Decentralized Cost-Aware Learning for Deadline-Reliable Offloading in Cloud Robotics

[Research]

Anonymous Author(s)

Abstract

Reliable low-latency offloading is critical for cloud robotics, where computationally heavy tasks must meet strict real-time deadlines despite highly variable end-to-end latency across commodity Wi-Fi/5G links and shared cloud resources. Existing approaches such as FogROS2-PLR improve deadline reliability through redundant multi-path offloading, but rely on static probabilistic models and fixed interface–server mappings that lead to excessive resource usage and brittle behavior under correlated latency disruptions.

We present CAR-UCB, a model-free, cost-aware offloading framework that adapts redundancy levels online to meet real-time deadlines while minimizing resource consumption. CAR-UCB treats path selection as an online learning problem, enabling the robot to identify the smallest subset of interfaces needed for reliable execution under changing network conditions. A structured exploration mechanism efficiently searches the enormous combinatorial action space, and a lightweight surrogate model allows the system to scale to hundreds of candidate paths. We further provide theoretical guarantees, proving that CAR-UCB achieves no-regret convergence and remains robust under approximation and partial exploration.

Through extensive trace-driven simulation and real-world field experiments, CAR-UCB achieves 99.6% deadline reliability, matching state-of-the-art static redundancy, while reducing data usage and operational costs by up to 92%. Moreover, CAR-UCB remains stable under correlated network failures where existing methods degrade sharply, demonstrating that adaptive, cost-aware learning provides a practical and robust foundation for deadline-critical cloud robotic systems.

Note that the algorithm is able to work on a resource constraint device

Keywords

Cloud robotics, Latency reliability, Multi-interface communication, Probabilistic scheduling, Offloading, Cloud computing, Bayesian Optimization, Multi-arm Bandits

1 Introduction

Motivation. Modern robotics faces a fundamental tension between the rapidly growing computational demands of intelligent algorithms and the strict resource constraints of onboard hardware. State-of-the-art models for perception, planning, and control, such as Large Language Models (LLMs) [20, 43] and Vision–Language Models (VLMs) [13, 26], offer transformative capabilities, yet their scale and resource intensity make them impractical for direct deployment on typical robotic platforms. This computational gap has brought *cloud robotics* [18, 19] to prominence, enabling robots to offload demanding workloads to powerful, elastic cloud servers. Leveraging cloud resources not only improves task performance but also reduces hardware cost, energy consumption, and system complexity on the robot [10]. However, this paradigm introduces a critical dependency on external infrastructure: for many applications, particularly those involving mobile robots in real-world environments (e.g., warehouses, public roads, etc.), connectivity relies on commodity wireless networks such as Wi-Fi and 5G cellular services. These networks, while ubiquitous, are not designed to provide the stringent real-time guarantees required for mission-critical robotic tasks with strict deadlines [25, 29, 32]. At the same time, cloud servers are typically shared across tenants, where resource contention, such as the common *noisy neighbor problem* [23, 28, 41], can introduce additional latency variability. Together, these factors undermine the timely completion of offloaded tasks.

Therefore, the central challenge is to ensure *deadline reliability* in cloud robotics offloading systems, despite fundamentally unreliable infrastructures.

State of the art. The *Impossibility Triangle* for Latency reliability, Singleton deployment, and Commodity infrastructure (LSC) [6] establishes that these three properties cannot be achieved simultaneously. In the context of cloud robotics, where robots typically rely on unreliable infrastructure, achieving latency reliability requires moving beyond singleton deployment. Recent research has therefore focused on redundancy-based strategies, such as equipping robots with multiple network interfaces (e.g., Wi-Fi and 5G) and replicating requests across geographically distributed cloud servers, using the first response that arrives [5–7, 42]. While the principle is sound, the methods for deciding which communication paths to use at any given moment fall into two main categories, each with significant limitations:

- **Heuristic-based strategies.**
- **Model-based optimization.**

Talk about congestion if flooding

Limitations of Model-Based Approaches. Recent work, such as FogROS2-PLR [6], FogROS2-FT [5], addresses deadline-aware offloading using online-estimated probabilistic models (e.g., Gaussian Mixture Models). These methods compute the deadline-hit probability $p_{jk} = \Pr(\text{latency}(i_j, s_k) \leq d)$ for each interface–server pair, enabling threshold-based task allocation. While adaptive, these approaches still depend on the accuracy and stability of explicit latency models.

In real-world deployments, however, latency is influenced by highly dynamic factors such as network congestion, routing changes, and cloud-side variability. Maintaining robust, accurate models under such conditions is difficult, even for adaptive systems. As noted in [2], latency in wireless networks often exhibits heavy tails and high variance due to queuing effects, bursty interference, and channel fading, making even sophisticated models brittle under strict deadline constraints.
Model-Free Approach. In this paper, we move beyond explicit statistical latency models and propose a *model-free*, learning-based framework for deadline-aware offloading. Instead of estimating latency distributions, we treat the offloading system as a black box and learn, from online experience, which combinations of interface–server pairs are most likely to meet deadline constraints.

We formulate the problem as *Bayesian Optimization over Cost-Varying Subsets* (BOCVS) [16, 39]. At each decision step, the robot selects a subset of interface–server pairs under cost, energy, or bandwidth constraints and observes a binary outcome: whether at least one response arrived before the deadline. Over time, the system adapts to changing conditions, learning to prefer subsets that maximize deadline success.

Our approach generalizes existing model-based strategies like FogROS2-PLR. It supports arbitrary subset selection, handles heterogeneous cost profiles, and adapts to dynamic conditions without requiring latency calibration or profiling.
Problem Statement. *Given a set of edge/cloud servers with unknown and time-varying latency, how should a robot offload time-sensitive tasks by selecting subsets of servers to maximize the empirical probability that at least one response arrives before a deadline, subject to cost or replication constraints?*

The problem captures both *task redundancy* (via subset selection) and *tail-latency-awareness*. For each subset $S' \subseteq S$, the system observes a binary feedback signal (success/failure) and must learn an optimal policy under resource constraints.
Key Insights and Contributions. Bayesian Optimization (BO) is well-suited to this setting due to its sample efficiency and robustness to noisy black-box functions. These properties are critical in robotics, where each offloading attempt incurs cost, latency, and potential safety implications. Learning from binary outcomes in a cost-constrained setting provides a powerful and flexible alternative to explicit latency modeling.

This paper makes the following **key contributions**:

- We formulate the deadline-constrained offloading problem in cloud robotics as a black-box subset selection task, capturing latency uncertainty and real-world cost constraints such as bandwidth or energy budgets.
- We We propose a Bayesian optimization-based model-free learning of deadline-reliable offloading strategies without requiring explicit latency modeling or system calibration, that driven from Bayesian optimization with cost-varying variable subsets [39] but customized for performing in a resource constraint robot. The algorithm reduce the comlexity from $O(t^3)$ to $O(I^2 t)$ with $M \ll t$ (see Section 4).
- We evaluate the approach through both simulation (Chanh: do not know yet but aim for ...and a real-world cloudrobotic testbed with of 32 robot arm cells[1]), showing that BOCVS outperforms static (e.g., always-local, threshold-based) (Chnh: will compare with model-based baselines (e.g., FogROS2-PLR)) in terms of deadline reliability and adaptability in dynamic environments.

To our knowledge, this is the first framework that combines GPC-based uncertainty modeling with a cost-aware optimistic selection rule to enable deadline-reliable, decentralized offloading across an exponential action space.

2 Related Work

A related direction is distributed no-regret learning in multi-stage systems with end-to-end bandit feedback, most notably the ϵ-EXP3 framework [17]. In this setting, a job traverses a tree of decision nodes, each choosing a single child and observing only the final leaf cost, leading to an exploration–exploitation –education trilemma. Our problem is structurally different: we select combinatorial subsets/paths rather than sequential single-child actions, optimize deadline reliability and latency distributions rather than scalar leaf costs, and operate with a single learner without downstream agents requiring "education". Moreover, our approach uses Bayesian optimization with sparse GPs to exploit correlation across subsets, whereas ϵ-EXP3 relies on unstructured adversarial-bandit updates.

Chinchali et al. [7] propose a deep RL framework for deciding whether a robot should offload perception tasks to the cloud or process them locally under variable network conditions . Their approach is related in its use of online learning for offloading, but it considers only a single binary offload decision and assumes a fixed cloud endpoint. In contrast, our setting allows a robot to offload to multiple heterogeneous edge and cloud servers, each with different latency, reliability, and cost characteristics. The learner must therefore adaptively select a subset of available execution paths to satisfy deadline-reliability constraints.

3 Problem statement

3.1 Ground set and Action space

Assume a system for robot offloading with a set of M network interfaces and N available servers. We define the ground set $\mathcal{P}$ as the set of all $K = N \times M$ possible atomic communication paths:

$$\mathcal{P} = \{(s_i, n_j) \mid i \in [1, N], j \in [1, M]\} \tag{1}$$

We define the full action space $\mathcal{A}$ as the entire collection of all possible actions, i.e., $\mathcal{A} = 2^{\mathcal{P}}$. At each decision point t, the agent selects an action $A_t \in \mathcal{A}$, i.e., a *subset* $A_t \subseteq \mathcal{P}$ of atomic paths to execute in parallel. The size of the action space is exponential in the number of atomic paths, $|\mathcal{A}| = 2^K$, making exhaustive search or simple exploration intractable.

[1]https://cloudgripper.org/

3.2 Cost function

Every action has an associated cost, representing resource consumption. The cost of an action A_t is the sum of the costs of its individual paths:

$$C(A_t) = \sum_{(i,j)\in A_t} c_{ij} \tag{2}$$

There exists a natural trade-off: larger subsets may offer greater reliability but incur higher costs $C(A_t)$.

