Optimized Selection of Wireless Network Topologies and Components via Efficient Pruning of Feasible Paths

Dmitrii Kirov[1], Pierluigi Nuzzo[2], Roberto Passerone[1], Alberto Sangiovanni-Vincentelli[3]

[1] University of Trento, Italy, {dmitrii.kirov, roberto.passerone}@unitn.it
[2] University of Southern California, Los Angeles, CA, USA, nuzzo@usc.edu
[3] University of California, Berkeley, CA, USA, alberto@eecs.berkeley.edu

ABSTRACT

We address the design space exploration of wireless networks to jointly select topology and component sizing. We formulate the exploration problem as an optimized mapping problem, where network elements are associated with components from pre-defined libraries to minimize a cost function under correctness guarantees. We express a rich set of system requirements as mixed integer linear constraints over path variables, denoting the presence or absence of paths between network nodes, and propose an algorithm for efficient, compact encoding of feasible paths that can reduce by orders of magnitude the complexity of the optimization problem. We incorporate our methods in a system-level design space exploration toolbox and evaluate their effectiveness on design examples from data collection and localization networks.

1 INTRODUCTION

The ubiquitous deployment of devices in today's Internet of Things (IoT) relies on wireless networks to guarantee functionality and connectivity. Designing dependable networked systems is, however, challenging; the realization of different system components, and the network itself, is heavily affected by decisions that are made in the early stages of the design process, when it is hard to foresee the impact on the final implementation. A major bottleneck is the lack of comprehensive frameworks for scalable, multi-dimensional design space exploration under heterogeneous network requirements, such as routing, latency, and lifetime. Design exploration based on simulations or testbeds is often time-consuming and limited in the number of configurations that can be evaluated. Methodologies and tools that enable efficient co-design and provide correctness guarantees for a set of system concerns are, therefore, highly desirable.

In this paper, we build on our previous results in the context of cyber-physical system architecture exploration [4, 8, 10, 13, 14] to address the above challenges. We propose an optimization-based methodology for the exploration of wireless network architectures that allows jointly selecting the components, their physical placement, and the network routes, while satisfying a set of system requirements and minimizing a cost function. We show that a variety of network requirements, such as link quality, energy consumption, and localization, can be expressed as mixed integer linear constraints over network paths, so that an optimal design can be found by solving a mixed integer linear program (MILP). The problem size becomes, however, soon intractable in realistic scenarios. We then provide an efficient method to prune feasible network paths and generate a compact, approximate encoding of the path constraints which results in optimization problems of much lower complexity, while providing solutions that are comparable with the exact ones. Our contributions can be summarized as follows:

- We provide a mathematical formulation of the exploration problem as a mapping problem that allows simultaneously selecting optimal network topologies (node placement and routing) and components to meet a rich set of requirements.
- We propose an algorithm to decrease the complexity of exhaustive enumeration of network paths between pairs of source and destination nodes, which relies on the Yen's K-shortest path routine to generate a smaller number of promising candidate paths. Limiting the scope of the formulation to these candidates results in a more compact and effective encoding of path constraints that significantly lowers the problem complexity and increases the scalability of our approach. In our formulation, the size of the search space can be tuned to trade the execution time with the quality of the solution. Moreover, our pruning method is general and does not depend on the specific problem formalization and application domain.
- We incorporate the proposed formulation and encoding algorithm in ArchEx [10], an open-source framework for system-level architecture exploration, and evaluate their effectiveness on two design examples: a data collection network and a localization network. In ArchEx, compact and human-readable specifications can be compiled using a pattern-based formal language, while the optimization constraints are generated automatically, thus enhancing the usability and maintainability of our formulation. By leveraging the approximate path encoding, we can achieve orders of magnitude reduction in problem complexity and optimization time with respect to formulations based on the full enumeration of network paths.

Related Work. State-of-the-art approaches for optimizing device placement and connectivity in wireless networks include dynamic programming [9], evolutionary algorithms [6], and MILP based techniques [15, 16]. With respect to these approaches, our mapping constraints allow for *component sizing* in addition to *topology selection*. Many parameters that were fixed in previous formulations (e.g., transmit power, antenna gain, current consumption) can now be selected based on a library of components. While previous formulations were mostly focused on specific design objectives and metrics (e.g., lifetime [9], coverage [6], or installation cost [15]), ArchEx can capture different design objectives as well as a richer

DAC '18, June 24–29, 2018, San Francisco, CA, USA

ACM ISBN 978-1-4503-5700-5/18/06…$15.00
https://doi.org/10.1145/3195970.3196086

set of requirements, such as resiliency to network faults, and different link quality metrics. Finally, we address problems of larger size (in terms of number of network nodes) with respect to previous works [6, 9, 16].

