Multi-Hop Relay-Aided Task Offloading for Tactical Edge Computing

Zhaofeng Zhang, Xuanli Lin, Guoliang Xue, Yanchao Zhang, Kevin S. Chan

***Abstract*—Recent years have witnessed explosive growth in the deployment of IoT devices over tactical edge networks, which demand flexible task offloading strategies to accommodate limited connectivity and computing resources. While many existing works assume direct access to edge servers, such assumptions often break down in adversarial and dynamic environments. This work introduces a novel edge task offloading framework that enables multi-hop communication and enables task execution not only at edge servers but also at relay nodes and local devices. We formulate the problem of joint task offloading, resource allocation, and time frame/slot assignment, under a time-division multiple access (TDMA) uplink scheme, as a nonlinear integer program to maximize total rewards, subject to various communication and resource constraints. To solve it optimally, we first construct a feasibility-aware directed graph that captures valid communication links based on signal-to-noise ratio (SNR) constraints, and then design a dynamic programming (DP) algorithm that integrates slot-minimizing offloading path selection with the assignment of the largest feasible number of time frames for each user to reduce total time slots consumption without violating task completion deadlines. To improve scalability, efficient heuristics are developed using scaling and rounding techniques. Extensive experiments demonstrate that our proposed DP algorithm achieves optimal task offloading and resource allocation, while our heuristic algorithms offer a computationally efficient alternative with competitive performance.**

1. Introduction

Tactical edge networks are increasingly populated with intelligent IoT devices generating computation-intensive data streams under stringent latency and reliability requirements [4], [6]. Nevertheless, these devices' constrained processing power and limited energy reserves present critical challenges, especially in dynamic and adversarial environments where rapid decision-making is essential. While task offloading directly to edge servers has proven effective under resource limitations [5], [7], this approach hinges on the assumption of persistent connectivity and low-latency access to edge servers, which are rarely guaranteed in tactical deployments.

Recent efforts have explored relay-aided offloading frameworks to overcome these limitations, where intermediate relay nodes facilitate communication with distant servers [2]. However, existing approaches often treat relays as passive data forwarders, overlooking their potential to perform computation. This observation motivates a more general and flexible task offloading paradigm incorporating multi-hop relaying and heterogeneous computing at relays and devices. To this end, this work investigates the following fundamental question: *"How can we design flexible resource-aware offloading strategies over multi-hop tactical edge networks where both servers and relays can provide computing services?"*

To answer this, we propose a novel offloading framework where each task may be executed locally, at a relay, or at an edge server—potentially through multi-hop relay paths. The proposed framework incorporates a TDMA-based uplink model in which users are allocated several communication frames and slots to transmit their task data. We formulate the resulting multi-hop relay-aided offloading (**MRO**) problem as a nonlinear integer program constrained by computing and communication resources. To solve it optimally, we design a pseudo-polynomial time dynamic programming (DP) algorithm. Here, to reduce per-frame slot demand while meeting deadline constraints, we adopt a key strategy that assigns each user the largest feasible number of time frames permitted by its task deadline. Another central component of our proposed solution is the construction of a feasibility-aware directed graph, which includes only communication links satisfying signal-to-noise ratio (SNR) constraints. Slot-minimizing offloading paths are then selected over this graph for each user. Moreover, we propose heuristic algorithms using scaling and rounding techniques to improve scalability. The main contributions of this paper are the following:

- We introduce a multi-hop relay-aided task offloading framework and formulate the resulting **MRO** problem as a nonlinear integer program subject to various constraints.
- We design an exact DP algorithm, leveraging minimum-slot time frames assignment and offloading path selection via a feasibility-aware graph, to optimally solve **MRO**.
- We propose efficient heuristics based on scaling and rounding techniques to reduce the time complexity.
- We conduct numerical evaluations and demonstrate that the proposed exact DP algorithm obtains optimal offloading strategies, while the heuristics achieve competitive performance with significantly reduced computation time.

We present the system model in §2 and formulate the **MRO** problem in §3. We present exact and heuristic algorithms in §4 and §5, respectively. We present evaluation results in §6

Zhaofeng Zhang, Xuanli Lin, Guoliang Xue, and Yanchao Zhang are affiliated with Arizona State University and DOD Center of Excellence in FutureG. Email: {xlin54, zzhan199, xue, yczhang}@asu.edu. Kevin S. Chan is affiliated with DEVCOM US Army Research Laboratory. Email: kevin.s.chan.civ@army.mil. The research was sponsored by DEVCOM US Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-23-2-0225. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies either expressed or implied, of DEVCOM Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

and conclude the paper in §7. Due to the limited space, we present all the proofs in the Appendix of the technical report for a better flow of the paper.

2. System Model

We consider a multi-server relay-aided tactical edge network where each edge server and relay node can provide computation offloading services to resource-constrained local mobile users. Assuming there are a set of N users $\mathcal{U} = \{u_1, u_2, ..., u_N\}$, a set of L relay nodes $\mathcal{H} = \{h_1, h_2, ..., h_L\}$, and a set of M servers $\mathcal{S} = \{s_1, s_2, ..., s_M\}$. Typically, each server is endowed with computing resources allocated by the network operator and is capable of wireless communication with mobile users and relays. Compared to edge servers, each relay node has relatively weaker computing capabilities. Local users can offload machine learning (ML) inference tasks to an edge server or a relay node for execution.

