Persistent Connected Power Constrained Surveillance with Unmanned Aerial Vehicles

Pradipta Ghosh¹, Paulo Tabuada², Ramesh Govindan¹, and Gaurav S. Sukhatme¹

Abstract—Persistent surveillance with aerial vehicles (drones) subject to connectivity and power constraints is a relatively uncharted domain of research. To reduce the complexity of multi-drone motion planning, most state-of-the-art solutions ignore network connectivity and assume unlimited battery power. Motivated by this and advances in optimization and constraint satisfaction techniques, we introduce a new persistent surveillance motion planning problem for multiple drones that incorporates connectivity and power consumption constraints. We use a recently developed constrained optimization tool (Satisfiability Modulo Convex Optimization (SMC)) that has the expressivity needed for this problem. We show how to express the new persistent surveillance problem in the SMC framework. Our analysis of the formulation based on a set of simulation experiments illustrates that we can generate the desired motion planning solution within a couple of minutes for small teams of drones (up to 5) confined to a \(7 \times 7 \times 1\) grid-space.

I. INTRODUCTION

Persistent surveillance with multiple unmanned aerial vehicles (UAV) or drones has applications to military operations, search and rescue, and monitoring dynamic situations. Motion planning of a fleet of drones requires rigorous reasoning about the hybrid system behavior of individual drones [1], their battery consumption [2], and their mutual interactions (for collaborative behavior). Besides, motion planning for persistent surveillance requires proper reasoning between a discrete abstraction for task planning (e.g., guaranteeing coverage of a discrete set of regions) and continuous trajectories for motion planning [3].

State-of-the-art. In recent years, researchers have explored persistent surveillance in the context of algorithmic control synthesis from formal specifications captured by a logic formalism, such as Linear Temporal Logic (LTL) [2], [4], [5]. A subset of these approaches discretize the problem space and apply an automata-theory based approach [6]. However, such approaches are impractical for more than a very small number of continuous states due to the curse of dimensionality; finite discretization of the infinite continuous state space leads to an exponential growth of the finite models [7]. Another class of approaches incorporates hierarchical strategies where a higher level planner deals with discrete space planning and a lower level planner generates collision-free dynamically feasible continuous space trajectory according to the high-level planner output [2]. Such a two-step approach often under-represents the dependency between the higher-level planner and the lower level planner and, thus, might not be practical. Tateo et al. [8] and Charrier et al. [9] explore the complexity of high-level (graph-based) motion planning and coverage. Other state-of-the-art motion planners divide a large workspace into smaller sub-regions (to isolate the motion planning for individual drones) and solve for individual sub-regions [10]; this also under-approximates the solution space. A majority of these state-of-the-art formulations lack proper accountability of battery/power consumption and almost none of them imposes practical connectivity constraints (e.g., limited communication range) for communication and information exchange. There do exist a few 2-D graph-theoretic methods for persistent surveillance that include connectivity constraints ([11], [12]), but these lack realistic connectivity and energy consumption models, and do not incorporate practical movement dynamics. Finally, some work has explored path planning with power constraints but without connectivity ([13], [14]).

Recently, Shoukry et al. proposed a promising constrained optimization approach called the Satisfiability Modulo Convex (SMC) Programming [15] to jointly optimize higher-level objectives and lower-level objective as needed for persistent path planning. They modeled [1] motion planning for persistent coverage as a feasibility problem over a combination of Boolean constraints (captures the coverage specifications) and convex constraints with real variables (captures the drone dynamics). SMC follows the logic of a Satisfiability Modulo Theory (SMT) solver [16] and introduces a set of pseudo-Boolean predicates for convex constraints. The SMC solver first solves for a discretized SAT representation of the problem with the pseudo-Booleans abstracting the low-level continuous dynamics. If a feasible solution exists, a convex programming solver checks the feasibility of the proposed plans against the convex constraints and provides an infeasibility certificate to the SAT solver detailing the cause of infeasibility (a counter-example), if any. The SAT solver incorporates the infeasibility certificate to generate a new plan and tries again until a feasible solution exists in both the Boolean and convex domains. Shoukry et al. [1]...
Fig. 1: Illustration of (a) a real world connected persistent surveillance problem and (b) its representation in the motion planning problem. The black regions illustrate the obstacles and the red region illustrates the charging station.

have shown that SMC scales better than existing alternatives. However, their work does not incorporate connectivity or energy consumption constraints which can add significant complexity to the path planning. This paper explores the applicability of SMC for persistent path planning subject to connectivity and energy consumption constraints. It is, to our knowledge, the first to do so.

Our Contribution. We build upon the work of Shoukry et al. [1] and ask: How can we model power and connectivity constraints under the SMC framework? How does the system scale with the additional constraints? To this end, we introduce a set of additional constraints to efficiently model the persistent surveillance motion planning problem subject to power and connectivity constraints. We introduce the concepts of logical links between the drones and a logical multi-hop adjacency matrix. Logical links correspond to a set of convex constraints that represent the real-world separation between the drones with a limit on the communication range ($r_c$). To model power consumption, we introduce a higher level abstraction with discretized power levels (similar to the higher-level path planning) and lower level power trajectory planning.

