Joint Task Allocation and Scheduling for Multi-Hop Distributed Computing

Ke Ma
Department of Electrical and Computer Engineering
University of California, San Diego and San Diego State University
La Jolla, USA
kem006@ucsd.edu

Junfei Xie
Department of Electrical and Computer Engineering
San Diego State University
San Diego, USA
jxie4@sdsu.edu

ICC 2024 - IEEE International Conference on Communications | 978-1-7281-9054-9/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICC51166.2024.10622383

***Abstract*—The rise of edge computing has shifted computing resources closer to end-users, benefiting numerous delay-sensitive, computation-intensive applications. To speed up computation, distributed computing is a promising technique that allows parallel execution of computation tasks across multiple compute nodes. However, current research predominantly revolves around the master-worker paradigm, limiting resource sharing within one-hop neighborhoods. This limitation can render distributed computing ineffective in scenarios with limited nearby resources or constrained/dynamic connectivity. In this paper, we address this limitation by introducing a new distributed computing strategy that extends resource sharing beyond one-hop neighborhoods through exploring layered network structures and multi-hop routing. Our approach involves transforming the network graph into a sink tree and solving a joint optimization problem formulated based on the layered tree structure for task allocation and scheduling. Simulation results demonstrate a significant improvement over the traditional distributed computing and computation offloading strategies.**

***Index Terms*—Edge computing, Distributed computing, Multi-hop offloading**

I. Introduction

The proliferation of the Internet of Things (IoT) devices has enabled a multitude of delay-sensitive yet computation-intensive applications, such as monitoring, logistic management, and automative [1]. The surge of these applications drives the migration of computing resources from the remote cloud to the network edge closer to end-users. While computing at the network edge offers compelling benefits, such as low latency, cost effectiveness, and improved data control and security, it also presents notable challenges. The distributed nature of edge servers, along with their inherent constraints in computing power, memory capacity, and available bandwidth when compared to the cloud, pose significant challenges to achieving high-performance edge computing [2].

To speed up computation, distributed computing can be employed. Existing distributed computing strategies typically adopt a master-worker paradigm [3], where a single master node partitions and distributes the task to multiple worker nodes that are directly connected to it. Although the master-worker paradigm is simple to implement, it restricts resource sharing within one-hop neighborhoods. In the edge computing paradigm with constrained computing nodes, scenarios may happen where the residual resources at nearby edge servers are very limited or even less than those at the master node, rendering such master-worker based distributed computing ineffective. This challenge becomes particularly pronounced in edge networks with restricted and/or dynamic connectivity, such as networked airborne computing systems comprised of drones serving as edge servers [4], [5].

In this paper, we overcome these challenges by exploring resources at distant servers located multiple hops away. While a similar idea has been explored in the mobile edge computing (MEC) domain, where computation offloading is proposed to address users' computing demands, most existing studies focus on offloading to a single MEC server [6]–[8]. A few recent works [9]–[11] considered offloading tasks to multiple servers, but they differ in their objectives and have overlooked the task scheduling problem considered in this work.

The main contribution of this paper is a new distributed computing strategy that explores layered network structures and multi-hop routing to fully utilize the capacity of the entire edge computing network for enhanced system performance. Our strategy transforms the network graph into a sink tree, based on which a mixed integer programming (MIP) problem is then formulated and solved to jointly optimize computation efficiency and energy consumption by addressing task allocation and scheduling simultaneously. To evaluate the performance of the proposed approach, we conduct comprehensive simulation studies. Our results demonstrate a significant improvement over the traditional master-worker paradigm and state-of-the-art computation offloading strategies. Notably, the proposed approach can be applied to any networked computing system such as cloud computing, MEC, and networked airborne computing systems.

The rest of the paper is organized as follows. Section II discusses related works. Sec. III describes the system model and the problem to be solved. Sec. IV introduces the proposed approach and Sec. V conducts simulation studies. Finally, Sec. VI presents conclusions and future works.

II. Related works

In the field of distributed computing, the master-worker paradigm has been widely used to implement parallel applications [3]. Another popular paradigm is the hierarchical master-worker paradigm [12], which involves a supervisor process

managing multiple sets of processes, each consisting of a master process and multiple worker processes. Differing from these paradigms, we investigate a multi-layer master-worker paradigm that is composed of a single master and multiple workers operating at different layers.

