Decentralized Communication and Energy-Aware Multi-UAV Coverage Path Planning

Felix Muller
ETH Zürich
fmuller@ethz.ch

Nick Truttmann
ETH Zürich
ntruttmann@ethz.ch

Abstract— **This paper presents a fully decentralized algorithm for multi-UAV coverage-path planning that simultaneously minimizes energy consumption and preserves network connectivity with limited communication. The approach combines centroidal Voronoi partitioning, Spanning-Tree Coverage for path generation, a fragment-merging algorithm to estimate the communication radius and asynchronous Bayesian optimization via Thompson sampling to optimize trajectories.**

The full source code and simulation framework are available at: `https://github.com/NickatETH/Decentralized_mCPP`.

Index Terms – Multi-UAV coverage path planning, centroidal Voronoi partitions, Spanning Tree Coverage, Asynchronous Bayesian optimization

I. INTRODUCTION

A. Distributed coverage path planning

Rapid growth in the use of unmanned aerial vehicles (UAVs) for applications such as environmental monitoring, surveillance, agriculture, and disaster response has created a pressing need for efficient and reliable UAV coverage path planning (CPP). Compared to single-UAV deployments, UAV teams can drastically increase coverage efficiency, mission reliability, and robustness. However, planning optimal paths for multiple UAVs introduces new challenges, especially for fixed-wing platforms, which must maintain forward motion and cannot hover. These challenges are further compounded in real-world scenarios where energy resources are limited and continuous inter-UAV communication must be maintained in potentially bandwidth-constrained mesh networks. As missions scale and complexity grows, there is a critical need for scalable distributed algorithms that allow teams of UAVs to efficiently cover complex environments while considering both energy efficiency and network connectivity constraints.

B. Related work

A wide variety of CPP and multi-UAV CPP (mCPP) strategies have been developed to address these challenges. Centralized algorithms, such as the Divide Areas based on Robots' initial Positions (DARP) [1] and Spanning-Tree Coverage (STC) [2], partition the workspace and generate coverage paths that guarantee full area coverage without overlap. Distributed and decentralized approaches often rely on Voronoi-based partitioning [3], [4], consensus-based formation control [5], [6], [7], or distributed model predictive control [8]. Energy-efficient planners increasingly incorporate realistic battery and flight models [9], [10], but relatively few methods explicitly address the dual challenge of maintaining continuous network connectivity while minimizing energy usage, particularly for fixed-wing UAVs. Recent work by Samshad and Rajawat [11] introduced a communication- and energy-aware mCPP pipeline that combines DARP, STC, and a centralized Bayesian optimization (BO) step for iterative trajectory refinement under connectivity constraints. BO, which leverages Gaussian process surrogates and acquisition functions such as *Expected Improvement* or *Thompson sampling*, has been widely adopted for efficient black-box optimization in robotics [12]. However, centralized BO can present scalability and resilience limitations in multi-agent systems. To address these issues, recent research has proposed distributed and asynchronous BO variants, including *asynchronous Thompson sampling* [13] and fully distributed BO based on stochastic policies (e.g. *Boltzmann sampling*) [14], which allows parallel optimization between agents while preserving sample efficiency and reducing communication overhead.

C. Contributions

This paper presents an algorithm for generating paths for multiple UAVs, including both fixed-wing and multirotor types. Although the individual methods employed are not novel, their integration into a complete, fully decentralized pipeline for optimal path planning is a key innovation. Specifically, we propose a fully distributed solution that combines distributed centroid Voronoi partitioning [3], [4], local STC [2], and asynchronous BO via Thompson sampling [13]. This integration enables scalable, energy-efficient, and communication-aware coverage planning for multi-UAV teams. We validate the effectiveness of our approach through simulations.

D. Organization

The remainder of this paper is structured as follows: **Section II** formalizes the multi-UAV coverage path planning problem and demonstrates the inadequacy of existing centralized solutions. **Section III** presents our decentralized coverage pipeline step-by-step, while **Section IV** reports ROS2 simulations that validate the approach, and **Section V** contextualizes the results and highlights practical deployment considerations. **Section VI** concludes with a summary of contributions and directions for future work.

II. Problem Formulation

A. System

We consider a team of N fixed-wing UAVs operating in a two-dimensional region of interest $\mathcal{A} \subset \mathbb{R}^2$. Each UAV is equipped with a downward-facing sensor with a fixed footprint and onboard localization. Communication between UAVs is possible via low-bandwidth, multi-hop radio links. The region $\mathcal{A}$ is discretized into a grid of size Δ, and each UAV is capable of observing a swath of width $\frac{1}{2}\Delta$ at a maximum speed $v_{\max}$. Furthermore, each UAV is equipped with an onboard localization and a computer which, in addition to running flight control software, provides computational resources to calculate new optimal paths in real time.

B. Objective

The goal is to adapt the communication- and energy-aware multi-UAV coverage path planning algorithm of Samshad and Rajawat [11] to operate on fixed-wing UAVs in a fully decentralized manner. Notably, the UAVs may carry out the planning process from different, potentially distant, locations within the region, highlighting the need for decentralized coordination without assuming any common starting point. We aim to compute trajectories that

(a) ensure near-complete coverage of $\mathcal{A}$
(b) maintain a connected communication network,
(c) minimize and balance energy usage across the team, and
(d) are computable on the distributed hardware of multiple drones.

