# Energy-Aware IoT Deployment Planning

Peiyuan Guan
University of Oslo
Oslo, Norway
peiyuang@ifi.uio.no

Animesh Dangwal
University of California, Santa Barbara
Santa Barbara, CA, USA
animesh@ucsb.edu

Amir Taherkordi
University of Oslo
Oslo, Norway
amirhost@ifi.uio.no

Rich Wolski
University of California, Santa Barbara
Santa Barbara, CA, USA
rich@cs.ucsb.edu

Chandra Krintz
University of California, Santa Barbara
Santa Barbara, CA, USA
ckrintz@cs.ucsb.edu

## ABSTRACT

Increasingly, the Internet of Things (IoT) is evolving toward an architecture consisting of sensing and actuation devices communicating with edge computers and storage systems. These "edge deployments" localize communication, computation, and storage for security, increased efficiencies (e.g. lower latency response), and reliability. In settings where electrical power infrastructure is lacking, however, these deployments typically rely on renewable energy and battery storage for power.

In this paper, we investigate power-optimizing scheduling for IoT communication in edge deployments that compose battery-powered sensors. We focus on radio communication since it is often the largest component of the sensor battery budget in edge deployments. We model radio communication between sensors and co-located base stations using their distance, bandwidth availability, and duty cycle for data transmission. We present three heuristic approaches that allocate radio bandwidth to minimize sensor power consumption. We consider both shared and decentralized battery infrastructure. We empirically analyze these approaches in terms of performance and computational efficiency and present a methodology for using these techniques to inform configuration and management of energy-efficient edge deployments.

## CCS CONCEPTS

• **Computing methodologies** → **Simulation evaluation**; *Modeling methodologies*; • **Networks** → *Mobile networks*; *Network management*; • **Hardware** → **Power estimation and optimization**; **Energy distribution**.

## KEYWORDS

communication scheduling, numerical optimization, IoT deployment planning


*CF '24, May 7–9, 2024, Ischia, Italy*

ACM ISBN 979-8-4007-0597-7/24/05
https://doi.org/10.1145/3649153.3649864

**ACM Reference Format:**
Peiyuan Guan, Animesh Dangwal, Amir Taherkordi, Rich Wolski, and Chandra Krintz. 2024. Energy-Aware IoT Deployment Planning. In *21st ACM International Conference on Computing Frontiers (CF '24), May 7–9, 2024, Ischia, Italy.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3649153.3649864

## 1 INTRODUCTION

The Internet of Things (IoT) describes a computing fabric that enables ordinary, physical objects to sense, analyze, actuate, and control their environment automatically. To enable this, IoT deployments increasingly consist of co-located computational and sensing devices that interoperate at the "edge" of the network. Edge computing provides low-latency request response, enhanced reliability, failure resiliency, reduced bandwidth use, and improved security and privacy, versus the centralized, cloud-direct model [16].

Edge deployments consist of devices with a wide range of capabilities, radio technologies, and resource constraints depending on the physical environments in which they are embedded and the workloads they support. This heterogeneity makes edge deployments very complex and challenging to design and allocate resources efficiently. Moreover, these challenges are compounded when electrical power infrastructure is lacking, forcing their reliance on renewable energy and battery storage for power. As such, novel approaches to resource management are needed to aid IoT deployment configuration and resource scheduling for such settings.

Toward this end, we investigate a new approach for designing edge deployments that compose battery-powered sensors. Recent research has shown that it is possible to formulate IoT scheduling problems (e.g. for resource allocation, energy optimization, improving QoS, etc.) as multi-objective, numerical optimization problems and to use optimization software (i.e., solvers) to efficiently produce possible solutions [1, 7, 20, 32, 33]. In this work, we use multi-objective optimization to efficiently allocate radio bandwidth to sensors in an edge deployment to minimize power consumption. We focus on radio communication because it is a primary consumer of power for wireless sensor networks that are remotely deployed and spatially distributed [17, 22, 24] – key characteristics (cf. Section 3) of the edge deployments that we target.

We model the communication between sensors and co-located access points using their distance, the total available bandwidth, and the period of time used by the sensors for data transmission, i.e.

their duty cycle. We investigate three popular heuristic approaches: MIDACO [1], a Mixed Integer Programming (MIP) and Mixed Integer Non-Linear Programming (MINLP) solver, Z3 [7] a Satisfiability Modulo Theory (SMT) solver, and a Genetic Algorithm (GA) that we specifically designed for this project. The heuristics allocate radio bandwidth to devices placed at different distances from the access point such that the worst case energy use is minimized. We consider per-node and shared battery infrastructure and empirically compare the approaches using different experimental settings. We evaluate the impact of limiting computational budget and the use of different device duty cycles (which impose deadlines on data transfer completion). Finally, we present a methodology for using these techniques to guide energy efficient, configuration planning and sensor placement for remote, edge-based IoT deployments.

Our results show that MIDACO outperforms SMT and GA, and that SMT outperforms GA in terms of solution quality. In our study, MIDACO is also always able to generate feasible solutions, even with very limited computational effort budgets. However, in some cases where the computation effort budget is limited, SMT is able to find better solutions compared to MIDACO. It is just not possible to predict whether SMT will do so for a specific deployment and effort budget. This result indicates that it is "best" to expend a fraction of the available computational budget on SMT and if it fails to find a feasible solution in using that fraction, invoke MIDACO with the remainder. We also find that this approach can be used to size of the battery infrastructure for a given deployment and that infeasible solutions can be intelligently adjusted (adapting sensor placement or duty cycle) while minimizing power use.

## 2 RELATED WORK

Past work has focused on resource allocation for wireless sensor networks (WSNs) [30, 31, 36]. Federated Learning [35] is a promising technology that couples bandwidth and power management to design communication-efficient systems. The authors of [20, 32, 33] show that numerical optimization solvers can be used effectively for scheduling resources, optimizing energy use, and improving quality of service for edge systems. Our work differs from this past work in that we use independent objectives (versus sum-weighted) to minimize power consumption and we show how to use solver-based solutions to co-design power infrastructure and IoT deployments (sensor placement and duty cycle) at the edge.

We investigate three popular heuristics for allocating radio bandwidth to sensors in an edge deployment, such that power consumption is minimized. They include MIDACO [1], Z3 [7], and a custom genetic algorithm (GA; cf. Section 3.3.3). MIDACO is a numerical high-performance solver for single- and multi-objective optimization. MIDACO is based on a derivative-free, evolutionary hybrid algorithm that treats the objective and constraint functions as a black-box which may contain critical function properties like non-linearity, non-convexity, discontinuities, or stochastic noise.