In the computing offloading domain, most existing studies consider a single-hop single-server offloading paradigm, where tasks are offloaded from users to a single edge server within their communication range [6]–[8]. Under this paradigm, many algorithms have been designed to make the optimal offloading decisions, commonly coupled with optimal allocation of resources such as transmission power and computing resources. As the resources of a single server are bounded, this paradigm is not scalable to the number of users.

To overcome the limitations of the single-hop paradigm, multi-hop offloading [9]–[11] has been proposed, enabling the offloading of tasks from users to remote servers. For instance, [9] investigates the joint routing and multi-part offloading for both data and result. It employs a flow model to capture data/result traffic and introduces a distributed algorithm that finds optimal solutions in polynomial time. [10] formulates the multi-hop offloading problem as a potential game. By dividing tasks subtasks of equal size, each device independently decides the number of subtasks to forward or compute based on its economic utility. Another relevant work is presented in [11], which considers a joint user association, channel allocation, and task offloading problem. It solves this problem by combining the genetic algorithm and deep deterministic policy gradient algorithm. Distinct from previous research, we delve into the essential benefits of multi-hop routing and multi-part offloading while investigating how network properties like topology and server resources affect system performance. We also address the task scheduling problem that arises when transmissions of subtasks share channels or relays, which has been overlooked by existing works.

III. System Model and Problem Description

In this section, we first present the system model for the network and then describe the problem to be solved.

A. System Model

Consider a network formed by $N+1$ edge servers, each with its own unique set of computing and communication capabilities. The servers can share resources with their neighbors through cables (in wired networks) or when they are within communication range (in wireless networks). The entire system is supervised and managed by a control center (e.g., a software defined networking controller [13]) to ensure that all tasks are completed efficiently and effectively. Suppose one of the servers, referred to as *master*, needs to execute a computation-intensive task that is arbitrarily decomposable, which could be generated by the server itself or requested by a user nearby. To complete the task in a timely and energy-efficient manner, the master decomposes the task into subtasks and distributes them to the other servers, referred to as *workers*. The master can transmit subtasks simultaneously to their neighboring servers. In the context of wireless communication, we assume the use of the Orthogonal Frequency Division Multiple Access technology [11]. However, for workers farther away, multi-hop routing is required, which means that each server in the network can act as a worker, a relay, or both. When a subtask arrives at a relay, it is added to a queue and processed in a first-in-first-out order. A worker will not start executing the assigned subtask until it receives the complete subtask package. When a server acts as both a worker and a relay, it can perform the relay process and execute the assigned task simultaneously.

In this preliminary study, we adopt several common assumptions made in existing studies [14], [15] to simplify our analysis. In particular, we assume that the network is stable with no package losses or retransmissions. Additionally, we assume that the computation result is relatively small, and hence the delay incurred in transmitting the result from workers back to the master is negligible. Under these assumptions, we model the network as follows.

1) Network Model: The network is modeled as a directed graph $\mathcal{G} = \{\mathcal{N}, \mathcal{E}\}$, where $\mathcal{N} = \{i | 0 \leq i \leq N\}$ is the set of edge servers and $\mathcal{E} = \{(i,j) | i,j \in \mathcal{N}, i \neq j\}$ is the set of server-to-server communication links that connect servers that can communicate directly.

2) Computing model: Let f_i denote the computing capacity of server i, i.e., CPU-cycle frequency (GHz). Given a task of size y, let b denote the total number of CPU cycles required to process one task size unit. The time required for server i to process this task can then be expressed by [16]:

$$T_i^{comp} = \frac{yb}{f_i} \tag{1}$$

3) Communication Model: Let $B_{i,j}$ denote the bandwidth of the communication link (i,j), which is assumed to be known. The data transmission rate from server i to server j can be expressed by the following equation according to Shannon's Theory [5]:

$$R_{i,j} = B_{i,j} log_2(1 + \frac{s_{i,j}}{n_{i,j}}) \tag{2}$$

where $s_{i,j}$ and $n_{i,j}$ represent the signal power and noise power, respectively.

4) Energy Consumption Model: The energy consumed for executing a task mainly constitutes two components: energy consumed for computing and energy consumed for communication. The energy consumed for server i to compute a task of size y is given by [17]:

$$E_i^{comp}(y) = \gamma_i y b (f_i)^2 \tag{3}$$

where γ_i is the effective switched capacitance that depends on the chip architecture of server i. The energy consumed for server i to transmit a task of size y to server j is given by [17]:

$$E_{i,j}^{comm}(y) = \frac{p_i y}{R_{i,j}} \tag{4}$$

B. Problem Description and Analysis

Without loss of generality, suppose server $i = 0$ receives a task of size $Y \in \mathrm{R}^+$ to complete. Given the computing and communication characteristics of the whole network, i.e., $\mathcal{G}$, $\{f_i, B_{i,j}, s_{i,j}, n_{i,j}, p_i\}, \forall i, j \in \mathcal{N}$ are known, the control center aims to jointly minimize the task completion time and energy consumption by partitioning the task into small subtasks and distributing them to other servers in the network.

