Rapid Top-Down Synthesis of Large-Scale IoT Networks

Pradipta Ghosh*, Jonathan Bunton†, Dimitrios Pylorof‡, Marcos Vieira§, Kevin Chan¶,
Ramesh Govindan*, Gaurav Sukhatme*, Paulo Tabuada† and Gunjan Verma¶
*Department of Computer Science, University of Southern California, Los Angeles, CA - 90089
†Department of Electrical and Computer Engineering, University of California Los Angeles, Los Angeles, CA
‡Georgetown University, Washington, DC 20057
§Computer Science Department at Federal University of Minas Gerais, Belo Horizonte - MG, 31270-901, Brazil
¶US Army Research Laboratory
Email: pradiptg@usc.edu, jonathan.m.bunton@gmail.com, dpylorof@austin.utexas.edu,
marcos.augusto.vieira@gmail.com, kevin.s.chan.civ@mail.mil, ramesh@usc.edu,
gaurav@usc.edu, tabuada@ucla.edu, gunjan.verma.civ@mail.mil

***Abstract*—Advances in optimization and constraint satisfaction techniques, together with the availability of elastic computing resources, have spurred interest in large-scale network verification and synthesis. Motivated by this, we consider the top-down synthesis of ad-hoc IoT networks for disaster response and search and rescue operations. This synthesis problem must satisfy complex and competing constraints: sensor coverage, line-of-sight visibility, and network connectivity. The central challenge in our synthesis problem is quickly *scaling* to large regions while producing cost-effective solutions. We explore a representation of the synthesis problems using a novel constraint satisfaction paradigm, satisfiability modulo convex optimization (SMC). We choose SMC because it matches the expressivity needs for our network synthesis. To scale to large problem sizes, we develop a *hierarchical synthesis* technique that independently synthesizes networks in sub-regions of the deployment area, then combines these. Our experiments show that SMC consistently generates better quality solutions than a baseline synthesis approach based on Mixed Integer Linear Programming (MILP).**

I. Introduction

Over the last few years, optimization and constraint satisfaction technologies have improved to the point where they can be applied to large-scale verification and synthesis tasks. This, coupled with the advent of cloud computing, has spurred recent work in verifying network configurations of campuses [1], [2] and data centers [3]. Beyond verification, these technologies have been applied to the synthesis of network configurations as well [4]. Ultimately, this line of work will result in the synthesis of correct-by-design networks.

The problem. In this paper, we extend this line of work by considering the *synthesis of a specific class of network topologies, namely, ad-hoc IoT networks*. These networks, often deployed on-demand (in a *deployment region*) during disaster response or search and rescue operations, must satisfy at least four *correctness* constraints (§II). First, every point of the deployment region must be covered by a small number (k) of sensors (the *coverage constraint*). This k-coverage permits trilateration (for $k \geq 3$) of objects of interest (vehicles, people) in the environment. Second, because sensors may have line-of-sight limitations, the synthesis must account for obstacles in the deployment region (*visibility constraints*). Third, obstacles constrain the placement of sensors (*placement constraint*). Lastly, the deployed nodes must form a connected network such that there exists a network path between every pair of nodes (we call this the *connectivity constraint*). This is necessary to convey sensor information between nodes or from node to a base station (potentially via other intermediate nodes).

Beyond these correctness constraints, we impose *performance* objectives. Motivated by our use case of rapidly-deployable IoT networks for disaster response or search and rescue, we require that the network synthesis (for brevity, henceforth we will use "network synthesis" to refer to the specific synthesis problem described above) *scale* to campus-area deployments while generating topologies on the order of *tens of minutes*. (By contrast, planning and deployment timescales for longer-lived networks, such as datacenters, can be on the order of weeks or months, if not longer [5]). Finally, to reduce deployment cost, the resulting synthesized network must have an objective of *using the fewest number of sensor nodes*.

While prior work (§VII) has explored the synthesis of wired networks [6], of sensor placement for sensing coverage [7], [8], to our knowledge, no prior work has explored joint synthesis of coverage and connectivity in obstructed environments at scale.

Goal and Approach. The general network synthesis problem is non-convex, since it needs to determine coverage of areas not occupied by obstacles, and these areas can be non-convex.

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

This material is based upon work supported in part by the National Science Foundation under Grant No. CNS 1901523

Because non-convex programming is fundamentally hard to scale, and because, over the past decade, significant effort has focused on speeding up solvers for lower-complexity formulations such as satisfiability (SAT) [9] and linear and quadratic mixed integer formulations [10], in this paper we explore synthesis using a recently proposed technique called Satisfiability Modulo Convex Optimization [11] that permits expression of convex constraints in addition to satisfiability constraints (§III).

Challenges. In expressing the network synthesis problem using SMC, our first challenge is to formulate the constraints described above in SMC, without significantly impacting *solution quality*. In our setting, we measure solution quality by *coverage redundancy*, the ratio between the average number of sensors covering a point in the deployment region to k. Our second challenge is *scaling*, by which we mean the ability to obtain solutions for larger problem sizes (*e.g.,* for a deployment region the size of a university campus).

Contributions. To this end, our paper makes two contributions. Our first contribution is an encoding of constraints in SMC (§IV). To address the first challenge discussed above, we employ a *grid-based* discretization of space, in which a grid is either occupied by an obstacle or not. Then, the coverage problem reduces to determining coverage of unoccupied grids; SMC can easily express these constraints. It can capture visibility using geometric constraints expressed in terms of half-planes that bound occupied grids, and network connectivity using constraints on a multi-hop adjacency matrix. Thus, SMC is able to jointly solve for coverage, visibility, and connectivity.

Our second contribution addresses the scaling challenge by developing a *hierarchical synthesis* technique, which partitions the deployment region in sub-areas, solves each sub-area independently, then, in a second step, adds additional *relay* nodes, using SMC's connectivity formulation, to ensure the network connectivity constraint. Hierarchical synthesis permits two relaxations that can speed up the synthesis: (a) solving only for coverage in the sub-areas (since, often, coverage will result in a substantially connected network) and establishing connectivity in the second step, and (b) solving for $j < k$ coverage in each of the sub-areas (since sensors in one sub-area can potentially cover sensors in another area), then repairing coverage in the second step.

Summary of Results. To analyze the performance of SMC, we also develop an MILP formulation (with hierarchy) of the synthesis problem with integer linear constraints[1]. The performance of SMC and MILP approaches is sensitive to the *extent* of obstacles (*i.e.,* the fraction of the deployment region covered by obstacles) and their *dispersion* (how the obstacles are spread across the region). While MILP can generate a network with slightly lower coverage redundancy (*i.e.,* a better solution quality) for few obstacle setting with small deployment regions, SMC performs consistently well for all settings regardless of the deployment region size.

[1]We omit details of the MILP formulation in this paper for lack of space.

II. The Network Synthesis Problem

In this section, we formulate the network synthesis problem and its associated challenges.

