Trajectory Synthesis for a UAV Swarm Based on Resilient Data Collection Objectives

A H M Jakaria, *Member, IEEE,* Mohammad Ashiqur Rahman, *Senior Member, IEEE,* Muneeba Asif, *Student Member, IEEE,* Alvi Ataur Khalil, *Student Member, IEEE,* Hisham A. Kholidy, *Senior Member, IEEE,* Matthew Anderson, and Steven Drager, *Senior Member, IEEE*

***Abstract*—The use of Unmanned Aerial Vehicles (UAVs) for collecting data from remotely located sensor systems is emerging. The data can be time-sensitive and require to be transmitted to a data processing center. However, planning the trajectory for a swarm of UAVs depends on multi-fold constraints, such as data collection requirements, UAV maneuvering capacities, and budget limitations. Since a UAV may fail or be compromised, it is important to provide necessary resilience to such contingencies, thus ensuring data security. It is important to provide the UAVs with efficient spatio-temporal trajectories so that they can efficiently cover necessary data sources. In this work, we present Synth4UAV, a formal approach for automated synthesis of efficient trajectories for a UAV swarm by logically modeling the aerial space and data point topology, UAV moves, and associated constraints in terms of the turning and climbing angle, fuel usage, data collection point coverage, data freshness, and resiliency properties. We use efficient, logical formulas to encode and solve the complex model. The solution to the model provides the routing and maneuvering plan for each UAV, including the time to visit the points on the paths and corresponding fuel usage such that the necessary data points are visited while satisfying the resiliency requirements. We evaluate the proposed trajectory synthesizer, and the results show that the relationship among different parameters follows the requirements while the tool scales well with the problem size.**

***Index Terms*—UAV swarm, sensor data collection, trajectory design, formal model, resiliency.**

I. Introduction

UNMANNED Aerial Vehicles (UAVs) are increasingly being used for data collection. They are successfully used or proposed to be used in scenarios where it is highly dangerous, as well as monotonous for human observers, or in some cases, excessively costly [1]–[3]. For instance, UAVs may be used to collect data at sensitive points in industrial/critical networks, such as the smart grid bus system [2]. UAV-assisted wireless sensor networks can also be used to collect data directly from sensor nodes [4]. Sensor nodes can be planted at prespecified remote locations on the ground. UAV waypoints can be located close to the data sensors. In many cases, a set of UAVs start their flights from one or more base stations and collect necessary data at different waypoints along the trajectories before reaching endpoints. The trajectory planning in scenarios that consider multi-fold constraints of the UAVs, such as maneuvering restriction, budget limitation, and data collection requirements, is generally an NP-hard problem [5].

The trajectories of UAVs may contain predefined waypoints in the three-dimensional (3D) Euclidean space; however, the UAVs have certain limitations in maneuvering and fuel capacity while going from one point to another. The operator of such a data collection system generally employs a swarm of UAVs to collect data from the data sources collaboratively. The deployment of UAVs is often limited by a budget. The data collection is time-sensitive, as data freshness is important in practical scenarios. Moreover, the operator may want a data point to be resiliently covered, which may need multiple UAVs to be involved such that even if some of the UAVs fail to operate normally due to malfunctions or cyberattacks, some others are there to provide the data reliably. However, the operator may also require to limit the difference between the times of visiting the data point by those UAVs. Therefore, there is a great need to efficiently solve the complex problem of synthesizing necessary trajectories for a UAV swarm.

This paper addresses this need by developing Synth4UAV, a formal model-based trajectory planner for a UAV swarm. We use Satisfiability Modulo Theories (SMT) [6], [7] to implement and solve the corresponding formal model. The solution to this model synthesizes a trajectory for the UAV swarm, if there is one, satisfying all the given constraints, including data freshness and budget constraints. The evaluation results show that our tool scales well to solve problems with large spatial systems. Our major contributions in this research consist of the following:

1) We formally model the UAV trajectory planning as a constraint satisfaction problem. We model the aerial space and data point topology, UAV properties, UAV moves, maneuver limitations, fuel usage, data freshness, budget limitation, resiliency properties for data collection, and other specifications.
2) We automate Synth4UAV that encodes the formal model into SMT formulas according to the given inputs and solve these formulas using an SMT solver. We demonstrate the implemented tool in a case study.
3) We analyze various characteristics of the trajectory syn-

A. Jakaria was with Electric Power Research Institute (EPRI) in Knoxville, TN.
E-mail: ajakaria@epri.com
M. Rahman, M. Asif, A. Khalil are with the Department of Electrical and Computer Engineering, Florida International University, Miami, FL, 33174.
E-mail: marahman@fiu.edu, masif004@fiu.edu, akhal042@fiu.edu
H. Kholidy is with the Department of Networks and Computer Security, State University of New York Polytechnic Institute, Utica, NY 13502.
E-mail: kholidh@sunypoly.edu
M. Anderson, S. Drager are with the Air Force Research Lab Information Directorate, Rome, NY, 13441.
E-mail: matthew.anderson.37@us.af.mil, steven.drager@us.af.mil
Manuscript received Month 00, 2022; revised September 00, 2022.

thesis problem. We also evaluate the scalability of the tool. We build a graphical simulator for visual analysis.

Earlier, we presented a formal approach of UAV trajectory planning for performing resilient and continuous surveillance of critical power transmission lines [8]. Unlike our previous work, where the surveillance problem was simplified considering a 2D space and limited to a specific setting, we target a generic and realistic design of the data collection problem, formulating the trajectories according to the 3D Euclidean space and addressing related modeling constraints.

The rest of this paper is organized as follows: Section II presents the research problems, challenges, and related works. We discuss the framework of our developed solution in Section III. Then, we describe the formal model of the constraints and requirements in Section IV. In the following section, we present the implementation details and a case study. We discuss the evaluation results of the implemented tool in Section VI. Finally, we conclude the paper in Section VII.

II. Background

In this section, we discuss the research problem, the challenges associated with it, and the related works.

A. Research Problem

A UAV swarm is a network of UAVs that can communicate with each other (mostly following the ad-hoc wireless networking) and usually perform a common task autonomously or with instructions from operators. Maintaining appropriate trajectories for all the UAVs in a network is important to avoid mid-air collision or restricted areas (e.g., adverse territory or restricted private locations). It is also required to cover the data points (the goal positions that UAVs need to visit) within a time constraint. In this research, we propose a formal framework that plans for the trajectories of a swarm of UAVs, covering all or a minimum fraction of the data points within a given period. The coverage requirement may ensure that at least one of the available UAVs should cover each data point once or at each operator-specified time interval within the period. The resiliency specification considered in this research states that even if a certain number (k) of UAVs fail to operate or provide correct data due to technical failures or cyberattacks, the data collection objective is still fulfilled. In other words, the remaining UAVs can still reliably collect data/measurements from the necessary data points.

Fig. 1 presents two sample trajectories for two different UAVs. They collect the available data in a combined way. The figure shows the starting points for both the UAVs and the positions or waypoints from where they can collect the data. Our research problem considers different sets of points in the 3D space to represent where the UAVs can technically travel, where data is available, and where traveling is prohibited.

B. Research Challenges

A swarm of UAVs is deployed to collect data over an area from a set of predefined sensors planted on the ground. The sensors are located on the XY plane of the 3D Euclidean space with X, Y, and Z axes. There are multiple points with xyz coordinates, of which some are directly above the data sensors or near to them from where the UAVs can collect the transmitted data. A higher z value denotes points with a higher altitude. A subset of the points are restricted, which must be avoided during the flight. Designing trajectories for all the UAVs so that the budget and time restrictions are met, the UAVs avoid restricted areas, and do not collide with one another, but still achieve the data collection goal is a multi-objective problem (NP-hard). We address this problem in this paper and present a formal model-based technique for automated generation of the trajectories in the 3D space that satisfies all the constraints/requirements. We use SMT, a state-of-the-art to solve this problem and generate the trajectories in the 3D space that satisfies all the constraints and requirements.

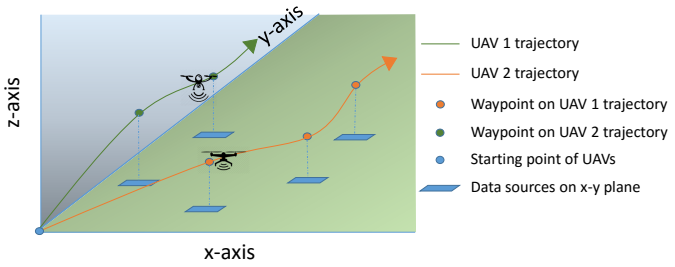


Fig. 1. Sample trajectories of two UAVs.