Finding the optimal solution to this problem is nontrivial and challenging since it requires making decisions on several aspects, including identifying which servers the master should assign subtasks to, determining the amount of workload to be assigned to each worker, and selecting the transmission route for sending the subtask. Moreover, the order in which the subtasks should be sent by the master is also a crucial decision to make.

IV. Multi-Hop Multi-Layer Distributed Computing Strategy

In this section, we present a multi-hop multi-layer distributed computing strategy to solve the problem described in the previous section. Motivated by the fact that a layered tree structure emerges when the master distributes tasks to other servers in the network, this strategy first transforms the network graph into a sink tree and exploits this layered tree structure to find optimal task allocation and scheduling solutions.

A. Transforming Graph into a Sink Tree

Given the network graph $\mathcal{G}$ and the characteristics of the servers and communication links forming the graph, we can find the shortest route from the master to each of the other servers in the network that takes the minimum time to transmit one bit of data. This can be achieved by defining the weight of each edge (i, j) as the inverse of the associated data transmission rate, i.e., $1/R_{i,j}$, and applying the Dijkstra's algorithm [18] to find the shortest path. The resulting shortest routes can then be used to construct a K-ary sink tree, where the master is the root node and all other servers are leaf or internal nodes reachable from the root via a unique path. This layered tree structure enables the distribution of tasks from the master to other servers in an efficient manner.

To facilitate subsequent analysis, we re-label the nodes in the tree level-by-level from the root downward, and from left to right within each level (see Fig. 1). Consequently, nodes in lower levels have larger indices. Let $\mathcal{I}_l$ denote the set of indices of nodes in level $l \in \{0, 1, \ldots, L\}$, where L is the height of the tree. Then, $\cup_{l=0}^{L} \mathcal{I}_l = \mathcal{N}$. Notably, the master (root) can transmit subtasks to its one-hop neighbors, i.e., nodes in Level 1, simultaneously using orthogonal channels. However, if any one-hop neighbor has children, the subtasks assigned to them, including the one-hop neighbor, have to be transmitted one by one. This is because they share the same channel between the master and the one-hop neighbor, and the data arriving at the one-hop neighbor is processed in a first-in-first-out manner. Therefore, the order in which these subtasks should be sent matters. Based on these analyses, we next formulate a joint task allocation and scheduling problem as a mixed integer programming (MIP) model.

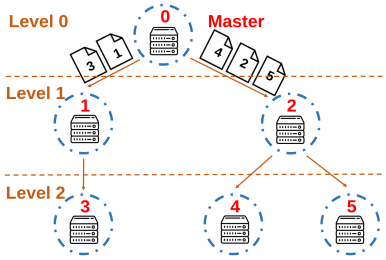


Fig. 1: An example network represented by a layered tree structure. Servers' indices are highlighted in red.

B. Mixed Integer Programming Model

1) Decision Variables: To specify the computation load allocated to each server $i \in \mathcal{N}$, we introduce decision variables $\boldsymbol{y} = \{y_0, y_1, \ldots, y_N\}$, where $y_i \in [0, Y]$, representing the size of the subtask assigned to server i. If $y_i = 0$, it implies that server i is not assigned any workload. Note that the master may choose to execute (part of) the task locally, in which case y_0 would be nonzero.

To describe the sending order for subtasks transmitted from the master to the other servers, we introduce decision variables $\boldsymbol{o} = \{o_1, o_2, \ldots, o_N\}$, where $o_i \in \mathcal{N} \setminus \{0\}, \forall i \in \mathcal{N} \setminus \{0\}$ and $o_i \neq o_j, \forall i, j \in \mathcal{N} \setminus \{0\}, i \neq j$. When $o_i > o_j$, server i has a higher priority than server j to receive its subtask, where $i, j \in \mathcal{N} \setminus \{0\}$ and $i \neq j$.