C. Assumptions

1) At the time of path computation, each UAV can reliably communicate with at least one neighbor, ensuring that the overall communication graph is connected.
2) The bandwidth is limited, making heavy communication packages infeasible.
3) The motion of all UAVs is confined to a two-dimensional plane; altitude changes are not modeled.
4) The operational area $\mathcal{A}$ is convex and can be discretized into a grid with lengths Δ.
5) The robots are assumed to be homogeneous, with equal flight time and characteristics.

D. Previous limitations

Most existing multi-robot coverage path planning (mCPP) approaches rely on centralized architectures, where a central coordinator is responsible for partitioning the area, assigning tasks, and optimizing paths for the entire team. While effective in well-connected settings, such systems suffer from several drawbacks: they incur single-point failures, impose significant communication overhead, and are difficult to scale as the number of agents increases. Moreover, the assumption of persistent, high-bandwidth connectivity between all agents and the central node is rarely satisfied in real-world, large-scale, or disaster environments. As highlighted by Almadhoun et al. [15], achieving robust, scalable, and efficient coverage under realistic communication constraints remains a key open challenge for mCPP.

III. Results

A. Overview

Our pipeline generates coordinated coverage paths for a team of UAVs by iteratively evaluating and refining joint candidate configurations, each specifying Voronoi seed positions and path start positions for all agents. These candidates define how the area is divided, how each UAV will plan its local coverage path, and the expected global cost.

To efficiently explore this large combinatorial space, our approach embeds candidate generation and evaluation within a fully distributed, asynchronous BO loop, as visualized in Figure 1. At each iteration, a UAV samples a promising joint configuration from its local Gaussian process surrogate via Thompson sampling and broadcasts it to all agents. Upon receipt, the UAVs collaboratively execute distributed centroidal Voronoi partitioning, in which each agent shares only its position and weight, achieving a balanced area division with minimal communication.

Once regions are assigned, each UAV independently plans its coverage path within its assigned area using the STC method. Each agent then computes its local energy cost for the proposed configuration. The agents participate in a lightweight consensus process to collaboratively compute a single connectivity cost value, reflecting the requirements for maintaining network-wide communication under the current candidate. Each UAV then sends its local energy cost to the proposing agent, which aggregates these together with the connectivity cost to obtain a global objective value for the candidate.

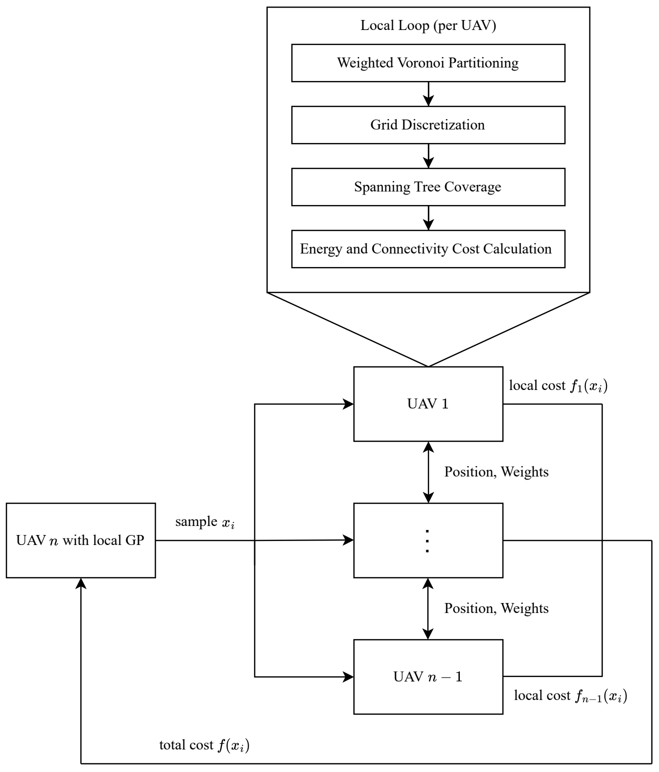

Fig. 1. Overview of one Bayesian Optimization Step

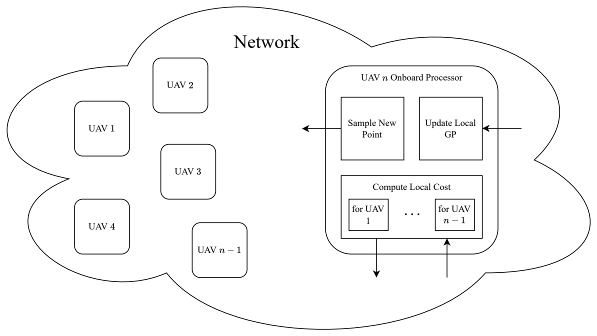

Fig. 2. Network of multiple UAVs running our algorithm

The network structure supporting this workflow is depicted in Figure 2, showing how each UAV maintains onboard optimization and estimation modules, exchanging only minimal information, such as candidate proposals, positions, weights, coordination messages and local cost values. This minimal communication approach ensures scalability and robustness, even under bandwidth constraints or network instability.

