2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Persistent Connected Power Constrained Surveillance with Unmanned Aerial Vehicles

Pradipta Ghosh[1], Paulo Tabuada[2], Ramesh Govindan[1], and Gaurav S. Sukhatme[1]

***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].

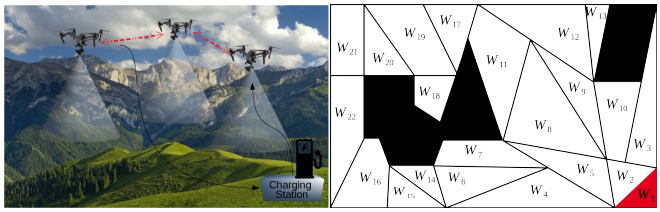
State-of-the-art. In recent years, researchers have explored persistent surveillance in the context of algorithmic control synthesis from formal specifications captured by a logic formalism, such as Linear Temporal Logic (LTL) [2], [4], [5]. A subset of these approaches discretize the problem space and 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]

Research reported in this paper was sponsored in part by the Army Research Laboratory under Cooperative Agreement W911NF-17-2-0196. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

Pradipta Ghosh, Ramesh Govindan, and Gaurav S. Sukhatme are with the Department of Computer Science, University of Southern California, Los Angeles, CA, USA, Email: {pradiptg@usc.edu, ramesh@usc.edu, gaurav@usc.edu}

[2]Paulo Tabuada is with the Department of Electrical and Computer Engineering, University of California at Los Angeles, Los Angeles, CA, USA, Email: {tabuada@ucla.edu}



(a) (b)

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 limitation 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 $\mathbf{R} = \{R_i | i \in \{1, \cdots, N\}\}$ of N drones tasked with the surveillance of a workspace $\mathbf{W} \subset \mathbb{R}^w$ ($w \in \{2, 3\}$ is the dimension of the workspace) with a set of obstacles $\mathbf{O} \subset \mathbb{R}^w$. Assume there exists a set of charging stations $\mathbf{E} = \{E_i \subset \mathbb{R}^w | i \in \{1, 2, ..., Q\}\}$ 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 $||.||_\infty$ and $||.||_2$ denote the infinity and Euclidean norms, respectively.

A. Workspace

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

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 ($\mathcal{G}_W$) 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 $\mathcal{G}_W$, 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 $\mathbf{E}$ to be a subset of $\overline{\mathbf{W}}$.

B. Motion Model

We assume that each drone, $R_i \in \mathbf{R}$, 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 $\overline{x}_0^i$. The state and input of a drone are bounded with respective upper bounds as $\overline{x}^i$ and $\overline{u}^i$, i.e.,

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

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 (δ_p^c) *i.e.,*

$$p_{t+1}^i = p_t^i + \delta_p^c \tag{4}$$

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 $f_{\mathbf{W}}(x_t^i)$ returns the location of drone R_i at time t, any two drones R_i and R_j should fulfil the following condition to avoid collision.

$$||f_{\mathbf{W}}(x_t^i) - f_{\mathbf{W}}(x_t^j)||_\infty \geq \epsilon \qquad \forall t \qquad \forall i \neq j \tag{5}$$

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:

$$||f_{\mathbf{W}}(x_t^i) - f_{\mathbf{W}}(x_t^j)||_2 \leq r_c \tag{6}$$

Direct links between drones in a network can be concisely represented in terms of an adjacency matrix $\mathbf{A}$ where $\mathbf{A}|_{ij}=1$ if there exists a direct link between drones R_i and R_j with $i \neq j$, and $\mathbf{A}|_{ij}=0$ otherwise. In a connected network, two non-neighboring drones can communicate by forming multi-hop paths via other drones in the network. The matrix $\mathbf{A}^n = \prod_{i=1}^{n} \mathbf{A}$ concisely represents the number of such n hop paths between every pair of drones *i.e.,* $\mathbf{A}^n|_{ij}$ 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.

$$\overline{\mathbf{A}}_t = \sum_{k=1}^{N-1} \mathbf{A}_t^k \tag{7}$$

where $\mathbf{A}_t$ is the adjacency matrix at time t. Now, $\overline{\mathbf{A}}_t|_{ij}$ 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 $\overline{\mathbf{A}}_t|_{ij} > 0 \ \forall i,j \in \{1,\cdots,N\},\ i \neq j$ and disconnected otherwise.

