Empirical Dataset Generation for AI-Optimized IoT Infrastructure Placement

Fayad Taleb, Georgios Bouloukakis, Khouloud Samrouth, Houssam Hajj

Empirical Dataset Generation for AI-Optimized IoT Infrastructure Placement

Fayad Taleb
Faculty of Engineering
Lebanese University
Lebanon
fayad.taleb@st.ul.edu.lb

Georgios Bouloukakis
Télécom SudParis
Institut Polytechnique de Paris
Paris, France
georgios.bouloukakis@telecom-sudparis.eu

Khouloud Samrouth, IEEE Member
Faculty of Computer Studies
Arab Open University
Beirut, Lebanon
ksamrouth@aou.edu.lb

Houssam Hajj Hassan
Télécom SudParis
Institut Polytechnique de Paris
Paris, France
houssam.hajj_hassan@telecom-sudparis.eu

***Abstract*—The strategic placement of nodes in Wireless IoT Networks (WIoTs) is crucial for ensuring optimal coverage, connectivity, and energy efficiency. Traditionally, node placement has relied on heuristic and manual methods, which often result in inefficiencies and suboptimal network performance. In this paper, we focus on optimizing the coverage performance of WIoTs, which play a pivotal role in environmental monitoring and event detection. In particular, we first develop a tool that allows IoT designers to simulate and generate datasets for multiple sensor deployment options. Then, we empirically generate a dataset that can contribute to the growing field of optimized sensor placement strategies by bridging algorithmic simulations with predictive modeling. Finally, we use the generated dataset to train a decision tree model for sensor node placement predictions. The prototype implementation of our tool and the generated datasets are publicly available for exploitation from the research community.**

***Index Terms*—IoT, smart environments, AI, Node placement**

I. Introduction

The Internet of Things (IoT) has revolutionized urban spaces by transforming them into intelligent environments [1]. IoT enables the integration of various devices and sensors, allowing for real-time data collection and analysis, which enhances the efficiency and quality of urban living. In smart cities, IoT applications are prevalent in areas such as environmental monitoring, where sensors track air and water quality; traffic management, which uses data to optimize traffic flow and reduce congestion; and public safety, where surveillance systems and emergency response mechanisms are improved through connected devices [2].

Wireless IoT Networks (WIoTs) are a crucial component of IoT systems, consisting of numerous small, autonomous devices known as nodes. These nodes, equipped with sensors and actuators, communicate wirelessly to form a logical network. Each node gathers data from its environment and transmits it hop-by-hop to a central management node, typically referred to as a sink or base station. WIoTs are deployed in diverse environments, both indoor and outdoor, to provide critical monitoring and sensing services in areas such as industrial automation, environmental monitoring, and public safety.

The efficiency and effectiveness of WIoTs largely depend on the strategic placement of nodes. Proper node placement ensures optimal coverage, enabling the network to monitor the entire area of interest. It also enhances connectivity, ensuring robust communication links between nodes, and improves energy efficiency by minimizing redundant data transmissions and balancing the energy consumption among nodes. These factors are vital for extending the operational lifespan of the network and maintaining reliable data collection and transmission. In a smart agricultural system, improperly placed soil moisture sensors can lead to uneven irrigation, with some areas being overwatered while others receive too little moisture. This inefficiency not only compromises irrigation performance but also raises operational costs and wastes valuable water resources [3].

The primary challenge lies in the suboptimal placement of IoT infrastructure, which can result in inefficiencies such as inadequate coverage, higher energy consumption, and decreased system reliability [3]–[5]. Traditionally, IoT infrastructure planning and deployment have primarily depended on manual methods, guided by human expertise and intuition [6]. Such approaches, while effective to a certain extent, are often time-consuming and prone to errors. Moreover, traditional manual placement methods lack the capability to manage the complexity and scale of modern smart environment deployments. For instance, in urban smart city applications, manually deploying air quality sensors across a vast cityscape to ensure uniform coverage and minimize redundancy can be an overwhelming challenge [7].

The emergence of Artificial Intelligence (AI) offers a promising opportunity to automate and optimize this process, enabling data-driven and precise placement strategies [8]. In this paper, we create an empirical dataset representing various deployment outcomes to train Machine Learning (ML) models for predicting optimal sensor placements. Our proposed

method enhances placement strategies, making the process faster, more accurate, and adaptable to diverse application requirements. It focuses on maximizing the coverage performance of WIoTs, a critical factor in environmental monitoring and event detection.

The remainder of this paper is organized as follows. Section II presents the two main traditional node placement approaches. Section III analyzes the exiting AI-based node placement methods. Sections IV and IV-B present our proposed method and prototype implementation. Section V presents our experimental results. Finally, we conclude this paper in Section VI.