It is important to note that $C(A_t)$ represents the *a priori* resource consumption cost, i.e., the price of taking the action, distinct from the *a posteriori* penalty cost incurred when a request fails. In the next subsection, we introduce the reliability-based objective function, treated as a black-box in our learning problem, that quantifies the probability that an action meets the latency deadline.

3.3 Black-box objective function

The central challenge lies in the unknown relationship between an action and its outcome. Let L_{ij} be the random variable for the end-to-end latency over path (i,j), whose realization is stochastic due to congestion, server load, and other time-varying effects.

When the agent executes all paths in A_t in parallel and accepts the first completion, the system behaves as a *first-to-respond race model* [33]. Hence, the resulting action latency is:

$$L(A_t) = \min_{(i,j)\in A_t} L_{ij} \tag{3}$$

Given a hard latency deadline τ, we define the success event as:

$$S(A_t) = \{L(A_t) \le \tau\} \tag{4}$$

Our goal is to maximize the probability of this event. We thus define the objective function [35] as:

$$f(A_t) = \mathbb{P}(S(A_t)) = \mathbb{P}\left(\min_{(i,j)\in A_t} L_{ij} \le \tau\right) \tag{5}$$

Here, $f(A_t)$ is a black-box function that maps an action (i.e., a subset of paths) to its probability of success. Its analytical form is unknown and non-differentiable because it depends on the complex, non-stationary, and potentially correlated joint latency distributions across all paths. As a result, $f(A_t)$ cannot be evaluated directly, and the agent must instead infer it from observed feedback.

3.4 Observation model with binary feedback

After selecting and executing an action A_t, the agent does not observe the continuous latency $L(A_t)$ nor the true underlying probability $f(A_t)$. Instead, the agent observes a single binary outcome $y_t \in \{0,1\}$, indicating the success or failure of the offloading attempt:

$$y_t = \begin{cases} 1, & \text{if } L(A_t) \le \tau \quad \text{(Success)}, \\ 0, & \text{if } L(A_t) > \tau \quad \text{(Failure)} \end{cases} \tag{6}$$

It follows that y_t is a sample drawn from a Bernoulli distribution whose parameter corresponds to the unknown success probability of the chosen action $f(A_t)$, i.e., $y_t \sim \text{Bernoulli}(f(A_t))$. This binary feedback model is what necessitates the use of Gaussian Process Classification in our proposed solution in Section 4.

3.5 Optimization goal and regret formulation

The agent must select an action $A \in \mathcal{A}$ that achieves high success probability $f(A)$ while incurring a low execution cost $C(A)$. To assess performance over time, we adopt a benchmark-based metric that quantifies how far the agent's choices deviate from the best possible one.

We first define the reliability benchmark representing the highest success probability achievable by any action, regardless of cost:

$$f^* = \max_{A\in\mathcal{A}} f(A) \tag{7}$$

Upon choosing action A_t at decision step t, the agent incurs a reliability loss of $f^* - f(A_t)$, which quantifies the performance sacrificed by deviating from the optimal action.

Because poor decisions are more harmful when they are also expensive, we scale this gap by the action cost and define the *instantaneous cost-weighted loss*:

$$r_t = C(A_t)\big(f^* - f(A_t)\big) \tag{8}$$

Aggregating r_t over T decision steps gives the *cost-varying cumulative regret*:

$$R_T = \sum_{t=1}^{T} r_t = \sum_{t=1}^{T} C(A_t)\big(f^* - f(A_t)\big) \tag{9}$$

Here, R_T captures the dual nature of our objective: the agent is penalized for choosing an unreliable action, and this penalty grows when the chosen action incurs high cost $C(A_t)$. In this sequential online decision-making setting, a procedure is said to be *"no-regret"* if its expected average cumulative regret converges to zero [14, 16, 37, 39]:

$$\lim_{T\to\infty} \frac{\mathbb{E}[R_T]}{T} = 0 \tag{10}$$

where $\mathbb{E}[R_T]$ denotes the expected cost-varying cumulative regret.

We use this criterion as the formal benchmark for evaluating decision-making strategies in our setting.

4 Decentralized Cost-Aware Bayesian Optimization for Reliable Multi-Agent Offloading

The exponential size of the action space ($|\mathcal{A}| = 2^K$) and the critical need to balance mission reliability against execution cost make exhaustive evaluation infeasible. We address this challenge with a structured learning framework in which each robot operates as an independent, decentralized agent. The framework efficiently infers the utility of candidate actions from binary feedback and prioritizes exploration toward the

Table 1: Summary of notation used.

Symbol	Meaning
$\mathcal{A}$	Action space (set of all offloading subsets)
A_t	Action selected at time t
$C(A)$	Cost of action A
$c_{\min}$	Minimum cost among a candidate set
$f(A)$	Success probability of action A
$g(A)$	Latent function value prior to probit link
$\Phi(\cdot)$	Gaussian CDF (probit link)
y_t	Binary observation (1 = success, 0 = failure)
$\mathcal{D}_t$	Data collected up to time t
$\mu_t(A)$	Posterior mean of $g(A)$ at time t
$\sigma_t(A)$	Posterior standard deviation of $g(A)$ at time t
$U_t(A)$	UCB acquisition value of A
β_t	Exploration parameter (controls optimism)
ε_t	Tolerance parameter (near-optimality filter)
γ	Fixed margin constant used in ε_t
r_t	Instantaneous regret
R_T	Cumulative regret over horizon T

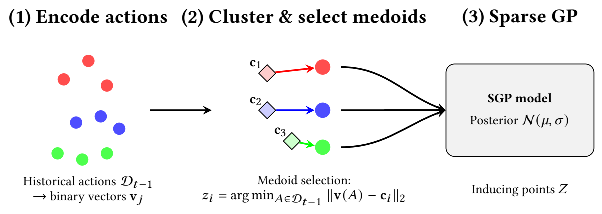

(1) Encode actions
(2) Cluster & select medoids
(3) Sparse GP
c_1
c_2
c_3
SGP model
Posterior $\mathcal{N}(\mu, \sigma)$
Historical actions $\mathcal{D}_{t-1}$ $\rightarrow$ binary vectors $\mathbf{v}_j$
Medoid selection: $z_i = \arg\min_{A \in \mathcal{D}_{t-1}} \|\mathbf{v}(A) - \mathbf{c}_i\|_2$
Inducing points Z

Figure 1: Inducing-point construction pipeline: (1) encode historical actions as binary vectors, (2) cluster and select K-medoids representatives Z, and (3) use Z as inducing points in the sparse GP model.

most promising, cost-effective regions of the combinatorial space.

Our solution combines two key components. First, we model the unknown success probability $f(A)$ using Sparse Gaussian Process Classification (SGP-C) [12, 22], a scalable Bayesian model for learning from binary observations. Second, we couple this surrogate model with a cost-aware acquisition mechanism based on the upper confidence bound for cost-varying variable subsets rule [39], which guides efficient exploration within a dynamically restricted subset of the action space.

Details of the proposed approach follow.

4.1 Surrogate model: Sparse gaussian process classification

The first challenge is to learn the unknown success–probability function $f : \mathcal{A} \rightarrow [0, 1]$ from binary observations $y_t \in \{0, 1\}$. Standard Gaussian Process (GP) regression is unsuitable here because it assumes continuous, noisy observations of the target function. We therefore adopt Gaussian Process Classification (GPC) [27], a Bayesian non-parametric model explicitly designed for binary feedback.

GPC introduces a latent, real-valued function $g(A)$ on which we place a GP prior:

$$g(A) \sim \mathcal{GP}\big(\mu_0(A),\ k(A, A')\big) \tag{11}$$

where μ_0 is a prior mean function and k is a kernel that encodes similarity between pairs of actions A and A'.

To convert the latent function $g(A)$ into a probability of success, we apply the Gaussian cumulative distribution function (CDF) [8], denoted Φ, also known as the *probit link* in classification models [4, 11]:

$$f(A) \approx \Phi(g(A)) \tag{12}$$

By this mapping, large positive values of $g(A)$ correspond to high success probabilities, while large negative values correspond to low success probabilities.

Given the dataset of observations $\mathcal{D}_{t-1}$ collected up to the current time step, the agent maintains an approximate posterior distribution over the latent function. While standard GPC provides a sound probabilistic model, its exact inference scales cubically with the number of observations, i.e., $O(|\mathcal{D}_{t-1}|^3)$, making it impractical for deployment on resource-constrained robots performing continuous online learning.

To overcome this limitation, we adopt a Sparse Gaussian Process (SGP) approximation with I inducing points $Z = \{z_1, \ldots, z_I\}$, where $I \ll |\mathcal{D}_{t-1}|$. As a result, these inducing points provide a compact summary of the history and reduce the computational complexity to $O(I^2|\mathcal{D}_{t-1}|)$.

Constructing discrete inducing points. In classical SGPs [12, 40], inducing locations are optimized continuously through variational inference. However, our input domain consists of discrete combinatorial actions, making such gradient-based optimization inapplicable. Instead, we construct Z by selecting representative historical actions using the K-medoids–style procedure [34] illustrated in Figure 1.

First, each action A_j is encoded as a multi-hot vector $\mathbf{v}_j \in \{0, 1\}^K$, where K is the number of atomic paths. K-Means clustering is then applied to the set $\{\mathbf{v}_j\}$ to identify I cluster centroids. Since these centroids do not generally correspond to valid actions, each is projected back to the discrete domain by selecting the nearest historical action:

$$z_i = \arg\min_{A \in \mathcal{D}_{t-1}} \|\mathbf{v}(A) - \mathbf{c}_i\|_2^2 \tag{13}$$

To keep computation low, we update the inducing set Z only periodically (e.g., every 50 steps).

Under this approximation, the agent's posterior over the latent function is

$$g(A) \mid \mathcal{D}_{t-1}, Z \approx \mathcal{N}\big(\mu_{t-1}(A),\ \sigma^2_{t-1}(A)\big) \tag{14}$$

The posterior mean–variance pair $(\mu_{t-1}, \sigma^2_{t-1})$ constitutes the agent's scalable *surrogate model* of the environment and provides the uncertainty estimates required for the CAR-UCB strategy described in section 4.2.