At any time, multiple UAVs may perform updates concurrently, or even several updates on one UAV in parallel. Hardware constraints can be accommodated by adjusting the number of BO iterations that run simultaneously. By iteratively repeating this distributed loop, the team progressively refines both region assignments and individual coverage paths, converging toward energy-efficient and communication-aware multi-UAV coverage plans. The following sections provide detailed descriptions of each component.

B. Area Division

To divide the area among the robots, the centroidal Voronoi partitioning algorithm is utilized. It is derived from a distributed version of the Lloyd algorithm [16] that divides the area based on its closest neighbor:

$$V_i = \{ q \in Q \mid \|q - p_i\| \leq \|q - p_j\|, \forall j \neq i \}. \quad (1)$$

Contrary to the original DARP algorithm, the Voronoi partitions do not have a balancing mechanism. For this reason, we introduce a weight to the partitioning algorithm as proposed in [17] and use the notation introduced in [18]. Let each robot i have seed point $p_i \in \mathbb{R}^2$ and weight w_i.

$$V_i = \{ q \in Q \bigm| \|q - p_i\|^2 - w_i \leq \|q - p_j\|^2 - w_j, \ \forall j \neq i \}. \quad (2)$$

The *area* $\mathcal{A}_i$ and *centroid* $c_i \in \mathbb{R}^2$ of V_i are

$$\mathcal{A}_i = \int_{V_i} dq, \qquad c_i = \frac{1}{\mathcal{A}_i} \int_{V_i} q \, dq. \quad (3)$$

Given a prescribed target area $\mathcal{A}_i^*$, update each weight by a simple gradient descent step:

$$w_i^{(t+1)} = w_i^{(t)} - \gamma \left(\mathcal{A}_i - \mathcal{A}_i^* \right), \quad (4)$$

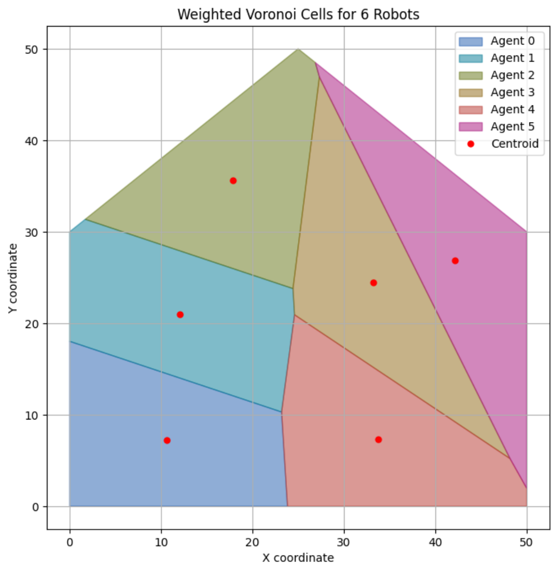

Fig. 3. Converged weighted Voronoi partitions

where $\gamma > 0$ is a fixed learning rate. For homogeneous robots with equal energy capacity, $\mathcal{A}^*$ was calculated by

$$\mathcal{A}_i^* = \mathcal{A}_{eq}^* = \frac{\mathcal{A}_{\text{tot}}}{n_{agents}} \text{ where } i \in 1, 2, ..., n_{agents} \quad (5)$$

After each weight-update iteration, we assess whether the areas of all robots are sufficiently close to the ideal value. We define the *relative area error* e_i for each robot as

$$e_i = \frac{\left| \mathcal{A}_i - \mathcal{A}^* \right|}{\mathcal{A}_i^*}, \qquad i = 1, 2, \ldots, n_{agents}. \quad (6)$$

Intuitively, e_i measures how far the actual coverage of each robot deviates from the ideal, as a fraction of A_i^*.

To decide on termination, we aggregate these into the *worst-case error* E_{max}:

$$E_{\max} = \max_{1 \leq i \leq n_{agents}} e_i. \quad (7)$$

If every cell is within a prescribed tolerance ε of the target area, i.e.

$$E_{\max} < \varepsilon, \quad (8)$$

we declare convergence and stop the iterative procedure. The result of this process is visualized in Figure 3.

Although in theory the algorithm only requires knowledge of the robots along all edges of the Delaunay graph [19], it was decided to share the position and weight information of all robots with each other robot. This increase in bandwidth required is minimal and allows the omission of the Delaunay graph calculation.

Communication is limited to two floats for the position of the robot and its respective weight w_i. The total message exchange scales with the number of iterations that is required for the algorithm to converge. Note that gossiping