Problem. Assume that we have a *deployment region* (the area in which to deploy the network) $\mathbf{L} \subset \mathbb{R}^2$. Let $\mathbf{O} \subset \mathbf{L}$ be the set of obstacles in the deployment region. Suppose we have N omni-directional sensors (*e.g.,* 360-degree cameras, acoustic sensors), and each sensor with location $s_i \in \mathbb{R}^2$ has a sensing radius r_i^s and a communication radius r_i^c.

The network synthesis problem seeks to *find locations* $\mathbf{S} = \{s_i \in \mathbb{R}^2 | i \in \{1, \cdots, N\}\}$ for the N sensors such that:

1) At least k sensors cover each point $l \in \mathbf{L} \setminus \mathbf{O}$.
2) The sensors form a connected network: *i.e.,* there exists a path from each node to every other node in the network.
3) The network uses as few of the N sensors as possible.

To concretely represent this synthesis problem, we need to model constraints imposed by limited sensing and communication ranges, as well as by visibility constraints in the environment. The following sub-paragraphs do this.

Coverage constraint (C1). Let $\text{dist}(x_i, x_j) = \|x_i - x_j\|_2$ represent the Euclidean distance between $x_i, x_j \in \mathbb{R}^2$, any two locations in the deployment region $\mathbf{L}$. The predicate $\mathcal{C}(s_i, l_j) : \mathbb{R}^2 \times \mathbb{R}^2 \rightarrow \{0, 1\}$ represents the fact that sensor $s_i \in \mathbf{S}$ *covers* location $l_j \in \mathbf{L} \setminus \mathbf{O}$:

$$\mathcal{C}(s_i, l_j) = \left(\text{dist}(s_i, l_j) \leq r_i^s\right) \bigwedge \mathcal{B}(s_i, l_j) \tag{1}$$

where r_i^s is the coverage radius of the sensor s_i. $\mathcal{B}(s_i, l_j)$ is a predicate that checks whether there exist a line of sight between s_i and l_j (defined below). Line-of-sight visibility constraints are important for sensors such as cameras. Furthermore, in practice, cameras have finite range because of the finite resolution of the camera itself: beyond a certain distance, objects become too small to be distinguishable to the human eye [12]. The k-coverage goal then reduces to:

$$\sum_{i=1}^{N} \mathcal{C}(s_i, l_j) \geq k \quad \forall \;\; l_j \in \mathbf{L} \setminus \mathbf{O} \tag{2}$$

This ensures that at least k sensors cover each location within the deployment region but not within an obstacle.

Environmental Visibility Constraints (C2). Obstacles pose a significant challenge for network synthesis. We explore synthesis for planar surfaces for which it suffices to model obstacles in two dimensions. Future work can generalize this to 3-D models of the environment.

Let $\mathcal{B}^o(s_i, l_j, o)$ be a predicate that checks whether obstacle o does *not* block the line of sight between s_i and l_j. Then, we can model the visibility between s_i and l_j, defined by the predicate $\mathcal{B}(s_i, l_j)$ as:

$$\mathcal{B}(s_i, l_j) = \bigwedge_{o \in \mathbf{O}} \mathcal{B}^o(s_i, l_j, o) \tag{3}$$

TABLE I: Summary of Problem Formulation

Coverage	(C1)	$\sum_{i=1}^{N} \mathcal{C}(s_i, l_j) \geq k \quad \forall \; l_j \in \mathbf{L} \setminus \mathbf{O}$
Visibility	(C2)	$\mathcal{B}(s_i, l_j) = \bigwedge_{o \in \mathbf{O}} \mathcal{B}(s_i, l_j, o)$ $\forall s_i \in \mathbf{S}$ and $l_j \in \mathbf{L} \setminus \mathbf{O}$
Connectivity	(C3)	$(\hat{\mathbf{A}}_{N-1})_{ij} > 0$ $\forall i, j \in \{1, \cdots, N\} \;\; and \;\; i \neq j$
Placement	(C4)	$\mathcal{V}(s_i) \quad \forall s_i \in \mathbf{S}$

Network connectivity constraint (C3). If predicate $\mathcal{P}(s_i, s_j) : \mathbb{R}^2 \times \mathbb{R}^2 \rightarrow \{0, 1\}$ represents the direct connectivity between sensors $s_i, s_j \in \mathbf{S}$, then:

$$\mathcal{P}(s_i, s_j) = \Big(dist(s_i, s_j) \leq r^c_{i,j} \Big) \tag{4}$$

where $r^c_{i,j} = \min\{r^c_i, r^c_j\}$ and r^c_i is the communication radius of sensor s_i. In this, we make two simplifying assumptions: that line-of-sight is not required for RF communication, and that wireless propagation follows a path-loss model with r^c_i as the maximum distance with acceptable radio signal strength. We have left to future work to relax these assumptions, because they can significantly impact the scaling of network synthesis.

The predicate $\mathcal{P}(s_i, s_j)$ alone is not sufficient to establish *network connectivity*. To do this, let $\hat{\mathbf{A}}_{N-1}$ be a matrix in which the (i, j)-th element represents the number of (N-1) hop paths without loops between sensors i and j. Then, network connectivity holds if there is at least one path between each pair of nodes:

$$(\hat{\mathbf{A}}_{N-1})_{ij} > 0 \quad \forall i, j \in \{1, \cdots, N\} \;\; \text{with} \;\; i \neq j. \tag{5}$$

Obstacles and sensor placement (C4). To prevent the synthesizer from placing sensor s_i on obstacles, let $\mathcal{V}^o(s_i, o)$ be a predicate that evaluates to true if s_i is *not* placed on obstacle o. Then, if $\mathcal{V}(s_i)$ is a predicate that checks whether sensor $s_i \in \mathbf{S}$ is not placed on any obstacle:

$$\mathcal{V}(s_i) = \bigwedge_{o \in \mathbf{O}} \mathcal{V}^o(s_i, o). \tag{6}$$

The overall formulation. Given this formulation, our network synthesis formulation reduces to finding the smallest number of sensors N that satisfy *all* four of the constraints listed in Table I.

Performance Goals. Table I lists constraints on the *correctness* of network synthesis. In addition, motivated by the problem of quick, ad-hoc deployments of IoT networks (§I) in large deployment areas, we impose two performance objectives: synthesizing a network, within *tens of minutes*, to cover a large urban campus of a *1-2 sq. kms.* As we show in the rest of the paper, these *scaling goals* stress the capabilities of existing synthesis methods.

III. A Brief Overview of SMC

Satisfiability Modulo Convex (SMC) Theory [11] extends Boolean Satisfiability (SAT). A Boolean Satisfiability (SAT) problem finds feasible assignments to Boolean variables consistent with a set of constraints, typically represented as the conjunction of a set of Boolean clauses, for example:

$$a_1 \wedge (a_2 \vee a_3) \wedge (a_1 \vee a_3) \tag{7}$$

Here, a_1, a_2, a_3 are Boolean variables. A SAT solver attempts to find an assignment for the Boolean variables, such that the entire clause evaluates to TRUE given the formula is satisfiable. For example, Equation 7 can be satisfied the following assignment: $a_1 =$ TRUE, $a_2 =$ FALSE, $a_3 =$ TRUE, and the SAT solver returns this assignment. On the other hand, if no such assignment exists, the function expressed by the formula is FALSE for all possible variable assignments and the formula is unsatisfiable and the SAT solver identifies that no satisfying assignment exists. For example, $a_1 \wedge \neg a_1$ is unsatisfiable. SAT problems are typically hard to solve, but recent SAT solvers ([9], [13]) scale well to problems of practical interest [14], [15].