From simulation experiments, we find that both constraints introduce considerable complexity to the persistent surveillance motion planning problem which becomes intractable beyond a moderate dimension of the workspace (approximately $7 \times 7 \times 1$ unit$^3$) and a small number of drones ($5$). Nonetheless, we can generate the desired motion planning solution within a couple of minutes for all the tractable scenarios. Interestingly, the runtime of the SMC solver also increases with smaller values of connectivity radius ($r_c$); in this regime, problem complexity increases, and SMC is unable to decrease the rate of search space by generating effective counterexamples.

II. PROBLEM FORMULATION

In this section, we model different aspects of the persistent surveillance motion planning problem such as workspace, drone dynamics, power consumption, and connectivity. Consider a set $\mathcal{R} = \{ R_i | i \in \{1, \cdots, N \} \}$ of $N$ drones tasked with the surveillance of a workspace $\mathcal{W} \subset \mathbb{R}^w$ ($w \in \{2, 3\}$ is the dimension of the workspace) with a set of obstacles $\mathcal{O} \subset \mathbb{R}^w$. Assume there exists a set of charging stations $\mathcal{E} = \{ E_i \subset \mathbb{R}^w | i \in \{1, 2, \cdots, Q\} \}$ where the drones start-off their path, end their path, and return to recharge. For simplicity, we discretize time, using $t$ to denote a specific time-slot. Let $||\cdot||_{\infty}$ and $||\cdot||_2$ denote the infinity and Euclidean norms, respectively.

A. Workspace

We assume that the workspace $\mathcal{W}$ consists of a discrete set of obstacles, $\mathcal{O} = \{ O_i \subset \mathbb{R}^w | i \in \{1, 2, \cdots, O\} \}$ where each obstacle is a convex polyhedron. With this assumption, we divide the entire workspace into a discrete set of convex polygons (illustrated in Fig. 1) represented by affine inequalities [1] of the form: $G_i f(x) + h_i \leq 0$, where $f_\mathcal{W}(\cdot)$ is the projection of a drone’s state onto the workspace. Now, denote the set of open/free spaces as: $\mathcal{W} = \{ W_i \subset \mathbb{R}^w | W_i \cap \mathcal{O} = \phi, i \in \{1, 2, \cdots, W\} \}$. To avoid colliding with the obstacles, each drone should always remain in one of the open/free spaces, $W_i \in \mathcal{W}$.

To ensure continuity in the motion, a drone is restricted to neighboring regions in consecutive time-slots. For this reason, we generate an adjacency graph for the workspace ($G_\mathcal{W}$) in which a link exists between two regions $W_i$ and $W_j$ if they are neighbors (they either share a vertex or an edge) to each other. With help of $G_\mathcal{W}$, we identify the neighborhood of each region $W_i$ as $\mathcal{N}(i)$. For example, $\mathcal{N}(1) = \{1, 2, 3, 4, 5\}$ for the workspace illustrated in Fig. 1. For uniform modeling, we assume the set of charging stations regions $\mathcal{E}$ to be a subset of $\mathcal{W}$.

B. Motion Model

We assume that each drone, $R_i \in \mathbb{R}$, follows a discrete time linear system dynamics [1] as follows.

$$x_{t+1}^i = A_i x_t^i + B_i u_t^i \tag{1}$$

where $x_t^i \in \mathbb{R}^n$ is the state of drone $R_i$ at time $t \in \mathbb{N}$, $u_t^i$ is the control input to drone $R_i$ at time $t \in \mathbb{N}$, and $A_i, B_i$ capture the drone dynamics. The initial state of a drone $R_i$ is $\pi_0$. The state and input of a drone are bounded with respective upper bounds as $\pi^i$ and $\pi^u$, i.e.,

$$||x_t^i||_{\infty} \leq \pi^i \text{ and } ||u_t^i||_{\infty} \leq \pi^u \tag{2}$$

We use feedback linearized dynamics for the state-space modelling of the nonlinear dynamics of a differentially flat or feedback linearizable drone.

C. Power Model

We assume that the power available in a fully-charged drone is $P$ units. Power consumption in a drone depends on its movement pattern in terms of its velocity, acceleration, or jerk embedded within the state of the drone ($x_t^i$). Thus, we model power consumption as a linear function of the state of a drone [2].

$$p_{t+1}^i = p_t^i - C_i x_t^i \tag{3}$$

where $C_i$ represents the relation between the movement and power consumption, and $p_t^i$ represents the available power.
of drone $R_i$ at time $t$. If the drone is in charging station, it will recharge the battery at a fixed rate ($\delta_p$) i.e.,

$$p_{i,t+1}^l = p_{i,t}^l + \delta_p$$

(4)

We leave recharging to future work.

D. Collision Avoidance

At any point in time, no two drones should collide with each other. This implies that the drones must be located at some distance ($\epsilon > 0$) from each other. Since $f_W(x_i^l)$ returns the location of drone $R_i$ at time $t$, any two drones $R_i$ and $R_j$ should fulfill the following condition to avoid collision.

$$||f_W(x_i^l) - f_W(x_j^l)||_\infty \geq \epsilon \quad \forall t \quad \forall i \neq j$$

(5)

where $\epsilon > 0$ is the minimum inter-drone distance allowed along any dimension.

