LAMPS: A Learning-based Mobility Planning via Posterior State Inference using Gaussian Cox Process Models

Egemen Erbayat[1*], Yongsheng Mei[1*], Gina Adam[1], Suresh Subramaniam[1], Sean Coffey[2], Nathaniel D. Bastian[2], Tian Lan[1]

[1]George Washington University, Washington, DC
[2]United States Military Academy, West Point, NY
{erbayat, ysmei, ginaadam, suresh, tlan} @gwu.edu, {sean.coffey, nathaniel.bastian}@westpoint.edu

Abstract

Learning-based mobility planning has proven effective in optimizing performance metrics like latency, throughput, and cost in applications such as path planning and network security. However, real-world networks often face partial or dynamic observability, limiting the applicability of existing robust optimization approaches, which can be conservative, inefficient, or require extensive retraining under changing conditions. This paper introduces LAMPS, a new learning-based mobility planning framework that leverages Gaussian Cox processes to estimate spatiotemporal network states and their uncertainty, enabling robust decision-making under varying observability without retraining. These posterior estimates are integrated into a utility-based planning algorithm that adapts policies trained under full observability to diverse conditions, optimizing average performance or ensuring robustness in near-worst-case scenarios. We analyze the LAMPS framework in a real-world situation involving UAV mobility and wireless resource management, demonstrating the framework's scalability, adaptability, and efficiency in dynamic network environments. Our evaluation results demonstrate the remarkable adaptability and robustness of the LAMPS. It consistently outperforms other methods under different observability conditions and setups, proving its effectiveness in dynamic environments without requiring retraining.

Introduction

Learning-based mobility planning and optimization has demonstrated great potential in many problem domains, such as path planning, network security, and traffic prediction. These domains benefit from efficient resource allocation, routing, and threat detection (Wang and Lin 2021; Luong et al. 2019; Nie et al. 2020). Such mobility planning problems are often formulated as a Markov Decision Process (MDP). The goal is to learn a decision-making policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$, which maps each network state to an optimal control action or a distribution over possible actions. This includes re-routing data, reallocating resources, or adjusting operational parameters to optimize performance metrics like latency, throughput, or cost efficiency.

In practice, obtaining complete network state information is often hard, especially when the network is decentralized, operates in dynamic settings, or needs to provide connectivity under unknown environments. In case of static partial observability (i.e., the level of partial observability does not change over time or during execution), existing work has considered robust optimization techniques (Moos et al. 2022) and partially-observable MDP (POMDP) models (Liu et al. 2022). Robust optimization, e.g., with respect to uncertainty set (Zhang, Feng, and Rong 2017) or worst-case analysis (Poursoltani and Delage 2022), can be too conservative and resource inefficient. For the POMDP approach, the idea is typically to extrapolate a belief state distribution from past observations, e.g., using LSTM (Meng, Gorbet, and Kulić 2021), RNN (Carr, Jansen, and Topcu 2020), or attention networks (Yang et al. 2023). However, these existing solutions for extrapolating belief states using neural networks are often not explainable. They can be ineffective in cases where the level of observability is dynamic and may vary during execution – thus leading to a distribution shift problem. As a result, the learned policies are either brittle or require expensive retraining – using data collected under new observability conditions or through additional interactions, e.g., under diminished observability due to congestion or adversaries.

This paper introduces LAMPS, a learning-based mobility planning framework that leverages the Gaussian Cox process – a doubly stochastic process model renowned for its efficacy in spatiotemporal data analysis (Møller, Syversveen, and Waagepetersen 1998) – to obtain efficient posterior estimates of network states and thus support utility-based decision making by augmenting a trained policy to work under dynamic observability changes. The framework extrapolates spatiotemporal network state information from available observations, providing efficient posterior estimates of the global network state, including its mean and variance. These estimates reflect the event density across the network and the uncertainty of the predictions. Building on these results, we propose a robust mobility planning algorithm capable of reusing decision-making policies trained under full observability while adapting to varying observability conditions without retraining. More precisely, we sample the underlying network state distribution from the posterior estimates, obtain the action value corresponding to different network states, and combine the results using a utility function for mobility planning and decision-making. The

*These authors contributed equally to this work.

LAMPS framework can incorporate many different utilities, from maximizing the average performance based on network state inference to robust optimization with respect to 95th-percentile performance to ensure robustness in near-worst-case conditions.

As a foundational step, we adopt the Gaussian Cox process model with smooth link functions to perform maximum *a posteriori* inference of the latent intensity function's posterior and the covariance of point events distributed across a 2D map. LAMPS builds on the intensity estimation in (Mei, Imani, and Lan 2024), which utilizes Laplace approximation and kernel transformations to reformulate the estimation problem within a reproducing kernel Hilbert space. In mobility planning, this method provides the probabilistic estimates necessary for integration into a reinforcement learning algorithm, where it explicitly informs the state representation (and associated uncertainty) of the network states.

Next, we design a robust mobility planning algorithm based on the posterior estimates of network state distribution. It quantifies the uncertainty due to partial observability and enables us to develop robust mobility planning under different utility functions, e.g., maximizing worst-case, 95-percentile, and average performance objectives. Specifically, we consider a mobility planning problem involving joint wireless resource management and unmanned aerial vehicle (UAV) mobility planning. The policy – using a multi-agent reinforcement learning (MARL) algorithm involving multiple UAVs – is initially trained under full observability, leveraging complete state information to develop robust decision-making policies. Using Gaussian Cox process-based intensity estimates, this framework extends its adaptability to environments with partial or dynamic observability without retraining. By integrating these estimates into the decision-making process, UAVs will adjust their actions dynamically to optimize data collection and transmission while maintaining resource efficiency. This approach ensures scalability and robustness, enabling UAVs to operate effectively across diverse and unpredictable network conditions.

