Task Offloading across Unreliable Edge Networks via Distributional Dynamic Programming

Xuanli Lin*, Zhaofeng Zhang†, Zunzheng Zhang*, Guoliang Xue*
*Arizona State University, Tempe, AZ, USA †Delaware State University, Dover, DE, USA
*{xlin54, zzhan621, xue}@asu.edu †zzhang@desu.edu

IEEE INFOCOM 2026 - IEEE Conference on Computer Communications | 979-8-3315-4961-9/26/$31.00 ©2026 IEEE | DOI: 10.1109/INFOCOM59046.2026.11571468

***Abstract*—As Internet of Things (IoT) networks evolve toward integrated sensing and communications (ISAC), ensuring reliable resource management at the mobile edge has become increasingly complex. The combination of limited communication and computing resources with unreliable wireless links necessitates highly robust offloading strategies. Traditional approaches that merely maximize expected network utility often fail to account for the stochastic link failures in dynamic sensing systems, resulting in severe utility degradation. To address this, we propose a risk-averse task offloading framework that systematically manages uncertainties in sensing-enabled mobile edge computing. We formulate a joint optimization problem that maximizes expected total task reward, subject to a conditional value-at-risk (CVaR) constraint, effectively mitigating the tail risk of communication task failures. We solve the resulting optimization problem via a novel distributional dynamic programming (DDP) approach. To improve computational efficiency, we integrate Pareto Frontier pruning based on first-order stochastic dominance (FSD) alongside scaling and rounding approximations. Evaluation results confirm that our framework successfully navigates the risk-reward trade-offs in resource-constrained deployments.**

***Index Terms*—edge computing, task offloading, risk-averse optimization, conditional value-at-risk, distributional dynamic programming.**

I. Introduction

The advent of 6G and next-generation Internet of Things (IoT) has driven the emergence of sensing-enabled edge networks. This operational environment is frequently characterized by intermittent connectivity, limited bandwidth, and resource-constrained devices. In addition, there is explosive growth in the number of IoT devices deployed across edge networks. These devices are increasingly required to perform complex sensing and computational tasks, such as real-time LiDAR data processing and joint sensing-communication coordination. To bridge the gap between heavy computational demands and limited device capabilities, task offloading has emerged as a crucial paradigm, enabling devices to offload intensive workloads to nearby edge servers. However, edge offloading environments differ significantly from commercial cellular networks. Wireless links are inherently unreliable due to dynamic sensing targets, mutual interference between sensing and communication payloads, mobility, and potential jamming, leading to stochastic link failures. When a link fails during offloading, the task may not be completed, resulting in the loss of critical utility.

In this context, standard resource management approaches that solely maximize expected rewards are insufficient. An average performance metric might mask the possibility of catastrophic failures in which critical data or tasks are dropped. For example, in mission-critical applications, such as autonomous vehicular tracking or distributed radar networks, system designs require a risk-averse strategy that provides statistical guarantees against worst-case scenarios.

This paper addresses the challenge of designing risk-averse, resource-aware offloading strategies in the presence of stochastic link failures. We introduce a risk metric based on conditional value at risk (CVaR), which focuses on the tail of the loss distribution and provides a more robust measure for critical sensing applications than other metrics, such as average utility or the variance of utility.

Our main contributions are as follows:

- We formulate risk-averse task offloading as a constrained optimization problem that maximizes expected reward subject to resource and CVaR constraints.
- We develop a distributional dynamic programming (DDP) approach that propagates full loss distributions rather than scalar expectations. To mitigate the curse of dimensionality, we employ first-order stochastic dominance (FSD) to prune suboptimal states from the Pareto Frontier without loss of optimality.
- We introduce restricted and relaxed approximation schemes based on capacity scaling and rounding to effectively trade off between solution quality and computational runtime.
- We provide evaluation results to validate the effectiveness of our framework in managing risk-reward trade-offs.

The remainder of this paper is organized as follows. Section II reviews the related work. The system model is detailed in Section III. In Section IV, we formulate a risk-neutral optimization problem and propose a Dynamic Programming (DP) solution. Section V extends this framework to a risk-averse setting, introducing a DDP algorithm along with its approximation schemes. Section VI presents the performance evaluation, and Section VII concludes the paper.

Research was sponsored by the 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 the 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.

