Cloud-Supported Coverage Control for Persistent Surveillance Missions

Jeffrey R. Peters†
Department of Mechanical Engineering
University of California
Santa Barbara, CA 93106
Email: jrpeters@engr.ucsb.edu

Sean Wang
Department of Mechanical Engineering
University of California
Santa Barbara, CA 93106
Email: seanwang@umail.ucsb.edu

Amit Surana
Systems Department
United Technologies Research Center
East Hartford, CT 06118
Email: suranaaa@utrc.utc.com

Francesco Bullo
Department of Mechanical Engineering
University of California
Santa Barbara, CA 93106
Email: bullo@engr.ucsb.edu

A cloud-supported coverage control scheme is proposed for multi-agent, persistent surveillance missions. This approach decouples assignment from motion planning operations in a modular framework. Coverage assignments and surveillance parameters are managed on the cloud, and transmitted to mobile agents via unplanned and asynchronous exchanges. These updates promote load-balancing, while also allowing effective pairing with typical path planners. Namely, when paired with a planner satisfying mild assumptions, the scheme ensures that (i) coverage regions remain connected and collectively cover the environment, (ii) regions may go uncovered only over bounded intervals, (iii) collisions (sensing overlaps) are avoided, and (iv) for time-invariant event likelihoods, a Pareto optimal configuration is produced in finite time. The scheme is illustrated in simulated missions.

1 Introduction
1.1 Cloud-Supported Multi-Agent Surveillance

Autonomous sensors are used in many modern surveillance missions, including search and rescue [1], environmental monitoring [2], and military reconnaissance [3]. Such missions often require agents to periodically exchange data with a central cloud (repository) and, when operating in non-ideal environments or under hardware limitations, these potentially sporadic exchanges may be the only means of sharing real-time information across agents. For example, autonomous underwater vehicles are virtually isolated due to difficulties in underwater data transfer, and rely on periodic surfacing to communicate with a tower [4]. Other applications that may have this constraint include data mules that periodically visit ground robots [5], and supervisory missions that require unmanned vehicles to send data to a remotely located operator [6]. Such scenarios require robust and flexible frameworks for real-time autonomous coordination.

Single-agent surveillance strategies range from simple a priori tour construction [7] to more complex methods involving Markov chains [8], optimization [9], or Fourier analysis [10]. However, it is not straightforward to generalize single-agent approaches for multi-agent missions: Naive approaches where each agent follows an independent policy often result in poor performance and introduce collision risks, while other generalizations may require joint optimizations that are intractable for even modestly sized problems [11]. Distributed control can sometimes alleviate scaling issues; however, such setups typically rely on ideal peer-to-peer data transfer making them ill-posed in many cloud-based setups.

In contrast, decomposition-based approaches to multi-agent surveillance, which decouple the assignment and routing problem by dividing the agent workspace, are popular in practice as they offer a straightforward, modular framework to reasonably accomplish surveillance goals, despite sacrificing optimality. However, in cloud-based architectures that rely solely on agent-cloud exchanges for real-time data transfer, implementation of such an approach is not straightforward using existing approaches to dynamic workspace de-
composition. Indeed, in this case, updated mission information is only relayed to one agent at a time, rendering traditional partitioning schemes, which rely on complete or pairwise coverage updates, impossible, and existing strategies that utilize single-agent updates, e.g., [12], may introduce undesirable configurations or collision-risks (Fig. 1).

This work extends existing literature by introducing a cloud-supported, decomposition-based framework for multi-agent persistent surveillance that promotes effective coverage without introducing collision (redundant sensing) risks and without requiring ideal or pre-planned data exchanges. As such, the proposed framework allows agents to effectively respond to changes in the environment (spatial event likelihood) without having to be collected and re-deployed due to the inability to transmit data to all agents simultaneously. That is, when mission goals (captured by the event likelihood) change after agents are already in the field, the cloud incrementally relays the updated information through sporadic exchanges with individual agents, driving them to a new coverage configuration without introducing undesirable intermediate states. This framework also allows the cloud to act as a data fusion center if necessary, dynamically combining local sensory information collected by individual agents.

1.2 Related Literature

Multi-agent coverage control problems have generated extensive research. Typical strategies involve optimization [13], auctions [14], meta-heuristics [15], potential fields [16], Markov decision processes [17], among others [18]. Of particular relevance is multi-agent persistent surveillance (monitoring), in which a sensor team is tasked with continual surveillance of some region, requiring sub-regions to be visited multiple (or infinitely many) times to minimize a cost, e.g., the time between visits or the likelihood of detecting events [11]. Persistent surveillance is a generalization of patrolling, where agents follow closed tours to protect or supervise an environment. Most current solutions to patrolling problems utilize operations research, non-learning multi-agent systems, and multi-agent learning [19]; however, formulations are often one-dimensional and solutions usually reduce to “back and forth” motions that do not readily extend to general scenarios, e.g., [20].

The framework herein employs a type of workspace decomposition in constructing solutions to the multi-agent problem. In the context of general robotic applications, the term workspace decomposition refers to any number of strategies that are employed in order to represent a complex workspace or solution space in a simpler, often lower-dimensional form. For robotic motion planning involving obstacles, this often involves the representation of the free configuration space in a graphical or modular form that captures essential connectivity properties, e.g., typical roadmap and cellular decomposition planning methods take this approach [21, 22]. In multi-agent applications, workspace decomposition methods can also be used to reduce the complexity of a problem through the assignment of sub-tasks to individual agents, i.e., for task assignment [23]. This is the approach taken herein to reduce the multi-agent problem into a set of single-agent problems. Since the assignment problem is often difficult to solve (it has been shown to be NP-hard in some domains [23]), multi-agent planning solutions based on this type of assignment usually lead to sub-optimal solutions. Despite this drawback, assignment-based approaches remain popular in practice due to their simplicity and scalability [11]. Outside of workspace decomposition, other strategies for reducing multi-agent planning problems into a set of single-agent problems include naive approaches, where each agent independently determines its own actions by locally solving a complete problem over the full workspace, and, provided sufficiently reliable sensing and data-transfer capabilities, distributed approaches, in which agents each solve a sub-problem based on information shared over a communication network [24].

For planar persistent surveillance, decomposition-based approaches typically consist of two primary components: partitioning and single-agent routing. The most common approaches to optimal partitioning in convex environments are based on Voronoi partitions [25], and effective schemes exist for constructing centroidal Voronoi, equitable, or other types of optimal partitions under communication, sensing, and workload constraints [26–28]. Non-convex workspaces are typically addressed by representing the environment as a graph, on which a number of graph partitioning schemes can be used [29]. In robotics, discrete partitioning is often considered under communication constraints, e.g., pairwise gossip [30] or asynchronous one-to-base station communication [12]. Our proposed scheme most closely mirrors [12], in which agents communicate sporadically with a base station; however, our approach employs additional logic to ensure the resultant coverage regions retain properties that are consistent with effective, decomposition-based surveillance.

Single-agent path planners for persistent surveillance typically operate on graphs [31, 32], and classical problems, e.g., the Traveling Salesperson Problem [33], often play a key role. Stochasticity can be introduced using tools such as Markov chains [8]. Schemes for non-discrete spaces (open subsets of Euclidean space), are less common. Here, strategies include a priori construction of motion routines [34], adaptation of static coverage strategies [35], the use of random fields [36], and spectral decomposition [10]. The modular framework herein incorporates any single-agent planner satisfying mild assumptions (see Section 5).

Remarkably few papers explicitly consider the implications of combining dynamic partitioning with continuous routing for multi-agent persistent surveillance. Existing research is mostly preliminary, considering ideal data transfer
and simplistic methods. The authors of [37] employ a sweeping algorithm for partitioning and guide vehicle motion via lawn-mower patterns, while [38] uses rectangular partitions and a reactive routing policy which, in ideal cases, reduces to spiral search patterns. The authors of [39] use slightly more sophisticated partitioning in tandem with lawn-mower trajectories. In [40], partitions are based on the statistical expectation of target presence, but ideal communication is assumed. Others, e.g., [41], employ decomposition-based structures, but focus on task-assignment without detailed treatment of the combined assignment/routing protocol.