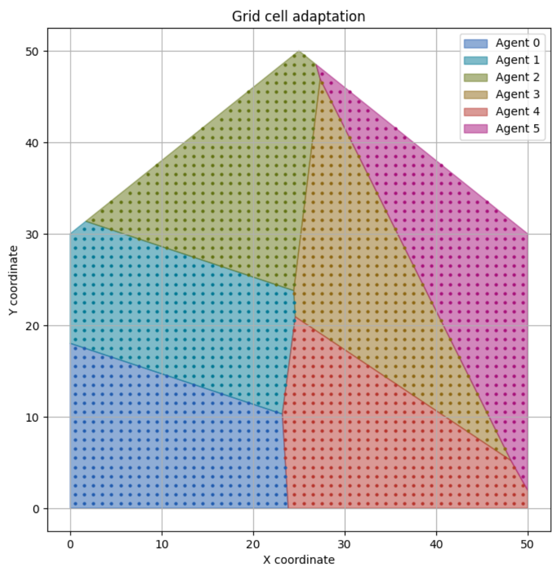

Fig. 4. Grid overlayed on partitions

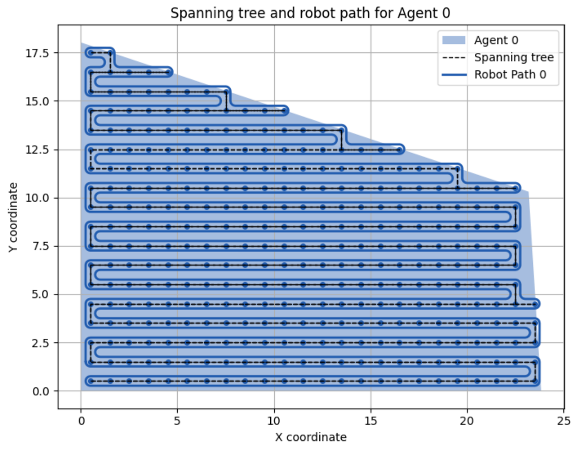

Fig. 5. Spanning tree and robot path

of other robots also takes some bandwidth, but this is limited for small numbers of robots and can be completely limited even further when calculating the Delaunay graph.

Following the division of the workspace, the area is discretized into a uniform grid with cells of size Δ. Each grid cell must be exclusively assigned to one robot to ensure complete and non-overlapping coverage. Let $\mathcal{G} \subset \mathbb{Z}^2$ denote the set of all grid cells, and let $\mathcal{G}_i \subset \mathcal{G}$ be the subset assigned to robot i. Then the assignment must satisfy the following:

$$\forall g \in \mathcal{G}, \quad \exists!\, i \in \{1, 2, \dots, N\} \text{ such that } g \in \mathcal{G}_i \tag{9}$$

This condition guarantees that every cell of the grid g belongs exactly to the region assigned to a robot $\mathcal{G}_i$, including the boundary cells. The resulting grid assignments shown in Figure 4 are then used for the subsequent path planning stages.

C. Path Planning

After partitioning the workspace into a uniform grid of square cells with edge length Δ, we construct the 4-neighbor graph and compute a depth first STC as introduced by Gabriely and Rimon [2]. Because the root choice does not affect completeness, we fix it deterministically as the lexicographically first cell,

$$p_{\text{root}} = \arg\min_{p \in \text{cells}} (p_y, p_x),$$

i.e., scanning rows bottom-to-top before columns left-to-right. We offset the STC path rooted at p_{root} by the starting point s_i in a counterclockwise direction. Note that the starting point $p_i \in [0, 1]$ is normalized to allow for varying segment lengths and describes the fraction of the starting point along the path. The STC tour is then traced *parallel* to the edges of the tree, offset by $\frac{1}{4}\Delta$ from the cell centers; this guarantees that the path covers each cell exactly once without gaps or redundant overlap [2]. To accommodate kinematic constraints and avoid complete stops at right-angle junctions, every corner is replaced by a circular arc of radius $\frac{1}{4}\Delta$, which yields a C^1-continuous path, as seen in Figure 5.

Each half grid cell $\frac{1}{2}\Delta$ corresponds to the distance that the UAV can cover in one sweep. For simulation, we used 100m as $\frac{1}{2}\Delta$, which is conservative for a medium-sized UAV with high-quality sensors. To facilitate calculations, the grid and path calculations are carried out on a small grid, and later on scaled up to the real dimensions.

D. Energy calculation

Detailed power models such as [20] capture aerodynamics well, at the expense of significant computational effort. Because our UAVs fly at a fixed speed and always use the same turn radius, we adopt a lightweight two-regime approximation: *straight–line cruise* and *turning*.

Let the trajectory be partitioned into N straight segments indexed by $i \in \mathcal{S}$ and M turn maneuvers indexed by $j \in \mathcal{T}$. With constant cruise power P_c and elevated turn power $P_t = \gamma P_c$ (where $\gamma = 1.5$ is the turn factor), the total energy expenditure E is

$$E = \underbrace{\sum_{i \in \mathcal{S}} P_c \, \Delta t_i}_{\text{cruise}} + \underbrace{\sum_{j \in \mathcal{T}} P_t \, \Delta t_j}_{\text{turns}} = P_c \sum_{i \in \mathcal{S}} \Delta t_i + \gamma P_c \sum_{j \in \mathcal{T}} \Delta t_j. \tag{10}$$

A closely related variant with three regimes: *hover*, *forward flight*, and *turn*, was shown to be effective in [11].

