathbf{x}}})$, $\mathbf{b}$

1 Initialize $\mathcal{B}_{\tilde{\mathbf{x}}} = (\mathcal{V}_{\tilde{\mathbf{x}}}, \mathcal{W}_{\tilde{\mathbf{x}}}, \mathcal{E}_{\tilde{\mathbf{x}}})$ and $\mathbf{b}$;
2 $\mathcal{W}_{\tilde{\mathbf{x}}} \leftarrow \{w_{jr} \mid j = 1, ..., J, r = 1, ..., n_j\}$, $\mathcal{V}_{\tilde{\mathbf{x}}} \leftarrow \{v_i \mid i = 1, ..., I\}$, $\mathcal{E}_{\tilde{\mathbf{x}}} \leftarrow \emptyset$;
3 **for each** $j \in \mathcal{J}$ **do**
4 $\quad \tilde{\mathbf{x}}_j \leftarrow \{\tilde{x}_{ijm} \in \tilde{\mathbf{x}} \mid i \in \mathcal{I}, m = 0, ..., \pi_j, \tilde{x}_{ijm} > 0\}$;
5 $\quad$ sort $\tilde{\mathbf{x}}_j$ in non-increasing order of m values (*ties broken arbitrary; $\tilde{x}_{j,s}$ denotes the s-th element in $\tilde{\mathbf{x}}_j$*);
6 $\quad$ **if** $n_j == 1$, **for** $s \leftarrow 1$ **to** $|\tilde{\mathbf{x}}_j|$ **do**
7 $\quad\quad$ Suppose $\tilde{x}_{j,s}$ corresponds to $\tilde{x}_{ijm}$;
8 $\quad\quad$ **Assign** $((v_i, w_{j1}), B_m, \tilde{x}_{ijm})$;
9 $\quad\quad$ **return**;
$\quad$ `/* for the case of` $n_j > 1$ `*/`
10 $\quad$ **for** $s \leftarrow 1$ **to** $|\tilde{\mathbf{x}}_j|$ **do**
11 $\quad\quad$ Suppose $r - 1 \leq \sum_{l=1}^{s-1} \tilde{x}_{j,l} < r$ for integer r;
12 $\quad\quad$ **if** $\sum_{l=1}^{s} \tilde{x}_{j,l} \leq r$ **then**
$\quad\quad\quad$ `/* create one edge for` $\tilde{x}_{j,s}$ `linked to` w_{jr} `*/`
13 $\quad\quad\quad$ Suppose $\tilde{x}_{j,s}$ corresponds to $\tilde{x}_{ijm}$;
14 $\quad\quad\quad$ **Assign** $((v_i, w_{jr}), B_m, \tilde{x}_{ijm})$;
15 $\quad\quad$ **else**
$\quad\quad\quad$ `/* create two edges for` $\tilde{x}_{j,s}$`, one to` w_{jr}`, one to` $w_{j,r+1}$ `*/`
16 $\quad\quad\quad$ Suppose $\tilde{x}_{j,s}$ corresponds to $\tilde{x}_{ijm}$;
17 $\quad\quad\quad$ **Assign** $((v_i, w_{jr}), B_m, r - \sum_{l=1}^{s-1} \tilde{x}_{j,l})$;
18 $\quad\quad\quad$ **Assign** $((v_i, w_{j,r+1}), B_m, \sum_{l=1}^{s} \tilde{x}_{j,l} - r)$
19 **Function** `Assign` (e, b, x) **:**
20 $\quad$ **if** $e \notin \mathcal{E}_{\tilde{\mathbf{x}}}$ **then** $\mathcal{E}_{\tilde{\mathbf{x}}} \leftarrow \mathcal{E}_{\tilde{\mathbf{x}}} \cup \{e\}, b(e) \leftarrow b$;
21 $\quad$ **if** $e \notin \mathcal{E}_{\tilde{\mathbf{x}}}, F_{\tilde{\mathbf{x}}}(e) \leftarrow x$; **else** $F_{\tilde{\mathbf{x}}}(e) \leftarrow \mathcal{F}_{\tilde{\mathbf{x}}}(e) + x$;

Time Complexity. For each AP j, there are at most I tasks and U discrete bandwidth allocation options. Therefore, $\tilde{\mathbf{x}}_j$ contains at most IU elements. Sorting $\tilde{\mathbf{x}}_j$ has a time complexity of $\mathcal{O}(IU \log(IU))$, and at most two edges are created for each $\tilde{x}_{ijm} > 0$ (lines 12–18). There are at most J APs. Therefore, the time complexity of `BGConstruct` for constructing $\mathcal{B}_{\tilde{\mathbf{x}}}$ is $\mathcal{O}(IJU \log(IU))$. Similarly, for each server k, there are at most I tasks and V discrete computational resource allocation options. The time complexity of `BGConstruct` for constructing $\mathcal{B}_{\tilde{\mathbf{y}}}$ is $\mathcal{O}(IKV \log(IV))$.

In the following lemma, we show that the total resource allocations corresponding to any matching of $\mathcal{B}_{\tilde{\mathbf{x}}}$ (or $\mathcal{B}_{\tilde{\mathbf{y}}}$) satisfies the resource constraints (6f) (or (6g)) of DTRP. This property will then be used to show that any matching of the weighted tripartite graph constructed in Subsection IV-C will also satisfy the resource constraints of DTRP.

Lemma 3. *Suppose $\mathbf{M_x}$ is any (integral) matching of $\mathcal{B}_{\tilde{\mathbf{x}}}$, and $\mathbf{M_y}$ is any matching of $\mathcal{B}_{\tilde{\mathbf{y}}}$. Then, the total allocated bandwidth or computation resource by an AP or a server does not exceed its resource capacity, i.e.,*

$$\textstyle\sum_{i=1}^{I} \sum_{r=1}^{n_j} M_{\mathbf{x}}(v_i, w_{jr}) b(v_i, w_{jr}) \leq b_j, \forall j \in \mathcal{J};$$
$$\textstyle\sum_{i=1}^{I} \sum_{s=1}^{n_k} M_{\mathbf{y}}(v_i, w_{ks}) c(v_i, w_{ks}) \leq c_k, \forall k \in \mathcal{K}.$$

Proof. We apply `BGConstruct` to construct two bipartite graphs ($\mathcal{B}_{\tilde{\mathbf{x}}}$ and $\mathcal{B}_{\tilde{\mathbf{y}}}$) based on the LP solution of RDP. This