E. Connectivity

For efficient and reliable operation of a group of drones, they need to communicate with each other [17]. To achieve this, at each point in time, the network of drones should be connected i.e., there must be a communication path between any two drones. Two drones can directly communicate with each other (i.e., are directly connected) if they are located within a distance threshold called the communication radius, $r_c$:

$$||f_W(x_i^l) - f_W(x_j^l)||_2 \leq r_c$$

(6)

Direct links between drones in a network can be concisely represented in terms of an adjacency matrix $A$ where $A_{i,j} = 1$ if there exists a direct link between drones $R_i$ and $R_j$ with $i \neq j$, and $A_{i,j} = 0$ otherwise. In a connected network, two non-neighboring drones can communicate by forming multi-hop paths via other drones in the network. The matrix $A^n = \prod_{k=1}^{n-1} A$ concisely represents the number of such $n$ hop paths between every pair of drones i.e., $A_{i,j}^n$ is the number of $n$-hop paths between drones $R_i$ and $R_j$. For a connected network with $N$ drones, there must exist at-least one communication path between any two drones with a maximum length of $N - 1$. Thus, we can construct a matrix as follows.

$$\overline{A}_t = \sum_{k=1}^{N-1} A^k$$

(7)

where $A_i$ is the adjacency matrix at time $t$. Now, $\overline{A}_{t,i,j}$ gives us the number of paths between drones $R_i$ and $R_j$ at time $t$ that are less than $N$ hops. A network of drones is connected if $\overline{A}_{i,j} > 0 \forall i,j \in \{1, \cdots, N\}, i \neq j$ and disconnected otherwise.

F. The Persistent Coverage Problem

Given a combination of $\{W, O, E\}$, the objective is to find a feasible plan for the motion of a set of drones $R$ that satisfies all of these conditions:

- Over any time window of length $T_i$, at least one drone covers every region $W_i \in W$
- At any point in time $t$, the network of drones is connected
- The drones start from one of the charging stations and return to any of the charging stations before running out of battery
- The whole system should run for a duration $T$ (the desired sensing duration)
- The movements of the drones follow feasible dynamics (introduced in §II-B).
- The drones should not collide with each other at any point in time.

If a solution does not exist, the solver should generate an infeasibility certificate.

III. SATISFIABILITY MODULO CONVEX (SMC) THEORY BACKGROUND

SMC theory builds upon the Boolean Satisfiability (SAT) [16] problem. In SAT, given a conjunction ($\land$) of multiple Boolean clauses, the objective is to find a feasible assignment to the respective Boolean variables to satisfy the Boolean clauses. Consider the following example:

$$\psi = a_1 \land (a_2 \lor a_3) \land (a_1 \lor a_3)$$

(8)

where, $a_1, a_2, a_3$ are three Boolean variables. A SAT solver like z3 [18] systematically explores assignments to the Boolean variables $a_1, a_2, a_3$ such that the entire clause evaluates to be TRUE i.e., $\psi = \text{TRUE}$. For the example in Eqn. 8, a SAT solver can return the following assignment: $a_1 = \text{TRUE}, a_2 = \text{TRUE}, a_3 = \text{FALSE}$. On the other hand, if no solution exists, a SAT solver can determine that and not return any feasible assignment. For example, $\psi = a_1 \land a_2 \land (\neg a_1 \lor \neg a_2)$ is not satisfiable under any assignment to the Boolean variables: $a_1$ and $a_2$.

SAT problems are NP-complete; however, recent developments in efficient SAT solvers like z3 [18] have made it possible to use SAT theory to solve a wide range of practical problems such as circuit design [19]. Nonetheless, SAT itself cannot express complex problems involving linear or convex constraints.

To this end, Shoukry et al. [15] have introduced the Satisfiability Modulo Convex (SMC) theory designed to address the feasibility of mixed-integer convex problems. In the SMC framework, a SAT solver suggests admissible assignments for a problem’s Boolean variables, a convex solver (e.g., [20]) suggests admissible values of the problem’s real variables, and a set of pseudo-Boolean variables forms a bridge between the SAT solver and the convex solver.

Consider the following example:

$$(a_1 \lor a_2) \land (a_2 \lor a_3) \land (a_2 \implies x_1^2 + x_2^2 \leq 10)$$

(9)

Here, $a_1, a_2, a_3$ are Boolean variables and $x_1, x_2$ are real variables. The first clause states that if $a_2$ is TRUE, there should also exist a solution to the convex constraint: $x_1^2 + x_2^2 \leq 10$. To tackle such constraint, an SMC solver introduces a pseudo-Boolean variable, say $a_4$, to replace the convex constraint and convert it to a SAT clause, and runs its embedded SAT solver. If the SAT solver returns an assignment such that $a_2 = \text{TRUE}$ and $a_4 = \text{TRUE}$, the SMC solver employs
TABLE I: Summary of Symbols and Boolean Predicates

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>Number of Drones</td>
</tr>
<tr>
<td>T</td>
<td>Maximum (discrete) time duration</td>
</tr>
<tr>
<td>W</td>
<td>Number of discrete regions in W</td>
</tr>
<tr>
<td>P</td>
<td>Maximum power level</td>
</tr>
<tr>
<td>N(j)</td>
<td>Neighborhood of region W_j including itself to account for staying in same region</td>
</tr>
<tr>
<td>E</td>
<td>Set of charging stations</td>
</tr>
<tr>
<td>T_j</td>
<td>Time period between consecutive surveillance for region W_j</td>
</tr>
<tr>
<td>A_t</td>
<td>Adjacency Matrix at time t</td>
</tr>
<tr>
<td>x_i^t</td>
<td>The state of drone i at time t</td>
</tr>
<tr>
<td>p_i^t</td>
<td>The available power of drone i at time t</td>
</tr>
</tbody>
</table>

Symbols Operations

∧ Logical ‘AND’

∨ Logical ‘OR’

⇒ Logical ‘Implies’

Table I lists the symbols and predicates we use.