Satisfiability Modulo Theories (SMT) solvers can also be used for constraint solving and optimization. They are used for application scheduling [2, 3, 5, 25, 28], resource allocation for the edge-cloud continuum [5, 11, 21, 28], and network optimization in Wireless Sensor Networks (WSNs) and Software Defined Networks(SDNs) for time sensitive applications [8, 12, 21, 27]. SMT is also considered a competitive alternative [4, 12, 25] to Mixed Integer Programming(MIP)/Mixed Integer Linear Programming (MILP) solvers (like MIDACO). SMT allows for more flexibility in constraint declaration compared to MIP/MILP solvers [4, 5, 12, 25]. We evaluate the use of the Z3 [7] SMT solver in this paper, and empirically compare its performance against MIDACO and GA.

Similarly, Gosh et al. explore the use of the optimization in a Satisfiability Modulo Convex Optimizer (SMC) using the Z3 SMT solver for WSNs in [12]. In this prior work, the authors compare Z3 against the CPLEX MILP solver [6], for synthesizing "pop up" IoT networks which are temporary, short lived WSNs used during crisis events [9]. They target constraints for coverage, visibility, and connectivity. They also investigate how to scale the solvers to construct larger networks hierarchically, and perform power optimization by controlling the number of sleeps/wake up cycles. While we also explore topics on sensor placement and power optimization, our approach differs in both target application and placement formation. Our focus is on permanent deployments for remote and highly resource constrained settings, as well as on tools that help deployment operators size their battery and renewable power generation, while minimizing worst case energy across deployments.

For more permanent sensor network deployments, the authors of [8, 23, 27] optimize node placement using network coverage and connectivity and conserve energy by manipulating sleep/wake cycles (similar to Gosh). In [23], the authors use a hybrid approach that combines mathematical modeling and Genetic Algorithms (GA) to do so. In our work, we optimize energy use across an edge deployment by focusing on radio communication (distance and payload size) and use the solutions to guide configuration of energy-efficient edge deployments.

## 3 EDGE DEPLOYMENT OPTIMIZATION

Edge environments for IoT applications vary widely. The focus of our work is on edge deployments that can be characterized as hard to reach/maintain, intermittently disconnected (from the Internet/cloud), large scale in terms of spatial layout, containing various obstacles (trees, vehicles, people etc.), with heterogeneous, resource constrained sensors and a small number of resource rich edge computers (aka base stations). The sensors (and in some cases the base stations) rely on solar energy and battery storage for electrical power (particularly for operation in darkness). These edge deployments are typical of digital agriculture, sustainable ecology (e.g. wildfire and animal monitoring), and intelligent infrastructure (e.g. smart roadways, bridges, and buildings).

Our model of operation is inspired by real-world remote deployments that our group maintains as part of the UCSB SmartFarm project [10, 15, 18, 29]. In this setting, sensors monitor the environment, periodically transmit their data on a duty cycle over radio links to one or more base stations, and deactivate (e.g. go into "deep sleep") to conserve battery power between duty cycles.

These deployments are employed by farms for a dynamically changing set of purposes related to precision agriculture, requiring them to be periodically updated for different use cases (by humans or ground robots). The base station is often grid-powered and part of a resource rich edge cloud (with intermittent Internet connectivity)

that is sited on-farm in an outbuilding. In this work, however, we also explore the use case where the base station or stations are also solar and battery powered using off-grid power infrastructure with greater capacity than the infrastructure supporting the sensors. The sensor nodes are battery powered and each sensor can have its own battery or multiple sensors can draw from a shared battery (both alternatives are charged via solar power).

A key challenge of these deployments is the need to co-design the IoT device network *with* the power infrastructure which can impose incompatible constraints. The goal of our work is to develop tools that simplify this co-design process for edge deployments consisting of sensors, a communication access point, and base station that executes a deployment planning tool.

Toward this end, we present a model for radio communication between the sensors nodes and base stations in an edge deployment. We use multi-objective numerical optimization to allocate the bandwidth to sensors in a way that minimizes power consumption of the sensors in the deployment. Our approach is unique in that we use it to facilitate comparison of popular optimization approaches as well as to support the co-design of power infrastructure, sensor placement, and data delivery for edge deployments.

We employ two objective functions that we believe are useful in these settings. The first, called `min-max`, minimizes the maximum energy consumption, $max\{E_i^t\}$, in each data transfer cycle $t$. The objective corresponds to deployments in which each sensor node has its own battery and our goal is to minimize the maximum energy draw by any individual node. The second, called `min-sum`, minimizes the total energy consumption, $\sum^i E_i^t$, in each cycle of data transfer $t$. This objective corresponds to deployments that use a shared power source (e.g. a single solar powered battery infrastructure that each node uses as their power source).

## 3.1 System Model and Optimization Problem

The system model consists of a set of $N$ nodes, each of which communicates using wireless radio communication with a single access point. Each node is assigned a data size $D$ that it must transmit to the access point, and a distance $d$ that gives its physical distance from the access point. All nodes share a common duty cycle during which time they must begin and complete the transfer of the data assigned to them. Each node can be configured to use a fraction of a total fixed bandwidth budget available during a duty cycle and to adjust it's signal power according to the assigned fraction.

Inspired by our SmartFarm micro-climate monitoring deployments [14, 15, 18, 29], the execution model is one in which all sensors "wake up" simultaneously (so that readings are time correlated), transmit sensor readings over a radio communication link to the access point during a fixed duty cycle, and then "sleep" to conserve battery power until the start of the next duty cycle. Both the sensor and access point placement are known *a priori*, as is the amount of data that must be communicated in each duty cycle.

This system model gives rise to the following optimization problem. *Given a set of nodes with assigned data transfer requirements and distances from the access point, what is the apportionment of bandwidth that minimizes the signal power each node must expend to transfer its data within the duration of a specified duty cycle?*

## 3.2 Data Transmission Model

To model the wireless communication scenario, we use the path-loss function [13] shown in Equation 1 to calculate the influence of distance for each sensor $i$.

$$PL_i = 38.77 + 16.7 * \log_{10} d_i + 18.2 * \log_{10} f_i \tag{1}$$

In Equation 1, $d_i$ is the distance (the units are meters) between sensor $i$ and the base station, whereas $f_i$ is the frequency (we set this to 5.9 GHz in this work) of the transmission signal. The units of path loss computed by Equation 1 are decibels (db). Parameters, such as 38.77, 16.7, and 18.2, are from the experiences in [13].