Algorithm 3: `WTGConsturct`

input : $\tilde{\mathbf{z}}$ (*optimal solution for* RDP)
output: $\mathcal{H} = (\mathcal{V}_1, \mathcal{V}_2, \mathcal{V}_3, \mathcal{E}), \mathbf{b}, \mathbf{c}, \mathbf{u}$

1 Initialize $\mathcal{H} = (\mathcal{V}_1, \mathcal{V}_2, \mathcal{V}_3, \mathcal{E})$, $\mathbf{b}$, $\mathbf{c}$ and $\mathbf{u}$;
2 Calculate $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{y}}$ based on Eqs. (9) and (10);
3 $\mathcal{B}_{\tilde{\mathbf{x}}} = (\mathcal{V}_{\tilde{\mathbf{x}}}, \mathcal{W}_{\tilde{\mathbf{x}}}, \mathcal{E}_{\tilde{\mathbf{x}}}) \leftarrow$ `BGConstruct`$(\tilde{\mathbf{x}})$;
4 $\mathcal{B}_{\tilde{\mathbf{y}}} = (\mathcal{V}_{\tilde{\mathbf{y}}}, \mathcal{W}_{\tilde{\mathbf{y}}}, \mathcal{E}_{\tilde{\mathbf{y}}}) \leftarrow$ `BGConstruct`$(\tilde{\mathbf{y}})$;
5 $\mathcal{V}_1 \leftarrow \mathcal{V}_{\tilde{\mathbf{x}}}, \mathcal{V}_2 \leftarrow \mathcal{W}_{\tilde{\mathbf{x}}}, \mathcal{V}_3 \leftarrow \mathcal{W}_{\tilde{\mathbf{y}}}, \mathcal{E} \leftarrow \emptyset$;
6 Sort $\tilde{\mathbf{z}}$ in non-increasing order of E_{ijmkn} values (*saved energy of combination* $\langle i, j, m, k, n \rangle$);
7 **forall** $\tilde{z}_{ijmkn} \in \tilde{\mathbf{z}}, \tilde{z}_{ijmkn} > 0$ **do**
8 Let $\mathcal{E}_{ijm}$ be the set containing the one or two edges in $\mathcal{E}_{\tilde{\mathbf{x}}}$ created based on $\tilde{x}_{ijm}$, i.e., $\tilde{x}_{j,s}$ (created in lines 12–18 of `BGConstruct`);
9 Let $\mathcal{E}_{ikn}$ be the set containing the one or two edges in $\mathcal{E}_{\tilde{\mathbf{y}}}$ created based on $\tilde{y}_{ikn}$, i.e., $\tilde{y}_{k,s}$ (created in lines 12–18 of `BGConstruct`);
10 **for each** $(v_i, w_{jr}) \in \mathcal{E}_{ijm}, (v_i, w_{ks}) \in \mathcal{E}_{ikn}$ **do**
11 Define edge $e = (v_i, w_{jr}, w_{ks})$;
12 **if** $e \notin \mathcal{E}$ **then**
13 $\mathcal{E} \leftarrow \mathcal{E} \cup \{e\}$, $u(e) \leftarrow E_{ijmkn}, b(e) \leftarrow b(v_i, w_{jr}), c(e) \leftarrow c(v_i, w_{ks})$;

construction method is inspired by the approach of Shmoys and Tardos [34], who used it to build a bipartite graph from the relaxed LP solution of a one-dimensional GAP. They proved (Theorem 2.1 in [34]) that the total resource usage of an integral matching (e.g., $\mathbf{M_x}$) in the constructed bipartite graph (e.g., $\mathcal{B}_{\tilde{\mathbf{x}}}$) does not exceed the resource capacity specified in the LP (e.g., $(1-\alpha)b_j$ in constraint (8c)) plus the maximum allocation allowed for a single task (e.g., αb_j in constraint (6h)). Although our problem allows different resource allocations per task, the result of Shmoys and Tardos remains applicable. We omit the formal proof here for brevity. Consequently, the total bandwidth demand of $\mathbf{M_x}$ on any AP j is at most $(1-\alpha)b_j + \alpha b_j = b_j$. Similarly, the total computation demand in $\mathcal{B}_{\tilde{\mathbf{y}}}$ on any server k is at most c_k. □

C. Weighted Tripartite Graph Construction

The constructed bipartite graphs $\mathcal{B}_{\tilde{\mathbf{x}}}$ and $\mathcal{B}_{\tilde{\mathbf{y}}}$ have the same set of task nodes, which have a one-to-one correspondence with all the tasks. Thus, we utilize `WTGConstruct` (Algorithm 3) to define a weighted tripartite graph $\mathcal{H} = (\mathcal{V}_1 \cup \mathcal{V}_2 \cup \mathcal{V}_3, \mathcal{E})$ by merging $\mathcal{B}_{\tilde{\mathbf{x}}}$ and $\mathcal{B}_{\tilde{\mathbf{y}}}$. For ease of presentation, we also use $\mathbf{b}$ and $\mathbf{c}$ to denote the communication and computation resource allocations associated with each edge in $\mathcal{E}$, and use $\mathbf{u}$ to denote the weight assigned to each edge in $\mathcal{E}$. Let $\mathcal{V}_1$ contain all the task nodes, $\mathcal{V}_2$ contain all the AP nodes, and $\mathcal{V}_3$ contain all the server nodes (line 5). We first sort $\tilde{\mathbf{z}}$ in non-increasing order of E_{ijmkn} values (ties broken arbitrarily). Each $\tilde{z}_{ijmkn} > 0$ will result in a $\tilde{x}_{ijm} > 0$ based on Eq. (9), which can define to at most two edges in $\mathcal{B}_{\tilde{\mathbf{x}}}$ based on lines 12–18 of `BGConstruct`. Similarly, each $\tilde{z}_{ijmkn} > 0$ will result in a $\tilde{y}_{ikn} > 0$ based on Eq. (10), which can define to at most two edges in $\mathcal{B}_{\tilde{\mathbf{y}}}$. Therefore, for each $\tilde{z}_{ijmkn} > 0$, we can determine the corresponding edge sets $\mathcal{E}_{ijm}$ and $\mathcal{E}_{ikn}$ (lines 8–9). Then, for each possible merged edge e, if it has not been added to set $\mathcal{E}$, we add it to set $\mathcal{E}$, and assign its weight $u(e)$, bandwidth allocation $b(e)$, and computation resource allocation $c(e)$ (lines 10–13).