A. User Tasks and Local Computing Resources

Each user $u_n \in \mathcal{U}$ has the following attributes:

- *Position*: The user u_n's position is represented by a coordinate $\left(x_n^{\mathcal{U}}, y_n^{\mathcal{U}}\right)$ in two-dimensional Euclidean space.
- *Deadline*: The task's deadline for user u_n is δ_n.
- *Data size*: D_n specifies the input data required, including program codes and parameters, for user u_n's task.
- *Task requirement*: Each user can choose one ML algorithm $k \in \mathcal{K} = \{1, ..., K\}$. The user u_n's task is characterized by a 4-tuple, $\left(\text{CPU}_{n,k}, \text{RAM}_{n,k}, \tau_{n,k}^{\text{exe}}, r_{n,k}\right)$, where $\text{CPU}_{n,k}$ and $\text{RAM}_{n,k}$ are positive integers indicating u_n's required CPU/RAM for processing ML algorithm k, and $\tau_{n,k}^{\text{exe}}$ denotes the algorithm k's execution time of user u_n's task, and $r_{n,k}$ is a positive real number denoting the reward earned when the task is completed before the deadline with algorithm k selected.
- CPU/RAM capacities: The user u_n's CPU/RAM capacities are denoted by two positive integers $C_n^{\mathcal{U}}$ and $R_n^{\mathcal{U}}$.

B. Offloading Strategies

Let $\mathcal{S}' = \mathcal{S} \cup \{s_0\}$, where s_0 indicates that the task is not offloaded to any edge server for execution. Let $\mathcal{H}' = \mathcal{H} \cup \{h_0\}$, where h_0 indicates that the task is not offloaded to a relay node either as the destination for executing or as an intermediate transfer hub. Let $\mathcal{K}' = \mathcal{K} \cup \{0\}$, where $k = 0$ indicates that the task is not executed at all. We formally define the offloading path for each user as follows:

Definition 1 (Offloading path)**.** *Define the offloading path for each user $u_n \in \mathcal{U}$ as $q_n = (v_{n,1}, v_{n,2}, ..., v_{n,J_n})$, where J_n is a positive integer, and we have*

- *$v_{n,1} = u_n$, i.e., the user is the source node;*
- *If $J_n \geq 3$, $v_{n,j} \in \mathcal{H}$ for $j = 2, ..., J_n - 1$, i.e., all intermediate nodes must be relays;*
- *$\forall J_n$, $v_{n,J_n} \in \mathcal{S} \cup \mathcal{H}$, i.e., the final destination can be an edge server or a relay.*

Let $Q(n)$ be the set of all the paths for the user $u_n \in \mathcal{U}$, and let $\mathcal{Q} = (Q(1), ..., Q(n))$. We define the offloading strategy:

Definition 2 (Offloading strategy)**.** *Define the task offloading strategy, which integrates the selection offloading paths and ML algorithms, as a mapping $\mathcal{A} = (A(1), ..., A(n)) : \mathcal{U} \to \mathcal{Q} \times \mathcal{K}'$. Specifically, for user $u_n \in \mathcal{U}$:*

$$A(n) = \begin{cases} \left(\left(u_n, v_{2,n}, ..., v_{J_n-1,n}, s_{m(n)}\right), k(n)\right) & \text{if } u_n \in \mathcal{U}_1, \\ \left(\left(u_n, v_{2,n}, ..., v_{J_n-1,n}, h_{l(n)}\right), k(n)\right) & \text{if } u_n \in \mathcal{U}_2, \\ \left(\left(u_n, s_{m(n)}\right), k(n)\right) & \text{if } u_n \in \mathcal{U}_3, \\ \left(\left(u_n, h_{l(n)}\right), k(n)\right) & \text{if } u_n \in \mathcal{U}_4, \\ \left((u_n), k(n)\right) & \text{if } u_n \in \mathcal{U}_5, \\ \left((u_n), 0\right) & \text{if } u_n \in \mathcal{U}_6, \end{cases}$$

where $k(n) \neq 0$ means the task is executed by algorithm $k(n)$. Moreover, we have:

- *$\mathcal{U}_1 = \{u_n \in \mathcal{U} | J_n \geq 3, k(n) \neq 0, v_{n,J_n} \in \mathcal{S}\}$, i.e., task is offloaded to server $s_{m(n)}$ via intermediate relays;*
- *$\mathcal{U}_2 = \{u_n \in \mathcal{U} | J_n \geq 3, k(n) \neq 0, v_{n,J_n} \in \mathcal{H}\}$, i.e., task is offloaded to relay $h_{l(n)}$ via intermediate relays;*
- *$\mathcal{U}_3 = \{u_n \in \mathcal{U} | J_n = 2, k(n) \neq 0, v_{n,J_n} \in \mathcal{S}\}$, i.e., task is directly offloaded to server $s_{m(n)}$;*
- *$\mathcal{U}_4 = \{u_n \in \mathcal{U} | J_n = 2, k(n) \neq 0, v_{n,J_n} \in \mathcal{H}\}$, i.e., task is directly offloaded to server $h_{l(n)}$;*
- *$\mathcal{U}_5 = \{u_n \in \mathcal{U} | J_n = 1, k(n) \neq 0\}$, i.e., task is not offloaded and executed locally;*
- *$\mathcal{U}_6 = \{u_n \in \mathcal{U} | J_n = 1, k(n) = 0\}$, i.e., task is dropped.*