Our work uses a cloud-supported computational framework. Cloud-based robotic infrastructures (cloud robotics) have generated growing research interest, as they can provide many benefits to complex systems, such as the storage and analysis of “big data,” the availability of parallel grid computing, the potential for collective learning, and the utilization of human computation [42]. In multi-agent systems, cloud-supported schemes have been used for tasks such as collective optimization [43], rendezvous [44], and coordinated task-assignment [12]. Our use of the cloud-based architecture is primarily motivated by supervisory systems involving unmanned vehicles. Here, this architecture arises naturally, since mobile agents are required to transmit sensor data to a remotely located human operator for analysis (thus requiring a central repository), and harsh operational environments often prohibit reliance on peer-to-peer communication. However, the proposed framework is suitable for use in any similar setup, and can also run in parallel with other cloud-supported operations (e.g., data analysis, learning, etc.) within complex missions.

1.3 Contributions

This work develops a cloud-supported, decomposition-based, multi-agent coverage control framework for persistent surveillance, which relies only on sporadic, unscheduled exchanges between agents and a central cloud for data transfer. In particular, we develop a sophisticated partitioning and coordination scheme that can be effectively paired with single-agent trajectory planners. This naturally leads to the complete, modular framework in which high-level coverage is coordinated on the cloud and agent trajectories are generated independently via on-board planners. We encompass realistic constraints including restrictive communication, dynamic environments, and non-parametric event likelihoods.

Specifically, our dynamic partitioning scheme only requires agents to sporadically upload and download data from the cloud. The cloud runs updates to govern region assignments, while also manipulating high-level surveillance parameters. We prove that this update structure has many desirable properties: coverage regions collectively form a connected $m$-covering and evolve at a time-scale that allows for appropriate agent reaction, no subregion remains uncovered indefinitely, local likelihood functions have disjoint support, among others. For certain cases, we show that the set of coverage regions and associated generators converges to a Pareto optimal pair in finite time. We show that the combination of our partitioning scheme with a generic trajectory planner ensures collision (sensing overlap) avoidance, provided the planner obeys natural restrictions. Finally, we illustrate our framework through numerical examples.

Our partitioning scheme is primarily motivated by [12]; however, the algorithms herein are explicitly designed to operate within a multi-agent surveillance framework and introduce additional logic parameters to evoke a set of desirable geometric and temporal properties. Our proposed scheme has the following advantages: First, our framework considers a modified cost function that uses subgraph distances to maintain connectivity of intermediate coverage regions, ensuring that agents can visit their entire assigned region without entering another agent’s territory. Second, timing parameters are manipulated to provide inherent collision (redundant sensing) avoidance when the scheme is paired with low-level motion planners (Fig. 2). Third, our algorithms explicitly manipulate local likelihood functions maintained by the agents to guarantee that each has support within an allowable region, promoting seamless and modular pairing with any trajectory planner that uses the support of the event likelihood to govern agent routes, e.g., [10]. The framework has these features while simultaneously maintaining similar convergence properties as the algorithms in [12].

For clarity and readability in what follows, we postpone all theorem proofs until the Appendix.

2 Mission Overview and Solution Approach

A team of $m$ mobile agents$^1$, each equipped with an on-board sensor, is tasked with persistently monitoring a non-trivial, planar region of interest. The primary goal of the mission is to collect sensor data about some dynamic event or characteristic, e.g., an intruder. Collected data is periodically uploaded to the cloud. Agents must move within the environment to obtain complete sensory information. Ideally, agent motion should be coordinated so that:

1. the sensing workload is balanced across agents,
2. no subregion goes unobserved indefinitely,
3. agents never collide (have sensor overlap), and
4. the search is biased toward regions of greater interest.

To achieve these goals, we propose a decomposition-based approach in which each agent’s motion is restricted to lie within a dynamically assigned coverage region. The partitioning component (operating on the cloud), defines these coverage regions and provides high-level restrictions on agent motion through the manipulation of surveillance parameters, while the trajectory planning component (operating on-board each agent) incrementally constructs agent motion paths. We assume only asynchronous, cloud-agent data transfer, i.e., agents sporadically exchange data with the cloud, where inter-exchange times are not specified a priori, but are subject to an upper bound.

Broadly, our framework operates as follows (Fig. 2). Initial coverage variables are communicated to the agents prior

---

$^1$Each agent is uniquely paired with a coverage region, so the quantity $m$ represents both the number of agents and the number of regions.
3 Problem Setup

The cloud, as well as each agent, has its own local processor. “Global” information is stored on the cloud, while each agent only stores information pertinent to itself.

**Convention 1.** Subscripts i, j, or ℓ denote an entity or set element relevant to agent i, j, or ℓ, resp. The superscript ‘A’ denotes an entity that is stored by the agent’s local processor.

A storage summary is shown in Table 1. We expand on these, and define other relevant mathematical constructs here.

3.1 Agent Dynamics

Each agent (sensor) i is a point mass that moves with maximum possible speed \( s_i > 0 \). Define \( s := \{s_i\}_{i=1}^m \). Notice that this setup allows heterogeneity with respect to speed, i.e., agents can travel at different speeds.

3.2 Communication Protocol

Each agent periodically exchanges data with the cloud. Assume the following:

1. each agent can identify itself to the cloud, and transmit/receive data,
2. there is a lower bound \( \Delta > 0 \) on the time between any two successive exchanges involving the cloud, and
3. there is an upper bound \( \overline{\Delta} > 0 \) on the time between any single agent’s successive exchanges with the cloud.

Assume that agent-cloud exchanges occur instantaneously, and notice that condition 2 implies no two exchanges involving any agents occur simultaneously\(^2\). Since computation time is typically small in comparison to inter-exchange times and exchanges occur sporadically, these assumptions are without significant loss of generality.

3.3 Environment

Consider a bounded surveillance environment as a finite grid of disjoint, non-empty, simply-connected subregions. We represent the grid as a weighted graph \( G(Q) := (Q,E) \), where \( Q \) is the set of vertices (each representative of a unique grid element), and \( E \) is the edge set comprised of undirected, weighted edges \( \{k_1,k_2\} \) spanning vertices representing adjacent \(^3\) grid elements. Let the weight associated to \( \{k_1,k_2\} \) be some finite upper bound on the minimum travel distance between any point in the grid element associated \( k_1 \) to any point in the grid element associated to \( k_2 \) along a path that does not leave the union of the two elements. Locations of environmental obstacles and prohibited areas are known and are not included in the graphical representation \( G(Q) \).

Consider \( Q' \subseteq Q \). A vertex \( k_1 \in Q \) is adjacent to \( Q' \) if \( k_1 \notin Q' \) and there exists \( \{k_1,k_2\} \in E \) with \( k_2 \in Q' \). Define \( G(Q') := (Q',E') \) as the subgraph of \( G(Q) \) induced by the vertex set \( Q' \). A path on \( G(Q') \) between \( k_1,k_2 \in Q' \) is a sequence \( (k_1,k_2,\ldots,k_n) \), where \( k_1,k_2,\ldots,k_n \in Q' \) and \( \{k_r,k_{r+1}\} \in E' \) for \( r \in \{1,\ldots,n-1\} \). We say \( Q' \) is connected if a path exists in \( G(Q') \) between any \( k_1,k_2 \in Q' \). Let \( d_{Q'} : Q' \times Q' \to \mathbb{R}_{\geq 0} \cup \{\infty\} \) be the standard distance on \( G(Q') \), i.e., the length of a shortest weighted path in \( G(Q') \) (if none exists, \( d_{Q'} \) takes value \( \infty \)). Notice that \( d_{Q'}(k_1,k_2) \leq d_{Q'}(k_1,k_2) \) for any \( k_1,k_2 \in Q' \). With slight abuse of notation, we also let \( d_{Q'} \) denote the map \( d_{Q'} : Q' \times 2^Q \to \mathbb{R}_{\geq 0} \cup \{\infty\} \), where \( d_{Q'}(k,Q'' \subseteq Q) \) is the length of a shortest weighted path in \( G(Q'') \) between \( k \) and any vertex in \( Q'' \).