Algorithm 1 Distributed Fragment-Merge Algorithm

```
1: procedure INITIALIZATION
2:     for all agents $v$ do
3:         fragID$_v \leftarrow v$        ▷ each in its own fragment
4:         root$_v \leftarrow v$
5:         children$_v \leftarrow \emptyset$
6:         $\mathcal{N}_v \leftarrow \emptyset$
7:     end for
8: end procedure

9: procedure GETBEST($\mathcal{N}_v$)
10:     $(nb, f) \leftarrow$ FINDCLOSESTNEIGHBOUR($\mathcal{N}_v$)
11:     $w_v \leftarrow$ CALCULATEDISTANCE($nb$)
12:     Send REPORT($nb$, $f$, $w_v$) to $root_v$
13: end procedure

14: procedure ONREPORT(nb, f, w) , at agent $v$
15:     store $(nb, f, w)$
16:     $n_{report} + = 1$
17:     if $n_{report} = (n_{children} + 1)$ then
18:         Pick $v^* = \arg\min_v w_v$ where $f' = f_v$
19:         Save $f_v^*$
20:         Send MERGEREQUEST($f'$) to agent $u^*$
21:     end if
22:     for all other children $v' \neq v^*$ do
23:         Send REJECT(nb) back to $v'$
24:     end for
25: end procedure

26: procedure ONREJECT(nb), at agent $v$
27:     Send GETBEST($\mathcal{N}_v \setminus \{nb\}$) back to $v'$
28: end procedure

29: procedure ONMERGEREQUEST(f'), at agent $v$
30:     if $v \neq$ root$_v$ then
31:         Forward MERGEREQUEST($f'$) to root$_v$
32:     else
33:         if $f' = f_v^*$ then
34:             Send ACCEPT($f_v$) to sender
35:             Merge fragment $f'$ into $f_v$
36:             root$_v \leftarrow \min($root$_v$, root$_{v'})$
37:             children$_v \leftarrow$ children$_v \cup$ children$_{v'}$
38:             Update $w_f = \arg\max(w_f, w_{f'})$
39:             CleanReports; $n_{\text{reports}} \leftarrow 0$
40:         else
41:             Send Reject($v$) to sender
42:         end if
43:     end if
44: end procedure

45: procedure REPORTUP(f)
46:     if $n_{children} = n_{agents}$ then
47:         Abort all agents $v$
48:         Return $w_f$
49:     end if
50: end procedure
```

E. Communication radius estimation

To estimate the maximum pairwise distance between agents in a connected graph at a given time τ, we construct a minimum spanning tree (MST) over the agent graph using a simplified Gallager–Humblet–Spira (GHS) algorithm [21]. In our implementation outlined in Algorithm 1, we abandon the original level-based merging scheme and instead have each fragment's $root_{v_i}$ maintain an explicit *list of its member agents children*$_v$, and the *total number of agents* n_{agents}. This approach is efficient for the moderate-sized swarms we target, while remaining manageable for the onboard memory of each agent. For very large networks, where maintaining large membership lists may become infeasible, one should revert to the standard GHS level mechanism. With our adaptation, the protocol operates effectively as an asynchronous, distributed variant of Kruskal's MST algorithm [22]. When two fragments merge, we designate the agent with the smaller fragID$_v$ as the new $root_f$. This rule is arbitrary, but deterministic, ensuring consistent root selection without additional coordination.

Each agent keeps track of the following things:

- fragID$_v$: fragment identifier (initially its own ID),
- root$_\text{v}$: root, responsible for managing the fragment,
- children$_\text{v}$: fragment children (only for the root),
- max_radius$_\text{v}$: longest edge in the fragment so far.

We estimate the maximum communication radius by sampling discrete instants along each agent's trajectory. Let $\{t_k^{(s,p)}\}_{k=1}^{N_{s,p}} \subset [0, 1]$ be the sampled time instants along the trajectory that uses the position seed p and starting points s. The estimated maximum communication radius for this seed–start pair is the largest sampled value:

$$\hat{r}_{s,p} = \max_{k=1,\ldots,N_{s,p}} r\big(t_k^{(s,p)}\big). \tag{11}$$

a) Sampling policy: The refined bound of [11], which leverages the $2v_{\max}$-Lipschitz continuity of the communication radius, i.e. a worst-case rate based on the maximum relative distance two UAVs can open per unit time, could in principle, reduce the number of required samples. However, its computation scales with the path length and is therefore impractical for the large workspaces studied here. Centroidal Voronoi partitioning produces cells that are oftentimes nearly congruent and convex, yielding agent trajectories with long straight legs and gentle turns, whereas DARP as employed in [11] can generate narrow "snakes" and acute angles. Monte-Carlo simulations confirm that, under these well-behaved conditions, the empirical variation of $r(\tau)$ is significantly smoother than the worst-case Lipschitz bound L. Empirically, a simple uniform oversampling strategy is both efficient and sufficiently accurate as seen in Figure 6.

b) Normalized time grid: To integrate the distributed BO step, we parameterize the sampling time by τ, the

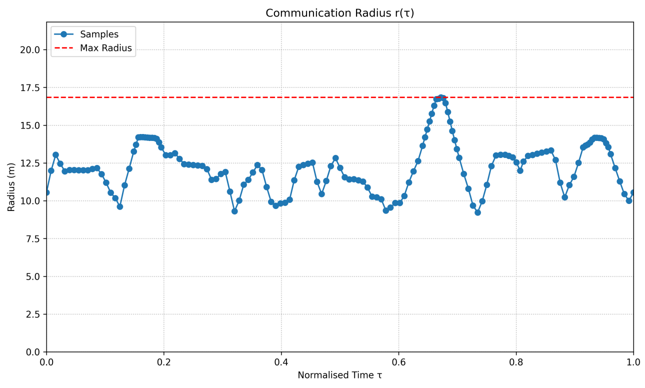

Fig. 6. Communication range requirement over a sample path

fraction of path length already traversed rather than by a time index. This parameterization naturally accommodates agents with unequal path lengths, which is an effect that arises from the slack in the centroidal Voronoi partitioning.