4.2 Acquisition function: Cost-aware action selection

Given the SCGP-C surrogate model, the agent must decide which action A_t to select next. This is governed by the acquisition function, which balances exploration (i.e., reducing uncertainty) and exploitation (i.e., choosing promising actions) while respecting cost. To guide action selection, we extend the Upper Confidence Bound (UCB) principle [1, 36, 39] to a cost-aware acquisition rule tailored to our combinatorial action space and binary feedback model.

4.2.1 Informed action candidate generation. The combinatorial action space $|\mathcal{A}| = 2^K$ is intractable to search exhaustively. To ensure real-time operation, we impose a candidate budget B that limits the number of actions evaluated at each step. Thus, instead of scanning the full space, the agent constructs a compact set $\mathcal{A}_{\text{cand}} \subset \mathcal{A}$ with $|\mathcal{A}_{\text{cand}}| \approx B \ll |\mathcal{A}|$.

The candidate set combines three sources:

$$\mathcal{A}_{\text{cand}} = \mathcal{A}_{\text{historical}} \cup \mathcal{A}_{\text{local}} \cup \mathcal{A}_{\text{random}} \tag{15}$$

Historical actions. $\mathcal{A}_{\text{historical}}$ contains the best action observed so far and the action executed at time $t-1$, ensuring continuity and exploitation of known-good behaviors.

Local refinement. $\mathcal{A}_{\text{local}}$ explores the neighborhood of the top-k historical actions (e.g., $k = 3$). For each action, we apply single-step modifications: *expansion* $A \cup \{p\}$ adds a path to improve reliability, and *contraction* $A \setminus \{p\}$ removes a path to reduce cost. These $O(K)$ local perturbations efficiently probe nearby high-value regions.

Cost-weighted exploration. $\mathcal{A}_{\text{random}}$ introduces newly sampled actions to guarantee coverage of the combinatorial space. To prevent costly and unproven actions from dominating this random set, each path $p \in \mathcal{P}$ is included in a randomly generated subset A with probability

$$P_{\text{inc}}(p) \propto \frac{1}{c(p)} \tag{16}$$

thereby biasing exploration toward cheaper paths and avoiding the creation of unnecessarily expensive action candidates.

Algorithm 1 details the construction of the candidate set $\mathcal{A}_{\text{cand}}$. Restricting the search to $\mathcal{A}_{\text{cand}}$ turns the infeasible $O(2^K)$ search over $\mathcal{A}$ into $O(B)$ work per step. The CAR-UCB rule introduced in the following operates only on $\mathcal{A}_{\text{cand}}$.

4.2.2 Optimistic value of an action. For any candidate action $A \in \mathcal{A}_{\text{cand}}$, we construct an optimistic estimate by utilizing the approximate posterior mean $\mu_{t-1}(A)$ and variance $\sigma_{t-1}(A)$ provided by the SGP-C model. The UCB score on the latent function is:

$$u_{t-1}(A) = \mu_{t-1}(A) + \beta_t \sigma_{t-1}(A) \tag{17}$$

where β_t is an exploration parameter that balances exploitation (mean) and exploration (uncertainty).

Since the probit function $\Phi(\cdot)$ is monotonically increasing, the optimistic estimate of our true objective $f(A)$ is obtained by applying the link function directly to this latent upper bound [15, 24]:

$$U_{t-1}(A) = \Phi(u_{t-1}(A)) \tag{18}$$

Here, $U_{t-1}(A)$ is the acquisition score, representing the agent's optimistic belief about the success probability of action A.

4.2.3 Cost-aware CAR-UCB rule. To incorporate cost into decision making, we propose a **C**ost-**A**ware **R**eliable **UCB** action selection strategy (CAR-UCB). CAR-UCB applies a two-stage filtering procedure that selects actions with high optimistic success probability while avoiding unnecessarily expensive choices. At each time step t, CAR-UCB proceeds as follows:

```
Require: History $\mathcal{D}_{t-1}$; budget $B$; path set $\mathcal{P}$; cost map $c$; local scope $k$
Output: Candidate set $\mathcal{A}_{\text{cand}}$ with $|\mathcal{A}_{\text{cand}}| \approx B$
1:  $\mathcal{A}_{\text{cand}} \leftarrow \emptyset$
2:  // Step 1: Historical exploitation
3:  $\mathcal{A}_{\text{cand}} \leftarrow \mathcal{A}_{\text{cand}} \cup \{A_{\text{best\_so\_far}}, A_{t-1}\}$
4:  // Step 2: Local search (expansion and contraction)
5:  $\mathcal{H}_{\text{top}} \leftarrow$ top-$k$ unique actions from $\mathcal{D}_{t-1}$ ranked by success rate
6:  for $A \in \mathcal{H}_{\text{top}}$ do
7:      for $p \in \mathcal{P}$ do
8:          if $p \notin A$ then
9:              $\mathcal{A}_{\text{cand}} \leftarrow \mathcal{A}_{\text{cand}} \cup \{A \cup \{p\}\}$    ▷ Expand
10:         else if $|A| > 1$ then
11:             $\mathcal{A}_{\text{cand}} \leftarrow \mathcal{A}_{\text{cand}} \cup \{A \setminus \{p\}\}$    ▷ Contract
12:         end if
13:     end for
14: end for
15: // Step 3: Cost-weighted random exploration
16: $N_{\text{need}} \leftarrow B - |\mathcal{A}_{\text{cand}}|$
17: if $N_{\text{need}} > 0$ then
18:     Compute inclusion probabilities: $P_{\text{inc}}(p) \propto 1/c(p)$ for all $p \in \mathcal{P}$
19:     for $i = 1$ to $N_{\text{need}}$ do
20:         $A_{\text{rand}} \leftarrow \emptyset$
21:         for $p \in \mathcal{P}$ do
22:             Add $p$ to $A_{\text{rand}}$ with probability $P_{\text{inc}}(p)$
23:         end for
24:         if $A_{\text{rand}} \neq \emptyset$ then
25:             $\mathcal{A}_{\text{cand}} \leftarrow \mathcal{A}_{\text{cand}} \cup \{A_{\text{rand}}\}$
26:         end if
27:     end for
28: end if
29: return $\mathcal{A}_{\text{cand}}$
```

Algorithm 1: Generate_Candidates $\mathcal{A}_{\text{cand}}$

(1) **Optimistic feasibility filter.** Let $g_t = \max_{A \in \mathcal{A}_{\text{cand}}} U_{t-1}(A)$ be the highest optimistic estimate over all actions. Actions whose optimistic value lies within a tolerance ϵ_t of this maximum form the feasible set:

$$S_1 = \{ A \in \mathcal{A}_{\text{cand}} \mid U_{t-1}(A) + \epsilon_t \geq g_t \} \tag{19}$$

After this filtering stage, the remaining actions are those that are potentially optimal under the current model uncertainty.

(2) **Cost filter.** Among the feasible actions, the algorithm keeps only the lowest-cost ones:

$$S_2 = \left\{ A \in S_1 \;\middle|\; C(A) = \min_{A' \in S_1} C(A') \right\} \tag{20}$$

This step enforces cost-awareness by preventing expensive actions from being chosen unless they are needed to remain competitive with the most promising candidates according to the UCB score.

Within the cost-minimal feasible set, CAR-UCB selects the action with the highest optimistic score:

$$A_t = \arg\max_{A \in S_2} U_{t-1}(A) \tag{21}$$

The agent then executes A_t, pays the cost $C(A_t)$, observes a single binary outcome y_t and updates its SGP-C posterior for the next step.

Algorithm 2 details the complete procedure. The algorithm operates in an online fashion in deployment, making decisions and incorporating feedback continuously. For experimental evaluation (see Section 7), we measure cumulative regret over a fixed horizon T following the standard practice in online learning and Bayesian optimization [16, 39].

```
Require: Action space $\mathcal{A} = 2^{\mathcal{P}}$; SGP-C prior $(\mu_0, \sigma_0)$;
    action cost function $C(\cdot)$; exploration schedule $\{\beta_t\}$;
    tolerance schedule $\{\varepsilon_t\}$; candidate budget $B$
Output: Online sequence of selected actions $\{A_t\}$
 1: for $t = 1, 2, \dots, T$ do
 2:     // Step 1: Generate candidate set
 3:     $\mathcal{A}_{\text{cand}} \leftarrow$ GENERATE_CANDIDATES$(\mathcal{D}_{t-1}, B)$
 4:     // Step 2: Optimistic estimates on $\mathcal{A}_{\text{cand}}$
 5:     for $A \in \mathcal{A}_{\text{cand}}$ do
 6:         $u_{t-1}(A) \leftarrow \mu_{t-1}(A) + \beta_t \sigma_{t-1}(A)$
 7:         $U_{t-1}(A) \leftarrow \Phi(u_{t-1}(A))$
 8:     end for
 9:     // Step 3: Best optimistic value
10:     $g_t \leftarrow \max_{A \in \mathcal{A}_{\text{cand}}} U_{t-1}(A)$
11:     // Step 4: Near-optimal set
12:     $S_1 \leftarrow \{A \in \mathcal{A}_{\text{cand}} : U_{t-1}(A) + \varepsilon_t \geq g_t\}$
13:     // Step 5: Cheapest near-optimal actions
14:     $c_{\min} \leftarrow \min_{A \in S_1} C(A)$
15:     $S_2 \leftarrow \{A \in S_1 : C(A) = c_{\min}\}$
16:     // Step 6: Select action
17:     $A_t \leftarrow \arg\max_{A \in S_2} U_{t-1}(A)$
18:     // Step 7: Execute and update model
19:     Execute $A_t$ and observe $y_t \in \{0, 1\}$
20:     $\mathcal{D}_t \leftarrow \mathcal{D}_{t-1} \cup \{(A_t, y_t)\}$
21:     Update sparse posterior $(\mu_t, \sigma_t^2)$ using $\mathcal{D}_t$
22: end for
```

Algorithm 2: CAR-UCB: Cost-Aware Reliable UCB Action Selection

4.3 Theoretical schedule parameters choices

Our learning framework relies on two scheduling rules adapted from the Bayesian optimization literature [8, 37–39] to ensure that the no-regret guarantees hold.