C. Communication Model

Since output sizes are small and downlink rates are high, we neglect the downlink delay [1]. We consider a wireless communication system employing time-division multiple access (TDMA) as its multiple access protocol in the uplink. The TDMA scheme separates the total communication time into finite-length frames X, partitioning each frame into T slots. Each time slot's length is z time units. Each user u_n is assigned $X_n \in [0, X]$ time frames, where X_n is an integer. Each user u_n is designated T_n time slots, where $T_n \in [0, T]$ is an integer, to transmit data within a time frame. Consequently, the user could transmit only during this period (assigned time slots) over all frequencies and then wait until its assigned transmission slots are available in the subsequent frame. Let P_n denote the transmit power of user u_n, $\forall u_n \in \mathcal{U}$. Since users alternately employ the communication channel when offloading tasks, the uplink interference between users is effectively alleviated. Thus, if a user is scheduled to offload, we let this user's transmit power be its full budget to maximize its Signal-to-Noise Ratio (SNR):

$$P_n = \begin{cases} P_{\max} & \text{if } u_n \in \bigcup_{i=1}^{4} \mathcal{U}_i, \\ 0 & \text{if } u_n \in \mathcal{U}_5 \cup \mathcal{U}_6. \end{cases} \tag{1}$$

Then, for user $u_n \in \bigcup_{i=1}^{4} \mathcal{U}_i$, given q_n, the SNR between node $v_{n,j}$ and $v_{n,j+1}$, where $j = 1, ..., J_n - 1$, is given by

$$\text{SNR}_{v_{n,j}, v_{n,j+1}} = \frac{P_{\max}}{\nu \left(d(v_{n,j+1}, v_{n,j+1})\right)^{\alpha}}, \tag{2}$$

where ν is the background noise, $d(v_{n,j+1}, v_{n,j+1})$ is the distance between nodes $v_{n,j}$ and $v_{n,j+1}$ on the path q_n,

and $\alpha \in [2,4]$ is the path loss exponent [3]. The effective transmission time between nodes $v_{n,j}$ and $v_{n,j+1}$ on q_n is

$$\tau^{\text{up}}_{v_{n,j},v_{n,j+1}} = \frac{D_n}{W \log_2(1+\text{SNR}_{v_{n,j},v_{n,j+1}})}. \quad (3)$$

D. Server and Relay Computing Resources

Server $s_m \in \mathcal{S}$ and relay $h_l \in \mathcal{H}$ have the following attributes:

- Position: The server s_m's and h_l's positions are represented by coordinates $(x^{\mathcal{S}}_m, y^{\mathcal{S}}_m)$ and $(x^{\mathcal{H}}_l, y^{\mathcal{H}}_l)$ in two-dimensional Euclidean space.
- CPU/RAM capacities: The CPU/RAM capacities of the servers s_m and h_l are denoted by positive integers C_m, R_m, C_l, and R_l.

3. Problem Formulation

Assuming $\mathcal{X} = \{X_1, ..., X_N\}$ and $\mathcal{T} = \{T_1, ..., T_N\}$. Based on the above system model, we formulate the multi-hop relay-aided task offloading (**MRO**) as the following maximization problem to jointly optimize the offloading strategy, resource allocation, and the time frames/slots assignments:

$$\textbf{MRO}: \max_{\mathcal{A},\mathcal{X},\mathcal{T}} \sum_{u_n \in \mathcal{U}/\mathcal{U}_6} r_{n,k(n)}, \quad (4)$$

$$\text{s.t.} \sum_{\substack{u_n \in \mathcal{U}_1 \cup \mathcal{U}_3 \\ m(n)=m}} \text{CPU}_{n,k(n)} \le C^{\mathcal{S}}_m, \ \forall s_m \in \mathcal{S}, \quad (5)$$

$$\sum_{\substack{u_n \in \mathcal{U}_1 \cup \mathcal{U}_3 \\ m(n)=m}} \text{RAM}_{n,k(n)} \le R^{\mathcal{S}}_m, \ \forall s_m \in \mathcal{S}, \quad (6)$$

$$\sum_{\substack{u_n \in \mathcal{U}_2 \cup \mathcal{U}_4 \\ l(n)=l}} \text{CPU}_{n,k(n)} \le C^{\mathcal{H}}_l, \ \forall h_l \in \mathcal{H}, \quad (7)$$

$$\sum_{\substack{u_n \in \mathcal{U}_2 \cup \mathcal{U}_4 \\ l(n)=l}} \text{RAM}_{n,k(n)} \le R^{\mathcal{H}}_l, \ \forall h_l \in \mathcal{H}, \quad (8)$$

$$\text{CPU}_{n,k(n)} \le C^{\mathcal{U}}_n, \ \forall u_n \in \mathcal{U}_5, \quad (9)$$

$$\text{RAM}_{n,k(n)} \le R^{\mathcal{U}}_n, \ \forall u_n \in \mathcal{U}_5, \quad (10)$$

$$\sum_{u_n \in \bigcup_{i=1}^{4} \mathcal{U}_i} T_n \le T, \quad (11)$$

$$0 \le \delta_n - \tau^{\text{exe}}_{n,k(n)}, \ \forall u_n \in \mathcal{U}_5, \quad (12)$$

$$X_n T z \le \delta_n - \tau^{\text{exe}}_{n,k(n)}, \ J_n \ge 2, \quad (13)$$

$$X_n T_n z \ge \sum_{j=1}^{J_n-1} \tau^{\text{up}}_{v_{n,j},v_{n,j+1}}, \ J_n \ge 2, \quad (14)$$

$$\text{SNR}_{v_{n,j},v_{n,j+1}} \ge \beta, \forall j = 1, ..., J_n - 1, J_n \ge 2. \quad (15)$$

Constraints (5)-(10) ensure that the CPU and RAM usage on each edge server, relay, and local device does not exceed what is available. Constraint (11) ensures that the total assigned time slots of scheduled tasks do not exceed the overall available time slots T. Constraints (12) and (13) ensure that each user's task is completed by its deadline. Constraint (14) ensures that for each user offloading its task, the effective transmission time is within its assigned time frames/slots. Constraint (15) ensures that for per-hop communication, the receiver successfully decodes the information transmitted from the transmitter, i.e., the SNR must be higher than β. The above **MRO** problem is a non-linear integer program with coupled computing resource and communication constraints, making it computationally challenging in general. To address this, we aim to design low-complexity solutions that are both computationally efficient and practically implementable while still achieving good performance.