This work differs from efforts aiming at polynomial-time approximate algorithms to solve the NP-hard exploration problem [5, 20], since it rather focuses on more compact approximate encodings that can still leverage the empirical advances of state-of-the-art MILP solvers. Previous work [16] applies Dijkstra's shortest path algorithm to devise a heuristic approach for the synthesis of large networks (>50 nodes) that cannot be handled by exact MILP-based formulations. Instead, we use Yen's algorithm [19], which is a generalization of Dijkstra's algorithm, within a MILP-based formulation, to *symbolically* generate compact mixed integer linear path constraints for networks that can scale to hundreds of nodes.

Satisfiability modulo theory (SMT) based techniques have also been proposed to find feasible solutions of wireless network scheduling problems [7], and more recently extended to solve optimization problems [11]. While we adopt a MILP-based formulation, the method for approximate encoding of path constraints proposed in this paper can also be used to lower the complexity of SMT-based formulations. Finally, our approach is complementary and can be combined with simulation-based design exploration techniques [3, 8, 13], as it provides system-level bounds that can be used to reduce the number of simulations needed for the exploration.

2 PROBLEM FORMULATION

We adopt the notion of architecture and the notation first introduced in the context of the ArchEx framework [4, 10]. We model a *network architecture* as a directed graph (V, E), where V is a set of components while each edge $e_{ij} \in E$ represents a wireless link. Edges are also interpreted as binary variables, which evaluate to one if the corresponding links are *active*, i.e., used in the network, and zero otherwise. A *template* $\mathcal{T}$ is a graph with fixed nodes but configurable links (edges). An assignment over the variables in E denotes a *configuration*, i.e., an instance of $\mathcal{T}$, marking the subset of nodes and edges of $\mathcal{T}$ that are used in the *physical topology* of the network. An edge is used if the corresponding variable evaluates to one; a node is used if it is connected. Every node $v_i \in V$ is also labeled with its coordinate vector λ_i (in an appropriate system) expressing its location. The set of locations for the nodes used in a configuration defines the *node placement*.

A network *path* (route) $\pi(v_0 \to v_n)$ is a sequence of distinct nodes $\{v_0, \dots, v_n\}$ such that $e_{v_i v_{i+1}} \in E$ for all $i = 0, \dots, n-1$. We write $|\pi|$ to denote the length of π. For a pair of nodes $q = (v_s, v_d)$, where v_s and v_d are, respectively, source and destination nodes, we define $\pi^q := \pi(v_s \to v_d)$. We also denote by Π^q the set of required paths between the nodes in q. Depending on the routing requirements, we have $|\Pi^q| = 0$ if no paths are required for q, and $|\Pi^q| > 0$ if one or more routes are needed. Each edge $e_{ij} \in E$ can be labeled with a binary variable y^{π}_{ij}, which evaluates to one if e_{ij} connects the nodes (v_i, v_j) in π, and zero otherwise. Therefore, every path π is characterized by a vector y^{π}, with $|y^{\pi}| = |E|$, marking the edges that are part of path π. Finally, we group the vectors y^{π_q} into a set $R = \{y^{\pi_q} | \pi^q \in \Pi^q, q \in Q\}$, Q being the subset of pairs of nodes in V for which at least one path is required. A *routing* is an assignment over the variables in R.

A *library* $\mathcal{L}$ is a collection of components (devices) and connection elements (wireless links), each having a set of *attributes* capturing functional and extra-functional properties. Nodes and edges of $\mathcal{T}$ are labeled with types and attributes corresponding to those in $\mathcal{L}$. We define as *component sizing* the map between "virtual" components (nodes) and connections (edges) in $\mathcal{T}$ and "real" devices and links in $\mathcal{L}$. Sizing is encoded by binary variables $m_{ij} \in M$, where m_{ij} is one if and only if component $v_j \in V$ is associated with device $l_i \in \mathcal{L}$. A similar encoding is used for connections.

Problem statement. Given a template $\mathcal{T}$ with candidate node locations and a library $\mathcal{L}$, we aim to find a network topology (E^*, R^*), i.e., node placement and routing, and a component sizing map M^* that minimize a cost function. The result is an assignment over a decision variable set $D = E \cup R \cup M$, i.e., an optimal network architecture that satisfies a set of requirements, such as routing, link quality, energy consumption, localization, expressed as MILP constraints, while minimizing a cost. In previous work, we discussed the conversion of interconnection and mapping constraints into mixed integer linear constraints [10]. We then focus on the constraints that are specific to wireless networks.

Routing constraints. We use linear arithmetic constraints to require a number of paths (routes) between nodes of the wireless network, as declared by the user. Given a set Q of source-destination pairs to be routed and a desired number of path replicas for each pair in Q, we use the following expressions to define network routes:

$$\text{(1a)} \quad \mathbf{c}(y^{\pi})^T = z^{\pi},$$

$$\text{(1b)} \quad y^{\pi}_{ij} \le e_{ij}, \qquad \forall\, i, j \in \mathbb{N} : 1 \le i, j \le |V|,$$

$$\text{(1c)} \quad \textstyle\sum_{j=1}^{|V|} y^{\pi}_{ij} \le 1,\ \sum_{j=1}^{|V|} y^{\pi}_{ji} \le 1, \qquad \forall\, i \in \mathbb{N} : 1 \le i \le |V|,$$

$$\text{(1d)} \quad y^{\pi_1}_{ij} + y^{\pi_2}_{ij} \le 1, \qquad \forall\, i, j \in \mathbb{N} : 1 \le i, j \le |V|,$$

$$\text{(1e)} \quad \textstyle\sum_i \sum_j y^{\pi}_{ij} \le (\ge, =)\, N^*_{hops}.$$