To validate the effectiveness of the proposed method, we test LAMPS on our defined environment—spanning a range of setups with varying map sizes, numbers of UAVs, and observability conditions. These setups are intended to simulate a variety of real-world scenarios, ranging from environments with limited visibility to expansive settings that provide extensive initial information. Our evaluation results demonstrate that LAMPS can effectively adapt to changes in observability conditions and consistently outperforms baseline methods. It notably achieves better performance in situations with dynamic partial observability, where other algorithms often struggle due to the need for retraining or manual adjustments. This adaptability is crucial in dynamic partially observable environments, demonstrating the method's robustness and practical application to changing challenges. LAMPS consistently delivers high performance in various scenarios, proving to be a reliable and efficient solution for navigation and decision-making in UAV operations.

Background

Partially Observable Markov Decision Process

We consider a multi-agent sequential decision-making problem under a fully cooperative setting, formalized as a decentralized partially observable Markov decision process, Dec-POMDP, (Oliehoek and Amato 2016). This framework is represented by the tuple $G = \langle \mathcal{S}, \mathcal{U}, \mathcal{P}, \mathcal{R}, \mathcal{Z}, \mathcal{O}, n, \gamma \rangle$. Here, $\mathcal{S}$ denotes the global state of the environment, and n is the number of agents involved in the task. At each timestep, each agent $a \in \mathcal{A} = \{1, \ldots, n\}$ selects an action $u^a \in \mathcal{U}$, with these actions collectively forming the joint action $\mathbf{u} \in \mathcal{U}^{\mathbf{n}}$. Then, the environment transitions to a new state according to the transition function $P(s'|s, \mathbf{u}) : \mathcal{S} \times \mathbf{U} \times \mathcal{S} \to [0, 1]$. All agents share the same reward function $r(s, \mathbf{u}) : \mathcal{S} \times \mathbf{U} \to \mathcal{R}$, with a discount factor $\gamma \in [0, 1]$ applied to future rewards.

In this partially observable environment, each agent receives an individual observation $z \in \mathcal{Z}$, generated by the observation function $O(s, \mathbf{u}) : \mathcal{S} \times \mathbf{U} \to \mathcal{Z}$. The history of an agent's actions and observations is represented as $\tau_a \in T_a \equiv (\mathcal{Z} \times \mathbf{U})^*$. Based on this history, each agent's policy is given by $\pi^a(u_a|\tau_a) : \mathcal{T} \times \mathcal{U} \to [0, 1]$. The joint policy π defines the joint action-value function:

$$Q^\pi(s_t, u_t) = \mathbb{E}_{s_{t+1:\infty}, u_{t+1:\infty}} \left[\mathcal{R}_t \mid s_t, u_t \right],$$

where t represents the current timestep, and $\mathcal{R}_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$ is the discounted cumulative reward. In this study, we adopt a centralized training paradigm with decentralized execution: the learning algorithm has access to the global state s and the complete set of action-observation histories T_a during training, while each agent, during execution, can only access its individual action-observation history.

2D Intensity Estimation with Gaussian Cox Process

Given a 2D observation space, we consider a Lebesgue measurable compact observation space $\mathcal{S} \subset \mathbb{R}^2$ and a latent random function $g(\cdot) : \mathcal{S} \to \mathbb{R}$ following Gaussian Process (GP), denoted by $\mathcal{GP}(g(\bar{\boldsymbol{t}})|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with mean and covariance $\theta_{\boldsymbol{\mu},\boldsymbol{\Sigma}} \triangleq (\boldsymbol{\mu}, \boldsymbol{\Sigma})$, respectively. Observations are generated from a point process modulated by the latent random function $g(\bar{\boldsymbol{t}})$ through a deterministic, non-negative link function that connects $g(\bar{\boldsymbol{t}})$ with latent intensity $\lambda(\bar{\boldsymbol{t}}) = \kappa(g(\bar{\boldsymbol{t}}))$. Given a set of $\hat{n}$ observed point events $\{\bar{\boldsymbol{t}}_i\}_{i=1}^{\hat{n}}$ in the observation space $\mathcal{S}$, we consider the likelihood function (regarding GP parameters $\theta_{\boldsymbol{\mu},\boldsymbol{\Sigma}}$) that is formulated as an expectation over the space of latent random functions, i.e.,

$$p(\{\bar{\boldsymbol{t}}_i\}_{i=1}^{\hat{n}}|\theta_{\boldsymbol{\mu},\boldsymbol{\Sigma}}) = \int_{g(\bar{\boldsymbol{t}})} p(\{\bar{\boldsymbol{t}}_i\}_{i=1}^{\hat{n}}|g(\bar{\boldsymbol{t}}))\, p(g(\bar{\boldsymbol{t}})|\theta_{\boldsymbol{\mu},\boldsymbol{\Sigma}})\, \mathrm{d}g(\bar{\boldsymbol{t}}), \tag{1}$$