I. Introduction

First, we discuss how we encode the higher-level planning for persistent surveillance. The convex solver to find a satisfying assignment for the convex constraint. If a solution exists, the SMC solver returns immediately (as in our example). Otherwise, it generates a counter-example (additional pseudo-Boolean constraints) to constrain the subsequent search space for the SAT solver, and re-invokes the SAT solver. The process continues until an assignment exists for both the Boolean variables and the real variables or the solver finds the problem to be unsolvable.

Why SMC? There exist other constrained optimization techniques like Mixed Integer Linear Programming (MILP) [21] that are often employed to solve complex real-world problems. However, Shoukry et al. [15] has already demonstrated that SMC outperforms such techniques for robotic path planning. Moreover, only SMC satisfies the expressivity needs of the persistent surveillance problem (§IV), so we use SMC as a framework to formulate and solve this problem.

IV. Problem Encoding in SMC

In this section, we describe how we encode the persistent surveillance problem in SMC. This encoding has two parts: a SAT formulation for the higher-level planning and a convex formulation for the lower level drone dynamics planning. Table I lists the symbols and predicates we use.

A. High-Level Discrete Planning

First, we discuss how we encode the higher-level planning for persistent surveillance.

Path constraints. With \( \Pi_{jt}^i \) as the Boolean predicate to represent the location of drone \( i \), we can model the path constraints as follows.

\[ \Pi_{jt}^i \] Occupies one grid at a time (\( \psi_1 \)). At time \( t \), a drone can be in one of the unoccupied regions (\( W_j \in \overline{W} \)); we represent it logically by restricting \( \Pi_{jt}^i = \text{TRUE} \) (we represent logical TRUE and FALSE via numerals: 1 and 0, respectively) for only one value of \( j \) for each drone \( R_i \) at time \( t \).

Start from charging station (\( \psi_2 \)). Each drone should start from one of the charging stations \( E \), i.e., \( \Pi_{10}^i = \text{TRUE} \) if and only if \( W_j \in E \). We assume, without loss of generality, that all drones start from the same charging stations (say \( W_j \in E \)); this implies \( \Pi_{10}^i = \text{TRUE} (j = 1 \text{ at time } t = 0) \) for every drone \( R_i \).

Finish at a charging station (\( \psi_3 \)). A drone should return to one of the charging stations after finishing the task.

Movement continuity (\( \psi_4 \)). After each time slot, a drone at region \( j \) can move to one of the regions in \( N(j) \), which includes the current region (Tbl. I) and its immediate neighbors (§II-A).

Surveillance. A drone needs to hover within a region \( W_j \) to sense that region. Thus, at least one robot must surveil each region \( W_j \in \overline{W} \) every time period \( T_j \) (§III). We encode this using the following logical constraints.

First surveillance (\( \psi_5 \)). At least one robot must visit each region \( W_j \) once within the first \( T_j \) time-slots.

Periodic surveillance (\( \psi_6 \)). If a region \( W_j \in \overline{W} \) is surveilled at time \( t \), the next surveillance must take place between \( t \) and \( t + T_j \) (for periodic coverage).

Power constraints. At any point in time \( t \), a drone \( R_i \)'s power is \( \langle p_i^t \rangle \). Power values range from 0 to \( P \). We discretize battery power into \( P + 1 \) discrete power levels. Let \( \Phi_{kt}^i \) imply

3We do not solve for initial allocation of drones among a set of charging stations; we assume this to be an input to the system.
that drone $R_i$ is in the power state of $k$ at time $t$ (Tbl. I). We model power consumption using the following constraints.

**Occupy one power state at a time** ($\psi_7$). At time $t$, a drone can be in one of the power states ($p \in \{0, 1, \ldots, P\}$); we represent this by restricting $\Phi_{kt} = \text{TRUE}$ for exactly one value of $k$ at time $t$.

**Initial Power State** ($\psi_8$). A fully charged drone starts with a power state of $P$ i.e., $\Phi_{k0} = \text{TRUE}$ ($j = P$ at time $t = 0$).

\[
\psi_7 := \bigwedge_{i=1}^{N} \bigwedge_{k=0}^{P} \left( \sum_{i=1}^{P} \Phi_{kt} = 1 \right), \quad \psi_8 := \bigwedge_{i=1}^{N} \Phi_{k0} \tag{15}\]

**Power Transition** ($\psi_9$). During operation, the battery power can only decrease since we do not consider re-charging. The power can decrease by any amount based on the movement of the drone. Thus, a drone $R_i$ can either stay at the same power state or transition into any of the lower power states:

\[
\psi_9 := \bigwedge_{i=1}^{N} \bigwedge_{k=0}^{P} \left( \Psi_{kt} \Rightarrow \left( \sum_{k=0}^{P} \Phi_{(k-l)(t+1)} = 1 \right) \right) \tag{16}\]

**Out of Power** ($\psi_{10}$). If a drone $R_i$ runs out of battery at time $t$ i.e., $\Phi_{0t} = 1$, the drone must be, at time $t$, at one of the charging stations i.e., \( \bigcup_{j \in \mathbb{R}} F_j \).

\[
\psi_{10} := \bigwedge_{i=1}^{N} \bigwedge_{j=1}^{W} \left( \Phi_{0t} \Rightarrow \left( \sum_{i=1}^{N} \Psi_{jt} \right) \right) \tag{17}\]

This constraint ensures that, just before it loses power, the drone must return to a charging station.