Constraints (1a)-(1c) are formulated for every required path. Constraint (1a) is a balance equation, which ensures that source and sink of π are connected by a path. The incidence matrix $\mathbf{c}$ is defined by considering all the possible edges in $\mathcal{T}$, while z^{π} is a column vector of length $|V|$, with $z^{\pi}_s = 1$, $z^{\pi}_d = -1$, and zero otherwise, s and d being indices denoting, respectively, the source and destination of π. Constraints (1b) relate the variables y^{π}_{ij} with the edge variables e_{ij} in E: if y^{π}_{ij} is true, then there must be an edge between nodes i and j. Constraints (1c) forbid loops in a path: every node in π must have at most one predecessor and at most one successor. The remaining constraints capture additional routing concerns. Constraints (1d) require two paths π_1 and π_2 to be disjoint, while constraint (1e) is used to set the maximum (minimum, exact) length of path π in terms of number of hops.

Link quality constraints. Many quality-of-service metrics of a wireless network, such as latency, packet loss, and energy consumption, depend on the link quality (LQ). It is then important to specify a bound on the quality of every link of a route. We can express an LQ constraint as a bound on the received signal strength (RSS) of a link as follows:

$$\text{(2a)} \quad RSS_{ij} = PL_{ij} + tx_i + g_i + g_j, \qquad 1 \le i, j \le |V|,$$

$$\text{(2b)} \quad RSS_{ij} y^{\pi}_{ij} \ge RSS^*, \qquad 1 \le i, j \le |V|.$$

RSS values for every link e_{ij} between a transmitter (TX) v_i and receiver (RX) v_j are real-valued decision variables. Constraint (2a) computes RSS_{ij} as a sum of the link path loss PL_{ij}, TX and RX antenna gains g_i and g_j, and TX power tx_i. The value of PL_{ij} can either be analytically estimated using a channel model or obtained from measurements; the other parameters come from the component attributes in $\mathcal{L}$. For example, let $g^{\mathcal{L}}$ be the vector of antenna gains for the components in $\mathcal{L}$ and m_{ij} be the element of the mapping matrix **m** associated with node v_j and component $l_i \in \mathcal{L}$; then, we have $g_j = \sum_{i=1}^{|\mathcal{L}|} m_{ij} g_i^{\mathcal{L}}$. Constraint (2b) places a lower bound RSS^* on all links belonging to path π using the variables y^π. The products in (2b) can be turned into linear terms using standard encoding techniques which we omit for brevity.

In addition to the RSS, ArchEx also supports other link quality metrics, such as bit error rate (BER) and signal-to-noise ratio (SNR), for which encodings are provided using linear or piecewise-linear functions. Because some of the metrics depend on the communication frequency and modulation, these are both part of the specification. Furthermore, to characterize the channel and compute the path loss values PL_{ij} for every link e_{ij} ArchEx supports several models with different complexity. In this work, we use the multi-wall model, an extension of the classical log-distance model, which also accounts for the attenuation in walls and other obstacles.

Energy consumption constraints. Every component in V can be labeled with the current drawn by its hardware (e.g., radio, CPU, sensors) in different operating modes. In this example, we distinguish between the radio TX and RX current c^{TX} and c^{RX}, while the remaining current values for active and sleep modes are cumulatively denoted by c^{active} and c^{sleep}, respectively. We also assume a collision-free TDMA protocol, in which the nodes wake up only within a few dedicated time slots for sending and receiving packets. There are n slots in a superframe, each with duration t^{slot}, so the superframe duration is $t^{SF} = nt^{slot}$. The following expressions can be written for each node v_i:

$$\text{(3a)} \quad \frac{B_i}{\epsilon_i^{radio} + \epsilon_i^{active} + \epsilon_i^{sleep}} \cdot t^{SF} \geq L^*$$

$$\text{(3b)} \quad \epsilon_{ij}^{TX} = ETX_{ij} \cdot c_{ij}^{TX} \cdot \frac{\mu}{b_{ij}}, \qquad \forall\, j \in \mathbb{N} : 1 \leq j \leq |V|.$$

The left-hand side of constraint (3a) is the *lifetime* of v_i, which is required to be greater than L^*. B_i is the node battery capacity, while the denominator is the total energy consumed within a superframe. The term ϵ_i^{radio} is the energy required for transmitting and receiving all packets for all routes in which v_i is involved. It can be expressed as $\epsilon_i^{radio} = \epsilon_i^{TX} + \epsilon_i^{RX} = \sum_\pi \left(\sum_j \epsilon_{ij}^{TX} y_{ij}^\pi + \sum_j \epsilon_{ji}^{RX} y_{ji}^\pi \right)$, where ϵ_{ij}^{TX} and ϵ_{ji}^{RX} are real-valued decision variables denoting the energy values of TX/RX links from/to node v_i. The remaining terms can be computed as $\epsilon_i^{active} = c_i^{active} \cdot t^{slot} \cdot k$ and $\epsilon_i^{sleep} = c_i^{sleep} \cdot t^{slot} \cdot (n - k)$, where k is the number of slots in which v_i must either transmit or receive, under the assumption that each TX and RX requires a separate slot.