4. An Exact Algorithm

In this section, we design an exact DP algorithm, where a feasible offloading path and time frame assignments with minimum time slot usage are selected for each user, to optimally solve the **MRO** problem.

A. Minimum-slot Assignment

By exploiting the structure of **MRO**, we observe that each feasible offloading path and time frame assignment for a user yields the same reward if the selection of the ML algorithm is fixed. Nevertheless, paths and time frame assignments differ in TDMA time slot consumption due to varying hop-wise transmission times and the same time length of each time frame. That is to say, *selecting a feasible path and time frame assignments with minimum time slot usage enables better utilization of limited resources without harming total rewards*. The following proposition formalizes this observation and justifies its use in our algorithm design.

Proposition 1 (Optimality of minimum-slot assignment)**.** *Restricting offloading strategies and time frame assignments to minimum-slot assignments preserves the optimality of any exact algorithm for solving the **MRO** problem.*

In the following, we construct a feasibility-aware directed graph that captures all valid communication links under SNR constraints and use it to compute slot-minimizing time frame assignments and offloading paths.

1) Feasibility-aware Graph Construction: We construct a directed graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \mathcal{U} \cup \mathcal{H} \cup \mathcal{S}$ and $(v, v') \in \mathcal{E}$ if $\text{SNR}_{v,v'} \ge \beta$. Each edge (v, v') satisfies the *structural rules*: 1) v is a user or a relay, and 2) v' is a relay or a server.

2) Computing Largest Feasible X_n: Given a selected ML algorithm, constraint (13) imposes an upper bound on assigned time frames:

$$X^{\text{ub}}_n = \left\lfloor \frac{\delta_n - \tau^{\text{exe}}_{n,k(n)}}{Tz} \right\rfloor. \quad (16)$$

Note that *there is no benefit in choosing a smaller X_n than this bound, since doing so increases the time slots usage of user u_n.* This is because the time slot usage $T_n = \sum_{j=1}^{J_n-1} \left\lceil \frac{\tau^{\text{up}}_{v_{n,j},v_{n,j+1}}}{X_n z} \right\rceil$ is monotonically decreasing in X_n. Then, edge $(v, v') \in \mathcal{E}$ is annotated with user-specific weights:

$$w_n(v, v') = \left\lceil \frac{D_n}{X^{\text{ub}}_n z \cdot W \log_2(1+\text{SNR}_{v,v'})} \right\rceil, \forall u_n \in \mathcal{U}, \quad (17)$$

which denotes the minimum number of time slots required to transmit task data from node v to v' for user u_n.

3) Slot-minimizing Path Computing: With the above computed edge weights, we leverage Dijkstra's algorithm on the feasibility-aware graph to obtain the slot-minimizing offloading paths. For each user $u_n \in \mathcal{U}$, we restrict the computation to only the reachable destinations in $\mathcal{V}^r(n) \subseteq \mathcal{H} \cup \mathcal{S}$. Here, we slightly abuse the notation without loss of clarity. For each destination $v \in \mathcal{V}^r(n)$, the above computing process yields the slot-minimizing path $q^*_{n,v}$ and its corresponding time slot usage $T^*_{n,v}$. This process is repeated for all users.

B. A Dynamic Programming Approach

We have computed the minimum time slot usage for each feasible user-destination pair. Another key observation is that constraints (5)-(8) consist of $M + L$ two-dimensional non-identical knapsack constraints, and constraint (11) is a single-dimensional knapsack constraint. This inspires us to design a DP algorithm to optimally solve the **MRO** problem. We first introduce the following definitions.

Definition 3 (Partial order relation of sets)**.** *Let $\mathbb{B}$ and $\mathbb{B}'$ be two vectors of $2(M + L) + 1$ dimensions. We say $\mathbb{B} \leq \mathbb{B}'$ if $\mathbb{B}[j] \leq \mathbb{B}'[j], j = 1, 2, \ldots, 2(M + L) + 1$.*

Definition 4 (Complete order relation of sets)**.** *Let $\mathbb{B}$ and $\mathbb{B}'$ be two vectors of $2(M+L)+1$ dimensions such that $\mathbb{B} \neq \mathbb{B}'$. We say $\mathbb{B} \prec \mathbb{B}'$, if there exists an integer $i \in \{1, 2, \ldots, 2(M+L)+1\}$ such that $\mathbb{B}[j] = \mathbb{B}'[j], j = 1, 2, \ldots, i-1$ and $\mathbb{B}[i] < \mathbb{B}'[i]$.*

Definition 5 (Required resources)**.** *If user $u_n \in \mathcal{U}$ chooses ML algorithm $k \in \mathcal{K}$ and offloads its task to server/relay $v \in \mathcal{V}^r(n)$, we define its task's required resources as a three-dimensional vector $\omega_{n,k,v} = \left(CPU_{n,k}, RAM_{n,k}, T^*_{n,v}\right)$. We further define the set of tasks' required resources as $\Omega = \{\omega_{n,k,v} | u_n \in \mathcal{U}, k \in \mathcal{K}, v \in \mathcal{V}^r(n)\}$.*