F. Distributed BO

To overcome the limitations of centralized BO in multi-UAV coverage path planning, namely scalability, resilience, and the risk of a single point of failure, we employ a fully distributed, asynchronous BO framework based on Thompson sampling [13]. In this architecture, all UAVs participate as independent optimization agents communicating over a low-bandwidth peer-to-peer network. At the start of the optimization process, a candidate set is generated, representing joint configurations of Voronoi seeds p_i and path starting positions s_i for the entire fleet. Each UAV maintains its own Gaussian process (GP) surrogate model over this candidate space, continually updated as new joint parameter–cost observations are shared among the agents.

The optimization proceeds asynchronously: whenever an agent becomes available, it draws a function sample $\tilde{f}$ from its GP posterior and selects the next candidate as

$$x^* = \arg\min_{x \in \mathcal{X}} \tilde{f}(x),$$

where $\mathcal{X}$ is the set of all candidate configurations. This acquisition strategy, known as Thompson sampling, is inherently stochastic, each agent samples a different function from the posterior even if their surrogate models are identical. This non-determinism is essential for enabling a fully asynchronous distributed architecture: agents can propose and evaluate candidates independently without the risk of duplicating efforts, which would otherwise occur if a deterministic acquisition function were used. In addition, Thompson sampling naturally balances exploration and exploitation: by sampling from the posterior, the agent is likely to select both uncertain regions of the parameter space (exploration) and areas predicted to yield low costs (exploitation), thereby achieving efficient optimization of expensive black-box objectives in a parallel, decentralized manner.

This candidate is then broadcast to the other agents, at which point the UAVs collaboratively execute area division, path planning, and the evaluation of energy and connectivity costs for the proposed configuration, following the distributed procedures described in the preceding sections. The partial evaluations from all UAVs are sent to the agent that originally proposed the candidate, which aggregates them into a global cost. Specifically, for each candidate configuration x, the global cost function is defined as

$$f(x) = \lambda \left(\sum_{i=1}^{N} E_i(x) \right) + r_{\max}(x),$$

where $E_i(x)$ denotes the energy expenditure of UAV i for configuration x, and $r_{\max}(x)$ is the maximum distance, at any point along the mission, between a UAV and its nearest neighbor. This formulation captures both the total energy usage and a measure of network connectivity, as minimizing $r_{\max}(x)$ promotes tighter formations that maintain communication links.

This total cost and the associated parameter vector are then broadcast to the other agents, allowing each UAV to update its local observation history and Gaussian process model accordingly, thereby maintaining consistency across the fleet as optimization progresses. Importantly, this entire process is fully asynchronous and decentralized: agents propose, evaluate, and incorporate new samples at their own pace, without waiting for global synchronization or a central coordinator.

This architecture ensures that the optimization progresses efficiently even in the presence of network delays, heterogeneous agent capabilities, or temporary disconnections. Faster agents naturally drive more of the exploration, while slower or recovering agents simply integrate new shared observations as they become available. Communication overhead is kept minimal, involving only parameter proposals and cost values per candidate evaluation, and the system is robust to individual agent failures. By maintaining consistent surrogate models across all agents via shared observations, the method preserves the theoretical sample efficiency and regret guarantees of standard BO, while enabling scalable, resilient, and energy- and communication-aware multi-agent coverage optimization in realistic, bandwidth-constrained environments.

IV. Simulation

The simulation framework for decentralized, communication- and energy-aware multi-UAV coverage path planning was implemented using the Robot Operating System 2 (ROS2) [23] middleware as a communication backbone. The architecture comprises a controller node and multiple autonomous UAV agent nodes, each realized as a ROS2 Node and communicating using standard message types, with tailored Quality of Service (QoS) profiles to balance reliability and bandwidth. In the distributed system, each UAV gets a controller node and multiple agent nodes. The simulation runs in real-time, without the