F. The Persistent Coverage Problem

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

- Over any time window of length T_i, at least one drone covers every region $W_i \in \overline{\mathbf{W}}$
- 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 ($\wedge$) 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:

$$\psi = a_1 \wedge (a_2 \vee a_3) \wedge (a_1 \vee a_3) \tag{8}$$

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.,* $\psi =$ TRUE. For the example in Eqn. 8, a SAT solver can return the following assignment: $a_1 =$ TRUE , $a_2 =$ TRUE, $a_3 =$ FALSE. On the other hand, if no solution exists, a SAT solver can determine that and not return any feasible assignment. For example, $\psi = a_1 \wedge a_2 \wedge (\neg a_1 \vee \neg a_2)$ 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:

$$(a_1 \vee a_2) \wedge (a_2 \vee a_3) \wedge (a_2 \implies x_1^2 + x_2^2 \leq 10) \tag{9}$$

Here, a_1, a_2, a_3 are Boolean variables and x_1, x_2 are real variables. The third clause states that if a_2 is TRUE, there should also exist a solution to the convex constraints: $x_1^2 + x_2^2 \leq 10$. 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 $a_2 =$ TRUE and $a_4 =$ TRUE, the SMC solver employs

TABLE I: Summary of Symbols and Boolean Predicates

Symbol	Description
N	Number of Drones
T	Maximum (discrete) time duration
W	Number of discrete regions in $\overline{\mathbf{W}}$
P	Maximum power level
$\mathcal{N}(j)$	Neighborhood of region W_j including itself to account for staying in same region
$\mathbf{E}$	Set of changing stations
T_j	Time period between consecutive surveillance for region W_j
A_t	Adjacency Matrix at time t
x_t^i	The state of drone i at time t
p_t^i	The available power of drone i at time t
Symbols	**Operations**
$\wedge$	Logical 'AND'
$\vee$	Logical 'OR'
$\implies$	Logical 'Implies'
Boolean Predicate	**Implications**
Π_{jt}^i	Drone R_i is located within an unoccupied region W_j at time t
Φ_{kt}^i	Drone R_i has at-least $k-1$ amount of power and at most k amount of power at time t *i.e.,* $k-1 < p_t^i \leq k$. However, $k=0$ implies $p_t^i = 0$
l_t^{ij}	Link exists between drones R_i and R_j at time t

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 Π_{jt}^i as the Boolean predicate to represent the location of drone i, we can model the path constraints as follows.

Occupy one grid at a time (ψ_1). At time t, a drone can be in one of the unoccupied regions ($W_i \in \overline{\mathbf{W}}$); 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 (ψ_2). Each drone should start from one of the charging stations $\mathbf{E}$, *i.e.,* $\Pi_{j0}^i = \text{TRUE}$ if and only if $W_j \in \mathbf{E}$. We assume, without loss of generality[1], that all drones start from the same charging stations (say $W_1 \in \mathbf{E}$); this implies $\Pi_{10}^i = \text{TRUE}$ ($j=1$ at time $t=0$) for every drone R_i.

$$\psi_1 := \bigwedge_{t=1}^{T} \bigwedge_{i=1}^{N} \overbrace{\Big(\sum_{j=1}^{W} \Pi_{jt}^i = 1\Big)}^{\text{occupy a single region}}, \qquad \psi_2 := \bigwedge_{i=1}^{N} \Pi_{10}^i \tag{10}$$

Finish at a charging station (ψ_3). A drone should return to one of the charging stations after finishing the task.

$$\psi_3 := \bigwedge_{i=1}^{N} \overbrace{\Big(\sum_{j|W_j \in \mathbf{E}} \Pi_{jT}^i = 1\Big)}^{\text{Finish at a charging station}} \tag{11}$$