**Connectivity.** Let $t^{ij}_t$ be the Boolean predicate that represents a link between drones $R_i$ and $R_j$ at time $t$ (Tbl. I). Persistent surveillance requires that the network of drones should be connected at all times; a drone can get disconnected from the network only when it runs out of battery power. To model this, we construct a logical adjacency matrix $A_t$ for time $t$ as well as a $N - 1$ hop adjacency matrix as follows (§II-E).

\[
A_t = \begin{bmatrix}
0 & t^{i2}_t & \cdots & t^{iN}_t \\
t^{12}_t & 0 & \cdots & t^{1N}_t \\
\vdots & \vdots & \ddots & \vdots \\
t^{N2}_t & t^{N1}_t & \cdots & 0 \\
\end{bmatrix} \quad \text{and} \quad \overline{A}_t = \sum_{k=1}^{N-1} A_t^k \tag{18}\]

**Disconnect Drones with Drained Battery** ($\psi_{11}$). If a drone’s battery is drained at time $t$ ($\Phi_{0t} = \text{TRUE}$), it does not form any communication link with other drones ($t^{ij}_t = \text{FALSE} \ \forall i \neq j$).

\[
\psi_{11} := \bigwedge_{i=1}^{N} \bigwedge_{j=1}^{N} \left( \Phi_{0t} \Rightarrow \bigwedge_{j \neq i} \neg t^{ij}_t \right) \tag{19}\]

**Multi-Hop Connectivity** ($\psi_{12}$). If two drones $R_i$ and $R_j$ have non-zero power at time $t$ ($\Phi_{0t} = \text{FALSE}$ and $\Phi_{0t} = \text{FALSE}$), to guarantee connectivity between them, \( \overline{A}_{t}^{ij} > 0 \); this implies drones $R_i$ and $R_j$ have at least one communication path with a maximum length of $N - 1$ hops.

\[
\psi_{12} := \bigwedge_{t=0}^{T} \bigwedge_{i=1}^{N} \bigwedge_{j=i+1}^{N} \left( \neg (\Phi_{0t} \lor \Phi_{0t}) \Rightarrow \overline{A}_{t}^{ij} > 0 \right) \tag{20}\]

**Overall Formulation.** The higher level abstraction of the persistent surveillance problem can be completely represented as: $\psi_h := \bigwedge_{t=0}^{T} \bigwedge_{i=1}^{N} \psi_{i}$.

B. Low-Level Trajectory Planning

While the upper level SAT encoding represents the mission objectives, it cannot directly output a valid trajectory for the drones since it does not model drone dynamics, state, and input constraints. To model these, we need to use pseudo-Boolean constraints as well as convex constraints on the real variables.

**Region Constraints.** ($\psi_{13}$) A logical solution with $\Pi_{jt} = \text{TRUE}$ implies that the physical location of drone $R_i$ with state $x^i_j$ must be within the polyhedral region $W_j$ at time $t$ i.e., $f_W(x^i_j)$ must satisfy the affine constraints (§II-A):

\[
G_j f_W(x^i_j) + h_j \tag{21}\]

**Power Constraints.** ($\psi_{14}$, $\psi_{15}$, $\psi_{16}$, $\psi_{17}$) Similar to the region constraints (see above), if the logical solution includes $\Phi_{kt} = \text{TRUE}$, the actual power ($p^i_t$) of drone $R_i$ must be between $k - 1$ and $k$ at time $t$ (if $k = 0$, $p^i_t = 0$):

\[
\psi_{14} := \bigwedge_{t=0}^{T} \bigwedge_{i=1}^{N} \bigwedge_{k=0}^{P} \left( \Phi_{kt} \Rightarrow (p^i_t > k - 1) \land (p^i_t \leq k) \right) \tag{22}\]

\[
\psi_{15} := \bigwedge_{t=0}^{T} \bigwedge_{i=1}^{N} \left( \Phi_{0t} \Rightarrow (p^i_t = 0) \right) \tag{23}\]

**Initial Power Constraints** ($\psi_{16}$) The initial power of each drone should be $P$ i.e., $p^i_0 = P \ \forall i \in \{1, \ldots, N\}$.