II. Related Work

Task offloading in mobile edge computing (MEC) has been extensively studied to alleviate the resource limitations of edge devices [1]. However, as next-generation networks evolve, research has expanded from pure communication models to ISAC frameworks.

A. Task Offloading in ISAC Networks

Recent literature has heavily emphasized the integration of MEC into ISAC systems to handle computationally intensive sensing tasks. For instance, Huang et al. [2] proposed a MEC-aided ISAC paradigm with short-packet transmissions, optimizing computing capacity and beamforming to reduce execution latency. Similarly, Liu et al. [3] developed a joint beamforming and offloading design for ISAC systems to process massive volumes of sensing echo signals at the network edge. In highly dynamic scenarios, such as UAV and vehicular networks, multi-agent reinforcement learning has been utilized to minimize offloading latency under strict ISAC constraints [4]. While these works successfully establish the foundation for ISAC offloading, they primarily optimize average performance metrics (e.g., expected delay or energy) in stable environments, lacking explicit safeguards against stochastic link outages.

B. Reliability and Risk in Edge Operations

To address uncertainty, research has pivoted toward reliability-aware resource management. Works such as Bennis et al. [5] highlighted the necessity of managing the "tail" of the latency distribution for ultra-reliable communications. In mission-critical and tactical scenarios, Perazzone et al. [6] and Zhang et al. [7] addressed resource-constrained offloading via multi-objective and joint optimization, which was extended to multi-hop scenarios [8]. However, these expectation-based models treat link failures as simple probability weights. This approach can mask high-risk, catastrophic task failures, which can be detrimental in critical ISAC operations.

C. Distributional Approaches to Risk

Standard dynamic programming (DP) is ill-suited to coherent risk measures such as CVaR due to its lack of linearity. To explicitly penalize catastrophic failure modes, our approach is inspired by distributional reinforcement learning [9], which advocates propagating the full distribution of returns rather than scalar expectations. Recent studies, such as Sun et al. [10], have validated the need for distribution-based decision-making in risk-sensitive offloading.

III. System Model

We consider a tactical edge network where users can offload machine learning (ML) inference tasks to an edge server for execution. The network consists of multiple servers that have fixed resources and can communicate wirelessly with users. In the following sections, we detail our network, user, communication, and link failure models.

A. Network and User Model

We consider an edge network consisting of a set of N users, denoted by $\mathcal{U} = \{u_1, u_2, \ldots, u_N\}$, and a set of M edge servers, denoted by $\mathcal{S} = \{s_1, s_2, \ldots, s_M\}$. Each user u_n is located at $(x_n^{\mathcal{U}}, y_n^{\mathcal{U}})$ and has a computation task characterized by a strict deadline δ_n and a data size D_n. Each task can be processed with an ML algorithm tier $k \in \{1, 2, \ldots, K\}$. An algorithm with a higher tier yields better reward, at the cost of increased resource usage. We assume that the reward obtained from an ML algorithm upon the task's completion is positively correlated with its resource requirements. For a user n offloading with algorithm k, the parameters are:

- $\text{CPU}_{n,k}$: Required number of vCPU cores on the server.
- $\text{RAM}_{n,k}$: Required RAM capacity on the server.
- $\tau_{n,k}^{\text{exe}}$: Execution time on the server.
- $r_{n,k}$: Reward obtained upon successful completion.

The server m is located at $(x_m^{\mathcal{S}}, y_m^{\mathcal{S}})$ and has fixed resource capacities C_m (vCPU cores) and R_m (RAM).

B. Communication Model

Communication is managed via a time division multiple access (TDMA) scheme. The total available offloading time is divided into X frames. Each frame is further partitioned into T time slots, each with a duration of z seconds.

For user n who is offloading a task, the system must assign a number of frames $X_n \in [0, X]$ and a number of slots per frame $T_n \in [0, T]$. The total effective transmission time allocated to user n is given by $X_n \cdot z \cdot T_n$.

The wireless channel is modeled using distance-dependent path loss. The signal-to-noise ratio (SNR) for a transmission from user n to server m is:

$$\text{SNR}_{n,m} = \frac{P_{\max}}{\nu \cdot (d(n,m))^{\alpha}}, \tag{1}$$

where $P_{\max}$ is the transmission power, $d(n,m)$ is the Euclidean distance between user n and server m, $\alpha \in [2,4]$ is the path loss exponent, and ν is the noise power [11]. Note that under this TDMA scheme, each (offloading) user is assigned exclusive communication time with a server; thus, there is no interference between users.