3.4 Coverage Regions

An \( m \)-covering of \( Q \) is a family \( P := \{P_i \subseteq Q\}_{i=1}^m \) satisfying (i) \( \bigcup_{i=1}^m P_i = Q \), and (ii) \( P_i \neq \emptyset \) for all \( i \). Define \( \text{Cov}_m(Q) \) as the set of all possible \( m \)-coverings of \( Q \). An \( m \)-partition of \( Q \) is an \( m \)-covering that also satisfies (iii) \( P_i \cap P_j = \emptyset \) for all \( i \neq j \). An \( m \)-covering or \( m \)-partition \( P \) is connected if each \( P_i \) is connected. In what follows, the cloud maintains an \( m \)-covering \( P \) of \( Q \), and surveillance responsibilities are assigned by pairing each agent \( i \) with \( P_i \in P \) (called agent \( i \)'s coverage region). Each agent maintains a copy \( P_i^A \) of \( P_i \). The cloud also stores a set \( c := \{c_i \in Q\}_{i=1}^n \) (\( c_i \) is the generator of \( P_i \)), and each agent \( i \) maintains a copy \( c_i^A \) of \( c_i \).

3.5 Identifiers, Timers, and Auxiliary Variables

The proposed algorithms introduce logic and timing variables to ensure an effective overall framework. To each \( k \in Q \), assign an identifier \( \text{ID}_k \in \{1,\ldots,m\} \). Define \( I\mathcal{D} := \{\text{ID}_k\}_{k \in Q} \), and let \( P_i^{ID} := \{P_{i\text{ID}}\}_{i=1}^m \), where \( P_i^{ID} := \{k \in Q \mid \text{ID}_k = i\} \).

\(^2\)Mathematically, the bound \( \Delta \) also prevents zeno behavior.

\(^3\)Travel between the elements without leaving their union is possible.
ID$_k = i$}. Notice $P_i^{tD}$ is an $m$-partition of $Q$. For each agent $i$, define a timer $T_i$ having dynamics $T_i = -1$ if $T_i \neq 0$, and $T_i = 0$ otherwise. Define $T := \{T_i\}_{i=1}^m$. Each agent $i$ maintains a local timing variable $\tau_i$. Even though $\tau_i$ plays a similar role to $T_i$, note that $\tau_i$ is constant unless explicitly updated, while $T_i$ has autonomous dynamics. Next, the cloud maintains a set $\omega := \{\omega_i\}_{i=1}^m$, where $\omega_i$ is the time of agent $i$’s most recent exchange with the cloud. Each agent maintains a copy $\omega_i^A$ of $\omega_i$. Finally, each agent stores a subset $P_i^{A,pld} \subseteq P_i^A$ which collects vertices that have recently been added to $P_i^A$.

### 3.6 Likelihood Functions

The likelihood of a relevant event occurring within any subset of the surveillance region is maintained on the cloud in the form of a time-varying probability mass function\(^a\) $\Phi : Q \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{> 0}$. For simplicity, assume that, at any $t$, the instantaneous support, $\text{supp}(\Phi(\cdot,t))$, equals $Q$.

Define each agent’s local likelihood $\Phi_i^A : Q \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{> 0}$ as the function that, loosely, represents the agent’s local belief regarding events. Specifically, define

$$
\Phi_i^A(k,t) = \begin{cases} 
\Phi(k,t), & \text{if } k \in P_i^A \text{ and } (t - \omega_i^A \geq \tau_i^A \text{ or } k \notin P_i^{A,pld}), \\
0, & \text{otherwise.}
\end{cases}
$$

(1)

The conditions defining $\Phi_i^A$ are understood as follows: at some time $t$, an element $k \in Q$ only belongs to $\text{supp}(\Phi_i^A(\cdot,t))$ if (i) $k \in P_i^A$, and (ii) sufficient time has passed since $k$ was first added to $P_i^A$, as determined by the parameters $\tau_i^A$, $\omega_i^A$, and $P_i^{A,pld}$. In general, each $\Phi_i^A$ will be different\(^b\) from $\Phi$.

**Remark 1 (Global Data).** If global knowledge of $\Phi$ is not available instantaneously to agent $i$, $\Phi_i^A$ can alternatively be defined by replacing $\Phi(k,t)$ in (1) by $\Phi(k, \omega_i^A)$. All subsequent theorems hold under this alternative definition.

**Remark 2 (Data Storage).** The cost of storing a graph as an adjacency list is $O(|Q| + |E|)$. The generator set $c$, each element of $P$, and the identifier set $I \mathcal{D}$ are stored as integral vectors. The timer set $T$ and the set $\omega$ are are stored as real vectors, while $\Phi$ is stored as a time-varying real vector. Thus, the cost of storage on the cloud is $O(m|Q| + |E|)$. Similarly, each agent’s local storage cost is $O(|Q| + |E|)$.

### 4 Dynamic Coverage Update Scheme

Adopt following convention for the remaining analysis.

**Convention 2.** Suppose that:

1. $\min \emptyset := \max \emptyset := 0$, and
2. given a specific time instant, superscripts ‘-’ or ‘+’ refer to a value before and after the instant in question, resp.

\(^a\) $\Phi(k,t)$ is a probability mass function for any $t$.

\(^b\) $\Phi_i^A$ is not normalized and thus may not be a time-varying probability mass function in a strict sense.

###表1. Storage Summary

<table>
<thead>
<tr>
<th>Stored on the Cloud</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G(Q)$</td>
<td>Graphical representation of the environment</td>
</tr>
<tr>
<td>$P_i^A$</td>
<td>Coverage region ($P_i^A \subseteq Q$)</td>
</tr>
<tr>
<td>$c_i^A$</td>
<td>Coverage region generator ($c_i^A \in Q$)</td>
</tr>
<tr>
<td>$P_i^{A,pld}$</td>
<td>Set of “recently added” vertices ($P_i^{A,pld} \subseteq P_i^A$)</td>
</tr>
<tr>
<td>$\tau_i^A$</td>
<td>Local timing parameter ($\tau_i^A \in \mathbb{R}$)</td>
</tr>
<tr>
<td>$\omega_i^A$</td>
<td>Most recent exchange time ($\omega_i^A \in \mathbb{R}_{\geq 0}$)</td>
</tr>
<tr>
<td>$\Phi_i^A$</td>
<td>Local likelihood ($\Phi_i^A : Q \times \mathbb{R}<em>{\geq 0} \to \mathbb{R}</em>{&gt; 0}$)</td>
</tr>
</tbody>
</table>

### 4.1 Additive Set

We start with a definition.

**Definition 1 (Additive Set).** Given $k \in P_i^{tD}$, the additive set $P_i^{add}(k) \subseteq Q$ is the largest connected subset satisfying:

1. $P_i^{tD} \subseteq P_i^{add}(k)$, and
2. for any $h \in P_i^{add}(k) \cap P_j$, where $j \neq i$:
   (a) $T_j = 0$, and
   (b) $\frac{1}{h}d_{i,pld}(h,k) < \frac{1}{h}d_p(h, c_j)$.

The following characterizes well-posedness of Definition 1.

**Proposition 1 (Well-Posedness).** If $P_i^{tD}$ is connected and disjoint from $\bigcup_{j \neq i} P_j$, then $P_i^{add}(k)$ exists and is unique for any $k \in P_i^{tD}$.

**Proof.** With the specified conditions, $P_i^{tD}$ is connected and satisfies 1 and 2 in Definition 1; $P_i^{add}(k)$ is the unique, maximally connected superset of $P_i^{tD}$ satisfying 1 and 2.

Under the conditions of Proposition 1, if $h \in P_i^{add}(k)$, then $\max\{T_j \mid j \neq i, h \in P_j\} = 0$ and there is a path from $k$ to $h$ in $G(P_i^{add}(k))$ that is shorter than the optimal path spanning $c_j$ and $h$ within $G(P_j)$, for any $j \neq i$ with $h \in P_j$. 

---

\(a\) $\Phi(k,t)$ is a probability mass function for any $t$.