C. Related Works

There are several works in the literature that provides solutions to the problems of path projection and path planning for UAVs. For example, Tisdale *et al.* proposed a mechanism to perform path planning while minimizing the uncertainty in sensing missions by considering a blend of online sensor models and estimation objectives [9]. Yeung *et al.* proposed two heuristic-based techniques to find an optimal route for aerial vehicles so that the original target and waypoints are not affected [5], [10]. They also consider the flight maneuvering constraints of the UAVs and keep the deviation from the original route to a minimum while maximizing the number of sensor tasks from the ground. They also consider the time deadlines to cover the task points and maintenance of QoS. Mekala *et al.* proposed a similar method to generate the flight trajectory, which is a collection of predefined waypoints [11]. They also propose an algorithm to optimize the flight path. Liu *et al.* studied age-optimal trajectory planning in UAV-enabled wireless sensor networks [12]. The trajectory ensures that the age of the data collected from different sensor nodes is bounded by a threshold. Samir *et al.* proved that such a problem is NP-hard and proposed an algorithmic approach [13].

Beard *et al.* designed semi-autonomous UAVs and proposed their graph-based approach to path planning and trajectory generation [14]. Ceccarelli *et al.* developed a path planning model for micro UAVs to obtain video footage of a set of known targets [15]. In the presence of constant winds, they find the best selection of waypoints for achieving the reconnaissance goals. A pheromone model, where the movement of a group of UAVs is guided by pheromone, was proposed by Kuiper *et al.* [16]. This model performs well in

reconnaissance. Li *et al.* proposed an ant colony optimization (ACO) algorithm for this purpose [17].

A two-step algorithm was proposed by Bortoff to obtain a path for UAVs traveling through unauthorized sites [18]. The algorithm performed a trade-off between the shortest path for traveling and avoiding radars. A trajectory design with the aid of an iterative algorithm was proposed by Wu *et al.* where aerial base stations for a wireless communication system on the ground were mounted on the UAVs [19]. Bagherian proposed a genetic and particle swarm algorithm to find the optimal 3D path in an adverse environment [20]. They considered the terrain information and the kinematic constraints of the UAVs. 3D path planning with network jamming was investigated by Wang *et al.* [21].

Tsourdos *et al.* presented a Kripke model for UAV path planning to cope with uncertainties and decision making involved in the process of collaboratively reaching the goal while covering a certain area of interest, avoiding obstacles [22]. They modeled the collision avoidance with the help of conditions like minimum separation and non-intersection of paths at equal length. Ruz *et al.* proposed a Mixed Integer Linear (MILP)-based solution to optimize paths in dynamic environments according to a known future probability of the paths to be avoided [23]. Their ILP model can choose the most suitable trajectory among different alternatives provided fuel and time constraints. Later, the authors targeted trajectory optimization for UAVs in the presence of obstacles, waypoints, and risk zones [24]. A similar approach was proposed by Radmanesh *et al.* [25].

Sallouha *et al.* devise an analytical framework to optimize the trajectory while investigating the usage of a moving UAV for ground node localization in urban settings while considering design factors including UAV height, hovering time, journey distance, and way-point number with limited onboard energy [26]. Further, in order to create an exploratory route for UAVs in difficult situations, such *et al.* used particle swarm optimization [27]. Ahmed *et al.* use an iterative method to handle the maximization problem sub-optimally and again until it finds the best possible overall solution in order to create an energy-efficient UAV relaying communication to support the GNs for a particular UAV flight period [28].

Jakaria *et al.* presented a formal model for verification of resilient communication in a UAV swarm while maintaining safe flight trajectories for all the UAVs [29], [30]. Jameson proposed a fuel consumption algorithm for UAVs that addresses the additional fuel requirements issue in the Aviation Weather Routing Tool (AWRT) developed by the Army Research Laboratory [31]. They developed a prototype algorithm that computes the ground speeds along with the different mission segments, fuel consumed, and fuel remaining while rerouting during adverse weather conditions. An energy-efficient trajectory was proposed by Wang *et al.* [32].

Chen *et al.* optimized the trajectory of quad-rotor UAVs in collaborative environments [33]. They proposed a hierarchic optimization technique to solve the trajectory optimization problem of quad-rotor UAVs. Similar algorithmic approaches were utilized in multi-UAV swarms in several otherworks (e.g., [34], [35]). Sun *et al.* created a trajectory-tracking controller where the autopilot is involved in the feedback control loop [36]. They proposed a generalized trajectory design model using Lyapunov-based backstepping. Fadlullah *et al.* proposed a simple trajectory control algorithm for UAVs that manages the throughput of UAV-aided networks, where UAVs work as network nodes [37]. Their algorithm adjusts the center coordinates and the radius of the trajectories to reduce delay in the communication networks.

Kvarnstrom *et al.* presented a framework for UAV mission planning that combines ideas from forward-chaining planning and partial-order planning [38]. They met the requirements of centralization, abstraction, and distribution of task assignments. Bethke *et al.* proposed a health management system for UAV teams that perform persistent surveillance [39]. Their methodology for planning missions can anticipate the negative effects of various types of anomalies on future mission states and choose actions to mitigate the effects.

Reinforcement Learning (RL)-based UAV path planning solutions have become increasingly popular due to their effectiveness in unknown settings. Khalil et al. proposed a novel economic trading based RL-framework for co-operative UAV path-planning in non-dynamic environments [40]. Alpdemir et al. provided a framework that combines a fundamental RL algorithm with a specific kind of transfer learning, as well as a comprehensive probabilistic radar behavior model environment that complies with the Markov Decision Process (MDP) [41]. Papoudakis et al. established the notion that the performance of deep RL is highly dependent on the environment properties by evaluating and comparing three different classes of multi-agent deep RL algorithms [42]. To deal with the issue of erroneous predictions in path-planning problems by deep RL at the early stage of training, Xie et al. proposed an action selection strategy that integrates the present reward value with the long-term reward value [43]. To analyze the collision probability of UAVs in a path-planning problem, Wan et al. proposed an RL-based motion control framework that leverages actor-critic architecture [44]. It performed dual-channel roll and speed control by anticipating the desired steering angle. To support the time and energy efficient video processing in a UAV surveillance system, Apostolopoulos et al. present a unique data offloading architecture by utilizing the capabilities of Fully Autonomous Aerial Systems (FAAS) and Mobile Edge Computing (MEC) [45].

However, none of these works present the path planning for resilient coverage of data points. We consider the resiliency of the planned trajectories, which ensures a certain coverage of the data sources with a certain frequency of timely visits of multiple UAVs. We solve an NP-hard problem [46], which involves trajectory synthesis while satisfying multi-objective constraints involving a comprehensive set of parameters such as budget and maneuvering limitations and operator requirements, such as resilient coverage.

III. SYNTH4UAV FRAMEWORK

We develop Synth4UAV, a formal model-based tool that creates a trajectory plan for each UAV of a UAV swarm, where a trajectory is a sequence of waypoints through which

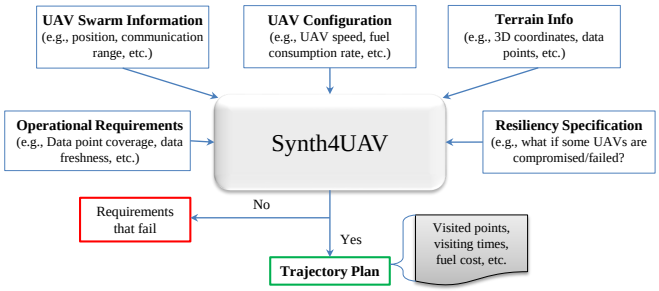


Fig. 2. A formal framework for trajectory planning.

the UAV will fly. The trajectory depends on current positions of the UAVs, their velocity, climb angle, and turn angle, etc. Different UAVs may have different speeds. The UAVs can start from different base stations. For a time period (represented as a series of time steps), the trajectory is modeled. Within a unit time step, the UAVs will not alter their flight parameters. The output trajectory is a set of coordinates in the 3D space, along with the climb angle and turn angle in each step. We model the fuel consumption where a UAV starts traveling with a certain amount of fuel and returns to a base station for refueling.

Fig. 2 presents the framework of Synth4UAV. The framework takes necessary inputs, namely, the UAV swarm and individual UAV properties, terrain information, data coverage requirements, and resiliency specifications. The swarm information includes the number of UAVs, initial positions of the UAVs, as well as their speeds and directions. Data collection requirements specify the points to visit and the characteristics/constraints associated with these visits. According to the inputs, the trajectory planning is formally modeled, which is eventually solved using SMT logic/solver, to identify the trajectory plan (*e.g.*, the flight angles), satisfying the data collection constraints. Our trajectory planning considers necessary invariants. For example, the climb and turn angles must be within the (practical) capacity of a UAV.

In Synth4UAV, we formalize the trajectory planning as a decision problem, namely constraint satisfaction problem (CSP). A CSP is generally defined using a set of constraints represented using a number of variables and the relations among them. We solve a CSP by seeking an assignment of values to all of the variables (completeness) that satisfies all the constraints (consistency). In this work, we apply SMT logic [6] to encode our proposed trajectory planning CSP. SMT provides different first order logic-based formulas (*e.g.*, arithmetic, uninterpreted functions, bit-vectors, etc.) to solve decision problems efficiently [47]. In particular, we use Z3 [48], an efficient SMT solver, to solve the formulated CSP.

