Optimizing Fault-Tolerant Quality-Guaranteed Sensor Deployments for UAV Localization in Critical Areas via Computational Geometry

Marco Esposito, Toni Mancini, and Enrico Tronci

***Abstract*—The increasing spreading of small commercial unmanned aerial vehicles (UAVs, also known as drones) presents serious threats for critical areas, such as airports, power plants, and governmental and military facilities. In fact, such UAVs can easily disturb or jam radio communications, collide with other flying objects, perform espionage activity, and carry offensive payloads, e.g., weapons or explosives. A central problem when designing surveillance solutions for the localization of unauthorized UAVs in critical areas is to decide how many triangulating sensors to use, and where to deploy them to optimize both coverage and cost effectiveness. In this article, we compute deployments of triangulating sensors for UAV localization, optimizing a given blend of metrics, namely: coverage under multiple sensing quality levels, cost-effectiveness, and fault-tolerance. We focus on large, complex three-dimensional (3-D) regions, which exhibit obstacles (e.g., buildings), varying terrain elevation, different coverage priorities, and constraints on possible sensors placement. Our novel approach relies on computational geometry and statistical model checking and enables the effective use of off-the-shelf AI-based black-box optimizers. Moreover, our method allows us to compute a closed-form, analytical representation of the region uncovered by a sensor deployment, which provides the means for rigorous, formal certification of the quality of the latter. We show the practical feasibility of our approach by computing optimal sensor deployments for UAV localization in two large, complex 3-D critical regions, the Rome Leonardo Da Vinci International Airport (FCO) and the Vienna International Center (VIC), using NOMAD as our state-of-the-art underlying optimization engine. Results show that we can compute optimal sensor deployments within a few hours on a standard workstation and within minutes on a small parallel infrastructure.**

***Index Terms*—Artificial intelligence (AI)-based black-box optimization (BBO), statistical model checking, three-dimensional (3-D) computational geometry, unmanned aerial vehicles (UAVs) localization.**

Manuscript received 7 May 2023; revised 7 September 2023; accepted 17 October 2023. This work was supported in part by the MUR Grant "Dip. Eccellenza 2018–2022" (Dept. CS, Sapienza Univ. Rome); in part by the INdAM "GNCS Project 2022"; in part by Sapienza under Project RG12117A8B393BDC, Project RG11916B892E54DB, and Project RG120172B9329D33; in part by Lazio POR FESR under Project E84G20000150006 and Project F83G17000830007; and in part by NRRP ECS 0000024, Mission 4, Comp. 2, Inv. 1.5, NextGenEU, MUR CUP B83C22002820006, Rome Technopole. This article was recommended by Associate Editor Q. Wang. *(Corresponding author: Toni Mancini.)*

The authors are with the Department of Computer Science, Sapienza University of Rome, 00198 Rome, Italy (e-mail: esposito@di.uniroma1.it; tmancini@di.uniroma1.it; tronci@di.uniroma1.it).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TSMC.2023.3327432.

Digital Object Identifier 10.1109/TSMC.2023.3327432

I. Introduction

In RECENT years, small unmanned aerial vehicles (UAVs), or drones, have become widespread, given their increasingly lower prices and useful features. Besides the use of these devices for leisure (e.g., photography) and public service activity like disaster monitoring, the public has become more and more aware of the risks that drones pose in several scenarios. A commercial drone, in fact, can disturb or jam radio communications, collide with other flying objects, perform espionage activity, and even carry offensive payloads like weapons or explosives. For these and other reasons, drones are seen as a serious threat in critical areas, such as airports, military bases, correctional centers, power plants, and government sites. Fig. 1 shows two examples of critical areas: 1) the Rome Leonardo Da Vinci International Airport (FCO) in Italy and 2) the Vienna International Center (VIC) in Austria, which we have used as our case studies.

Many surveillance solutions, based on a variety of technologies, have been designed to quantify and reduce the risks stemming from the presence of unauthorized drones in critical areas. In general, an anti-drone system has one or more goals, which may target both the drone and its controller: detection, localization, identification, tracking, and countering. In this article, we focus on deployments of sensors aiming at localizing unauthorized (possibly malicious) drones.

Each technology has its advantages and limitations. We refer the reader to [1] and [2] for a survey. In our setting, i.e., in the case of critical areas, an anti-drone system must have certain specific features, and this limits the suitability of most sensor typologies. Namely, the system must be able to detect drones over possibly large distances (e.g., up to several kilometers within an airport); the localization must be reliable with minimal to no false positives and false negatives; the system must be safely employable in public areas and at close distance to people. Such requirements rule out video and acoustic sensors, due to their low detection reliability, and radar systems for their high-power electromagnetic emissions, which may produce adverse health effects. Radio frequency sensors, which detect and localize drones based on the radio signals they emit, are among the most used technologies for dealing with the problem [1]. They operate over lower frequencies than radars and can effectively isolate the signals emitted by drones from other radio frequency signals originating from other sources (e.g., WiFi). Direction-finding radio frequency

(a)

(b)

Fig. 1. Satellite views of our case studies (Google ©). (a) Leonardo Da Vinci Airport, Rome, Italy (FCO). (b) Vienna International Center, Austria (VIC).

sensors are able to localize drones via triangulation and offer high reliability over large distances (up to several kilometers in ideal conditions), with low to no impact from light and weather conditions.

A. Motivation

The choice of the right sensor typology or a mix of different typologies only solves a part of the general problem. When designing a system for drone localization, it is crucial to deploy the sensors in the best possible way. The two main factors that influence the quality of an anti-drone system are its cost-effectiveness and its ability to cover the Region of Interest (RoI). The latter is usually formalized as coverage, a measure of the extent of the portion of the monitored region where the deployed sensors can effectively localize targets. Typically, higher coverage requires higher costs, which stem from, e.g., the need of a higher number of sensors or the necessity to deploy sensors in inconvenient positions (e.g., on the roof or walls of a building, or where additional infrastructures must be built), which increase installation costs. We note that a single sensor, at the time of writing, may cost several tens of thousands of dollars.

The problem of optimally placing a set of sensors (with respect to a suitable blend of the conflicting objectives coverage and cost) has been widely explored in the literature, due to its relevance in many areas, e.g., sensor networks, infrastructure security and safety, smart cities, and smart homes. A discussion on related work is provided in Section II

Determining a deployment of sensors that shows good coverage and cost-effectiveness measures may still not be enough for many critical scenarios. For example, more elaborate coverage measures may need to be considered, which take into account portions of the RoI with different priorities, multiple sensing quality levels, and tolerance to sensor faults. Also, in such settings, providing as output only aggregate–although quantitative–quality measures (e.g., the ratio of the volume of the RoI satisfactorily covered), is often not enough to build trust in the quality of the deployment found, and delivering compelling evidence about which portions of the RoI are satisfactorily covered and which are not is crucial to provide means for rigorous auditing and formal certification of the quality of the deployment. It may be the case, for instance, that a deployment which appears to well cover a critical portion of the RoI is actually problematic, as it would not withstand the failure of a single sensor. Or that a deployment which covers most of the volume of the RoI still leaves drones the sufficient amount of freedom to move undetected via, e.g., narrow, worm-shaped corridors (which, e.g., could form only after the sabotage of a single sensor), and approach, still undetected, a critical target, thus causing serious damage. Note that, being able to determine coverage point by point, as in [3], [4], and [5] in a discrete space is not enough to study this kind of properties of a deployment.

B. Contribution

In this article, we present a novel approach to the sensor placement problem based on computational geometry and artificial intelligence (AI) black-box optimization (BBO).

In recent years, AI-based BBO techniques have witnessed exceptional improvements; new algorithms and tools are now able to tackle large and hard optimization problems [6], [7], [8]. These optimizers make use of a black box that implements the objective function and, possibly, the problem constraints. Thanks to powerful AI heuristics and surrogate models automatically learned during the search, these tools are powerful enough to optimize the objective, subject to the provided constraints, even if the black box is computationally expensive to evaluate (our case).

We designed and developed a geometry-based sensor deployment coverage analyzer (GD-Cover), a software tool that efficiently computes (via computational geometry and statistical model checking) quality metrics for a given sensor deployment, as well as a closed-form, analytical representation of the uncovered region which provides the means for rigorous, formal certification of its quality.

We show that, using GD-Cover as a black box, the sensor deployment problem can be efficiently tackled by off-the-shelf AI-based BBO solvers (NOMAD [9] in our experiments). Our method scales very well over large and complex scenarios with many obstacles and over large numbers of sensors.

To the best of our knowledge, our approach is also the first to enable the computation of a closed-form three-dimensional (3-D) representation of the region not covered by a set of sensors. Finally, thanks to our proof-of-concept visualization Web app, such a representation can be navigated by the users, so to precisely understand the characteristics of deployments and to facilitate possible audits and certification.

II. State of the Art

The problem of determining the optimal placement of sensors inside a given region has been extensively studied in the literature. An optimal placement is defined as a positioning and configuration of a set of sensors that optimize given key performance indicators (KPIs). Most studies consider the coverage performance metric, which measures the quality of a deployment based on the portion of the RoI that it can monitor. The configurations may vary depending on the kind of sensors; for instance, camera sensors can be configured by, e.g., tweaking orientation, pan, tilt, and zoom. Due to the complexity of the problem and the large size of the scenarios of interest, exact optimization approaches are not generally viable, and virtually all works in the literature exploit incomplete (best-effort) optimization techniques, typically metaheuristic methods (e.g., evolutionary algorithms and particle swarm optimization) and various forms of gradient descent.

Sensor deployments are often very costly; therefore, many existing techniques also minimize the overall cost of the deployment. Other performance metrics have been studied, such as least exposure coverage, fault tolerance, and minimum overlapping (see [5], [10], [11], [12], [13], [14]).

Several works, e.g., [15], [16], [17], [18], [19], and [20], investigate the related problems of computing deployments of wireless sensor networks and of UAVs optimizing additional KPIs, such as connectivity, energy efficiency, and reliability.