where $p(g(\bar{\boldsymbol{t}})|\theta_{\boldsymbol{\mu},\boldsymbol{\Sigma}})$ is the Gaussian prior. Within equation (1), the log-probability conditioned on a specific random function $g(\bar{\boldsymbol{t}})$ takes the following form:

$$\log p(\{\bar{\boldsymbol{t}}_i\}_{i=1}^{\hat{n}}|g(\bar{\boldsymbol{t}})) = \sum_{i=1}^{\hat{n}} \log \lambda(\bar{\boldsymbol{t}}_i) - \int_{\mathcal{S}} \lambda(\bar{\boldsymbol{t}})\, \mathrm{d}\bar{\boldsymbol{t}}, \tag{2}$$

where latent intensity function $\lambda(\bar{\boldsymbol{t}}) = \kappa(g(\bar{\boldsymbol{t}}))$ is obtained using the deterministic non-negative link function $\kappa(\cdot) : \mathbb{R} \rightarrow \mathbb{R}^+$. For simplicity of notations, we will use a short form $\boldsymbol{g}$ to represent $g(\bar{\boldsymbol{t}})$ in the rest of the paper.

Problem Definition

Dynamic Decision-Making/Planning

We consider a mobility planning problem involving joint wireless resource management and UAV mobility planning in environments with dynamic partial observability. Wireless resource management focuses on allocating bandwidth, constrained by the total available bandwidth, with the objective of transmitting as many events as possible to the central office. UAV mobility planning involves controlling movements to maximize event discovery while balancing the trade-off of staying close to the central office to improve data rates and transmission efficiency. Constraints include map boundaries and collision avoidance as illustrated in Figure 1. The goal is to maximize the efficiency of collecting and transmitting data to a central office (CO) while adapting to changes in observability, event locations, and network demands.

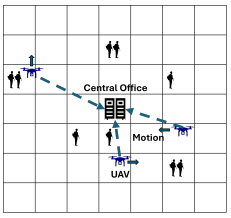

Figure 1: System model.

A major challenge of this joint network and mobility planning problem is the dynamic change of observability, which can be caused by adversarial conditions or environmental factors (such as network congestion or interference during execution). It means that, unlike static observability, the amount of network state information that is observable and available to the planning model can vary over time. This dynamic partial observability introduces unpredictability, forcing UAVs to operate with incomplete and/or shifting data, leading to distribution shifts that make existing solutions unreliable and often require resource-intensive retraining. Our key idea is to leverage Gaussian Cox process models to obtain posterior network state distribution from available partial observations. Then, during planning, we sample the posterior network state distribution, estimate the action-value distribution, and employ a utility function for robust decision-making. By leveraging such state estimation and utility-based planning, we can avoid retraining existing policies and ensure robust performance in uncertain and evolving environments. An illustration of LAMPS is provided in Figure 2.

This problem formulation allows us to capture a wide range of design objectives. Each UAV's decision-making policy involves multiple concurrent objectives, such as minimizing latency, avoiding congestion, and conserving bandwidth. Rewards are structured to reflect successful data transmissions, timeliness, and network reliability, while penalties discourage undesirable behaviors such as UAV collisions. This multi-objective approach encourages UAVs to balance exploring new events with efficient data relays to the central office.

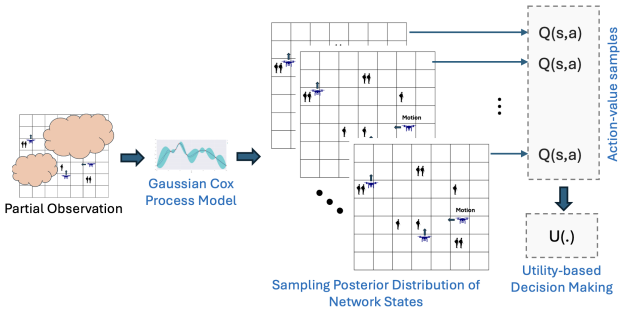

Figure 2: An illustration of LAMPS, mobility planning with Gaussian Cox Process for network state inference and utility-based decision-making.

Our System & Channel Model

The scenario includes multiple UAVs serving as surveillance devices and a CO needing necessary data processing and computing. UAVs will capture related data from a 2D map. We assume different events can occur in the spatial domain, and each event may have a fixed packet size, σ (bits), or importance. The main objective for UAVs is to explore the events and send CO the required packets with low latency. Assuming that all the UAVs are located at the same flight altitude H, the impact of altitude can be disregarded.

The map is represented by a grid where each cell has a fixed distance, d (meters), between adjacent cells. Any UAV can move one cell at each time slot with a duration of t_M (seconds) in one of the four cardinal directions. Each UAV operates with its queue and sends packets immediately if the queue is not empty. If there is a queue, a UAV transmits packets following a first-come-first-serve (FCFS) order. Additionally, if an event occurs at the UAV's current position, the UAV captures the event data and places it in the queue if there is space, and the grid cell is left without the event afterward.

Consider the set of UAVs denoted by $\mathcal{A} = \{1, 2, ..., A\}$ and central office is denoted by Θ. We consider slotted time and index the time using $t \in \{1, 2, ..., T\}$. The location of UAV a at time slot t can be represented as $\mathbf{w_t^a} = [x_t^a, y_t^a]$. The location of CO can be represented as $\mathbf{w^\Theta} = [x^\Theta, y^\Theta]$.