Constraints (3b) compute the energy ϵ_{ij}^{TX}, where μ is the packet length, b_{ij} is the bit rate associated with link e_{ij}, and ETX is the number of expected transmissions of a packet necessary for it to be received without error at its destination. ETX depends on the path loss and interference. In this work, we neglect the effect of packet collisions and model the interference with the background noise γ_{ij} associated with every link. ETX_{ij} is then computed from the corresponding signal-to-noise value (SNR_{ij}), which is equal to $RSS_{ij} - \gamma_{ij}$. The mathematical expression of the constraints is omitted for brevity. Nonlinear terms in (3a) and (3b) can all be expressed in linear form using standard techniques. Similar constraints can be used to compute ϵ_{ji}^{RX}, as well as the required energy for contention-based protocols.

Localization constraints. We focus on range-based localization systems that estimate distances between anchor nodes and a target node by using the received signal strength, time of arrival, or other metrics. Evaluation of such systems is typically performed using a set of locations in the network deployment area, in which the quality of localization (e.g., accuracy, precision) is estimated [18]. This set of "evaluation locations" can be seen as possible locations of a mobile device. Let Λ^{eval} be the array of these locations, while $\Lambda^{\mathcal{T}}$ is the array of locations of nodes in $\mathcal{T}$, so that $|\Lambda^{\mathcal{T}}| = |V|$. Also, let r_{ij} be the entries of the reachability matrix **r**. We impose the following constraints:

$$\text{(4a)} \quad r_{ij} = (RSS_{ij} \geq RSS^*) \wedge \delta_i, \quad 1 \leq i \leq |\Lambda^{\mathcal{T}}|,\ 1 \leq j \leq |\Lambda^{eval}|,$$

$$\text{(4b)} \quad \sum_{i=1}^{|\Lambda^{\mathcal{T}}|} r_{ij} \geq N, \qquad \forall\, j \in \mathbb{N} : 1 \leq j \leq |\Lambda^{eval}|,$$

where δ_i is a binary variable equal to one if the component v_i is used and zero otherwise. Constraint (4a) forces the value of r_{ij} to be true if the mobile node located at $\lambda_j^{eval} \in \Lambda^{eval}$ is *reachable* by v_i, i.e., it is able to receive the signal from a node v_i located at $\lambda_i^{\mathcal{T}} \in \Lambda^{\mathcal{T}}$ with signal strength of at least RSS^*. The values of RSS_{ij} can be computed similarly to (2a) and other LQ metrics can also be used instead of RSS. Constraint (4b) requires that every location from Λ^{eval} be reachable by at least N nodes from V. It can be used to ensure, for example, that at least 3 distances to the target node are computed, which allows to calculate its 2D position using trilateration. This requirement does not depend on the ranging technique and can guarantee a reliable coverage of the area.

Cost function. We associate every node and every edge in $\mathcal{T}$ with a cost value. This value can be related to any attribute or their combination, e.g., monetary cost, weight, energy, or lifetime. We then consider objective functions combining different concerns as weighted sums, where the weights are set by the user.

3 APPROXIMATE ENCODING OF PATHS

Any path π can be encoded using n^2 variables from the set y^π, n being equal to $|V|$, which correspond to all the edges of $\mathcal{T}$. This encoding allows exhaustive exploration of network topologies, since any node and any edge may be a member of π, but becomes inefficient when either the size of $\mathcal{T}$ or the number of required paths increase. For every path π at least $n^2 + 3n$ constraints (formulas (1a)-(1c)) are added to the problem formulation. Variables from y^π are further used in other network constraints (e.g., LQ or energy constraints), often multiplied by other decision variables. Each product of binary variables must be translated into a linear constraint by introducing auxiliary variables and constraints to the original non-linear formulation. All of these steps may result in a significant growth in problem size and solver time.