IV. Trajectory Planning Problem Model

In this section, we design the UAV trajectory planning CSP in detail by formally modeling the constraints. Table I lists several variables used in the modeling.

A. Modeling Preliminary

To model the trajectory planning for the UAVs, we consider a set of predefined points (waypoints) in the 3D space. Each point is considered to have Cartesian (xyz) coordinates in the Euclidean space. The UAVs start and end their flights at certain points, which are generally base stations. Between the starting and ending points, the UAVs may visit several other points. All these points constitute the trajectories for all the available UAVs. Our basic intention is to automatically generate these trajectories such that certain constraints and requirement goals are met. Here we formally model the route constraints, as well as the requirements of the users of the implemented tool. We make sure that the points containing the data sensors are covered with certain resiliency specifications.

B. Modeling UAV Movement

If $M_u^{\hat{p};p}$ denotes that a UAV u from a set of UAVs U moves from a point $\hat{p}$ to another point p, u has already visited point $\hat{p}$ (denoted by $V_u^{\hat{p}}$). Besides, it must be possible for a UAV to travel from point $\hat{p}$ to p, i.e., there should be a link between points $\hat{p}$ and p (denoted by $L_p^{\hat{p}}$). Hence, $M_u^{\hat{p};p}$ is formalized as follows:

$$M_u^{\hat{p};p} \quad ! \quad V_u^{\hat{p}} \wedge L_p^{\hat{p}} \tag{1}$$

We define the Boolean variable $D_p^{\hat{p}}$ as true only if the distance between point $\hat{p}$ and p ($dist_p^{\hat{p}}$) is within a certain threshold, $d_{u;th}$. The threshold is specified by the properties of UAV u.

$$D_p^{\hat{p}} \quad \$ \quad dist_p^{\hat{p}} \quad d_{u;th} \tag{2}$$

The UAVs have certain maneuvering constraints. We consider the restrictions on the turn angle and the climb angle in the 3D space [20]. The turn angle is the angle that a UAV has to turn with respect to the existing direction on the horizontal plane. We first calculate the angle for a UAV for going from $\hat{p}$ to p and the existing angle which the UAV is following to reach $\hat{p}$ from p^0. Then we calculate the deviation which the UAV has to turn. Similarly, we calculate the climb angle on

TABLE I
Important Variables/Notations

Notation	Definition
U	Set of number of UAVs required for the data collection operation
S	Set of Data points in the given trajectory
	Turn angle when moving from one point to another
	Climb angle when moving from one point to another
$M_u^{\hat{p};p}$	Boolean representing movement of UAV u from point $\hat{p}$ to p
V_u^p	Boolean representing that UAV u has visited point p
$L_p^{\hat{p}}$	Boolean representing the possibility of traveling from point $\hat{p}$ to p
C_u^p	Real expression denoting the cost of fuel for UAV u for reaching point p
T_u^p	Real expression denoting the time taken by UAV u to reach point p since its starting point
H_u^p	Integer expression denoting the time a UAV u hovers over point p
$D_p^{\hat{p}}$	Boolean representing the distance between $\hat{p}$ and p is within threshold
$p^0;\hat{p};p$	The deviation of turn angle to reach p from $\hat{p}$, given a UAV reached $\hat{p}$ from p'
$p^0;\hat{p};p$	The deviation of climb angle to reach p from $\hat{p}$, given it reached $\hat{p}$ from p'

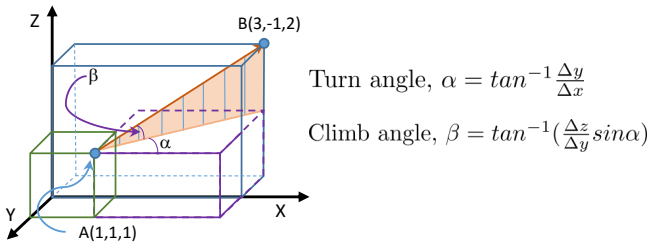


Fig. 3. Turn and climb angles while moving from one point to another.

the vertical plane. Fig. 3 shows an example scenario of moving from one point to another in the 3D space. is the turn angle for moving from point A to B, provided the direction of a UAV at A is parallel to the X axis. is the climb angle. Fig. 3 also provides the formulas for calculating the angles based on the horizontal and vertical movements.

The deviation of turn angle to reach p from $\hat{p}$, given a UAV reached $\hat{p}$ from p^0 is represented by $^{p^0;\hat{p};p}$, while the deviation of climb angle to reach p from $\hat{p}$, given it reached $\hat{p}$ from p^0, is denoted by $^{p^0;\hat{p};p}$. The deviations must be within certain thresholds ($^{t}_{th}$ and $^{c}_{th}$). If this is satisfied, then we can say that there is a link that a UAV can move.

$$L^{\hat{p}}_{p} \quad ! \quad D^{\hat{p}}_{p} \wedge (\ ^{p^0;\hat{p};p} \quad ^{t}_{th}) \wedge (\ ^{p^0;\hat{p};p} \quad ^{c}_{th}) \tag{3}$$

Similarly, if a UAV u moves from p^0 to $\hat{p}$ followed by p, then the deviation of the turn angle and climb angle to move from the path p^0 to $\hat{p}$ and the path $\hat{p}$ to p should be within user-defined thresholds.

$$M^{p^0;\hat{p}}_{u} \wedge M^{\hat{p};p}_{u} \quad ! \quad (\ ^{p^0;\hat{p};p} \quad ^{t}_{th}) \wedge (\ ^{p^0;\hat{p};p} \quad ^{c}_{th}) \tag{4}$$

A UAV visits a particular point from one of its neighboring points. If UAV u has visited the point p, that means it has traveled from one of the neighboring points, $\hat{p}$.

$$V^{p}_{u} \quad ! \quad \bigvee_{\hat{p}} M^{\hat{p};p}_{u} \tag{5}$$

The starting point S is the first point on the trajectory of any UAV. If the current point that a UAV u is visiting is the starting point, then it has not come from any other point $\hat{p}$.

$$\delta_{\hat{p}}(p = S) \quad ! \quad : M^{\hat{p};p}_{u} \tag{6}$$

If a UAV u has reached the destination point D, it should not move to anywhere from there. This is represented by the following equation:

$$\delta_{p}(\hat{p} = D) \quad ! \quad : M^{\hat{p};p}_{u} \tag{7}$$

A UAV can travel to a certain point only from another point (not from multiple points). If a UAV u has moved from $\hat{p}$ to p, then it should not have come from any other point $\hat{p}^0$.

$$M^{\hat{p};p}_{u} \quad ! \quad \bigwedge_{\hat{p}^0 \neq \hat{p}} : M^{\hat{p}^0;p}_{u} \tag{8}$$

Similarly, from any particular point, a UAV can only travel to one point and not multiple points. If UAV u goes from $\hat{p}$ to p, then it must not have gone to any other point p^0 from $\hat{p}$. This is formally represented as the following:

$$M^{\hat{p};p}_{u} \quad ! \quad \bigwedge_{p^0 \neq p} : M^{\hat{p};p^0}_{u} \tag{9}$$

A UAV u must visit the starting point S and the destination point D. This is true for all the collaborating UAVs in the network and is presented by the following:

$$\delta_{u}(V^{S}_{u} \wedge V^{D}_{u}) \tag{10}$$

Some of the points where the UAVs can go are in close proximity to the data collecting sensors. These points need to be visited by the UAVs so that they can collect the data. We assume the communication between the UAVs and sensor nodes can be done via any personal area network (PAN). We assumed Bluetooth, Zigbee, Wi-Fi, etc., as feasible communication systems. We consider that a UAV can collect information from a data point by moving in close proximity to the point. The proximity however depends on the communication system utilized, and a handshake protocol must be executed between the UAV and data point. We leave the technical details out since it is out of the scope of this trajectory modeling. If S is the set of all the points that have data points, then at least a threshold (v_{th}) fraction of all the data points must be visited by the UAVs during their journeys.

$$\frac{\sum_{u;p2S} V^{p}_{u}}{|S|} \quad v_{th} \tag{11}$$

Some of the points in the 3D space may have to be avoided by the UAVs while making their journeys, for these points may currently reside in restricted zones. If F denotes the set of all the forbidden points that should be avoided, then no UAV should visit them.