The free-space path loss model between the UAV a and CO in the t-th time slot can be characterized by:

$$g_t^{a,\Theta} = \frac{\alpha_a}{d_{a,\Theta}[t]^2}$$

where α_a represents the channel power gain for the UAV a under the reference distance $d = 1$m and $d_{a,\Theta}[t] = \sqrt{H^2 + \|\mathbf{w_t^a} - \mathbf{w^\Theta}\|_2^2}$ is the Euclidean distance between the UAV a and central office Θ in tth time slot. Then, we

introduce a set of bandwidth assignment ratios $b_t^u \in [0,1]$ to represent the proportion of bandwidth that each UAV can get from the central office and consider the total bandwidth as B.

$$\sum_{u \in \mathcal{U}} b_t^u \leq 1. \tag{3}$$

Bandwidth is allocated among UAVs based on their weights ω_t. These weights directly impact the bandwidth allocation to each UAV, allowing them to manage traffic load dynamically. Increasing the weight of a UAV results in higher bandwidth allocation, enabling it to process tasks faster and reduce queue lengths, especially in congested areas. A UAV with a higher weight, ω_t^a receives a larger share of the total available bandwidth, while those with lower weights receive less. The allocation formula is:

$$b_t^a = \frac{\omega_t^a}{\sum_{j=1}^{A} \omega_t^j} \times B$$

where b_t^a is the bandwidth for the a-th UAV at time slot t, ω_t^a is its weight, and B is the total available bandwidth. This approach enables adaptive resource allocation, allowing higher-weight UAVs to address critical areas more effectively. Then, we denote the transmit power of the central office to be p_{max}, and the corresponding signal-to-noise ratio (SNR) at the receiver can be derived by:

$$\psi_t^{a,\Theta} = \frac{p_{max} g_t^{a,\Theta}}{B b_t^a N_0},$$

where N_0 denotes the power spectral density of Gaussian White Noise. Then, we can derive the corresponding achievable channel capacity:

$$C_t^{a,\Theta} = B b_t^a \log_2(1 + \psi_t^{a,\Theta}).$$

Finally, the number of transmitted bits in each time slot for UAV u in slot t can be calculated as:

$$\text{Transmitted bits per slot} = C_t^{a,\Theta} \cdot t_M.$$

Mobility Planning with State Predictions

We aim to develop a mobility planning framework capable of operating successfully in various maps, even when obtaining a complete network state is hard to reach. Our approach incorporates MDP formulations, state estimation techniques, utility-based decision-making, and reinforcement learning algorithms, enabling UAVs to perform dynamic and robust mobility planning. Specifically, we train a fully observable policy $\pi : S \to U$ using an MDP framework to optimize cumulative rewards. To handle partial observability, we use the Gaussian Cox process model to estimate event intensities, providing posterior mean and covariance data. These estimates are then used as inputs to the fully observable model, allowing it to adapt dynamically to partially observable environments. Following the estimation, we generate sampled maps based on the intensity data, which are then used in utility functions, such as $U_{\text{avg}}(u)$ for average return and $U_{95}(u)$ for near-worst-case performance, ensuring robust and efficient decision-making under dynamic conditions.

Markov Decision Process Formulation

We consider a mobility planning problem involving joint wireless resource management and UAV mobility planning. The training of a fully observability policy for mobility planning is considered as an MDP, represented by the tuple: $G = \langle \mathcal{S}, \mathcal{U}, \mathcal{P}, \mathcal{R}, \mathcal{O}, \gamma \rangle$ where:

- $\mathcal{S}$: The state space includes:
  - The locations of UAVs on a grid: $w_t^a = [x_t^a, y_t^a]$ for each UAV a.
  - Queue lengths indicating the number of packets each UAV is waiting to transmit.
  - Weights ω_t^a for UAV a at time t, affecting bandwidth allocation and task prioritization, bounded between 0 and 5.
  - Event distribution grid representing spatial event occurrences.
- $\mathcal{U}$: The action space such that each UAV can perform one of the following actions at any given time step:
  - Move in one of four cardinal directions.
  - Adjust weights by increments or decrements.
- $\mathcal{P}(s'|s,u)$: The state transition probability function, modeling system dynamics, including UAV position updates and changes in event grids.
- $\mathcal{R}(s,u)$: The reward function:
  - Positive rewards for successful data transmission.
  - Negative rewards for collisions, time delays, and out-of-bound movements.
- $\mathcal{O}$: The observation space, capturing the partial or full view of the state, including:
  - Current location of UAVs
  - Weights
  - Queue Lengths
  - The observation of the event distribution at any given time can be either fully or partially observable, depending on the scenario.
- $\gamma \in [0,1]$: The discount factor balancing immediate and future rewards.

Our goal is to learn a policy π for the joint wireless resource management and UAV mobility planning problem with full observability. Later, we will leverage the learned policy with network state estimates to enable robust mobility planning under changing levels of partial observability. We denote this learned policy by $\pi : \mathcal{S} \to \mathcal{U}$ that maximizes the expected cumulative discounted reward:

$$J(\pi) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, u_t) \mid \pi\right].$$

Such a policy with full observability can be obtained using any MARL algorithms (e.g., QMIX) and do not require extensive training data.

State Prediction using Posterior Mean and Covariance Estimation

Given the partially observable environment, UAVs estimate the probability of events occurring in each cell using intensity functions. This estimation leverages a trained model developed in fully observable conditions. The intensity of each cell is estimated using a specific kernel density estimation (Hastie and TIBSHIRANI 2001) approach, computing event occurrence probabilities from the estimated intensity function over all given events.