Many existing studies assume a very simplified environmental setting, where the RoI is defined as a two-dimensional (2-D) space [12], [13], [21], [22], [23], [24], [25], even if, in some cases (e.g., [26]), sensors can be deployed at different heights. These techniques cannot be applied in the case of UAVs localization, where targets fly in a 3-D space, as the coverage of a 2-D region cannot be easily generalized to that of a large 3-D scenario with obstacles.

Several approaches consider the problem of monitoring a 3-D space, but with significant limitations affecting the applicability of these approaches to localizing small UAVs in large critical areas, such as those considered here. In [5], [11], [14], [27], [28], and [29], the candidate points for placement all lie on a 3-D surface; in [3], [4], [30], [31], [32], and [33], admissible sensor positions or points to be monitored (or both) belong to finite sets in the 3-D space. In particular, in most existing 3-D approaches, the RoI is discretized in cells. The visibility algorithms work on the assumption that if a point in the cell (e.g., the centroid) is visible, then the whole cell is covered. This assumption becomes problematic in our setting. Consider, e.g., our FCO case study: this environment has an area of around 16 km^2 and a height of 100 m. If we discretize the space into 3-D cubic cells with edges of length 50 m, we would obtain around 12 800 cells (or target points). Although this number is still manageable by existing approaches, we note that a sensor deployment covering the centroid of a 125 000 m^3 cell does not guarantee that it can localize a 50-cm long drone anywhere within the cell. Conversely, if we use much smaller cell edge lengths, such as 1 m, we would obtain approximately 1.6 billion cells. Such a large number would not be feasible to handle by any existing method with reasonable time and computational resources, even if exploiting graphical processing units, as in [34].

In practical applications, not all sensor deployments have the same economic cost. Typically, the cost depends not only on the number of deployed sensors but also on their characteristics and positions. The minimization of the number of sensors is studied in [4], [21], [22], [28], [31], and [32]; however, it is often the case that the available sensors have different prices due to different characteristics, so minimizing the number of sensors does not imply that the overall cost is minimized. Furthermore, the cost of physically deploying a sensor also depends on its position. For instance, mounting a heavy sensor on a wall costs more than mounting it on a roof, which in turn costs more than placing it on the ground. References [3], [5], [27], and [35] assume sensors have a fixed cost, while [14] and [36] assign to each sensor a cost based only on the altitude and the roughness of the terrain in its position. Conversely, our approach handles this issue as a first-class citizen.

Summing up, to our knowledge (see also the recap in Appendix A in the supplementary material), no other available approach optimizes sensor deployments for UAV localization in large, complex 3-D regions with obstacles, varying terrain elevation, and in the presence of constraints on admissible placements, by simultaneously taking into account sensors of different typologies, different placement costs, multiple sensing quality levels, and fault tolerance. Also, no other approach supports the computation of a closed-form 3-D representation of the region not covered by a candidate deployment.

III. Problem Modeling

In this section, we present our geometric modeling approach to the computation of an optimal sensor deployment. In the following, $\mathbb{R}$, $\mathbb{R}_{0+}$, and $\mathbb{R}_{+}$ denote the set of all, non-negative, and strictly positive real numbers, while $\mathbb{N}$ and $\mathbb{N}_{+}$ denote the set of non-negative and strictly positive integers.

Although our forthcoming definitions are well posed for real spaces of any number of dimensions, they will be given for regions of the 3-D space $\mathbb{R}^3$, since this is what we need for our problem. We thus use the general term region to denote any set of points in $\mathbb{R}^3$. Also, given three points $A, B, C \in \mathbb{R}^3$, we define by AB the straight-line segment between A and B, by $\overline{AB}$ its length (i.e., the Euclidean distance between A and B), and by $\angle BAC \in [0, \pi]$ the angle formed by AB and AC. Finally, given two regions $R, R' \subseteq \mathbb{R}^3$, the distance between R and R', notation $\mathrm{dist}(R, R')$, is defined as $\min\{\overline{XY} \mid X \in R, Y \in R'\}$. This notion naturally reduces to the Euclidean distance $\overline{XY}$ between two points X and Y, if $R = \{X\}$ and $R' = \{Y\}$ are both singleton sets.

A. Region of Interest and Obstacles

The RoI, i.e., the region where the presence of unauthorized UAVs is to be detected, is some $\mathcal{R} \subset \mathbb{R}^3$ having finite volume.

The RoI can exhibit obstacles, e.g., buildings, other artifacts, or simply varying ground elevations, which can hinder the radio visibility of some target points by a deployed sensor. Being able to explicitly model the position and shape of such obstacles is thus crucial to accurately evaluate the quality of a sensor deployment. We assume that the space occupied by obstacles in the RoI is defined as a (typically disconnected) region $\mathcal{O} \subset \mathcal{R}$.

B. Priorities

For certain kinds of environments, the coverage of some portions of the RoI $\mathcal{R}$ is more important than that of others. In an airport, for instance, being able to localize a UAV flying above the runways could be more important than localizing one close to the terminals.

We assume that $\mathcal{R}$ is partitioned into a finite number of regions having different priorities. Priorities are encoded as elements in finite set $\mathcal{H}$. The overall RoI $\mathcal{R}$ is hence partitioned into $\{\mathcal{R}_h \mid h \in \mathcal{H}\}$, where $\mathcal{R}_h$ is the portion of $\mathcal{R}$ having priority h. Being a partition, each point in $\mathcal{R}$ is assigned exactly one priority value.

C. Sensor Deployments, Quality-Guaranteed Point Coverage by Triangulating Sensors

The uncertainty in detecting a target provided by two triangulating sensors is known to vary with respect to the target–sensor distance and the angle θ between the target and the sensors (see [37]).

Multiple Sensing Quality Levels: We support optimization with respect to multiple sensing quality levels. To this end, we assume that a finite set $\mathcal{Q}$ is defined to denote different requested sensing quality levels. This set is ordered so that higher values in $\mathcal{Q}$ denote higher quality levels.

Sensing Angles: For each $q \in \mathcal{Q}$, we assume that a sensing angle range $[\theta_{\min,q}, \theta_{\max,q}] \subset (0, \pi)$ is provided, defining bounds to be respected (Definition 1) by the angle θ between the target and the two triangulating sensors, to ensure quality level q. For physical reasons, we can assume that $[\theta_{\min,q'}, \theta_{\max,q'}] \subseteq [\theta_{\min,q}, \theta_{\max,q}]$ for $q \leq q'$. Note that sensing angle ranges are within $(0, \pi)$ so that, to be localized with any useful accuracy, the target must not be collinear with the two sensors (as collinearity would hinder triangulation [37]).

Sensors: Each sensor s is defined in terms of its admissible positions $\mathcal{A}_s \subseteq \mathbb{R}^3$ (i.e., where it can be placed), the function $\text{cost}_s : \mathcal{A}_s \to \mathbb{R}_+$ defining the cost to deploy the sensor in each admissible position, and its technical capabilities. Admissible positions may not correspond with $\mathcal{R}$: indeed, there may be portions of $\mathcal{R}$ where sensors cannot be deployed (e.g., bodies of water or airport runways) or, conversely, it may be possible to deploy sensors outside $\mathcal{R}$. The technical capabilities of s are available in the form of a set $\{r_{s,q}, f_{s,q} \mid q \in \mathcal{Q}\}$. For each sensing quality level q, $r_{s,q} \in \mathbb{R}_+$ and $f_{s,q} \in \mathbb{R}_+$ are, respectively, the maximum target distance and the radius of the first Fresnel zone (FFZ) needed by s to detect a target UAV with the accuracy required to satisfy quality level q (Definition 1). The FFZ of sensor s when detecting a target at position $X \in \mathcal{R}$ and ensuring quality level q is the 3-D ellipsoid whose main axis is the segment connecting the position of s and X and whose secondary axis has length $2f_{s,q}$. The FFZ radius of $s(f_{s,q})$ is the greatest value such that, if a stray component of the signal transmitted by a UAV bounces off an object within the FFZ and then arrives at s, the resulting phase shift will be considered to have a negative impact on the signal quality incompatible with sensing quality level q. Given the ranges on the transmitting/receiving power of the employed antennas and on the band used, as well the maximum distance $r_{s,q}$ for each $q \in \mathcal{Q}$, upper bounds to the values of $f_{s,q}$ for each s and q can be computed once and for all. Again, for physical reasons, we assume that $r_{s,q} \geq r_{s,q'}$ and $f_{s,q} \leq f_{s,q'}$ for $q \leq q'$.

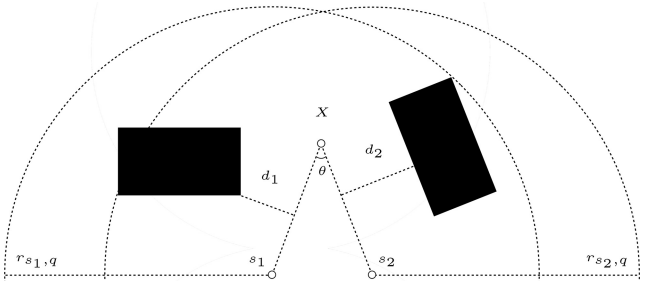

Fig. 2. 2-D q-coverage of point X by two sensors, with two obstacles (in black); $d_i = \text{dist}(X\mathcal{D}(s_i), \mathcal{O}) > f_{s_i,q}$, $i \in [1, 2]$.

Sensor Deployment: We finally define a deployment of a set of sensors $\mathcal{S}$ a function $\mathcal{D}$ assigning an admissible position $\mathcal{D}(s) \in \mathcal{A}_s$ to each $s \in \mathcal{S}$.

Definition 1 (see Fig. 2 for an illustration in 2-D) formalizes our criterion to establish whether a point X is covered by two sensors with quality at least q.