$$\bigwedge_{p2F} : V^{p}_{u} \tag{12}$$

C. Modeling UAV Fuel Consumption and Cost

While calculating the fuel consumption, we not only consider the distance but also consider the turn and climb angles for going from a certain point to another. The direction of the turn angle has no effect on fuel consumption, but the direction of the climb angle does affect the amount of fuel consumption. If a UAV is going up against gravity, then it has to burn more fuel. On the other hand, if it is going down, then less fuel is required. We consider a positive climb angle if the UAV is going up with respect to its current direction, while it is counted as negative if it is doing downwards. If $d^{\hat{p}}_{p}$ is the distance between point $\hat{p}$ and p, $^{\hat{p};p}_{u}$ denotes the turn angle of UAV u for going to p from $\hat{p}$, $^{\hat{p};p}_{u}$ is the climb angle to move, and r_u is the fuel consumption rate (*e.g.*, mpg) of a UAV u, then the required fuel, $f^{\hat{p};p}_{u}$, is calculated as follows:

$$f^{\hat{p};p}_{u} = \frac{d^{\hat{p}}_{p} + k_1 \quad j \ ^{\hat{p};p}_{u} j + k_2 \quad ^{\hat{p};p}_{u}}{r_u}$$

Here, k_1 and k_2 are simple constants that regulate the impact of turn and climb angles, respectively, of the UAV. The fuel usage does not depend on the direction of the turn angle $^{\hat{p};p}_{u}$, while it directly depends on the direction of the climb angle $^{\hat{p};p}_{u}$. If the UAV moves to a point located at a lower altitude than the source point, it travels with gravity. As a result, there is less usage of fuel. On the other hand, if it has to travel against gravity, then fuel usage increases. We assume that the UAVs are of the same type, *e.g.*, fixed-wing or quadcopter. The

assumption is that, depending on the type, using the properties, it is possible to retrieve the factors and constants that determine the fuel consumption of this certain type.

If c_e is the cost of per unit fuel, then the cost, $c_u^{p;p}$, for a UAV for moving from p to p is calculated as follows:

$$c_u^{p;p} = f_u^{p;p} \quad c_e$$

The cumulative cost for traveling to point p for UAV u is the cost up to the previous point p, plus the cost to move from p to p, $c_u^{p;p}$.

$$M_u^{p;p} \quad ! \quad C_u^p = C_u^p + c_u^{p;p} \tag{13}$$

If S is the starting point, the cost at the starting point for any UAV u should be zero.

$$C_u^S = 0 \tag{14}$$

D. Modeling UAV Travel Time

We assume a constant speed for each UAV while making its flight from a point to another particular point. $t_u^{p;p}$ is the time taken by UAV u to move from p to p if s_u is the speed of the UAV. It is calculated as follows:

$$t_u^{p;p} = \frac{d_p^p}{s_u}$$

T_u^p is the time taken by UAV u to reach up to p from the starting point. H_u^p is the time of hovering of UAV u at a previous point p. We consider this hovering time to be either 0 or 1, for simplicity. A UAV may or may not hover over a point depending on the requirement of avoiding a collision.

$$M_u^{p;p} \quad ! \quad T_u^p = T_u^p + t_u^{p;p} + H_u^p \tag{15}$$

$$(H_u^p = 0) \_ (H_u^p = 1) \tag{16}$$

If t_{st} is the start time, then it is the time for all the UAVs at the starting point S.

$$T_u^S = t_{st} \tag{17}$$

The following constraint ensures that no two UAVs visit the same point at the same time. This restriction is required to avoid collision between any two UAVs.

$$V_u^p \wedge V_{u^0}^p \quad ! \quad T_u^p \neq T_{u^0}^p \tag{18}$$

In fact, there should be a threshold time difference between the times of visit of any two UAVs to a certain point. If both UAVs u and u^0 visit the same point p, then there must be some time difference between their visits to avoid mid-air collision.

$$V_u^p \wedge V_{u^0}^p \quad ! \quad T_u^p \quad T_{u^0}^p \quad \hat{t}_{th} \tag{19}$$

E. Modeling User Requirements

One of the objectives of the trajectory planning model is to keep the budget within a limit. We consider the budget or cost limit for each UAV to be b_c, while b_t is the allowed time or threshold time within which the flights need to be performed. The cost of the fuel by UAV u should be less than a user-specified budget, b_c.

$$C_u^D \quad b_c \tag{20}$$

Each UAV should reach the destination point from the starting point within a certain time limit, b_t.

$$T_u^D \quad b_t \tag{21}$$

Again, a point cannot be reached by the UAVs by taking more cost than the budget or the time allows. If a UAV visits point p, then the cost for reaching up to that point should be less than the budget.

$$V_u^p \quad ! \quad C_u^p \quad b_c \tag{22}$$

The time taken in reaching any point p should be within the time limit.

$$V_u^p \quad ! \quad T_u^p \quad b_t \tag{23}$$

F. Modeling Resiliency Specifications

Our trajectory modeling considers resiliency with respect to attack/contingency scenarios. The satisfaction of the resiliency specification provides resilient trajectory plans. The resiliency specification is stated such that at least a certain percentage ($r_{th}\%$) of the data points are (securely) covered by a set of UAVs (Equation 27), even if a certain number (k) of UAVs among them fail to operate or are compromised.

We abstract the resiliency requirement by introducing constraints on the data collection requirements. If p is a data point ($p \in S$) and it satisfies the k resiliency data collection, then at least $k+1$ number of UAVs must visit (collect data at) this point during their travel to the destination. That is, to provide a single ($k = 1$) UAV failure resiliency to a data point, two ($k+1$) UAVs will need to visit the point. R_c^p specifies if point p is under k resilient data collection.

$$8_{p2S} R_c^p \quad ! \quad \sum_u V_u^p \quad k+1 \tag{24}$$

The resilient trajectory design often needs to satisfy further requirements. Such a requirement can be data freshness or closeness constraint that ensures that the times at which data is collected by $k+1$ UAVs at a data point should not be apart from one another more than a threshold value (say t_{th}), so that the collected data records can be verified for any discrepancy due to an unexpectedly long time difference. Such a constraint may differ for different data points based on the data change frequency (which may depend on the data type, location, etc.) and/or the criticality of the data. This constraint ensures that even if k UAVs fail to visit a point p, there is at least a UAV u, which along with another UAV u^0 will cover this point. This constraint is formalized as below:

$$8_{p2S} R_t^p \quad ! \quad \min_{u \quad u^0;u \neq u^0} \sum \hat{V}_{u;u^0}^p \quad k+1 \tag{25}$$

TABLE II
SAMPLE CODE

```
(assert (and Visit_1_1 Visit_1_18))
(assert (and Visit_2_1 Visit_2_18))
...
(assert (not Visit_2_6))
(assert (not Visit_2_14))
...
(assert (=> Travel_1_1_2 (and Visit_1_1 Links_1_2)))
(assert (=> Travel_1_1_2
     (= T_1_2 (+ T_1_1 TTmp_1_2 (to_real Hover_1_2)))))
(assert (=> Travel_1_1_2 (= P_1_2 (+ P_1_1 PTmp_1_2))))
...
(assert (=> (and Visit_1_2 Visit_2_2) (or
     (>= (- T_1_2 T_2_2) 1.0) (>= (- T_2_2 T_1_2) 1.0))))
...
(assert (<= T_1_18 2000.0))
(assert (<= P_1_18 6000.0))
...
```

Here, $\hat{V}^{p}_{u;u^0}$ specifies that if UAVs u and u^0 visit point p, then the visiting times (u earlier than u^0, required for ordering with respect to the visiting time) are no more than the threshold time apart. $\hat{V}^{p}_{u;u^0}$ is defined as follows:

$$\hat{V}^{p}_{u;u^0} \;!\; V^{p}_{u} \wedge V^{p}_{u^0} \wedge (T^{p}_{u} \quad T^{p}_{u^0}) \wedge (T^{p}_{u^0} \quad T^{p}_{u} \quad t^{p}_{th}) \quad (26)$$

We need to make sure that at least a certain percentage (k_{th} %) of the data points are covered by the UAVs such that the points follow the k resiliency property.

$$(\frac{\sum_{p} R^{p}_{c}}{|S|} \quad r_{th}) \wedge (\frac{\sum_{p} R^{p}_{t}}{|S|} \quad k_{th}) \quad (27)$$

V. IMPLEMENTATION AND A CASE STUDY

In this section, we present the implementation details of our proposed model. We also present a detailed case study for further explaining how the model works.

A. SMT Encoding and Target Variables

We implement Synth4UAV by encoding the formal model, presented in Section IV, into SMT formulas [47]. In this encoding purpose, we use Z3, an efficient SMT solver [49]. The solver checks the verification constraints and provides a satisfiable (SAT) result if all the constraints are satisfied. The SAT result provides the value assignments to the parameters of the model. According to our objective, we are mainly interested in the assignments to the following variables: (i) the decision variable referring to whether a UAV u has visited a particular point p, V^{p}_{u} and (ii) the variable denoting whether a UAV u travels from a point $\hat{p}$ to p, $M^{\hat{p};p}_{u}$. We can retrieve the trajectories for all the UAVs with the values of these two variables. The assignments of the variables are printed in an output text file. The values of other variables such as T^{p}_{u} can provide information on which point was traveled at what time, while C^{p}_{u} can state the fuel cost associated with traveling up to certain waypoints on the trajectory.