We propose to trade generality with complexity by implementing a more compact, yet approximate, encoding of network paths. The main idea behind our method is to direct the search toward a smaller number of candidate paths. We use the estimated link path loss as a weight for the edges and execute Yen's K-shortest path algorithm [19] to select a number K^* of path candidates that minimize the overall path loss for every network route specified in

Algorithm 1: Approximate path encoding

```
Given: Network template $\mathcal{T} = (V, E)$
Input: Set $Q$ of pairs $(s, d)$, path loss matrix $PL$, # candidate paths $K^*$
Output: Set $R = \{y^q | q \in Q\}$ of path variables, set $Cons$ of path
        constraints
1  $Cons \leftarrow \emptyset$
2  $Q^+ \leftarrow$ FindReplicas($Q$)
3  forall $q \in Q^+$ do
4      $(K, N^{rep}) \leftarrow$ BreakDown($K^*$)
5      $(y_1^q, \ldots, y_{|E|}^q) \leftarrow 0$; $PL' \leftarrow PL$; $NewCons \leftarrow$ false
6      for $n = 1$ to $N^{rep}$ do
7          $(p_1, \ldots, p_K) \leftarrow$ kShortest($PL'$, $s_q$, $d_q$, $K$)
8          for $k = 1$ to $K$ do
9              $v \leftarrow$ GetVariables($p_k$)
10             $y^q \leftarrow$ AddVariables($v$)
11             $NewCons \leftarrow NewCons \vee \bigwedge_{i=1}^{|v|} v_i$
12         $PL' \leftarrow$ DisconnectMinDisjointPath($PL'$, $(p_1, \ldots, p_K)$)
13     $R \leftarrow R \cup y^q$
14     $Cons \leftarrow Cons \cup NewCons$
15 return $(R, Cons)$
```

the requirements. We then symbolically encode the proposed paths using a smaller number of edge variables to obtain the final path constraint, as summarized in Algorithm 1.

The function FindReplicas(Q) extends the input set Q to a set Q^+ with a number of copies of each source-destination pair (s, d) corresponding to the required amount of path replicas, i.e., redundant paths, required for this pair by the designer. By analyzing the routing requirements for a pair $q \in Q^+$, the function BreakDown(K^*) splits the required number of candidate paths K^* into N^{rep}, the required number of *disjoint* replicas for q, and K, the required number of candidate paths for each replica, such that $N^{rep} \cdot K \geq K^*$ (line 4). Then, q is associated with a vector y^q, where $|y^q| = |E|$, and K^* candidate paths are generated for q as follows (lines 6-13). kShortest runs Yen's K-shortest path routine to generate the K "best" paths $p_1 \ldots p_K$ in non-decreasing order of cost (line 7), by using the link path loss matrix PL to assign weights to the edges. Every generated path p_k is then processed and a binary variable is assigned to every edge between the nodes of p_k (line 9). The vector v of these variables is added to y^q (line 10). The path constraint $NewCons$ is also updated (line 11) to require that one of the proposed paths be selected in the final topology. The path generation procedure above is repeated N^{rep} times. At each iteration, the function DisconnectMinDisjointPath identifies a path that has the largest number of edges in common with other paths, i.e., it is "minimally disjoint" from others. This path is disconnected from the graph, so that the following iteration will generate at least one new candidate path that is completely independent from the previous ones. In this way, we guarantee that at least N^{rep} of the K^* proposed paths will be disjoint, as per the routing requirement. In practice, since the edges that occur in most of the previously generated paths cannot be used anymore, the number of resulting disjoint paths is larger than N^{rep}, which results in multiple feasible solutions. As we execute the shortest path routine, we can disconnect a path by appropriately setting the weights of corresponding edges. Similarly, we can disregard links with path loss below a certain threshold to ensure that the all the candidate paths meet the LQ requirements. The process is repeated for every q until all the candidate paths and the corresponding constraints are generated.

By Algorithm 1, the worst case number of path variables y^π needed for every required route is $K^*(n-1)$, rather than n^2, assuming that every new path consists of $n = |V|$ nodes and all the K^* paths are disjoint. However, the situation is much better in practice, since the actual paths typically contain only few hops and share multiple links. The complexity of many optimization constraints (e.g., LQ and energy) is then reduced, since they only need to be defined for the nodes and the edges in the candidate paths. Further, the routing constraints (1a)-(1c) can be omitted, since the validity of the generated paths is guaranteed by the shortest path routine. K^* can be adjusted, as discussed in Sec. 4.3, to trade optimality with execution time. Finally, the proposed path pruning algorithm is general and can be applied to any architecture that can be modeled as a weighted directed graph, independently of the specific application domain.

4 IMPLEMENTATION AND EVALUATION

ArchEx is implemented in Matlab by leveraging CPLEX [1] for solving MILPs, and Yalmip [12] for facilitating the problem formulation. The tool accepts as inputs a problem description, a library of components and a floor plan. The first two are text files, while the latter is an SVG file, which stores information about space dimensions, obstacles (e.g., walls, doors, windows) and locations of network devices. The problem description includes system requirements as well as the parameters of the channel model, the protocol, and the battery. Below we demonstrate the effectiveness of our approach on two classical wireless sensor network (WSN) design problems that are central to many IoT applications, namely, data collection and localization. All experiments were performed on an Intel Core i7 3.4-GHz processor with 8-GB RAM.

4.1 Data Collection Network

We first consider an indoor WSN for periodic data collection. This network consists of end devices (sensors), which measure or detect some physical environment phenomena, one or more base stations, which collect and process the sensor data, and routing devices, or relays, that forward the messages towards the base station. The floor plan of the building is shown in Fig. 1a. There are 35 sensors (in green) and one base station (in red) whose positions are fixed. The remaining nodes represent candidate locations for relays. The total number of nodes in the template $\mathcal{T}$ is 136.