According to Shannon's capacity formula, the time required to transmit data of size D_n from user n to server m is:

$$\tau_{n,m}^{\text{up}} = \frac{D_n}{W \log_2(1 + \text{SNR}_{n,m})}, \tag{2}$$

where W is the channel bandwidth.

C. Feasible Time Assignments

A critical aspect of the system model is determining the feasible region for time assignments. For a task to be completed successfully, the total time (transmission plus execution) must be within the deadline δ_n, and the allocated transmission time must be sufficient for the data size. Given a selected algorithm $k(n)$ for user n, the maximum number of frames X_n that can

be utilized without violating the deadline is bounded by the remaining time after execution. Hence we must have

$$X_n \cdot z \cdot T + \tau^{\text{exe}}_{n,k(n)} \leq \delta_n. \tag{3}$$

Rearranging (3), we obtain an upper bound for X_n:

$$X^{\text{ub}}_n = \min\left(X, \left\lfloor \frac{\delta_n - \tau^{\text{exe}}_{n,k(n)}}{T \cdot z} \right\rfloor\right). \tag{4}$$

There is no benefit in selecting a smaller X_n than this upper bound, as doing so would require a larger T_n (i.e., more slots per frame) to maintain the same throughput, consuming shared resources. Thus, we fix $X_n = X^{\text{ub}}_n$.

With X_n fixed, the minimum required slots per frame, T_n, to satisfy the transmission requirement $\tau^{\text{up}}_{n,m}$ is:

$$T_n = \left\lceil \frac{\tau^{\text{up}}_{n,m}}{X_n \cdot z} \right\rceil. \tag{5}$$

If $T_n > T$, the offloading is infeasible for this server-algorithm pair.

D. Probabilistic Link Failures

We model the success of an offloading attempt from user n to server m as a Bernoulli random variable. The success probability is denoted by $p_{n,m}$, and the failure probability is $1 - p_{n,m}$. If a link fails, the task is not completed, and the reward is 0. The failure can be caused by changes in the radio channel (fading, blockage, etc.), network errors, and other factors. The link failure probability can be estimated from historical data.

IV. Risk-Neutral Offloading

Given a problem instance specified in Section III, a natural goal is to maximize the expected total reward while ensuring the solution's feasibility.

A. Decision Variables

The offloading strategy is defined by a mapping $\mathcal{A} = (A(1), \ldots, A(N))$, which is a mapping of $\mathcal{U} \to \mathcal{S} \times \mathcal{K}$, where $\mathcal{K}$ is the set of all offloading tiers. For each user n:

$$A(n) = \begin{cases} (s_{m(n)}, k(n)) & \text{if offloading to server } m(n) \neq 0, \\ (s_0, 0) & \text{if not offloading.} \end{cases} \tag{6}$$

We also determine the set of time slot assignments $\mathcal{T} = \{T_1, \ldots, T_N\}$.

B. Objective and Constraints

The primary objective is to maximize the expected total reward, subject to constraints on the servers, physical channel, and task deadlines. Formally:

1) **Server capacities**: the total CPU and RAM usage for each server must not exceed its total capacity. For each server $s_m \in \mathcal{S}$.

$$\sum_{u_n \in \mathcal{U}, m(n)=m} \text{CPU}_{n,k(n)} \leq C_m, \tag{7}$$

$$\sum_{u_n \in \mathcal{U}, m(n)=m} \text{RAM}_{n,k(n)} \leq R_m. \tag{8}$$

2) **Time slot capacity**: The sum of all slots assigned to offloading users must not exceed the frame size T.

$$\sum_{u_n \in \mathcal{U}, m(n) \neq 0} T_n \leq T. \tag{9}$$

3) **Time constraints**: The transmission and execution of each task must be completed by its deadline, and the allocated frames and timeslots must allow the data to be uploaded completely. For all users $u_n \in \mathcal{U}, m(n) \neq 0$:

$$X_n z T + \tau^{\text{exe}}_{n,k(n)} \leq \delta_n, \tag{10}$$

$$X_n z T_n \geq \tau^{\text{up}}_{n,m(n)}. \tag{11}$$

4) **Link quality**: for any link between a user n and server s_m, it must have an SNR no less than γ.

$$\text{SNR}_{n,m(n)} \geq \gamma. \tag{12}$$

The overall optimization problem is defined as follows.

$$\max_{\mathcal{A},\mathcal{T}} \sum_{\substack{u_n \in \mathcal{U} \\ m(n) \neq 0}} p_{n,m(n)} r_{n,k(n)} \tag{13}$$

s.t. constraints (7)-(12).