Movement continuity (ψ_4). After each time slot, a drone at region j can move to one of the regions in $\mathcal{N}(j)$, which includes the current region (Tbl. I) and its immediate neighbors (§II-A).

$$\psi_4 := \bigwedge_{t=0}^{T-1} \bigwedge_{i=1}^{N} \Big(\overbrace{\Pi_{jt}^i}^{\text{Current region is } j} \implies \overbrace{\bigvee_{k \in \mathcal{N}(j)} \Pi_{k(t+1)}^i}^{\text{Next region can be } k \in \mathcal{N}(j)} \Big) \tag{12}$$

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

First surveillance (ψ_5). At least one robot must visit each region W_j once within the first T_j time-slots.

$$\psi_5 := \bigwedge_{j=1}^{W} \Big(\overbrace{\sum_{t=1}^{T_j} \sum_{i} \Pi_{jt}^i}^{\text{Number of Surveillance within } (0,\, T_j]} \geq 1 \Big) \tag{13}$$

Periodic surveillance (ψ_6). If a region $W_j \in \overline{\mathbf{W}}$ is surveilled at time t, the next surveillance must take place between t and $t+T_j$ (for periodic coverage).

$$\psi_6 := \bigwedge_{j=1}^{W} \bigwedge_{t=0}^{T-T_j} \Big(\overbrace{\sum_{i=1}^{N} \Pi_{jt}^i \geq 1}^{\text{Surveilled at time } t} \implies \overbrace{\sum_{t'=t+1}^{t+T_j} \sum_{i=1}^{N} \Pi_{jt'}^i \geq 1}^{\text{Next surveillance at } t' \in (t,\, t+T_j]} \Big) \tag{14}$$

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

[1]We 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 (ψ_7). At time t, a drone can be in one of the power states ($p \in \{0, 1, \cdots, P\}$); we represent this by restricting $\Phi^i_{kt} = \text{TRUE}$ for exactly one value of k at time t.

Initial Power State (ψ_8). A fully charged drone starts with a power state of P *i.e.,* $\Phi^i_{P0} = \text{TRUE}$ ($j = P$ at time $t = 0$).

$$\psi_7 := \bigwedge_{t=1}^{T} \bigwedge_{i=1}^{N} \overbrace{\Big(\sum_{k=0}^{P} \Phi^i_{kt} = 1\Big)}^{\text{occupy a single power state}}, \quad \psi_8 := \bigwedge_{i=1}^{N} \Phi^i_{P0} \tag{15}$$

Power Transition (ψ_9). During operation, the battery power can only decrease since we do not consider recharging. 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:

$$\psi_9 := \bigwedge_{t=0}^{T-1} \bigwedge_{i=1}^{N} \bigwedge_{k=0}^{P} \Big(\overbrace{\Phi^i_{kt}}^{\text{Current state is } k} \implies (\sum_{l=0}^{k} \overbrace{\Phi^i_{(k-l)(t+1)} = 1}^{\text{next state can be: } k-l}) \Big) \tag{16}$$

Out of Power (ψ_{10}). If a drone R_i runs out of battery at time t *i.e.,* $\Phi^i_{0t} = 1$, the drone must be, at time t, at one of the charging stations *i.e.,* $\bigvee_{j|W_j \in \mathbf{E}} \Pi^i_{jt}$.

$$\psi_{10} := \bigwedge_{t=1}^{T} \bigwedge_{i=1}^{N} \Big(\overbrace{\Phi^i_{0t}}^{\text{Out of power}} \implies \overbrace{\bigvee_{j|W_j \in \mathbf{E}} \Pi^i_{jt}}^{\text{Located at a charging station}} \Big) \tag{17}$$

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

Connectivity. Let l^{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).

$$A_t = \begin{bmatrix} 0 & l^{12}_t & \cdots & l^{1N}_t \\ l^{21}_t & 0 & \cdots & l^{2N}_t \\ \vdots & \vdots & \vdots & \vdots \\ l^{N1}_t & l^{N2}_t & \cdots & 0 \end{bmatrix} \text{ and } \overline{A}_t = \sum_{k=1}^{N-1} A^k_t \tag{18}$$