\(b\) $\Phi_i^A$ is not normalized and thus may not be a time-varying probability mass function in a strict sense.
4.2 Cloud-Supported Coverage Update

Define a cost function $\mathcal{H} : Q^m \times \text{Cov}_m(Q) \times \mathbb{R}_{\geq 0} \to \mathbb{R}$ by

$$\mathcal{H}(c,P,t) = \sum_{k \in Q} \min_{\Theta(k,c)} \left\{ \frac{1}{s_i}d_P(k,c) \mid k \in P_i \right\} \Phi(k,t).$$

If (i) each agent is solely responsible for events within its coverage region, and (ii) events occur proportionally to $\Phi$, then $\mathcal{H}(c,P,t)$ is understood as the expected time required for an agent to reach a randomly occurring event from its region generator at time $t$; related functions are studied in [12,26,30]. Algorithm 1 defines the operations performed on the cloud when agent $i$ makes contact at time $t_0$. Here, the result of the update. Additional remarks to aid in reader understanding are given by the comments within the algorithms (italicized text preceded by a “%” character).

Algorithm 1: Cloud-Supported Coverage Update

| Data: $t_0$, $P$, $\phi$, $\omega$, $\bar{\Delta}$, $\Delta_H$, $T$, $I\mathcal{D}$, $s$ | Result: $P$, $c$, $P^A$, $c^A$, $P^{A,pd}$, $T$, $\tau^A$, $\Phi^A$, $\omega$, $\omega^A$, $I\mathcal{D}$ |
| begin |
| % Initialize, remove regions others have claimed |
| 1 Initialize $P^* = P_{\text{test}} = P$, $c^* = c_{\text{test}} = c$ |
| 2 Set $P^*_i = P_i(\mathcal{D})$ |
| % If timer is non-zero and no regions have been claimed since last update, perform trivial update |
| if $T_j > 0$ and $P^*_i = P_i$ then |
| 3 Set $\tau^A_j = \tau^A_j - t_0 + \omega_j$ and $\omega^A_j = \omega_i = t_0$ |
| else |
| % Iterate through generator locations to find a cost-minimizing configuration |
| for $k \in P^*_i \mathcal{D}$ do |
| 4 % Update timer and variables |
| 5 if $\mathcal{H}(c_{\text{test}}, P_{\text{test}}, t_0) < \mathcal{H}(c^*, P^*, t_0)$ then |
| 6 Set $P^* = P_{\text{test}}$, $c^* = c_{\text{test}}$ |
| % Call Algorithm 2 and obtain output $\Phi^A$, $\omega$, $T$, $\tau^A$ |
| 7 Call Alg. 2 and obtain output $\Phi^A_i$, $\omega_i$, $T_i$, $\tau^A_i$ |
| 8 Set $P_i = P^A_i$, $c_i = c^A$, $\omega^A_i = \omega_i$ |
| 9 for $k \in P_i$ do Set $\mathcal{D}_i = i$ |
| return $P$, $c$, $P^A$, $c^A$, $P^{A,pd}$, $T$, $\tau^A$, $\Phi^A$, $\omega$, $\omega^A$, $I\mathcal{D}$ |

Consider the following initialization assumptions.

**Assumption 1 (Initialization).** The following properties are satisfied when $t = 0$:

1. $P$ is a connected $m$-partition of $Q$.
2. $P = P^{\mathcal{D}}$, and
3. for all $i \in \{1, \ldots, m\}$,
   - $P_i = P^A_i$,
   - $c_i = c^A_i \in P^A_i$,
   - $P^{A,pd}_i = \emptyset$,
   - $T_i = \omega_i = \omega^A_i = 0$, and
   - $\tau^A_i = -\Delta_H$.

Notice 1 and 3b together imply that $c_i \neq c_j$ for any $j \neq i$. Our first result guarantees well-posedness of Algorithm 1.

**Theorem 1 (Well-Posedness).** Under Assumption 1, a scheme in which, during each agent-cloud exchange, the cloud executes Algorithm 1 to update relevant global and local variables is well-posed. That is, operations required by Algorithm 1 are well-posed at the time of execution.

Algorithm 1 does not ensure that coverage regions (elements of $P$) remain disjoint. It does, however, guarantee that the $m$-covering $P$, the local coverage regions $P^A_i := \{P^A_i\}_i=1$.
and the local likelihoods \( \{ \Phi_i^A \}_{i=1}^m \) retain properties that are consistent with a decomposition-based scheme. Namely, the coverings \( P \) and \( P^* \) maintain connectivity, and each \( \Phi_i^A \) has support that is disjoint from that of all other local likelihoods, yet evolves to provide reasonable global coverage. Further, Algorithm 2 ensures that agents can “safely” vacate areas that are reassigned before newly assigned agents enter. We expand upon these ideas in the following two subsections.

### 4.3 Set Properties

The next result formalizes key set properties.

**Theorem 2 (Set Properties).** Suppose Assumption 1 holds, and that, during each agent-cloud exchange, the cloud executes Algorithm 1 to update relevant global and local variables. Then, the following hold at any time \( t \geq 0 \):

1. \( P^{(0)} \) is a connected \( m \)-partition of \( Q \).
2. \( P \) is a connected \( m \)-covering of \( Q \).
3. \( c_i \in P_i \) and \( c_i \neq c_j \) for any \( i \neq j \).
4. \( \supp(\Phi_i^A(\cdot, t)) \subseteq P_i \) for any \( i \), and
5. \( \bigcap_{i=1}^m \supp(\Phi_i^A(\cdot, t)) = \emptyset \).

When the cloud makes additions to an agent’s coverage region, newly added vertices are not immediately included in the instantaneous support of the agent’s local likelihood. If agent movement is restricted to lie within this support, the delay temporarily prohibits exploration of newly added regions, allowing time for other agents to vacate. Conversely, when regions are removed from an agent’s coverage region, Algorithm 1 ensures a “safe” path, i.e., a path with no collision risk, exists and persists long enough for the agent to vacate. Let \( \overline{d} := \max \frac{1}{s_i} \sum_{(k_1, k_2) \in E} d_q(k_1, k_2) \), and define agent \( i \)'s prohibited region, \( \text{Proh}_i(t) \), as the set of vertices not belonging to the support of \( \Phi_i^A(\cdot, t) \), i.e., \( \text{Proh}_i(t) := \{ k \in Q \mid k \notin \supp(\Phi_i^A(\cdot, t)) \} \). We formalize this discussion here.

**Theorem 3 (Coverage Quality).** Suppose Assumption 1 holds, and that, during each agent-cloud exchange, the cloud updates relevant global and local coverage variables via Algorithm 1. Then, for any \( k \in Q \) and any \( t \geq 0 \):

1. \( k \) belongs to at least one agent’s coverage region \( P_i \),
2. if \( k \in \text{Proh}_i(t) \cap P_i \) for some \( i \), then there exists \( t_0 \) satisfying \( t < t_0 < t + \Delta H \) such that, for all \( \tilde{t} \in [0, t_0 + \Delta H] \), the vertex \( k \) belongs to the set \( P_i \setminus \text{Proh}_i(\tilde{t}) \), and
3. if \( k \) is removed from \( P_i \) at time \( t \), then, for all \( \tilde{t} \in (t, t + \frac{1}{s_i}d_{P^*_i}(k, P_i^{(0)})) \), we have

\[
(a) \quad P_i^{(0)} \subseteq P_i, \quad \text{and} \\
(b) \quad \text{there exists a length-minimizing path on } G(P_i^*) \text{ from } k \text{ into } P_i^{(0)}, \text{ and all of the vertices along any such path (except the terminal vertex) belong to the set } \text{Proh}_{ID_k^*}(\tilde{t}) \setminus \bigcup_{j \notin ID_k^*} P_j.
\]

4.4 Convergence Properties

Our proposed strategy differs from [12] due to logic, i.e., timing parameters, etc., that ensures effective operation within a decomposition-based framework. Note also that \( \mathcal{H} \) differs from previous partitioning cost functions in [12, 26, 30], since it uses subgraph, rather than global graph, distances. As such, convergence properties of the algorithms herein do not follow readily from existing results. Consider the following definition.