2) Objective Function: We aim to achieve two objectives simultaneously: minimize the time spent and minimize the energy consumed by each server for executing the task. By employing a weighted sum method, we define the objective function as follows:

$$J(\boldsymbol{y}, \boldsymbol{o}) = \max_{i \in \mathcal{N}} w_1 T_i^{total}(\boldsymbol{y}, \boldsymbol{o}) + w_2 E_i^{total}(\boldsymbol{y}) \tag{5}$$

where $w_1, w_2 \geq 0$ are the weights, representing the relative importance of the two objectives. $T_i^{total}(\boldsymbol{y}, \boldsymbol{o})$ is the total time required for server i to receive its subtask from the master and complete the assigned subtask. Note that the time required for completing the whole task is $\max T_i^{total}(\boldsymbol{y}, \boldsymbol{o})$, $\forall i \in \mathcal{N}$. $E_i^{total}(\boldsymbol{y})$ is the total energy consumed by server i during task execution. Next, we derive the formulas for $T_i^{total}(\boldsymbol{y}, \boldsymbol{o})$ and $E_i^{total}(\boldsymbol{y})$.

3) Time Consumption: The task completion time $T_i^{total}(\boldsymbol{y}, \boldsymbol{o})$ is comprised of three components: 1) time taken to transmit subtask of size y_i from the master to server i, denoted as $T_i^{tran}(y_i)$; 2) time spent waiting in the queues of relays along the path to server i if any, denoted as $T_i^{wait}(\boldsymbol{y}, \boldsymbol{o})$; and 3) time to execute the subtask, i.e., $T_i^{comp}(y_i)$. It is noted that the waiting time $T_i^{wait}(\boldsymbol{y}, \boldsymbol{o})$ is impacted by the task sizes assigned to other servers and the sending order, which complicates the optimization problem considered in this study.

To obtain the transmission time $T_i^{tran}(y_i)$, we introduce the notation $\mathcal{P}_i$ to denote the sequence of servers that lie on the path from the master to server i, and the notation p_k to denote

the k-th server in the sequence, where $1 \leq k \leq |\mathcal{P}_i|$, $p_1 = 0$ and $p_{|\mathcal{P}_i|} = i$. T_i^{tran} can then be expressed by:

$$T_i^{tran}(y_i) = \begin{cases} 0, & \text{if } i = 0 \\ \sum_{k=1}^{|\mathcal{P}_i|-1} \frac{y_i}{R_{p_k,p_{k+1}}}, & \text{else} \end{cases} \tag{6}$$

Let's now consider the waiting time $T_i^{wait}(\boldsymbol{y}, \boldsymbol{o})$. Let a_i denote the ancestor of server i at Level 1, i.e., $a_i \in \mathcal{I}_1$. In the special case where server i is at Level 1, we have $i = a_i$. Additionally, let $\mathcal{A}_i$ denote the full set of servers in the subtree of the master with a_i as the root of the subtree. Note that $i \in \mathcal{A}_i$. Additionally, define $\mathcal{B}_i = \{j | o_j > o_i, j \in \mathcal{A}_i, i \neq j\}$ as the set of servers whose subtasks will be transmitted before server i. Note that if $j \notin \mathcal{A}_i$, the subtask for server j is transmitted using a different channel that is orthogonal to the one used for server i, and hence server i does not need to wait for server j's subtask to be transmitted even if $o_j > o_i$. Based on these definitions, we can then express the waiting time as follows:

$$T_i^{wait}(\boldsymbol{y}, \boldsymbol{o}) = \begin{cases} 0, & \text{if } i = 0 \text{ or } \mathcal{B}_i = \emptyset \\ \sum_{j \in \mathcal{B}_i} \sum_{k=1}^{|\mathcal{P}_{i,j}|-1} \frac{y_j}{R_{p_k,p_{k+1}}}, & \text{else} \end{cases} \tag{7}$$

where $\mathcal{P}_{i,j} = \mathcal{P}_i \cap \mathcal{P}_j$.

Based on (1), (6), and (7), we then have

$$T_i^{total}(\boldsymbol{y}, \boldsymbol{o}) = T_i^{trans}(y_i) + T_i^{wait}(\boldsymbol{y}, \boldsymbol{o}) + T_i^{comp}(y_i) \tag{8}$$

4) Energy Consumption: With $T_i^{total}(\boldsymbol{y}, \boldsymbol{o})$ and (3)-(4), the energy consumption $E_i^{total}(\boldsymbol{y})$ can then be expressed by:

$$E_i^{total}(\boldsymbol{y}) = E_i^{comp}(y_i) + \sum_{j \in \mathcal{C}_i} E_{i,j}^{comm}(y_j) \tag{9}$$

In the above equation, $\mathcal{C}_i$ is the set of children of server i, whose subtasks will be relayed by server i.