SMC [11], designed to address the feasibility of mixed-integer convex problems, uses a SAT solver to suggest admissible assignments for a problem's Boolean variables[2] and a convex solver (*e.g.,* [16]) to suggest admissible values of the problem's real variables. To bridge the two tools, it represents convex constraints using *pseudo-Boolean* variables. Consider the following example:

$$a_1 \wedge (a_2 \vee a_3) \wedge (a_1 \rightarrow x_1 + x_2 = 4) \tag{8}$$

Here, a_1, a_2, a_3 are Boolean variables and x_1, x_2 are real variables. The third clause implies that if a_1 is TRUE, we should be able to find a valid value of x_1 and x_2 such that $x_1 + x_2 = 4$ (a convex constraint).

SMC replaces the convex constraint with a pseudo-Boolean variable, say a_4, then runs the SAT solver. If the solver produces a TRUE assignment for both a_4 and a_1, then it attempts to use the convex solver to find a satisfying assignment for the convex constraint. If none are found (in our example, there exists a satisfying assignment), the output of the convex solver is used to produce a *counter-example* to constrain the search space for the SAT solver. This leads to a search over a more constrained SAT solution space by excluding conflicting combinations of pseudo-boolean variables corresponding to the counter-example. This process repeats until the suggested combination of convex constraints is satisfiable.

When used for network synthesis, SMC produces a *feasible* solution, given N, the number of sensors, as input (or indicates infeasibility). To synthesize a network with the fewest sensors, we perform a binary search on N.

IV. SMC Formulation of Network Synthesis

In this section, we describe how we cast network synthesis in the SMC framework [11].

Area Coverage. The coverage constraint (C1, Equation 1) specifies *point* coverage, where the locations l_j represent discrete locations in the deployment region $\mathbf{L}$. In practice, IoT deployments desire *area* coverage, where at least k sensors cover every point within $\mathbf{L}$ that is not covered by an obstacle.

[2]Note that one can almost trivially encompass integer variables via a larger number of Boolean variables, following the appropriate transformations.

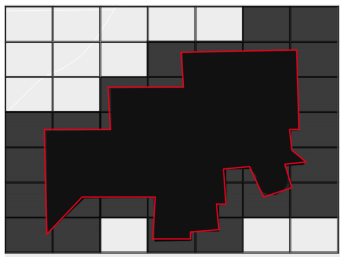


Fig. 1: A complex obstacle can be modeled as a polygon, but this can be computationally difficult. We discretize space into grids and model each grid as occupied (if even a part of an obstacle overlaps with the grid) or unoccupied. Occupied grids are shaded in a dark color.

The area coverage problem in presence of obstacles is not convex, so it is hard to represent it using SMC.

Discretizing the deployment region. To address this challenge, we discretize space into a uniform grid which induces a new discretized space $\mathbf{D}$. Each grid element $d \in \mathbf{D}$ can be represented in 2-D space as the *intersection of four half planes*:

$$\mathcal{I}^d(\mathbf{x}) = \mathcal{I}^d(x,y) = (x > x^d_{min}) \wedge (x < x^d_{max}) \\ \wedge (y > y^d_{min}) \wedge (y < y^d_{max}) \tag{9}$$

The respective four physical corners are $\mathbf{L}^d = \{(x^d_{min}, y^d_{min}), (x^d_{max}, y^d_{min}), (x^d_{max}, y^d_{max}), (x^d_{min}, y^d_{max})\}$.

Representing obstacles. In general, we can represent an obstacle as a fine-grain polygon (Figure 1). However, the finer the representation of the obstacles, the less scalable the network synthesis process becomes. A single obstacle with P vertices creates $\mathcal{O}(M \cdot N \cdot P)$ constraints, where M represents the set of points covered by sensors. So, we leverage the space discretization to improve synthesis scaling: we say that each grid element $d \in \mathbf{D}$ is *occupied* if an obstacle covers even a part of the element, else the element is *unoccupied* (Figure 1). Furthermore, since every $d \in \mathbf{D}$ is either occupied by an obstacle or free, $\mathbf{D}$ is the union of two disjoint sets $\mathbf{O}$ and $\mathbf{U}$, corresponding to occupied (obstacles) and unoccupied spaces, respectively.

Defining area coverage. If a sensor s_i covers the four corners of a grid element $u_g \in \mathbf{U}$, then it covers all points within the element. Thus, we can represent the problem of covering the entire area $\mathbf{L}$ by the problem of covering the corners of unoccupied grids. Formally, let $\mathbf{L} = \bigcup_{u_i \in \mathbf{U}} \mathbf{L}^u_i$ where $\mathbf{L}^u_i$ denotes the four corners of the grid u_i. Then, our formulation uses the pseudo-Boolean variables in Table II to directly represent the convex and complex constraints (§II).

We can encode the k-coverage problem using SMC as follows.

Coverage Constraints (C1). We ensure that at least k sensors cover each grid region u_g as follows:

$$\left(\sum_{i=1}^{N} b^u_{ig} \geq k \right) \tag{10}$$

where covering a grid implies that the same sensor covers all four corners of the grid $(b^u_{ig} \rightarrow \bigwedge_{l_j \in \mathbf{L}^u_g} b^s_{ij})$ which

TABLE II: Pseudo Boolean for SMC Formulation

Description		All Possible Combinations	Pseudo Boolean Variable		Respective Predicate or Constraints
Location Coverage	$\psi_1 =$	$\bigwedge_{i=1}^{N} \bigwedge_{l_j \in \mathbf{L}}$	$\left(b^s_{ij}\right.$	$\rightarrow$	$\left.\mathcal{C}(s_i, l_j)\right)$
Grid Coverage	$\psi_2 =$	$\bigwedge_{u_g \in \mathbf{U}}$	$\left(b^u_{ig}\right.$	$\rightarrow$	$\left.\bigwedge_{l_j \in \mathbf{L}^u_g} \mathcal{C}(s_i, l_j)\right)$
Visibility	$\psi_3 =$	$\bigwedge_{i=1}^{N} \bigwedge_{l_j \in \mathbf{L}} \bigwedge_{o_g \in \mathbf{O}}$	$\left(b^o_{ijg}\right.$	$\rightarrow$	$\left.\mathcal{B}^o(s_i, l_j, o_g)\right)$
Link Connectivity	$\psi_4 =$	$\bigwedge_{i=1}^{N} \bigwedge_{j=1, j \neq i}^{N}$	$\left(b^c_{ij}\right.$	$\rightarrow$	$\left.\mathcal{P}(s_i, s_j)\right)$
Placement	$\psi_5 =$	$\bigwedge_{i=1}^{N} \bigwedge_{o_g \in \mathbf{O}}$	$\left(b^v_{ig}\right.$	$\rightarrow$	$\left.\mathcal{V}^o(s_i, o_g)\right)$

expresses the constraint that if sensor s_i covers a grid u_g ($b^u_{ig} = 1$ *or* TRUE), then it covers all four corners of the grid $l_j \in \mathbf{L}^u_g$.