C. Problem Analysis

Problem (13) is a non-linear integer problem due to the presence of binary decision variables $\mathcal{A}$. A closer inspection reveals the knapsack nature of resource constraints–each server's CPU and RAM resources can be thought of as a knapsack with varying capacities (C_m and R_m for $s_m \in \mathcal{S}$), and each offloading task has certain "weights" or resource requirements. As such, there are M 2-dimensional, non-identical knapsacks for the servers, plus one special knapsack with a capacity of T that represents the slot assignment for each frame. The expected reward for scheduling a user n with algorithm tier $k(n)$ at server $m(n)$ is $p_{n,m(n)} r_{n,k(n)}$.

D. A Dynamic Programming Approach

The knapsack nature of the problem (13) invites a classical DP approach. Intuitively, we can iteratively build a look-up table where a column represents a possible residual capacity of all the resources, and a row represents a user. Each cell contains a feasible offloading action for the user that yields the best reward. This way, we capture the best offloading strategy overall at the end.

To realize this idea, we define an array $\mathbb{B}$ of length $2M+1$, where $\mathbb{B}[2m-1]$ and $\mathbb{B}[2m]$ represent (residual) CPU and RAM capacities at server s_m, and $\mathbb{B}[2M+1]$ represents (residual) time slots.

In addition, let $\mathcal{U}_n = \{u_1, \ldots, u_n\}$ be the set of the first n users in $\mathcal{U}$. Define $\text{OPT}(n, \mathbb{B})$ as the largest total reward for scheduling the tasks of $\mathcal{U}_n$ for a given knapsack capacity $\mathbb{B}$. $\text{OPT}(n, \mathbb{B})$ can be calculated as follows:

Given u_n, scan all servers $s_m \in \mathcal{S}$ and $k \in \mathcal{K}$

1) Check the SNR requirement.
2) Update the knapsacks:
   - $\mathbb{B}'[2m-1] = \mathbb{B}[2m-1] - \text{CPU}_{n,k}$;

- $\mathbb{B}'[2m] = \mathbb{B}[2m] - \text{RAM}_{n,k}$;
- $\mathbb{B}'[2M+1] = \mathbb{B}[2M+1] - T_n$.

3) Check the residual resources $\mathbb{B}'$:
   a) If any element in $\mathbb{B}'$ is negative: do not update $\text{OPT}(n, \mathbb{B})$.
   b) Otherwise, $\text{OPT}(n, \mathbb{B}) = \max\{\text{OPT}(n-1, \mathbb{B}), \text{OPT}(n-1, \mathbb{B}') + p_{n,m} r_{n,k}\}$

A simple exact DP algorithm is provided in Algorithm 1.

Algorithm 1 `ExactDP`$(\mathcal{U}, \mathcal{B})$

Input: Users $\mathcal{U}$, Capacities $\mathcal{B}$
Output: $\mathcal{A}^*$: optimal offloading strategy; $\mathcal{T}^*$: optimal time slot assignments

```
Initialize OPT(0, B) ← 0, ∀B ∈ B;
for u_n ∈ U do
    for each B ∈ B in its complete order do
        Compute OPT(n, B);
        Update A(n, B), T(n, B);
    end for
end for
return A*, T*
```

Note that $\mathcal{B}$ is a set that contains the Cartesian product of all knapsack capacities: $\mathcal{B} = \mathcal{C}_1 \times \mathcal{R}_1 \times \ldots \times \mathcal{C}_M \times \mathcal{R}_M \times \mathcal{T}'$. where $\mathcal{C}_m = \{i \in \mathbb{Z} | 0 \le i \le C_m\}$, $\mathcal{R}_m = \{i \in \mathbb{Z} | 0 \le i \le R_m\}$ for $m = \{1, \ldots M\}$, and $\mathcal{T}' = \{i \in \mathbb{Z} | 0 \le i \le T\}$.