Exploration Schedule (β_t). To obtain valid high-probability confidence intervals, the exploration schedule β_t must be non-decreasing and large enough to ensure that the latent function $g(A)$ remains within the GP posterior confidence region. Formally, for a confidence parameter $\delta \in (0, 1)$, it is required that:

$$\Pr[\, g(A) \in \mu_{t-1}(A) \pm \beta_t\, \sigma_{t-1}(A),\ \forall A \in \mathcal{A},\ t \geq 1] \geq 1 - \delta \tag{22}$$

Consistent with the theoretical foundations of GP-UCB [37] and BOCVS [39], the high-probability requirement above is satisfied by adopting a logarithmically growing exploration schedule. Specifically, choosing

$$\beta_t = \sqrt{2 \log(1 + t)} \tag{23}$$

is sufficient to guarantee that the GP confidence bounds hold with probability at least $1 - \delta$, thereby ensuring sublinear regret.

Importantly, β_t is **not** treated as a tunable hyperparameter in our implementation. Its growth rate is prescribed by the theory: smaller schedules may break the confidence guarantee, while larger ones only slow convergence without changing the qualitative behavior. For this reason, we fix the above choice of β_t in all experiments and do not include it in the ablation analysis.

Tolerance Schedule (ε_t). The tolerance parameter ε_t controls how aggressively the algorithm prunes the action space when constructing the candidate set S_1. We adopt a standard decaying schedule:

$$\varepsilon_t = \frac{\gamma}{\sqrt{\log(1 + t)}} \tag{24}$$

where γ determines the initial width of the exploration region.

The regret guarantee holds for any fixed $\gamma > 0$, while values in the range $\gamma \in [0.5, 2.0]$ are typically sufficient to ensure stable filtering [8, 38].

Unlike β_t, the constant $\gamma > 0$ is a genuine design parameter that trades off early exploration against pruning. We therefore perform an ablation study over γ, which directly induces different ε_t schedules to understand how this choice influences the behavior of the algorithm (see Section 7.1.5).

5 Theoretical Guarantees

This section establishes two fundamental guarantees of the proposed framework. First, we show that the cost-aware CAR-UCB action-selection rule achieves *no-regret* performance in expectation, ensuring asymptotic optimality in selecting low-cost, high-reliability actions. Second, we demonstrate that when the algorithm is deployed independently across many robots without communication or coordination, the fleet exhibits *stable, non-synchronized* behavior and avoids the classical thundering-herd instability.

5.1 No-Regret Optimality

The regret analysis of CAR-UCB is conducted under two standard assumptions characterizing the approximate posterior and the restricted action optimization used by the algorithm:

1. **Approximate posterior accuracy** (inducing points Z). We assume that the Sparse GP posterior constructed with inducing points Z satisfies the usual concentration properties required for regret analysis of approximate GP models.
2. **Approximate action optimization** (candidate budget B and local scope k). We assume that the candidate-generation mechanism (Algorithm 1) provides a sufficiently dense covering of the action space such that the selected action A_t approximates the maximizer of the acquisition function. Formally, the optimization error introduced by restricting to $\mathcal{A}_{\text{cand}}$ is assumed to be bounded or negligible relative to the exploration bonus.

Under these assumptions, CAR-UCB admits the following guarantee.

Lemma 5.1 (No-Regret Guarantee in Expectation). *Under standard regularity assumptions on the Gaussian Process prior, with exploration and tolerance schedules $\{\beta_t\}$ and $\{\varepsilon_t\}$ chosen as presented in Section 4.3, and given the use of the Sparse Gaussian Process approximation and the fixed Candidate Budget B, the CAR-UCB algorithm achieves asymptotic optimality in expectation, satisfying equation (10).*

Proof Sketch. The proof adapts standard GP-UCB analysis to the Sparse GP setting, while accounting for cost-aware pruning through ε_t. The key idea is that (1) the approximate GPC posterior contracts sufficiently under the realizability assumption, (2) the confidence bonus β_t and the relaxation ε_t are scheduled at sublinear rates to ensure decreasing uncertainty, and (3) the cumulative error from the sparse approximation is bounded, ensuring that the expected average regret converges to zero. Full details are given in Appendix A. □

Remark on Approximation Errors. Lemma 5.1 establishes convergence under idealized assumptions. In Appendix B, we relax these assumptions by explicitly quantifying the errors introduced by the Sparse GP approximation (Δ_{SGP}) and the restricted candidate search (Δ_{opt}). The regret decomposition shows that these terms enter additively, yielding the bound

$$\frac{\mathbb{E}[R_T]}{T} = O\left(\frac{\sqrt{\gamma_T}}{\sqrt{T}}\right) + O(\Delta_{\text{SGP}}) + O(\Delta_{\text{opt}})$$

Consequently, even under approximation, the system exhibits *graceful degradation*: while the vanishing term disappears as $T \to \infty$, the average regret converges to a constant floor determined by the approximation quality:

$$\lim_{T\to\infty} \frac{\mathbb{E}[R_T]}{T} \le C_{\max}\,(2\Delta_{\text{SGP}} + \Delta_{\text{opt}})$$

This formalizes the intuition that CAR-UCB remains near-optimal as long as the surrogate model and the candidate generator are reasonably accurate.

5.2 Decentralized stability and herd avoidance

Each robot r runs CAR-UCB using only its own observation history $\mathcal{D}_t^{(r)}$. When the end-to-end path latency fails, the instantaneous success probability $f(A)$ for actions A involving that congested path becomes unstable. Each robot then observes an independent Bernoulli outcome.

Because these observations are independent across robots, their histories $\mathcal{D}_t^{(r)}$ differ with high probability, and so do their posteriors $(\mu_{t-1}^{(r)}, \sigma_{t-1}^{(r)})$. As a result, the optimistic scores

$$U_{t-1}^{(r)}(A) = \Phi\Big(\mu_{t-1}^{(r)}(A) + \beta_t \sigma_{t-1}^{(r)}(A)\Big) \tag{25}$$

and the filtered sets $S_1^{(r)}, S_2^{(r)}$ also differ across robots.

In particular, the decision to stop exploiting a congested resource is driven by a robot-specific threshold on $U_{t-1}^{(r)}(A)$ compared to alternatives. Since these individual thresholds are crossed at different times for different robots due to desynchronized observations, the probability that all robots simultaneously drop actions using s in the same round is negligible.

Under CAR-UCB, increased uncertainty on congested actions inflates the exploration bonus $\beta_t \sigma_{t-1}^{(r)}(A)$, so different robots are steered toward diverse low-cost alternatives in the action space $\mathcal{A}$. This produces a staggered, diversified migration away from the congested resource rather than a synchronized stampede. As load on the original server decreases, its performance recovers, its uncertainty shrinks, and CAR-UCB gradually reintroduces actions involving that path into the candidate sets, yielding a dynamic but stable equilibrium rather than persistent oscillation.

We empirically confirm this emergent behavior in Section 7.1.4.

6 Experiment Setting

In this section, we present the experimental setting used to evaluate the proposed solution. We begin by outlining the state-of-the-art baseline methods that tackle the same problem. We then describe the evaluation metrics used to compare these approaches and quantify their performance.

6.1 Baseline

PLR. Motivated by the probabilistic redundancy strategy to reduce deadline misses in FogROS2-PLR [6], we implement a more realistic PLR-style baseline. Unlike the original one-interface–one-server assignment in FogROS2-PLR, real robotic systems allow a single interface (e.g., WiFi or 5G) to send requests to multiple servers; thus, we treat each interface–server path independently.

For each path p, PLR maintains an empirical estimate of the probability of meeting the deadline, $q_p = \Pr(L_p \leq \tau)$ computed from per-path success counts. Assuming independent failures, the estimated reliability of a replica set A is $R(A) = 1 - \prod_{p \in A}(1 - q_p)$. At each timestep, the approach selects the smallest replica set whose estimated reliability exceeds a target ρ. If no such set exists, PLR falls back to the set A with the highest estimated $R(A)$.

FT-Flood. FogROS2-FT [5] uses a fixed small replica set, which is unstable under dynamic latency: if those few paths become slow, FT cannot adapt and suffers persistent deadline misses. To provide a meaningful and stronger baseline, we use a flood variant that sends each request to all interface–server paths and returns the earliest response.

Oracle. We implement an *unimplementable* upper-bound baseline that assumes perfect knowledge of future latencies. At each decision step, the Oracle identifies the set of paths satisfying the latency deadline (τ). From this feasible set, it selects the path with the minimum cost. This baseline provides an ideal performance ceiling against which realistic algorithms can be compared.

The specific hyperparameter settings for all evaluated algorithms are detailed in Table 2.

Table 2: Hyperparameter settings for the evaluated algorithms.

Algorithm	Parameter	Value
CAR-UCB	History window W	100
	Local search k	2
	Exploration γ	2.0
	Max subset size	2 paths
	Candidate Budget (B)	$1.5 \times K$
PLR	Reliability target	0.90
	Subset sizes	1–2 paths

6.2 Evaluation metrics

We evaluate all methods using latency- and robustness-oriented metrics commonly used in networked systems and cloud robotics.