The acceptance signal power, in decibel-milliwatts (dbm), is given by Equation 2 where $P_i$ is the wireless signal power (in the range of 0.1 to 21 dbm) per the wireless standard [13].

$$P_i^{'} = P_i - PL_i. \tag{2}$$

Note that db and dbm are on a log scale making the units of $P_i^{'}$ dbm. Converting $P_i^{'}$ to milliwatts (mw) using Equation 3

$$P_i^{''} = 10^{\frac{P_i^{'}}{10}} \tag{3}$$

allows the computation of the signal-to-noise ratio (SNR) for each node using Equation 4

$$SNR_i = \frac{P_i^{''}}{N_0 * B_i} \tag{4}$$

where $N_0$ is the power of environment noise, and it equals $10^{-11.4}$ mw [19]. Given the Signal-Noise-Ratio ($SNR_i$) and Bandwidth ($B_i$ in Mhz), the transmission speed of sensor $i$, $TS_i$, is computed using the Shannon equation shown in Equation 5.

$$TS_i = B_i * \log_2 (1 + SNR_i) \tag{5}$$

Dividing the data size $D_i$ transmitted by each sensor $i$ by $TS_i$ yields the transmission time for sensor $i$:

$$TT_i = \frac{D_i}{TS_i}. \tag{6}$$

When determining bandwidth allocations, we specify that $TT_i$ must be less than or equal to the deadline imposed by the duty cycle for the deployment for all sensors. Finally, sensor energy consumption $i$ is obtained via:

$$E_i = TT_i * 10^{\frac{P_i}{10}}. \tag{7}$$

It is worth noting that there is a trade-off between $P_i$ and $TT_i$. If $B_i$ is fixed, then when $P_i$ increases $TT_i$ decreases. If $P_i$ is fixed, then the allocation of $B_i$ is non-linear. Thus, in terms of the data transmission model, the problem of determining the values of $B_i$ and $P_i$ that yield a minimized set of $E_i$ values, subject to the constraint that all $TT_i$ are less than or equal to a fixed deadline (imposed by a given duty cycle), is an optimization problem we must solve.

## 3.3 Optimization Solvers

We define two different optimization problems for determining a bandwidth allocation that minimizes power usage. The first, termed `min-max`, is shown in Equation 8. It minimizes the maximum value of $E_i$ for a set of sensors. This objective function corresponds to

a deployment in which each sensor has its own separate power source that can be exhausted (e.g. a battery).

$$F(O) = max\{E_i\}, \tag{8}$$

The second function, termed `min-sum` is given by Equation 9. It uses the sum of $E_i$ as an objective function.

$$F(O) = sum\{E_i\}. \tag{9}$$

This function represents the case where the set of sensors share a common power source that can be exhausted. We investigate three approaches for minimizing $F(O)$: mixed-integer non-linear programming, satisfiability modulo theories (SMT), and a genetic algorithm (GA).

*3.3.1 MIDACO.* MIDACO is a commercial solver for numerical optimization that is based on the Ant Colony algorithm [26], which can be applied to continuous non-linear optimization (NLP) problems, discrete/integer (IP) optimization problems, and mixed integer (MINLP) problems [1]. In our scenario, a deployment is specified as a set of data transmission requirements $D_i$, a set of distances $d_i$ and a deadline for a set of sensors indexed by $i$ which is an integer value in the range 1 to $N$. We then minimize $F(O)$ as specified by Equation 8 or Equation 9.

To do so, we formulate a set of internal constraints that any feasible solution (i.e. one that meets the deadline) must also obey. These internal constraints are shown in Equation 10.

$$\begin{aligned} &G(0) = B_1 + B_2 + ... + B_N - total\ bandwidth \\ &G(1) = deadline - TT_1 \\ &... \\ &G(i) = deadline - TT_i, \end{aligned} \tag{10}$$

where G(0)=0 and other G()≥0. $N$ is the number of sensors.

*3.3.2 SMT and Z3.* Z3, as an SMT solver, includes limited capabilities for determining the satisfiability of constraints that require the evaluation of numerical variables. In particular, the support in Z3 for log functions, exponentiation, and real-valued variables either performs poorly or is non-existent. Z3 only includes optimization capabilities for equational constraints that are linear [4]. Thus, to use Z3 for optimizing energy usage, we must transform the real-valued, non-linear problem formulation described in the previous subsection into an integer-valued, linear problem.

Equation 11 shows a discretized integer representation of $P_i$. We limit the range of signal power values that the access point and sensors can adopt to the range 0.01 to 21 dbm. We then multiply this value by 100 so that it is expressed as an integer with a value between 1 and 2100. Equation 12 captures the total bandwidth budget (shown as an internal constraint in the MIDACO formulation in Equation 10) as an independent and explicit constraint for the SMT formulation.

To capture the inverse relationship between a signal power and bandwidth allocation shown in Equations 1 through 5, we add a constraint that forces the solver to minimize the maximum signal power $P_i$ while, at the same time, maximizing the minimum bandwidth $B_i$ for all sensors, as shown in Equations 13 and 14, respectively. Similarly, for the objective of minimizing the sum of energy usage, we substitute Equations 15 for Equation 13 in the constraint set to compute `min-sum`.

Because Z3 cannot evaluate non-linear, real-valued, log functions, it cannot incorporate a deadline constraint because it cannot solve Equations 1 through 6. However, it is able to find multiple unique solutions for a single constraint problem. We exploit this feature to "search" the constraint space for the best solution (the one with the lowest energy objective function value).

To do so, we run the solver until it finds a solution and then apply Equations 1 through 6 to the resulting values to determine if the solution meets the deadline. We also compute $F(O)$ using Equation 7 for the solution. If the solution meets the deadline, it is saved in a set of feasible solutions. If it does not, but its $F(O)$ value is lower than the $F(O)$ value of a previous infeasible solution, it is saved as the "best" infeasible solution. We then continue the solver and repeat the process for the next solution it finds. The search terminates after either 900 iterations (the maximal case) or a fixed time limit (cf. Section 4). We explore the use of feasible and infeasible solutions in Section 5.

$$1 \le P_i \le 2100,\ 1 \le B_i \le 100 \tag{11}$$

$$\sum_{i=1}^{N} B_i == total\ bandwidth \tag{12}$$

$$\forall_{i=1}^{N} P_{max} \ge P_i,\ minimize(P_{max}) \tag{13}$$

$$\forall_{i=1}^{N} B_{min} \ge B_i,\ maximize(B_{min}) \tag{14}$$

$$minimize(\sum_{i=1}^{N} P_i) \tag{15}$$

This search process can be improved by adding an additional constraint that captures the influence of the data size $d_i$ that each sensor must send and the distance $D_i$ from the access point for each sensor. Specifically, we would like the solver to avoid solutions where nodes sending large amounts of data that are located far from the access point are assigned less bandwidth than nodes closer to the access point with smaller data transfer requirements. There may be many such solutions that satisfy the constraints but result in infeasible (and poor) allocations.