Definition 6 (All-capacities knapsack)**.** *An all-capacities knapsack is defined by a $(2(M + L) + 1)$-dimensional vector $\mathbb{B}[1 : 2(M + L) + 1]$, where $\mathbb{B}[2m - 1]$ and $\mathbb{B}[2m]$ are residual CPU/RAM at server s_m, $m = 1, 2, ..., M$, $\mathbb{B}[2(M + l) - 1]$ and $\mathbb{B}[2(M + l)]$ are residual CPU/RAM at relay h_l, $l = 1, 2, ..., L$, and $\mathbb{B}[2(M + L) + 1]$ is residual time slots. The elements of this vector take non-negative integer values, where $\mathbb{B}[2m - 1] \leq C_m$, $\mathbb{B}[2m] \leq R_m$, $m = 1, 2, ..., M$, $\mathbb{B}[2(M+l) - 1] \leq C^{\mathcal{H}}_l$, $\mathbb{B}[2(M+l)] \leq R^{\mathcal{H}}_l$, $l = 1, 2, ..., L$, and $\mathbb{B}[2(M + L) + 1] \leq T$. We further define the set of all-capacities knapsacks as $\mathcal{B} = \left\{\mathbb{B} | \mathbb{B} \leq \bar{\mathbb{B}}\right\}$, where $\bar{\mathbb{B}} = \left(C^{\mathcal{S}}_1, R^{\mathcal{S}}_1, ..., C^{\mathcal{S}}_M, R^{\mathcal{S}}_M, C^{\mathcal{H}}_1, R^{\mathcal{H}}_1, ..., C^{\mathcal{H}}_L, R^{\mathcal{H}}_L, T\right)$.*

In the following, we first introduce the all-capacities variant of the **MRO** and how to compute the optimal value of it. Then, we propose the DP algorithm and analyze its complexity.

*1) All-capacities Variant of **MRO**:* Based on the above definitions, we introduce the variant of **MRO** for a user subset $\mathcal{U}(n) = \{u_1, ..., u_n\}$ and a possible all-capacities knapsack $\mathbb{B} \leq \bar{\mathbb{B}}$. Formally, given a set of tasks' required resources Ω and $\mathbb{B} \leq \bar{\mathbb{B}}$, the corresponding all-capacities variant of **MRO** is defined as the following maximization problem:

$$\textbf{MRO}'(n, \mathbb{B}, \Omega): \max_{\mathcal{A},\mathcal{X},\mathcal{T}} \sum_{u_{n'} \in \mathcal{U}(n)/\mathcal{U}_6(n)} r_{n',k(n')}, \tag{18}$$

$$\text{s.t.} \sum_{\substack{u_{n'} \in \mathcal{U}_1(n) \cup \mathcal{U}_3(n) \\ m(n')=m}} \text{CPU}_{n',k(n')} \leq \mathbb{B}[2m-1], \ \forall s_m \in \mathcal{S},$$

$$\sum_{\substack{u_{n'} \in \mathcal{U}_1(n) \cup \mathcal{U}_3(n) \\ m(n')=m}} \text{RAM}_{n',k(n')} \leq \mathbb{B}[2m], \ \forall s_m \in \mathcal{S},$$

$$\sum_{\substack{u_{n'} \in \mathcal{U}_2(n) \cup \mathcal{U}_4(n) \\ l(n')=l}} \text{CPU}_{n',k(n')} \leq \mathbb{B}[2(M+l)-1], \ \forall h_l \in \mathcal{H},$$

$$\sum_{\substack{u_{n'} \in \mathcal{U}_2(n) \cup \mathcal{U}_4(n) \\ l(n')=l}} \text{RAM}_{n',k(n')} \leq \mathbb{B}[2(M+l)], \ \forall h_l \in \mathcal{H},$$

$$\text{CPU}_{n',k(n')} \leq C^{\mathcal{U}}_{n'}, \ \forall u_{n'} \in \mathcal{U}_5(n),$$

$$\text{RAM}_{n',k(n')} \leq R^{\mathcal{U}}_{n'}, \ \forall u_{n'} \in \mathcal{U}_5(n),$$

$$\sum_{u_{n'} \in \bigcup_{i=1}^{4} \mathcal{U}_i(n)} T_{n'} \leq T,$$

$$0 \leq \delta_{n'} - \tau^{\text{exe}}_{n',k(n')}, \ \forall u_{n'} \in \mathcal{U}_5(n),$$

$$X_{n'} T z \leq \delta_{n'} - \tau^{\text{exe}}_{n',k(n')}, \ J_{n'} \geq 2,$$

$$X_{n'} T_{n'} z \geq \sum_{j=1}^{J_{n'}-1} \tau^{\text{up}}_{v_{n',j}, v_{n',j+1}}, \ J_{n'} \geq 2,$$

$$\text{SNR}_{v_{n',j}, v_{n',j+1}} \geq \beta, \forall j = 1, ..., J_{n'} - 1, J_{n'} \geq 2,$$

with the optimal value $\texttt{OPT}(n, \mathbb{B}; \Omega)$. The objective and constraints of the problem in $\textbf{MRO}'(n, \mathbb{B}; \Omega)$ are in a similar spirit as those of the original **MRO**. The only difference is that $\textbf{MRO}'(n, \mathbb{B}; \Omega)$ is dynamically defined based the user subset $\mathcal{U}_n$ for $n = 0, ..., N$ and a all-capacities knapsack $\mathbb{B} \leq \bar{\mathbb{B}}$.