Time Complexity. The time complexity of `BGConstruct` for constructing $\mathcal{B}_{\tilde{\mathbf{x}}}$ and $\mathcal{B}_{\tilde{\mathbf{y}}}$ are $\mathcal{O}(IJU\log(IU))$ and $\mathcal{O}(IKV\log(IV))$, respectively. The number of positive $\tilde{z}_{ijmkn}$ is at most $IJUKV$, and the time complexity for sorting them is $\mathcal{O}(IJUKV\log(IJUKV))$. Each positive $\tilde{z}_{ijmkn}$ can define at most 4 edges. Therefore, the time complexity of `WTGConstruct` is $\mathcal{O}(IJUKV\log(IJUKV))$.

The following two lemmas summarize some properties of the constructed weighted tripartite graph. feasible combinations $\langle i, j, m, k, n \rangle$ can have $\tilde{z}_{ijmkn} > 0$, and Lemma 5 holds because of Lemma 3.

Lemma 4. *$\forall (v_i, w_{jr}, w_{ks}) \in \mathcal{E}$, the deadline of task i can be met with the task-AP-server combination (i, j, k) and resource allocations $(b(v_i, w_{jr}, w_{ks}), c(v_i, w_{jr}, w_{ks}))$.*

Proof. Based on line 7 of `WTGConstruct`, we only construct an edge $\forall (v_i, w_{jr}, w_{ks}) \in \mathcal{E}$ when the corresponding $\tilde{z}_{ijmkn} \geq 0$. We only define variable $\tilde{z}_{ijmkn}$ when the combination $\langle i, j, m, k, n \rangle$ is feasible, which meets the task deadline requirement (Proposition 1). Based on line 13 of `WTGConstruct`, we set $b(v_i, w_{jr}, w_{ks}) = b(v_i, w_{jr})$, where $b(v_i, w_{jr}) \geq B_m$ based on line 20 of `BGConstruct`. Similarly, we have $c(v_i, w_{jr}, w_{ks}) \geq C_n$. Given the task mapping to AP j and server k, since the deadline of task i can be met with resource allocation B_m and C_n, the deadline of task i can also be met with resource allocation $b(v_i, w_{jr})$ and $c(v_i, w_{ks})$ as the same or more resource are allocated. □

Lemma 5. *Suppose function $\mathbf{M_z}$ is any (integral) matching of the constructed weighted tripartite graph $\mathcal{H} = (\mathcal{V}_1 \cup \mathcal{V}_2 \cup \mathcal{V}_3, \mathcal{E})$. We can obtain the following conclusions:*

$$\sum_{i=1}^{I} \sum_{r=1}^{n_j} \sum_{w_{ks} \in \mathcal{V}_3} M_{\mathbf{z}}(v_i, w_{jr}, w_{ks}) b(v_i, w_{jr}, w_{ks}) \leq b_j, \forall j \in \mathcal{J};$$

$$\sum_{i=1}^{I} \sum_{w_{jr} \in \mathcal{V}_2} \sum_{s=1}^{n_k} M_{\mathbf{z}}(v_i, w_{jr}, w_{ks}) c(v_i, w_{jr}, w_{ks}) \leq c_k, \forall k \in \mathcal{K}.$$

Proof. Based on a matching $\mathbf{M_z}$ for the tripartite graph $\mathcal{H}$, we can easily construct a matching $\mathbf{M_x}$ for the bipartite graph $\mathcal{B}_{\tilde{\mathbf{x}}}$ by letting

$$M_{\mathbf{x}}(v_i, w_{jr}) = \sum_{w_{ks} \in \mathcal{V}_3} M_{\mathbf{z}}(v_i, w_{jr}, w_{ks}).$$

Similarly, we can construct a matching $\mathbf{M_y}$ for the bipartite graph $\mathcal{B}_{\tilde{\mathbf{y}}}$ by letting

$$M_{\mathbf{y}}(v_i, w_{ks}) = \sum_{w_{jr} \in \mathcal{V}_2} M_{\mathbf{z}}(v_i, w_{jr}, w_{ks}).$$

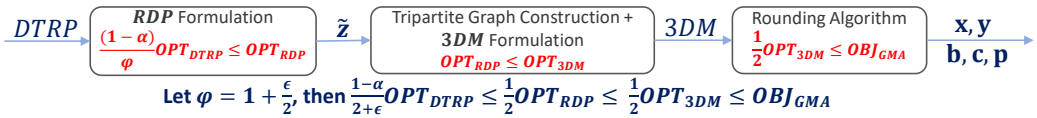

Fig. 3. GMA flow and its approximation ratio induction

Since $b(v_i, w_{jr}, w_{ks}) = b(v_i, w_{jr})$ and $c(v_i, w_{jr}, w_{ks}) = c(v_i, w_{ks})$ according to line 13 of WTGConstruct, this lemma can be proved following the result of Lemma 3. □

By combining Lemma 4 and Lemma 5, we can conclude that any matching of $\mathcal{H}$ satisfies the deadline, offloading power, and resource constraints in DTRP. As each task i has only one corresponding task node v_i, and every edge (v_i, w_{jr}, w_{ks}) of $\mathcal{H}$ corresponds to a task-to-AP-server mapping (i, j, k), a matching of $\mathcal{H}$ can be converted into a feasible solution for DTRP. We first define a relaxed maximum weighted 3-dimensional matching problem, which aims to identify a fractional matching $F_{\mathbf{z}}$ for $\mathcal{H}$ with maximum total weight. Let e represent edge (v_i, w_{jr}, w_{ks}), and $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2 \cup \mathcal{V}_3$. We denote this LP problem as 3DM and formulate it as follows.

$$\text{(3DM)} \quad \max \sum_{e\in\mathcal{E}} F_{\mathbf{z}}(e) u(e) \tag{12}$$

subject to:

$$\sum_{e\in\{e':e'\in\mathcal{E}, v\in e'\}} F_{\mathbf{z}}(e) \le 1, \forall v \in \mathcal{V} \tag{12a}$$

$$F_{\mathbf{z}}(e) \ge 0, \forall e \in \mathcal{E} \tag{12b}$$

Eq. (12a) ensures that the total fractions ($F_{\mathbf{z}}(e)$) of edges incident on node v do not exceed 1 for any $v \in \mathcal{V}$. We can derive the following lemma based on the formulation of 3DM.