To guide the solver toward better solutions in the space, we add a "weighting" constraint that creates a relationship between the product of the data size and distance and the sum of the signal power and allocated bandwidth. Equation 16 shows this "guidance" constraint. Note that this is the only constraint in the SMT formulation that relates bandwidth allocation $B_i$ to signal power $P_i$ – without it the solver is free to find solutions that assign values to these variables independently. As such, it constitutes a linear multi-objective function for Z3 to optimize. We also use a Z3 feature that prioritizes Pareto-optimal solutions to multi-objective functions in its solution search [4]. Doing so ensures that the relationship in Equation 16 is favored by Z3.

$$\forall_{i=1}^{N} (D_i * d_i \le D_{i+1} * d_{i+1} \implies B_i + P_i \le B_{i+1} + P_{i+1}) \tag{16}$$

This optimization to the constraint search introduces additional complexity, however. Using Pareto priority, Z3 can return repeated solutions when it cycles through all of the Pareto points the solver can find. When the solver returns repeated solutions, we introduce

a "soft" constraint that allows it to consider additional possible solutions. When the solver stops and offers a repeated solution, we compute a *target* $P_i$ for each node based on the $d_i, D_i$ values for the node from the deployment and the $B_i$ in the current (and repeated) solutions. We then add the soft constraint shown in Equation 17 if *target* $P_i$ satisfies the constraint shown in Equation 11. The soft constraint forces Z3 internally to search for new solutions using Pareto priority but in the course of that search, if it can find a lower value for the power as an optimal point, it can select it even if it violates the soft constraint. Thus the soft constraint allows Z3 to make an "up-hill' climb in the search space in its search for Pareto optimal solutions.

$$\begin{aligned} &\#target\ P_i\ computed\ using\ Equations\ 1\ through\ 6 \\ &\#from\ the\ values\ of\ B_i,\ D_i, d_i \\ &(add\ soft)\ P_i \geq target\ P_i \end{aligned} \tag{17}$$

*3.3.3 Genetic Algorithm.* We also developed a custom Genetic Algorithm (GA) for comparison in this study given it is a widely used heuristic-based optimization approach. In our GA formulation, bandwidth $\{B_i\}$ and signal power $\{P_i\}$ are represented as separate chromosomes. The GA first creates a group of random chromosomes as parents and crosses each *father* in the group with a random *mother*. A cross is performed by selecting a random position for each chromosome and concatenating the mother with the father using this position. For example, a child bandwidth chromosome using random position $k$ consists of father bandwidth items $B_1$–$B_k$ followed by mother bandwidth items $B_{k+1}$–$B_n$ for list length $n$. After the crossing, the sum of child bandwidth may differ from the total bandwidth. We update the items to conform to this constraint. Specifically, we compute $\Delta = sum\{child\_B_i\} - total\ bandwidth$. If $\Delta > 0$, we select a random item in the $child\_B_i$ and subtract 1 from it (if $\Delta < 0$, we add 1). We repeat this process $\Delta - 1$ times.

After crossing, we perform mutation to introduce chromosome diversity. We choose a random number $\mu \in (0, 1)$ and use it to mutate the child items. For bandwidth, if $0.5 < \mu < 0.85$, we do not mutate. If $\mu \geq 0.85$, we left-rotate the items one position. If $\mu \leq 0.5$, we subtract 1 from the maximum item and add 1 to the minimum item in $child\_P_i$. For power, we use a different range for $\lambda \in (0, 1)$. For each item in $child\_P_i$, if $0 < \lambda < 0.15$, there is no mutation. When $0.15 \leq \lambda \leq 0.6$, we add add or subtract 1 to/from $child\_P_i$ using a 50-50 probability and we only update the value if doing so results in a value that is in the range (0.1 dbm, 21 dbm). Similarly, we mutate each item ±0.1 and ±0.01, when $\lambda$ is in the range (0.6, 0.85) and (0.85, 1), respectively.

# 4 SOLVER EVALUATION

The overall goal of this investigation is to develop a tool for aiding deployment design in remote IoT settings (cf. Section 5). Doing so requires the use of a solver that is capable of optimizing the bandwidth allocation among sensors so that the energy they consume is minimized. In this section, we evaluate three different solvers: MIDACO, Z3 (SMT), and a genetic algorithm (GA) written specifically for this project. In evaluating the solvers, we wish to determine which solver provides the highest quality solution (i.e. the bandwidth allocation that minimizes energy consumption) and also to investigate the relationship between solution quality and the computational effort (measured in CPU time required by a solver to find the solution).

## 4.1 Solver Evaluation Experimental Setup

All of the experiments were run using an AMD Ryzen 7 5800H @3.20 GHz CPU with 16GB RAM and 4 cores. For the mixed-integer programming solver, we use MIDACO Version 6.0 under a commercial license, for SMT, we use Z3 Version 4.12.2, which is available as open source from [34], and we developed the genetic algorithm as a bespoke approach specifically for this paper.

To explore solver efficacy, we generate a set of 100 randomized deployments and compare the solvers using this set as an input. Each deployment in the set consists of 10 sensor nodes, where a node's data transmission requirement is drawn from a uniform distribution on the interval $(50, 150)$ and its distance from the access point is drawn from a uniform distribution on the interval $(50, 200)$. For a specific individual deployment, we measure solution quality as the energy resulting either from minimizing the `min-max` or the `min-sum` objective function.

In all experiments, the units of data transmission size are megabits, the distance units are meters, and we measure energy consumption in millijoules (mJ). When reporting aggregate solution quality statistics (e.g. minimum, maximum, mean), we set a fixed duty cycle (the units are seconds) for the 10 nodes and aggregate over the 100 randomized deployments.

Finally, to capture the "effort" each solver requires to create a solution, we define the "computational effort budget" to be the maximum wall-clock time the solver is allowed to use to compute a solution. A solver may use less than this budget, but not more. We define an "infinite" computational effort budget to be 5 hours.

Note that it is possible to generate a deployment for which there is no allocation of bandwidth that will allow all 10 nodes to finish transmission within the specified duty cycle. It is also possible that a solution exists, but that a solver is not able to "find" it within its computation effort budget. Finally, it is possible that multiple solutions exist, in which case, we expect that the solver will return the "best" one (i.e. the one that minimizes the objective function). We term a solution that allows all 10 nodes to complete transmission before the end of the duty cycle a "feasible" solution. We select the 100 deployment evaluation set such that all 100 deployments have feasible solutions given a 2 second duty cycle (however, a particular solver may not be able to generate a feasible solution for all 100 deployments).

## 4.2 Unlimited Solver Computational Effort

Figure 1 shows the efficacy of each solver for the 100-deployment test set with respect to the `min-max` objective function (cf. Equation 8): a practically unlimited computational effort budget, and a 3-second duty cycle. In the figure, the best (lowest) maximum energy, mean energy, and minimum energy were generated by the MIDACO solver. The SMT solver generated solutions that were somewhat less energy efficient, and the GA solutions were less efficient than the SMT solutions. This comparison illustrates the effectiveness of using a high-quality, commercial solver over perhaps a more intuitive constraint-based approach (SMT) or a solution that is

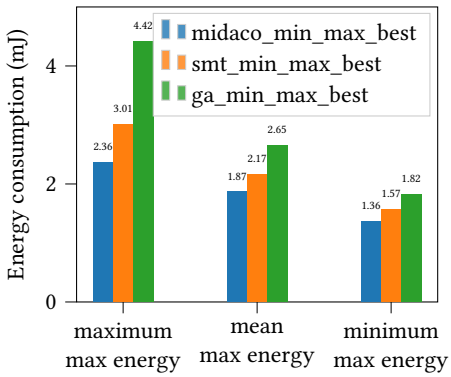


**Figure 1: The figure compares the solvers for the `min-max` objective function and an unlimited computational effort budget. The y-axis is maximum energy consumption over the 10 nodes (in mJ) and x-axis represents the maximum, mean and minimum of over 100 randomized experiments. The duty cycle is 3 seconds.**