Visibility Constraints (C2). The four half-planes (Equation 9) of each occupied grid element can also model visibility. Let $\mathbf{L}^o_g$ represent the respective four corners of occupied grid element $o_g \in \mathbf{O}$. Then, line of sight depends upon two conditions:

(v1) This condition applies when all four vertices of an obstacle are on the same half plane created by the line joining a sensor location s_i and a sensed location l_j (Figure 2a). If the line equation joining s_i, l_j is $f_{ij}(\mathbf{x} = (x,y)) = 0$ then all four vertices should satisfy either $f_{ij}(\mathbf{x}) > 0$ or $f_{ij}(\mathbf{x}) < 0$, expressed as follows:

$$\mathcal{B}_1(s_i, l_j, o_g) = \Big(\bigwedge_{\mathbf{x} \in \mathbf{L}^o_g} (f_{ij}(\mathbf{x}) > 0) \Big) \bigvee \Big(\bigwedge_{\mathbf{x} \in \mathbf{L}^o_g} (f_{ij}(\mathbf{x}) < 0) \Big) \tag{11}$$

(v2) This condition applies when both the sensor location and the sensed location are on the outer half-plane of at least one obstacle face (Figure 2b). Mathematically, both points should satisfy one of the following four conditions: $x < x^o_{min}$, $x > x^o_{max}$, $y < y^o_{min}$, $y > y^o_{max}$, expressed as follows:

$$\mathcal{B}_2\Big(s_i = (x_1, y_1), l_j = (x_2, y_2), o_g\Big) = (x_1, x_2 < x^o_{min}) \\ \vee (x_1, x_2 > x^o_{max}) \vee (y_1, y_2 < y^o_{min}) \vee (y_1, y_2 > y^o_{max}) \tag{12}$$

If $\mathcal{B}^o(s_i, l_j, o_g)$ is a predicate that checks whether an obstacle grid element o_g does not block the line of sight between s_i and l_j, then we can write:

$$\mathcal{B}^o(s_i, l_j, o_g) = \mathcal{B}_1(s_i, l_j, o_g) \vee \mathcal{B}_2(s_i, l_j, o_g) \tag{13}$$

Using this, we can compute the visibility constraint (Equation 3).

Connectivity Constraints (C3). In a connected network, there should exist a communication path (single-hop or multihop) between every pair of nodes. Consider the adjacency matrix $\mathbf{A}$: $\mathbf{A}_{ij} = b^c_{ij}$ where $b^c_{ij} \in \{0, 1\}$ with True as 1 and False as 0. Now, let $\hat{A}_h$ represent the h-hop adjacency matrix (whose i,j-th entry is the number of h-hop paths between s_i and s_j), then we can write the connectivity constraint as:

$$\bigwedge_{j=2}^{N} \Big((\hat{\mathbf{A}}_{N-1})_{1j} > 0 \Big). \tag{14}$$

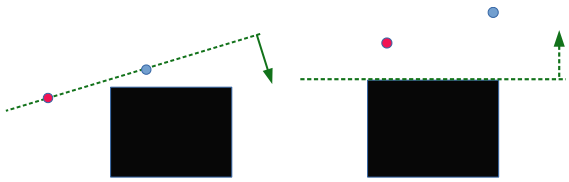


Fig. 2: Two visibility conditions. Red dot signifies a sensor and Blue dot signifies a sensed location.

TABLE III: Summary of SMC Encoding of k-Coverage

Coverage	(C1)	$\psi_6 = \bigwedge_{u_g \in \mathbf{U}} \left(\sum_{i=1}^{N} b_{ig}^{u} \geq k \right)$
Visibility	(C2)	$\bigwedge_{o_g \in \mathbf{O}} b_{ijg}^{o}$
Connectivity	(C3)	$\psi_7 = \bigwedge_{j=2}^{N} \left((\hat{\mathbf{A}}_{N-1})_{1j} > 0 \right)$
Placement	(C4)	$\psi_8 = \bigwedge_{i=1}^{N} \bigwedge_{o_g \in \mathbf{O}} b_{ig}^{v}$

This is a simplified version of (Equation 5): we only check connectivity between node 1 and every other node because, if there exists a path from node 1 to every other node, they can always connect via node 1.

Obstacles and sensor placement (C4). To prevent the synthesizer from placing sensors in grid elements occupied by obstacles, we can define constraints that ensure that each sensor s_i's position is outside the four half planes (Equation 9) defined by each grid element, as follows:

$$\mathcal{V}^{o}\big(s_i = (x, y), o_g\big) = (x < x^{o}_{min}) \vee (x > x^{o}_{max}) \vee (y < y^{o}_{min}) \vee (y > y^{o}_{max}) \tag{15}$$

where $x^{o}_{min}, x^{o}_{max}, y^{o}_{min}, y^{o}_{max}$ define the four halfplanes of the obstacle grid. Finally, to make sure that the SMC solver does not put the sensor inside the obstacles, we add the following sets of clauses, the equivalent of (Equation 6):

$$\bigwedge_{i=1}^{N} \bigwedge_{o_g \in \mathbf{O}} b_{ig}^{v} \tag{16}$$

This forces the solver to only select locations from the unoccupied grid elements.

The overall formulation. Putting all these together, the four key constraints can be summarized in Table III. From this, we arrive at the overall SMC formulation: $\psi = \bigwedge_{i=1}^{8} \psi_i$. With this formulation, we perform a binary search on N to find the solution with the fewest number of sensors.

Constraint pre-computation. Our SMC encoding discretizes the deployment region. If two such locations l_i and l_j are more than twice the sensing radius apart, then no single sensor can cover them. While SMC can eventually determine this through counter-examples with our encoding, we have found that it significantly improves the speed of synthesis to provide these as pre-computed constraints. We do this by (a) finding all pairs l_i and l_j that are greater than twice the sensing radius apart and (b) adding a constraint that prevents a single sensor from covering them. (We add similar constraints for connectivity as well).

V. Hierarchical Synthesis

SMC, by itself, fails to achieve our scaling goals (§III). We apply hierarchical synthesis to achieve these.

In hierarchical synthesis, we subdivide the deployment area $\mathbf{L}$ into smaller sub-areas (Figure 3a) and solve for coverage (and not for connectivity) in each of the sub-areas separately. Then, once we have individual solutions, we need to connect them so that we have a connected network. To connect the individual sub-problems in SMC, we first combine the solutions to check for connected pairs of nodes, then collapse each of the connected sub-networks into a single node (Figure 3b). Then, we use constraints ψ_4 and ψ_7 in the SMC formulation to ensure connectivity between the sub-areas.