Lemma 6. $OPT_{\text{RDP}} \le OPT_{\text{3DM}}$

Proof. In Algorithm 3, we construct edges for all $\tilde{z}_{ijmkn} > 0$. Thus, for an optimal solution $\tilde{\mathbf{z}}$ of RDP, we can construct a feasible solution $\mathcal{M}_{\mathbf{z}}$ of 3DM that satisfies

$$\sum_{r=1}^{n_j} \sum_{s=1}^{n_k} \mathcal{M}_{\mathbf{z}}(v_i, w_{jr}, w_{ks}) = \sum_{m=0}^{\pi_j} \sum_{n=0}^{\lambda_k} \tilde{z}_{ijmkn}.$$

Since the objective value corresponding to this constructed $\mathcal{M}_{\mathbf{z}}$ is no greater than OPT_{3DM}, Lemma 6 follows. □

Here, we employ an algorithm introduced by Chan and Lau, namely the ***k***-Dimensional Matching Algorithm (kDMA) [35], to convert the optimal fractional solution for 3DM into a matching of $\mathcal{H}$. In kDMA, Chan and Lau first solve 3DM, and sort all edges e with positive $F_{\mathbf{z}}(e)$ based on a partial ordering. Then, a recursive function is applied to the sorted list. In each recursive call, one edge e is considered, and the marginal utility of remaining edges is updated based on the selection decision of e. This recursive function eventually returns a (integral) matching $\mathbf{M}_{\mathbf{z}}$ for $\mathcal{H}$. (The details of kDMA are presented in Appendix D.)

Time Complexity. In WTGConstruct, there are at most $IJUKV$ positive $\tilde{z}_{ijkmn}$, which defines at most $4IJUKV$ edges in the weighted tripartite graph $\mathcal{H}$. Therefore, 3DM has at most $4IJUKV$ edges, solving which takes $\mathcal{O}((IJUKV)^3)$ time [36]. The edge sorting operation takes $\mathcal{O}((IJUKV)^2)$. In the recursive function, there are at most $4IJUKV$ recursive layers. In each recursive layer, the marginal utilities of at most $4IJUKV$ remaining edges are updated. Therefore, the time complexity of algorithm kDMA is $\mathcal{O}((IJUKV)^3)$. Based on their findings, we present the following proposition.

Proposition 2. *(Theorem 2.6, [35])* kDMA *can obtain a matching $\mathbf{M}_{\mathbf{z}}$ of $\mathcal{H}$ from the optimal solution of* 3DM*, satisfying the condition*

$$OBJ_{\text{GMA}} = \sum_{e\in\mathcal{E}} M_{\mathbf{z}}(e) u(e) \ge \tfrac{1}{2} OPT_{\text{3DM}}.$$

Next, we prove the theoretical guarantee of GMA. For ease of understanding, we also provide an algorithm flow of GMA and its approximation ratio induction overview in Fig. 3.

Theorem 1. *Let ϵ denote the resource discretization loss, where $\epsilon > 0$ and $\varphi = 1 + \frac{\epsilon}{2}$ (φ is the discretization step defined in Subsection IV-A).* GMA *yields a feasible task offloading and resource allocation solution $\{\mathbf{x}, \mathbf{y}, \mathbf{b}, \mathbf{c}, \mathbf{p}\}$ for* DTRP *with an objective value $OBJ_{\text{GMA}} \ge \frac{1-\alpha}{2+\epsilon} OPT_{\text{DTRP}}$.*

Proof. For each $\mathcal{M}_{\mathbf{z}}(v_i, w_{jr}, w_{ks}) = 1$, GMA sets $x_{ij} = 1$, $y_{ik} = 1$, $b_{ij} = b(v_i, w_{jr}, w_{ks})$, and $c_{ik} = c(v_i, w_{jr}, w_{ks})$. As each task only has one corresponding task node in graph $\mathcal{H}$, at most one edge that contains node v_i can be selected for each task i in a matching of $\mathcal{H}$. Thus, $OBJ_{\text{GMA}} = \sum_{e\in\mathcal{E}} M_{\mathbf{z}}(e) u(e)$, and the resulting solution $\{\mathbf{x}, \mathbf{y}, \mathbf{b}, \mathbf{c}, \mathbf{p}\}$ satisfies constraints (6c)∼(6e) of DTRP. Based on Lemma 4 and Lemma 5, the resulting solution also satisfies the deadline, offloading power and resource constraints of DTRP. Therefore, the resulting solution is a feasible solution for DTRP.

According to Proposition 2, $OBJ_{\text{GMA}} \ge \frac{1}{2} OPT_{\text{3DM}}$. Besides, based on Lemma 6 ($OPT_{\text{3DM}} \ge OPT_{\text{RDP}}$) and Lemma 2 ($OPT_{\text{RDP}} \ge \frac{1-\alpha}{\varphi} OPT_{\text{DTRP}}$), we can conclude

$$OBJ_{\text{GMA}} \ge \tfrac{1}{2} OPT_{\text{3DM}} \ge \tfrac{1}{2} OPT_{\text{RDP}} \ge \tfrac{1-\alpha}{2\varphi} OPT_{\text{DTRP}}.$$

Substituting $\varphi = 1 + \frac{\epsilon}{2}$, we get $OBJ_{\text{GMA}} \ge \frac{1-\alpha}{2+\epsilon} OPT_{\text{DTRP}}$. In Subsection IV-A, we set $\pi_j = \lceil \log_{\varphi}(\alpha b_j) \rceil$. To ensure π_j is polynomial in the size of input b_j, φ should be strictly greater than 1. As a result, $\epsilon > 0$. □

GMA Time Complexity Analysis:

- Line 1: The LP problem RDP has at most $IJUKV$ variables, solving which takes $\mathcal{O}((IJUKV)^3)$ time [36].
- Line 2: Using WTGConstruct to construct the tripartite graph takes $\mathcal{O}(IJUKV \log(IJUKV))$ time.
- *Line 3*: The time complexity of kDMA is $\mathcal{O}((IJUKV)^3)$. Therefore, the time complexity for obtaining a matching $\mathcal{H}$ is $\mathcal{O}((IJUKV)^3)$.

- *Line* 4–7: The time complexity is $\mathcal{O}(I)$ since at most I tasks can be offloaded.

As a result, the time complexity of GMA is $\mathcal{O}((IJUKV)^3)$. In Subsection IV-A, we introduce a logarithmic function to discretize the resource allocation. This log-based discretization ensures that U and V grow polynomially with respect to the input sizes of b_j and c_k. Hence, the overall time complexity of GMA becomes polynomial in the input size of DTRP.