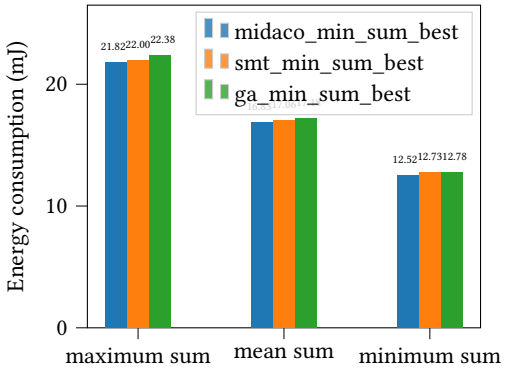


**Figure 2: The figure compares the solvers for the `min-sum` objective function and an unlimited computational effort budget. The y-axis is the maximum energy consumption over the 10 nodes (in mJ) and x-axis represents the maximum, mean and minimum of over 100 randomized experiments. The duty cycle is 3 seconds.**

easy to conceive, develop, and debug (GA), given an unconstrained computational effort budget.

Figure 2 shows the same comparison as depicted in Figure 1, but for the `min-sum` objective function (cf. Equation 9). These results indicate that with "infinite" computational effort, each optimization technique generates solutions that are approximately equivalent in terms energy efficiency in each deployment. That is, it is "easier" to optimize the total energy consumption (i.e `min-sum`) for the collection of sensors compared to optimizing the energy consumption for individual sensors (i.e. `min-max`). As such, this result argues for deployment architectures in which the power infrastructure is shared rather than implemented separately for each sensor.

Note that in our experiments, GA substantially underperforms MIDACO and SMT for the `min-max` objective function (and all three are approximately equivalent for the `min-sum` objective function).

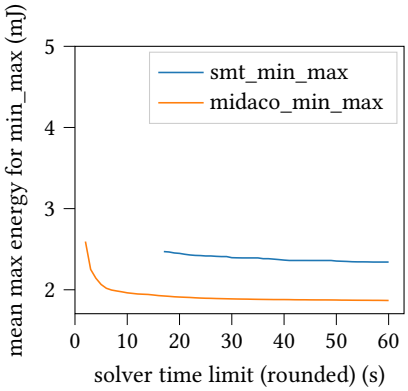


**Figure 3: The figure compares the solvers for the *min-max* objective function when each solver is given the same computational effort budget. The y-axis is the average maximum energy consumption over the 10 nodes (in mJ) and x-axis represents the computational effort budget used to generate the corresponding y-axis value. The duty cycle is 3 seconds.**

Thus, for brevity, we focus in comparing MIDACO to SMT using `min-max` for the remainder of this investigation.

## 4.3 Limited Solver Computational Effort

We next compare the solvers in terms of their computational efficiency relative to the energy efficiencies of the solutions they produce. In these experiments, we limit the computational effort budget given to each solver and look at the quality of the solutions that are generated within the budget.

Figure 3 plots the average maximum energy consumption of the 10 nodes across the 100 deployments in the test set on the y-axis, as a function of the computational effort budget shown on the x-axis. Note that SMT is not able to find at least one feasible solution for all 100 test deployments with a computational effort budget of less than 17 seconds (the graph only depicts averages calculated over 100 test deployments). In contrast, MIDACO is able to find *some* feasible solution for all 100 deployments using a computational effort budget of as little as 1 second. Moreover, the average quality of the solutions (the average energy consumption) improves rapidly with additional effort budget until approximately 6 seconds, after which the improvement per additional second of solver computation increases far more slowly until it is near zero.

Overall, the figure illustrates the average "efficiency" of the MIDACO and SMT solvers in terms of solution quality. That is, as each solver is given more computational effort budget the quality of the solution improves (i.e. the mean energy consumption is lower). Moreover, for the same computational effort budget of 17 seconds or greater, the mean solution quality generated by MIDACO is higher (i.e. the mean maximum energy consumption is lower) compared to SMT. Thus in terms of average expected performance against a fixed computational effort budget, MIDACO is more performant than SMT.

However, while SMT does not a produces a feasible solution for all 100 test deployments when the effort budget is less than 17 seconds, it does find feasible solutions for a subset of the 100 test

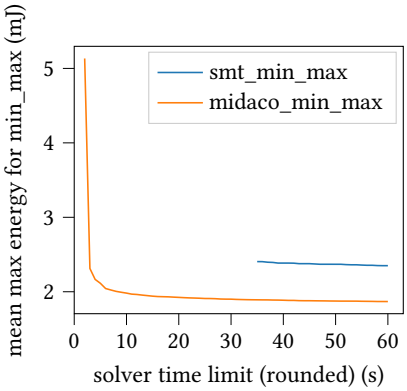


**Figure 4: The figure compares the solvers for the *min-max* objective function when each solver is allowed the same computational effort. The y-axis is the average maximum energy consumption over the 10 nodes (in *mJ*) and x-axis represents the computational effort budget used to generate the corresponding y-axis value. The duty cycle is 2.8 seconds.**

deployments. Table 1 shows a sample of 5 deployments from the deployment set. Column 1 shows the deployment index, column 2 shows the maximum energy consumption (in mJ) over the 10 nodes in the deployment and the time (in seconds) taken by MIDACO (in parentheses) which was limited to 4 seconds. Column 3 shows the energy consumption (in mJ) and the time (in seconds) taken by SMT to find its *first* feasible solution for that particular experiment. In column 4, we show the "chase" time – the additional time (in seconds) MIDACO required to find a solution equivalent or better than the first solution found by SMT shown in column 3. For example, in *Exp #11*, MIDACO achieved a solution in which the maximum energy consumption was 6.02 *mJ* in 4 seconds. SMT found its first feasible solution, corresponding to a maximum energy consumption of 2.80 *mJ* in 6 seconds. MIDACO required an additional 12*seconds* to achieve a solution in which the maximum energy consumption was less than or equal to the 2.80 *mJ* achieved by SMT.

Thus, when SMT finds a solution with little computational effort budget, it is can be considerably better than the solution MIDACO finds with the same computational effort budget. These results suggest that we can use SMT and MIDACO in concert when the computational effort is small. Specifically, we devote one half of the computational effort budget to SMT and the other to MIDACO. If SMT returns a solution, we compare it to the solution generated by MIDACO and use the solution that results in less energy consumption. Otherwise, we use the MIDACO solution.

Finally, the "complexity" of the optimization problem each solver must solve is roughly proportional to the deadline imposed by the duty cycle. As duty cycle decreases, feasible solutions become more "difficult" to find. Figure 3 shows results for a duty cycle of 3 seconds and Figure 4 shows results for a duty cycle of 2.8 seconds.