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

$$\psi_{11} := \bigwedge_{t=1}^{T} \bigwedge_{i=1}^{N} \Big(\overbrace{\Phi^i_{0t}}^{\text{Out of power}} \implies \bigwedge_{j=1, j\neq i}^{N} \overbrace{\neg l^{ij}_t}^{\text{No communication link}} \Big) \tag{19}$$

Multi-Hop Connectivity (ψ_{12}). If two drones R_i and R_j have non-zero power at time t ($\Phi^i_{0t} = \text{FALSE}$ and $\Phi^j_{0t} = \text{FALSE}$), 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.

$$\psi_{12} := \bigwedge_{t=0}^{T} \bigwedge_{i=1}^{N} \bigwedge_{j=i+1}^{N} \Big(\overbrace{\neg(\Phi^i_{0t} \vee \Phi^j_{0t})}^{\text{Both drones have non-zero power}} \implies \overline{A}_t|_{ij} > 0 \Big) \tag{20}$$

Overall Formulation. The higher level abstraction of the persistent surveillance problem can be completely represented as: $\psi_h := \wedge_{i=1}^{12} \psi_i$.

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. (ψ_{13}) A logical solution with $\Pi^i_{jt} = \text{TRUE}$ implies that the physical location of drone R_i with state x^i_t must be within the polyhedral region W_j at time t *i.e.,* $f_{\mathbf{W}}(x^i_t)$ must satisfy the affine constraints (§II-A): $G_j f_{\mathbf{W}}(x^i_t) + h_j$. We represent this mapping between the logical and the physical space as:

$$\psi_{13} := \bigwedge_{t=0}^{T} \bigwedge_{i=1}^{N} \bigwedge_{j=1}^{W} \Big(\Pi^i_{jt} \implies G_j f_{\mathbf{W}}(x^i_t) + h_j \leq 0 \Big) \tag{21}$$

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

$$\begin{aligned} \psi_{14} &:= \bigwedge_{t=1}^{T} \bigwedge_{i=1}^{N} \bigwedge_{k=0}^{P} \Big(\Phi^i_{kt} \implies (p^i_t > k-1) \wedge (p^i_t \leq k) \Big) \\ \psi_{15} &:= \bigwedge_{t=1}^{T} \bigwedge_{i=1}^{N} \Big(\Phi^i_{0t} \implies (p^i_t = 0) \Big) \end{aligned} \tag{22}$$

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

Power Transition Constraints (ψ_{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.

$$\psi_{16} := \bigwedge_{i=1}^{N} (p^i_0 = P) \quad \psi_{17} := \bigwedge_{t=0}^{T-1} \bigwedge_{i=1}^{N} \Big(p^i_{t+1} = p^i_t - C_i x^i_t \Big) \tag{23}$$

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

$$\psi_{18} := \bigwedge_{t=0}^{T} \bigwedge_{i=1}^{N} \bigwedge_{j=i+1}^{N} \Big(l^{ij}_t \implies ||f_{\mathbf{W}}(x^i_t) - f_{\mathbf{W}}(x^j_t)||_2 \leq r_c \Big) \tag{24}$$

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:

$$\psi_{19} := \bigwedge_{t=0}^{T-1} \bigwedge_{i=1}^{N} \left(x_{t+1}^i = A_i x_t^i + B_i u_t^i \right), \ \psi_{20} := \bigwedge_{i=1}^{N} \left(x_0^i = \overline{x}_0^i \right)$$
$$\psi_{21} := \bigwedge_{t=0}^{T} \bigwedge_{i=1}^{N} \left(||x_t^i||_\infty \leq \overline{x}^i \right), \ \psi_{22} := \bigwedge_{t=0}^{T} \bigwedge_{i=1}^{N} \left(||u_t^i||_\infty \leq \overline{u}^i \right) \tag{25}$$