Discussion. In GMA, we incorporate the tripartite graph matching algorithm proposed by Chan and Lau [35]. However, their method alone does not determine task-specific resource allocations. To address this, GMA introduces a novel combination of resource discretization and tripartite graph construction, enabling joint optimization of task mapping (to both APs and servers), resource allocation (for both offloading and processing), and offloading power control. This constitutes the first approximation algorithm for DTRP with polynomial-time complexity. In real-world cloud infrastructures, resource allocation bounds are typically tight. For instance, Alibaba datacenters allow servers with over 96 logical CPU cores, yet cap per-task allocations at 16 cores [37]. Similarly, Google Cloud Run [25], Azure Functions [38], and AWS Lambda [27] limit allocations to 8, 4, and 6 cores, respectively. These configurations imply that the resource allocation bound α is generally no greater than $\frac{1}{6}$ in practice, making the approximation ratio of GMA close to $\frac{1}{2}$ (since ϵ is a small positive constant). Finally, we note that DTRP is significantly more complex than GAP, for which the best-known and tight approximation ratio is $\frac{1}{2}$ [34]. Therefore, achieving a provable bound exceeding $\frac{1}{2}$ for DTRP is unlikely. These insights suggest that GMA offers a practical deterministic bound approaching its maximum likely approximation ratio of $\frac{1}{2}$.

V. Experimental Evaluation

In this section, we evaluate the performance of GMA through numerical simulations. To ensure a comprehensive assessment, we compare GMA with two existing heuristic algorithms based on the achieved performance ratio R, defined as

$$R = \frac{\text{total saved energy by the algorithm}}{\text{total saved energy by an optimal policy}}.$$

Optimal Policy: Since obtaining the exact solution to DTRP is intractable, we formulate a new LP problem derived from RDP by replacing $(1-\alpha)$ in Eqs. (8c) and (8d) with the discretization factor φ. It can be shown (similar to the proof of Lemma 2) that the optimal solution of this LP problem is at least as large as OPT_{DTRP}. Therefore, we adopt this solution as the optimal policy for computing R. Since the denominator is no less than OPT_{DTRP}, the computed value of R underestimates the true performance ratio based on the actual optimal saved energy OPT_{DTRP}. As a result, the values reported in Figs. 4 and 5 serve as lower bounds on the actual performance that the algorithms can achieve.

A. MEC Architecture and Taskset Generation

We sample system-related parameters and tasksets from ranges considered in existing studies and practical systems. The numbers of APs and servers are fixed at 12 and 15, respectively. 12 of the 15 servers are co-located with APs. We sample c_k of each server $k \in \mathcal{K}$ within $20 \sim 30$ Giga cycles/s [39], and choose b_j of each AP $j \in \mathcal{J}$ from $\{80, 120\}$ MHz (802.11n Wi-Fi Protocol) [32]. The backhaul network delay δ_{jk} is sampled from $3 \sim 30$ ms [24]. We set the wireless network channel gain G_{ij} as -50 dB [40], noise power σ^2 as $8e^{-8}$, and maximum offloading power p_{max} as 0.1 W [41].

We generate tasksets with varying resource utilization targets $\langle r_b, r_c \rangle$, and taskset size I. The resource utilization target r_b (r_c) of a taskset is the ratio of the total targeted communication (computation) resource demand of all tasks in a taskset to the total communication (computation) resource of the system [19]. We consider two ranges, $LR = [0.7, 1]$ and $HR = [1.2, 1.5]$, and sample $\langle r_b, r_c \rangle$ from four different range combinations: $\langle LR, LR \rangle$, $\langle LR, HR \rangle$, $\langle HR, LR \rangle$, and $\langle HR, HR \rangle$. For each range combination, we sample 30 different $\langle r_b, r_c \rangle$ values. For each sampled $\langle r_b, r_c \rangle$, we sample 30 different I from $[50, 200]$.

Given the values for $\langle r_b, r_c \rangle$ and I, we generate a single taskset as follows. We randomly sample s_i in $[100, 200]$ Kb that matches the size of a typical image and set $\eta_i = 150$ for each task i [17]. Next, we sample the local computation resource capacity f_i within $[1, 2]$ Giga cycles/s [41]. Given r_b and r_c, let $R_b = r_b \sum_{j \in \mathcal{J} b_j}$ and $R_c = r_c \sum_{k \in \mathcal{K}} c_k$ be the targeted bandwidth and computation resource demand of the taskset, respectively. Then, we use Stafford's Randfixedsum Algorithm [42] to distribute R_b and R_c to each individual task in the taskset in a uniformly random and unbiased manner. Let R_b^i and R_k^i be the assigned targeted bandwidth and computation resource demand of task i. We set d_i with $d_i = \frac{s_i}{r_{ij}} + \delta_i + \frac{s_i \eta_i}{R_k^i}$, where $\delta_i \sim \mathcal{N}(8, 3)$ ms covers half of the sample range of δ_{jk}, and r_{ij} is computed based on Eq. (2) with $p_{ij}^o = p_{max}$ and $b_{ij} = R_b^i$. For each taskset, we choose $|\mathcal{J}_i|$ in $\{2, 3\}$ and randomly assign tasks to APs, while ensuring that the number of tasks that can be offloaded to each AP is drawn from a normal distribution. This, combined with Stafford's Randfixedsum Algorithm for tasks' targeted resource demand assignment, ensures we generate various distributions of workload demand and their assignment to APs, in an unbiased manner.

Baseline Algorithms. We employ two heuristic algorithms proposed by Gao *et al.* [19], namely ZSG and LDM, as the baseline algorithms. Their study proposed a similar MEC architecture, and jointly considered task mapping (to both APs and servers) and resource allocations (for both offloading and processing) for deadline-constrained tasks, with the aim of maximizing user-defined profit.

1) For each task mapping, ZSG estimates the time for task offloading and task processing based on the task's data size and required compute cycles, which are then used to compute the resource allocations. Then, ZSG greedily selects the mapping and resource allocation with the highest energy-to-resource allocation ratio whenever the system possesses sufficient resources.

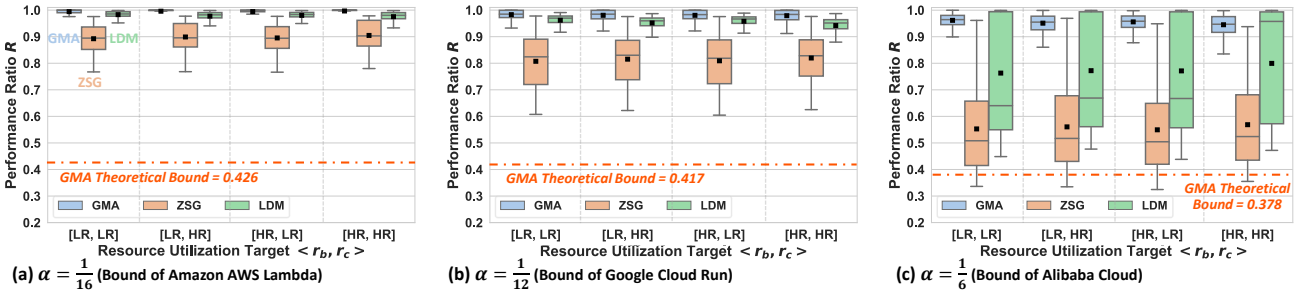

Fig. 4. Performance Ratio (R) by different algorithms (GMA, ZSG, LDM) under varying resource allocation bounds α