Definition 1 (Point q-Coverage by Triangulating Sensors): Point $X \in \mathcal{R} - \mathcal{O}$ is covered by two triangulating sensors s_1 and s_2 of deployment $\mathcal{D}$ with quality at least q (in short: X is q-covered by s_1 and s_2) if:

1) X is within the ranges of s_1 and s_2 for quality level q: $\text{dist}(X, \mathcal{D}(s_i)) \leq r_{s_i,q}$, $i \in [1, 2]$;
2) the line of sight between X and each of the two sensors lies at a distance higher than the radius of the sensor's FFZ for q from any existing obstacle: $\text{dist}(X\mathcal{D}(s_i), \mathcal{O}) > f_{s_i,q}$, $i \in [1, 2]$;
3) the angle θ formed by the points where the two sensors are placed and X is within the sensing angle range for q: $\angle\mathcal{D}(s_1)X\mathcal{D}(s_2) \in [\theta_{\min,q}, \theta_{\max,q}]$.

We write $\text{cover}^q(X, s_1, s_2) = 1$ to denote that X is q covered by s_1 and s_2, and $\text{cover}^q(X, s_1, s_2) = 0$ otherwise.

D. Fault-Tolerant Quality-Guaranteed Coverage

In critical settings such as ours, tolerance to sensor faults is an important issue. Hence, we will define quality-guaranteed coverage even in the (limited) presence of sensor faults. Namely, Definition 2 defines the whole portion of the RoI not guaranteed to be covered by a sensor deployment $\mathcal{D}$ with a required quality level $q \in \mathcal{Q}$ when in the presence of at most $j \geq 0$ (unknown) faulty sensors (in short: (j, q)-uncovered region).

Definition 2 [(j, q)-Uncovered Region]: The set of points $X \in \mathcal{R} - \mathcal{O}$ (j, q)-uncovered by a deployment $\mathcal{D}$ is

$$\mathcal{U}^{j,q} = \{X \in \mathcal{R} - \mathcal{O} \mid \text{cover}^{j,q}(X) = 0\} \tag{1}$$

where $\text{cover}^{j,q}(X) = \min_{F \in 2^{\mathcal{S}}, |F| \leq j} \max_{(s_1,s_2) \in (\mathcal{S}-F)^2, s_1 \neq s_2} \text{cover}^q(X, s_1, s_2)$. Namely, $\text{cover}^{j,q}(X)$ is 0 for those points X for which there exists a set F of at most j sensors that, if faulty, would prevent q-coverage of X by the others; $\text{cover}^{j,q}(X)$ is 1 otherwise.

E. Objective

Given an RoI $\mathcal{R}$ with obstacles $\mathcal{O}$, a partitioning of $\mathcal{R}$ into priority regions $\{\mathcal{R}_h \mid h \in \mathcal{H}\}$, sets $\mathcal{Q}$ (sensing quality levels) and $\mathcal{S}$ (sensors), and a maximum number $k > 0$ of sensors which can be faulty, our goal is to find an (admissible) deployment $\mathcal{D}^* = \mathcal{S} \to \mathbb{R}^3$ of $\mathcal{S}$ that minimizes an objective function of the form

$$\mathcal{D}^* = \arg\min_{\mathcal{D}}(\text{Placem}(\mathcal{D}) + \text{Uncov}(\mathcal{D})) \tag{2}$$

given as the linear combination of the following (possibly conflicting) KPIs.

1) Placem($\mathcal{D}$) is the sensors placement cost of $\mathcal{D}$, i.e., the actual expense due to the chosen placement of the sensors

$$\text{Placem}(\mathcal{D}) = \sum_{s \in \mathcal{S}} \text{cost}_s(\mathcal{D}(s)). \tag{3}$$

2) Uncov($\mathcal{D}$) is the cost due to lack of coverage of $\mathcal{D}$, i.e., the weighted volume of $\mathcal{R}$ not (satisfactorily) covered by $\mathcal{D}$, under at most k faulty sensors

$$\text{Uncov}(\mathcal{D}) = \sum_{j=0}^{k} \sum_{q \in \mathcal{Q}} \sum_{h \in \mathcal{H}} w(j,q,h) \times \text{volume}\left(\mathcal{U}^{j,q} \cap \mathcal{R}_h\right). \tag{4}$$

Each weight $w(j, q, h)$ denotes the implicit cost of not ensuring (j, q)-coverage of a unit of volume in $\mathcal{R}_h$.

Such two (possibly conflicting) KPIs are defined as to represent an amount of money, and this allows us to sum them up into a single objective value (2), the overall deployment cost (ODC) of $\mathcal{D}$. Thus, the objective function would consider both the actual expense for the envisioned placement of the sensors in the chosen locations and the implicit costs due to their lack of (satisfactory) coverage.

IV. Black-Box Optimization

We cast the problem of finding an optimal sensor deployment as a constrained optimization problem, where the objective function is (2), the search space is the set of assignments of 3-D coordinates to each available sensor s, and where constraints enforce each sensor s to be positioned within its admissible region $\mathcal{A}_s$ and to triangulate with at least one other sensor (i.e., all sensors contribute to the coverage).

The complexity of computing the regions $\mathcal{U}^{j,q}$ [see (4)] needed to evaluate the objective function and their volumes hinders the possibility to exploit symbolic approaches (e.g., mixed-integer linear programming, constraint optimization, and the like) for scenarios of practical relevance, even those approaches explicitly aimed at solving very large instances, such as, e.g., [38] and [39].

We thus exploit AI-based BBO to solve realistic instances of the problem. BBO solvers make use of a black box that implements the objective function and the problem constraints. Thanks to powerful AI heuristics and surrogate models automatically learned during the search, these tools are powerful enough to optimize the objective, subject to the provided constraints, even if the black box is computationally expensive to evaluate (our case). In particular, state-of-the-art BBO solvers aim at reducing as much as possible the number of invocations of the (expensive) black box.

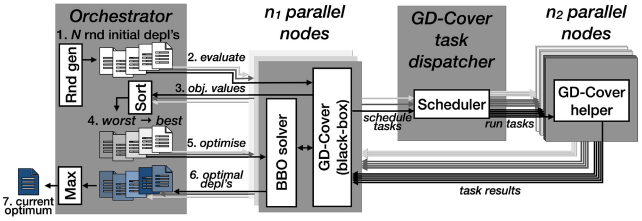

Fig. 3. High-level architecture of our BBO-based approach.

A BBO solver (we experimented with the state-of-the-art NOMAD [9] optimizer) repeatedly invokes our simulator (GD-Cover, see Section V) as a black box on multiple, intelligently chosen candidate deployments. GD-Cover is in charge of computing both the objective value of the input deployment $\mathcal{D}$ and how much $\mathcal{D}$ violates the problem constraints (or, conversely, how robustly $\mathcal{D}$ satisfies such constraints). This combined feedback is then provided back to the BBO solver, which is in charge to find a better candidate deployment to submit to GD-Cover, also building and exploiting a local surrogate model of the solution space.

Given that our problem is highly nonlinear and realistic instances are very large, finding a global optimum is an unviable option. BBO solvers like NOMAD are intrinsically incomplete, but guarantee global convergence to local optima and include sophisticated AI-based heuristics and random restarts to drive the search toward high-quality optima.

To exploit the availability of highly parallel computational infrastructures, possibly concurrently used for many tasks, we designed the two-level loosely coupled parallel architecture shown in Fig. 3. The overall system is assumed to work on overall n computational nodes. An orchestrator process (left) samples a high number N of random deployments (step 1) and asks GD-Cover (a highly parallel process itself, see below) to evaluate their objective values (steps 2 and 3). These N random deployments are then sorted from the best to the worst by the orchestrator (step 4) and then used (step 5) as initial assignments to launch, in parallel, $n_1 \leq N$ BBO solvers, where random deployments are assigned to the available solvers in the order defined earlier (best initial deployments first), as soon as they become idle (this is equivalent to the sequentially repeated invocation of a single optimizer using N restarts, where the best random deployments are used first). Each of the n_1 solvers uses GD-Cover as its black box. The best deployment found is given as the final solution. However, during the process, the current optimal deployment can be returned at any time, should the available time budget be over. GD-Cover itself (see Section V) is deployed as a highly parallel computational service consisting of a front-end process for each of the n_1 BBO solvers, a centralized task dispatcher, and n_2 parallel helper processes, running on the remaining nodes (thus, $n_2 = n - n_1 - 2$).

V. GD-Cover

In this section, we present GD-Cover, a highly parallel tool written in Java that, given a description of the geometry of the

RoI $\mathcal{R}$ and of its obstacles $\mathcal{O}$ (Section III-A), a partitioning $\{\mathcal{R}_h \mid h \in \mathcal{H}\}$ of $\mathcal{R}$ in priority regions (Section III-B), a deployment $\mathcal{D}$ of a set of sensors $\mathcal{S}$ with known properties for a given set of sensing quality levels $\mathcal{Q}$, an angle interval $\left[\theta_{\min,q}, \theta_{\max,q}\right]$ for each $q \in \mathcal{Q}$, the maximum number k of sensors that can be faulty offers the following services (one or more services can be requested at the same time).

1) Computes a quantitative measure of how much problem constraints are violated by $\mathcal{D}$ (thus proving that $\mathcal{D}$ is not admissible), or of how robustly they are satisfied (thus certifying that $\mathcal{D}$ is admissible).
2) Computes a closed-form representation of the region (j, q)-uncovered by $\mathcal{D}$ for any $j \in [0, k]$ and $q \in \mathcal{Q}$.
3) Efficiently estimates, by means of statistical model checking, the value of the objective function (Section III-E) with user-specified precision and statistical confidence.

GD-Cover has been designed to serve as a black box for our pool of BBO solvers seeking an optimal sensor deployment for a shared scenario. Hence, it is deployed as a highly distributed system, with each process being configured with a copy of the scenario of interest (e.g., RoI, priorities, and obstacles) and problem parameters. In particular, n_1 parallel processes are deployed, one per BBO solver, which act as front-ends (see Fig. 3), and delegate the most intensive computations to a pool of n_2 helper processes, orchestrated by a centralized task dispatcher which guarantees adequate load balancing among the n_1 optimization processes. The following sections describe the computations carried out by GD-Cover in more detail.