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]: $\{(c_{kt}^{ij}, d_{kt}^{ij}) | k \in \{1, \cdots, w\}, i, j \in \{1, \cdots, N\}, i \neq j\}$ where w is the dimension of the space. Using these variables, we introduce the following set of constraints to ensure collision avoidance.

$$\psi_{23} := \bigwedge_{t=1}^{T} \bigwedge_{i=1}^{N} \bigwedge_{j=i+1}^{N} \bigwedge_{k=1}^{w} \left(c_{kt}^{ij} \implies (f_{\mathbf{W}}^k(x_t^i) - f_{\mathbf{W}}^k(x_t^j)) \geq \epsilon \right)$$
$$\psi_{24} := \bigwedge_{t=1}^{T} \bigwedge_{i=1}^{N} \bigwedge_{j=i+1}^{N} \bigwedge_{k=1}^{w} \left(d_{kt}^{ij} \implies (-f_{\mathbf{W}}^k(x_t^i) + f_{\mathbf{W}}^k(x_t^j)) \geq \epsilon \right)$$
$$\psi_{25} := \bigwedge_{t=1}^{T} \bigwedge_{i=1}^{N} \bigwedge_{j=i+1}^{N} \left(\sum_{k=1}^{w} (c_{kt}^{ij} + d_{kt}^{ij}) \geq 1 \right) \tag{26}$$

where $f_{\mathbf{W}}^k(.)$ 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 $\psi_l := \wedge_{i=13}^{25} \psi_i$. The complete SMC encoding of the persistent surveillance problem is $\psi := \psi_h \wedge \psi_l = \wedge_{i=1}^{25} \psi_i$

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, $T_i = T \ \ \forall W_i \in \overline{\mathbf{W}}$ (§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.

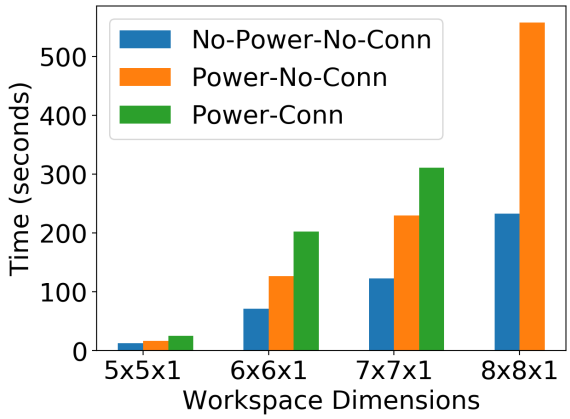


Fig. 2: The runtime of the solver for varying workspace dimensions. The missing bar for $8 \times 8 \times 1$ with **Power-Conn** implies no solution was generated.

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 $\Omega\left(T \cdot N^2\right)$ and $\Omega\left(T \cdot N \cdot P\right)$, 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 $N = 4$

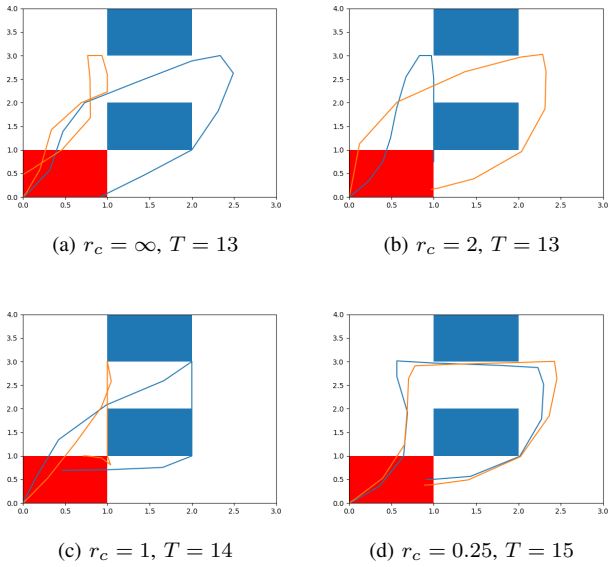
Size	Scenario	T	# of Boolean Variables	# of Real Variables	# of Boolean Constraints	# of Convex Constraints
5x5x1	No-Power-No-Conn	7	2984	672	1364	1140
	Power-No-Conn	7	3544	700	1944	1528
	Power-Conn	7	3594	700	1968	1576
6x6x1	No-Power-No-Conn	11	6600	1008	2945	2160
	Power-No-Conn	11	7540	1052	3877	2772
	Power-Conn	11	7612	1052	3913	2844
7x7x1	No-Power-No-Conn	12	10044	1092	4244	2880
	Power-No-Conn	12	11004	1140	5264	3548
	Power-Conn	12	11082	1140	5303	3626
8x8x1	No-Power-No-Conn	16	17564	1428	7325	4656
	Power-No-Conn	15	17716	1404	8141	5204
	Power-Conn	20	23482	1844	10984	7050