The intensity estimation of events on the 2D map can be obtained via approximations. As proposed by Mei, Imani, and Lan (2024), we can project the original problem into reproducing kernel Hilbert space (RKHS) and acquire the mean and covariance estimation of the intensity over the map based on given observations. The optimization objective is the minimization of the Poisson point process likelihood, which is:

$$\min_{g} \left\{ -\sum_{i=1}^{\hat{n}} \log \kappa(g(\bar{\boldsymbol{t}}_i)) + \int_{\mathcal{S}} \kappa(g(\bar{\boldsymbol{t}}))\, \mathrm{d}\bar{\boldsymbol{t}} \right\}. \tag{4}$$

Given a non-empty domain $\mathcal{S}$ and a symmetric positive definite kernel $k : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$, a unique RKHS $\mathcal{H}_k$ can be constructed. If we can formulate a regularized empirical risk minimization (ERM) problem as $\min_{h\in\mathcal{H}_k} R(\{h(\bar{\boldsymbol{t}}_i)\}_{i=1}^{\hat{n}}) + \gamma\Omega(\|h(\bar{\boldsymbol{t}})\|_{\mathcal{H}_k})$, where $R(\cdot)$ denotes the empirical risk of h, γ is the penalty factor, and $\Omega(\cdot)$ is a non-decreasing error in the RKHS norm, a unique optimal solution exists given by representer theorem (Schölkopf, Herbrich, and Smola 2001) as $h^*(\cdot) = \sum_{i=1}^{\hat{n}} \alpha_i k(\bar{\boldsymbol{t}}_i, \cdot)$, and the optimization can be cast regarding dual coefficients $\alpha \in \mathbb{R}^2$.

Since $\kappa(\cdot)$ is non-negative smooth link function, we define $h(\bar{\boldsymbol{t}}) \triangleq \kappa^{\frac{1}{2}}(g(\bar{\boldsymbol{t}}))$ so that $h^2(\bar{\boldsymbol{t}}) = \kappa(g(\bar{\boldsymbol{t}})) : \mathcal{S} \to \mathbb{R}^+$. This definition will provide us access to the property of L_2-norm that simplifies the problem-solving in the next. Then, we can formulate a minimization problem of the penalized negative log-likelihood as regularized ERM, with a penalty factor γ, given by:

$$\min_{h(\bar{\boldsymbol{t}})} \left\{ -\sum_{i=1}^{\hat{n}} \log h^2(\bar{\boldsymbol{t}}_i) + \int_{\mathcal{S}} h^2(\bar{\boldsymbol{t}})\, \mathrm{d}\bar{\boldsymbol{t}} + \gamma \|h(\bar{\boldsymbol{t}})\|^2_{\mathcal{H}_k} \right\}. \tag{5}$$

To solve this optimization problem, we show that the term can be merged into the square norm term $\gamma\|h(\bar{\boldsymbol{t}})\|^2_{\mathcal{H}_k}$ by a change of kernel technique (resulting in a new RKHS) and using the Mercer's theorem (Rasmussen and Williams 2006). The result is stated in the following lemma (Mei, Imani, and Lan 2024).

Lemma 1 (Kernel transformation)**.** *The minimization objective $J(h)$ in equation (5) can be written using a new kernel $\tilde{k}(\bar{\boldsymbol{t}}, \bar{\boldsymbol{t}}')$ as:*

$$J(h) = -\sum_{i=1}^{\hat{n}} \log h^2(\bar{\boldsymbol{t}}_i) + \|h(\bar{\boldsymbol{t}})\|^2_{\mathcal{H}_{\tilde{k}}}, \tag{6}$$

where the new kernel function defined by Mercer's theorem is $\tilde{k} = \sum_{i=1}^{\infty} \eta_i(\eta_i + \gamma)^{-1}\phi_i(\bar{\boldsymbol{t}})\phi_i(\bar{\boldsymbol{t}}')$ given that $\{\eta_i\}_{i=1}^{\infty}$, $\{\phi_i(\bar{\boldsymbol{t}})\}_{i=1}^{\infty}$ represent the eigenvalues and orthogonal eigenfunctions of the kernel function k, respectively.

The transformed new objective (6) is now solvable by applying the representer theorem. Based on this, we have the mean and covariance estimation of the intensity given by the following lemmas.

Lemma 2 (Intensity mean (Mei, Imani, and Lan 2024))**.** *Given observations $\{\bar{\boldsymbol{t}}_i\}_{i=1}^{\hat{n}}$ in $\mathcal{S}$, the posterior mean $\boldsymbol{\mu}$ of GP is the solution $\hat{\boldsymbol{g}}$ of the minimization problem (6), taking the form $\hat{\boldsymbol{g}} = \kappa^{-1}(\hat{h}^2(\bar{\boldsymbol{t}}))$, where $\hat{h}(\cdot) = \sum_{i=1}^{\hat{n}} \alpha_i \tilde{k}(\bar{\boldsymbol{t}}_i, \cdot)$.*