A. Polyhedral Geometry-Based Reasoning

GD-Cover performs its computations exploiting concepts from computational geometry. The key observation of the suitability of geometric reasoning to perform the computations above is forthcoming Proposition 1, which shows how a representation of the (j, q)-uncovered region $\mathcal{U}^{j,q}$ (Definition 2) can be provided in geometric terms, by relying on the following geometric notions: 1) the β-bloating of region R, which is the set of points having distance at most $\beta \in \mathbb{R}_+$ from R: $\text{bloat}(R, \beta) = \left\{X \in \mathbb{R}^3 \mid \text{dist}(X, R) \leq \beta\right\} \supseteq R$ and 2) the projection of point X onto region R, which is the region of points Y such that the segment XY intersects R: $\text{proj}(X, R) = \left\{Y \in \mathbb{R}^3 \mid XY \cap R \neq \emptyset\right\}$ (this is a specialized version of the projection defined in [40]). Note that, given point X and region R, the set of points Y such that the segment XY has a distance from R at most a given threshold β can be defined as $\text{proj}(X, \text{bloat}(R, \beta))$.

Proposition 1 (Geometric Representation of the (j, q)-Uncovered Region): The (j, q)-uncovered region of Definition 2 can be equivalently defined as

$$\mathcal{U}^{j,q} = \bigcup_{\substack{F \in 2^{\mathcal{S}} \\ |F| \leq j}} \bigcap_{\substack{(s_1,s_2) \in (\mathcal{S}-F)^2 \\ s_1 \neq s_2}} \mathcal{U}^{q}_{s_1,s_2} - \mathcal{O} \tag{5}$$

with

$$\mathcal{U}^{q}_{s_1,s_2} = \left[\mathcal{U}^{\text{range},q}_{s_1} \cup \mathcal{U}^{\text{range},q}_{s_2}\right] \cup \left[\mathcal{U}^{\mathcal{O},q}_{s_1} \cup \mathcal{U}^{\mathcal{O},q}_{s_2}\right] \cup \mathcal{U}^{\text{angle},q}_{s_1,s_2} \tag{6}$$

where, for $i \in [1, 2]$:

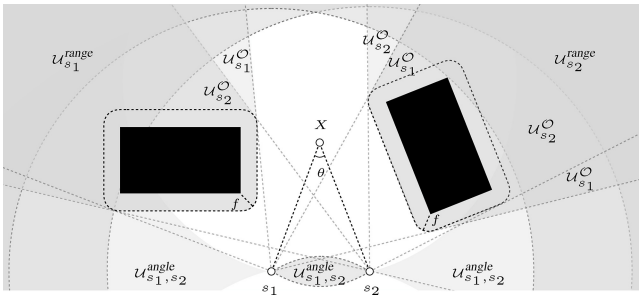

Fig. 4. 2-D example of region $\mathcal{U}_{s_1,s_2}$ uncovered by sensors s_1 and s_2 [gray region, (6)]. Different elements of the union (6) are highlighted with different tones of gray. The figure assumes, for simplicity, $f_{s_1} = f_{s_2} = f$.

1) $\mathcal{U}^{\text{range},q}_{s_i}$ is the region out of range of s_i

$$\mathcal{U}^{\text{range},q}_{s_i} = \left\{X \in \mathcal{R} \mid \text{dist}(X, \mathcal{D}(s_i)) > r_{s_i,q}\right\}$$

i.e., the complement of a sphere of radius $r_{s_i,q}$ centered in the position of sensor s_i, $\mathcal{D}(s_i)$.

2) $\mathcal{U}^{\mathcal{O},q}_{s_i}$ is the region not covered by s_i because of obstacles. It is the set of points $X \in \mathcal{R}$ such that the distance between an obstacle (a point in $\mathcal{O}$) and the straight-line segment connecting X with $\mathcal{D}(s_i)$ is at most the FFZ radius of s_i for quality level $q, f_{s_i,q}$

$$\mathcal{U}^{\mathcal{O},q}_{s_i} = \text{proj}\big(\mathcal{D}(s_i), \text{bloat}\big(\mathcal{O}, f_{s_i,q}\big)\big) \cap \mathcal{R}.$$

3) $\mathcal{U}^{\text{angle}}_{s_1,s_2}$ is the region not covered by s_1 and s_2 because of excessive sensing error. It is the set of points $X \in \mathcal{R}$ such that the angle $\angle\mathcal{D}(s_1)X\mathcal{D}(s_2)$ formed by X with the positions of the two sensors is outside range $\left[\theta_{\min,q}, \theta_{\max,q}\right]$

$$\mathcal{U}^{\text{angle},}_{s_1,s_2} = \left\{X \in \mathcal{R} \mid \angle\mathcal{D}(s_1)X\mathcal{D}(s_2) \notin \left[\theta_{\min,q}, \theta_{\max,q}\right]\right\}.$$

Proofs are delayed to Appendix B in the supplementary material. Fig. 4 illustrates Proposition 1 in 2-D and under no faults (i.e., $j = 0$), by showing the region uncovered by the sensors in Fig. 2.

To carry out its tasks efficiently, GD-Cover approximates the environment, i.e., the RoI and obstacles thereof, and performs all its computations in terms of bounded and unbounded convex polyhedra in $\mathbb{R}^3$. Polyhedral representations are indeed standard when handling data about geographic areas, terrain asperities, buildings, and other kinds of artifacts and shapes (e.g., the geometry of the runways at an airport) within geographic database systems and computer-aided design tools. Such a representation has several advantages: 1) every 3-D region can be (both over- and under-) approximated with arbitrary precision by a set of convex polyhedra (typically a small number of polyhedra guarantees good approximations) and 2) since convex polyhedra can be defined via linear constraints, they are easy to manipulate efficiently using standard computational geometry techniques and libraries.

A union of convex polyhedra, in general, is not a convex polyhedron. However, there are well-known techniques to manipulate unions (i.e., sets) of convex polyhedra very efficiently (some are briefly outlined in Section V-C).

Being able to represent arbitrary regions with unions of polyhedra yields a very convenient framework to perform complex operations. In fact, the union or the intersection

of unions of polyhedra can still be efficiently computed as a union of polyhedra. In the case of the difference and complement operations, the result is a union of nonclosed polyhedra; however, for our purposes such regions can be safely over- or under-approximated with unions of closed polyhedra with arbitrary precision. In the sequel, we will refer to convex closed polyhedra simply as polyhedra, and use the term polyhedral representation of a region to signify that the region is defined as a union of closed convex polyhedra.

Although the GD-Cover primary inputs are polyhedral, some of the computed regions (mainly those described in Proposition 1) are nonpolyhedral. GD-Cover computes polyhedral (under- and over-) approximations for them accurate up to a user-specified error threshold $\rho \in \mathbb{R}_+$. This will be the maximum Euclidean distance between a nonpolyhedral region (e.g., $\mathcal{U}_s^{\text{range},\,q}$, $\text{bloat}\big(\mathcal{O}, f_{s,q}\big)$, and $\mathcal{U}_{s_1,s_2}^{\text{angle},}$ as defined in Proposition 1, with $s, s_1, s_2 \in \mathcal{S}, q \in \mathcal{Q}$) and its computed polyhedral (under- and over-) approximations.

B. Evaluation of Constraints

The sensor positioning requirements outlined in Section IV break down to a number of problem constraints. GD-Cover evaluates each of them to a positive value when violated (in which case the resulting value is an indication of how much the constraint is violated), and to a zero-or-negative value when satisfied (in which case the resulting value is an indication of how robustly the constraint is satisfied with respect to perturbations of the candidate deployment).

Let q_0 be the lowest sensing quality level in $\mathcal{Q}$. For each sensor $s \in \mathcal{S}$, constraints are as follows.

1) Sensor Is Placed Not Within or Too Close to Obstacles: If $\mathcal{D}(s) \in \text{bloat}\big(\mathcal{O}, f_{s,q_0}\big)$ (i.e., if s is within an obstacle or too close to an obstacle even for the lowest sensing quality level of interest), then the constraint is declared *violated* with cost $\text{dist}\big(\mathcal{D}(s), \mathcal{R} - \text{bloat}\big(\mathcal{O}, f_{s,q_0}\big)\big)$. Otherwise, the constraint is declared satisfied with robustness value: $-\text{dist}(\mathcal{D}(s), \text{bloat}\big(\mathcal{O}, f_{s,q_0}\big))$.

2) Sensor Is Placed Within Its Admissible Region: If $\mathcal{D}(s) \notin \mathcal{A}_s$ (i.e., if s is positioned outside its admissibility region), then the constraint is declared violated with cost $\text{dist}(\mathcal{D}(s), \mathcal{A}_s)$. Otherwise, the constraint is declared satisfied with robustness value: $-\text{dist}(\mathcal{D}(s), \mathcal{R} - \mathcal{A}_s)$.

3) Sensor Is Not Isolated: The constraint evaluates to $d = \min_{s' \in \mathcal{S} - \{s\}}\big(\text{dist}\big(\mathcal{D}(s), \mathcal{D}\big(s'\big)\big) - r_{s,q_0} - r_{s',q_0}\big)$. Thus, if $d > 0$, then the constraint is declared violated with cost d, which is an indication of how much s must be moved to become nonisolated. Otherwise, if $d \leq 0$, then the constraint is declared satisfied with robustness value d, which is an indication of how much s should be moved to become too far with respect to all sensors with which s could now triangulate.

C. Computing Closed-Form Polyhedral Approximations of the Uncovered Region

Here, we show how GD-Cover computes polyhedral representations of $\mathcal{U}^{j,q}$, the regions (j, q)-uncovered (for $j \in [0, k]$ and $q \in \mathcal{Q}$) by the specific deployment $\mathcal{D}$ of sensors $\mathcal{S}$ given as input, as defined in Proposition 1. From this representation, it will be easy to compute any kind of additional quality metrics of the input sensor deployment, hence also any (computable) objective function. This makes our approach extremely flexible (see Section V-D).