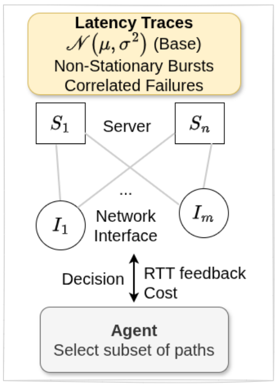
- **Success rate (higher is better).** The fraction of requests that meet the latency deadline τ:

$$\text{SuccessRate} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{1}[L_t \leq \tau] \tag{26}$$

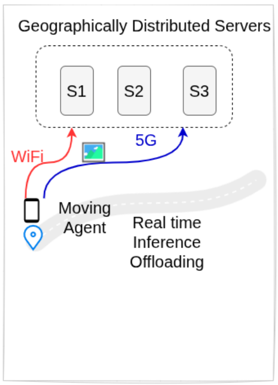
  This metric captures deadline reliability, which is essential for real-time robotic workloads.
- **95th percentile latency (P95) (lower is better).** A standard tail metric indicating the latency below which 95% of requests complete. Lower P95 reflects improved tail performance.
- **CVaR95 (lower is better).** The conditional value-at-risk at the 95th percentile, defined as the mean latency of the slowest 5% of requests. CVaR95 captures the severity of rare but extreme tail events that harm real-time control loops.

To quantify decision quality and behavioral consistency, we also report:

- **Cumulative Cost-Weighted Regret (R_T).** We consider a reliability benchmark of $f^* = 1$, representing perfect success probability. At each time step t, the instantaneous loss is weighted by the action cost $C(A_t)$.The cumulative regret over a horizon T is defined as $R_T = \sum_{t=1}^{T} C(A_t)\left(f^* - f(A_t)\right)$, where $f(A_t)$ is a binary success indicator (i.e., 1 if the deadline is met, 0 otherwise). This metric penalizes unreliable decisions proportionally to their cost, thereby capturing both reliability and cost awareness. Lower regret indicates better per-timestep decision quality.
- **Cost Efficiency.** This metric measures the average replication cost per request. Each path p has an associated cost c_p, and the cost of an action A_t is defined as $C(A_t) = \sum_{p \in A_t} c_p$. Lower values indicate that the algorithm achieves reliability without relying on unnecessary replication.



(a) Trace-driven Simulation


Geographically Distributed Servers
S1
S2
S3
WiFi
5G
Moving Agent
Real time Inference Offloading

(b) Field Deployment

Figure 2: Summary of the experimental methodologies.

6.3 Experimental Methodology

We evaluate the approaches through two complementary methodologies: (1) a trace-driven simulation to study scalability and behavior under controlled non-stationary network conditions, and (2) a real-world field deployment to validate robustness under practical wireless dynamics.

For illustration, figure 2 summarizes the experimental methodology.

6.3.1 Simulation Environment. We build a trace-driven simulator with M interfaces and N servers, creating a total of $K = M \times N$ potential communication paths.

Latency and Cost Model. We model each interface–server path as a non-stationary latency process with 500-step phases. In every phase, 20% of the paths operate in a good state with latencies drawn from $\mathcal{N}(50, 8^2)$ ms, while the rest paths follow $\mathcal{N}(120, 8^2)$ ms. Additionally, to capture transient noise, each path experiences three independent random bursts that add +30 ms for 50 consecutive steps. Path costs are heterogeneous and sampled once from a discrete distribution $\{1, 2, 3\}$ with probabilities 0.3, 0.5, and 0.2, respectively.

Congestion Constraints. To model self-interference on shared wireless interfaces (e.g., CSMA/CA in WiFi [21] or scheduling queues in 5G [30]), we introduce an interface-level contention penalty. Consistent with Bianchi's analysis of medium access control [3], increasing the number of contending flows on a single interface triggers exponential backoff and queuing delays. Therefore, if an agent simultaneously selects $k > 1$ paths sharing the same physical interface, a latency penalty of $+30 \times (k - 1)$ ms is added to all packets on that interface. This magnitude is calibrated to match the stochastic burst size (i.e., 30 ms), ensuring that self-induced congestion delays are comparable to external environmental noise.

System Scale. We evaluate the algorithm across three scale regimes: small ($K = 6$), medium ($K = 50$), and large ($K = 200$), enabling both oracle comparison and high-dimensional stress testing.

Experimental Procedure. To account for the stochastic nature of the network environment and the exploration policies, all results are averaged over 20 independent trials with distinct random seeds.

Table 3 summarizes the parameter settings used in the simulation.

Simulations ran on a machine with a 12th Gen Intel Core i7–1260P (12 cores, 16 threads, up to 4.7 GHz), 15 GB RAM, running Ubuntu 22.04.3 LTS (Linux 6.8.0–88–generic).

6.3.2 Real-World Field Experiment. To evaluate performance under realistic wireless and mobility conditions, we conducted a field experiment that emulates a mobile robot traversing an urban environment while offloading computation over heterogeneous networks. We deployed an object-detection service on three geographically distributed servers reachable via both a public WiFi network and a 5G connection.

A Google Pixel 6 Pro served as the edge device, continuously capturing video frames and offloading them to the remote service. During a 2 km traversal aboard an urban bus, the device executed each path-selection algorithm in real time, dynamically selecting among the two network interfaces and three servers based on instantaneous network conditions to satisfy the latency deadline.

Table 3: Summary of simulation parameters modeling the dynamic network environment.

Parameter	Value / Distribution
Phase Length	500 timesteps
Good Paths Ratio	20% of total K
Good Latency Dist.	$\mathcal{N}(50, 8^2)$ ms
Bad Latency Dist.	$\mathcal{N}(120, 8^2)$ ms
Burst Events	3 independent events per path
Burst Magnitude	+30 ms (Duration: 50 steps)
Cost Distribution	30% (1.0), 50% (2.0), 20% (3.0)
System Scales (K)	6, 50, 200
Total Time Horizon (T)	5000 timesteps
Latency Deadline (τ)	100 ms
Congestion Penalty	30 ms

7 Results and Discussion

In this section, we report results from the simulation and real-world field deployment, which form the basis for evaluating the efficiency and performance of the proposed algorithm in comparison with state-of-the-art baselines.

7.1 Simulation Results

Table 4 reports the mean and standard deviation of all performance metrics, aggregated over 20 independent runs for each system configuration (i.e., K paths). All experiments use the environment configuration in Table 3 and the hyperparameter settings for CAR-UCB and PLR given in Table 2.

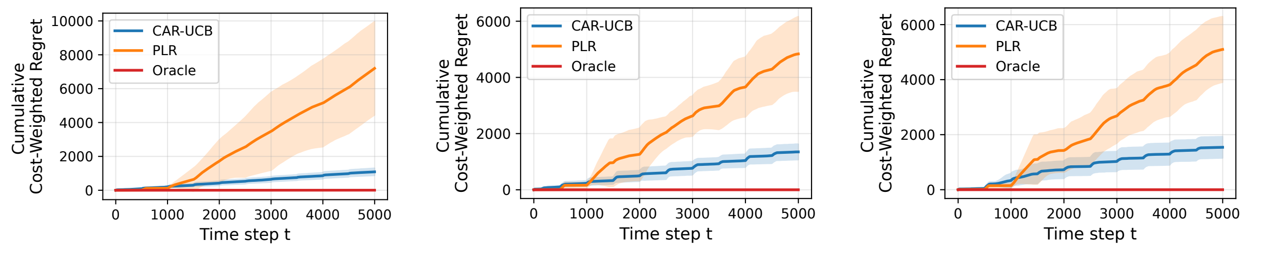
7.1.1 Smart Redundancy under Cost and Congestion. As expected, the Oracle (while impractical for real-time deployment, serves as an idealized upper bound) consistently achieves the highest success rate, reaching 100% in most scenarios. The only exception occurs in the small-scale setting, where the success rate drops marginally to 99.95% due to rare time steps in which no single path satisfies the strict deadline.

In contrast, FT-Flood highlights the drawbacks of naive over-provisioning. In the small-scale scenario ($K = 6$), excessive replication results in a success rate of only 10.28%. Performance degrades further as the system scales: in medium ($K = 50$) and large ($K = 200$) networks, the success rate collapses to zero. As shown in Table 4, the substantial signaling overhead (avg. cost 95.85 and 382.40, respectively) induces severe interface contention, increasing average latencies to 307 ms and 602 ms, well beyond the 100 ms deadline. Such behavior aligns with prior observations in distributed and wireless systems [9, 31], where excessive request duplication amplifies queuing delays and negates the benefits of path diversity.

CAR-UCB, by contrast, effectively balances redundancy and contention. Unlike PLR, whose success rate fluctuates between 62% and 74% due to limited adaptability to bursty interference, CAR-UCB maintains stable performance across scales. It consistently achieves a success rate of approximately

Table 4: Performance comparison across network scales. Scale: Small ($M = 2, N = 3, K = 6$), Medium ($M = 5, N = 10, K = 50$), and Large ($M = 10, N = 20, K = 200$). Results represent the mean ± standard deviation over 20 independent trials.