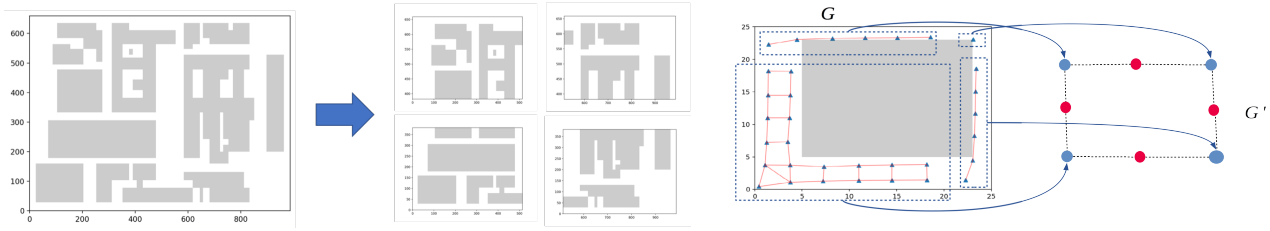
We could have solved for both coverage and connectivity in each of the sub-areas. In a complex deployment region, solving for both connectivity and coverage in each subproblem often over-provisions the network. Also, a k-coverage solution often results in connected sub-networks, so our approach enables faster synthesis by reducing computation (§VI).

Incremental Coverage Repair. Hierarchical synthesis can help scaling but may result in redundant coverage, because it solves each sub-region independently, so more than k sensors may cover parts of the sub-region near the boundaries (*e.g.,* by sensors from neighboring sub-regions). To circumvent this, we first solve each sub-area for $k = 1$ coverage. Then, for each grid element u, we measure the obtained coverage k_u from this solution. The residual coverage requirement for u, then, is $k - k_u$; we now run the coverage problem again for each sub-area with these residual coverage requirements as constraints, which effectively "fills" up the coverage on grid points interior to the sub-area. After this, we apply the connectivity repair described above.

VI. Evaluation

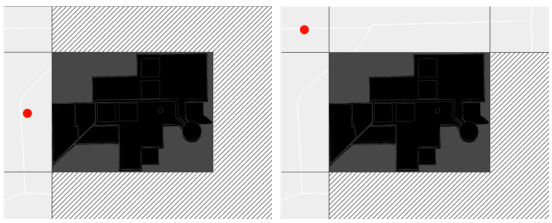
A. Methodology

Comparison. To understand the performance of SMC in the context of network synthesis, we developed a baseline network synthesis formulation using the well-known Mixed Integer Linear Programming (MILP) framework. To model visibility in MILP formulation, we assume that a sensor, located in a space directly adjacent to an obstacle, cannot sense any discrete space at or past the obstacle's position. For example, if a sensor is in a space diagonal to an obstacle, it cannot sense any space on the vertically or horizontally opposite side of the object (Figure 4a and Figure 4b). The MILP formulation transforms the visibility and coverage constraints into a graph vertex cover problem. The solution to this vertex cover problem is the placement of sensors that satisfies the visibility and coverage constraints, but the resulting network may not be connected. We restore connectivity via a graph based Steiner tree approach similar to the SMC hierarchical approach (§V). We do not present the MILP formulation details due to page limitations.



(a) (b)

Fig. 3: (a) Hierarchical synthesis solves sub-regions, then combines these solutions to satisfy coverage and connectivity over the entire deployment region using a (b) Connectivity repair method between sub-problems



(a) (b)

Fig. 4: Visibility restrictions placed on the sensors in discrete space. We under-approximate visibility so the resulting assignment of discrete locations is valid for any real-valued location within the discrete region.

Implementation. Our SMC implementation uses Z3 [9] for SAT solving, and CPLEX [16] for convex constraints. We use CPLEX also for the MILP formulation.

Inputs. There are six inputs in our evaluation. The *deployment region* $\mathbf{L}$ denotes the discretized grid for which we synthesize the network. For most of our experiments, we use two different sizes of scenarios: (1) 20×20 (*Small*) and (2) 50×50 (*Large*) grid. In the *Small*, we use hierarchy with SMC but not with MILP since the latter scales to this problem size. For the *Large*, we use hierarchy for both since neither is able to scale to this problem size. The performance of synthesis algorithms depends heavily on the fraction of the deployment region occupied by obstacles (*obstacle extent*), and on the spread of these obstacles across the deployment region (*obstacle dispersion*). The *coverage radius* r_c and the *communication radius* r_s are also inputs; our results are sensitive to β, the ratio of the communication radius to the coverage radius. The *grid granularity* controls how finely we can represent the deployment region; a finer grid implies lower coverage redundancy at the expense of scalability. Finally, we fix the coverage goal at $k = 3$ (at least three sensors must cover each grid).

Scalability and Performance. Since we are interested in synthesis for ad-hoc rapid deployments, we use the time taken to arrive at a good solution for a given problem size as a measure of scalability. Since we compare iterative synthesis methods, we specify this time as a constraint: we allocate a fixed amount of execution time T_e on a fixed computing configuration for each of our methods and use the solution obtained by the end of that duration (or earlier if the synthesis has converged). A problem setting that does not scale will have arrived at a sub-optimal solution after T_e. In our experiments, we fix T_e to 1 hour.

Metric. We then measure the performance of network synthesis by its *coverage redundancy*. If the average number of sensors covering each location in the solution is k_{avg}, *coverage redundancy* is $\frac{k_{avg}}{k}$ where k is the desired coverage. For example, if, on average, 6 sensors cover every location, but we desire 3-coverage, then coverage redundancy is 2.