5) Problem Formulation: Mathematically, the multi-objective optimization problem can be formulated as follows:

$$\begin{aligned} \mathcal{P}_0: \ & \min_{\boldsymbol{y},\boldsymbol{o}} J(\boldsymbol{y}, \boldsymbol{o}) \\ s.t. \ & \sum_{i=0}^{N} y_i = Y & C1 \\ & 0 \leq y_i \leq Y, \forall i \in \mathcal{N} & C2 \\ & o_i \in \mathcal{N} \setminus \{0\}, \forall i \in \mathcal{N} \setminus \{0\} & C3 \\ & o_i \neq o_j, \forall i, j \in \mathcal{N} \setminus \{0\}, i \neq j & C4 \end{aligned}$$

C. Solution

To solve problem $\mathcal{P}_0$, we first transform it into a mixed integer linear programming problem by introducing an auxiliary variable z as follows:

$$\begin{aligned} \mathcal{P}_1: \ & \min_{\boldsymbol{y},\boldsymbol{o},z} \quad z \\ s.t. \ & z \geq w_1 T_i^{total}(\boldsymbol{y}, \boldsymbol{o}) + w_2 E_i^{total}(\boldsymbol{y}), \forall i \in \mathcal{N} \\ & C1 - C4 \end{aligned}$$

Problem $\mathcal{P}_1$ can be further decomposed into two subproblems. The first subproblem aims to optimize the task allocation $\boldsymbol{y}$, given a particular sending order denoted as $\boldsymbol{o} = \boldsymbol{o}_k$:

$$\begin{aligned} \mathcal{P}_1^{(a)}: \ & \min_{\boldsymbol{y},z} \quad z \\ s.t. \ & z \geq w_1 T_i^{total}(\boldsymbol{y}, \boldsymbol{o}) + w_2 E_i^{total}(\boldsymbol{y}), \forall i \in \mathcal{N} \\ & C1 - C2 \end{aligned}$$

Denote the optimal solution to problem $\mathcal{P}_1^{(a)}$ at $\boldsymbol{o} = \boldsymbol{o}_k$ as $\{\boldsymbol{y}^*(\boldsymbol{o}_k), z^*(\boldsymbol{o}_k)\}$. The second subproblem aims to optimize the sending order, given $\boldsymbol{y} = \boldsymbol{y}^*(\boldsymbol{o}_k)$ and $z = z^*(\boldsymbol{o}_k)$:

$$\mathcal{P}_1^{(b)}: \ \min_{\boldsymbol{o}_k} \quad z^*(\boldsymbol{o}_k)$$

Now let's consider subproblem $\mathcal{P}_1^{(a)}$, which can be solved using Lagrange multipliers [19]. Particularly, the Lagrangian function can be defined as follows:

$$\mathcal{L}(\boldsymbol{y}, z, \boldsymbol{\lambda}, \mu) = z + \sum_{i=0}^{N} \lambda_i \left[c_i(\boldsymbol{y}) - z\right] + \mu \left(\sum_{i=0}^{N} y_i - Y\right).$$

where $c_i(\boldsymbol{y}) = w_1 T_i^{total}(\boldsymbol{y}, \boldsymbol{o}_0) + w_2 E_i^{total}(\boldsymbol{y}, \boldsymbol{o}_0)$. $\boldsymbol{\lambda} = \{\lambda_i\}_{i=0}^N$ and μ are Lagrangian multipliers. $\lambda_i \geq 0$, $\forall i \in \mathcal{N}$. Define

$$g(\boldsymbol{\lambda}, \mu) = \min_{\boldsymbol{y},z} \mathcal{L}(\boldsymbol{y}, z, \boldsymbol{\lambda}, \mu)$$

The dual optimization problem is then constructed as follows:

$$\begin{aligned} \max_{\boldsymbol{\lambda},\mu} \quad & g(\boldsymbol{\lambda}, \mu) \\ s.t. \quad & \boldsymbol{\lambda} \geq 0 \end{aligned} \tag{10}$$