This approach works well for typical reward-oriented scenarios, such as commercial network offloading. However, it may fall short in critical or hostile situations, including emergency response and tactical edge networks. An offloading strategy that maximizes expected rewards may rely on less reliable links, leading to catastrophic failures. Motivated by this observation, we seek to design a risk-averse solution that accounts for potential losses while maximizing expected rewards.

V. Risk-Averse Offloading

A. Risk Metric Preliminaries

We define the stochastic loss for a chosen strategy $\mathcal{A}$ under a realization of link failures Ξ as the sum of potential rewards lost due to failure:

$$L(\mathcal{A}, \Xi) = \sum_{u_n \in \mathcal{U}, m(n) \neq 0} (1 - \xi_n) r_{n,k(n)}, \tag{14}$$

where $\xi_n \sim \text{Bernoulli}(p_{n,m(n)})$.

Value at risk (VaR) and CVaR are two widely used risk measures in economics and finance [12]. For a random loss variable L and a confidence level $\beta \in (0,1)$, $\text{VaR}_\beta(L)$ is the smallest threshold such that the probability of the loss not exceeding this threshold is at least β:

$$\text{VaR}_\beta(L) = \inf\{x \in \mathbb{R} | \Pr(L \le x) \ge \beta\} \tag{15}$$

VaR provides a cutoff point for losses but ignores the magnitude of losses beyond that point. In edge offloading scenarios, "tail" events (rare but catastrophic failures) are critical, and we need to incorporate them into our formulation. Toward this goal, we adopt CVaR, an idea closely related to VaR that captures the severity of the failure. It is the expected tail loss beyond VaR:

$$\text{CVaR}_\beta(L) = \mathbb{E}[L | L \ge \text{VaR}_\beta(L)] \tag{16}$$

An equivalent convex formulation is given by [12]

$$\text{CVaR}_\beta(L) = \min_{\eta \in \mathbb{R}} \left\{ \eta + \frac{1}{1-\beta} \mathbb{E}[(L-\eta)^+] \right\}. \tag{17}$$

where $(x)^+ = \max\{x, 0\}$.

Using (17), we introduce robustness in our design. The risk-averse optimization problem is:

$$\max_{\mathcal{A}, \mathcal{T}} \quad \sum_{u_n \in \mathcal{U}, m(n) \neq 0} p_{n,m(n)} r_{n,k(n)} \tag{18}$$

$$\text{s.t.} \quad \text{Constraints } (7) - (12),$$

$$\text{CVaR}_\beta(L(\mathcal{A}, \Xi)) \le \Gamma, \tag{19}$$

where Γ is the maximum acceptable risk tolerance.

It is worth noting that CVaR is a *coherent* risk measure, meaning that it satisfies properties such as sub-additivity (i.e., $\text{CVaR}(X+Y) \le \text{CVaR}(X) + \text{CVaR}(Y)$). This complicates the design of our solution, as standard DP relies on the linearity of the objective function. Because of this sub-additivity (and resulting non-linearity), the Bellman equation does not hold for scalar risk values, and decisions optimal for a subproblem may not be optimal globally. Thus, there is a need to track distributions under a global risk constraint. In the next section, we present our solution for this problem.

B. Distributional Dynamic Programming

Because CVaR is not a linear operator, the optimal substructure property does not hold for scalar values of reward when a global risk constraint is present. A partial solution with high expected reward might have a risk profile that, when combined with future decisions, violates the global constraint. Therefore, we must propagate the *distribution* of the loss.

Due to the stochastic nature of the loss, we introduce a Pareto frontier to capture the risk and reward. For a state $(n, \mathbb{B})$, we define $\text{Frontier}(n, \mathbb{B}) = \{(P, R)\}$, a set of non-dominated pairs where P is the PMF of the accumulated loss, and R is the accumulated expected reward for the first n users given the residual capacity $\mathbb{B}$. In essence, the Pareto frontier stores all reward-risk tradeoffs that could be optimal for each state.