Algorithm	Success Rate (%)	Avg. Cost	Avg. Latency (ms)	P95 Latency (ms)	CVaR95 Latency (ms)
Small Scale ($M = 2, N = 3, K = 6$)					
CAR-UCB	91.15 ± 0.62	3.79 ± 0.54	64.61 ± 2.46	118.42 ± 0.95	128.21 ± 1.46
PLR	62.46 ± 9.96	3.33 ± 0.71	82.51 ± 5.71	140.76±10.44	146.70 ± 8.88
FT-Flood	10.28 ± 0.49	12.00 ± 1.48	111.01 ± 0.49	126.97 ± 2.29	136.36 ± 4.41
Oracle	99.95 ± 0.14	2.06 ± 0.31	51.45 ± 0.54	69.23 ± 3.42	81.84 ± 4.64
Medium Scale ($M = 5, N = 10, K = 50$)					
CAR-UCB	91.22 ± 1.28	3.75 ± 0.37	60.96 ± 1.44	117.10 ± 1.96	128.39 ± 1.79
PLR	73.86 ± 4.92	2.78 ± 0.48	74.05 ± 2.96	145.52 ± 4.68	151.22 ± 2.50
FT-Flood	0.00	95.85 ± 4.78	307.82 ± 0.05	315.02 ± 0.11	316.57 ± 0.11
Oracle	100.00	1.01 ± 0.03	44.39 ± 1.31	56.53 ± 2.32	62.22 ± 4.26
Large Scale ($M = 10, N = 20, K = 200$)					
CAR-UCB	90.36 ± 2.22	3.71 ± 0.36	58.89 ± 1.68	115.72 ± 5.74	126.07 ± 2.67
PLR	72.21 ± 3.80	2.72 ± 0.51	77.82 ± 2.97	148.97 ± 3.13	153.59 ± 2.10
FT-Flood	0.00	382.40 ± 9.52	602.81 ± 0.05	608.44 ± 0.11	609.56 ± 0.10
Oracle	100.00	1.00	37.34 ± 0.57	44.62 ± 0.94	46.37 ± 1.10



(a) #Network interface = 2, #Server = 3 **(b) #Network interface = 5, #Server = 10** **(c) #Network interface = 10, #Server = 20**

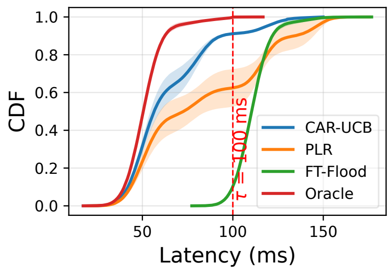
Figure 3: Cumulative regret over time for CAR-UCB, PLR, and Oracle across three system scales ($K = 6, 50, 200$). Shaded regions show variability across 20 independent trials.

90%–91% for all network sizes (e.g., 91.15% at small scale and 90.36% at large scale), while converging to a stable average per-request cost of ≈ 3.7. These results show that CAR-UCB learns to replicate requests over a small number of low-cost paths, improving the likelihood that at least one copy meets the deadline, without overloading shared interfaces.

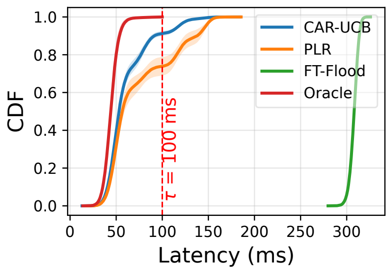
Figure 3 further confirms this trend through the cumulative cost-weighted regret analysis. The Oracle incurs zero regret by definition, while PLR accumulates regret rapidly due to frequent selections that are either unreliable or unnecessarily expensive. In contrast, CAR-UCB exhibits a markedly flatter regret curve, demonstrating its ability to quickly identify and exploit low-cost, high-reliability subsets. This behavior is driven by its structured candidate generation strategy (Algorithm 1), which avoids exhaustive exploration of the combinatorial action space while remaining responsive to non-stationary conditions.

7.1.2 Latency and Stability Analysis. Examining the latency metrics, CAR-UCB delivers substantially lower average latency than PLR, reducing latency by approximately 13–20 ms across all scenarios. At the same time, CAR-UCB remains close to the theoretical optimum, i.e., its gap to the Oracle baseline is approximately 14 ms in the small-scale setting and 19 ms in the large-scale setting.

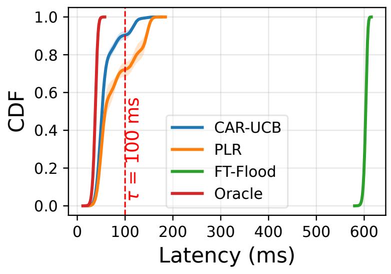
The cumulative distribution functions (CDFs) in Figure 4 further illustrate the robustness of CAR-UCB. Across 20 independent trials, the latency CDFs show very limited variability, demonstrating stable behavior under non-stationary latency dynamics. Such robustness stems from the algorithm's core online learning mechanism (Algorithm 2): by continuously updating confidence bounds based on observed outcomes, the algorithm quickly penalizes degraded paths and shifts selection to better subsets when bursts or phase changes occur.



(a) #Network interface = 2, #Server = 3



(b) #Network interface = 5, #Server = 10



(c) #Network interface = 10, #Server = 20

Figure 4: Latency CDFs for the four approaches across three system scales ($K = 6, 50, 200$). Shaded regions show variability across 20 independent trials. The latency deadline τ is shown as a vertical dashed line.

In contrast, PLR produces significantly wider confidence bands in the latency CDFs, reflecting its sensitivity to non-stationary latency dynamics. This instability is a direct consequence of the mismatch between PLR's model-based reliability estimates and the environment's rapidly changing latency patterns. When bursts or phase shifts violate its distributional assumptions, PLR's estimates become inaccurate, causing it to persist with suboptimal paths and resulting in highly inconsistent performance across trials.

7.1.3 Qualitative Analysis: Alignment of Decisions with Network Dynamics. To understand how CAR-UCB, PLR, and the Oracle react to evolving network conditions, we visualize each algorithm's decisions alongside the ground-truth latency dynamics to examine how well the decisions align with the underlying network behavior.

Figure 5 presents a representative small-scale scenario ($K = 6$) over a 2000-step window, showing how the algorithms track and react to underlying path fluctuations. The ground-truth latencies are shown in Figure 5a, while Figures 5b, 5c, and 5d visualize, respectively for CAR-UCB, PLR, and the Oracle, the selected path at each time step and the number of paths used. The table at the bottom of the figure lists the cost of each path for this specific trial.

CAR-UCB consistently aligns its decisions with the underlying dynamics: when bursts or phase shifts degrade a path, the algorithm effectively down-weights it and switches to better alternatives. Figure 5b also reveals that CAR-UCB frequently selects a secondary path concurrently with the primary good path. This behavior serves a dual purpose: it provides immediate redundancy to handle potential jitter (exploitation) while gathering data on alternative paths to maintain accurate reliability estimates (exploration).

In contrast, PLR exhibits a slower response to dynamic network changes (Figure 5c). Finally, the Oracle, leveraging perfect knowledge of path conditions, prioritizes cost minimization above all else. It selects the lowest-cost path that satisfies the deadline, using latency only as a tie-breaker when multiple paths incur the same cost (Figure 5d).

7.1.4 Multi-Agent Stability and Herd Avoidance. To evaluate the theoretical claim of decentralized stability (presented in Section 5.2), we simulated a small-scale scenario with three robots, each independently executing CAR-UCB to select paths from a shared pool of $K = 6$ paths. The environment was configured such that, in every phase, exactly two paths provided low latency (i.e., below the deadline $\tau = 100$ ms). Each phase lasted 500 time steps, after which the environment shifted and burst patterns were introduced, following the same dynamics as in the previous experiment (see Table 3).

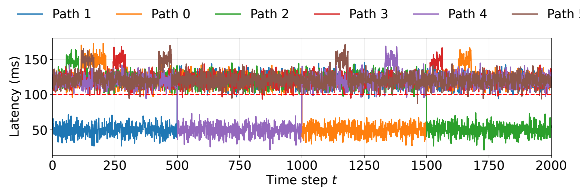
Figure 6 shows the behavior of three agents running the CAR-UCB algorithm over the first 1000 time steps. For $t < 500$, the agents settle on different good paths while still performing mild exploration: Agent 1 predominantly selects Path 1, Agent 3 prefers Path 4, and Agent 2 alternates between these two paths rather than committing to either one.

At $t = 500$, both Path 1 and Path 4 experience a latency spike. All three agents eventually abandon these degraded paths, but not at the same timestep. The multi-colored fluctuations in the path-selection panel of Figure 6a immediately after the spike indicate that each agent explores different fallback paths rather than converging simultaneously on a single alternative.

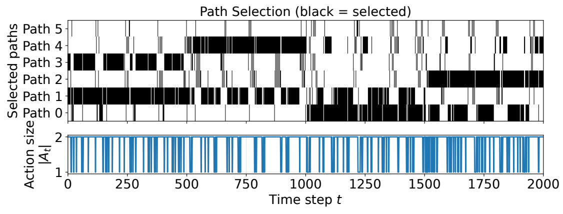
The zoomed view in Figure 6b focuses specifically on Path 1. The distinct temporal usage patterns show that the three agents stop using, and/or revisit Path 1 at different times. This confirms that the adaptation is desynchronized: each agent updates its belief state independently and reaches a different confidence threshold for abandoning or reconsidering Path 1.

The asynchronous migration is crucial. Because the agents do not switch simultaneously, they avoid creating a synchronized load spike on any single alternative path. For $t > 500$, the agents redistribute across the remaining paths in a differentiated manner. Since they do not move to the same fallback path at the same time, the transition load is spread out, reducing the chance of a secondary congestion spike and supporting stable recovery.