2) Computing $\texttt{OPT}(n, \mathbb{B}; \Omega)$: Assuming $\texttt{OPT}(n', \mathbb{B}'; \Omega)$ is already computed for all $n' < n$ and $\mathbb{B}' \in \mathcal{B}$ in the complete order as Def.4, we can consider an additional task from the user u_n and compute the corresponding $\texttt{OPT}(n, \mathbb{B}; \Omega)$. First, we check if the user u_n's task can be executed locally. If the execution time, CPU and RAM usage on the local device does not exceed what is available, we update $\texttt{OPT}(n, \mathbb{B}; \Omega)$ as

$$\texttt{OPT}(n, \mathbb{B}; \Omega) = \max_k \left\{\texttt{OPT}(n-1, \mathbb{B}; \Omega) + r_{n,k}\right\}.$$

Then, we check if we can offload the user u_n's task to server $s_m \in \mathcal{V}^r(n)$ and processed by ML algorithm k. Define its minimum time slot usage is $T^*_{n,m}$, we update $\hat{\mathbb{B}}$ as

$$\begin{aligned} \hat{\mathbb{B}}[2m-1] &= \mathbb{B}[2m-1] - \text{CPU}_{n,k}; \\ \hat{\mathbb{B}}[2m] &= \mathbb{B}[2m] - \text{RAM}_{n,k}; \\ \hat{\mathbb{B}}[2M+1] &= \mathbb{B}[2M+1] - T^*_{n,m}. \end{aligned} \tag{19}$$

If any of the elements of $\hat{\mathbb{B}}$ is smaller than 0, the value of $\texttt{OPT}(n, \mathbb{B}; \Omega)$ will be set to $\texttt{OPT}(n-1, \mathbb{B}; \Omega)$. This is because the considered knapsacks are too small to contain the user

u_n's task. If all $\hat{\mathbb{B}}$'s elements are non-negative, we compute $\texttt{OPT}(n, \mathbb{B}; \Omega)$ by the Bellman recursive formula:

$$\begin{aligned}&\texttt{OPT}(n, \mathbb{B}; \Omega)\\ =&\max\left\{\texttt{OPT}(n-1, \mathbb{B}; \Omega), \texttt{OPT}(n-1, \hat{\mathbb{B}}; \Omega) + r_{n,k}\right\}.\end{aligned}$$

Finally, we check if we can offload the user u_n's task to server $h_l \in \mathcal{V}^r(n)$ and processed by ML algorithm k. Define its minimum time slot usage is $T^*_{n,l}$, we update $\hat{\mathbb{B}}$ as

$$\begin{aligned}&\hat{\mathbb{B}}[2(M+l))-1] = \mathbb{B}[2(M+l))-1] - \text{CPU}_{n,k};\\ &\hat{\mathbb{B}}[2(M+l))] = \mathbb{B}[2(M+l))] - \text{RAM}_{n,k};\\ &\hat{\mathbb{B}}[2M+1] = \mathbb{B}[2M+1] - T^*_{n,l}. \end{aligned} \tag{20}$$

We can compute $\texttt{OPT}(n, \mathbb{B}; \Omega)$ using the Bellman recursive formula similarly to the case of offloading to servers. The process of computing $\texttt{OPT}(n, \mathbb{B}; \Omega)$ is summarized in Alg.1.

Algorithm 1: $\texttt{OPT}(n, \mathbb{B}; \Omega)$

Output: $\texttt{OPT}(n, \mathbb{B}; \Omega)$: optimal value for first n users under knapsack capacities $\mathbb{B}$

```
value ← OPT(n−1, B; Ω);
m(n) ← 0, l(n) ← 0, k(n) ← 0;
for k ∈ K do
    if CPU_{n,k} > C_n^U or RAM_{n,k} > R_n^U or δ_n > τ^{exe}_{n,k} then
        continue
    temp ← r_{n,k} + OPT(n−1, B; Ω);
    if temp > value then
        value ← temp; k(n) ← k;
for s_m, h_l ∈ V^r(n) do
    for k ∈ K do
        X_n ← ⌊(δ_n − τ^{exe}_{n,k}) / (Tz)⌋;
        Compute edge weights based on (17);
        Compute slot-minimizing path using Dijkstra;
        Update B̂ as (19) or (20);
        if any element in B̂ is negative then
            continue
        temp ← r_{n,k} + OPT(n − 1, B̂; Ω);
        if temp > value then
            value ← temp;
            Update offloading path q(n); k(n) ← k;
Output value
```

3) DP Algorithm: Based on the above discussion, we summarize the proposed optimal DP algorithm in Alg. 2, ensuring optimality over all the users and capacity configurations, where the boundary conditions are

- *Initialization:* $\texttt{OPT}(0, \mathbb{B}; \Omega) = 0, \forall \mathbb{B} \in \mathcal{B}$;
- *Resource exhausted:* $\texttt{OPT}(n, (0, ..., 0); \Omega) = 0, \forall u_n \in \mathcal{U}$.

The following theorem delivers the DP's time complexity.

Theorem 1. *The proposed DP algorithm can solve the* **MRO** *problem in $O(|\mathcal{B}|N(M+L)K)$ time.*

Algorithm 2: $\texttt{DP}(\mathcal{U}, \mathcal{B}; \Omega)$

Output: $\mathcal{A}^*$: optimal offloading strategy; $\mathcal{T}^*$: optimal time slots assignments; $\mathcal{X}^*$: optimal time frames assignments;

```
Set boundary conditions;
for u_n ∈ U do
    for each B ∈ B in its complete order do
        Compute OPT(n, B; Ω);
        Update A(n, B; Ω), T(n, B; Ω), X(n, B; Ω);
Output A*, T*, X*
```

5. Efficient Heuristics

The proposed DP algorithm optimally solves **MRO**. However, as established in Theorem 1, its time complexity is pseudo-polynomial. This becomes computationally expensive when servers/relays have sufficient resources, i.e., when the knapsack capacities are large, leading to a massive size of $\mathcal{B}$ since $|\mathcal{B}| = \prod_{j=1}^{2(M+L)+1}(\bar{\mathbb{B}}[j]+1)$. To address this scalability issue, we propose heuristics based on scaling and rounding techniques to reduce the time complexity.