**Power Transition Constraints** ($\psi_{17}$). While Eqn. 16 models the logical transition of power using discrete abstraction, the actual power transition should be continuous (§II-C) and follow Eqn. 3.

\[
\psi_{16} := \bigwedge_{i=1}^{N} \left(p^i_0 = P \right) \quad \psi_{17} := \bigwedge_{t=0}^{T} \bigwedge_{i=1}^{N} \left(p^{i+1}_t = p^i_t - C_i x^i_t \right) \tag{23}\]

**Link Connectivity Constraints.** ($\psi_{18}$) Link connectivity between two drones $R_i$ and $R_j$ at time $t$ ($t^{ij}_t = \text{TRUE}$) implies they are located at least $r_c$ distance from each other (§II-E).

\[
\psi_{18} := \bigwedge_{t=0}^{T} \bigwedge_{i=1}^{N} \bigwedge_{j=i+1}^{N} \left( t^{ij}_t \Rightarrow ||f_W(x^i_t) - f_W(x^j_t)|| \leq r_c \right) \tag{24}\]
State Transition Constraints. \((\psi_{19}, \psi_{20}, \psi_{21}, \psi_{22})\) Let \(x_t^i\) be the state of drone \(R_i\) at time \(t\) that includes the position, velocity, acceleration, and jerk of the drone. We need to add the state transition equations (§II-B) directly to the SMC formulation as they must be valid regardless of the higher level planning:

\[
\psi_{19} := \bigwedge_{t=0}^{T-1} \bigwedge_{i=1}^{N} \left( x_{t+1}^i = A_i x_t^i + B_i u_t^i \right), \quad \psi_{20} := \bigwedge_{i=1}^{N} \left( x_0^i = x_0^i \right)
\]

\[
\psi_{21} := \bigwedge_{t=0}^{T} \bigwedge_{i=1}^{N} \left( ||x_t^i||_\infty \leq \tau \right), \quad \psi_{22} := \bigwedge_{t=0}^{T} \bigwedge_{i=1}^{N} \left( ||u_t^i||_\infty \leq \pi \right)
\]

(25)

Collision Avoidance. \((\psi_{23}, \psi_{24}, \psi_{25})\) At any point in time, no two drones should collide with each other i.e., they must be at some distance \(\epsilon > 0\) from each other (§II-D). To encode this, we introduce additional pseudo-Boolean variables similar to [1]: \(\{(c_{ij}^k, d_{ij}^k)\}_{k \in \{1, \ldots, w\}, i,j \in \{1, \ldots, N\}, i \neq j}\) where \(w\) is the dimension of the space. Using these variables, we introduce the following set of constraints to ensure collision avoidance.

\[
\psi_{23} := \bigwedge_{t=1}^{T} \bigwedge_{j=1}^{N} \bigwedge_{k=1}^{w} (c_{ij}^k \implies (f_W^k(x_t^i) - f_W^k(x_t^j)) \geq \epsilon)
\]

\[
\psi_{24} := \bigwedge_{t=1}^{T} \bigwedge_{j=1}^{N} \bigwedge_{k=1}^{w} (d_{ij}^k \implies (-f_W^k(x_t^i) + f_W^k(x_t^j)) \geq \epsilon)
\]

\[
\psi_{25} := \bigwedge_{t=1}^{T} \bigwedge_{j=1}^{N} \bigwedge_{k=1}^{w} \left( \sum_{k=1}^{w} (c_{ij}^k + d_{ij}^k) \geq 1 \right)
\]

(26)

where \(f_W^k(.)\) represents the location of a drone along the \(k\)-th dimension of the workspace.

Overall Formulation. In summary, the low level dynamics can be modeled as \(\psi_1 := \bigwedge_{i=1}^{25} \psi_i\). The complete SMC encoding of the persistent surveillance problem is \(\psi := \psi_h \land \psi_l = \bigwedge_{i=1}^{25} \psi_i\)

V. EXPERIMENTS AND RESULTS

In this section, we analyze different aspects of the proposed persistent surveillance formulation such as the complexity of the problem, the effect of the connectivity radius, and the effect of the number of drones on the motion planning solution.

Method. We evaluate the formulation for the scenario in which each region is sensed at least once within the entire path planning duration \(T\); in other words, \(T_i = T\forall W_i \in \mathcal{W}\) (§II-F). We consider a simple grid/cubic partitioning of the workspace instead of polyhedral partitioning (§II-A). In this setting, we increase \(T\) from 0 until our solver outputs a feasible trajectory or a maximum value of \(T\) is reached.

Comparison. To understand the effect of the connectivity and power constraints, we consider three different variants of the SMC formulation: (1) No-Power-No-Conn: A formulation without the power and connectivity constraints, (2) Power-No-Conn: A formulation with the power constraints but not the connectivity constraints, and (3) Power-Conn: A formulation with the power and connectivity constraints.

Experimental setup. The experiments run on an Intel Core i7 8700 CPU with 6 3.20GHz cores. The statistics presented here is partially limited by the available memory and compute power; however, the analysis is valid for a broader range of machine specifications.

A. Complexity Analysis

To analyze how much complexity the connectivity and power constraints add to the basic path planning problem [1], we run a set of simulations with four drones \((N = 4)\) for four different sizes of the workspace: \(5 \times 5 \times 1\), \(6 \times 6 \times 1\), \(7 \times 7 \times 1\), and \(8 \times 8 \times 1\). The number of obstacles is kept constant at 4 in all scenarios. We limit the maximum runtime of the solver to 30 minutes.