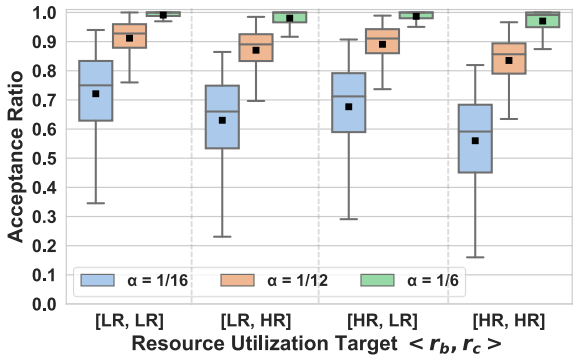

Fig. 5. Achieved Acceptance Ratio by GMA

2) LDM reformulates DTRP into an Integer Linear Programming (ILP) problem by discretizing the system's resource allocations using *equal-sized intervals*, where the interval sizes of 1 MHz and 50 Mega cycles/s are used for bandwidth and computational resources, respectively. Then, an ILP solver is employed to solve the ILP problem.

We set ϵ as 0.2 for GMA, and allocate the same runtime for LDM as GMA. For each taskset, we run GMA, ZSG, and LDM with varying α in $\{\frac{1}{16}, \frac{1}{12}, \frac{1}{6}\}$ (bounds obtained from Amazon AWS Lambda, Google Cloud Run, and Alibaba Cloud). Thus, each algorithm runs 10800 simulations. Experiments were conducted on a desktop PC with an Intel(R) Xeon(R) E7-8880V4 2.20GHz CPU and 32GB of RAM.

B. Performance Evaluation

We evaluate the performance of different algorithms by comparing the achieved performance ratio R under various α values (Fig. 4). (Note that the line in the middle of the box refers to the median value, and the black square dot refers to the average value.) GMA achieves an average performance ratio of 99.5% for $\alpha = \frac{1}{16}$, 98.1% for $\alpha = \frac{1}{12}$, and 95.3% for $\alpha = \frac{1}{6}$. Considering $\epsilon = 0.2$, GMA provides a theoretical bound of 0.426 for $\alpha = \frac{1}{16}$, 0.417 for $\alpha = \frac{1}{12}$, and 0.378 for $\alpha = \frac{1}{6}$. As a result, GMA's practical performance surpasses its theoretical bounds by an average of 56.93%, and shows its stability under varying resource allocation bounds and resource usage levels. The results also show that GMA effectively bridges the gap between its theoretical bound and the optimal solution, showing the practical efficiency of GMA.

GMA exhibits an average performance ratio that is 22.04% and 7.36% higher than ZSG and LDM, respectively. Notably, unlike GMA, both ZSG and LDM do not have any theoretical guarantees. Compared to ZSG, GMA performs significantly better and has far less variability. Besides, the performance of LDM is comparable with GMA when α equals $\frac{1}{16}$ and $\frac{1}{12}$, but degrades and becomes highly variable when $\alpha = \frac{1}{6}$. Thus, in conclusion, GMA provides the first algorithm for solving DTRP with a theoretical approximation bound and excellent performance with low variation for practical systems.

We also evaluate the achieved acceptance ratio by GMA under various α values (Fig. 5). The acceptance ratio is the ratio of the number of tasks being offloaded to the number of tasks in the taskset. GMA achieves an average acceptance ratio of 64.7% for $\alpha = \frac{1}{16}$, 87.7% for $\alpha = \frac{1}{12}$, and 98.2% for $\alpha = \frac{1}{6}$. As illustrated in Fig. 5, the acceptance ratio exhibits a notable sensitivity to the choice of α, where a higher α corresponds to an elevated average acceptance ratio. This correlation arises because a larger α increases the resources that can be allocated to each task, thereby enhancing their ability to meet deadlines and resulting in an elevated acceptance ratio. Furthermore, our observations indicate that an increasing resource utilization target $\langle r_b, r_c \rangle$ is associated with a marginal decrease in the acceptance ratio. This phenomenon arises due to the inherent complexity of provisioning tasksets with higher resource utilization targets, making them comparatively more challenging to accommodate.

VI. Conclusion

This paper investigated the deadline-constrained task offloading and resource allocation problem in MEC with both communication and computation resource contentions. In this general system, we jointly optimized task mapping to both APs and servers, resource allocation for offloading and processing, and dynamic offloading power control. To address this problem, we proposed the Graph-Matching-based Approximation Algorithm (GMA), the first polynomial-time approximation algorithm of its kind. GMA achieves a provable approximation ratio of $\frac{1-\alpha}{2+\epsilon}$, where α is the resource allocation bound and ϵ is a small positive constant. Experimental results demonstrated

that GMA consistently outperforms existing baseline algorithms in both effectiveness and stability.

For future work, we plan to investigate the resource augmentation problem in MEC. Specifically, given that a feasible solution exists that admits all tasks, the goal is to find a solution that minimizes overall system resource usage while ensuring all tasks are still successfully admitted.

Appendix A
Fractional Matching and Matching

Consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the node set and $\mathcal{E}$ is the edge set. A fractional matching [43] of $\mathcal{G}$ is a function $\mathbf{m}$ that assigns each edge $e \in \mathcal{E}$ with a fraction m_e in the range of $[0, 1]$, such that for every node $v \in \mathcal{V}$, the total fractions of edges incident on v is at most 1, i.e., $\sum_{e:v\in e, e\in\mathcal{E}} m_e \leq 1, \forall v \in \mathcal{V}$. If $m_e \in \{0, 1\}$ for each edge in $\mathcal{E}$, $\mathbf{m}$ is a (integral) matching [44] of $\mathcal{G}$. An example is provided in Fig. 6.

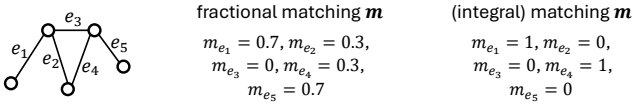

Fig. 6. An example of a fractional matching and a (integral) matching of a graph with 5 nodes and 5 edges.

Appendix B
Bipartite Graph

A bipartite graph [44] $\mathcal{B} = (\mathcal{V}, \mathcal{W}, \mathcal{E})$ is a graph whose nodes can be divided into two disjoint and independent sets $\mathcal{V}$ and $\mathcal{W}$, and every edge in the edge set $\mathcal{E}$ connects a node in $\mathcal{V}$ to a node in $\mathcal{W}$. Besides, no edge connects two nodes in the same set. An example of a bipartite graph is provided on the left of Fig. 7, where each edge comprises two nodes, one from each partitioned node set.