We first present the following lemma.

Lemma 1. *Given x, y, and $b \in \mathbb{R}$, we have*

$$x + y \le b \Rightarrow \lfloor x \rfloor + \lfloor y \rfloor \le \lfloor b \rfloor, \tag{21}$$

$$\lceil x \rceil + \lceil y \rceil \le \lfloor b \rfloor \Rightarrow x + y \le b. \tag{22}$$

In the following, we present two variants of **MRO** and design the corresponding efficient heuristics based on solving them.

A. Relaxed/Restricted Version of MRO

Based on Lemma 1, we propose the following relaxed and restricted versions of **MRO** by scaling down the CPU/RAM capacities and rounding them to an integer $\lfloor \lambda \rfloor$, where λ is a positive scaling factor.

$$\textbf{P1}\,(\texttt{Relaxed}):\ \textbf{MRO}'\left(N, \bar{\mathbb{B}}_r, \Omega^{\text{relaxed}}\right); \tag{23}$$

$$\textbf{P2}\,(\texttt{Restricted}):\ \textbf{MRO}'\left(N, \bar{\mathbb{B}}_r, \Omega^{\text{restricted}}\right). \tag{24}$$

Here, the maximum knapsacks capacities are characterized by a $(2(M+L)+1)$-dimensional vector $\bar{\mathbb{B}}_r = (\lfloor \lambda \rfloor, ..., \lfloor \lambda \rfloor, T)$. Ω^{relaxed} is the set of required resources after relaxation, where $\omega^{\text{relaxed}}_{n,k,v} = \left(\left\lfloor \lambda \frac{\text{CPU}_{n,k}}{C_v} \right\rfloor, \left\lfloor \lambda \frac{\text{RAM}_{n,k}}{R_v} \right\rfloor, T^*_{n,v}\right)$. $\Omega^{\text{restricted}}$ is the set of required resources after restriction, where $\omega^{\text{restricted}}_{n,k,v} = \left(\left\lceil \lambda \frac{\text{CPU}_{n,k}}{C_v} \right\rceil, \left\lceil \lambda \frac{\text{RAM}_{n,k}}{R_v} \right\rceil, T^*_{n,v}\right)$.

B. Heuristics based on Solving P1 and P2

The proposed relaxed and restricted formulations preserve the structure of the original **MRO** problem. Consequently, both **P1** and **P2** can be solved using the proposed DP algorithms, denoted by $\texttt{DP}(\mathcal{U}, \bar{\mathbb{B}}_r; \Omega^{\text{relaxed}})$ and $\texttt{DP}(\mathcal{U}, \bar{\mathbb{B}}_r; \Omega^{\text{restricted}})$, respectively. In both cases, the size of the knapsack capacity set is given by $|\bar{\mathbb{B}}_r| = (\lfloor \lambda \rfloor + 1)^{2(M+L)}(T+1)$.

Remark 1. *Based on Theorem 1, the proposed DP algorithm solves both **P1** and **P2** in time complexity* $O((\lfloor\lambda\rfloor + 1)^{2(M+L)}(T+1)N\,(M+L)\,K)$. *Moreover, the following inequality holds for the corresponding optimal values:*

$$\texttt{OPT}(\mathcal{U}, \bar{\mathbb{B}}_r; \Omega^{\textit{restricted}}) \leq \texttt{OPT}(\mathcal{U}, \bar{\mathbb{B}}; \Omega) \leq \texttt{OPT}(\mathcal{U}, \bar{\mathbb{B}}_r; \Omega^{\textit{relaxed}}).$$

This implies that choosing a smaller λ allows computing faster, potentially at the cost of under- or over-approximating the total earned reward. Therefore, an appropriate choice of λ can effectively balance computational efficiency and approximation accuracy.

6. Performance Evaluation

A. Evaluation Setup

We generate the following four types of test cases with different sizes: `small` with 5 users, 1 servers, 2 relays, and 2 ML algorithms; `medium` with 10 users, 2 servers, 2 relays, and 2 ML algorithms; `med-large` with 12 users, 2 servers, 3 relays, and 2 ML algorithms; and `large` with 15 users, 2 servers, 3 relays, and 2 ML algorithms. We set maximum transmit power to $P_{\max} = 10$ W. The total frequency bandwidth is $B = 40$ MHz; The background noise is $\nu = -65$ dBm. The path loss exponent is $\alpha = 3$. The SNR threshold β is set to 15 dB for all test cases. We assume the total time slots $T = 40$, each time slot's length $z = 0.025s$. Lastly, we assume the scaling factor $\lambda = 3$.

All the experiments are conducted on a workstation running Ubuntu 20.04 with Intel Xeon Gold 6226R CPU (64 cores @ 2.90 GHz) and 512 GB memory. All the algorithms are implemented in C++.