Table II presents a tiny excerpt (a short listing) of the z3 code generated by the trajectory generation tool. It shows a few of the expressions that assert the constraints according to our CSP model. These expressions are generated for the satisfiable case study shown in Section V-D. For example, the first line in the presented code has two Boolean variables `Visit_1_1` and `Visit_1_18`, which denote that UAV 1 should visit both points 1 and 18.

Throughout the model of the network synthesis, for simplicity, we have used the sum of Boolean variables several times for formally modeling the constraints. To encode the summation of Boolean variables, we have converted them to integer variables (0 or 1).

B. Optimal Trajectory Synthesis

The synthesis result represents a comprehensive trajectory plan for the UAVs to perform the intended tasks. There is usually more than one satisfiable solution that can satisfy the constraints. These solutions (model states) will provide different costs and times for the overall process, although all of them will be within the budget (b_c) and allowed time (b_t). We can choose the most cost- or time-efficient trajectory plans among all alternative satisfiable models for the same set of constraints. We develop Algorithm 1, a binary search-based procedure, to find such optimal solutions. The algorithm starts with the minimum (b^{min}_{c}) and maximum (b^{max}_{c}) possible values of the budget, updates them based on the solver's outcome, and accordingly sets the budget constraint (b_c) to find the optimal trajectory.

Finding the optimal solution naturally requires more time than looking for a single satisfiable answer. The algorithm requires several invocations of the solver. If an invocation confronts UNSAT result, it often takes a longer time. The total time complexity of the framework consists of the time required for one model execution and the time required for the budget-based solution optimization. Let the time required for one model execution be T_M and the time needed for the budget-based solution optimization be T_O. Then, the overall time complexity of the framework will be $O(T_M \quad T_O)$. Employing the knowledge of binary-search complexity analysis [50], T_O can be written as $\log_2 D$, where D is the difference between b^{max}_{c} and b^{min}_{c} which control the length of the binary search array. T_M is the time taken by the SMT solver to synthesize the trajectory by satisfying the given formal constraints. State-of-the-art SMT solvers translate the SMT constraints into a set of propositional clauses and employ sophisticated

Algorithm 1 Find Optimal Solution based on Budget

1: $b^{max}_{c} := b_c$
2: $b^{min}_{c} := 0$
3: **if** Solver returns SAT **then**
4: Get the model state, $\mathcal{M}$
5: $b^{max}_{c} := \mathcal{M}.C^{d}_{u}$
6: $I = 1$ ▷ Number of model executions
7: **repeat**
8: $b_c := (b^{min}_{c} + b^{max}_{c})$ / 2
9: **if** Solver returns SAT **then**
10: Get the model state, $\mathcal{M}$
11: $b^{max}_{c} := \mathcal{M}.C^{d}_{u}$
12: **else**
13: $b^{min}_{c} := b_c$
14: **end if**
15: $I := I + 1$
16: **until** $((b^{max}_{c} - b^{min}_{c}) \approx 0) \parallel (I \geq I^{max})$
17: **end if**

and optimal approaches like the Davis-Putnam-Logemann-Loveland (DPLL) algorithm to solve the satisfiability problem. DPLL utilizes backtracking and works by selecting a variable and apportioning a truth value to it, hence, streamlining the formula. If the streamlined formula is satisfied (true), the original formula will also be satisfied. Otherwise, the same recursive check is performed by assigning the alternate value to the variable. Since the DPLL algorithm branches out in a binary fashion, the time complexity is 2^C, where C is the number of unknown variables (in the constraints) to which the solver has to assign values [6]. In the case of Synth4UAV the number of unknown variables varies with that of UAVs *(n)* and waypoints *(m)*. Thus, C will be of the order of *(n)* and *(m)*. Hence, we can re-write the T_M complexity as 2^{nm}. Moreover, to amortize the searching cost and to find a more time-efficient solution, DPLL employs a pruning strategy based on falsified/conflicting clauses to navigate the search space. Hence, the obtained cost is often much smaller than the worst-case $O(2^{nm})$ scenario [51]. However, the user can also control the number of executions of the solver (I^{max}) and generate a sub-optimal solution in a shorter time.

C. Trajectory Smoothing

The synthesized trajectories can be smoothed by applying a 3D Bezier curve interpolation [46]. Careful studies need to be done to make sure that the smooth trajectories remain adequately close to the data sources so that the UAVs and the sources are within range of each other. While this is out of the scope of this research, we leave it for our future works.

D. An Example Case Study

UAVs can be used to collect data from sensors deployed in remote points, which is essentially an alternative to the use of UAVs of a surveillance purpose. This is efficient technology when it is not feasible (either technically or economically) to deploy necessary communication infrastructures. In the cases where sensors cannot report data by themselves to the data center due to the lack of proper infrastructure or energy-saving purposes, UAVs provide convenient options to collect data from them. Since offline path-planning is a viable option in many patrolling and data harvesting problems, our framework can be utilized in many of these application domains. The execution of the tool can be done in any computer system with standard configuration, which can be a system already part of the application infrastructure or can be an additional device added to the infrastructure. We need to ensure that a certain percentage of the points with sensors are covered by the set of UAVs, even if a certain number of UAVs fail while in flight due to malfunction or cyberattacks, etc.

In this case study, we consider 30 waypoints in the 3D space, of which 15 contain data sources. Table III lists the example case study inputs to Synth4UAV. The inputs include the number of waypoints, the Cartesian coordinates of the points in the three dimensional space, the number of UAVs that will be considered, along with their velocities, mileages, etc. For example, (100, 100, 0), (500, 1000, 100), and (500, 2000, 100) are some of the points that the UAVs can visit.

TABLE III
THE INPUT TO THE CASE STUDY

```
# Number of waypoints
30
# UAV x-coordinates
100 500 500 1000 1000 2000 2000 2000 2500 3000 3000 3000 4000 4500
4500 5000 5500 6000 1000 2700 2000 500 2000 3500 5000 4000 500 5500
1500 3500
# UAV y-coordinates
100 1000 2000 500 2000 500 1500 2500 2000 1000 1500 2500 1500 1000
2500 2000 1500 2000 3000 3400 4000 3500 2000 3500 3500 3000 3000 500
3500 4000
# UAV z-coordinates
0 100 100 200 250 200 150 150 200 200 200 200 200 200 200 200 150 200
200 200 200 200 200 200 200 200 150 100 50 0
# Number of UAVs
5
# UAV velocities (m/s)
50 48 49 51 52
# UAV mileage (mpg)
10 12 11 10 12
# Initial UAV angles (in degrees w.r.t. with x axis)
0 0 0 0 0
# Turn angle threshold
90
#Climb angle threshold
30
# Source, Destination
1 1 1 1 1 18 18 18 18 18
# Forbidden points
4
6 14 19 30
# Points with data sources
15
2 3 5 9 10 13 20 12 16 18 29 8 26 7 22
# Data coverage threshold (%)
80
# Data freshness threshold (sec)
20
# k  resiliency
3
# Resilient data coverage threshold (%)
70
# Fuel price ($/gal)
3
# Time constraint (sec)
2000
# Cost constraint ($)
6000
```

The source and destination points, the forbidden points which the UAVs should avoid, and the points that are on top or in close proximity of the data sources are provided as inputs. Here, point 1 is the starting point and point 18 is the ending point for all the UAVs. There are four forbidden points (*i.e.*, 6, 14, 19, and 30) that should be avoided by the UAVs while flying. The data coverage threshold specifies percentage of the data points that need to covered by the UAVs collectively. The required resiliency specifications are also provided in the input file. The time constraint or the deadline and the available cost constraint or the budget are also supplied.

We consider 5 UAVs in this example. The average velocity, mileage, and angle with X axis (at the beginning) of the UAVs are also provided. The restrictions of turn and climb angles (in degrees) of the UAVs while moving from one point to another are given in the input file. We consider the deviation of the moving direction of the UAVs. That is, after a UAV reaches a particular point, how much it can deviate from its current trajectory both vertically and horizontally in order to move to the next point.

A Satisfiable Case: The data coverage threshold is selected to be 80% in this case study. That means, at least 80% of the data points must be covered by the UAVs while they are

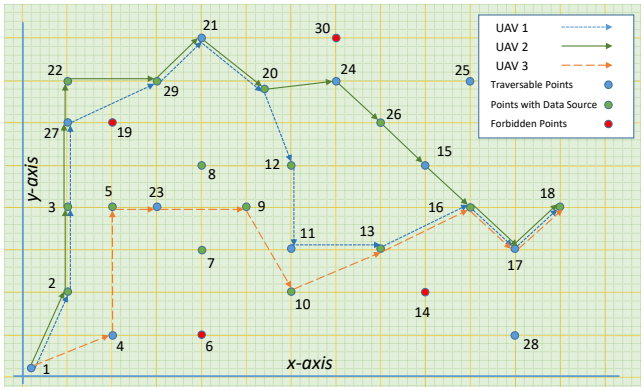


Fig. 4. Generated trajectories of three of the five UAVs on XY plane visiting waypoints near data sensors.