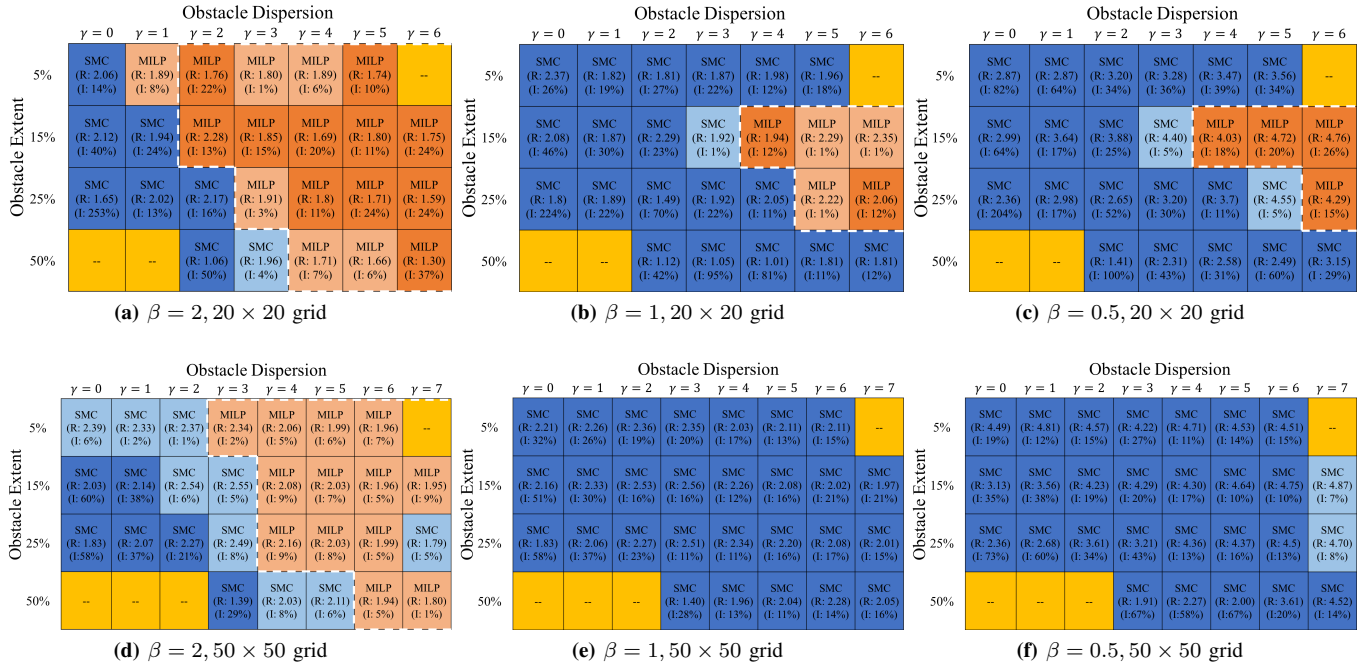
B. The Role of Obstacles

Obstacles determine the visibility and placement constraints (§II) and significantly impact the performance of network synthesis. To evaluate how obstacle extent and obstacle dispersion affect the performance of SMC, we perform a set of experiments on synthetically generated obstacle distributions.

To quantify the obstacle dispersion, we define γ as the average number of adjacent grids with obstacles for each obstacle grid. γ can theoretically vary from 0 to 8 with $\gamma = 7, 8$ being highly rare. For a fixed obstacle extent, *lower values of* γ *occur for small, widely dispersed obstacles*. We evaluate and compare the performance of MILP and SMC for all feasible combinations of four different obstacle extents 5%, 15% 25%, 50%, and eight different values of $\gamma = \{0, 1, 2, 3, 4, 5, 6, 7\}$ (we do not use $\gamma = 7$ for *Small* as it is very rare). In each case, we place obstacles on a 20×20 grid and a 50×50 grid and generate five obstacle placements (coverage redundancy varies little across placements, so we use 5 placements to minimize the total time to run experiments). We set the coverage radius to 6 units and explore the sensitivity of results to three different choices of $\beta = \{2, 1, 0.5\}$.

Figure 5 presents the results of these experiments, both for *Large* and *Small*, where two methods are comparable if either their coverage redundancy 95% confidence intervals overlap or the means are within 10% of each other. These results indicate four distinct regimes of operation, discussed below.

Small Obstacle Extent ($< 15\%$). Figure 5 illustrates that for both *Large* and *Small*, with few obstacles (roughly $< 15\%$) SMC outperforms MILP for $\beta = 1, 0.5$. On the other hand, for $\beta = 2$ the performance largely depends on the obstacle dispersion (Figure 5a and 5d) with slightly better performance for MILP with lower obstacle dispersion ($\gamma > 3$). For $\beta = 2$,



(a) $\beta = 2, 20 \times 20$ grid **(b)** $\beta = 1, 20 \times 20$ grid **(c)** $\beta = 0.5, 20 \times 20$ grid

(d) $\beta = 2, 50 \times 50$ grid **(e)** $\beta = 1, 50 \times 50$ grid **(f)** $\beta = 0.5, 50 \times 50$ grid

Fig. 5: Performance comparison for different combinations of obstacles extent, β and γ. **Blue** $\implies$ SMC performs significantly better, **Light Blue** $\implies$ SMC performs slightly better ($< 10\%$), **Light Orange** $\implies$ MILP performances slightly better (within 10% of each other), **Orange** $\implies$ MILP performs significantly better, and **Yellow** $\implies$ infeasible test scenario.

the coverage solution is also connected [17]; MILP converges faster to solutions with slightly lower coverage redundancy in these settings both because its visibility and connectivity approximation is less in obstructed environments and because the visibility graphs are less dense. SMC still performs comparable to MILP for most settings. MILP performs less well for $\beta = 1, 0.5$ because of its graph based connectivity repair technique. For $\beta = 1$ (when the connectivity same as the sensing radius) and $\beta = 0.5$ (when the connectivity radius is half the sensing radius), MILP has to deploy a large number of relay nodes to repair connectivity (made larger by the under-approximation of connectivity), which increases coverage redundancy significantly compared to SMC.

High Obstacle Dispersion ($\gamma \leq 3$). Figure 5 also illustrates that, regardless of the obstacle extent and value of β, with dispersed obstacles (i.e., $\gamma \leq 3$) SMC outperforms or is comparable to MILP. With many small sized obstacles, MILP under-approximates visibility significantly and ends up placing more sensors. Thus, even with a large obstacle extent, if obstacles are small and widely dispersed, MILP's performance degrades.

Medium Obstacle Extent ($15\% - 25\%$) and Low Obstacle Dispersion ($\gamma > 3$). In this regime, MILP often outperforms SMC regardless of the values of β for *Small* when MILP does not require any hierarchy, but SMC does (Figure 5a, 5b, 5c). In more obstructed environments, SMC requires a deeper hierarchy to scale well. However, as we introduce hierarchy to MILP (*Large*), MILP's performance deteriorates due to over-provisioning as a result of hierarchy and SMC becomes the better alternative (Figure 5d, 5e, 5f).

Large Obstacle Extent ($> 25\%$) and Low Obstacle Dispersion ($\gamma > 3$). In this regime, the relative performance of these schemes again depends on the value of β (similar to Small Obstacle Extent ($< 15\%$)). For $\beta = 2$, MILP performance is slightly better (*i.e.,* comparable) than SMC (Figure 5a and 5d) and for $\beta = 1, 0.5$, SMC outperforms MILP (Figure 5b, 5c, 5e, 5f). This results from the connectivity repair technique used in MILP. For $\beta = 2$, the coverage solution is guaranteed to be connected [17].

Effect of β. When $\beta = 2$, a solution for coverage also results in a connected network. In contrast, when $\beta = 0.5$, ensuring connectivity requires additional sensors. This is evident from the coverage redundancy values in Figure 5. For $\beta = 1, 2$, the best coverage redundancy is mostly within a factor of 2 of the optimal across the entire range of obstacle distributions while it is within a factor of 4 of the optimal for $\beta = 0.5$. In Figure 5, we can see a nice separation between two regimes where SMC and MILP perform well for $\beta = 2$ (Figure 5a and 5d). As β goes from 2 to 1 to 0.5, the regime where MILP outperforms SMC vanishes. Since the primary difference between these regimes is the importance of connectivity, this shows that SMC handles connectivity constraints better than MILP.