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: ∞, 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 ≈ 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 ≤ 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.



(a) $r_c = \infty$, $T = 13$ (b) $r_c = 2$, $T = 13$

(c) $r_c = 1$, $T = 14$ (d) $r_c = 0.25$, $T = 15$

Fig. 3: Illustration of the effect of the connectivity radius r_c for the $3 \times 4 \times 1$ scenario. The blue rectangular areas represent the obstacles and the red rectangular region represent the charging region. For $r_c = \infty$ both drones travel independently whereas with decreasing value of r_c, the drones stay much closer to each other thereby increasing the path length T.

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 ($\mathbf{W}$), 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

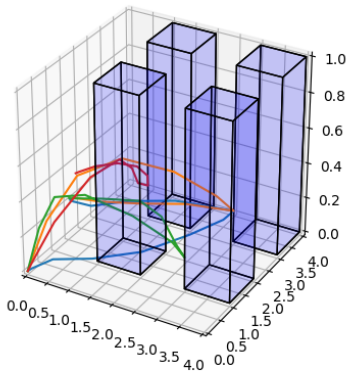


Fig. 4: A sample 3D path output with 4 drones

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

# of drones	Scenario	T	# of Boolean Variables	# of Real Variables	# of Boolean Constraints	# of Convex Constraints
1	No-Power-No-Conn	21	5635	2450	462	1458
	Power-No-Conn	21	6055	2903	483	1751
	Power-Conn	21	6055	2903	483	1751
2	No-Power-No-Conn	16	8714	3685	714	2260
	Power-No-Conn	16	9354	4371	746	2706
	Power-Conn	16	9371	4388	746	2723
3	No-Power-No-Conn	15	12339	5154	1008	3228
	Power-No-Conn	15	13239	6117	1053	3855
	Power-Conn	15	13287	6149	1053	3903
4	No-Power-No-Conn	13	14420	5921	1176	3792
	Power-No-Conn	13	15460	7029	1228	4516
	Power-Conn	13	15544	7071	1228	4600
5	No-Power-No-Conn	13	18095	7390	1470	4810
	Power-No-Conn	13	19395	8775	1535	5715
	Power-Conn	13	19535	8831	1535	5855
6	No-Power-No-Conn	13	21798	8859	1764	5856
	Power-No-Conn	13	23358	10521	1842	6942
	Power-Conn	13	23568	10591	1842	7152
7	No-Power-No-Conn	13	25529	10328	2058	6930
	Power-No-Conn	13	27349	12267	2149	8197
	Power-Conn	13	27643	12351	2149	8491
8	No-Power-No-Conn	13	29288	11797	2352	8032
	Power-No-Conn	13	31368	14013	2456	9480
	Power-Conn	13	31760	14111	2456	9872

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.

REFERENCES

[1] Y. Shoukry, P. Nuzzo, A. Balkan, I. Saha, A. L. Sangiovanni-Vincentelli, S. A. Seshia, G. J. Pappas, and P. Tabuada, "Linear temporal logic motion planning for teams of underactuated robots using satisfiability modulo convex programming," in *2017 IEEE 56th annual conference on decision and control (CDC)*. IEEE, 2017, pp. 1132–1137.