The size of the set $\text{Frontier}(n, \mathbb{B})$ can grow exponentially. To manage this, we prune the frontier using FSD. A candidate solution (P', R') dominates another solution (P, R) if: 1) $R' \ge R$ (Better or equal expected reward), and 2) $F_{P'}(\ell) \ge F_P(\ell)$ for all ℓ, where $F_P(\ell) = \Pr[L \le \ell]$. This implies that for any loss threshold, solution P' has a higher probability of keeping the loss *below* that threshold, indicating strictly lower risk.

If a solution is dominated, it is removed from the frontier.

C. Algorithm Description

The DDP algorithm proceeds stage by stage (user by user).

1) **Initialization:** Frontier$(0, \mathbb{B}) = \{(d_0, 0)\}$ for all $\mathbb{B} \in \mathcal{B}$, where d_0 is the Dirac delta PMF (zero loss).
2) **Transition:** For user n, and for each state $\mathbb{B}$, we consider all valid actions (offload to s_m with alg. k, or no offload).
3) **Update:** For a valid action with reward r and success prob p, we update each (P, R) in the previous frontier. The new reward is $R + p \cdot r$. The new loss PMF is the convolution of P with the current user's Bernoulli loss distribution.
4) **Pruning:** Apply FSD to reduce the frontier size.
5) **Final Selection:** At stage N, calculate CVaR for all surviving PMFs. Select the solution with max R that satisfies CVaR $\leq \Gamma$.

Algorithm 2 captures steps 2 and 3 above.

Algorithm 2 `Frontier`$(n, \mathbb{B})$

Input: user $u_n \in \mathcal{U}$, capacities $\mathbb{B}$
Output: `Frontier`$(n, \mathbb{B})$: set of non-dominated pairs $(\mathcal{P}, \mathcal{R})$ under $\mathbb{B}$

$\mathcal{F} \leftarrow \emptyset$
for all $(\mathcal{P}, \mathcal{R}) \in$ `Frontier`$(n-1, \mathbb{B})$ **do**
  Insert $(\mathcal{P}, \mathcal{R})$ into $\mathcal{F}$
end for
for all $s_m \in \mathcal{S}$ **do**
  if $\mathrm{SNR}_{n,m} < \gamma$ **then**
    continue
  end if
  for all $k \in \mathcal{K}$ **do**
    $X_{n,k} \leftarrow \left(X, \left\lfloor \frac{\delta_n - \tau^{\mathrm{exe}}_{n,k}}{Tz} \right\rfloor\right)$
    if $X_{n,k} \leq 0$ **then**
      continue
    end if
    $T_{n,m,k} \leftarrow \left\lceil \frac{\tau^{\mathrm{up}}_{n,m}}{X_{n,k}\, z} \right\rceil$
    $\hat{\mathbb{B}} \leftarrow \mathbb{B}$
    $\hat{\mathbb{B}}[2m-1] \leftarrow \mathbb{B}[2m-1] - \mathrm{CPU}_{n,k}$
    $\hat{\mathbb{B}}[2m] \leftarrow \mathbb{B}[2m] - \mathrm{RAM}_{n,k}$
    $\hat{\mathbb{B}}[2M+1] \leftarrow \mathbb{B}[2M+1] - T_{n,m,k}$
    if any element of $\hat{\mathbb{B}}$ is negative **then**
      continue
    end if
    $p \leftarrow p_{n,m}$; $\Delta\ell \leftarrow r_{n,k}$
    for all $(\mathcal{P}, \mathcal{R}) \in$ `Frontier`$(n-1, \hat{\mathbb{B}})$ **do**
      $\mathcal{P}'(\ell) \leftarrow p\,\mathcal{P}(\ell) + (1-p)\,\mathcal{P}(\ell - \Delta\ell)$ for all ℓ
      $\mathcal{R}' \leftarrow \mathcal{R} + p \cdot r_{n,k}$
      Insert $(\mathcal{P}', \mathcal{R}')$ into $\mathcal{F}$
    end for
  end for
end for
Prune $\mathcal{F}$ based on the dominance rule
return $\mathcal{F}$

The `Frontier` algorithm is executed on $n = 1, 2, \ldots, N$ and $\forall \mathbb{B} \in \mathcal{B}$ in its complete order. At the end of execution, we prune the resulting non-dominating set $\mathcal{F}$, then extract the solution with the maximum R that satisfies the CVaR restriction. This is listed in Algorithm 3. Note that $\bar{\mathbb{B}} = (C_1, R_1, \ldots C_M, R_M, T)$, which is the maximum capacity for all knapsacks.