Lemma 3 (Intensity covariance (Mei, Imani, and Lan 2024))**.** *Given observations $\{\bar{\boldsymbol{t}}_i\}_{i=1}^{n}$ in $\mathcal{S}$, the posterior covariance matrix $\boldsymbol{A}^{-1}$ is given by:*

$$\boldsymbol{A}^{-1} = \big[\boldsymbol{\Sigma}^{-1} - \operatorname{diag}\left(\begin{cases} \frac{\ddot{\kappa}(\hat{\boldsymbol{g}}_i)\kappa(\hat{\boldsymbol{g}}_i) - \dot{\kappa}^2(\hat{\boldsymbol{g}}_i)}{\kappa^2(\hat{\boldsymbol{g}}_i)} - \ddot{\kappa}^2(\hat{\boldsymbol{g}}_i)\Delta\bar{\boldsymbol{t}} & i = j \\ -\ddot{\kappa}^2(\hat{\boldsymbol{g}}_j)\Delta\bar{\boldsymbol{t}} & i \neq j \end{cases}\right)\Bigg]^{-1}, \tag{7}$$

where $\hat{\boldsymbol{g}}_i, \hat{\boldsymbol{g}}_j$ represent $\hat{g}(\bar{\boldsymbol{t}}_i), \hat{g}(\bar{\boldsymbol{t}}_j)$, and j and $\Delta\bar{\boldsymbol{t}}$ are from the m-partition Riemann sum of the second term in equation (4) as $\int_{\mathcal{S}} \kappa(\hat{\boldsymbol{g}})\, \mathrm{d}\bar{\boldsymbol{t}} \approx \sum_{j=1}^{m} \kappa(\hat{\boldsymbol{g}}_j)\Delta\bar{\boldsymbol{t}}$.

We note that in Lemma 3, the diagonal values of the covariance matrix $\boldsymbol{A}^{-1}$ exhibit two distinct patterns, depending on whether the observation point $\bar{\boldsymbol{t}}_i$ overlaps with the partition of the Riemann sum $\bar{\boldsymbol{t}}_j$. However, the computation of intensity mean and covariance requires solving the new kernel function $\tilde{k}$ in Lemma 1, which may not yield a closed-form solution for kernels that cannot be expanded explicitly by Mercer's theorem. To tackle this, we usually adopt the Nyström approximation (Williams and Seeger 2000) to compute the results numerically.

Utility-based Decision-Making

After generating the intensity map, we use it to create multiple grid maps by producing M possible realizations of the event distribution. These realizations reflect the probabilistic estimation derived from the intensity function. Each sampled grid represents a potential configuration of event occurrences, which supports robust decision-making under uncertainty.

In this paper, we illustrate our utility-based decision-making using two different utility functions: maximizing the average return and maximizing the 95th percentile performance. The LAMPS framework can be applied to a wide range of utility functions to capture different design trade-offs in mobility planning. For each sampled grid map, we calculate the Q-function $Q(s, u)$ for all possible actions $u \in \mathcal{U}$ based on the expected cumulative reward from the current state s:

$$Q(s, u) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, u_t) \mid s_0 = s, u_0 = u, \pi\right].$$

To evaluate the utility of each action across the sampled grids, we define two utility functions, offering different criteria for decision-making

1. **Maximizing the Average Return:** The first utility function computes the mean Q-value across all sampled grid maps:
$$U_{\text{avg}}(u) = \frac{1}{M} \sum_{m=1}^{M} Q(s, u; \mathcal{G}_m),$$
where $\mathcal{G}_m$ is the m-th sampled grid.
2. **Maximizing the 95th Percentile Performance:** The second utility function evaluates the 95th percentile of Q-values across the sampled grids, capturing the action's performance in near-worst-case scenarios:
$$U_{95}(u) = \mathrm{P}_{95}\left(\{Q(s, u; \mathcal{G}_m)\}_{m=1}^{M}\right).$$

Action u^* is chosen to maximize the utility function, ensuring either the highest average return or strong performance under challenging conditions.
$$u^* = \arg\max_{u \in \mathcal{U}} (U_{\text{avg}}(u) \textbf{ or } U_{95}(u)).$$

By combining these utility functions, the UAVs can dynamically adapt their decision-making strategy based on operational requirements. Specifically, $U_{\text{avg}}(u)$ is suited for environments that prioritize overall efficiency and high throughput, while $U_{95}(u)$ is ideal for scenarios demanding robust performance under uncertainty. This approach enables UAVs to account for both average-case and worst-case scenarios, ensuring resilient and efficient mobility planning across a variety of operational conditions. Utility-based decision-making leverages the posterior distribution network states and can be easily extended to incorporate more design objectives.

Learning Algorithm for Decentralized Planning

In the LAMPS framework that is a multi-agent, decentralized mobility planning framework, a key requirement is enabling a team of agents, such as UAVs, to learn effective coordination strategies even under decentralized execution. We aim to establish a robust decision-making system that allows each UAV to act optimally based on localized observations and individual past experiences while contributing to the team's overall goals. This capability is essential for tasks demanding rapid response and adaptability, such as dynamic data collection and transmission within network environments that experience fluctuating event intensities and congestion.