Appendix C
Weighted Tripartite Graph

A tripartite graph [35] $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ is a hypergraph whose node set $\mathcal{V}$ can be partitioned into three disjoint and independent sets, $\mathcal{V}_1, \mathcal{V}_2$ and $\mathcal{V}_3$. Each edge $e \in \mathcal{E}$ is a subset of $\mathcal{V}$, which contains exactly three nodes and intersects each partitioned set ($\mathcal{V}_1, \mathcal{V}_2$ or $\mathcal{V}_3$) in exactly one node. A *weighted tripartite graph* is a tripartite graph where each edge $e \in \mathcal{E}$ is associated with a real-valued weight u_e. An example of a tripartite graph is provided on the right of Fig. 7.

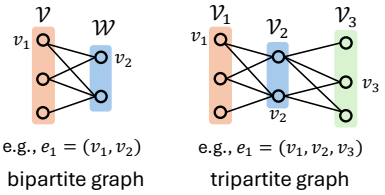

Fig. 7. An example of a bipartite graph (left) and a tripartite graph (right).

Appendix D
k-Dimensional Matching Algorithm

Algorithm 4: kDMA [35]

Find an optimal basic solution $\mathbf{F_z}$ to 3DM. Remove every hyperedge e from $\mathcal{E}$ with $F_{\mathbf{z}}(e) = 0$. Initialize $\mathcal{Q} \leftarrow \emptyset$;
for $s \leftarrow 1$ *to* $|\mathcal{E}|$ **do**
  /* Let $\mathcal{N}(e)$ denote the set of hyperedges that intersect e, including e itself */
  Find a hyperedge e with $F_{\mathbf{z}}(\mathcal{N}(e)) \leq 2$;
  Add e to the end of $\mathcal{Q}$, and remove it from $\mathcal{E}$;
  Remove $F_{\mathbf{z}}(e)$ from $\mathbf{F_z}$;
$\mathbf{M_z} \leftarrow$ **LocalRatio**$(\mathcal{Q}, \mathbf{u})$;
return $\mathbf{M_z}$;
/* a recursive subroutine */
Function LocalRatio$(\mathcal{Q}, \mathbf{u})$**:**
  Remove from $\mathcal{Q}$ all hyperedges with non-positive weights;
  if $\mathcal{Q} = \emptyset$ **then return** $\emptyset$;
  Choose the leftmost hyperedge e from updated $\mathcal{Q}$. Decompose the weight vector $\mathbf{u} = \mathbf{u_1} + \mathbf{u_2}$ where

$$u_1(e') = \begin{cases} u(e) & \text{if } e' \in \mathcal{N}(e), \\ 0 & \text{otherwise.} \end{cases}$$

  $\mathcal{S}' \leftarrow$ **LocalRatio**$(\mathcal{Q}, \mathbf{u_2})$;
  if $\mathcal{S}' \cup \{e\}$ *is a matching* **then return** $\mathcal{S} = \mathcal{S}' \cup \{e\}$; **else return** $\mathcal{S} = \mathcal{S}'$;

References

[1] M. Aazam, S. Zeadally, and K. A. Harras, "Fog computing architecture, evaluation, and future research directions," *IEEE Communications Magazine*, vol. 56, no. 5, pp. 46–52, 2018.

[2] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, vol. 8, pp. 58 443–58 469, 2020.

[3] S. Ramanathan, N. Shivaraman, S. Suryasekaran, A. Easwaran, E. Borde, and S. Steinhorst, "A survey on time-sensitive resource allocation in the cloud continuum," *it - Information Technology*, vol. 62, no. 5-6, pp. 241–255, 2020. [Online]. Available: https://doi.org/10.1515/itit-2020-0013

[4] H. Zhou, Z. Zhang, D. Li, and Z. Su, "Joint optimization of computing offloading and service caching in edge computing-based smart grid," *IEEE Transactions on Cloud Computing*, vol. 11, no. 2, pp. 1122–1132, 2023.

[5] J. Liu, S. Guo, Q. Wang, C. Pan, and L. Yang, "Optimal multi-user offloading with resources allocation in mobile edge cloud computing," *Computer Networks*, vol. 221, p. 109522, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128622005564

[6] C. Xu, G. Zheng, and X. Zhao, "Energy-minimization task offloading and resource allocation for mobile edge computing in noma heterogeneous networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 16 001–16 016, 2020.

[7] P. Dai, K. Liu, X. Wu, H. Xing, Z. Yu, and V. C. S. Lee, "A learning algorithm for real-time service in vehicular networks with mobile-edge computing," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–6.

[8] W. Fan, Y. Su, J. Liu, S. Li, W. Huang, F. Wu, and Y. Liu, "Joint task offloading and resource allocation for vehicular edge computing based on v2i and v2v modes," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 4, pp. 4277–4292, 2023.

[9] M. Zhao, J.-J. Yu, W.-T. Li, D. Liu, S. Yao, W. Feng, C. She, and T. Q. S. Quek, "Energy-aware task offloading and resource allocation for time-sensitive services in mobile edge computing systems," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 10, pp. 10 925–10 940, 2021.

[10] X. Chen and G. Liu, "Energy-efficient task offloading and resource allocation via deep reinforcement learning for augmented reality in mobile edge networks," *IEEE Internet of Things Journal*, vol. 8, no. 13, pp. 10 843–10 856, 2021.

[11] J. Baek and G. Kaddoum, "Heterogeneous task offloading and resource allocations via deep recurrent reinforcement learning in partial observable multifog networks," *IEEE Internet of Things Journal*, vol. 8, no. 2, pp. 1041–1056, 2021.

[12] H. Yuan and M. Zhou, "Profit-maximized collaborative computation offloading and resource allocation in distributed cloud and edge computing systems," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 3, pp. 1277–1287, 2021.

[13] J. Fan, X. Wei, T. Wang, T. Lan, and S. Subramaniam, "Deadline-aware task scheduling in a tiered iot infrastructure," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, 2017, pp. 1–7.

[14] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint offloading and resource allocation in vehicular edge computing and networks," in *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1–7.

[15] Y. Dai, K. Zhang, S. Maharjan, and Y. Zhang, "Edge intelligence for energy-efficient computation offloading and resource allocation in 5g beyond," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 10, pp. 12 175–12 186, 2020.