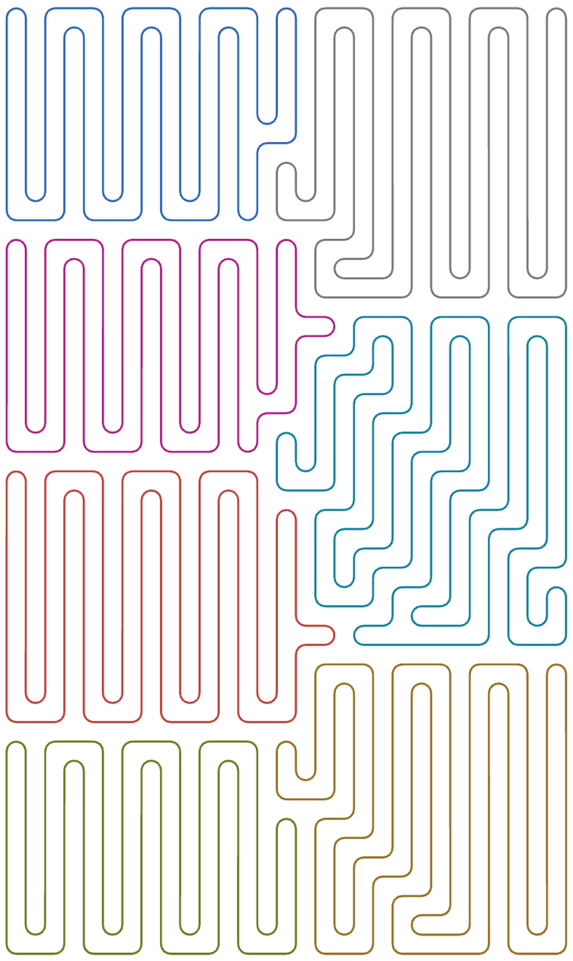

Fig. 7. Agent paths generated in real-time

requirement for any synchronization beyond the exchanged messages required for tasks. In the current framework, the computation and messaging of each agent are simulated on a single computer, but without a global clock or barrier synchronization. However, this lack of synchronization can lead to minor discrepancies in the centroidal Voronoi partitions, manifesting as small coverage gaps or overlapping cells. This is not a concerning issue and can be dealt with by overlaying a lightweight local collision avoidance controller on top of the path planner.

To assess the impact of distributed Bayesian optimization (BO), we compared its performance to a random sampling baseline. Figure 8 displays the cost function $f(x) = r_{max} + \lambda E_{tot}$ over successive iterations for both approaches. While both methods yield improvements as more candidates are evaluated, BO consistently identifies lower-cost solutions than random sampling, as reflected in the stepwise reduction in the minimum-cost trace. This demonstrates that distributed BO can discover more efficient, communication-aware coverage plans under the same evaluation budget.

Future work will focus on closing the gap between our real-time, single-host emulator and a genuinely distributed UAV deployment. To address these shortcomings, we will refactor the codebase so that each UAV operates in its own process or on separate machines, communicating over real network interfaces subject to realistic latency, jitter, and packet loss. This redesign requires implementing efficient buffer management, and node-failure recovery to ensure robust operation under adverse conditions. We will also integrate a local collision avoidance controller, to prevent crashes in the event of overlapping line segments. Additionally, the Matern kernel used in the BO step can be tuned to converge faster and the overall speed of the simulation should be improved.

V. Discussion

Our decentralized coverage–control pipeline consisting of centroidal Voronoi partitioning, an MST generator using fragments, and lightweight analytic cost coupled with a distributed optimization loop achieves reliable performance for swarms of roughly a dozen agents. All computations rely solely on neighbor-to-neighbor messages, so no global map or central coordinator is required. The optimizer balances energy expenditure and communication range, producing trajectories that meet radio-range limits without degrading flight efficiency. Because the constant-plus-turn energy model is closed-form, re-planning can be performed in real time.

Limitations: The decision to replace the GHS level mechanism with explicit membership lists simplifies implementation and reduces messaging for moderate swarm sizes, but that bookkeeping would become unwieldy in very large collectives; a hybrid strategy that reintroduces levels beyond a threshold is advisable. The centroidal Voronoi partitioning algorithm results in similarly sized areas. Modifying the partitioning would allow us to use nonhomogeneous robots or balance the available energy on each UAV. Uniform oversampling of the communication radius has worked well in practice, yet offers no worst-case guarantee; adaptive sampling driven by curvature estimates could close this gap. Finally, the simplified energy model ignores wind, altitude changes, and nonlinearities in the powertrain, so a more complete model will be required for high-fidelity estimation.

VI. Conclusions

This study demonstrates that a decentralized, communication-aware coverage strategy can meet both energy and connectivity objectives with modest computational cost. While several components need to be hardened for application in large, real swarms, the proposed architecture provides a promising foundation for scalable multi-UAV operations.

Extending the planner to heterogeneous fleets with diverse speed envelopes and battery capacities would greatly broaden its applicability and could be done with limited time investment. Ultimately, hardware-in-the-loop experiments are essential for capturing radio attenuation, GPS noise, and

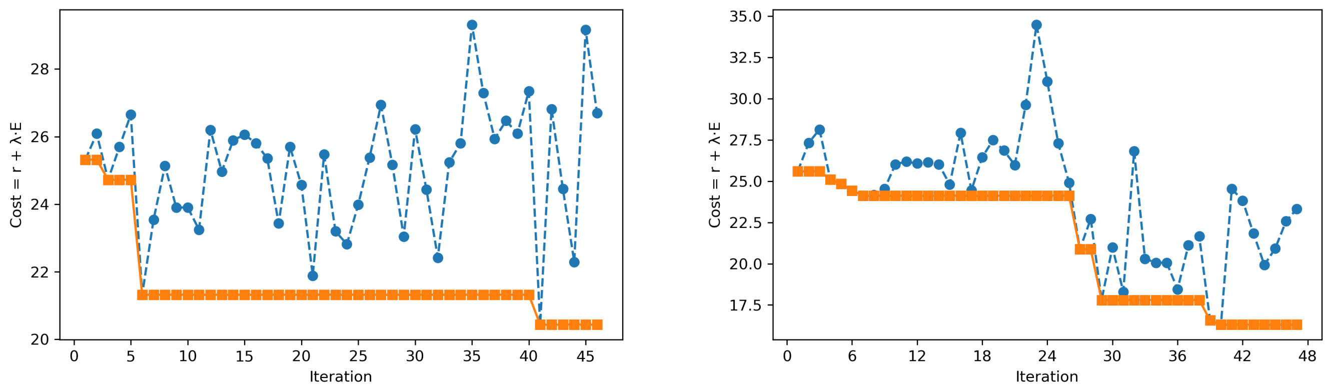

Fig. 8. Evolution of the cost function $f(x) = r + \lambda E$ over optimization iterations: **Left:** random sampling baseline. **Right:** distributed Bayesian optimization (BO) with Thompson sampling. Blue points show cost per iteration; orange points track the incumbent minimum. BO achieves faster and more reliable convergence to lower costs.