![Fig. 2: The runtime of the solver for varying workspace dimensions. The missing bar for \(8 \times 8 \times 1\) with Power-Conn implies no solution was generated.](image-url)

The comparison of the solver runtime (presented in Fig. 2) shows that the runtime is highest for the Power-Conn formulation (it could not generate a solution for the \(8 \times 8 \times 1\) scenario). The runtime for Power-No-Conn is also higher than the base No-Power-No-Conn formulation. This illustrates that the power and connectivity constraints add significant complexity to the persistent surveillance path planning. To analyze this further, we compare the number of Boolean and real variables as well as the Boolean and convex constraints (summarized in Tbl. II). Tbl. II shows that the number of variables and constraints is significantly higher with the power and connectivity constraints which in turn increases the search space dimensions and the time to find a solution. An analysis of the encoding (§IV) verifies that the total number of constraints to encode connectivity and power are \(\Omega(T \cdot N^2)\) and \(\Omega(T \cdot N \cdot P)\), respectively, where \(T\) is the time duration, \(N\) is the number of drones, and \(P\) is the number of discrete power levels; this validates our findings in Tbl. II.
TABLE II: Statistics of the number of constraints for $N = 4$

<table>
<thead>
<tr>
<th>Size</th>
<th>Scenario</th>
<th>$T$</th>
<th># of Boolean Variables</th>
<th># of Real Variables</th>
<th># of Boolean Constraints</th>
<th># of Convex Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>5x5x1</td>
<td>No-Power-No-Conn</td>
<td>7</td>
<td>6584</td>
<td>772</td>
<td>1904</td>
<td>1426</td>
</tr>
<tr>
<td></td>
<td>Power-Conn</td>
<td>7</td>
<td>3544</td>
<td>700</td>
<td>1944</td>
<td>1528</td>
</tr>
<tr>
<td></td>
<td>Power-No-Conn</td>
<td>7</td>
<td>3594</td>
<td>700</td>
<td>1968</td>
<td>1578</td>
</tr>
<tr>
<td>6x6x1</td>
<td>No-Power-No-Conn</td>
<td>11</td>
<td>10694</td>
<td>1055</td>
<td>2995</td>
<td>2105</td>
</tr>
<tr>
<td></td>
<td>Power-Conn</td>
<td>11</td>
<td>7540</td>
<td>1052</td>
<td>3877</td>
<td>2772</td>
</tr>
<tr>
<td></td>
<td>Power-No-Conn</td>
<td>11</td>
<td>7612</td>
<td>1052</td>
<td>3913</td>
<td>2844</td>
</tr>
<tr>
<td>7x7x1</td>
<td>No-Power-No-Conn</td>
<td>12</td>
<td>16604</td>
<td>1092</td>
<td>6244</td>
<td>2855</td>
</tr>
<tr>
<td></td>
<td>Power-Conn</td>
<td>12</td>
<td>11004</td>
<td>1140</td>
<td>5264</td>
<td>3548</td>
</tr>
<tr>
<td></td>
<td>Power-No-Conn</td>
<td>12</td>
<td>11082</td>
<td>1140</td>
<td>5303</td>
<td>3626</td>
</tr>
<tr>
<td>8x8x1</td>
<td>No-Power-No-Conn</td>
<td>16</td>
<td>17564</td>
<td>1242</td>
<td>7325</td>
<td>6556</td>
</tr>
<tr>
<td></td>
<td>Power-Conn</td>
<td>15</td>
<td>17716</td>
<td>1404</td>
<td>8141</td>
<td>5204</td>
</tr>
<tr>
<td></td>
<td>Power-No-Conn</td>
<td>20</td>
<td>23482</td>
<td>1844</td>
<td>10984</td>
<td>7050</td>
</tr>
</tbody>
</table>

B. Effect of Connectivity Constraints

To understand the effect of the connectivity constraints on the problem complexity, we perform another set of experiments with the Power-Conn formulation for 2 drones where we vary the connectivity radius $r_c$. Keeping the workspace dimensions fixed, we pick different values for $r_c$: $\infty$, 2, 1.5, 1, 0.5, 0.25, 0.1. For this set of experiments, we choose three different workspaces with dimensions: $3 \times 4 \times 1$, $4 \times 4 \times 1$, and $5 \times 5 \times 1$. The solver was able to generate a solution for all different values of $r_c$ (within 30 minutes) for the $3 \times 4 \times 1$, and $4 \times 4 \times 1$ workspace. For the $5 \times 5 \times 1$ workspace, the solver could generate solutions only for $r_c = \infty$, 2, and 1.5. This is a surprising find as the number of constraints does not change for different values of $r_c$. Careful analysis of the results shows that smaller values of $r_c$ makes the runtime higher due to two reasons. Firstly, with a smaller value of $r_c$, the drones need to maneuver in close proximity and the system does not benefit from having multiple drones for the surveillance task (illustrated in Fig. 3). This causes longer paths (in terms of $T$) with increased search complexity as shown in Fig. 3. Secondly, for smaller values of $r_c$, the rate of reduction in search space via counterexample decreases which in turn increases the number of iterations in the search process and consecutively the time to complete. Note that, for better illustration, we present only the 2D projections of the 3D paths in Fig. 3. A sample 3D path output from the solver is illustrated in Fig. 4.