II. Background

Node placement in WIoTs can be categorized into two primary strategies: static and dynamic placement [6].

- **Static Placement**: Nodes are deployed in predetermined, fixed locations. This strategy is ideal for stable environments where the monitoring requirements and conditions do not change frequently such as in environmental monitoring in forest reserves [9]. In a forest, sensors may be installed at fixed points to monitor parameters like temperature, humidity, and soil moisture. These sensors remain stationary because the monitored variables are relatively stable over time and their geographic distribution is designed to provide representative data across the area. Static placement ensures that the network covers the designated area effectively from the outset, making it simpler to plan and implement. However, it lacks the flexibility to adapt to changing conditions, which can be a limitation in dynamic or unpredictable environments. A good example is the *MAX-AVG-COV* coverage strategy [6] that places sensors to maximize the average coverage across the grid. The algorithm iteratively positions sensors while updating a "miss probability" matrix, reflecting areas that still need improved coverage. The placement continues until each grid cell meets a predefined coverage threshold; or the maximum sensor count is reached.
- **Dynamic Placement**: This strategy involves the adjustment of node positions in response to varying environmental conditions and operational requirements [6]. This strategy is suitable for environments where factors such as coverage area, obstacles, and network topology may change over time. For example, in a large-scale farming mobile sensors can be mounted on automated vehicles or drones to dynamically monitor soil moisture, crop health, or pest activity [10]. Unlike static systems, these sensors adapt to the varying conditions across different regions of the farm, ensuring targeted and efficient resource usage. Dynamic placement enhances the network's ability to maintain optimal performance by relocating nodes as needed to ensure continuous coverage, connectivity, and energy efficiency. While this is more complex to implement, dynamic placement provides a resilient solution capable of adapting to the evolving demands of the monitored environment.

III. Related Works

AI-based node placement in WIoTs is a critical area of research aimed at optimizing network performance, coverage, and connectivity. Various approaches use AI and ML techniques to enhance the deployment strategies of sensor nodes, ensuring efficient resource utilization and prolonged network lifetimes [11].

While Particle Swarm Optimization (PSO) has gained widespread use in ML-related applications [12], many existing methods build upon and extend this algorithm to effectively tackle various optimization challenges. For instance, authors in [13] use the Enhanced-PSO (EPSO) algorithm, customized for various frequency bands, to strategically position sensor nodes. The proposed method aims to prevent clustering, ensuring that each target is monitored by at least one node. Using a probabilistic coverage model based on Euclidean distances, EPSO identifies and addresses coverage gaps (holes) in the initial sensor deployment. To further enhance network coverage, Delaunay triangulation (DT) [14] is incorporated to optimize node positions across the terrain, filling gaps and refining node placement for maximal coverage in WIoTs.

Similarly, authors in [15] combine the PSO with Chaos optimization to enhance coverage optimization. In their method, the positions of all sensor nodes are encoded together as a particle position in the swarm. Initially, PSO was applied to guide the sensors toward their optimal positions. Then, a Variable Domain Chaos Optimization Algorithm (VDCOA) is used to further improve the coverage rate, as well as to increase uniformity in sensor distribution and reduce the average movement distance of the nodes.

Authors in [16] introduce a hybrid version of the "Hitchcock bird-inspired algorithm" (HBIA) metaheuristic algorithm. During the optimization process, they focus on received signal maximization between nodes and antennas. Signal estimations are provided by the machine learning "K Nearest Neighbors" (KNN) algorithm working with real measured data.

IV. Proposed Method

This paper focuses on optimizing the coverage performance of WIoTs, which is essential for environmental monitoring and event detection. Since coverage significantly impacts the ability of WIoTs to reliably monitor designated areas, it must be treated as a primary performance metric in deployment design. Ensuring high coverage levels enables sensors to gather accurate, comprehensive data while optimizing resource usage, which is crucial for real-world applications like environmental monitoring, hazard detection, and security surveillance. In the following subsections, we first provide an overview of our proposed method, followed by a detailed explanation of its implementation.

A. Proposed Method Overview

Figure 1 illustrates our proposed method overview. We begin by implementing a static node placement strategy, the *Max-AVG-COV* (as explained in Section II), as a benchmark for coverage optimization. Then, we run simulations of the

static deployment strategy to generate an empirical dataset representing different deployments.

Following the data simulation, we exploit the generated dataset to train ML models to predict optimal sensor placements. By learning from diverse configurations, this model can generalize to suggest deployment strategies for new environments.