Number of grid elements. As the number of grid elements increases (we fix the grid dimensions), MILP and SMC need hierarchy to scale. Figure 5 shows that for *Small* there exists a difference between MILP and SMC (*e.g.,* Figure 5b); MILP does not need hierarchy but SMC does. This difference disappears in the *Large* (Figure 5e). For MILP, the problem

TABLE IV: Summary of the Experiments

Obstacle Extent	Obstacle Distribution γ	β	Best Synthesis Method	
			$\lvert\mathbf{O}\rvert \leq \chi$	$\lvert\mathbf{O}\rvert > \chi$
Any	> 3	> 1	MILP	MILP/SMC
Any	≤ 3	> 1	SMC	SMC
$< 15\%$	Any	≤ 1	SMC	SMC
$\geq 15\%, \leq 25\%$	> 3	≤ 1	MILP/SMC	SMC
$\geq 15\%, \leq 25\%$	≤ 3	≤ 1	SMC	SMC
$> 25\%$	Any	≤ 1	SMC	SMC

size beyond which it does not scale can be characterized by the number of unoccupied grids, $|\mathbf{O}|$. If this number is greater than a threshold χ, then MILP does not scale without hierarchy because the number of constraints and solution search space increases dramatically (the number of constraints in MILP is a function of the number of open grids, since these determine its visibility and communication graphs). χ is a function of the capacity of the solver [9] and the computing configuration used for the synthesis. We can experimentally profile this quantity; in our experiments, χ is approximately 1200.

SMC requires hierarchy in both scenarios. Its performance improves relative to MILP, because MILP performance degrades because of the coverage and connectivity approximations. These arise from limitations in MILP's expressivity with respect to our network synthesis problem. This is evident from Figure 5e and 5f: for the *Large* with $\beta = 1, 0.5$, SMC always outperform MILP significantly. For $\beta = 2$, we can see that the change is less severe as for $\beta = 2$ the coverage solution is the connectivity solutions and thus adding hierarchy deteriorates the performance of MILP slightly and makes it comparable to SMC.

To check whether the boundary between the two changes at larger settings, we performed a set of experiments with a 75×75 grid scenario and generated the respective heatmap (also omitted for brevity). This heatmap is comparable to the *Large* results in Figure 5. With increasing problem size, both methods appear to be equally affected by the addition of deeper levels of hierarchy needed to scale the synthesis.

Synthesis method selection. Taken together, these results suggest that MILP and SMC perform well in different regimes (indicated by the dashed line boundary in Figure 5), so network synthesis should select which method to use depending on (a) the obstacle extent, (2) obstacle dispersion (γ), (3) the coverage to communication radius ratio (β), and (4) the number of open grids ($|\mathbf{O}|$). Table IV summarizes these choices. Table IV illustrate that SMC is a better choice for large scale network synthesis and one can pick SMC for network synthesis regardless of the values of the four factors (discussed above) to produce a network with acceptable performance.

VII. Related Work

Table V shows how our work relates to the existing literature.

Wireless Sensor Networks. Over the last two decades, researchers have studied sensor network placement problems [7], [8]. Common approaches rely on random dense deployment followed by a careful selection among the deployed sensors to fulfill the sensing goal [18]: often, this work has not considered obstacles. We highlight the most relevant prior work in this area: [19], [20] present a more complete treatment.

Discrete Set of Sensors and Locations. With a set of randomly pre-deployed sensors, the network topology design reduces to controlling the sleep pattern of the sensors [21], [17], [22], [23]. This can be easily encoded using SAT [24] or framed using computational geometry [25]. None of these approaches deals with obstacles and visibility constraints for sensing. In contrast, we focus on a methodical top-down synthesis and support richer sensing goals such as k-coverage. Moreover, unlike these approaches, we focus on scaling to large deployments in highly occluded settings.

Submodular Optimization. Prior work has explored a greedy algorithm for submodular optimization of sensor placement and scheduling [26]. The proposed method does not consider connectivity constraints. Similarly, [27] uses greedy heuristics to solve several problems in selecting an optimal placement of sensors in a water network with submodular objectives. While sub-modularity assumptions enable rapid synthesis, they lack the expressivity of SMC which is necessary to jointly address coverage and connectivity constraints.

TABLE V: Summary of Related Work

Related Work	Coverage		Modelling				Connectivity	Scalability	Topdown Synthesis
	k-Cover	Area Cover	Sensing Range	Comm Range	Obstacle model	Visibility model			
[17]	✓	✓	✓	✓			✓		
[18]		✓	✓	✓			✓		
[22]			✓	✓			✓		
[23]	✓	✓							
[26]		✓						✓	
[27]								✓	
[28]	✓		✓	✓			✓		✓
[29]			✓	✓	✓		✓		
[30]							✓		✓
[31]		✓	✓			✓			
[32]		✓		✓		✓	✓		✓
[33], [34]		✓	✓		✓	✓			✓
[35], [36]		✓	✓		✓	✓			✓
Our Work	✓	✓	✓	✓	✓	✓	✓	✓	✓

Relay Placement. Prior work has also explored connectivity repair using relays [28], [29]. In [30], the authors address the problem of placing relay nodes to create $k = 1, 2$, and higher connectivity in both one and two-way communication scenarios. The problem is NP-hard, and a series of polynomial approximation algorithms are presented to solve the problem for $k = 1, 2$ and a generalization is given for $k > 2$. In contrast, we do not focus on just relay placement, but on the synthesis of a connected network satisfying coverage objectives, a much harder problem. In [29], a set of relays is placed to establish a communication backbone. They consider obstacles and compute an Euclidean Obstacle-Avoidance Steiner tree to place the relays but this solution does not scale.

Camera Placement. There exists a body of work on camera placement related to our work. Solutions to the well-known art-gallery problem for placing visual sensor [31] generally assume infinite sensing range, and do not consider connectivity. One work [32] explores an indoor camera placement problem with connectivity constraints, but in an uncluttered environment and assumes unlimited sensing range. Another work [33] considers limited camera ranges and polygonal obstacles, but does not consider connectivity. Other prior work has used quadratic convex programming [34], integer linear

programming [35], and linear programming [36] for camera placements, but again do not consider connectivity.

Wired Network Synthesis. Finally, prior work has explored synthesis of network topologies and routing configurations [4], [6]. Our problem is qualitatively different in that it includes coverage requirements and visibility constraints.

VIII. Conclusion

Motivated by advances in solver technology, this paper considers the top-down synthesis of large-scale ad-hoc IoT networks. We explore a formulation based on the SMC framework that permits convex constraints. We also develop a hierarchical synthesis technique to scale to large problem sizes. Our results show that, SMC's solution quality is better than a baseline MILP formulation at larger problem sizes. This is likely due to the fact that MILP can only approximately capture coverage and connectivity constraints, and these approximations result in higher coverage redundancy. Several directions of future work remain including: extending the synthesis to accommodate heterogeneous and directional sensors, and determining scaling techniques for these; incorporating computational elements into the network synthesis to place fusion nodes that can optimally process sensor data; and large scale real world experimentation to validate the synthesized networks.

References

[1] P. Kazemian, G. Varghese, and N. McKeown, "Header space analysis: Static checking for networks," in *Proc. Usenix NSDI*, 2012.