**Definition 2 (Pareto Optimality).** The pair \((c, P)\) is Pareto optimal at time \( t \) if

1. \( H(c, P, t) \leq H(\bar{c}, P, t) \) for any \( \bar{c} \in Q^m \), and
2. \( H(c, P, t) \leq H(c, P, t) \) for any \( \bar{c} \in \text{Covm}(Q) \).

When \( \Phi \) is time-invariant (and Assumption 1 holds), Algorithm 1 produces finite-time convergence of coverage regions and generators to a Pareto optimal pair. The limiting coverage regions are “optimal” in that they balance the sensing load in a way that directly considers the event likelihood. Further, the operation only requires sporadic and unplanned agent-cloud exchanges. We formalize this result here.

**Theorem 4 (Convergence).** Suppose Assumption 1 holds and that, during each agent-cloud exchange, the cloud updates relevant global and local coverage variables via Algorithm 1. If \( \Phi \) is time-invariant, i.e., \( \Phi(\cdot, t_1) = \Phi(\cdot, t_2) \) for all \( t_1, t_2 \), then the \( m \)-covering \( P \) and the generators \( c \) converge in finite time to an \( m \)-partition \( P^* \) and a set \( c^* \), resp. The pair \((c^*, P^*)\) is Pareto optimal at any time following convergence.

**Remark 5 (Weighted Voronoi Partitions).** It can be shown that Pareto optimality of \((c^*, P^*)\) in Theorem 4 implies that, following convergence, \( P^* \) is a multiplicatively-weighted Voronoi partition (generated by \( c^* \), weighted by \( s \), subject to density \( \Phi(\cdot, t) \)) by standard definitions (e.g., [12]). If the centroid set of each \( P_i \) is defined as \( \arg \min_{h \in P_i} \sum_{k \in P_i} d_P(k, h) \Phi(k, t) \), then \( P^* \) is also centroidal.
Our coverage scheme balances the sensing load in that it updates coverage responsibilities in a way that locally minimizes the expected time required for an appropriate agent to travel from its region generator to a randomly occurring event within the environment. In essence, this serves to avoid unreasonable configurations, e.g., configurations where one agent is assigned responsibility of all important areas and remaining agents are only given non-important regions. Further, the update rules consider agent speeds, so faster agents will generally be assigned larger (weighted by the likelihood) coverage regions than slower agents. Similar strategies are employed in traditional load-balancing algorithms that are based on Voronoi partitions and operate over environments with stochastic event likelihoods, e.g., [12, 26, 30].

Theorem 4 and Remark 5 provide a rigorous characterization of the type of load balancing provided in the static likelihood case. A few comments are in order. First, Pareto optimal pairs (and multiplicatively-weighted, centroidal Voronoi partitions) are non-unique. Theorem 4 only guarantees that one possible Pareto optimal pair will be found in the static likelihood case, and does not exclude the existence of a lower-cost configuration. Second, the coverage configurations produced by our algorithms may not be equitable, i.e., the probability of events in each coverage region may vary across agents, even in the static case. The development of a strategy that produces equitable partitions within a similar cloud-supported architecture is an open problem and an interesting area of future research.

5 Decomposition-Based Surveillance.

This section pairs the proposed partitioning framework with a generic, single-vehicle trajectory planner, forming the complete, multi-agent surveillance framework.

5.1 Complete Routing Algorithm

By Theorem 2, the support of each \( \Phi_i \) (i) lies entirely within the coverage region \( P_i \), and (ii) is disjoint from the support of other local likelihoods. By Theorem 3, (i) any vertex can only go uncovered over bounded intervals, and (ii) the parameter \( \Delta_H \) is a lower bound on the time that a recently uncovered vertex must remain covered before it can become uncovered again. These results suggest that an intelligent routing scheme that carefully restricts motion according to the instantaneous support of the local likelihood functions could achieve adequate coverage while maintaining collision avoidance. This motivates the following assumption.

Assumption 2 (Agent Motion). Each agent \( i \) has knowledge of its position at any time \( t \), and its on-board trajectory planner operates under the following guidelines:

1. generated trajectories obey agent motion constraints,
2. trajectories are constructed incrementally and can be altered in real-time, and
3. the agent is never directed to enter regions associated with \( \text{Proh}_i(t) \).

Each agent precisely traverses generated trajectories.

Note that condition 2 of Assumption 2 implies that the agent is never directed to leave regions associated with \( P_i \).

Algorithm 3 presents the local protocol for Agent \( i \). Here, the on-board trajectory planner is used to continually update agent trajectories as the mission progresses (line 1). As such, the low-level characteristics of each individual agent’s motion (i.e., the relation between the underlying likelihood function, the coverage configuration, and the resultant trajectory) depends on the particular planner employed.

5.2 Collision Avoidance

Although Assumption 2 locally prevents agents from entering prohibited regions, dynamic coverage updates can still place an agent within its prohibited region if the vertex corresponding to its location is abruptly removed during an update. If this happens, Algorithm 3 constructs a route from the agent’s location back into a region where there is no collision risk. With mild assumptions, this construction (i) is well-defined, and (ii) does not present a transient collision risk. We formalize this result here.

Theorem 5 (Collision Avoidance). Suppose Assumptions 1 and 2 hold, and that each agent’s initial position lies within its initial coverage region \( P \). If each agent’s motion is locally governed according to Algorithm 3, where the update in line 4 is calculated by the cloud via Algorithm 1, then no two agents will ever collide.

Remark 6 (Agent Dynamics). We assume point mass dynamics for simplicity. However, all theorems herein also apply under alternative models, e.g., non-holonomic dynamics, provided that the surveillance environment is discretized so that (i) travel between adjacent grid elements is possible without leaving their union, (ii) agents can traverse the aforementioned paths at maximum speed, and (iii) edge weights accurately upper bound travel between adjacent regions. Typically, these conditions can be met by choosing
discretization cells that are sufficiently large. For example, under a Dubins vehicle model, choosing square cells whose edge lengths are at least twice the minimum turning radius of any of the vehicles is sufficient. If these conditions are not met, Theorems 1, 2, 3, and 4 still apply, though Theorem 5 is no longer guaranteed since more sophisticated logic would be needed to ensure agent trajectories remain within the allowable regions. The development of an algorithmic extension that would guarantee collision avoidance for general non-holonomic vehicles is not straightforward and is left as a topic of future research.

In practice, however, we note that even if Theorem 5 is not satisfied in a strict sense, implementation of the algorithms herein within a decomposition-based scheme will usually still provide a significantly reduced collision or redundant sensing risk, provided that vehicles remain close to the allowable regions specified by the updates.

6 Numerical Examples

This section presents numerical examples to illustrate the proposed framework’s utility. In all examples, updates are performed on the cloud via Algorithm 1 during each agent-cloud exchange, and each agent’s local processor runs the motion protocol in Algorithm 3. For incremental trajectory construction (Algorithm 3, line 1), all examples use a modified Spectral Multiscale Coverage (SMC) scheme [10], which creates trajectories to mimic ergodic dynamics while also locally constraining motion to lie outside of prohibited regions. Notice this planner satisfies Assumption 2. Initial region generators were selected randomly (enforcing non-coincidence), and each agent was initially placed at its region generator. The initial covering \( P \) was created by calculating a weighted Voronoi partition, and remaining initial parameters were chosen to satisfy Assumption 1. It is assumed that relevant initial variables are uploaded to the agents’ local servers prior to initial deployment, i.e., each agent has full knowledge of relevant initial information at time 0. For each simulation, randomly chosen agents sporadically exchanged data with the cloud. Agent-cloud exchange times were randomly chosen, subject to a maximum inter-exchange time \( \Delta \).

6.1 Time-Invariant Likelihood

Consider a 4 agent mission, executed over a 100 x 100 surveillance region that is subject to a time-invariant, Gaussian likelihood centered near the bottom left corner. The region is divided into 400, 5 x 5 subregions. Regions are considered adjacent if they share a horizontal or vertical edge. Here, each agent had a maximum speed of 1 unit distance per unit time, and \( \Delta = 10 \) time units. Fig. 3 shows the evolution of the coverage regions for an example simulation run. Agent trajectories are shown by the colored lines. Note that Fig. 3 only shows each agent’s active coverage region, i.e., \( P_i \setminus \text{Proh}_i(t) \). The family of active coverage regions does not generally form an \( m \)-covering of \( Q \); however, elements of this family are connected and never intersect as a result of inherent collision avoidance properties.