Comparing Figures 3 and 4 which show the mean solution quality (in *mJ*), SMT is unable to find 100 feasible solutions in less than 35 seconds of solver time with a duty cycle of 2.8 seconds compared to

**Table 1: Solution energy (mJ), solver duration (seconds), and chase time for 5 specific randomized deployment solutions.**

| Exp # | MIDACO solution mJ (sec) | SMT solution mJ (sec) | chase sec |
|---|---|---|---|
| 11 | 6.02 (4) | 2.80 (6) | 12 |
| 51 | 2.04 (4) | 1.99 (1) | 11 |
| 52 | 3.21 (4) | 2.40 (2) | 14 |
| 61 | 2.66 (4) | 2.21 (12) | 11 |
| 76 | 2.55 (4) | 2.25 (<1) | 1 |

17 seconds of solver time when the duty cycle is 3 seconds. When the solver time is sufficient in both cases, the average quality of the corresponding solutions is approximately the same. Thus, the length of the duty cycle affects the ability of SMT to find a feasible solution with a fixed computational effort budget and shorter duty cycles require larger budgets.

### 4.4 Discussion of Solver Comparison

Given the formulation of the optimization problem described in Section 3, it is perhaps unsurprising that MIDACO is almost always more effective than SMT. For minimizing maximum energy consumption, the problem is a non-linear mixed integer programming optimization problem where only one quantity (communication bandwidth) is integer-valued. MIDACO is a high-quality commercial solution designed specifically for such problems.

What is surprising is that SMT can find a "good" solution with limited computational effort for *some* deployments and that solution may be considerably better than the one generated by MIDACO with the same effort budget. From a practical perspective, when the computational effort budget is small, these results militate for "trying" SMT for a given deployment using some fraction of the effort budget to possibly generate a "good" feasible solution. And then giving the remainder of the budget to MIDACO (which is sure to generate a feasible solution) to exploit the strengths of each solver.

## 5 SOLVER-AIDED DEPLOYMENT DESIGN

The results discussed in Section 4 for a randomized test set indicate that the MIDACO solver is generally more performant (in terms of solution quality) compared to SMT or GA for the `min-max` objective function. In this section, we outline our use of a solver to aid in deployment planning.

Our overall approach is for the deployment engineer or maintenance staff member to generate a list of data transfer requirements (one per sensor node), a set of distances between each node and the access point, and a duty cycle. Solving the `min-max` or `min-sum` problem yields a list of signal power requirements and bandwidth allocated to each sensor for each duty cycle.

From this information, the battery "drain" (in units of power) required by each node in the deployment can be computed by multiplying the energy required by the length of the duty cycle and computing how many duty cycles take place during the worst case hours of darkness. Using the charging rate and the worst-case hours of available solar, it is possible to size the solar capacity to match

or slightly exceed the battery drain such that both are minimal. At present, deployment engineers typically overprovision the power infrastructure in these settings using the manufacturer-reported rates and maximal energy use by all devices. Using a solver-based approach, the expense and infrastructure footprint, compared to a maximal approach, can be reduced.

For example, consider the deployment transfer requirements and distances shown in Equation 19. For a 3-second duty cycle, the MIDACO solver generates the energy consumption estimates shown in Equation 20 for the 10 sensors using the `min-max` objective function. Converting $mJ$ to Watts ($W$) for a 3 second duration (the length of the duty cycle) yields the value in Equation 21 for each duty cycle. In California (at one of our test site locations), the longest nighttime duration in 2024 will be 14.12 hours (on December 23). The battery drain for each sensor due to communication, then, is the product of the values shown in Equation 21 and the number of 3-second duty cycles that will take place during the hours of darkness. As an example, if the sensors are "awake" for a 3-second duty cycle every minute, the batteries must sustain 848 duty 3-second cycles in 14.12 hours. Multiplying the values in Equation 21 by 848 would yield the battery drain, in Watt-seconds, that the battery must sustain and recharge in every diurnal cycle due to communication.

## 5.1 Making Infeasible Solutions Feasible

It is also possible to adapt our methodology to cases where the solver cannot find a feasible solution. As discussed in Section 4, longer duty cycles (at the possible expense of additional battery capacity) improves optimization performance. However if the deployment functionality depends on a specific duty cycle for which the solver yields no feasible solution, it still may be possible to employ our approach. In particular, if the sensor nodes can be moved closer to the access point or the amount of data they are required to transfer per duty cycle can be reduced, it is possible to take the "best" infeasible solution and adjust either the distances or the transfer sizes to make that solution feasible.

For example, consider again the deployment configuration shown in Equation 19, the `min-max` objective function, and a duty cycle of 2 seconds. The MIDACO solver does not generate a feasible solution for this deployment using an unlimited computational effort budget (and SMT does not either). However, it does allow us to rank the infeasible solutions in terms of the objective function and the "best" of the infeasible solutions is shown in Equation 18. This solution is infeasible because the first and fourth sensors in the list cannot complete their communication transfers within the 2-second duty cycle. That is, in Equation 18, the first and fourth sensor completes their data transfer in 2.53 and 2.08 seconds, which is longer than the 2-second duty cycle.

$$\begin{aligned} energy(mJ) &= [2.58, 1.67, 1.33, 2.13, 1.55, \\ &\quad 1.6, 1.17, 1.99, 1.55, 1.34] \\ transfer\_time(s) &= [2.53, 1.63, 1.30, 2.08, 1.51, \\ &\quad 1.56, 1.14, 1.94, 1.51, 1.31] \end{aligned} \tag{18}$$

We initially focus on changing the size of the data, to satisfy the duty cycle constraint. In Equation 6, we keep the value of deadline as 2 seconds, the data size as an unknown variable to solve for, and the transmission speed is the transmission speed we derive from Equations 1 through 5. Here, we use the values of signal power and bandwidth from the "best" infeasible solution and keep the same distance value from Equation 19 for sensor 1 and sensor 4. After deriving the transmission speeds for each sensor, we solve for the data sizes that can satisfy the duty cycle constraint, using the other values from "best" infeasible solution generated by MIDACO. This means the worst case energy consumption is still minimized to the extent it could be by the "best" infeasible solution MIDACO generated. Using this adjustment generates data size values shown in Equation 22. Note that a comparison of Equation 22 to the data list in Equation 19 for sensors 1 and 4 indicates that reducing the data transfer of sensor 1 from 106 megabits to 86.76 megabits, and sensor 4 from 133 megabits to 127.33 megabits creates a feasible solution for a duty cycle of 2s.

Alternatively, if the data size cannot be changed, but the sensor can be moved, it is possible to perform a similar adjustment for distance using Equation 1. This process of derivation is a bit more involved as we start backwards from Equation 6 to calculate the target transmission speed that would meet the duty cycle constraint of 2 seconds, i.e the transmission time would be 2 seconds and the data size would be the same as the input (from Equation 19) for sensors 1 and 4.