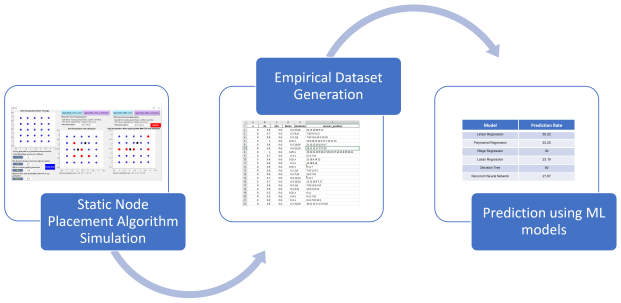


Fig. 1: Proposed Method Overview

B. Prototype Implementation

1) Notations: We leverage the grid-based approach where the area to be covered is represented as a grid with n rows and n columns as illustrated in Figure 2. Each grid point (node) represents a potential sensor location. For clarity, we define the following parameters:

- ***Grid Dimension*** n: defines the grid size in one dimension, resulting in $n * n$ grid points, each labeled sequentially from 1 to n^2
- ***Sensor Quality*** α: this parameter determines the effectiveness or range of individual sensors. A higher α implies sensors with greater coverage or sensitivity, which impacts how densely sensors need to be distributed to meet coverage requirements.
- ***Spacing Between Adjacent Points*** d_0: controls the physical distance between neighboring grid points, which influences the coverage overlap and potential detection sensitivity of sensor nodes.
- ***Maximum Miss Probability*** (M_{min}): defines the acceptable miss probability threshold for any grid point, ensuring that each location on the grid is covered with sufficient reliability. Placement algorithms strive to maintain miss probabilities below this threshold across the grid.
- ***Obstacle Locations***: defines obstacles locations within the grid, which restrict certain cells from being viable sensor locations. These obstacles force placement algorithms to adapt their placement strategy, ensuring that coverage goals are met despite these limitations. We use 'x' to denote the absence of any obstacles. If an obstacle is present, it is specified by indicating the two grid points between which the obstacle is located.
- ***Sensor Positions***: correspond to locations (grid node positions) where sensors should be placed to achieve optimal coverage based on the specified parameters and constraints such as obstacles and miss probability thresholds.

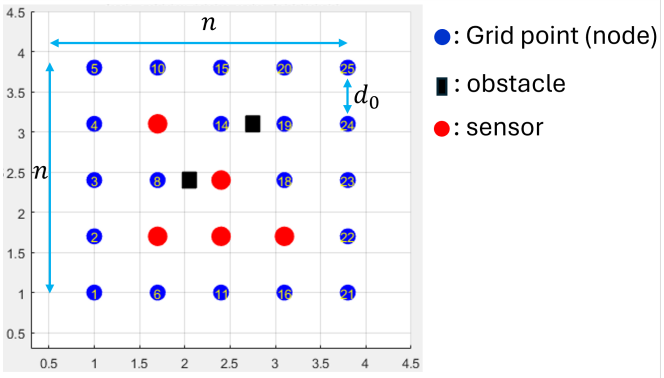


Fig. 2: Grid Visualization with Obstacles: the grid illustrates potential sensor locations, with obstacles disrupting connectivity between nodes. In this example ($n = 5$), obstacles are placed between nodes 8 and 13 and between nodes 14 and 19. Sensors are strategically positioned at nodes 7, 9, 12, 13, and 17 to ensure optimal coverage despite the constraints.

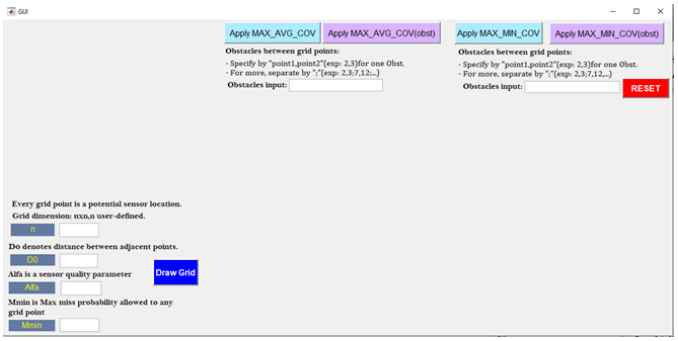
2) MAX-AVG-COV Algorithm Simulation: As a next step, we simulate under various parameter conditions the sensor placement *MAX-AVG-COV* algorithm, designed to maximize the average coverage across the grid. In particular, we calculate the coverage performance by determining the proportion of grid points within the sensing range of the deployed sensors while accounting for obstacles that may block coverage. The algorithm iterates over potential sensor placements, computing the average coverage across the grid by summing the coverage of individual grid points and normalizing it by the total number of points according to Eq. 1, where C_i is a binary value indicating whether the $i-th$ grid point is covered (1 if covered, 0 if not) and n is the grid dimension introduced earlier. A grid point is considered covered if it falls within the sensing range of at least one sensor and is not obstructed by obstacles.