The left plot in Fig. 4 depicts the maximum amount of time that any grid square went uncovered, i.e. the grid square did not belong to any agent’s active covering, during each of 50 simulation runs. Here, the maximum amount of time that any region went uncovered was 186 units, though the maximum for most trials was less than 75 units. This is well below the loose bound \( \Delta + d = 770 \) predicted by Theorem 3 (see Remark 4). Note that this metric does not capture the time between the agent’s actual visits to the grid-square, only the length of intervals on which no agent was allowed to visit the square. The time between visits is governed by the particular choice of trajectory planner and the parameter \( \Delta_H \).

The right plot in Fig. 4 shows the mean values of the cost function \( \mathcal{H} \), calculated over the same 50 simulations runs. Here, error bars represent the range of cost values at select points. The variance between runs is due to the stochastic nature of the agent-cloud exchange patterns. Notice the cost is non-increasing over time, eventually settling as the coverage regions/generators reach their limiting configuration, e.g., see Fig. 3. These configurations are each Pareto optimal and form a multiplicatively-weighted Voronoi partition (Remark 5). The resultant coverage assignments provide load-balancing that takes into account the event likelihood. If the low-level trajectory planner biases trajectories according to the event likelihood, this results in desirable coverage proper-
ties. Under the modified SMC planner used here, the temporal distribution of agent locations closely resembles the spatial likelihood distribution in the limit, as shown in Fig. 5.

Further, during the simulation, no two agents ever occupied the same space due to the careful parameter manipulations employed by Algorithm 1. Fig. 6 illustrates the logic governing these manipulations through a simplistic example: During the first update, the blue agent acquires some of the red agent’s coverage region. Rather than immediately adding these regions to its active covering, the blue region waits until sufficient time has passed to guarantee that the red agent has updated and moved out of the reassigned regions. Under Algorithm 3, once the red agent communicates with the cloud, it immediately vacates the reassigned regions, after which the blue agent can add the region to its active covering. This procedure guarantees that no two agents will never have overlapping active coverings and thus never collide (Theorem 5). This same logic results in inherent collision prevention over more complex scenarios.

We can also compare the coverage regions produced by Algorithm 1 to those produced by the partitioning algorithm in [12]. The two algorithms were simulated in parallel, performing updates with the same randomly chosen agent-cloud exchange orderings across the two conditions. The left and the right plots in Fig. 7 show the mean coverage cost over 50 simulation runs, calculated using $\mathcal{H}_{\min}$ (defined in [12], Section II-C) and $\mathcal{H}$ (Section 4.2), respectively (portions of the curves extending above the axes indicate an infinite value). The function $\mathcal{H}_{\min}$ is defined nearly identically to $\mathcal{H}$, but uses global graph distances, rather than subgraph distances. It is clear that the evolution produced by the algorithm in [12] converges to a final configuration slightly faster than that produced by Algorithm 1 whenever costs are quantified using $\mathcal{H}_{\min}$. However, when costs are calculated using $\mathcal{H}$, the algorithms in [12] produced intermediate configurations with infinite cost, indicating disconnected regions, while Algorithm 1 maintained connectivity. In contrast to [12], our surveillance framework allows for complete coverage without requiring the agents to leave their assigned regions, allowing it to operate more effectively within a multi-agent surveillance scheme.

### 6.2 Time-Varying Likelihood

We now illustrate how the proposed coverage framework reacts to changes in the underlying likelihood. Specifically, we study a particular type of time-varying likelihood in which the spatial distribution only changes at discrete time-points, i.e., $\Phi(k, \cdot)$ is piece-wise constant for any $k \in Q$. This type of scenario is common in realistic missions, e.g., when the cloud’s estimate of the global likelihood is only re-formulated if some agent’s sensor data indicates a drastic change in the underlying landscape. For this purpose, we adopt identical parameters as in the first example, with the exception of the likelihood $\Phi$, whose spatial distribution abruptly switches at select time-points. If the switches are sufficiently spaced in comparison to the rate of convergence, then the coverage regions dynamically adjust to an optimal configuration that is reflective of the current state. For example, Fig. 9 shows the coverage region evolution after the underlying likelihood undergoes a single switch between the likelihoods in Fig. 8 at time $t = 2000$. In contrast, if the likelihood changes faster than the rate of convergence, coverage regions are constantly in a transient state. Despite this,
the proposed framework still provides some degree of load-balancing. To illustrate, the left plot in Fig. 10 shows the value of $\mathcal{H}$ during a simulation in which the underlying likelihood switches at 12 randomly chosen time-points over a 1000 unit horizon. Each switch re-defined the spatial likelihood as a Gaussian distribution centered at a randomly selected location. Notice that the cost monotonically decreases a quick shift away from high-cost configurations. Upon reassignment, the timers remain in these sets until another reassignment. Thus, state-
tional antennae for wireless transmission. Extensions to the proposed algorithms to incorporate explicit area constraints on coverage regions, as well as more general vehicle dynamics, should also be explored. Other areas of future re-search include the combination of peer-to-peer and cloud-based communication, performance comparisons between specific trajectory planners when used within our framework, e.g., those involving ergodic Markov chains, and further theoretical performance characterization.

Appendix: Proofs

Proposition 2 (Sets). Suppose Assumption 1 holds, and that, at the time of each exchange occurring prior to a fixed time $t \geq 0$, required algorithmic constructions are well-posed so that the cloud can perform updates via Algorithm 1. Then, for any $k \in Q$ and any time $t \leq \tilde{t}$:

1. $k \in P_{ID_k}$
2. $k$ belongs to at most 2 elements of $P$
3. if $T_{ID_k} = 0$, then $k \notin P_i$ for any $i \neq ID_k$, and
4. if $k \in P_i$, $j \neq ID_k$, then $P_j \cap P_{ID_k} = \emptyset$ for $j \notin \{j, ID_k\}$

Proof. Fix $\tilde{t} \geq 0$, $k \in Q$. When $t = 0$, $P = P^{ID}$ is an m-partition of $Q$, implying the proposition. Since $k$ is not removed from $P_{ID_k}$ or added to any $P_i$ with $i \neq ID_k$ until its first reassignment, i.e., when $ID_k$ is changed, the proposition holds for all $t$ prior to the first reassignment. Suppose the proposition holds for all $t$ prior to the $p$th reassignment, which occurs at time $t_0$. Suppose $ID_k = j$, $ID_k^+ = i \neq j$. Alg. 1 defines $P_k^{ID, +} = P_i - P_k^- = P_k^+$. Thus, $k \in P_{ID_k}^+ = P_k^+$ and remains in these sets until another reassignment. Thus, statement 1 holds for all $t$ prior to the $p + 1$st reassignment. Now note that, by Alg. 2, reassignment cannot occur at $t_0$ unless $T_{ID_k} = 0$. By inductive assumption, statement 3 of the proposition holds when $t = t_0^-$, implying $k \notin P_i^-$ for any $i \neq j$. Upon reassignment, the timers $T_j, T_i$ are modified such that $T_j^+, T_i^+ > \omega_j^- + \bar{\Delta} - t_0$. Since (i) $ID_k$ cannot change when $T_j^+ > 0$, and (ii) agent $j$ exchanges data with the cloud and removes $k$ from $P_j$ prior to time $\omega_j^+ + \bar{\Delta}$, we deduce that $k$ solely belongs to $P_j, P_i$ until the $p + 1$st reassignment. Further, for any $t \geq t_0^+$ at which $T_0 = 0$ and the $p + 1$st reas-

7 Conclusion

This work develops a cloud-supported, decomposition-based, coverage control framework for multi-agent surveillance. In particular, a dynamic partitioning strategy balances the surveillance load across available agents, requiring only sporadic and unplanned agent-cloud exchanges. The partitioning update algorithm also manages high-level logic parameters to guarantee that the resulting coverage assignments have geometric and temporal properties that are amenable for combination with generic single vehicle trajectory planners. In certain cases, the proposed algorithms produce a Pareto optimal configuration, while ensuring collision avoidance throughout.