Consider sensor 1 as an example. From its target transmission speed, we can derive the value of the target SNR using Equation 5. Here we keep the bandwidth value $B_1$ as the value from the "best" infeasible solution generated from by solver. Using this value, we derive the target $P_1^{''}$ from Equation 4 and then calculate the target $P_1^{'}$ from Equation 3. Finally we calculate the path loss ($PL_1$) for sensor 1 from Equation 2, keeping target $P_1^{'}$ from the previous equation and the signal power $P_1$ as the value received from the "best" infeasible solver solution. We substitute the value for path loss in Equation 1 and solve the equation for $d_1$, the distance value for sensor 1. The same steps apply for sensor 4 as well.

The modified distance values are shown in Equation 23. As with adjusting the data requirements, we can compare Equation 23 to the distance list in Equation 19 for sensors 1 and 4. We find that moving sensor 1 from 156 meters from the access point to 86.76 meters, and sensor 4 from 67 meters to 58.74 meters, creates a feasible solution for a duty cycle of 2 seconds.

While in this example, only two values exceed the deadline imposed by the duty cycle, there can be situations where multiple values can exceed the deadline (up to all of the sensors may need data or distance modifications to reach the deadline). The methodology works for either scenario as the data size transfer, distance are independent across sensors (i.e. changing a value in one does not impact the value of another).

$$\begin{aligned} data(MB) &= [106, 91, 136, 133, 121, 96, 113, 85, 78, 52] \\ distance(m) &= [156, 97, 29, 67, 55, 75, 35, 199, 184, 178] \end{aligned} \tag{19}$$

$$\begin{aligned} energy(mJ) &= [1.74, 1.67, 1.75, 1.70, 1.67, \\ &\quad 1.60, 1.64, 1.67, 1.75, 1.75] \end{aligned} \tag{20}$$

$$power(W) = [0.58, 0.56, 0.58, 0.57, 0.56, 0.53, 0.55, 0.56, 0.58, 0.58] \quad (21)$$

$$new\_data(MB) = [83.77, 91, 136, 133, 121, 96, 113, 85, 78, 52] \quad (22)$$

$$new\_dist(m) = [86.76, 97, 29, 58.74, 55, 75, 35, 199, 184, 178] \quad (23)$$

Thus, for those applications where the duty cycle cannot be extended and the solver cannot find a feasible solution, the deployment engineer can choose to move sensors or to recode sensors to transfer less data per duty cycle (or to move some and recode others) in the "best" infeasible solution to create a feasible one. If the engineer's goal is to make the smallest possible change to the deployment that is infeasible, understanding whether changing data transfer or distance requires a larger perturbation.

These results illustrate how solver-aided deployment design can be used to "co-design" a deployment. In the cases where a deployment results in a feasible solution, the power infrastructure capacity that is necessary can be directly computed. When no feasible solutions can be found, the approach allows a deployment engineer to understand what duty cycle (and resulting power infrastructure capacity) is feasible (by experimentally extending the duty cycle). When an acceptable duty cycle cannot be found, the engineer can use the method to determine a feasible solution from the best infeasible solution. Thus, when there is a choice, it is possible to determine whether changing the data payload for one or more sensors or moving one or more sensors (or some combination) will perturb the deployment less and fit the goals of the deployment best. Engineers can then make informed decisions about how to change a deployment to meet the objectives.

## 6 CONCLUSIONS

In this paper, we study the problem of optimizing edge deployments using wireless network allocation between sensors and a wireless access point, such that the energy required to facilitate communication is minimized. We define a non-linear optimization problem that minimizes radio energy used by a set of sensors based on the amount of data transmitted by each sensor to a wireless access point and its distance from the access point. Additionally, we investigate three methods for solving this optimization problem and detail how the problem must be formulated to fit the requirements of each.

Our results show that a commercial mixed-integer non-linear programming solver (MIDACO) is generally more effective than both an SMT solver (Z3 in this study) and a bespoke genetic algorithm. However, for a specific deployment, SMT can produce the best solution with the least amount of computational budget even though it is not possible to predict *a priori* whether it will produce any solution at all. We also find that for randomized deployments, infeasible solutions are often "close" to feasible ones. Thus, in a practical system, when a solver is unable to generate a feasible solution, it is possible to modify the best infeasible solution to produce a feasible one using only small perturbations.

Finally, this work illustrates the process of co-design that is necessary for remote IoT in which the sensing and actuation functionality, the networking functionality, the electrical power infrastructure, and spatial infrastructure placement must all be designed together. It also outlines the practical considerations associated with building tooling to facilitate this co-design process for edge deployments.

This work has been supported in part by NSF award CNS-2107101 and by Norwegian Research Council DILUTE award 262854/F20 and AirQMan award 322473.