Proposition 1 defines region $\mathcal{U}^{j,q}$ as unions of intersections of a number of regions, $\mathcal{U}_{s_1,s_2}^{q}$, one for each pair of distinct sensors s_1 and s_2, from which the region occupied by the obstacles ($\mathcal{O}$) must be removed. Each $\mathcal{U}_{s_1,s_2}^{q}$ in turn is defined by a union of five regions: $\mathcal{U}_{s_1}^{\text{range}}, \mathcal{U}_{s_2}^{\text{range},\,q}, \mathcal{U}_{s_1}^{\mathcal{O},q}, \mathcal{U}_{s_2}^{\mathcal{O},q}$, and $\mathcal{U}_{s_1,s_2}^{\text{angle},}$, which are not polyhedral (see Proposition 1). GD-Cover thus computes polyhedral approximations for them. To overcome the possible errors in such approximations, the tool can compute both polyhedral under- and over-approximations of such regions, using the value ρ given as input (see Section V-A) as tolerance. Such under- and over-approximations in turn allow the tool to compute both a polyhedral under-approximation $\lfloor \mathcal{U}^{j,q} \rfloor$ and a polyhedral over-approximation $\lceil \mathcal{U}^{j,q} \rceil$ of the entire uncovered region $\mathcal{U}^{j,q}$. Hence, $\lfloor \mathcal{U}^{j,q} \rfloor \subseteq \mathcal{U}^{j,q} \subseteq \lceil \mathcal{U}^{j,q} \rceil$.

Thus, points in $\mathcal{R}$ belonging to $\lfloor \mathcal{U}^{j,q} \rfloor$ are certainly (j, q) uncovered by the given sensor deployment, points outside $\lceil \mathcal{U}^{j,q} \rceil$ are certainly (j, q)-covered, while points lying in $\lceil \mathcal{U}^{j,q} \rceil - \lfloor \mathcal{U}^{j,q} \rfloor$ are possibly (j, q)-uncovered, with the uncertainty due to the possible errors introduced when computing polyhedral approximations of $\mathcal{U}_{s_1}^{\text{range},\,q}, \mathcal{U}_{s_2}^{\text{range},\,q}, \mathcal{U}_{s_1}^{\mathcal{O},q}, \mathcal{U}_{s_2}^{\mathcal{O},q}$, and $\mathcal{U}_{s_1,s_2}^{\text{angle},}$ for all pairs of distinct sensors s_1 and s_2. Priorities of $\mathcal{R}$ ($\{\mathcal{R}_h \mid h \in \mathcal{H}\}$, Section III-B) can be straightforwardly considered on top of $\lfloor \mathcal{U}^{j,q} \rfloor$ and $\lceil \mathcal{U}^{j}, q \rceil$: the uncovered portion of the RoI with priority h is sandwiched between $\lfloor \mathcal{U}^{j,q} \rfloor \cap \mathcal{R}_h$ and $\lceil \mathcal{U}^{j,q} \rceil \cap \mathcal{R}_h$.

GD-Cover exploits the C++ Parma Polyhedra Library [41] for the efficient manipulation of convex polyhedra. Unfortunately, most of the needed computations suffer from combinatorial explosion in the worst case. To this end, GD-Cover takes clever countermeasures to handle realistic scenarios efficiently (see Appendix C in the supplementary material for details). For example, when performing operations on intersections of unions of polyhedra (which could need to consider all tuples of polyhedra, one per union being considered, and could result in an algorithm whose time complexity in the worst-case is exponential in the number of such polyhedra) and to efficiently seek polyhedra of interest for the various computations, GD-Cover implements a spatial indexing method based on AABB Trees [42], which greatly reduces the number of operations needed in most cases (see Algorithm 1 in Appendix C in the supplementary material).

To keep the size of the unions of polyhedra manipulated by the algorithm small, and to mitigate the quadratic explosion arising when considering all possible pairs of sensors, GD-Cover performs parallel computation exploiting helper processes via a centralized task dispatcher aimed at keeping load balancing (see Fig. 3 and Algorithm 2 in Appendix C in the supplementary material). Also, GD-Cover partitions the RoI in m identical cells $(\mathcal{R}_1, \ldots, \mathcal{R}_m)$ and delegates again the available helper processes to compute the (j, q)-uncovered portion of each $\mathcal{R}_c$ $(c \in [1, m]), \mathcal{U}_c^{j,q}$. Indeed, computing each single $\mathcal{U}_c^{j,q}$ is way faster than computing the entire $\mathcal{U}^{j,q}$,

because, on average, the sizes of the unions of polyhedra manipulated by the algorithm are smaller and the distances involved (much higher than the range of each sensor) imply that several sensors (and sensor pairs) are too far to possibly contribute to the coverage of the considered cell and can be excluded upfront. As a consequence, the computation of several of the $\mathcal{U}_c^{j,q}$ s, also thanks to the AABB Tree-based spatial indexing, takes negligible time. Dynamic load balancing is dealt with by taking m much larger than the number of helper processes, an approach which is trivial to implement (see [10], [43], [44]). Section VI-C2 experimentally evaluates the scalability of this parallelization technique for the problem at hand.

We also implemented a proof-of-concept Web app which, given the output of GD-Cover (e.g., the uncovered region for the optimal deployment computed by the BBO solver), allows the user to visually and interactively navigate the RoI as a 3-D space, see where sensors are actually planned to be deployed and which portions of the RoI are (un)covered, together with their priorities (details are delayed to Appendix E in the supplementary material). All this enables any interactive analyses of the results, e.g., visually inspecting any dangerous uncovered regions, e.g., worm-shaped corridors which could be used by an attacker to move across the RoI undetected.

D. Statistical Model Checking-Based Estimation of the Objective Value

By computing closed-form polyhedral representations of the uncovered regions $\mathcal{U}^{j,q}$, any objective function can be evaluated by analyzing such regions. However, this computation is an intensive task (see Section VI-C2) and is not strictly needed to effectively guide the optimization process, when only the objective value and an evaluation of the problem constraints are needed by the BBO solvers.

GD-Cover uses statistical model checking techniques (see [45] for a survey) to estimate the value of the objective function (2) efficiently via Monte Carlo sampling while offering statistical guarantees on the accuracy of the approximation. Note that sampling points on a 3-D fixed-step grid, instead of that in the whole (continuous) RoI, would not yield any guarantees on the accuracy of the estimation beyond the grid step length, and would not be as effective in guiding the optimization. In Section VI-C1, we show that the approximation of the objective value requires only a tiny fraction ($\ll$ 1% in our case studies) of the time required to compute the uncovered regions in closed form, and so the objective function exactly. Hence, GD-Cover computes the uncovered regions in closed form only for the final (optimal) deployment and upon explicit user request.

To compute such an approximation of the objective value, GD-Cover uses a Monte Carlo-based algorithm along the lines of [46]. Namely, it combines the EBGStop approximation algorithm [47] and the hypothesis testing technique from [48]. Given values for two parameters, $\varepsilon, \delta \in (0, 1)$, the algorithm computes an (ε, δ)-approximation of the mean value μ of a bounded random variable Z, i.e., a value $\hat{\mu}$ guaranteed to lie within $\mu(1 \mp \varepsilon)$ with probability at least $(1-\delta)$. The algorithm iteratively generates (again exploiting the available pool of parallel helpers, via the intercession of the task dispatcher to handle load balancing among the n_1 optimization processes) i.i.d. samples of Z until the termination condition of [46] is satisfied, which implies that the objective value estimated from the samples is a (ε, δ)-approximation of the true value.

This approach can be used whenever the objective value for a candidate deployment can be expressed as the expected value of a bounded random variable Z. Definition 3 and Observation 1 show that this is the case for the objective function in (2) The complexity of (2) (which considers multiple sensing quality levels as well as fault tolerance) also indirectly shows that such an approach is very flexible, and many other objective functions fall in this class (if not, our BBO-based approach can still be used, but GD-Cover must be asked to compute the uncovered regions in polyhedral form on each candidate deployment to enable the computation of the objective values, leading to longer optimization times).

Definition 3: Given $\mathcal{R}, \mathcal{D}, \mathcal{H}, \mathcal{Q}, \mathcal{U}$, and $\mathcal{R}_h (h \in \mathcal{H})$ as in Section III-E, let $V_{\mathcal{R}} = \text{volume}(\mathcal{R})$ (a constant).

Let also $\left(\Omega^{j,q}, \mathcal{F}^{j,q}, \Pr^{j,q}\right) (j \in [0, k], q \in \mathcal{Q})$ be the probability spaces such that:

1) $\Omega^{j,q} = \{\perp\} \cup \{h \mid h \in \mathcal{H}\}$ is the space of outcomes ($\perp \notin \mathcal{H}$);
2) $\mathcal{F}^{j,q} = 2^{\Omega^{j,q}}$ is the space of events;
3) $\Pr^{j,q} : \mathcal{F}^{j,q} \rightarrow [0, 1]$ is the following probability measure.
   a) $\Pr^{j,q}(\perp) = 1 - ([\text{volume}\left(\mathcal{U}^{j,q}\right)]/V_{\mathcal{R}})$.
   b) $P^{j,q}(h) = ([\text{volume}\left(\mathcal{U}^{j,q} \cap \mathcal{R}_h\right)]/V_{\mathcal{R}})$ for $h \in \mathcal{H}$.
   c) $\Pr^{j,q}(E) = \sum_{\omega \in E} P^{j,q}(\omega)$ for any $E \subseteq \Omega^{j,q}$.

Since $\{\mathcal{R}_h \mid h \in \mathcal{H}\}$ is a partition of $\mathcal{R}$, $\Pr^{j,q}\left(\Omega^{j,q}\right) = 1$.

Observation 1: For every $j \in [0, k], q \in \mathcal{Q}$, let $Z^{j,q}$ be a real-valued random variable defined on probability space $\left(\Omega^{j,q}, \mathcal{F}^{j,q}, P_r^{j,q}\right)$ (Definition 3) as: $Z^{j,q}(\perp) = 0; Z^{j,q}(h) = V_{\mathcal{R}} \times w(j, q, h)$ (for $h \in \mathcal{H}$).