B. Evaluation Results

As shown in Tab.I, our results demonstrate that the proposed DP algorithm consistently achieves the highest total rewards across all tested scenarios. The DP solution is the performance upper bound for our multi-hop relay-aided offloading framework by optimally allocating computing resources and selecting minimum-slot offloading paths. However, this optimality comes at a significant computational cost, especially in `med-large` and `large` settings where the feasible configuration space grows rapidly. For example, as shown in Tab.II, in the `large` test case with 15 users, 2 servers, and 3 relays, the DP algorithm incurs substantial runtime overhead due to the explosion in the number of multi-dimensional knapsack states. This indicates the practical limitations of the exact method in scenarios where low-latency decision-making is essential.

	small	medium	med-large	large
Restr/Exact	0.99259	0.98137	0.98283	0.97482
Exact/Relax	1.00000	1.00000	0.99833	0.99451
Restr/Relax	0.99259	0.98137	0.98119	0.96958

TABLE I: Ratio of total rewards across all test case sizes

To address this scalability challenge, we evaluate two efficient heuristics derived from relaxed and restricted variants of the original **MRO** problem. To quantify the quality of these approximations, we use the three reward ratios: 1) the ratio of the total reward achieved by solving the restricted problem to that of the optimal DP solution, 2) the ratio of the total reward achieved by the optimal DP solution to that of solving the relaxed problem, and 3) the ratio of the total reward achieved by solving the restricted problem to that of solving the relaxed problem. As shown in Tab.I, across all experiments, the relaxed heuristic consistently achieved reward ratios close to 1, while the restricted heuristic produced slightly more conservative results but retained high performance. As shown in Tab.II, both heuristics reduce the execution time significantly compared to the DP algorithm. These findings confirm that both heuristics maintain near-optimal reward outcomes while dramatically reducing computation time, offering a scalable and practical alternative for real-world deployment.

	small	medium	med-large	large
Restricted	0.425	15.758	195.036	242.915
Relaxed	0.439	16.458	205.151	255.470
Exact	1.106	580.99	4142.968	5143.313

TABLE II: Running time (s) across all test case sizes

7. Conclusion

This paper presents a multi-hop relay-aided task offloading framework that enables flexible task execution at local devices, relay nodes, and edge servers under a TDMA-based communication model. We formulate a problem that jointly optimizes offloading strategies, resource allocation, and time frame/slot assignment. Our approach constructs a feasibility-aware directed graph and assigns the largest feasible number of time frames per user to improve bandwidth utilization. We select slot-minimizing offloading paths for each user to reduce slot consumption further. We develop an exact DP algorithm to solve the problem optimally and propose scalable heuristic methods based on resource scaling and rounding. Extensive experiments show that the proposed DP algorithm achieves optimal results, while the heuristic methods provide competitive performance with significantly lower runtime.

References

[1] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 974–983, 2014.

[2] Y. Deng, Z. Chen, X. Chen, and Y. Fang, "Task offloading in multi-hop relay-aided multi-access edge computing," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 1, pp. 1372–1376, 2022.

[3] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Transactions on Information Theory*, vol. 46, no. 2, pp. 388–404, 2000.

[4] S. Lin, Z. Zhou, Z. Zhang, X. Chen, and J. Zhang, *Edge intelligence in the making: Optimization, DL, and applications*. Springer, 2021.

[5] J. Perazzone, M. Dwyer, K. Chan, C. Anderson, and S. Brown, "Enabling machine learning on resource-constrained tactical networks," in *2022 IEEE Military Communications Conference (MILCOM)*. IEEE, 2022, pp. 932–937.

[6] Z. Zhang, S. Lin, M. Dedeoglu, K. Ding, and J. Zhang, "Data-driven distributionally robust optimization for edge intelligence," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 2619–2628.

[7] Z. Zhang, X. Lin, G. Xue, Y. Zhang, and K. S. Chan, "Joint optimization of task offloading and resource allocation in tactical edge networks," in *MILCOM 2024-2024 IEEE Military Communications Conference (MILCOM)*. IEEE, 2024, pp. 703–708.

APPENDIX

A. Proof of Proposition 1

Proof. Let $\left(X_n^{(1)}, q_n^{(1)}\right)$ and $\left(X_n^{(2)}, q_n^{(2)}\right)$ be two feasible configurations of time frame assignment and offloading paths from u_n to the same destination with selecting the algorithm k, such that $T_n^{(1)} < T_n^{(2)}$, where $T_n^{(i)}$ is the number of time slots induced by selecting path $\left(X_n^{(i)}, q_n^{(i)}\right)$. Since the reward is the same and the first path consumes fewer time slots, choosing $\left(X_n^{(1)}, q_n^{(1)}\right)$ results in a more favorable residual resource state for subsequent users in the optimization process. Thus, any exact algorithm aiming to maximize the total reward under resource constraints can achieve at least the same or better total reward by selecting the minimum-slot time frame assignment and offloading path. Consequently, restricting the solution space to such minimum-slot assignments is optimal. □

B. Proof of Theorem 1

Proof. In the worst case, computing $\texttt{OPT}(n, \mathbb{B}; \Omega)$ requires iterating over all the sections of K ML algorithms, M edge servers, L relays, and processing locally. Hence, the time complexity of computing $\texttt{OPT}(n, \mathbb{B}; \Omega)$ is $O((M + L)K)$. The whole DP process requires computing all the cases of $\texttt{OPT}(n, \mathbb{B}; \Omega)$, which requires $O(|\mathcal{B}|N)$ time. Thus, the final time complexity of the proposed DP algorithm is $O(|\mathcal{B}|N(M + L)K)$. □

C. Proof of Lemma 1

Proof. Given that $x + y \leq b$, we have

$$\lfloor x \rfloor + \lfloor y \rfloor \leq \lfloor x + y \rfloor \leq \lfloor b \rfloor. \tag{25}$$

The second inequality is because the floor function is monotonically non-decreasing. This yields (21).

Similarly, given that $\lceil x \rceil + \lceil y \rceil \leq \lfloor b \rfloor$, we have

$$x + y \leq \lceil x + y \rceil \leq \lceil x \rceil + \lceil y \rceil \leq \lfloor b \rfloor \leq b, \tag{26}$$

which yields (22). □