## REFERENCES

[1] 2023. Mixed Integer Distributed Ant Colony Optimization. http://www.midaco-solver.com/ Accessed Sep 2023.
[2] Cosmin Avasalcai, Christos Tsigkanos, and Schahram Dustdar. 2019. Decentralized Resource Auctioning for Latency-Sensitive Edge Computing. In *2019 IEEE International Conference on Edge Computing (EDGE)*. 72–76.
[3] Cosmin Avasalcai, Christos Tsigkanos, and Schahram Dustdar. 2021. Adaptive Management of Volatile Edge Systems at Runtime With Satisfiability. *ACM Trans. Internet Technol.* 22, 1 (2021).
[4] Nikolaj BjØrner, Anh-Dung Phan, and Lars Fleckenstein. 2015. VZ - An Optimizing SMT Solver. In *Proceedings of the 21st International Conference on Tools and Algorithms for the Construction and Analysis of Systems - Volume 9035*. 194–199.
[5] Lianjie Cao, Anu Merican, Diman Zad Tootaghaj, Faraz Ahmed, Puneet Sharma, and Vinay Saxena. 2021. ECaaS: A Management Framework of Edge Container as a Service for Business Workload. In *Proceedings of the 4th International Workshop on Edge Systems, Analytics and Networking* (Online, United Kingdom) *(EdgeSys '21)*. Association for Computing Machinery, New York, NY, USA, 73–78.
[6] IBM ILOG Cplex. 2009. V12. 1: User's Manual for CPLEX. *International Business Machines Corporation* 46, 53 (2009), 157.
[7] Leonardo de Moura and Nikolaj Bjørner. 2008. Z3: An Efficient SMT Solver. In *Tools and Algorithms for the Construction and Analysis of Systems*. 337–340.
[8] Rong Du, Ming Xiao, and Carlo Fischione. 2019. Optimal Node Deployment and Energy Provision for Wirelessly Powered Sensor Networks. *IEEE Journal on Selected Areas in Communications* 37, 2 (2019), 407–423. https://doi.org/10.1109/JSAC.2018.2872380
[9] Michelle Effros, Andrea Goldsmith, and Muriel Médard. 2010. The rise of instant wireless networks. *Scientific American* 302, 4 (2010), 72–77.
[10] A. Rosales Elias, N. Golubovic, C. Krintz, and R. Wolski. 2017. Where's the Bear? – Automating Wildlife Image Processing Using IoT and Edge Cloud Systems. In *Internet-of-Things Design and Implementation (IoTDI), 2017 IEEE/ACM Second International Conference on*. IEEE, 247–258.
[11] Julien Gascon-Samson, Mohammad Rafiuzzaman, and Karthik Pattabiraman. 2017. ThingsJS: Towards a Flexible and Self-Adaptable Middleware for Dynamic and Heterogeneous IoT Environments. In *Proceedings of the 4th Workshop on Middleware and Applications for the Internet of Things*. Association for Computing Machinery, New York, NY, USA, 11–16. https://doi.org/10.1145/3152141.3152391
[12] Pradipta Ghosh, Jonathan Bunton, Dimitrios Pylorof, Marcos A. M. Vieira, Kevin Chan, Ramesh Govindan, Gaurav S. Sukhatme, Paulo Tabuada, and Gunjan Verma. 2023. Synthesis of Large-Scale Instant IoT Networks. *IEEE Transactions on Mobile Computing* 22, 3 (2023), 1810–1824.
[13] Marco Giordani, Takayuki Shimizu, Andrea Zanella, Takamasa Higuchi, Onur Altintas, and Michele Zorzi. 2019. Path loss models for V2V mmWave communication: Performance evaluation and open challenges. In *2019 IEEE 2nd Connected and Automated Vehicles Symposium (CAVS)*. IEEE, 1–5.
[14] N. Golubovic, C. Krintz, R. Wolski, B. Sethuramasamyraja, and B. Liu. 2018. A Scalable System for Executing and Scoring K-Means Clustering Techniques and Its Impact on Applications in Agriculture. *International Journal of Big Data Intelligence* (July 2018).
[15] N. Golubovic, R. Wolski, C. Krintz, and M. Mock. 2019. Improving the Accuracy of Outdoor Temperature Prediction by IoT Devices. In *IEEE Conference on IoT*.
[16] Najmul Hassan, Saira Andleeb Gillani, Ejaz Ahmed, Ibrar Yaqoob, and Muhammad Ali Imran. 2018. The Role of Edge Computing in Internet of Things. *IEEE Communications Magazine* 56 (2018), 110–115.
[17] H. Jawad, R. Nordin, S. Gharghan, A. Jawad, and M. Ismail. 2017. Energy-Efficient Wireless Sensor Networks for Precision Agriculture: A Review. *Sensors* 17, 1781 (2017).
[18] C. Krintz, R. Wolski, N. Golubovic, B. Lampel, V. Kulkarni, B. Sethuramasamyraja, B. Roberts, and B. Liu. 2016. SmartFarm: Improving Agriculture Sustainability Using Modern Information Technology. In *KDD Workshop on Data Science for Food, Energy, and Water*.
[19] Yue Li, Mohammad Ghasemiahmadi, and Lin Cai. 2016. Uplink cooperative transmission for machine-type communication traffic in cellular system. In *2016 IEEE 84th Vehicular Technology Conference (VTC-Fall)*. IEEE, 1–5.
[20] Chang Liu, Jin Wang, Liang Zhou, and Amin Rezaeipanah. 2022. Solving the multi-objective problem of IoT service placement in fog computing using cuckoo search algorithm. *Neural Processing Letters* 54, 3 (2022), 1823–1854.

[21] Jude Okwuibe, Juuso Haavisto, Ivana Kovacevic, Erkki Harjula, Ijaz Ahmad, Johirul Islam, and Mika Ylianttila. 2021. SDN-Enabled Resource Orchestration for Industrial IoT in Collaborative Edge-Cloud Networks. *IEEE Access* 9 (2021).

[22] G. Pottie and W. Kaiser. 2000. Wireless Integrated Network Sensors. *Commun. ACM* 43, 5 (2000).

[23] A Prasanth and S Jayachitra. 2020. A novel multi-objective optimization strategy for enhancing quality of service in IoT-enabled WSN applications. *Peer-to-Peer Networking and Applications* 13 (2020), 1905–1920.

[24] T. Rault, A. Bouabdallah, and Y. Challal. 2014. Energy efficiency in wireless sensor networks: A top-down survey. *Computer Networks* 67 (2014).

[25] Sabino Francesco Roselli, Kristofer Bengtsson, and Knut Åkesson. 2019. SMT Solvers for Flexible Job-Shop Scheduling Problems: A Computational Analysis. In *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*.

[26] M. Schlueter, J. Egea, and J. Banga. 2009. Extended Ant Colony Optimization for non-convex Mixed Integer Nonlinear Programming. *Computers and Operations Research* 36, 7 (2009).

[27] Binbin Shi, Wei Wei, Yihuai Wang, and Wanneng Shu. 2016. A Novel Energy Efficient Topology Control Scheme Based on a Coverage-Preserving and Sleep Scheduling Model for Sensor Networks. *Sensors* 16, 10 (2016).

[28] Hui Song, Rustem Dautov, Nicolas Ferry, Arnor Solberg, and Franck Fleurey. 2020. Model-Based Fleet Deployment of Edge Computing Applications. In *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS '20)*. Association for Computing Machinery, New York, NY, USA, 132–142.

[29] UCSB SmartFarm 2024. UCSB SmartFarm. https://sites.cs.ucsb.edu/~ckrintz/projects/. [Online; accessed 17-April-2023].

[30] Can Wang, Sheng Zhang, Yu Chen, Zhuzhong Qian, Jie Wu, and Mingjun Xiao. 2020. Joint configuration adaptation and bandwidth allocation for edge-based real-time video analytics. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 257–266.

[31] Jun-Bo Wang, Jinyuexue Zhang, Changfeng Ding, Hua Zhang, Min Lin, and Jiangzhou Wang. 2020. Joint optimization of transmission bandwidth allocation and data compression for mobile-edge computing systems. *IEEE Communications Letters* 24, 10 (2020), 2245–2249.

[32] Wenting Wei, Ruying Yang, Huaxi Gu, Weike Zhao, Chen Chen, and Shaohua Wan. 2021. Multi-objective optimization for resource allocation in vehicular cloud computing networks. *IEEE Transactions on Intelligent Transportation Systems* 23, 12 (2021), 25536–25545.

[33] Yu Yu, Jie Tang, Jiayi Huang, Xiuyin Zhang, Daniel Ka Chun So, and Kai-Kit Wong. 2021. Multi-objective optimization for UAV-assisted wireless powered IoT networks based on extended DDPG algorithm. *IEEE Transactions on Communications* 69, 9 (2021), 6361–6374.

[34] Z3 Prover [n. d.]. Z3 Prover. https://github.com/Z3Prover/z3. [Online; accessed 12-September-2023].

[35] Qunsong Zeng, Yuqing Du, Kaibin Huang, and Kin K Leung. 2020. Energy-efficient radio resource allocation for federated edge learning. In *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 1–6.

[36] Sheng Zhang, Can Wang, Yibo Jin, Jie Wu, Zhuzhong Qian, Mingjun Xiao, and Sanglu Lu. 2021. Adaptive configuration selection and bandwidth allocation for edge-based video analytics. *IEEE/ACM Transactions on Networking* 30, 1 (2021), 285–298.