Future work should further relax communication assumptions to reflect additional limitations, e.g., use of directional antennae for wireless transmission. Extensions to the proposed algorithms to incorporate explicit area constraints on coverage regions, as well as more general vehicle dynamics, should also be explored. Other areas of future research include the combination of peer-to-peer and cloud-based communication, performance comparisons between specific trajectory planners when used within our framework, e.g., those involving ergodic Markov chains, and further theoretical performance characterization.
assumption has not yet occurred, $k \in P_t$ exclusively (addition to other sets in $P$ without reassignment is impossible). We deduce statements 2 and 3 for any $t$ prior to the $p + 1$st reassignment. Finally, considering Alg. 2, it is straightforward to show that $T_j^\ominus = 0$ implies $P_j^\ominus = P_j^{\ominus \delta \omega_1\omega_2\omega_3\omega_4\omega_5\omega_6\omega_7}$ ($T_j = 0$ only if the most recent exchange that manipulated elements of $P_j^{\ominus \delta}$ involved agent $j$, after which $P_j = P_j^{\ominus \delta \omega_1\omega_2\omega_3\omega_4\omega_5\omega_6\omega_7}$. Further, (i) no agent claims vertices from $P_j^\ominus$ unless $T_j = 0$, and (ii) no vertex is added to a coverage region without reassignment. As such, $P_j \cap P_i^{\ominus \delta} = \emptyset$ for any $i \neq j$. This implies that Prop. 2 holds at any time prior to the $p + 1$st reassignment. Assume the $p$th exchange occurs at time $t_0$ and involves agent $j$. Recall that $P_i^{\ominus \delta \omega_1\omega_2\omega_3\omega_4\omega_5\omega_6\omega_7}$ is an $m$-partition of $Q$. To show $P_i^{\ominus \delta \omega_1\omega_2\omega_3\omega_4\omega_5\omega_6\omega_7}$ is connected, first notice $P_i^{\ominus \delta \omega_1\omega_2\omega_3\omega_4\omega_5\omega_6\omega_7}$. Since either $P_i^{\ominus \delta \omega_1\omega_2\omega_3\omega_4\omega_5\omega_6\omega_7}$ (connected by Def. 1) or $P_i^{\ominus \delta \omega_1\omega_2\omega_3\omega_4\omega_5\omega_6\omega_7}$ (connected by assumption), connectivity of $P_j^{\ominus \delta \omega_1\omega_2\omega_3\omega_4\omega_5\omega_6\omega_7}$ holds. Now consider $P_j^{\ominus \delta \omega_1\omega_2\omega_3\omega_4\omega_5\omega_6\omega_7}$. Thus, there exists $k_1 \in P_i^{\ominus \delta \omega_1\omega_2\omega_3\omega_4\omega_5\omega_6\omega_7}$ (such that $k_1 \notin P_i^{\ominus \delta \omega_1\omega_2\omega_3\omega_4\omega_5\omega_6\omega_7}$, and (ii) any optimal path in $G(P_i^{\ominus \delta \omega_1\omega_2\omega_3\omega_4\omega_5\omega_6\omega_7})$ contains some $k_2 \in P_i^{\ominus \delta \omega_1\omega_2\omega_3\omega_4\omega_5\omega_6\omega_7}$. Select one such path and vertex $k_2$. Without loss of generality, assume $|k_1, k_2| \in E$. Def. 1 implies $\frac{1}{d_{P_i^{\ominus \delta \omega_1\omega_2\omega_3\omega_4\omega_5\omega_6\omega_7}}(k_2, 1) < \frac{1}{d_{P_i^{\ominus \delta \omega_1\omega_2\omega_3\omega_4\omega_5\omega_6\omega_7}}(k_1, 1)$. Since $T_j^\ominus = 0$ and $ID_{k_1} = j$. Prop. 2 implies $\frac{1}{d_{P_i^{\ominus \delta \omega_1\omega_2\omega_3\omega_4\omega_5\omega_6\omega_7}}(k_2, 1) < \frac{1}{d_{P_i^{\ominus \delta \omega_1\omega_2\omega_3\omega_4\omega_5\omega_6\omega_7}}(k_1, 1)$. Thus, $P_i^{\ominus \delta \omega_1\omega_2\omega_3\omega_4\omega_5\omega_6\omega_7}$ is connected. Invoking Prop. 2 statement 3, the inductive assumption holds for all $t$ prior to the $p + 1$st exchange, implying well-posedness.

Proof of Theorem 2.

Statement 1: The proof of Thm. 1 implies the statement.

Statement 2: $P$ is an $m$-covering of $Q$ since $P_i^{\ominus \delta \omega_1\omega_2\omega_3\omega_4\omega_5\omega_6\omega_7}$ is an $m$-partition of $Q$, and $P_i^{\ominus \delta \omega_1\omega_2\omega_3\omega_4\omega_5\omega_6\omega_7} \subseteq P_t$ for any $i$ (Prop. 2, statement 1). The covering $P$ is connected, since $P_i = P_i^{\ominus \delta \omega_1\omega_2\omega_3\omega_4\omega_5\omega_6\omega_7}$ (connected by statement 1) immediately following any agent-cloud exchange and is unchanged in between updates.

Statement 3: It suffices to show $ID_{c_i} = i$ for any $t$, $i$: this would imply $c_i \neq c_j$ for any $i \neq j$, and $c_i \in P_t$ (Prop. 2). Since $ID_{c_i} = i$ for all $i \neq 0$, the same holds for any $t$ prior to the first agent-cloud exchange. Suppose $ID_{c_i} = i$ for all $i$ (thus $c_i \neq c_j$ for any $i \neq j$) prior to the $p$th exchange. If agent $i$ is the $p$th communicating agent, lines 2, 9 of Alg. 1 imply $ID_{c_i} = i$. Since $d_{P_j^{\ominus \delta \omega_1\omega_2\omega_3\omega_4\omega_5\omega_6\omega_7}}(c_j, c_j) = 0$ for any $j$, we have $c_j \notin P_i^{\ominus \delta \omega_1\omega_2\omega_3\omega_4\omega_5\omega_6\omega_7}$. Thus, $ID_{c_j} = j$, and induction proves the statement.

Proof of Theorem 3. Thm. 2 implies statement 1.

Statement 2: For any $t$, (i) $t_0 = 0$ when $t = 0$, and (ii) $\frac{1}{d_{Q}(k_1, k_2)} < \frac{1}{d_{Q}(k_1, k_2)} < \frac{1}{d_{Q}(k_1, k_2)} \leq \frac{1}{d_{Q}(k_1, k_2)} \leq \frac{1}{d_{Q}(k_1, k_2)}$. Thus, it is straightforward to show, for any $t$, $i$, we have the bound $T_j \leq \Delta + \Delta H + \Delta$. We show by induction that, for any $i$, $j$, the bound $\tau_{i,j}^\ominus = \tau_{i,j}^\ominus < \tau_{i,j}^\ominus \leq \Delta + \Delta H$ also holds: $T_j = 0$ and $\tau_{i,j}^\ominus = \Delta H$ when $t = 0$, and $\tau_{i,j}^\ominus = \Delta H + \Delta$, thus $T_j > 0$ when $t = t_1$, and $k$ is removed from both $P_j$ and $\supp(\Phi_i^\ominus(t), t)$, then, for all $t > t_1 + \tau_{i,j}^\ominus$ and before the $p + 1$st reassignment, $k$ belongs exclusively to $\supp(\Phi_i^\ominus(t), t)$. Using the aforementioned two bounds, we have $\tau_{i,j}^\ominus < \Delta H + \Delta$. Fix $i$ and $j \in \text{Proh}_i(t) \cap P_t$. Then, $k \in P_{i,t}^\ominus = P_{i,t}^\ominus$, $k \in P_{i,t}^\ominus \Delta H$, and $t - \omega_{\Delta H} < \tau_{i,j}^\ominus + \tau_{i,j}^\ominus$. Setting $t_0 = \omega_{\Delta H} + \tau_{i,j}^\ominus$, we have $t < t_0 \leq t + \Delta H$. Since $T_j \geq \tau_{i,j}^\ominus + \Delta H$ at time $\omega_{\Delta H} + \tau_{i,j}^\ominus$, $k$ is not reassigned during the interval $[\omega_{\Delta H} + \tau_{i,j}^\ominus + T_j] \subseteq [\omega_{\Delta H} + t_0 + \Delta H]$. Thus $k \in P_{i,t}^\ominus \text{Proh}_i(t) \cap P_t$ for the same time interval.