aerodynamic effects that cannot be fully reproduced in simulation.

References

[1] A. C. Kapoutsis, S. A. Chatzichristofis, and E. B. Kosmatopoulos, "Darp: Divide areas algorithm for optimal multi-robot coverage path planning," *Journal of Intelligent & Robotic Systems*, vol. 86, pp. 663–680, 2017.

[2] Y. Gabriely and E. Rimon, "Spanning-tree based coverage of continuous areas by a mobile robot," *Annals of Mathematics and Artificial Intelligence*, vol. 31, pp. 77–98, 2001.

[3] M. Schwager, D. Rus, and J.-J. Slotine, "Decentralized, adaptive coverage control for networked robots," *The International Journal of Robotics Research*, vol. 28, no. 3, pp. 357–375, 2009.

[4] M. Todescato, A. Carron, R. Carli, G. Pillonetto, and L. Schenato, "Multi-robots gaussian estimation and coverage control: From client–server to peer-to-peer architectures," *Automatica*, vol. 80, pp. 284–294, 2017.

[5] N. H. M. Li and H. H. T. Liu, "Formation uav flight control using virtual structure and motion synchronization," in *2008 American Control Conference*, 2008, pp. 1782–1787.

[6] W. Suo, M. Wang, D. Zhang, Z. Qu, and L. Yu, "Formation control technology of fixed-wing uav swarm based on distributed ad hoc network," *Applied Sciences*, vol. 12, no. 2, p. 535, 2022.

[7] L. He, P. Bai, X. Liang, J. Zhang, and W. Wang, "Feedback formation control of uav swarm with multiple implicit leaders," *Aerospace Science and Technology*, vol. 72, pp. 327–334, 2018.

[8] M. Ariola, M. Mattei, E. D'Amato, I. Notaro, and G. Tartaglione, "Model predictive control for a swarm of fixed wing uavs," in *Proceedings of the International Council of the Aeronautical Sciences (ICAS) 2016*, 2016.

[9] D. Datsko, F. Nekovář, and R. Pešek, "Energy-aware multi-uav coverage mission planning with optimal speed of flight," *IEEE Robotics and Automation Letters*, vol. 9, no. 3, pp. 1234–1241, 2024.

[10] L. Bauersfeld and D. Scaramuzza, "Range, endurance, and optimal speed estimates for multicopters," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2953–2960, 2022.

[11] M. Samshad and K. Rajawat, "Communication and energy-aware multi-uav coverage path planning for networked operations," *arXiv preprint arXiv:2411.02772*, 2024.

[12] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2016.

[13] K. Kandasamy, A. Krishnamurthy, J. Schneider, and B. Poczos, "Parallelised bayesian optimisation via thompson sampling," in *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, vol. 84, Apr 2018, pp. 133–142.

[14] J. Garcia-Barcos and R. Martinez-Cantin, "Fully distributed bayesian optimization with stochastic policies," *CoRR*, vol. abs/1902.09992, 2019. [Online]. Available: http://arxiv.org/abs/1902.09992

[15] R. Almadhoun, T. Taha, L. Seneviratne, and Y. Zweiri, "A survey on multi-robot coverage path planning for model reconstruction and mapping," *SN Applied Sciences*, vol. 1, 08 2019.

[16] Y. Elmaliach, N. Agmon, and G. A. Kaminka, "Multi-robot coverage: Theoretical foundations and experimental evaluation," *arXiv preprint arXiv:0903.3642*, 2009.

[17] F. Aurenhammer, "Power diagrams: properties, algorithms and applications," *SIAM Journal on Computing*, vol. 16, no. 1, pp. 78–96, 1987.

[18] A. Pierson, L. C. Figueiredo, L. C. A. Pimenta, and M. Schwager, "Adapting to sensing and actuation variations in multi-robot coverage," *The International Journal of Robotics Research*, vol. 36, no. 3, pp. 337–354, 2017.

[19] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*. Springer, 2008.

[20] Y. Zeng and R. Zhang, "Energy-efficient uav communication with trajectory optimization," *IEEE Transactions on Wireless Communications*, vol. 16, no. 6, pp. 3747–3760, 2017.

[21] R. G. Gallager, P. A. Humblet, and P. M. Spira, "A distributed algorithm for minimum-weight spanning trees," *ACM Transactions on Programming Languages and Systems*, vol. 5, no. 1, pp. 66–77, 1983.

[22] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical Society*, vol. 7, pp. 48–50, 1956.

[23] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.abm6074