$$\text{Coverage Performance} = \frac{\sum_{i=1}^{n^2} C_i}{n^2} \tag{1}$$

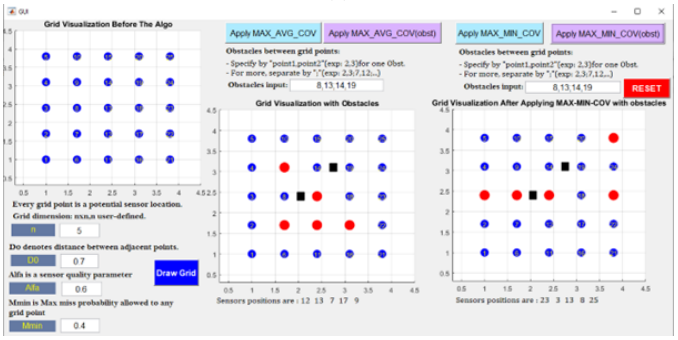
We implement the *Max-Avg-Cov* algorithm using MATLAB with a custom graphical user interface (GUI) to streamline user interaction and simplify experimentation. Our implementation is available on https://github.com/ksamrouth/Empirical-Dataset-Generation-for-AI-Optimized-IoT-Infrastructure-Placement.git.

As illustrated in Figure 3.a, the GUI includes text boxes for entering input features (grid dimension n, the spacing between grid points d0", "maximum miss probability, sensor quality α, and obstacle locations). Additionally, the GUI allows users to draw the environment grid and simulate placement algorithms, making it easy to test various configurations and visualize placement outcomes (See Figure 3.b). This setup enables intuitive experimentation across different input scenarios, enhancing the usability of the simulation platform.

By adjusting the inputs (grid size n, sensor quality α, spacing between grid points d_0, and obstacle layout), we perform multiple simulations to observe the placement behavior across diverse scenarios. This iterative testing provides a broad set of results, allowing for in-depth analysis of how the algorithm adapts to different deployment constraints and objectives.



(a)



(b)

Fig. 3: Graphical User Interface GUI for the WIoT Data Simulation Configuration (a) including text boxes for entering input features, and (b) drawing the environment grid and visualizing WIoT placement outcomes.

3) Empirical Dataset Generation: By running multiple simulations of the static deployment strategy, we then generate an empirical dataset representing different deployment outcomes, including the following variables:

- ***Input Features***:
  - Grid Dimension n
  - Sensor Quality Parameter α
  - Spacing Between Adjacent Points d_0
  - Maximum Miss Probability (M_{min})
  - Obstacle Locations
- ***Output***:
  - Sensor Positions

After every simulation, we provide the results in a tabular format to create a structured dataset. For each simulation, we document the input parameters as feature variables, including grid dimension n, sensor spacing α, and obstacle locations. We then record the outputs which represent the grid points chosen for sensor placement by the *MAX-AVG-COV* algorithm. This systematic documentation ensures that each input-output pair is accurately captured, creating a dataset that reflects the full range of configurations tested.

4) Prediction using ML Models: In this work, the primary objective is to predict the positions of sensors based on various input features, including the presence and location of obstacles. Since the sensors' positions are represented as continuous linear values, we employ regression models to capture the underlying relationships and predict the exact positions. Regression models are well-suited for this type of problem because they allow for predicting continuous outcomes based on input features.

To tackle this problem, we explore a variety of ML models to thoroughly explore their predictive capabilities. In particular: Linear Regression, Polynomial Regression, Ridge Regression, Lasso Regression and Decision Tree.

V. Experimental Results

In our experimental evaluation, we empirically change the input features as follows:

- $n \in [4, 5, 6]$
- $d_0 \in [0.6, 0.7, 0.8, 0.9, 1]$
- $\alpha \in [0.5, 0.6]$
- $M_{min} \in [0.2, 0.25, 0.3, 0.4]$
- Obstacles locations: None (denoted as 'x'), or between 2 nodes randomly selected

As a result, we obtain a simulated dataset of 214 samples representing different WIoT scenarios and configurations. Table I shows the sample of the simulated dataset with the input features grid dimension n, sensor quality α, spacing between grid points d_0, miss probability and obstacle locations and calculated sensor positions.