As the objective function and the inequality constraints in our problem are convex, and the equality constraints are affine and strictly feasible, Slater's condition [20] is satisfied and the strong duality holds. That means $p^* = d^*$, where p^* and d^* are the optimal values of the primal problem $\mathcal{P}_1^{(a)}$ and the dual problem (10), respectively. The optimal solution to $\mathcal{P}_1^{(a)}$ can then be found by applying the Karush-Kuhn-Tucker (KKT) conditions [21] as follows:

$$\begin{aligned} & \tfrac{\partial}{\partial y_i} \mathcal{L}(\boldsymbol{y}, z, \boldsymbol{\lambda}, \mu) = 0, \ \forall i \in \mathcal{N} \\ & \tfrac{\partial}{\partial z} \mathcal{L}(\boldsymbol{y}, z, \boldsymbol{\lambda}, \mu) = 0 \\ & \textstyle\sum_{i=0}^{N} y_i = Y \\ & \lambda_i (c_i(\boldsymbol{y}) - z) = 0, \ \forall i \in \mathcal{N} \\ & c_i(\boldsymbol{y}) - z \leq 0, \ \forall i \in \mathcal{N} \\ & \lambda_i \geq 0, \ \forall i \in \mathcal{N} \end{aligned} \tag{11}$$

Now we have a method for determining the optimal solution to problem $\mathcal{P}_1^{(a)}$. Problem $\mathcal{P}_1^{(b)}$ can be solved using exhaustive search. However, as $\boldsymbol{o}$ can take $N!$ possible values, evaluating each possible value is time-consuming. A significant reduction in the number of possible values to evaluate can be achieved by exploiting the parallelism in sending subtasks belonging to different subtrees of the master. Specifically, the sending orders for servers in any subtree $\mathcal{A}_i$ are independent of those in any other subtree $\mathcal{A}_j$, where $i, j \in \mathcal{I}_1$ and $i \neq j$. Therefore, the number of possible values of $\boldsymbol{o}$ that need to be evaluated can be reduced to $\prod_{i \in \mathcal{I}_1} |\mathcal{A}_i|!$.

To further improve efficiency, we apply a greedy search that evaluates the sending orders for the subtrees of the master one by one and selects the best order for each subtree at the moment. By doing so, the search space is further reduced to $\sum_{i\in\mathcal{I}_1}|\mathcal{A}_i|!$. As we will demonstrate in the following section, although this strategy causes our approach to lose optimality, the resulting solution is comparable to the optimal solution.

V. Simulation Studies

In this section, we conduct simulation studies to evaluate the performance of the proposed multi-hop multi-layer distributed computing strategy.

A. Experiment Setting

We evaluate our proposed approach on five different network topologies, each transformed into a tree with varying depths and breadths as shown in Fig. 2. In each network topology, we configure the computing capacities f_i of the servers by randomly generating values from the range of $[1.4, 2.6]$MHz. The values of the data transmission rates R_{ij} fall within the range of $[40, 50]$Mbps. Moreover, we set $\gamma_i = 10^{-27}$, and $p_i = 30$dBm for $\forall i, j \in \mathcal{N}, i \neq j$. The task size is set to $Y = 1$ Mbits and $b = 1000$ cycles/Mbit.

For comparison, we implement four state-of-the-art distributed computing and computation offloading schemes as benchmarks.

- **Local computing (Local)**: In this approach, the master executes the entire task locally.
- **Partial offloading (Partial)**: In this approach, the master offloads part of the task to one of its one-hop neighbors. The offloading ratio and offloadee selection are optimized to minimize the task completion time.
- **Master-worker distributed computing (Master-worker)**: In this approach, the master distributes the task to its one-hop neighbors using the master-worker paradigm. The task allocation is optimized to minimize the task completion time.
- **Multi-hop offloading (Multi-hop)** [22]: In this approach, the master offloads the whole task to the most powerful and reliable server in the network, which may be multiple hops away.