[2] A. Fogel, S. Fung, L. Pedrosa, M. Walraed-Sullivan, R. Govindan, R. Mahajan, and T. Millstein, "A general approach to network configuration analysis," in *Proc. Usenix NSDI*. Oakland, CA: USENIX Association, May 2015, pp. 469–483. [Online]. Available: https://www.usenix.org/conference/nsdi15/technical-sessions/presentation/fogel

[3] H. Zeng, S. Zhang, F. Ye, V. Jeyakumar, M. Ju, J. Liu, N. McKeown, and A. Vahdat, "Libra: Divide and conquer to verify forwarding tables in huge networks," in *Proc. Usenix NSDI*, 2014.

[4] R. Beckett, R. Mahajan, T. Millstein, J. Padhye, and D. Walker, "Network configuration synthesis with abstract topologies," in *ACM SIGPLAN Notices*, vol. 52, no. 6. ACM, 2017, pp. 437–451.

[5] M. Zhang, R. N. Mysore, S. Supittayapornpong, and R. Govindan, "Understanding lifecycle management complexity of datacenter topologies," in *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2019.

[6] B. Schlinker, R. N. Mysore, S. Smith, J. C. Mogul, A. Vahdat, M. Yu, E. Katz-Bassett, and M. Rubin, "Condor: Better topologies through declarative design," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 449–463, 2015.

[7] M. Younis and K. Akkaya, "Strategies and techniques for node placement in wireless sensor networks: A survey," *Ad Hoc Networks*, vol. 6, no. 4, pp. 621–655, 2008.

[8] Y.-C. Wang, C.-C. Hu, and Y.-C. Tseng, "Efficient placement and dispatch of sensors in a wireless sensor network," *IEEE Transactions on Mobile Computing*, vol. 7, no. 2, pp. 262–274, 2007.

[9] L. De Moura and N. Bjørner, "Z3: An efficient smt solver," in *Proceedings of the Theory and Practice of Software*, ser. TACAS'08/ETAPS'08, 2008, pp. 337–340.

[10] *Gurobi Optimizer Reference Manual*, 2019. [Online]. Available: http://www.gurobi.com

[11] Y. Shoukry, P. Nuzzo, A. L. Sangiovanni-Vincentelli, S. A. Seshia, G. J. Pappas, and P. Tabuada, "Smc: Satisfiability modulo convex programming," *Proceedings of the IEEE*, vol. 106, no. 9, pp. 1655–1679, 2018.

[12] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli *et al.*, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[13] Minisat, "http://minisat.se/," 2019.

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

[15] I. Lynce and J. Ouaknine, "Sudoku as a sat problem." in *ISAIM*, 2006.

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

[17] G. Xing, X. Wang, Y. Zhang, C. Lu, R. Pless, and C. Gill, "Integrated coverage and connectivity configuration for energy conservation in sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 1, no. 1, pp. 36–72, 2005.

[18] X. Bai, S. Kumar, D. Xuan, Z. Yun, and T. H. Lai, "Deploying wireless sensors to achieve both coverage and connectivity," in *Proceedings of Mobihoc*, 2006, pp. 131–142.

[19] B. Wang, "Coverage problems in sensor networks: A survey," *ACM Computing Surveys*, vol. 43, no. 4, p. 32, 2011.

[20] C. Zhu, C. Zheng, L. Shu, and G. Han, "A survey on coverage and connectivity issues in wireless sensor networks," *Journal of Network and Computer Applications*, vol. 35, no. 2, pp. 619–632, 2012.

[21] P. Santi, "Topology control in wireless ad hoc and sensor networks," *ACM Computing Surveys (CSUR)*, vol. 37, no. 2, pp. 164–194, 2005.

[22] C.-P. Chen, S. C. Mukhopadhyay, C.-L. Chuang, M.-Y. Liu, and J.-A. Jiang, "Efficient coverage and connectivity preservation with load balance for wireless sensor networks," *IEEE Sensors Journal*, vol. 15, no. 1, pp. 48–62, Jan 2015.

[23] Z. Abrams, A. Goel, and S. Plotkin, "Set k-cover algorithms for energy efficient monitoring in wireless sensor networks," in *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*, ser. IPSN '04. ACM, 2004, pp. 424–432.

[24] M. Ashouri, Z. Zali, S. R. Mousavi, and M. R. Hashemi, "New optimal solution to disjoint set k-coverage for lifetime extension in wireless sensor networks," *IET Wireless Sensor Systems*, vol. 2, no. 1, pp. 31–39, March 2012.

[25] X. Bai, S. Kumar, D. Xuan, Z. Yun, and T. H. Lai, "Deploying wireless sensors to achieve both coverage and connectivity," in *Proceedings of the 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, ser. MobiHoc '06, 2006, pp. 131–142.

[26] A. Krause, R. Rajagopal, A. Gupta, and C. Guestrin, "Simultaneous optimization of sensor placements and balanced schedules," *IEEE Transactions on Automatic Control*, vol. 56, no. 10, pp. 2390–2405, Oct 2011.

[27] A. Krause, J. Leskovec, C. Guestrin, J. VanBriesen, and C. Faloutsos, "Efficient sensor placement optimization for securing large water distribution networks," *Journal of Water Resources Planning and Management*, vol. 134, no. 6, pp. 516–526, 2008.

[28] D. Yang, S. Misra, X. Fang, G. Xue, and J. Zhang, "Two-tiered constrained relay node placement in wireless sensor networks: Efficient approximations," in *SECON 2010*, June 2010, pp. 1–9.

[29] E. R. S. Santos and M. A. M. Vieira, "Autonomous wireless backbone deployment with bounded number of networked robots," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2014, pp. 3740–3746.

[30] X. Han, X. Cao, E. L. Lloyd, and C.-C. Shen, "Fault-tolerant relay node placement in heterogeneous wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 9, no. 5, pp. 643–656, May 2010.

[31] H. González-Banos, "A randomized art-gallery algorithm for sensor placement," in *Proceedings of the seventeenth annual symposium on Computational geometry*. ACM, 2001, pp. 232–240.

[32] H. Huang, C.-C. Ni, X. Ban, J. Gao, A. T. Schneider, and S. Lin, "Connected wireless camera network deployment with visibility coverage," in *IEEE INFOCOM 2014*, pp. 1204–1212.

[33] U. M. Erdem and S. Sclaroff, "Automated camera layout to satisfy task-specific and floor plan-specific coverage requirements," *Computer Vision and Image Understanding*, vol. 103, no. 3, pp. 156–169, 2006.

[34] B. Ghanem, Y. Cao, and P. Wonka, "Designing camera networks by convex quadratic programming," in *Computer Graphics Forum*, vol. 34, no. 2. Wiley Online Library, 2015, pp. 69–80.

[35] K. Yabuta and H. Kitazawa, "Optimum camera placement considering camera specification for security monitoring," in *2008 IEEE International Symposium on Circuits and Systems*. IEEE, 2008, pp. 2114–2117.

[36] E. Hörster and R. Lienhart, "On the optimal placement of multiple visual sensors," in *Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks*. ACM, 2006, pp. 111–120.