TABLE IV
THE OUTPUT TO THE CASE STUDY

#Required verification time: 336.77			
#We have a solution			
1	1	0	0
1	2	20.79898987	1
1	3	41.79898987	0
1	27	62.82397427	0
…	…	…	…
1	17	194.898209	0
1	18	209.0756559	0
…	…	…	…
4	1	0	1
4	4	19.7056384	0
4	5	38.7084393	1
…	…	…	…
#All trajectories:			
UAV	Src	Dest	
1	1	2	
1	2	3	
1	3	27	
1	27	29	
…	…	…	
1	21	20	
1	20	12	
1	12	11	
…	…	…	
4	1	4	
4	4	5	
…	…	…	

making their journeys. The data freshness threshold specifies the time (in seconds) within which a certain number of UAVs must visit a data point to ensure the freshness of the data. At the beginning of this case study, we consider this to be 20 sec. At this point, we consider the value of k to be 2, while the resilient data coverage is chosen to be 70%. This means there should be at least 70% of the data points, such that they are covered by at least $k = 2$ UAVs every 20 seconds. We also need to ensure that a minimum of k UAVs cover at least 70% of the data points.

The price of fuel is provided in the input file. The available budget and the total available time within which the UAVs need to finish their task by traveling from the start point to the end point are also provided. For the first run of the tool in this case study, we consider a budget of $6000, and the total allowed time is chosen to be 2000 sec.

Now we try to solve this multi-objective problem in order to find the trajectories of the UAVs. The solution should provide the paths for each of the UAVs that satisfy all the conditions mentioned in Section IV. In this case, the solver gives a SAT answer, *i.e.*, there is a solution to (a complete and consistent state of) the CSP given the inputs. We print the necessary

TABLE V
COMPUTATIONAL COST

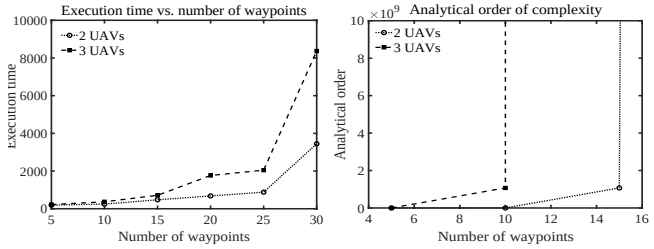
Number of Waypoints	Experimental Time (ms)		Analytical order of Complexity		CPU Usage (%)
	2 UAVs	3 UAVs	2 UAVs	3 UAVs	
5	191.63	221.75	1024	32768	65.32
10	249.40	369.33	1.04E+6	1.07E+9	74.81
15	474.75	713.56	1.07E+9	3.51E+13	75.61
20	680.01	1765.55	1.09E+12	1.15E+18	84.54
25	878.22	2046.24	1.12E+15	3.77E+22	86.38
30	3447.02	8369.96	1.15E+18	1.23E+27	96.14

outputs to a file. Table IV presents the contents of the output file generated by the trajectory generation tool. The printed results include the times at which the UAVs visit each of the points on their trajectories. For example, UAV 1 is at point 1 at the beginning (0 sec), it is at point 2 at 20.79 sec, at point 3 at 41.79 sec, and so on. If a UAV needs to hover over any point to avoid a mid-air crash, the time of hovering is also printed on the output file. The output also provides the trajectories. For example, UAV 1 visits the following points during its journey in sequence: 1, 2, 3, 27, 29, 21, 20, 12, 11, 13, 16, 17, and 18. Similarly, the trajectories of all other UAVs are printed. Fig. 4 presents a top view (*i.e.*, a 2D view on the XY plane) of the overall trajectory plan for three of the UAVs. The data points are colored green, while the forbidden ones are red. All other points are colored blue. All of the UAVs start from waypoint 1 and end their flight at waypoint 18. During their flights, they cover the points that have the data sources with the required resiliency.

An Unsatisfiable Case: For the next run, we increase the value of k to 3. In this scenario, the solver provides an UNSAT result, which means there is no satisfiable solution to the problem, and there cannot be any trajectory for the UAVs unless some conditions are modified. Increasing the budget, number of UAVs, etc., or decreasing the data coverage threshold may result in a SAT result.

E. Computational and Implementation cost

We used an 11th Gen Intel(R) Core (TM) i7-1195G7 @2.90GHz processor with 16 GB memory to solve the formal model to synthesize trajectories for 5, 10, 15, 20, 25 and 30 waypoints, respectively. For instance, for synthesizing a trajectory for 5 waypoints (m) for 2 UAVs (n) at its optimal budget, our model takes 191.63ms to devise a satisfiable



(a) (b)

Fig. 5. (a) Framework execution time and (b) analytical order of complexity w.r.t. the number of waypoints for 2 and 3 UAVs, respectively. Execution time in (a) is seen to be well within the worst-case scenario of analytically computed order of complexity i.e., 2^{nm} as depicted in (b).

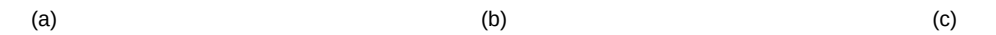


Fig. 6. (a) The coverage of the data points ($p_2(S)$) w.r.t. the budgeted fuel cost for 2 and 4 UAVs, respectively, (b) the time for coverage of data points ($p_2 \in S$) w.r.t. the number of UAVs (U) for 40% and 60% coverage, respectively, and (c) the minimum number of required UAVs w.r.t. the required coverage in the percentage of the data points ($p_2(S)$) for 50 and 90 points, respectively.

trajectory while consuming 65.32% of the CPU usage. It is worth mentioning that CPU usage is a percentage of the total time taken to execute the model. Hence, the CPU time taken to run the model for 5 waypoints can be computed as 65.32% of 191.63ms which is 125.17ms of CPU time. For a larger m, e.g., in the case of 10 waypoints, the model takes a longer time (249.40ms) and a higher CPU usage (74.81%). We also measured execution time and CPU usage for the cases of 15, 20, 25, and 30 waypoints, as presented in Table IV and Fig. 5(a). We found that the execution time increases exponentially with the number of waypoints. It can also be observed that the execution time increases for a larger number of UAVs. The analytically computed orders (in worst-case scenarios) for all the cases are also shown in Table V and the corresponding growth curve in Fig. 5(b). It can be easily be seen by comparing by comparing Fig. 5(a) and Fig. 5(b) that the real-execution grows much slower than the theoretical order of growth.

Furthermore, the implementation cost to execute a mission in practice will include the cost of the UAVs (initial cost) plus the cost of the fuel/charge required for the UAV trips. The initial cost will depend on the number of needed UAVs, which will be the one at which the optimal solution or SAT is found. Getting fewer UAVs than the optimal one will result in an UNSAT result.

VI. Evaluation

Here, we evaluate Synth4UAV to discern the relationship between different synthesis factors, along with its scalability.

A. Methodology

We create various synthetic problems in a 3D Cartesian space, where there are different points indicating the probable points for UAVs to fly. We randomly specify some data points (50-80% of all the points), which must be covered by the UAVs while performing their flights from a pre-specified starting point to an end point. There are several forbidden points in the space as well, indicating the areas of geographical obstacles or unauthorized zones.

We execute Synth4UAV, i.e., the corresponding formal model on a Windows 10 machine equipped with an Intel Core i5 processor and 12 GB of memory. We change different input parameters of the model and observe the relationships between the parameters and the tool's scalability. The evaluation heavily depends on the placement of the available points in the 3D space and the allowed ranges of the UAVs. Hence, for a particular evaluation, we consider executions of the model on several arbitrary data point settings and take the average. The maximum allowed turn angle and climb angle are chosen according to the placements of the points.

B. Relationship between Model Parameters

We present the relationship among different parameters in trajectory design in Fig. V-D and V-D. We observe the maximum coverage produced by a SAT result while varying the budget for each UAV in Fig. 6(a). We present the analysis for two cases: one with 2 UAVs and the other with 4 UAVs while allowing adequate time for coverage of 70 points. It is obvious that more points can be covered by the UAVs if there is more budget (for fuel). If we use more UAVs, then we can cover more data points with the same per UAV budget. For example, if we use 2 UAVs, then we can cover 77% of the data points with a budget of \$700 per UAV. However, using 4 UAVs with the same per UAV budget allows us to cover all the data points.

In Fig. 6(b), we present the coverage time of the data points for the different numbers of UAVs with budget set to \$500. We show results for the requirement of 40% coverage and 60% coverage. Note that even if the UAVs have the same source and destination points, they reach the destination point at different times, as they may have different speeds and are not allowed to visit the same point at the same time. Hence, we consider the coverage time as the time required by the last UAV to reach the destination. We observe that the required time decreases as we have more UAVs to cover the given percentage of data points.