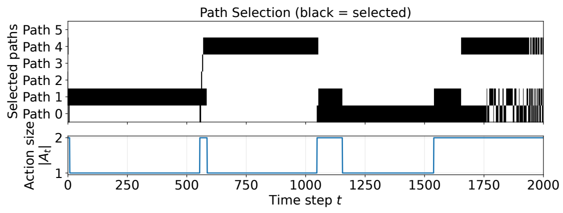
7.1.5 Ablation Study and Parameter Sensitivity. Following the comparative evaluation in the previous section, where CAR-UCB is assessed under a fixed reference configuration (Table 2), we examine its sensitivity to key hyperparameters. In



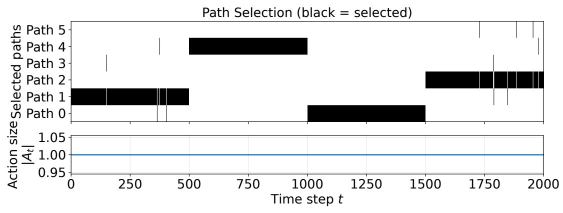
(a) Ground-truth network dynamics



(b) CAR-UCB's decision



(c) PLR's decision



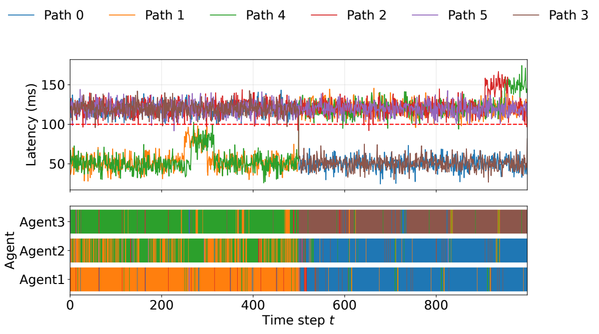
(d) Oracle's decision

Path ID	0	1	2	3	4	5
Cost	1.0	2.0	3.0	1.0	1.0	2.0

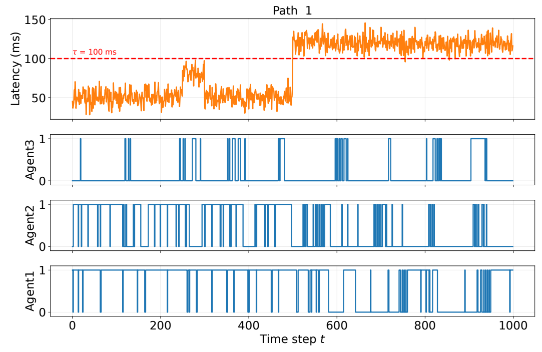
Figure 5: Qualitative alignment between each algorithm's decisions and the underlying network dynamics. The table shows the per-path costs for this trial.

particular, we study the exploration tolerance γ and the history window size W, which jointly govern the exploration–exploitation trade-off and the algorithm's responsiveness to non-stationarity. The maximum subset size is fixed to 3 paths and is not varied as a separate parameter. In practice, varying γ already induces a broad range of effective replication behaviors, from predominantly single-path selections at low γ to more aggressive multi-path exploration at higher γ. We also fix the local search parameter to $k = 2$, as increasing the search radius was observed to raise computational overhead without improving reliability.

7.1.6 *Real-Time Feasibility and Computational Overhead.*



(a) Path selected by each agent over time



(b) Zoomed view of Path 1 usage. For each agent, a value of 1 indicates that the agent selected Path 1 at that time step, while 0 indicates no usage

Figure 6: Herd-avoidance behavior exhibited by the CAR-UCB algorithm with three independent agents.

7.2 Field Deployment Results

8 Conclusion and Future Work

A Proof of No-Regret Optimality (Lemma 5.1)

Proof. Let

$$R_T = \sum_{t=1}^{T} r_t = \sum_{t=1}^{T} C(A_t)\big(f^* - f(A_t)\big)$$

denote the cost-varying cumulative regret over horizon T, where $f^* = \max_{A\in\mathcal{A}} f(A)$ is the optimal success probability and $C(A_t)$ is the cost of the selected action. Let $C_{\max} = \max_{A\in\mathcal{A}} C(A)$ denote the maximum possible cost of any action.

We rely on the standard GP realizability assumption: the true function g lies in the Reproducing Kernel Hilbert Space (RKHS) of the kernel used by the SGP model. Under this assumption, the concentration inequality for the approximate posterior [8] states that for any $\delta \in (0, 1)$, with probability at least $1 - \delta$, the following confidence bound holds:

$$\big|g(A) - \mu_{t-1}(A)\big| \le \beta_t\, \sigma_{t-1}(A), \qquad \forall A \in \mathcal{A},\ t \ge 1 \qquad (27)$$

Let $\mathcal{E}$ be the "Good Event" where inequality (27) holds for all t. On $\mathcal{E}$, we define the upper confidence bound:

$$U_{t-1}(A) = \Phi(\mu_{t-1}(A) + \beta_t \sigma_{t-1}(A))$$

Since the probit link $\Phi(\cdot)$ is monotonic, on $\mathcal{E}$ we have $f(A) \leq U_{t-1}(A)$ for all A.

The CAR-UCB algorithm selects A_t such that $A_t \in S_2 \subseteq S_1$. By the definition of the feasibility set S_1, we have:

$$U_{t-1}(A_t) + \varepsilon_t \geq \max_A U_{t-1}(A) \geq U_{t-1}(A^*) \geq f^*$$

Rearranging this, the reliability gap is bounded by:

$$f^* - f(A_t) \leq U_{t-1}(A_t) - f(A_t) + \varepsilon_t$$

Using the 1-Lipschitz property of Φ and the confidence bound (27), on event $\mathcal{E}$:

$$U_{t-1}(A_t) - f(A_t) \leq \beta_t \sigma_{t-1}(A_t)$$

Thus, the instantaneous regret on the Good Event is bounded by:

$$r_t = C(A_t)(f^* - f(A_t)) \leq C_{\max}\big(\beta_t \sigma_{t-1}(A_t) + \varepsilon_t\big)$$

Summing over T gives the bound on the cumulative regret conditioned on $\mathcal{E}$:

$$R_T \leq C_{\max}\beta_T \sum_{t=1}^{T} \sigma_{t-1}(A_t) + C_{\max} \sum_{t=1}^{T} \varepsilon_t$$

Using the standard information gain bound $\sum \sigma_{t-1}(A_t) \leq \sqrt{T\gamma_T}$:

$$R_T|_{\mathcal{E}} \leq C_{\max}\beta_T \sqrt{T\gamma_T} + C_{\max} \sum_{t=1}^{T} \varepsilon_t$$

To prove the bound in expectation, we account for the failure probability δ. In the worst case (event $\neg\mathcal{E}$), the regret per step is at most $C_{\max}$.

$$\mathbb{E}[R_T] \leq (1-\delta)\,(R_T|_{\mathcal{E}}) + \delta(T \cdot C_{\max})$$

Setting $\delta = 1/T$. Then:

$$\mathbb{E}[R_T] \leq C_{\max}\beta_T \sqrt{T\gamma_T} + C_{\max} \sum_{t=1}^{T} \varepsilon_t + C_{\max}$$

Dividing by T:

$$\frac{\mathbb{E}[R_T]}{T} \leq C_{\max}\beta_T \sqrt{\frac{\gamma_T}{T}} + \frac{C_{\max}}{T} \sum_{t=1}^{T} \varepsilon_t + \frac{C_{\max}}{T} \tag{28}$$

Using our schedules

$$\beta_t = O(\sqrt{\log t}), \qquad \varepsilon_t = O\left(\frac{1}{\sqrt{\log t}}\right), \qquad \gamma_T = o(T),$$

each term on the right-hand side of equation (28) converges to 0 as $T \to \infty$. Thus,

$$\lim_{T\to\infty} \frac{\mathbb{E}[R_T]}{T} = 0$$

which proves that CAR-UCB achieves no-regret. □

B Regret Analysis Under Approximations

In this section, we relax the exact inference assumptions to account for the Sparse GP approximation errors introduced in Section 4.

Theorem B.1 (Bounded Regret Floor). *Let Δ_{SGP} be the bound on the model approximation error and Δ_{opt} be the bound on the candidate search error. The average expected regret of CAR-UCB is bounded by:*

$$\lim_{T\to\infty} \frac{\mathbb{E}[R_T]}{T} \leq C_{\max}(2\Delta_{SGP} + \Delta_{opt})$$

This implies that while the regret does not vanish, it converges to a constant floor determined by the quality of the approximation.

Proof. Let $\hat{U}_{t-1}(A)$ be the acquisition function computed using the sparse model. We define the error terms as follows:

(1) **Model Error ($\Delta_{\mathbf{SGP}}$):** $|\hat{U}_{t-1}(A) - U_{t-1}(A)| \leq \Delta_{\text{SGP}}$.
(2) **Optimization Error ($\Delta_{\mathbf{opt}}$):** $\max_A \hat{U}(A) - \max_{A\in\mathcal{A}_{\text{cand}}} \hat{U}(A) \leq \Delta_{\text{opt}}$.

We decompose the instantaneous regret $r_t = f(A^*) - f(A_t)$, where A^* is the global optimum and A_t is the action selected by the agent from the candidate set:

$$\begin{aligned} r_t &= f(A^*) - f(A_t) \\ &\leq U_{t-1}(A^*) - f(A_t) \quad \text{(Optimism of exact UCB)} \\ &\leq \hat{U}_{t-1}(A^*) + \Delta_{\text{SGP}} - f(A_t) \quad \text{(Model Error)} \\ &\leq (\hat{U}_{t-1}(A_t) + \Delta_{\text{opt}}) + \Delta_{\text{SGP}} - f(A_t) \quad \text{(Optimization Error)} \\ &\leq (U_{t-1}(A_t) + \Delta_{\text{SGP}}) + \Delta_{\text{opt}} + \Delta_{\text{SGP}} - f(A_t) \quad \text{(Model Error on } A_t\text{)} \\ &= (U_{t-1}(A_t) - f(A_t)) + 2\Delta_{\text{SGP}} + \Delta_{\text{opt}} \end{aligned}$$

Substituting the standard confidence bound $U_{t-1}(A_t) - f(A_t) \leq 2\beta_t\sigma_{t-1}(A_t)$, we have:

$$r_t \leq 2\beta_t\sigma_{t-1}(A_t) + (2\Delta_{\text{SGP}} + \Delta_{\text{opt}})$$

Summing over T, dividing by T, and taking the limit $T \to \infty$:

$$\lim_{T\to\infty} \frac{R_T}{T} \leq \underbrace{\lim_{T\to\infty} \frac{\sum 2\beta_t\sigma_{t-1}}{T}}_{\to 0} + (2\Delta_{\text{SGP}} + \Delta_{\text{opt}})$$

The first term vanishes because $\sum \beta_t\sigma_{t-1}(A_t) \leq O^*(\sqrt{T\gamma_T})$, which is sublinear in T for standard kernels (Srinivas et. al. [37], Theorem 1). This leaves the approximation error floor as the asymptotic bound. □

References

[1] Peter Auer. 2002. Using confidence bounds for exploitation-exploration trade-offs. *Journal of machine learning research* 3, Nov (2002), 397–422.
[2] Mehdi Bennis, Mérouane Debbah, and H Vincent Poor. 2018. Ultrareliable and low-latency wireless communication: Tail, risk, and scale. *Proc. IEEE* 106, 10 (2018), 1834–1853.
[3] Giuseppe Bianchi. 2002. Performance analysis of the IEEE 802.11 distributed coordination function. *IEEE Journal on selected areas in communications* 18, 3 (2002), 535–547.
[4] Guo Chen and Hiroki Tsurumi. 2010. Probit and logit model selection. *Communications in Statistics—Theory and Methods* 40, 1 (2010), 159–175.