The value of the objective function (2) evaluated for deployment $\mathcal{D}$ is $\text{Placem}(\mathcal{D}) + \sum_{j=0}^{k} \sum_{q \in \mathcal{Q}} \mathbb{E}\left(Z^{j,q}\right)$, where $\mathbb{E}(Z^{j,q})$ is the expected value of $Z^{j,q}$.

Observation 1 is proved in Appendix B in the supplementary material. Random variables $Z^{j,q}$, being bounded, clearly meet the requirements for the application of the statistical model checking algorithm described above. To generate i.i.d. samples for $Z^{j,q}$, each delegated helper (running in parallel) samples points $X \in \mathcal{R}$ uniformly at random. For each sample X, the helper determines whether X is (j, q)-covered for every j and q (or falls within an obstacle, notation $\text{cover}^{j,q}(X) = 1$) or not $\left(\text{cover}^{j,q}(X) = 0\right)$ by deployment $\mathcal{D}$. This is implemented by exploiting standard polyhedral geometry operations (note that all conditions of Definitions 1 and 2 can be checked in this way, simply by looping through the set of pairs of sensors). Value for each random variable $Z^{j,q}$ is computed from X as follows: $Z^{j,q} = V_{\mathcal{R}} \times w(j, q, h) \times \left(1 - \text{cover}^{j,q}(X)\right)$, where h is the (single) priority value of point X.

VI. Experiments

We exercised our parallel system by computing optimal anti-drone sensor deployments on two real-world case studies, the FCO in Rome, Italy, and the VIC in Vienna, Austria, described below, having complementary properties: while the

former consists in a large environment with wide open spaces and relatively low obstacles, the latter presents several tall buildings condensed in a small area. In all experiments, we used NOMAD v. 3.9.1 as our BBO solver.

A. Experimental Setting

1) Sensors: For each case study, we considered sensors of two types (namely, T1 and T2), with different characteristics (chosen in accordance to reference values from the literature, e.g., [1]), and costs. For generality, we normalized all costs for each case study to the price of a single sensor of type T1 (the cheapest type) simply installed on a pole on the ground, at a height between 5 and 10 m (the cheapest installation). We sought for optimal fault-tolerant quality-guaranteed coverage with two sensing quality levels ($\mathcal{Q} = \{q_0, q_1\}$, with $q_0 < q_1$) and one possible sensor fault (i.e., $k = 1$).

2) Priority Regions: The RoI of each case study was partitioned into two priority regions (low and high priority).

3) Weights of the Objective Function: Values of $w(j, q, h)$ (the cost of not (j, q)-covering a unit of volume of the region having priority h, Section III-E) are reported in Appendix D-A2 in the supplementary material. For example, in FCO, weights model indifference criteria such as: an increase of one volume unit of the $(0, q_1)$-covered high priority region ($w(0, q_1, \text{high}) = 20$) is equally exchanged with an increase of two volume units of the $(0, q_0)$-covered low priority region ($w(0, q_0, \text{low}) = 10$).

4) Approximation Thresholds and Initial Deployments: The thresholds ε and δ used by GD-Cover to estimate the objective value for each candidate deployment produced during optimization were both set to 1%, thus guaranteeing that, with probability at least 99%, the estimated objective value is within a 1% error margin from its true value. Threshold ρ used to compute polyhedral under- and over-approximations of the regions mentioned in Section V-A was set to 10 m. Finally, the number of random deployments generated and fed as starting points to the multiple BBO solvers was set to $N = 100$.

5) Computational Infrastructure: Experiments were performed on a cluster of identical machines, each one equipped with 2 AMD EPYC 7301 CPUs (overall 64 cores) and 256-GB RAM. Our loosely coupled architecture (Fig. 3) is particularly suited for off-premise clusters as ours, which are shared among a high number of competing processes (a common paradigm aimed at keeping the cost of running parallel software low). Below, we present the completion times that would be obtained in three reference infrastructures, namely, if fully reserving the following number of machines: 1) 1 machine ($n = 64$ nodes, using $n_1 = 5$ BBO solvers and the remaining $n_2 = 57$ nodes as GD-Cover helpers); 2) 10 machines ($n = 640$ nodes, $n_1 = 20, n_2 = 618$); and 3) 50 machines ($n = 3200$ nodes, $n_1 = 100, n_2 = 3098$). A full scalability analysis is delayed to Appendix D-B in the supplementary material.

B. Case Studies

1) Leonardo Da Vinci International Airport: This is a prototypical example of a critical area, with a total surface of approximately 16 km^2, most of which is taken by the three 4-km long runways. We created a simplified 3-D model of the RoI as the union of 11 polyhedra with a height of 100 m. The total volume to be monitored is thus 1.6 km^3. The case study presents many obstacles, modeled with 52 polyhedra, ranging from large buildings, such as hangars and terminals to small service buildings. Section I shows an aerial view of FCO, while Fig. 6 shows screenshots of our visualizer with our 3-D model.

Sensor costs are 1 (for type T1, reference cost) and 1.5 (T2). Sensors can be placed everywhere on the ground (except for the runways), and over the walls and roofs of (most of) the obstacles (in the latter cases with cost overheads of 10% and 20%, respectively). Sensors of type T1 (respectively, T2) can detect targets distant up to 1000 m (respectively, 1250 m) with quality level q_0, and up to 900 m (respectively, 1110 m) with quality level q_1. Sensing angle ranges for any sensor pair are $[25°, 155°]$ (for q_0) and $[30°, 150°]$ (for q_1). All sensors have the same FFZ (5 m) for both q_0 and q_1. The high-priority portion of the RoI includes the runways and the space above them, while the remaining region is low priority.

2) Vienna International Center: This is a complex of several buildings that hosts the headquarters of important organizations of the United Nations. Its political and economic relevance makes this environment a critical area. VIC has a substantially different structure than FCO: while the total volume is only a fraction of the volume of FCO, the RoI is much more densely occupied by tall buildings. This makes the optimal placement problem harder, since each sensor will only cover a small portion of the region, regardless of its position, due to lack of line-of-sight visibility. Our 3-D RoI model is a cube with edges of length 400 m containing simplified shapes for all the relevant buildings (modeled with 51 polyhedra).

Sensor costs are 1 (type T1, reference cost) and 1.17 (T2). All sensors can be placed on the ground, and T2 sensors also over the roofs and on the concrete walls of the towers, i.e., not on the glass façades (in the latter cases with cost overheads of 5% and 10%, respectively). Sensors of type T1 (respectively, T2) can detect targets up to 500 m (respectively, 700 m) with quality level q_0, and up to 400 m (respectively, 600 m) with quality level q_1. Sensing angle ranges for any sensor pair are $[25°, 155°]$ (for q_0) and $[30°, 150°]$ (for q_1). All sensors have the same FFZ (5 m) for both q_0 and q_1.

The high-priority portion of the RoI is composed of two parts: the first one is a portion of a spherical shell on top of the buildings, constituting a sort of dome on the RoI; such region is deemed as highly important because, in such a small environment (densely occupied by buildings), it is crucial to detect the arrival of a UAV as quickly as possible. The second portion includes the area at ground level, that is the busiest area where people walk to enter the buildings.

Section I shows an aerial view of VIC, while Fig. 6 shows screenshots of our visualizer with our 3-D model.

C. Experimental Results

1) Optimization: We ran multiple experiments for each case study, and with various configurations, where each

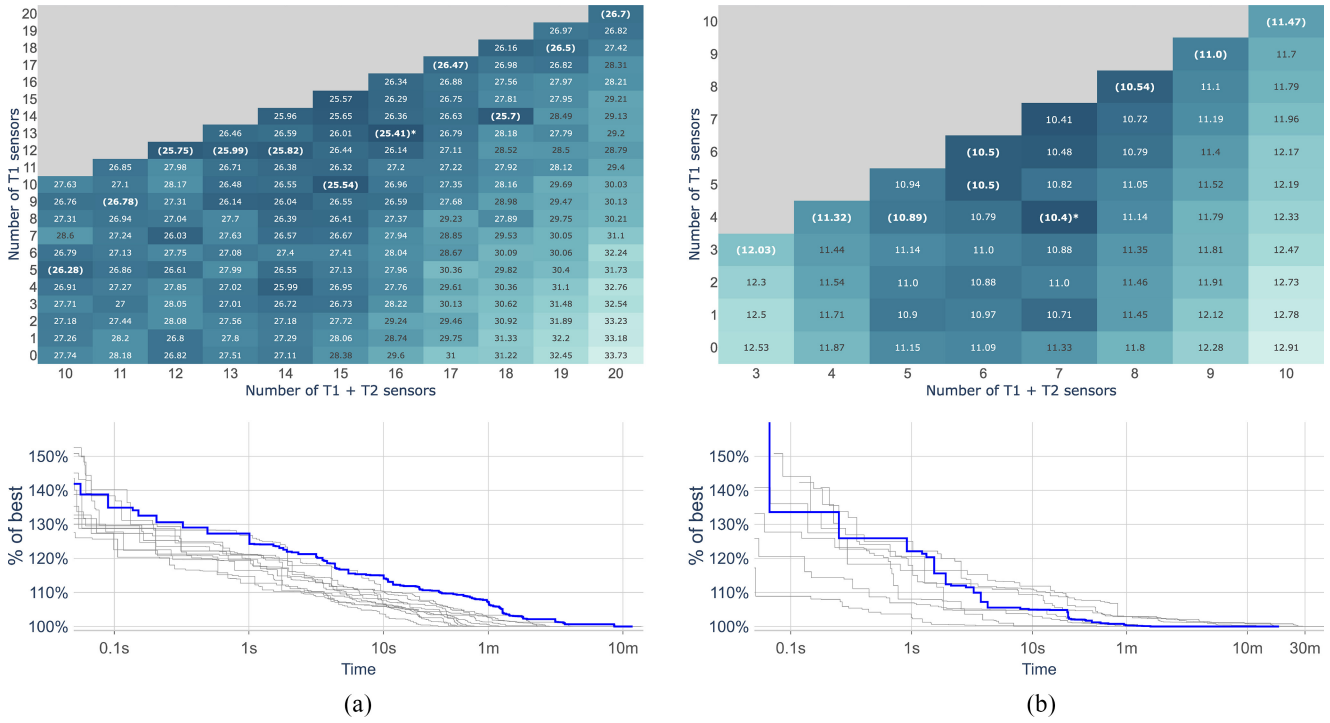

Fig. 5. Up: Best ODC found for different numbers of T1 versus T2 sensors (configurations). Down: Time course of the objective value during optimization, when fully reserving 640 nodes (i.e., ten machines, infrastructure 2) in Section VI-A5; one line per configuration; the blue lines refer to optimal configurations). (a) FCO. (b) VIC.

configuration defines an overall number of sensors and the relative numbers of T1 versus T2 sensors. We used between 10 and 20 sensors for FCO and between 3 and 10 for VIC. Indeed, preliminary experiments showed that higher numbers of sensors would not bring any significant further improvement of the coverage, while fewer sensors would simply be ineffective.