[2] K. Leahy, D. Zhou, C.-I. Vasile, K. Oikonomopoulos, M. Schwager, and C. Belta, "Persistent surveillance for unmanned aerial vehicles subject to charging and temporal logic constraints," *Autonomous Robots*, vol. 40, no. 8, pp. 1363–1378, 2016.

[3] E. Plaku and S. Karaman, "Motion planning with temporal-logic specifications: Progress and challenges," *AI communications*, vol. 29, no. 1, pp. 151–162, 2016.

[4] A. Pnueli, "The temporal logic of programs," in *18th Annual Symposium on Foundations of Computer Science (SFCS 1977)*. IEEE, 1977, pp. 46–57.

[5] P. Tabuada and G. J. Pappas, "Linear time logic control of discrete-time linear systems," *IEEE Transactions on Automatic Control*, vol. 51, no. 12, pp. 1862–1877, 2006.

[6] A. Ulusoy, S. L. Smith, X. C. Ding, C. Belta, and D. Rus, "Optimal multi-robot path planning with temporal logic constraints," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 3087–3092.

[7] M. Rungger, M. Mazo Jr, and P. Tabuada, "Specification-guided controller synthesis for linear systems and safe linear-time temporal logic," in *Proceedings of the 16th international conference on Hybrid systems: computation and control*, 2013, pp. 333–342.

[8] D. Tateo, J. Banfi, A. Riva, F. Amigoni, and A. Bonarini, "Multiagent connected path planning: Pspace-completeness and how to deal with it," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[9] T. Charrier, A. Queffelec, O. Sankur, and F. Schwarzentruber, "Reachability and coverage planning for connected agents," 2019.

[10] J. Modares, F. Ghanei, N. Mastronarde, and K. Dantu, "Ub-anc planner: Energy efficient coverage path planning with multiple drones," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 6182–6189.

[11] J. Scherer and B. Rinner, "Multi-robot persistent surveillance with connectivity constraints," *IEEE ACCESS*, 2020.

[12] ——, "Persistent multi-uav surveillance with energy and communication constraints," in *2016 IEEE International Conference on Automation Science and Engineering (CASE)*. IEEE, 2016, pp. 1225–1230.

[13] I. Shnaps and E. Rimon, "Online coverage of planar environments by a battery powered autonomous mobile robot," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 425–436, 2016.

[14] G. P. Strimel and M. M. Veloso, "Coverage planning with finite resources," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 2950–2956.

[15] Y. Shoukry, P. Nuzzo, A. L. Sangiovanni-Vincentelli, S. A. Seshia, G. J. Pappas, and P. Tabuada, "Smc: Satisfiability modulo convex optimization," in *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*, 2017, pp. 19–28.

[16] A. Biere, M. Heule, H. van Maaren, and T. Walsh, *Handbook of Satisfiability: Volume 185 Frontiers in Artificial Intelligence and Applications*. Amsterdam, The Netherlands, The Netherlands: IOS Press, 2009.

[17] P. Ghosh, J. Bunton, D. Pylorof, M. Vieira, K. Chan, R. Govindan, G. Sukhatme, P. Tabuada, and G. Verma, "Rapid top-down synthesis of large-scale iot networks," in *29th International Conference on Computer Communications and Networks (ICCCN)*, 2020.

[18] L. De Moura and N. Bjørner, "Z3: An efficient smt solver," in *Proceedings of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, ser. TACAS'08/ETAPS'08, 2008, pp. 337–340.

[19] H. Fraisse, A. Joshi, D. Gaitonde, and A. Kaviani, "Boolean satisfiability-based routing and its application to xilinx ultrascale clock network," in *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, 2016, pp. 74–79.

[20] *IBM ILOG CPLEX Optimization Studio V12.9.0 documentation*.

[21] J. P. Vielma, "Mixed integer linear programming formulation techniques," *SIAM Review*, vol. 57, no. 1, pp. 3–57, 2015.

[22] J. Banfi, N. Basilico, and S. Carpin, "Optimal redeployment of multirobot teams for communication maintenance," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3757–3764.