B. Experiment Results

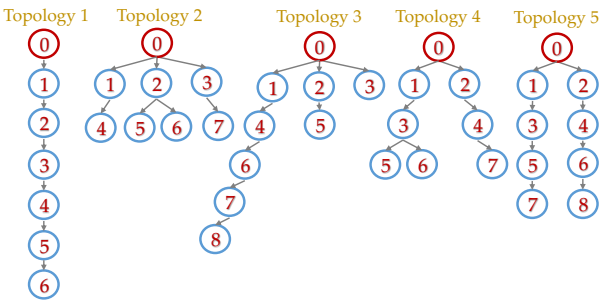
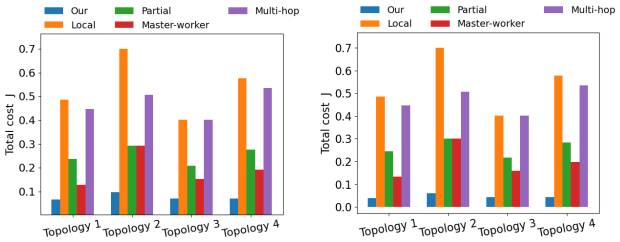
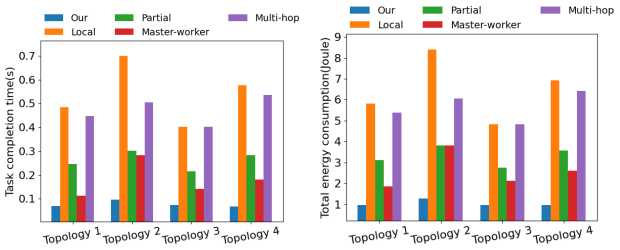


Fig. 2: Network topologies evaluated in simulation studies.



(a) (b)

Fig. 3: Total cost J of different methods when (a) $w_1 = 1$, $w_2 = 0$; and (b) $w_1 = 0.5$, $w_2 = 0.0416$.



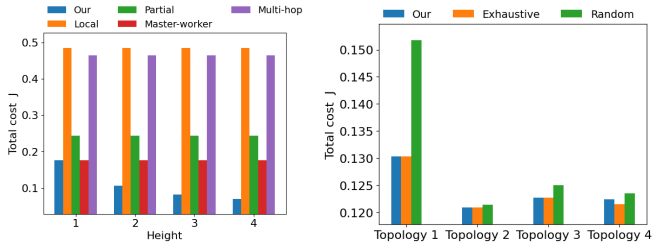
(a) (b)

Fig. 4: (a) Task completion time and (b) maximum energy consumption of different methods when $w_1 = 0.5$, $w_2 = 0.0416$.

of our approach to minimize the task completion time only, just like the benchmarks. As shown in Fig. 3a, our approach outperforms all benchmarks across all scenarios. Among the benchmarks, multi-hop offloading and local computing have the poorest performance since they only use the computing resources from a singe server. Partial offloading outperforms local computing and multi-hop offloading by utilizing the resources from two servers. The master-worker distributed computing achieves even better performance by utilizing computing resources from all servers within one hop. This experiment provides evidence that increasing the utilization of resources leads to better computing performance.

In the second experiment, we randomly set the weights to $w_1 = 0.5$ and $w_2 = 0.0416$, so that both computation efficiency and energy consumption are considered in our approach. Note that these weight values are also used in the following experiments. Fig. 3b shows the comparison results, which demonstrate the promising performance of our approach. In Fig. 4a and Fig. 4b, we also show the task completion time, i.e., $\max_{i\in\mathcal{N}} T_i^{total}$, and the maximum energy consumption by any server, i.e., $\max_{i\in\mathcal{N}} E_i^{total}$, respectively. The results indicate that, despite the trade-off between task completion time and energy consumption in our approach, it still outperforms all benchmarks when each aspect is considered separately.

To understand the impact of the height of the topology tree, we evaluate the performance of different methods on Topology 5 (see Fig. 2). The height of the tree is varied by removing nodes from lower levels. The results, shown in Fig. 5a, reveal that the advantage of our approach becomes more apparent as the height increases. However, when the height exceeds a certain threshold, the performance converges or even degrades



(a) (b)

Fig. 5: Performance comparison of (a) different methods as the height of the topology tree increases; and (b) different methods for optimizing the sending order $\mathbf{o}$.

due to the cost of offloading tasks to servers that are too far away outweighing the benefits in terms of time or energy savings. Notably, the performance of all benchmarks, except for multi-hop offloading, remains constant as the height increases as they only consider servers within one hop.

In the last experiment, we evaluate the performance of our greedy search strategy for determining the sending order $\boldsymbol{o}$ by comparing it with 1) the exhaustive search method that examines all possible sending orders and 2) the random selection method that randomly picks a value for $\boldsymbol{o}$. The results, shown in Fig. 5b, demonstrate that our method achieves comparable performance to the exhaustive search algorithm.