Algorithm 3 `FinalStage`$(N, \bar{\mathbb{B}}, \Gamma)$

$\mathcal{F} \leftarrow$ `Frontier`$(N, \bar{\mathbb{B}})$
$\mathcal{R}_{\text{best}} \leftarrow -\infty$
$\mathcal{A}^\star \leftarrow \emptyset$
for all $(\mathcal{P}, \mathcal{R}) \in \mathcal{F}$ **do**
  Compute corresponding CVaR
  if CVaR $\leq \Gamma$ **and** $\mathcal{R} > \mathcal{R}_{\text{best}}$ **then**
    $\mathcal{R}_{\text{best}} \leftarrow \mathcal{R}$
    Update $\mathcal{A}^\star$ by backtracking
  end if
end for
return $\mathcal{A}^\star$

D. Complexity and Approximations

The exact DDP iterates over $O(|\mathcal{B}| \cdot N \cdot M \cdot K)$ states, where $|\mathcal{B}|$ is the size of the knapsack space. Since the state space grows with the product of resource capacities, this is pseudo-polynomial.

To reduce complexity, we apply scaling and rounding. For each server s_m, its CPU capacity C_m and RAM capacity R_m are scaled down to $\lfloor \lambda \rfloor$, where λ is a positive scaling factor. After the scaling, the resource knapsack becomes $(\bar{\mathbb{B}}_r = (\lfloor \lambda \rfloor, \ldots, \lfloor \lambda \rfloor, T))$. In addition, we can apply relaxation or restriction on the user's resource requirements in connection with the scaled server capacities.

- **Relaxation:** User n's CPU and RAM requirements are scaled to $\lfloor \lambda \frac{\mathrm{CPU}_{n,k(n)}}{C_m} \rfloor$ and $\lfloor \lambda \frac{\mathrm{RAM}_{n,k(n)}}{R_m} \rfloor$, respectively. This is an optimistic bound (which may admit infeasible solutions).
- **Restriction:** User n's CPU and RAM requirements are scaled to $\lceil \lambda \frac{\mathrm{CPU}_{n,k(n)}}{C_m} \rceil$ and $\lceil \lambda \frac{\mathrm{RAM}_{n,k(n)}}{R_m} \rceil$, respectively. This guarantees feasibility in the original domain but may miss the optimal solution.

Proper selection of λ balances runtime and optimality gap.

VI. Performance Evaluation

A. Simulation Setup

We evaluated the framework on three network sizes:

- **Small:** 5 users, 2 servers, 2 ML algorithms.
- **Medium:** 10 users, 2 servers, 2 ML algorithms.
- **Large:** 15 users, 2 servers, 2 ML algorithms.

For all network sizes, we have $T = 40$ slots, a slot duration of $z = 0.025$s, and a scaling factor of $\lambda = 4$. Failure probabilities between users and servers were randomly drawn from $\mathcal{U}_{[0,1]}$.

B. Rounding and Scaling Accuracy

We first evaluate the accuracy of the proposed rounding and scaling scheme. Table I shows the ratio of the total expected reward obtained by the approximation algorithms (`Restricted` and `Relaxed`) to the `Exact` algorithm. The `Relaxed` version serves as an upper bound, while the `Restricted` version is a lower bound.

In the risk-averse scenario ($\beta = 0.90$), the `Restricted` algorithm maintains high accuracy, achieving 97.2% of the optimal reward even in the Large test case. The gap between the `Restricted` and `Relaxed` bounds is consistently small ($\leq 2\%$ difference), indicating that the actual optimal value is tightly bracketed.

Table I: Approximation Accuracy (Reward Ratio) for Risk-Averse Strategy ($\beta = 0.90$)

Ratio Metric	Small	Medium	Large
Restricted / Exact	1.000	0.987	0.972
Exact / Relaxed	1.000	0.994	0.992
Restricted / Relaxed	1.000	0.981	0.986

C. Computational Efficiency

The primary motivation for approximation is runtime reduction. Table II compares the running times across different strategies and problem sizes. While the `Max-Exp` (risk-neutral) DP is relatively fast (2.49s for Large), adding risk constraints (risk-averse) causes the `Exact` algorithm's runtime to explode to 196 seconds for the Large case due to the overhead of managing full loss distributions. However, the `Restricted` approximation remains highly efficient, solving the Large risk-averse problem in just 11.4 seconds—an order of magnitude faster than `Exact`, with minimal loss in optimality (as shown in Table I).