C. Effect of Varying Number of Drones

In our third and final set of experiments, we vary the number of drones ($N$) from 1 to 8 for all three variants for a fixed workspace with dimensions $5 \times 5 \times 1$. The SMC solver was able to generate a solution for the No-Power-No-Conn problem regardless of the value of $N$ whereas it could not solve for both the Power-No-Conn and Power-Conn formulations with more than 5 drones. To understand the reason, we compare the number of variables and constraints in Tbl. III. Tbl. III shows that the number of Boolean and Real variables in either Power-No-Conn or Power-Conn is $\approx 2000$ more in number than No-Power-No-Conn for more than 5 drones whereas this difference is in the order of $500 - 1000$ for $\leq 5$ drones. This explains the SMC solver’s incapability to produce a solution for Power-No-Conn or Power-Conn with more than 5 drones. This observation qualitatively matches the findings of [8] which states that deciding whether a feasible plan exists for multiple drones with connectivity constraints is a PSPACE-complete problem; the power constraints adds more to the complexity. This suggests that to generate a solution for a large number of UAVs, we need to either compress the number of variables and constraints or rapidly reduce the search space; we have left this as a future work.

Interestingly, for this particular instantiation of the surveillance problem, adding more drones does not help beyond a certain threshold (Tbl. III demonstrates no improvement in path length for more than 4 drones). This implies that for a generic persistent surveillance problem one could optimize (reduce) the required number of drones while fulfilling all the requirements.

VI. CONCLUSION

We presented a new formulation of the well-known persistent surveillance problem that incorporates practical constraints pertaining to network connectivity and battery power consumption. Using Satisfiability Modulo Convex optimization, we proposed a solution to this problem. We showed that the connectivity and battery power constraints add significant complexity to multi-UAV persistent surveillance motion planning and thus cannot be solved at a large scale with the SMC optimizer. To this, we identified the three main factors that control the SMC solver runtime: the workspace dimensions ($W$), the number of robots ($N$), and the inter-robot communication radius ($r_c$). With this work as the first step towards the development of techniques for...
the deployment of connected multi-drone systems, we plan to explore a number of future research directions: a detailed formal analysis of the computational complexity of the formulation; a comparison of our integrated approach for path planning (high-level logical planning and low-level trajectory planning) with a state-of-the-art approach that decouples high-level planning from low-level planning to quantify the limitations and the advantages of our approach; a Mixed Integer Non-Linear Programming (MINLP) formulation to solve the problem which might be able to express more general connectivity constraints [22]. Other future directions include, but are not limited to, solving this problem at larger scales by adding approximations or hierarchy, incorporating battery recharging constraints, and experimenting with a real-world testbed.

TABLE III: Statistics of the number of constraints for varying number of drones

<table>
<thead>
<tr>
<th># of drones</th>
<th>Scenario</th>
<th># of Boolean Variables</th>
<th># of Real Variables</th>
<th># of Boolean Constraints</th>
<th># of Convex Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No-Power-No-Conn</td>
<td>21</td>
<td>6055</td>
<td>2903</td>
<td>483</td>
</tr>
<tr>
<td></td>
<td>Power-No-Conn</td>
<td>10</td>
<td>9534</td>
<td>4371</td>
<td>746</td>
</tr>
<tr>
<td>2</td>
<td>No-Power-No-Conn</td>
<td>15</td>
<td>13239</td>
<td>6117</td>
<td>1053</td>
</tr>
<tr>
<td></td>
<td>Power-No-Conn</td>
<td>15</td>
<td>13237</td>
<td>6149</td>
<td>1053</td>
</tr>
<tr>
<td>3</td>
<td>No-Power-No-Conn</td>
<td>15</td>
<td>13239</td>
<td>6117</td>
<td>1053</td>
</tr>
<tr>
<td></td>
<td>Power-No-Conn</td>
<td>15</td>
<td>13237</td>
<td>6149</td>
<td>1053</td>
</tr>
<tr>
<td>4</td>
<td>No-Power-No-Conn</td>
<td>13</td>
<td>14420</td>
<td>5921</td>
<td>1176</td>
</tr>
<tr>
<td></td>
<td>Power-No-Conn</td>
<td>13</td>
<td>14420</td>
<td>5921</td>
<td>1176</td>
</tr>
<tr>
<td>5</td>
<td>No-Power-No-Conn</td>
<td>13</td>
<td>14835</td>
<td>6930</td>
<td>1470</td>
</tr>
<tr>
<td></td>
<td>Power-No-Conn</td>
<td>13</td>
<td>14835</td>
<td>6930</td>
<td>1470</td>
</tr>
<tr>
<td>6</td>
<td>No-Power-No-Conn</td>
<td>13</td>
<td>15395</td>
<td>8853</td>
<td>1535</td>
</tr>
<tr>
<td></td>
<td>Power-No-Conn</td>
<td>13</td>
<td>15395</td>
<td>8853</td>
<td>1535</td>
</tr>
<tr>
<td>7</td>
<td>No-Power-No-Conn</td>
<td>13</td>
<td>15395</td>
<td>8853</td>
<td>1535</td>
</tr>
<tr>
<td></td>
<td>Power-No-Conn</td>
<td>13</td>
<td>15395</td>
<td>8853</td>
<td>1535</td>
</tr>
<tr>
<td>8</td>
<td>No-Power-No-Conn</td>
<td>13</td>
<td>15395</td>
<td>8853</td>
<td>1535</td>
</tr>
<tr>
<td></td>
<td>Power-No-Conn</td>
<td>13</td>
<td>15395</td>
<td>8853</td>
<td>1535</td>
</tr>
</tbody>
</table>

REFERENCES