We present the number of UAVs minimally required to cover different percentages of the data points in Fig. 6(c) for two problem sizes: 50 points and 90 points. The figure shows that as we increase our coverage requirement, the minimum number of required UAVs also increases if we keep the per UAV budget and the visiting time thresholds constant (set to 300s and \$500 respectively). We show the results for two problem sets: one with 50 possible waypoints and another with 90 waypoints.

The change in the percentage of coverage with respect to the allowed time to reach the destinations is presented in Fig. 7(a).

(a) (b) (c)

Fig. 7. (a) The coverage of data points p_2 ($p_2 \in S$) w.r.t. the budgeted time for 2 and 5 UAVs, respectively, (b) The minimum required budget w.r.t. the number of data points ($p \in S$) for a total of 50 and 80 points, respectively, and (c) the minimum required budget w.r.t. the resiliency for 30 and 50 data points in ($p \in S$), respectively.

(a) (b) (c)

(d) (e) (f)

Fig. 8. The CSP/formal model solving time w.r.t. (a) the number of points for SAT and UNSAT results, (b) the number of UAVs for 60 and 90 points, respectively, (c) the required data points coverage ($p_2 \in S$) for 50 and 70 points, respectively, (d) the number of data points ($p \in S$) for a totl number of 50 and 70 points, respectively , (e) the resiliency for 30 and 50 data points ($p_2 \in S$), and (f) the k resiliency with data freshness requirement for 30 and 50 data points ($p \in S$).

This time is the maximum time allowed for any UAV to reach the end point of its trajectory. We show two cases, one for 2 UAVs and another for 5 UAVs, for a total of 100 points, among which 40 are data points. We keep the value of k (i.e., the resiliency specification) as 2. In the figure, we observe that as the allowed time increases, more of the data points can be covered. The higher number of UAVs means that more of the data points can be covered. For the allowed time of 1800 sec, 5 UAVs can visit all the data points and reach the final point on their trajectories.

In Fig. 7(b), we present the minimum required budget per UAV while varying the number of data points. We keep the total number of points, required coverage, the value of k, etc., unchanged for this evaluation. The minimum required budget is the budget for which the solver returns a SAT answer (i.e., provides a trajectory). As expected, the required budget increases with the increment of the number of possible data points in the 3D space.

The minimum required budget per UAV with the change of the value of k is presented in Fig. 7(c). We can observe that with the increment of k, the required budget increases quickly. As the value of k increases, a higher number of UAVs are required to cover more data points, which results in more trajectories for the UAVs. Hence, a bigger budget is required to cover the fuel cost of the UAVs.

C. Scalability

We present the scalability results of our tool in solving the formal model, i.e., synthesizing a trajectory plan in Fig. VI-B. To the best of our knowledge, there is no similar work that uses formal methods to design efficient trajectories based on resilient data collection objectives. Hence, we do not have a baseline to compare the synthesis times or the scalability with prior works.

Fig. 8(a) shows the trajectory synthesis time for different problem sizes (i.e., the number of points the UAVs can go). We vary the number of points in the 3D space and observe the time required to obtain the trajectory plan. We can observe that the required time increases with the increment of the number of points. As we have more points in the problem space, the solver requires to deal with more constraints, which leads to apparently a linear increment of the synthesis time. We also observe that the UNSAT cases require more synthesis time than the SAT cases.

In Fig. 8(b), we present the trajectory synthesis time by varying the number of UAVs. We consider two cases with different problem sizes: 60 points and 90 points. In these cases, we keep the required percentage of coverage and other parameters constant. We observe that as the number of UAVs increases, the synthesis/execution time also increases. In the case of 90 points, we need more clauses to model the problem, and thus the execution time is higher compared to the case of 60 points. The increase in time with the increase of the number of UAVs is also higher for the case of 90 points.

We present the trajectory synthesis time by varying the required coverage of the data points in Fig. 8(c). As the requirement to cover more points grows, the synthesis time increases. For a larger problem set (e.g., 90 points), the time grows quickly as more constraints are satisfied.

The trajectory synthesis time with respect to the number of data points is observed in Fig. 8(d). We show the time for two cases based on the problem size (the number of total points): 50 points and 80 points. We keep all other parameters constant. For example, the required coverage is kept constant at 80%, and the value for k is 2. We can observe that the synthesis time increases slowly with the increment of the number of data points for a certain problem size. This is because, with the increment of data points, the UAVs require to visit more points, and the solver needs to solve a stricter set of constraints. However, the number of clauses does not increase drastically as the total points remain the same.

Next, in Fig. 8(e), we present the trajectory synthesis time by varying the value of k, which is the minimum number of UAVs that need to visit at least a certain percentage of data points. Here, no data freshness restriction is considered for visiting a certain data point by k UAVs. We observe that the execution time increases as k increases. With the increment of k, the solver needs to satisfy stricter requirements of visiting a data point with a larger number of UAVs. Fig. 8(f) presents the synthesis times in the case of resiliency with the data freshness constraint, i.e., the time difference among the visits by a set of k UAVs to a certain data point cannot be more than a threshold value (here, 20 sec). We keep other parameters, such as the number of UAVs and total points, required coverage, etc., constant for both of the cases of 30 data points and 50 data points. As the figure shows, as k increases, the synthesis time increases. However, as we compare the results with that of in Fig. 8(e), the time increases more rapidly than the case of increasing k without the data freshness constraint. This is because the solver needs to verify a highly strict condition of visiting a data point by a set of k UAVs within a certain time period.

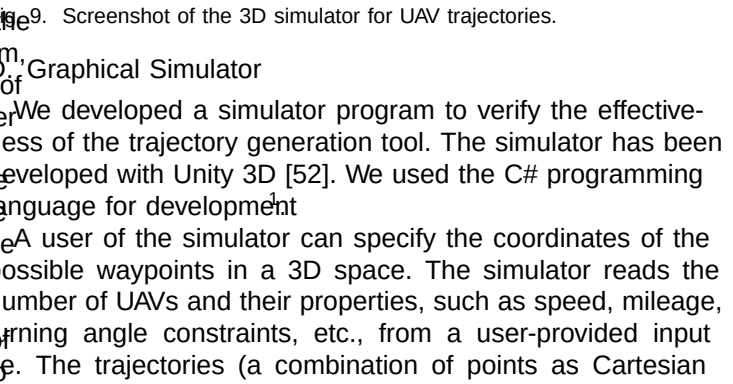


Fig. 9. Screenshot of the 3D simulator for UAV trajectories.

D. Graphical Simulator

We developed a simulator program to verify the effectiveness of the trajectory generation tool. The simulator has been developed with Unity 3D [52]. We used the C# programming language for development.[1]

A user of the simulator can specify the coordinates of the possible waypoints in a 3D space. The simulator reads the number of UAVs and their properties, such as speed, mileage, turning angle constraints, etc., from a user-provided input file. The trajectories (a combination of points as Cartesian coordinates in a 3D space) are provided based on the output of the trajectory generator. The times of reaching the waypoints on their trajectories and that of hovering at the points to avoid collision and proper data collection are also provided to the simulator. When started, the simulator shows the movement of the UAVs according to its input and proves the effectiveness of the trajectory generator by showing that the UAVs do not collide but collectively collect the required data and complete the job before the deadline in order to prevent fuel outage, as well as satisfy the resiliency requirements.

Fig. 9 presents a screenshot of the simulator tool that we developed. The figure shows two UAVs flying from point to point collecting data. The spheres represent the waypoints above or in close proximity to the data sources. The red spheres denote the forbidden points that the UAVs avoid. The simulator presents an animation of the data collection scenario by the UAVs in a 3D environment. The simulation confirms the effectiveness of the generated trajectories by showing the absence of collisions, as well as the time to reach the end points of the trajectories of the UAVs.

VII. Conclusion

UAVs are increasingly used in surveillance, as well as data collection from remote sensors. In this paper, we have developed Synth4UAV that can automatically generate appropriate trajectories for a set of UAVs to efficiently perform their intended tasks collaboratively. We have considered various constraints related to UAV routing and maneuvering, fuel cost, visiting time, and resiliency requirements for the data

[1]https://tinyurl.com/y2z8aena

collection. We have designed the trajectory planning, which is a combinatorically hard problem, as an SMT-based constraint satisfaction problem, the solution to which generates trajectories that the UAVs should follow. We have analyzed different trajectory planning factors like data coverage, swarm size, budget, and resiliency properties. Our evaluation results also have proved the scalability of the tool.

REFERENCES

[1] Lili Ma and YangQuan Chen. Aerial surveillance system for overhead power line inspection. Center for Self-Organizing and Intelligent Systems, 2004.