Our reference library $\mathcal{L}$ includes the following components: Sensor, Relay, and Sink. Each element is labeled with its cost, TX power, antenna gain, and current consumptions for the radio and other hardware components, as explained in Section 2. The characteristics are based on commercial WSN transceivers and integrated circuits [2]. In this example, we assume a bit rate of 250 kbps and a noise level of -100 dBm for all links, as well as a 2.4-GHz frequency and a QPSK modulation. Further, our network uses a TDMA protocol with a slot duration of 1 ms and 16 slots in a superframe, and a packet length of 50 bytes. Sensors transmit a packet every 30 s and have zero cost. We assume that the power is constrained by two 1.5-V AA batteries, each of 1500 mAh.

Every sensor in a data collection network must have a path to the base station. We also improve the network resiliency to faults by adding some redundancy. To do this, we require two disjoint routes for every sensor to the base station by using the patterns `name = has_path(A,B)` and `disjoint_links(name1,name2)`, where `name`, `A`, and `B` refer, respectively, to a path, a source, and a sink.

Table 1: Final number of nodes, dollar cost, average node lifetime (in years), and solver time for a data collection WSN optimized for different objectives.

Objective	# Nodes	$ cost	Lifetime (y)	Time (s)
$ cost	61	1022	7.33	45
Energy	63	1480	12.24	260
$ + Energy	61	1241	9.69	66

We require 70 routes in total, 2 for each of 35 sensors. The pattern `min_signal_to_noise` is imposed to specify a minimum SNR of 20 dB for every link. Finally, with the `min_network_lifetime` pattern we require the node batteries to last for at least 5 years.

Table 1 shows the solver time and the synthesis results obtained while optimizing for different objectives: dollar cost, network energy consumption, and an equally weighted combination. Fig. 1b shows the result of mapping the requirements to an aggregation of library devices minimizing the overall dollar cost. The selected components have different TX power, while some of them also have an external antenna to satisfy the LQ constraints. On the other hand, minimizing for energy consumption results in a network with a much higher dollar cost. In fact, power consumption can be reduced by decreasing the TX power. However, for certain links, this would result in the violation of LQ constraints, unless more expensive low-power components are selected, for example, with smaller radio TX and RX currents or smaller CPU standby current. The tradeoff between dollar cost and energy consumption can be explored when optimizing for a combination of objectives.

For each experiment we obtained MILP formulations of around 1.5×10^5 constraints and 4.5×10^4 variables. The number of candidate paths K^* generated by Algorithm 1 for every required connection was set to 10. Exhaustive path enumeration led to problems with over 10^7 constraints and 1.5×10^6 variables, which required several hours just for problem encoding, to be contrasted with the few minutes required by our approach. The execution time with exhaustive path enumeration always exceeded the 8-h timeout. Overall, our path encoding algorithm reduced the problem size by two orders of magnitude. In ArchEx, the constraints were automatically generated from a pattern-based specification including only 150 lines of code, and resulted in designs that favorably compare with the ones in the literature [5, 6, 9, 15, 16] in terms of size, number of supported requirements, and dimensionality of the design space, including the sizing of the network components.

4.2 Localization Network

We specify 150 candidate node positions and 135 evaluation (mobile node) locations for the same building floor. Our localization system has a star topology, where anchor nodes that have to be allocated by the tool communicate directly with the mobile device. The latter estimates its position using a set of distance measurements obtained from the anchors. The `min_reachable_devices` pattern implements the localization constraints (4a)-(4b), and we apply it to require that, at every test point, the mobile node must be able to receive signals from at least 3 distinct anchors. Furthermore, with the same pattern, we request that only reliable links with a minimum RSS of -80 dBm be selected. This also contributes to decreasing the ranging error, which rapidly grows for larger path losses and unstable signals.

We solve the problem for two different cost functions: dollar cost (result in Fig. 1c) and difference of sum of distances (DSOD)

Table 2: Final number of nodes, dollar cost, average number of reachable anchors by the mobile node, and solver time for a localization network optimized for different objectives.

Objective	# Nodes	$ cost	Reachable	Time (s)
$ cost	28	1050	3.1	115
DSOD	24	1310	3.6	121
$ + DSOD	24	1180	3.03	144

between network nodes and test points. The latter was proposed in the literature [17] as a linear version of the Cramer Rao lower bound – a metric used in the accuracy evaluation of localization systems. To set up the reachability matrix **r** and the localization constraints, we use Algorithm 1 with $K^* = 20$ candidate anchors for every test point.

Results in Table 2 show that optimizing for the DSOD objective produces a placement with smaller number of more expensive nodes equipped with antennas, i.e., their signal can reach more test locations. This system also has smaller power consumption. In all experiments, the number of variables and constraints counts up to, respectively, 3×10^4 and 3.5×10^4. A full enumeration of all test points reachable by all anchors would lead to several millions variables and constraints, making the design exploration intractable.