[16] T. T. Vu, D. N. Nguyen, D. T. Hoang, E. Dutkiewicz, and T. V. Nguyen, "Optimal energy efficiency with delay constraints for multi-layer cooperative fog computing networks," *IEEE Transactions on Communications*, vol. 69, no. 6, pp. 3911–3929, 2021.

[17] C. Xu, G. Zheng, and X. Zhao, "Energy-minimization task offloading and resource allocation for mobile edge computing in noma heterogeneous networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 16 001–16 016, 2020.

[18] Q. Li, J. Zhao, and Y. Gong, "Cooperative computation offloading and resource allocation for mobile edge computing," in *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2019, pp. 1–6.

[19] C. Gao, A. Shaan, and A. Easwaran, "Deadline-constrained multi-resource task mapping and allocation for edge-cloud systems," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, 2022, pp. 5037–5043.

[20] R. E. Burkard, M. Dell'Amico, and S. Martello, *Assignment Problems*. Philadelphia, PA: SIAM, 2009, chapter 7: Multi-dimensional Assignment Problems.

[21] A. Islam, A. Debnath, M. Ghose, and S. Chakraborty, "A survey on task offloading in multi-access edge computing," *Journal of Systems Architecture*, vol. 118, p. 102225, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1383762121001570

[22] F. Saeik, M. Avgeris, D. Spatharakis, N. Santi, D. Dechouniotis, J. Violos, A. Leivadeas, N. Athanasopoulos, N. Mitton, and S. Papavassiliou, "Task offloading in edge and cloud computing: A survey on mathematical, artificial intelligence and control theory solutions," *Computer Networks*, vol. 195, p. 108177, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128621002322

[23] N. Santi and N. Mitton, "A resource management survey for mission critical and time critical applications in multi access edge computing," *ITU Journal on Future and Evolving Technologies*, vol. 2, no. 2, Nov. 2021. [Online]. Available: https://hal.science/hal-03420193

[24] H. A. Alameddine, S. Sharafeddine, S. Sebbah, S. Ayoubi, and C. Assi, "Dynamic task offloading and scheduling for low-latency iot services in multi-access edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 3, pp. 668–682, 2019.

[25] G. Cloud, "Cloud run cpu limits," 2023. [Online]. Available: https://cloud.google.com/run/docs/configuring/cpu

[26] K. Wang, Y. Li, C. Wang, T. Jia, K. Chow, Y. Wen, Y. Dou, G. Xu, C. Hou, J. Yao, and L. Zhang, "Characterizing job microarchitectural profiles at scale: Dataset and analysis," in *Proceedings of the 51st International Conference on Parallel Processing*, ser. ICPP '22. New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: https://doi.org/10.1145/3545008.3545026

[27] A. AWS, "Aws lambda now supports up to 10 gb of memory and 6 vcpu cores for lambda functions," 2020. [Online]. Available: https://aws.amazon.com/about-aws/whats-new/2020/12/aws-lambda-supports-10gb-memory-6-vcpu-cores-lambda-functions/

[28] Qualcomm, "802.11ac mu-mimo: Bridging the mimo gap in wi-fi," 2023. [Online]. Available: https://www.qualcomm.com/content/dam/qcomm-martech/dm-assets/documents/802.11ac_mu-mimo_bridging_the_mimo_gap_in_wi-fi.pdf

[29] D. Han, W. Chen, and Y. Fang, "Joint channel and queue aware scheduling for latency sensitive mobile edge computing with power constraints," *IEEE Transactions on Wireless Communications*, vol. 19, no. 6, pp. 3938–3951, 2020.

[30] L. Li, T. Q. Quek, J. Ren, H. H. Yang, Z. Chen, and Y. Zhang, "An incentive-aware job offloading control framework for multi-access edge computing," *IEEE Transactions on Mobile Computing*, vol. 20, no. 1, pp. 63–75, 2021.

[31] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[32] Intel, "Different wi-fi protocols and data rates," 2023. [Online]. Available: https://www.intel.com/content/www/us/en/support/articles/000005725/wireless/legacy-intel-wireless-products.html

[33] T. J. Xia and G. A. Wellbrock, "Commercial 100-gbit/s coherent transmission systems," *Optical Fiber Telecommunications*, pp. 45–82, 2013.

[34] D. B. Shmoys and É. Tardos, "An approximation algorithm for the generalized assignment problem," *Mathematical programming*, vol. 62, no. 1, pp. 461–474, 1993.

[35] Y. H. Chan and L. C. Lau, "On linear and semidefinite programming relaxations for hypergraph matching," *Mathematical programming*, vol. 135, no. 1, pp. 123–148, 2012.

[36] P. M. Vaidya, "An algorithm for linear programming which requires o(((m+n)n2+(m+n)1.5n)l) arithmetic operations," in *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, ser. STOC '87. New York, NY, USA: Association for Computing Machinery, 1987, p. 29–38. [Online]. Available: https://doi.org/10.1145/28395.28399

[37] Q. Weng, W. Xiao, Y. Yu, W. Wang, C. Wang, J. He, Y. Li, L. Zhang, W. Lin, and Y. Ding, "MLaaS in the wild: Workload analysis and scheduling in Large-Scale heterogeneous GPU clusters," in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. Renton, WA: USENIX Association, Apr. 2022, pp. 945–960. [Online]. Available: https://www.usenix.org/conference/nsdi22/presentation/weng

[38] Microsoft, "Azure functions premium plan," 2023. [Online]. Available: https://learn.microsoft.com/fi-fi/azure/azure-functions/functions-premium-plan?tabs=portal#available-instance-skus

[39] B. Xiang, J. Elias, F. Martignon, and E. Di Nitto, "A dataset for mobile edge computing network topologies," *Data in Brief*, vol. 39, p. 107557, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352340921008337

[40] F. Zhou, Y. Wu, R. Q. Hu, and Y. Qian, "Computation rate maximization in uav-enabled wireless-powered mobile-edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 9, pp. 1927–1941, 2018.

[41] K. Li, J. Zhao, J. Hu, and Y. Chen, "Dynamic energy efficient task offloading and resource allocation for noma-enabled iot in smart buildings and environment," *Building and Environment*, vol. 226, p. 109513, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360132322007430

[42] P. Emberson, R. Stafford, and R. I. Davis, "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)*, 2010, pp. 6–11.

[43] R. Aharoni and O. Kessler, "On a possible extension of hall's theorem to bipartite hypergraphs," *Discrete Mathematics*, vol. 84, no. 3, pp. 309–313, 1990. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0012365X90901366

[44] L. Lovász and M. D. Plummer, *Matching theory*. American Mathematical Soc., 2009, vol. 367.