VI. Conclusion and Future Works

This paper presents a new distributed computing strategy that leverages layered network structures and multi-hop routing to enable resource sharing beyond one-hop neighborhoods, effectively utilizing the resources of the entire edge computing system. To jointly optimize task allocation and scheduling, we formulate a MIP problem and derive its optimal solution using Lagrange multipliers. Comprehensive simulation studies demonstrate the promising performance of our approach, outperforming the state-of-the-art distributed computing and computation offloading schemes. Our future research will extend this work to consider multi-user, multi-task scenarios as well as dynamic and mobile networks. We will also investigate the hierarchical master-work paradigm.

Acknowledgment

We would like to thank the National Science Foundation under Grant CAREER-2048266 and CCRI-1730675 for the support of this work.

References

[1] S. Chen, H. Xu, D. Liu, B. Hu, and H. Wang, "A vision of iot: Applications, challenges, and opportunities with china perspective," *IEEE Internet of Things journal*, vol. 1, no. 4, pp. 349–359, 2014.

[2] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, 2016.

[3] J Linderoth, J. Goux, and M Yoder, "Metacomputing and the master-worker paradigm," *Mathematics and Computer Science Division, Argonne National Laboratory, Tech. Rep. ANL/MCS-P792–0200*, 2000.

[4] K. Lu, J. Xie, Y. Wan, and S. Fu, "Toward uav-based airborne computing," *IEEE Wireless Communications*, vol. 26, no. 6, pp. 172–179, 2019.

[5] H. Zhang, B. Wang, R. Wu, *et al.*, "Exploring networked airborne computing: A comprehensive approach with advanced simulator and hardware testbed," *Unmanned Systems*, 2023.

[6] 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. 102 225, 2021.

[7] M. A. Hossain, W. Liu, and N. Ansari, "Computation-efficient offloading and power control for mec in iot networks by meta reinforcement learning," *IEEE Internet of Things Journal*, 2024.

[8] D. Xu, "Device scheduling and computation offloading in mobile edge computing networks: A novel noma scheme," *IEEE Transactions on Vehicular Technology*, 2024.

[9] J. Zhang, Y. Liu, and E. Yeh, "Result and congestion aware optimal routing and partial offloading in collaborative edge computing," *arXiv preprint arXiv:2205.00714*, 2022.

[10] J. Xie, Y. Jia, W. Wen, Z. Chen, and L. Liang, "Dynamic d2d multihop offloading in multi-access edge computing from the perspective of learning theory in games," *IEEE Transactions on Network and Service Management*, vol. 20, no. 1, pp. 305–318, 2022.

[11] H. Zhang, Y. Yang, B. Shang, and P. Zhang, "Joint resource allocation and multi-part collaborative task offloading in mec systems," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 8, pp. 8877–8890, 2022.

[12] K. Aida, W. Natsume, and Y. Futakata, "Distributed computing with hierarchical master-worker paradigm for parallel branch and bound algorithm," in *The 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid*, IEEE, 2003, pp. 156–163.

[13] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2014.

[14] 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. DOI: 10.1109/JSAC.2018.2864426.

[15] H. Li, K. Ota, and M. Dong, "Learning iot in edge: Deep learning for the internet of things with edge computing," *IEEE network*, vol. 32, no. 1, pp. 96–101, 2018.

[16] J. Chen and J. Xie, "Joint task scheduling, routing, and charging for multi-uav based mobile edge computing," in *ICC 2022 - IEEE International Conference on Communications*, 2022, pp. 1–6. DOI: 10.1109/ICC45855.2022.9839040.

[17] Q. Hu, Y. Cai, G. Yu, Z. Qin, M. Zhao, and G. Y. Li, "Joint offloading and trajectory design for uav-enabled mobile edge computing systems," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1879–1892, 2019. DOI: 10.1109/JIOT.2018.2878876.

[18] E. W. Dijkstra, "A note on two problems in connexion with graphs," in *Edsger Wybe Dijkstra: His Life, Work, and Legacy*, 2022, pp. 287–290.

[19] S. Boyd, *Convex optimization–boyd and vandenberghe*, 2004.

[20] A. Auslender and M. Teboulle, "Lagrangian duality and related multiplier methods for variational inequality problems," *SIAM Journal on Optimization*, vol. 10, no. 4, pp. 1097–1115, 2000.

[21] Z.-Q. Luo and W. Yu, "An introduction to convex optimization for communications and signal processing," *IEEE Journal on selected areas in communications*, vol. 24, no. 8, pp. 1426–1438, 2006.

[22] W. Gao, "Opportunistic peer-to-peer mobile cloud computing at the tactical edge," in *IEEE Military Communications Conference*, IEEE, 2014, pp. 1614–1620.