4.3 Impact of Approximate Path Encoding

We test the scalability of our techniques on data collection network architectures with an increasing number of total nodes and end devices (sensors) in the template $\mathcal{T}$, and with $K^* = 10$, as reported in Table 3. The dramatic reduction in problem complexity and execution time shows the advantage of using the approximate encoding of network paths in Algorithm 1. We also provide the measured (or estimated, for larger instances) number of constraints in the case of full path enumeration, which are several orders of magnitude larger. Execution times of the order of days are expected to solve these large problem instances, since only a few smaller networks were synthesized within an 8-h timeout.

The effect of different values for K^* is illustrated in Table 4 when the objective is the dollar cost, but similar trends were also observed for other objectives. Comparison with the optimal solution obtained without approximation is only possible for the small WSN template, since exhaustive exploration becomes soon intractable. Increasing K^* leads to higher quality results in terms of cost; when $K^* \to \infty$ all possible paths are enumerated, leading to the global optimum. However, large values of K^* also result in a substantial growth in execution time; a small decrement in cost for $K^* > 10$ comes at a very large price in terms of performance. On the other hand, for small values of K^*, slightly increasing K^* significantly improves the cost function while incurring a reasonable time overhead. When $K^* = 1$ only one candidate path is proposed for every required route, i.e., the routing is fixed. In this case, the performance is comparable to the one achieved by heuristic algorithms previously proposed in the literature [16], but we additionally guarantee optimal component sizing for the selected topology.

For a given problem, K^* can be systematically selected by a search algorithm that generates multiple topologies for different values of K^* and terminates once the execution time becomes higher than a predefined threshold or there is no further improvement in the objective. In fact, as K^* increases, we obtain solutions that are at least as good as, if not better than, the previous ones. For example, for network sizes in the range of our examples, a guideline

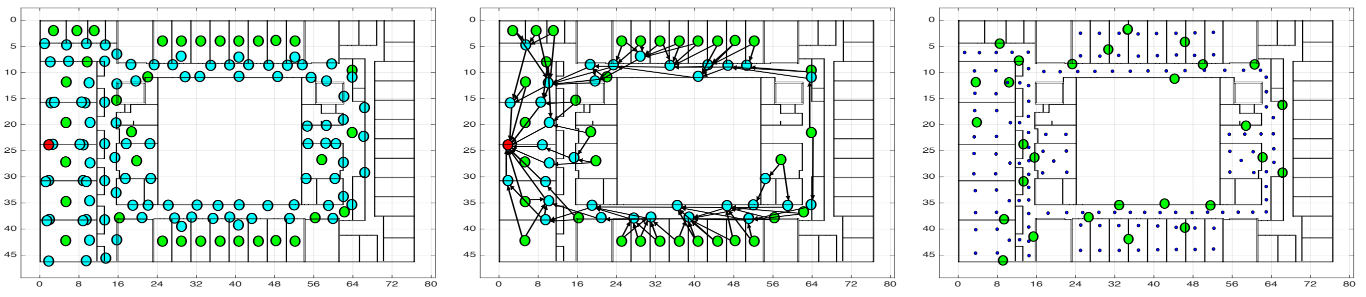

Figure 1: (a) Template for the data collection WSN (sensors, base station and candidate locations of relays); (b) Generated topology of the data collection WSN; (c) Evaluation points and generated anchor placement for the localization network.

Table 3: Number of constraints and solver time for different network architecture sizes generated by using the approximate path encoding algorithm ($K^* = 10$) compared to full enumeration of paths.

#Nodes (total in $\mathcal{T}$)	**#End devices** (to be routed)	**#Constraints, $\times 10^3$** (full / approx)	**Time (s)** (full / approx)
50	20	862 / 24	8233 / 12
100	20	1743 / 54	TO / 28
100	50	~ 3800 / 125	TO / 55
100	75	~ 4800 / 150	TO / 93
250	50	~ 3500 / 108	TO / 340
250	100	~ 5700 / 175	TO / 1175
250	200	~ 10000 / 310	TO / 1708
500	50	~ 7400 / 230	TO / 818
500	100	~ 11000 / 346	TO / 5330
500	200	~ 21000 / 655	TO / 8354

Table 4: Costs and solver times for data collection networks with a small template $\mathcal{T}_1$ (20 end devices, 50 nodes total) and a larger template $\mathcal{T}_2$ (200 end devices, 250 nodes total) synthesized using different values of K^*, compared with the optimal solution (only for $\mathcal{T}_1$).

	Result	**$K^* = 1$**	**$K^* = 3$**	**$K^* = 5$**	**$K^* = 10$**	**$K^* = 20$**	**opt**
$\mathcal{T}_1$	Cost ($)	920	861	805	642	619	579
	Time (s)	3	7	10	12	442	8233
$\mathcal{T}_2$	Cost ($)	2594	2280	2083	1909	1842	-
	Time (s)	8	85	358	1708	15334	TO

would be to select K^* between 3 and 10, since values outside of this interval provided marginal advantages in terms of cost versus execution time.