To handle decentralized execution with partial observability, we employ QMIX (Rashid et al. 2020). QMIX facilitates centralized training with decentralized execution (CTDE) by factorizing the joint action-value function Q_{total} into individual agent utilities, ensuring monotonicity:
$$Q_{\text{total}}(s, \mathbf{u}) = f_{\text{mix}}(Q_1(s_1, u_1), \ldots, Q_n(s_n, u_n)),$$
where Q_{total} is the global action-value function, $Q_i(s_i, u_i)$ is the local action-value function of agent i, and f_{mix} is a mixing network preserving monotonicity. During training, we use the global reward with standard Temporal Difference (TD) loss to guide the learning of Q_{total}, i.e.,
$$\mathcal{L}(\theta) = \sum_{b} \left[y_b^{tot} - Q_{\text{total}}(s, \mathbf{u}, \theta)\right]^2,$$

Algorithm 1: The LAMPS Framework

```
Initialize UAV states and parameters.
while Step Number < Episode Length do
    Observe the current state s_t partially
    Estimate the intensity map using observations and κ.
    Generate M sampled grids from the intensity map.
    for each grid m ∈ M do
        Compute Q(s_t, u; G_m) across sampled grids.
    end for
    Evaluate utility functions U_avg(u) or U_95(u).
    Select action u_t based on the desired utility function.
    Execute u_t and transition to the next state s_{t+1}.
    Update the reward R(s_t, u_t).
end while
```

where b is a summation over the batch size of transitions sampled from the replay buffer, and $y_b^{tot} = r + \gamma \max_u Q_{\text{total}}(s, \mathbf{u}, \theta^-)$, and θ^- are the parameters of a target network. The gradient updates are back-propagated through the mixing network to update the models of per-agent networks. During execution, each agent makes her local decisions using only the local action-value functions $Q_i(s_i, u_i)$, based on the local observation s_i. The monotonicity of the mixing network ensures the individual-global-maximum (IGM) property, i.e., choosing local actions to maximize each local action value also ensures global optimality. We note that by leveraging QMIX for distributed decision-making, we only need to pool UAV observations for posterior network state estimates while the decision-making process is fully decentralized.

The QMIX framework and its implementation (Hu et al. 2021) enable UAVs to optimize their actions based on local observations and historical experiences while aligning with global objectives in this work. This design is particularly effective in dynamic and partially observable environments, enhancing robustness and adaptability. The LAMPS framework is summarized in Algorithm 1.

Evaluation

We evaluated the LAMPS framework[1] across six distinct experimental setups, each designed to test scalability and adaptability under varying network sizes and UAV counts. Specifically, each setup differs in the number of UAV agents deployed, the size of the operational map, and observability, creating unique challenges in data relay and spatial coverage. LAMPS is compared to four baseline models based on the partially observable Markov decision process (POMDP) framework, including (i) a fully observable model, (ii) a partially observable model with low observability that is a radius of 2, and (iii) a partially observable model with high observability that is a radius of 10 (iv) random action selection. To ensure consistency, we define specific constant parameters across setups. These include simulation constants essential to the framework's reproducibility, as shown in Ta-

[1]This code has been made available at https://github.com/erbayat/LAMPS

Table 1: Comparison of Algorithms by Average Total Return Across Setups

Algorithm	Setup 1	Setup 2	Setup 3	Setup 4	Setup 5	Setup 6	Setup 7	Total Reward
LAMPS	690	818	811	2660	2647	2845	3111	13582
MDP†	55	332	695	1464	1523	2557	3008	9634
POMDP[1]	740	602	327	2431	2168	1290	439	7997
POMDP[2]	581	656	770	2312	2329	2329	1840	10817
Random	326	281	319	1011	1007	998	994	4936

† fully observable model, [1] a model trained with low observability (radius = 2), [2] a model trained with high observability (radius = 10)
Setup 1: 25x25 map with 2 UAVs and observability radius = 4, Setup 2: 25x25 map with 2 UAVs and observability radius = 7
Setup 3: 25x25 map with 2 UAVs and observability radius = 12, Setup 4: 50x50 map with 7 UAVs and observability radius = 4
Setup 5: 50x50 map with 7 UAVs and observability radius = 7, Setup 6: 50x50 map with 7 UAVs and observability radius = 12
Setup 7: 50x50 map with 7 UAVs and observability radius = 20

ble 2. At the start of each simulation, a random position is selected as the center of the event distribution. A radius is then chosen such that the probability of an event occurring in each cell within this radius is 0.9, while outside the radius, it is 0.1, ensuring an overall mean probability of 0.5. Then, UAVs are placed randomly within the map. At the same time, the central office is consistently positioned at the center of the map to act as a focal point for data relay and communication. This setup introduces realistic spatial dynamics, requiring UAVs to efficiently balance coverage and connectivity to the central office.

Parameter	Value	Parameter	Value
H	100 m	N_0	-167dB
d_{cell}	20 m	Episode Length	100
t_M	3 seconds	Collision R.	-1
σ	1Mb	Out of Bound R.	-0.5
α_a	-40dB	Event Transfer R.	10
B	100kHz	Timestep R.	-0.1
p_{max}	100 mW	γ	0.99

Table 2: Constant parameters in the UAV network model.

In each setup, we tested LAMPS approach using a thresholding approach. Since observations tend to change gradually, we update our estimation every five steps and use the same estimate until the next update. After estimating event distribution intensity, we apply min-max scaling and the use threshold of 0.5 to decide whether there is an event or not. Then, we use the average utility function. Table 1 summarizes the average total returns of each algorithm across the six setups for 30 different episodes.