[5] Kaiyuan Chen, Kush Hari, Trinity Chung, Michael Wang, Nan Tian, Christian Juette, Jeffrey Ichnowski, Liu Ren, John Kubiatowicz, Ion Stoica, et al. 2024. FogROS2-FT: Fault Tolerant Cloud Robotics. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 1390–1397.

[6] Kaiyuan Chen, Nan Tian, Christian Juette, Tianshuang Qiu, Liu Ren, John Kubiatowicz, and Ken Goldberg. 2025. Fogros2-plr: Probabilistic latency-reliability for cloud robotics. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 16290–16297.

[7] Sandeep Chinchali, Apoorva Sharma, James Harrison, Amine Elhafsi, Daniel Kang, Evgenya Pergament, Eyal Cidon, Sachin Katti, and Marco Pavone. 2021. Network offloading policies for cloud robotics: a learning-based approach. *Autonomous Robots* 45, 7 (2021), 997–1012.

[8] Sayak Ray Chowdhury and Aditya Gopalan. 2017. On kernelized multi-armed bandits. In *International Conference on Machine Learning*. PMLR, 844–853.

[9] Jeffrey Dean and Luiz André Barroso. 2013. The tail at scale. *Commun. ACM* 56, 2 (2013), 74–80.

[10] Zhihui Du, Ligang He, Yinong Chen, Yu Xiao, Peng Gao, and Tongzhou Wang. 2017. Robot cloud: Bridging the power of robotics and cloud computing. *Future Generation Computer Systems* 74 (2017), 337–348.

[11] Augusto Fasano and Daniele Durante. 2022. A class of conjugate priors for multinomial probit models which includes the multivariate normal one. *Journal of Machine Learning Research* 23, 30 (2022), 1–26.

[12] Vincent Fortuin, Gideon Dresdner, Heiko Strathmann, and Gunnar Rätsch. 2021. Sparse Gaussian processes on discrete domains. *IEEE Access* 9 (2021), 76750–76758.

[13] Jensen Gao, Bidipta Sarkar, Fei Xia, Ted Xiao, Jiajun Wu, Brian Ichter, Anirudha Majumdar, and Dorsa Sadigh. 2024. Physically grounded vision-language models for robotic manipulation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 12462–12469.

[14] Shogo Hayashi, Junya Honda, and Hisashi Kashima. 2022. Bayesian optimization with partially specified queries. *Machine Learning* 111, 3 (2022), 1019–1048.

[15] René Henrion and Werner Römisch. 2010. Lipschitz and differentiability properties of quasi-concave and singular normal distribution functions. *Annals of Operations Research* 177, 1 (2010), 115–125.

[16] Vu Viet Hoang, Quoc Anh Hoang Nguyen, et al. 2025. Bayesian Optimization for Unknown Cost-Varying Variable Subsets with No-Regret Costs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 39. 27000–27008.

[17] I-Hong Hou. 2024. Distributed No-Regret Learning for Multi-Stage Systems with End-to-End Bandit Feedback. In *Proceedings of the Twenty-fifth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*. 41–50.

[18] Guoqiang Hu, Wee Peng Tay, and Yonggang Wen. 2012. Cloud robotics: architecture, challenges and applications. *IEEE network* 26, 3 (2012), 21–28.

[19] Ben Kehoe, Sachin Patil, Pieter Abbeel, and Ken Goldberg. 2015. A survey of research on cloud robotics and automation. *IEEE Transactions on automation science and engineering* 12, 2 (2015), 398–409.

[20] Yeseung Kim, Dohyun Kim, Jieun Choi, Jisang Park, Nayoung Oh, and Daehyung Park. 2024. A survey on integration of large language models with intelligent robots. *Intelligent Service Robotics* 17, 5 (2024), 1091–1107.

[21] Rafael Laufer and Leonard Kleinrock. 2015. The capacity of wireless CSMA/CA networks. *IEEE/ACM Transactions on Networking* 24, 3 (2015), 1518–1532.

[22] Neil Lawrence, Matthias Seeger, and Ralf Herbrich. 2002. Fast sparse Gaussian process methods: The informative vector machine. *Advances in neural information processing systems* 15 (2002).

[23] Wes Lloyd, Shrideep Pallickara, Olaf David, Mazdak Arabi, and Ken Rojas. 2017. Mitigating resource contention and heterogeneity in public clouds for scientific modeling services. In *2017 IEEE International Conference on Cloud Engineering (IC2E)*. IEEE, 159–166.

[24] Arpan Losalka and Jonathan Scarlett. 2023. Benefits of monotonicity in safe exploration with Gaussian processes. In *Uncertainty in Artificial Intelligence*. PMLR, 1304–1314.

[25] Arman Maghsoudnia, Eduard Vlad, Aoyu Gong, Dan Mihai Dumitriu, and Haitham Hassanieh. 2024. Ultra-reliable low-latency in 5G: A close reality or a distant goal?. In *Proceedings of the 23rd ACM Workshop on Hot Topics in Networks*. 111–120.

[26] Aoran Mei, Guo-Niu Zhu, Huaxiang Zhang, and Zhongxue Gan. 2024. Replanvlm: Replanning robotic tasks with visual language models. *IEEE Robotics and Automation Letters* (2024).

[27] Hannes Nickisch, Carl Edward Rasmussen, et al. 2008. Approximations for binary Gaussian process classification. *Journal of Machine Learning Research* 9, 10 (2008), 2035–2078.

[28] Seyed Hossein Nikounia and Siamak Mohammadi. 2015. Hypervisor and neighbors' noise: Performance degradation in virtualized environments. *IEEE Transactions on Services Computing* 11, 5 (2015), 757–767.

[29] Hebatalla Ouda, Khalid Elgazzar, and Hossam S Hassanein. 2025. Analyzing the Impact of Network Variability on the Performance of Telesurgery. In *2025 International Wireless Communications and Mobile Computing (IWCMC)*. IEEE, 1744–1750.

[30] Yueyang Pan, Ruihan Li, and Chenren Xu. 2022. The first 5G-LTE comparative study in extreme mobility. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 6, 1 (2022), 1–22.

[31] Michele Polese, Marco Giordani, and Michele Zorzi. 2018. 3GPP NR: the standard for 5G cellular networks. *5G Italy White eBook: from Research to Market* (2018).

[32] Mina Rady, Oana Iova, Hervé Rivano, Angeliki Deligianni, and Leonidas Drikos. 2024. How does Wi-Fi 6 fare? An industrial outdoor robotic scenario. *Ad Hoc Networks* 156 (2024), 103418.

[33] Jochen Ranger, Jörg-Tobias Kuhn, and José-Luis Gaviria. 2015. A race model for responses and response times in tests. *Psychometrika* 80, 3 (2015), 791–810.

[34] LKPJ Rdusseeun and P Kaufman. 1987. Clustering by means of medoids. In *Proceedings of the statistical data analysis based on the L1 norm conference, neuchatel, switzerland*, Vol. 31. 28.

[35] Sheldon M Ross. 1995. *Stochastic processes*. John Wiley & Sons.

[36] Aleksandrs Slivkins et al. 2019. Introduction to multi-armed bandits. *Foundations and Trends® in Machine Learning* 12, 1-2 (2019), 1–286.

[37] Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. 2010. Gaussian process optimization in the bandit setting: no regret and experimental design. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*. 1015–1022.

[38] Yanan Sui, Alkis Gotovos, Joel Burdick, and Andreas Krause. 2015. Safe exploration for optimization with Gaussian processes. In *International conference on machine learning*. PMLR, 997–1005.

[39] Sebastian Tay, Chuan Sheng Foo, Daisuke Urano, Richalynn Leong, and Bryan Kian Hsiang Low. 2023. Bayesian optimization with cost-varying variable subsets. *Advances in Neural Information Processing Systems* 36 (2023), 3008–3031.

[40] Michalis Titsias. 2009. Variational learning of inducing variables in sparse Gaussian processes. In *Artificial intelligence and statistics*. PMLR, 567–574.

[41] Joel Vallone, Robert Birke, and Lydia Y Chen. 2017. Making neighbors quiet: An approach to detect virtual resource contention. *IEEE Transactions on services computing* 13, 5 (2017), 843–856.

[42] Yusuke Wakasa, Kenichi Hakamada, Hajime Morohashi, Takahiro Kanno, Kotaro Tadano, Kenji Kawashima, Yuma Ebihara, Eiji Oki, Satoshi Hirano, and Masaki Mori. 2024. Ensuring communication redundancy and establishing a telementoring system for robotic telesurgery using multiple communication lines. *Journal of Robotic Surgery* 18, 1 (2024), 9.

[43] Jiaqi Wang, Enze Shi, Huawen Hu, Chong Ma, Yiheng Liu, Xuhui Wang, Yincheng Yao, Xuan Liu, Bao Ge, and Shu Zhang. 2025. Large language models for robotics: Opportunities, challenges, and perspectives. *Journal of Automation and Intelligence* 4, 1 (2025), 52–64.