5 CONCLUSION

We presented an optimization-based methodology for wireless network architecture exploration, where network components, placement, and routes are jointly selected to minimize an objective under correctness guarantees. We proposed an approximate encoding of the path constraints that makes the formulation and the solution of large problems tractable. We validated the proposed techniques, implemented as a part of the ArchEx toolbox, on network designs of realistic size. Future work includes investigating the combination of our methods with simulation, and using them to explore design tradeoffs across the HW/SW boundary (e.g., including communication protocol parameters).

REFERENCES

[1] (2018, Apr.). IBM ILOG CPLEX Optimizer. [Online]. Available: http://www.ibm.com/software/commerce/optimization/cplex-optimizer/.

[2] (2018, Apr.). Texas Instruments: Zigbee products. [Online]. Available: http://www.ti.com/lsds/ti/wireless-connectivity/zigbee/products.page.

[3] A. Davare et al. metro II: A Design Environment for Cyber-Physical Systems. *ACM Transactions on Embedded Computing Systems*, 12(1s), 2013.

[4] N. Bajaj, P. Nuzzo, M. Masin, and A. Sangiovanni-Vincentelli. Optimized selection of reliable and cost-effective cyber-physical system architectures. In *Proc. Design, Automation and Test in Europe*, pages 561–566, 2015.

[5] A. Chelli, M. Bagaa, D. Djenouri, I. Balasingham, and T. Taleb. One-step approach for two-tiered constrained relay node placement in wireless sensor networks. *IEEE Wireless Communications Letters*, 5(4):448–451, 2016.

[6] D. S. Deif and Y. Gadallah. Wireless sensor network deployment using a variable-length genetic algorithm. In *Proc. of the Wireless Communications and Networking Conference (WCNC)*, pages 2450–2455, 2014.

[7] Q. Duan, S. Al-Haj, and E. Al-Shaer. Provable configuration planning for wireless sensor networks. In *Proc. of the 8th International Conference on Network and Service Management*, 2012.

[8] J. Finn, P. Nuzzo, and A. Sangiovanni-Vincentelli. A mixed discrete-continuous optimization scheme for cyber-physical system architecture exploration. In *Proc. of the Int. Conf. Computer-Aided Design (ICCAD)*, pages 216–223, 2015.

[9] A. Gogu, D. Nace, E. Natalizio, and Y. Challal. Using dynamic programming to solve the wireless sensor network configuration problem. *Journal of Network and Computer Applications*, 83:140–154, 2017.

[10] D. Kirov, P. Nuzzo, R. Passerone, and A. Sangiovanni-Vincentelli. ArchEx: An Extensible Framework for the Exploration of Cyber-Physical System Architectures. In *Proc. of the 54th Design Automation Conference (DAC)*, 2017.

[11] Y. Li, A. Albarghouthi, Z. Kincaid, A. Gurfinkel, and M. Chechik. Symbolic optimization with SMT solvers. In *ACM SIGPLAN Notices*, volume 49, 2014.

[12] J. Löfberg. YALMIP : A Toolbox for Modeling and Optimization in MATLAB. In *Proceedings of the CACSD Conference*, Taiwan, 2004.

[13] A. Moin, P. Nuzzo, A. L. Sangiovanni-Vincentelli, and J. M. Rabaey. Optimized Design of a Human Intranet Network. In *Proceedings of the 54th Design Automation Conference (DAC)*, 2017.

[14] P. Nuzzo, H. Xu, N. Ozay, J. B. Finn, A. L. Sangiovanni-Vincentelli, R. M. Murray, A. Donzé, and S. A. Seshia. A contract-based methodology for aircraft electric power system design. *IEEE Access*, 2:1–25, 2014.

[15] A. Pinto, M. D'Angelo, C. Fischione, E. Scholte, and A. Sangiovanni-Vincentelli. Synthesis of embedded networks for building automation and control. In *Proceedings of the American Control Conference (ACC)*, pages 920–925, 2008.

[16] A. Puggelli, M. M. R. Mozumdar, L. Lavagno, and A. L. Sangiovanni-Vincentelli. Routing-aware design of indoor wireless sensor networks using an interactive tool. *IEEE Systems Journal*, 9(3):714–727, 2015.

[17] A. E. Redondi and E. Amaldi. Optimizing the placement of anchor nodes in RSS-based indoor localization systems. In *Ad Hoc Networking Workshop (MED-HOC-NET)*, pages 8–13. IEEE, 2013.

[18] T. Haute et al. Performance analysis of multiple Indoor Positioning Systems in a healthcare environment. *Intern. Jour. of Health Geographics*, 15(1), 2016.

[19] J. Y. Yen. Finding the k shortest loopless paths in a network. *Management Science*, 17(11):712–716, 1971.

[20] Y. Zhou, Z. Sheng, C. Mahapatra, V. C. Leung, and P. Servati. Topology design and cross-layer optimization for wireless body sensor networks. *Ad Hoc Networks*, 59:48–62, 2017.