[2] Chung Deng, Shengwei Wang, Zhi Huang, Zhongfu Tan, and Junyong Liu. Unmanned aerial vehicles for power line inspection: A cooperative way in platforms and communications. J. Commun., 9(9):687–692, 2014.

[3] Carlos A Trasviña-Moreno, Rubén Blasco, Álvaro Marco, Roberto Casas, and Armando Trasviña-Castro. Unmanned aerial vehicle based wireless sensor network for marine-coastal environment monitoring. Sensors, 17(3):460, 2017.

[4] Ahmed EAA Abdulla, Zubair Md Fadlullah, Hiroki Nishiyama, Nei Kato, Fumie Ono, and Ryu Miura. An optimal data collection technique for improved utility in uas-aided networks. In INFOCOM, 2014 Proceedings IEEE, pages 736–744. IEEE, 2014.

[5] San Yeung, Sanjay Kumar Madria, Mark Linderman, and James Milligan. Routing and scheduling of spatio-temporal tasks for optimizing airborne sensor system utilization. In Proceedings of the 10th ACM Intl. Conf. on Distributed and Event-based Systems, pages 145–152. ACM, 2016.

[6] Leonardo de Moura and Nikolaj Bjørner. Z3: An ef cient SMT solver. In Conf. on Tools and Algo. for the Constr. and Analysis of, 2008.

[7] Leonardo De Moura and Nikolaj Bjørner. Satis ability modulo theories: introduction and applications. Communications of the ACM, 54(9):69–77, 2011.

[8] Mohammad Ashiqur Rahman, Rahat Masum, Matthew Anderson, and Steven L. Drager. Formal synthesis of trajectories for unmanned aerial vehicles to perform resilient surveillance of critical power transmission lines. In 25th International Conference on Engineering of Complex Computer Systems (ICECCS), pages 103–112, October 2020.

[9] John Tisdale, ZuWhan Kim, and J Karl Hedrick. Autonomous uav path planning and estimation. IEEE Robotics & Automation Magazine, 16(2):35–42, 2009.

[10] San Yeung, Sanjay Kumar Madria, and Mark Linderman. A trajectory recommendation system via optimizing sensors utilization in airborne systems (demo paper). In International Symposium on Spatial and Temporal Databases, pages 508–513. Springer, 2015.

[11] Amarender Reddy Mekala, Sanjay Madria, and Mark Linderman. Aerial vehicle trajectory design for spatio-temporal task aggregation. In Unmanned Aircraft Systems (ICUAS), 2016 International Conference on, pages 1085–1094. IEEE, 2016.

[12] Juan Liu, Xijun Wang, Bo Bai, and Huaiyu Dai. Age-optimal trajectory planning for uav-assisted data collection. arXiv preprint arXiv:1804.09356, 2018.

[13] Moataz Samir, Sanaa Sharafeddine, Chadi Assi, Tri Nguyen, and Ali Ghrayeb. Uav trajectory planning for data collection from time-constrained iot devices. IEEE Transactions on Wireless Communications, 2019.

[14] Randal W Beard, Derek B Kingston, Morgan Quigley, Deryl Snyder, Reed Christiansen, Walt Johnson, Timothy W McLain, and Michael A Goodrich. Autonomous vehicle technologies for small fixed-wing uavs. JACIC, 2(1):92–108, 2005.

[15] Nicola Ceccarelli, John J Enright, Emilio Frazzoli, Steven J Rasmussen, and Corey J Schumacher. Micro uav path planning for reconnaissance in wind. In American Control Conference, 2007. ACC'07, pages 5310–5315. IEEE, 2007.

[16] Erik Kuiper and Simin Nadjm-Tehrani. Mobility models for uav group reconnaissance applications. In Wireless and Mobile Communications, 2006. ICWMC'06. Intl. Conf. on, pages 33–33. IEEE, 2006.

[17] Bo Li, Xiaogang Qi, Baoguo Yu, and Lifang Liu. Trajectory planning for uav based on improved aco algorithm. IEEE Access, 2019.

[18] Scott A Bortoff. Path planning for uavs. In American Control Conf., 2000. Proceedings of the 2000, volume 1, pages 364–368. IEEE, 2000.

[19] Qingqing Wu, Yong Zeng, and Rui Zhang. Joint trajectory and communication design for multi-uav enabled wireless networks. IEEE Trans. on Wireless Communications, 17(3):2109–2121, 2018.

[20] M Bagherian and A Alos. 3d uav trajectory planning using evolutionary algorithms: A comparison study. The Aeronautical Journal, 119(1220):1271–1285, 2015.

[21] Haichao Wang, Jin Chen, Guoru Ding, and Jiachen Sun. Trajectory planning in uav communication with jamming. In 2018 10th International Conference on Wireless Communications and Signal Processing (WCSP), pages 1–6. IEEE, 2018.

[22] Antonios Tsourdos. A formal model approach for the analysis and validation of the cooperative path planning of a uav team. 2005.

[23] José J Ruz, Orlando Arévalo, Gonzalo Pajares, and M Jesus. Uav decision making among alternative routes for uavs in dynamic environments. In Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on, pages 997–1004. IEEE, 2007.

[24] Jose J Ruz, Orlando Arevalo, Gonzalo Pajares, and M Jesus. Uav trajectory planning for static and dynamic environments. Aerial Vehicles. InTech, 2009.

[25] Mohammadreza Radmanesh, Manish Kumar, Alireza Nemati, and Mohammad Sarim. Dynamic optimal uav trajectory planning in the national airspace system via mixed integer linear programming. Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering, 230(9):1668–1682, 2016.

[26] Hazem Sallouha, Mohammad Mahdi Azari, and So e Pollin. Energy-constrained uav trajectory design for ground node localization. In 2018 IEEE Global Communications Conference (GLOBECOM), pages 1–7, 2018.

[27] Jesús Sánchez-García, Daniel Gutiérrez, and S.L. Toral. A distributed pso-based exploration algorithm for a uav network assisting a disaster scenario. Future Generation Computer Systems, 90, 07 2018.

[28] Shakil Ahmed, Mostafa Zaman Chowdhury, and Yeong Min Jang. Energy-ef cient uav relaying communications to serve ground nodes. IEEE Communications Letters, 24(4):849–852, 2020.

[29] AHM Jakaria and Mohammad Ashiqur Rahman. Formal analysis of k-resiliency for collaborative uavs. In 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), pages 583–592. IEEE, 2018.

[30] AHM Jakaria and Mohammad Ashiqur Rahman. Safety analysis for uav networks. In 2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI), pages 294–295. IEEE, 2018.

[31] Terry Jameson. A fuel consumption algorithm for unmanned aircraft systems. Technical report, Army Research Lab White Sands Missile Range NM Computational And Info Science Directorate, 2009.

[32] Qian Wang, Zhi Chen, and Hang Li. Energy-ef cient trajectory planning for uav-aided secure communication. China Communications, 15(5):51–60, 2018.

[33] Yongbo Chen, Jianqiao Yu, Yuesong Mei, Siyu Zhang, Xiaolin Ai, and Zhenyue Jia. Traj. optimization of multiple quad-rotor uavs in collaborative assembling task. Chinese Journal of Aeronautics, 29(1):184–201, 2016.

[34] Wei Zhao, Wen Qiu, Taoyang Zhou, Xun Shao, and Xiujun Wang. Sarsa-based trajectory planning of multi-uavs in dense mesh router networks. In 2019 International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pages 1–5. IEEE, 2019.

[35] Dileep M Vijayakumari, Seungkeun Kim, Jinyoung Suk, and Hyemin Mo. Receding-horizon trajectory planning for multiple uavs using particle swarm optimization. In AIAA Scitech 2019 Forum, page 1165, 2019.

[36] Liang Sun and Randal W Beard. Trajectory tracking for unmanned aerial vehicles with autopilot in the loop.

[37] Zubair Md Fadlullah, Daisuke Takaishi, Hiroki Nishiyama, Nei Kato, and Ryu Miura. A dynamic trajectory control algorithm for improving the communication throughput and delay in uav-aided networks. IEEE Network, 30(1):100–105, 2016.

[38] Jonas Kvarnström and Patrick Doherty. Automated planning for collaborative uav systems. In Control Automation Robotics & Vision (ICARCV), 2010 11th Intl. Conf. on, pages 1078–1085. IEEE, 2010.

[39] Brett Bethke, Jonathan P How, and John Vian. Group health management of uav teams with applications to persistent surveillance. In American Control Conference, 2008, pages 3145–3150. IEEE, 2008.

[40] Alvi Ataur Khalil, Alexander J Byrne, Mohammad Ashiqur Rahman, and Mohammad Hossein Manshaei. Replanner: Ef cient uav trajectory-planning using economic reinforcement learning. In 2021 IEEE International Conference on Smart Computing (SMARTCOMP), pages 153–160. IEEE, 2021.

[41] M Nedim Alpdemir. Tactical uav path optimization under radar threat using deep reinforcement learning. Neural Computing and Applications 34(7):5649–5664, 2022.