Table 1 compares the performance of algorithms across two main groups of setups: Setups 1-3, which involve a 25x25 map with 2 UAVs and varying observability radii, and Setups 4-7, which involve a 50x50 map with 7 UAVs. The second group has more initial information about the map due to the larger number of UAVs, which directly affects the results. In the first group, as observability increases, the performance of the model trained with low observability (radius = 2) decreases, while the performance of the model trained with high observability (radius = 10) improves significantly. In the second group, however, the increase in performance for the model trained with high observability is less pronounced, as the higher number of UAVs already provides a richer understanding of the environment. When the evaluation observability radius becomes very high (r = 20), performance declines for most algorithms due to observability shifts. Random action selection remains unaffected by changes in observability, while MDP without estimation performs better and converges to fully observable results as evaluation observability increases. Notably, the LAMPS approach outperforms all other methods in almost all cases, demonstrating its exceptional adaptability to changing observability conditions without requiring retraining. Unlike other methods that need adjustments or retraining to maintain performance in environments with varying levels of observability, this approach seamlessly adapts to different environments, whether they feature low or high observability. This adaptability ensures that the model remains effective across a wide range of scenarios, including dynamic partial observability, where conditions may shift unpredictably. Its ability to consistently deliver superior performance, regardless of changes in observability, highlights its robustness and makes it a highly effective and reliable solution for complex and evolving environments.

Conclusion

This paper presents a novel mobility planning framework that successfully bridges the gap between learning-based optimization and real-world network uncertainty. By integrating Gaussian Cox processes and utility-based planning, we have demonstrated that robust decision-making can be achieved without the traditional tradeoffs of retraining requirements. The framework's performance in UAV mobility and wireless resource management scenarios confirms its ability to maintain optimal performance across varying levels of network observability. LAMPS offers a scalable and practical solution for dynamic network environments by enabling adaptive policy deployment without retraining. These results mark a significant advancement in learning-based mobility planning, providing a foundation for robust and efficient network management systems that can operate reliably under real-world conditions.

References

Carr, S.; Jansen, N.; and Topcu, U. 2020. Verifiable RNN-based policies for POMDPs under temporal logic constraints. *arXiv preprint arXiv:2002.05615*.

Hastie, T.; and TIBSHIRANI, R. 2001. JH: The elements of statistical learning: data mining, inference, and prediction: with 200 full-color illustrations.

Hu, J.; Jiang, S.; Harding, S. A.; Wu, H.; and Liao, S.-w. 2021. RIIT: Rethinking the Importance of Implementation Tricks in Multi-Agent Reinforcement Learning. *arXiv preprint arXiv:2102.03479*.

Liu, Q.; Chung, A.; Szepesvári, C.; and Jin, C. 2022. When is partially observable reinforcement learning not scary? In *Conference on Learning Theory*, 5175–5220. PMLR.

Luong, N. C.; Hoang, D. T.; Gong, S.; Niyato, D.; Wang, P.; Liang, Y.-C.; and Kim, D. I. 2019. Applications of deep reinforcement learning in communications and networking: A survey. *IEEE communications surveys & tutorials*, 21(4): 3133–3174.

Mei, Y.; Imani, M.; and Lan, T. 2024. Bayesian Optimization through Gaussian Cox Process Models for Spatio-temporal Data. In *The Twelfth International Conference on Learning Representations*.

Meng, L.; Gorbet, R.; and Kulić, D. 2021. Memory-based deep reinforcement learning for pomdps. In *2021 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, 5619–5626. IEEE.

Møller, J.; Syversveen, A. R.; and Waagepetersen, R. P. 1998. Log gaussian cox processes. *Scandinavian journal of statistics*, 25(3): 451–482.

Moos, J.; Hansel, K.; Abdulsamad, H.; Stark, S.; Clever, D.; and Peters, J. 2022. Robust reinforcement learning: A review of foundations and recent advances. *Machine Learning and Knowledge Extraction*, 4(1): 276–315.

Nie, L.; Ning, Z.; Obaidat, M. S.; Sadoun, B.; Wang, H.; Li, S.; Guo, L.; and Wang, G. 2020. A reinforcement learning-based network traffic prediction mechanism in intelligent internet of things. *IEEE Transactions on Industrial Informatics*, 17(3): 2169–2180.

Oliehoek, F.; and Amato, C. 2016. A Concise Introduction to Decentralized POMDPs.

Poursoltani, M.; and Delage, E. 2022. Adjustable robust optimization reformulations of two-stage worst-case regret minimization problems. *Operations Research*, 70(5): 2906–2930.

Rashid, T.; Samvelyan, M.; De Witt, C. S.; Farquhar, G.; Foerster, J.; and Whiteson, S. 2020. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(178): 1–51.

Rasmussen, C. E.; and Williams, C. K. I. 2006. *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press. ISBN 026218253X.

Schölkopf, B.; Herbrich, R.; and Smola, A. J. 2001. A generalized representer theorem. In *International Conference on Computational Learning Theory*, 416–426. Springer.

Wang, W.; and Lin, Y. 2021. Trajectory Design and Bandwidth Assignment for UAVs-enabled Communication Network with Multi - Agent Deep Reinforcement Learning. In *2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall)*, 1–6.

Williams, C.; and Seeger, M. 2000. Using the Nyström method to speed up kernel machines. *Advances in neural information processing systems*, 13.

Yang, M.; Liu, G.; Zhou, Z.; and Wang, J. 2023. Partially observable mean field multi-agent reinforcement learning based on graph attention network for UAV swarms. *Drones*, 7(7): 476.

Zhang, Y.; Feng, Y.; and Rong, G. 2017. New robust optimization approach induced by flexible uncertainty set: Optimization under continuous uncertainty. *Industrial & Engineering Chemistry Research*, 56(1): 270–287.