Fig. 5 (up) shows the ODCs of the best deployments found for each configuration (T1 versus T2 sensors). The darker a cell in the heat maps, the better the quality of the final deployment found for that configuration. Numbers in cells denote the objective value (2) of the optimal deployments found. Values in parenthesis denote the best solution for each number of sensors. The value with a "*" (in the darkest cell) denotes the best solution overall.

Our system achieved a final (optimal) deployment yielding, on average across the various configurations, an expected reduction of the ODC of 27.37% (FCO) and 26.69% (VIC), where the expectation is computed with respect to the N initial random generated admissible deployments (time zero). Expected ODC reductions on the best configurations are of 31.89% (FCO) and 32.01% (VIC).

Fig. 5 (down) shows, for each configuration and each timepoint t, the objective value of the best deployment that our system (when running on infrastructure 2), i.e., on 640 nodes fully reserved for the job) would have found if halted at time t. The bold curves refer to the configurations of T1 versus T2 sensors yielding the overall optimum, namely, 13 T1 and 3 T2 sensors for FCO, and 4 T1 and 3 T2 sensors for VIC (see Fig. 5, up). The objective values for each configuration have been normalized as percentages of the objective value of the final deployment found for that configuration. The system terminated in 11m45s (FCO) and 18m18s (VIC), and found a solution whose objective value is just 10% above the final optimum in only 34 s (FCO) and 6 s (VIC).

An analysis of the scalability of our parallel system on fully reserved infrastructures of various sizes is delayed to Appendix D-B in the supplementary material. Here, we just mention that, on infrastructure 3) (i.e., using 3200 nodes), it would have terminated in only 3m5s (FCO) and 3m40s (VIC), and would have found a solution whose objective value is 10% above the final optimum in only 9 s (FCO) and 5 s (VIC). Conversely, on infrastructure 1) (i.e., using 64 nodes, i.e., just one machine), it would have required 2h2m36s (FCO) and 3h16m23s (VIC) to terminate, and 2m29s (FCO) and 22 s (VIC) to get to 10% above the final optimum.

The time required by GD-Cover to estimate, via statistical model checking, the objective value yielded by each candidate deployment ranges within 1.8–8.5 s (FCO, 10–20 sensors) and 2–32 s (VIC, 3–10 sensors), and grows with the number of pairs of sensors that might triangulate. The higher times measured in VIC (which has a smaller RoI than FCO, although with more obstacles) are indeed due to the fact that each sensor can in principle cooperate with most of the others. Hence, the number of pairs of sensors which might triangulate is much higher, and determining the coverage of each sampled point requires more effort. However, deploying GD-Cover using a high enough number of helper processes successfully mitigates this issue. Details are delayed to Appendix D-D in the supplementary material.

Finally, Appendix D-C in the supplementary material evaluates the effectiveness of launching the parallel optimizers

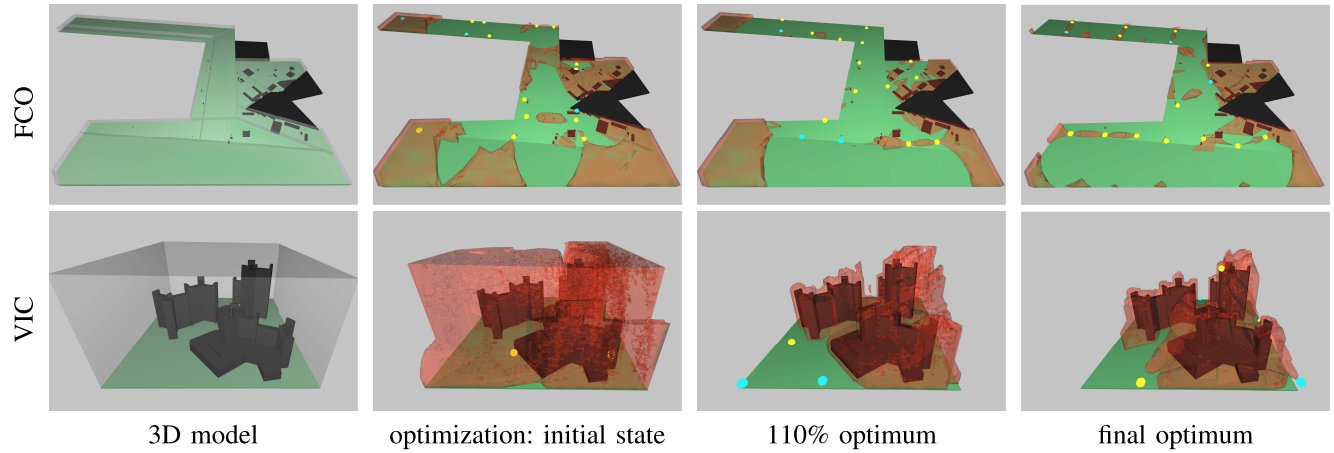

Fig. 6. 3-D models of our case studies and computation of optimal deployments at different states of search (green: RoI; red: uncovered region under no sensors faults).

starting from the best initial deployments. This step greatly helps when in the presence of tight time budgets and small computational infrastructures. Namely, for both case studies, the first deployment within 110% of the final optimum was achieved from a parallel NOMAD run starting from one of the 4 best random initial deployments, and the final optimum from one of the 20 best initial deployments (with one single exception).

2) Computation of Uncovered Region in Closed Form: This job is computationally way more intensive than estimating the objective value of a candidate deployment via statistical model checking. This is why such computation is performed only at the end of the optimization process, or on user demand.

We ran GD-Cover to compute the uncovered region for the final (optimal) deployments. As explained in Section V-C, we enabled parallel computation also for this job, by splitting the two RoIs in different numbers of identical cells, which were processed in parallel by the available helper processes. A full scalability analysis for this job is delayed to Appendix D-B2 in the supplementary material. Here, we just mention that, differently from what happens during optimization, the geometric reasoning required by the computation of the uncovered region is somewhat hindered when using large infrastructures. This is unsurprising, since processing too many small cells may yield duplication of efforts, because some of the computed polyhedra (those needed to represent the regions of Proposition 1 would span a high number of cells. The optimal splitting of each RoI appears to be in a few thousands of cells. This yields the computation terminate in 31m33s (FCO, 6400 cells of size $100 \times 100 \times 10$ m^3) and 14m5s (VIC, 6498 cells of size $21 \times 21 \times 22$ m^3) on infrastructure 1) (64 nodes). By comparing these durations with those of the statistical model checking-based estimation of the objective value (which is also much more suitable to be massively parallelized), we see that driving the optimization with such approximations is extremely beneficial.

Finally, Fig. 6 shows, for each case study, the computed positions of the sensors and the uncovered region at three milestones of the optimization process (when using the best number of sensors): the initial deployment (time zero), the first deployment whose objective value is 10% above the final optimum, and the final (optimal) deployment. The pictures are taken from our 3-D visualizer.

VII. Conclusion

In this article, we proposed a novel approach to the computation of optimal (with respect to coverage, cost-effectiveness, multiple sensing quality levels, and tolerance to sensor faults) deployments of triangulating sensors for unauthorized UAV localization in large, complex critical 3-D regions exhibiting obstacles (e.g., buildings), varying terrain elevation, portions with different coverage priorities, and in the presence of constraints on where sensors can actually be placed. Our approach relies on computational geometry and statistical model checking, effectively exploiting off-the-shelf AI-based BBO solvers and enables the computation of a closed-form, analytical representation of the region uncovered by a sensor deployment, which provides the means for rigorous, formal certification of the quality of the latter. To our knowledge, no other method is available which addresses all such aspects (see also the recap in Appendix A in the supplementary material).

We have demonstrated the practical feasibility of our approach by computing, in a *few minutes on a small parallel infrastructure* (or a few hours on a single workstation), optimal sensor deployments for UAV localization in two large, complex regions, FCO in Rome and VIC, using multiple instances of the NOMAD state-of-the-art AI-based BBO solver as the underlying optimization engine.

Future work includes the evaluation of other (e.g., derivative-free) optimization techniques and the improvement of GD-Cover to further reduce its computation time.

Acknowledgment

The authors are grateful to the anonymous reviewers as well as to G. Nazzaro and C. Giannetti for their support in implementing their 3-D visualizer.

References

[1] X. Shi, C. Yang, W. Xie, C. Liang, Z. Shi, and J. Chen, "Anti-drone system with multiple surveillance technologies: Architecture, implementation, and challenges," *IEEE Commun. Mag.*, vol. 56, no. 4, pp. 68–74, Apr. 2018.

[2] I. Guvenc, F. Koohifar, S. Singh, M. Sichitiu, and D. Matolak, "Detection, tracking, and interdiction for amateur drones," *IEEE Commun. Mag.*, vol. 56, no. 4, pp. 75–81, Apr. 2018.

[3] S. Jun, T.-W. Chang, and H.-J. Yoon, "Placing visual sensors using heuristic algorithms for bridge surveillance," *App. Sci.*, vol. 8, no. 1, p. 70, 2018.

[4] A. Saad, M. Senouci, and O. Benyattou, "Toward a realistic approach for the deployment of 3D wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 21, no. 4, pp. 1508–1519, Apr. 2022.