Table II: Running Time (seconds) Comparison

Strategy	Method	Small	Medium	Large
Max-Exp (Risk Neutral)	Restricted	0.068	0.115	0.159
	Relaxed	0.067	0.119	0.161
	Exact	0.208	1.710	2.490
Risk-Averse ($\beta = 0.90$)	Restricted	0.747	1.600	11.40
	Relaxed	0.748	1.610	11.30
	Exact	2.290	32.30	196.0

D. Risk-Reward Trade-off Analysis

Finally, we analyze the impact of the risk constraint stringency (Γ) on system performance. Table III shows the performance of the risk-averse strategy relative to the unconstrained `Max-Exp` baseline.

Table III: Risk-Averse Performance vs. `Max-Exp` Baseline

Metric	Small	Medium	Large
Tight Constraint ($\Gamma = 6$)			
Reward Ratio	0.478	0.505	0.638
CVaR Ratio	0.327	0.357	0.371
Loose Constraint ($\Gamma = 10$)			
Reward Ratio	0.725	0.749	0.761
CVaR Ratio	0.552	0.602	0.645

Under a tight constraint ($\Gamma = 6$), the system is highly conservative. For the Large case, it sacrifices roughly 36% of potential reward (reward ratio 0.638) but successfully suppresses the tail risk to just 37% of the baseline level (CVaR Ratio 0.371). Under a loose constraint ($\Gamma = 10$), the system operates with greater flexibility, capturing 76% of the optimal reward while reducing risk exposure to 64.5% of the baseline. This demonstrates that the proposed framework provides a tunable knob for operators to balance mission rewards against the risk of failure. These results validate that the proposed framework enables operators to strictly enforce risk limits while effectively navigating the trade-off between aggressive performance maximization and operational safety.

VII. Conclusion

This paper presents a risk-averse task offloading framework for edge networks. By integrating CVaR into the optimization objective and solving it via DDP with FSD pruning, we provide a method to control the tail risk of performance loss. Our heuristics effectively reduce computational complexity, making the approach viable for larger networks. Possible future works include deep reinforcement learning to handle time-varying channel dynamics and to extend the model to multi-hop relay scenarios.

References

[1] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE communications surveys & tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[2] N. Huang, C. Dou, Y. Wu, L. Qian, B. Lin, H. Zhou, and X. Shen, "Mobile edge computing aided integrated sensing and communication with short-packet transmissions," *IEEE Transactions on Wireless Communications*, vol. 23, no. 7, pp. 7759–7774, 2023.

[3] P. Liu, Z. Fei, X. Wang, Y. Zhou, Y. Zhang, and F. Liu, "Joint beamforming and offloading design for integrated sensing, communication, and computation system," *IEEE Transactions on Vehicular Technology*, 2025.

[4] A. Paul, K. Singh, C.-P. Li, and S. Mumtaz, "Minimizing urllc task offloading latency with full-duplex star-ris-aided drl-isac systems," in *GLOBECOM 2024-2024 IEEE Global Communications Conference*. IEEE, 2024, pp. 73–78.

[5] M. Bennis, M. Debbah, and H. V. Poor, "Ultrareliable and low-latency wireless communication: Tail, risk, and scale," *Proceedings of the IEEE*, vol. 106, no. 10, pp. 1834–1853, 2018.

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

[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.

[8] ——, "Multi-hop relay-aided task offloading for tactical edge computing," in *MILCOM 2025 - 2025 IEEE Military Communications Conference (MILCOM)*. IEEE, 2025.

[9] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *International conference on machine learning*. PMLR, 2017, pp. 449–458.

[10] Y. Sun, X. Guo, S. Zhou, Z. Niu, Y. Cao, and L. Zhang, "Risk-sensitive task offloading in mobile edge computing: A distributional reinforcement learning approach," *IEEE Transactions on Wireless Communications*, vol. 22, no. 5, pp. 3215–3229, 2023.

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

[12] R. T. Rockafellar and S. Uryasev, "Optimization of conditional value-at-risk," *Journal of risk*, vol. 2, pp. 21–42, 2000.