Statement 3: Fix $t$ and suppose $k \in P_{i,t}^\ominus \text{Proh}_i(t)$ (in this proof,
Proof. Suppose Assumption 1 holds and \( \omega_{i}^{A} \) is added to \( P_{i}^{A_{k}} \), and (iii) \( \tau_{k} \) is time-invariant, \( T_{i} \). To complete the proof, Alg. 2 implies \( P_{i}^{A_{k}} \). Since \( k \in P_{j}^{+} \), we have \( P_{i}^{A_{k}}(c_{i}^{+}) \) as constructed during the update, implying \( \frac{1}{s_{i}}d_{P_{i}^{+}}(k, c_{i}^{+}) \leq \frac{1}{s_{j}}d_{P_{i}^{+}}(k, c_{i}^{+}) \). Thus, \( k \in P_{j}^{+} \), and \( T_{j} = 0 \) and \( ID_{k} = j \). Prop. 2 implies \( \frac{1}{s_{i}}d_{P_{i}^{+}}(k, c_{i}^{+}) \leq \frac{1}{s_{j}}d_{P_{i}^{+}}(k, c_{i}^{+}) \). Thus, \( d_{P_{i}^{+}}(c_{i}^{+}, k) = d_{P_{i}^{+}}(c_{i}^{+}, k) \), which, by inductive assumption, implies that \( k \) contributes equally to the value of both \( H(c^{+}, P^{+}, t_{0}) \) and \( \mathcal{H}(c^{+}, P^{+}, t_{0}) \). We conclude \( H(c^{+}, P^{+}, t_{0}) = \mathcal{H}(c^{+}, P^{+}, t_{0}) \). Since \( P \), \( P^{+} \), and \( c \) are static between updates, the statement follows by induction.

**Proposition 3 (Cost).** Suppose Assumption 1 holds and, during each agent-cloud exchange, the cloud updates \( c_{i}^{+} \). Then, for all \( t_{0}, t_{2} \), then \( H(c^{+}, P^{+}, \cdot) = \mathcal{H}(c^{+}, P^{+}, \cdot) \).

Proof. Since \( \Phi(\cdot, t_{1}) = \Phi(\cdot, t_{2}) \) for all \( t_{1}, t_{2} \), then \( H(c^{+}, P^{+}, \cdot) \).

- **Proof of Theorem 4.** The cost \( H(c^{+}, P^{+}, t_{0}) \) is static in between-agent-cloud exchanges, as \( P \) and \( c \) are unchanged. Consider an exchange occurring at time \( t_{0} \) involving agent \( i \). By Prop. 3, we have \( H(c^{+}, P^{+}, t_{0}) \leq H(c^{+}, P^{+}, t_{0}) \). Since \( \text{Cov}_{\infty}(Q) \) is static, there is some time \( t_{0} \) after which the value of \( H \) is unchanged. Consider fixed \( t > t_{0} \) at which some agent \( i \) exchanges data with the cloud. Since the value of \( H \) is unchanged, Alg. 1 implies that \( P^{+} \) and \( c \) are unchanged by the update. It follows that \( c \) and \( P^{+} \) converge in finite time. Further, since \( P^{+} \subseteq P \) for any \( t \) (Prop. 2), we have \( P^{+} = P^{+} \subseteq P^{+} \). By persistence of exchanges imposed by \( \Lambda \), this implies that after some finite time, \( P \) and \( P^{+} \) are concurrent.

To prove Pareto optimality of the limiting configuration, consider \( t_{0} \), such that for all \( t > t_{0}, (c, P) \) is unchanged and \( P \) is a \( m \)-partition of \( Q \). Timers are only reset when \( P \) is altered, so assume without loss of generality that \( T_{i} = 0 \). Suppose agent \( i \) exchanges data with the cloud at time \( t > t_{0} \). Alg. 1 implies that there is no \( k \in P \) such that \( \sum_{k \in P} d_{P}(h, k) \Phi(h, t) < \sum_{k \in P} d_{P}(h, c) \Phi(h, t) \) (if not, the cost is lowered by moving \( c \)). Similarly, for \( P \), \( P_{j} = P_{j} \). For \( j \) and \( k \) that is adjacent to \( P_{j} \), we have \( \frac{1}{s_{i}}d_{P}(c_{i}^{+}, k) \geq \frac{1}{s_{j}}d_{P}(c_{i}^{+}, k) \). Thus, \( k \in P_{j} \) (Prop. 2), \( k \in P_{j} \), \( k \in P_{j} \), \( k \in P_{j} \), \( k \in P_{j} \). By persistence of exchanges imposed by \( \Lambda \), this implies that after some finite time, \( P \) and \( P^{+} \) are concurrent.

**Proof of Theorem 5.** By Assumption 2, each agent’s local trajectory planner never directs the agent into its prohibited region, so if no agent-cloud exchange occurs that removes the vertex corresponding to the relevant agent’s location from its coverage region, then the statement is immediate. Suppose, at some time \( t_{0} \), agent \( i \), whose location is associated with some \( k \in P_{i} \), exchanges data with the cloud and \( k \) is removed, i.e., \( k \notin P_{i} \). At time \( t_{0} \), agent \( i \) executes lines 5 and 6 of Alg. 3. Thm. 3 ensures that (i)
exists a path in $G(P_{i}^{\text{A+}})$ between $k$ and $P_{i}^{\text{D-}}$, (ii) all vertices along the path belong to $\text{Proh}_{\text{ID}_{i}^{t}}(\cdot) \setminus \bigcup_{j \neq i} \text{ID}_{j}^{t} P_{j}^{t}$ whenever $i \in (0,t_{0} + \frac{1}{\tau_{i}} d_{P_{i}^{t}}(k,P_{i}^{D-})), \text{and (iii) } P_{i}^{D-} \subseteq P_{i}^{A}$ over the same interval. Thus, if agent $i$ immediately moves along the path, it will exclusively within $\text{Proh}_{\text{ID}_{i}^{t}}(\cdot)$ until it reaches $P_{i}^{A+}$. It remains to show that the agent does not enter $\text{Proh}_{\text{ID}_{i}^{t}}(\cdot) \cap P_{i}^{d}$ while traversing the aforementioned path.

Without loss of generality, consider the update at time $t_{0}$ previously described. Since $k$ is reassigned prior to the update, we have $\text{Proh}_{\text{ID}_{i}^{t_{0}}} \cap P_{i}^{t} = \emptyset$ (vertices in $P_{i}$ are not claimed unless $T_{i} = 0$, implying $t_{0} - \omega_{i}^{A-} > \tau_{i}^{A-}$). By Prop. 2, we deduce $T_{i}^{\text{T+}} > 0$, so no vertices in $P_{i}^{\text{A-}} \cap P_{i}^{\text{ID}_{i}^{t}}$ can belong to $P_{i}^{\text{A+pd+}}$, and no vertex on the constructed path belongs to $\text{Proh}_{\text{ID}_{i}^{t_{0}}}$. Since $T_{i}^{t} > \tau_{i}^{A+} > \frac{1}{\tau_{i}} d_{P_{i}^{t}}(k,P_{i}^{D-}), \text{Proh}_{\text{ID}_{i}^{t}}(\cdot)$ is constant over the interval $(t_{0},t_{0} + \frac{1}{\tau_{i}} d_{P_{i}^{t}}(k,P_{i}^{D-})]$. 

References