[5] H. R. Topcuoglu, M. Ermis, and M. Sifyan, "Positioning and utilizing sensors on a 3-D terrain part I—Theory and modeling," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 41, no. 3, pp. 376–382, May 2011.

[6] C. Audet and W. Hare, *Derivative-Free and Blackbox Optimization*. Cham, Switzerland: Springer, 2017.

[7] J. Larson, M. Menickelly, and S. M. Wild, "Derivative-free optimization methods," *Acta Numerica*, vol. 28, pp. 287–404, Jun. 2019.

[8] K. Vu, C. D'Ambrosio, Y. Hamadi, and L. Liberti, "Surrogate-based methods for black-box optimization," *Int. Trans. Oper. Res.*, vol. 24, no. 3, pp. 393–424, 2017.

[9] S. Le Digabel, "Algorithm 909: NOMAD: Nonlinear optimization with the MADS algorithm," *ACM Trans. Math. Softw.*, vol. 37, no. 4, pp. 1–15, 2011.

[10] Q. Chen, A. Finzi, T. Mancini, I. Melatti, and E. Tronci, "MILP, pseudo-Boolean, and OMT solvers for optimal fault-tolerant placements of relay nodes in mission critical wireless networks," *Fundamenta Informaticae*, vol. 174, nos. 3–4, pp. 229–258, 2020.

[11] V. Akbarzadeh, J.-C. Lévesque, C. Gagné, and M. Parizeau, "Efficient sensor placement optimization using gradient descent and probabilistic coverage," *Sensors*, vol. 14, no. 8, pp. 15525–15552, 2014.

[12] E. Amaldi, A. Capone, M. Cesana, and I. Filippini, "Coverage planning of wireless sensors for mobile target detection," in *Proc. MASS*, 2008, pp. 48–57.

[13] K. An, S. Xie, and Y. Ouyang, "Reliable sensor location for object positioning and surveillance via trilateration," *Transp. Res. B, Methodol.*, vol. 117, pp. 956–970, 2018.

[14] B. Cao, J. Zhao, P. Yang, P. Yang, X. Liu, and Y. Zhang, "3-D deployment optimization for heterogeneous wireless directional sensor networks on smart city," *IEEE Trans. Ind. Informat.*, vol. 15, no. 3, pp. 1798–1808, Mar. 2019.

[15] K. Zaimen, M.-E.-A. Brahmia, L. Moalic, A. Abouaissa, and L. Idoumghar, "A survey of artificial intelligence based WSNs deployment techniques and related objectives modeling," *IEEE Access*, vol. 10, pp. 113294–113329, 2022.

[16] D. Sikeridis, E. Tsiropoulou, M. Devetsikiotis, and S. Papavassiliou, "Wireless powered public safety IoT: A UAV-assisted adaptive-learning approach towards energy efficiency," *J. Netw. Comput. Appl.*, vol. 123, pp. 69–79, 2018.

[17] W. Wang et al., "Placement of unmanned aerial vehicles for directional coverage in 3D space," *IEEE/ACM Trans. Netw.*, vol. 28, no. 2, pp. 888–901, Apr. 2020.

[18] M. Basharat, M. Naeem, Z. Qadir, and A. Anpalagan, "Resource optimization in UAV-assisted wireless networks—A comprehensive survey," *Trans. Emerg. Telecommun. Technol.*, vol. 33, no. 7, 2022, Art. no. e4464.

[19] Z. Na, M. Zhang, J. Wang, and Z. Gao, "UAV-assisted wireless powered Internet of Things: Joint trajectory optimization and resource allocation," *Ad Hoc Netw.*, vol. 98, Mar. 2020, Art. no. 102052.

[20] N. Parvaresh, M. Kulhandjian, H. Kulhandjian, C. D'Amours, and B. Kantarci, "A tutorial on AI-powered 3D deployment of drone base stations: State of the art, applications and challenges," *Veh. Commun.*, vol. 36, Aug. 2022, Art. no. 100474.

[21] A. Altahir et al., "Visual sensor placement based on risk maps," *IEEE Trans. Inst. Meas.*, vol. 69, no. 6, pp. 3109–3117, Jun. 2020.

[22] L. Dai and B. Wang, "Sensor placement based on an improved genetic algorithm for connected confident information coverage in an area with obstacles," in *Proc. LCN*, 2017, pp. 595–598.

[23] M. P. Fanti, G. Faraut, J.-J. Lesage, and M. Roccotelli, "An integrated framework for binary sensor placement and inhabitants location tracking," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 48, no. 1, pp. 154–160, Jan. 2018.

[24] X. Yang, H. Li, T. Huang, X. Zhai, F. Wang, and C. Wang, "Computer-aided optimization of surveillance cameras placement on construction sites," *Comput.-Aided Civil Infrastruct. Eng.*, vol. 33, no. 12, pp. 1110–1126, 2018.

[25] Y. Yoon and Y.-H. Kim, "Maximizing the coverage of sensor deployments using a memetic algorithm and fast coverage estimation," *IEEE Trans. Cybern.*, vol. 52, no. 7, pp. 6531–6542, Jul. 2022.

[26] Y.-G. Fu, J. Zhou, and L. Deng, "Surveillance of a 2D plane area with 3D deployed cameras," *Sensors*, vol. 14, no. 2, pp. 1988–2011, 2014.

[27] H. Topcuoglu, M. Ermis, and M. Sifyan, "Positioning and utilizing sensors on a 3-D terrain part II—Solving with a hybrid evolutionary algorithm," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 41, no. 4, pp. 470–480, Jul. 2011.

[28] S. Temel, N. Unaldi, and O. Kaynak, "On deployment of wireless sensors on 3-D terrains to maximize sensing coverage by utilizing cat swarm optimization with wavelet transform," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 44, no. 1, pp. 111–120, Jan. 2014.

[29] L. Kloeker, J. Quakernack, B. Lampe, and L. Eckstein, "Generic approach to optimized placement of smart roadside infrastructure sensors using 3D digital maps," in *Proc. ITSC*, 2022, pp. 1311–1316.

[30] S. Jun, T.-W. Chang, H. Jeong, and S. Lee, "Camera placement in smart cities for maximizing weighted coverage with budget limit," *IEEE Sensors J.*, vol. 17, no. 23, pp. 7694–7703, Dec. 2017.

[31] E. Becker, G. Guerra-Filho, and F. Makedon, "Automatic sensor placement in a 3D volume," in *Proc. PETRA*, 2009, pp. 1–8.

[32] S. Indu, S. Chaudhury, N. Mittal, and A. Bhattacharyya, "Optimal sensor placement for surveillance of large spaces," in *Proc. ICDSC*, 2009, pp. 1–8.

[33] M. Zhang, J. Yang, and T. Qin, "An adaptive three-dimensional improved virtual force coverage algorithm for nodes in WSN," *Axioms*, vol. 11, no. 5, p. 199, 2022.

[34] J. Dybedal and G. Hovland, "GPU-based occlusion minimisation for optimal placement of multiple 3D cameras," in *Proc. ICIEA*, 2020, pp. 967–972.

[35] A. Albahri and A. Hammad, "Simulation-based optimization of surveillance camera types, number, and placement in buildings using BIM," b*J. Comput. Civil Eng.*, vol. 31, no. 6, 2017, Art. no. 4017055.

[36] B. Cao, J. Zhao, Z. Lv, and X. Liu, "3D terrain multiobjective deployment optimization of heterogeneous directional sensor networks in security monitoring," *IEEE Trans. Big Data*, vol. 5, no. 4, pp. 495–505, Dec. 2019.

[37] A. Kelly, "Precision dilution in triangulation based mobile robot position estimation," in *Proc. IAS*, 2003, pp. 1046–1053.

[38] M. Cadoli and T. Mancini, "Combining relational algebra, SQL, constraint modelling, and local search," *Theory Pract. Logic Program.*, vol. 7, nos. 1–2, pp. 37–65, 2007.

[39] T. Mancini, P. Flener, and J. Pearson, "Combinatorial problem solving over relational databases: View synthesis through constraint-based local search," in *Proc. SAC*, 2012, pp. 80–87.

[40] T. Mancini, "Now or Never: Negotiating efficiently with unknown or untrusted counterparts," *Fundamenta Informaticae*, vol. 149, nos. 1–2, pp. 61–100, 2016.

[41] R. Bagnara, P. Hill, and E. Zaffanella, "The parma polyhedra library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems," *Sci. Comput. Progr.*, vol. 72, nos. 1–2, pp. 3–21, 2008.

[42] G. V. D. Bergen, "Efficient collision detection of complex deformable models using AABB trees," *J. Graph. Tools*, vol. 2, no. 4, pp. 1–13, 1997.

[43] T. Mancini, F. Mari, A. Massini, I. Melatti, and E. Tronci, "SyLVaaS: System level formal verification as a service," *Fundamenta Informaticae*, vol. 149, nos. 1–2, pp. 101–132, 2016.

[44] S. Sinisi, V. Alimguzhin, T. Mancini, E. Tronci, and B. Leeners, "Complete populations of virtual patients for in silico clinical trials," *Bioinformatics*, vol. 36, nos. 22–23, pp. 5465–5472, 2020.

[45] G. Agha and K. Palmskog, "A survey of statistical model checking," *ACM Trans. Model. Comput. Simulat.*, vol. 28, no. 1, p. 6, 2018.

[46] T. Mancini et al., "Parallel statistical model checking for safety verification in smart grids," in *Proc. SmartGridComm*, 2018, pp. 1–6.

[47] V. Mnih, C. Szepesvári, and J. Audibert, "Empirical Bernstein stopping," in *Proc. ICML*, 2008, pp. 672–679.

[48] R. Grosu and S. Smolka, "Monte Carlo model checking," in *Tools and Algorithms for the Construction and Analysis of Systems* (Lecture Notes in Computer Science 3440). Heidelberg, Germany: Springer, 2005, pp. 271–286.

Open Access funding provided by 'Università degli Studi di Roma "La Sapienza" 2' within the CRUI CARE Agreement