Our generated dataset is rich in variability and serves as a valuable resource for further analysis. It is worth noting that that researchers can exploit such a dataset before deployment to evaluate multiple sensor placement options. Then, ML models can be used to automatically identify the best sensor placement using our tool: https://github.com/ksamrouth/Empirical-Dataset-Generation-for-AI-Optimized-IoT-Infrastructure-Placement.git.

To train the selected regression models, we split the simulated dataset (80% for training, 20% for testing) and we split the "Obstacles" column to 2 other columns "Obstacle start" and "Obstacle end" with padding to the sensor positions to ensure uniformity. In addition, we use the Mean Squared Error (MSE) as a training criterion for the different ML models.

Table II shows the evaluation results in terms of prediction accuracy representing the number of correctly predicted positions of sensor nodes. The decision tree regressor model outperforms the linear regression models with and without regularization (Lasso and Ridge) scoring the best evaluation metric of 78.94% accuracy. This is due to the inherent complexity and non-linearity in the relationship between the input features (grid dimension n, α, d_0, M_{min}, obstacles location) and the sensor node placement.

Unlike linear models, which may struggle with capturing the non-linear relationships, the decision tree can segment the

TABLE I: Sample of our simulated empirical dataset

n	α	d_0	M_{min}	Obstacles	Sensors Positions
4	1.0	0.6	0.4	x	6 11 7 10 3 14
4	0.9	0.6	0.4	x	6 11 7 10
4	0.8	0.6	0.4	x	6 11 7 10
4	0.7	0.6	0.4	x	6 11 7
4	0.6	0.6	0.4	x	6 11 7
4	0.5	0.6	0.4	x	6 11
4	0.5	0.6	0.2	x	6 11 7 10
4	0.5	0.6	0.3	x	6 11 7
4	0.5	0.6	0.25	x	6 11 7
4	0.5	0.6	0.35	x	6 11
4	0.6	0.6	0.4	11;14	11 6 7
4	0.6	0.6	0.4	7;12	6 10 7 11
4	0.7	0.6	0.2	6;11	7 10 8 9 15 2
4	0.7	0.6	0.2	1;6	7 10 11 6 8 9 15 3
4	0.7	0.6	0.2	11;14	6 7 10 11 2
4	0.7	0.6	0.2	7;12	6 7 10 11 2
4	0.7	0.6	0.3	6;11	7 10 8 9
4	0.7	0.6	0.3	1;6	7 10 11 6 8 9
4	0.7	0.6	0.3	11;14	6 11 7 10
4	0.7	0.6	0.3	7;12	6 7 10 11
4	0.7	0.6	0.4	6;11	7 10 8 9
4	0.7	0.6	0.4	1;6	7 10 11 6 8 9
4	0.7	0.6	0.4	11;14	6 11 7
5	0.7	0.6	0.3	8;13	12 14 7 19 9 17
5	0.7	0.6	0.3	1;8	13 8 18 14 12
5	0.7	0.6	0.3	15;18	13 12 18 8 7 17 15
5	0.7	0.6	0.3	17;24	13 12 14 8 17 19
5	0.7	0.6	0.4	8;13	12 14 7 19
5	0.7	0.6	0.4	1;8	13 12 18 14
5	0.7	0.6	0.4	15;18	13 12 18 8 7 17
5	0.7	0.6	0.4	17;24	13 12 14 8 17
6	0.6	0.6	0.2	10;16	21 15 22 14 23 27 9 17
6	0.6	0.6	0.2	22;29	16 21 15 17 27 14 10 20 9
6	0.6	0.6	0.2	21;26	16 21 15 22 20 17 9 28
6	0.6	0.6	0.2	8;15	16 21 22 20 10 23 27 17 28
6	0.6	0.6	0.3	10;16	21 15 22 14 23 27
6	0.6	0.6	0.3	22;29	16 21 15 17 27 14 10
6	0.6	0.6	0.3	21;26	16 21 15 22 20 7
6	0.6	0.6	0.3	8;15	16 21 22 20 10 23 27
6	0.6	0.6	0.4	10;16	21 15 22 14 23
5	0.8	0.6	0.2	8;13	12, 14, 7, 19, 9, 17, 18, 8
5	0.8	0.6	0.2	1;8	13, 8, 18, 14, 12, 19, 9, 17, 3
5	0.8	0.6	0.2	15;18	13, 12, 8, 18, 17, 7, 19, 9, 23
5	0.8	0.6	0.2	17;24	13, 12, 14, 8, 17, 19, 7, 9
5	0.8	0.6	0.3	8;13	12, 14, 7, 19, 9, 17
5	0.8	0.6	0.3	1;8	13, 8, 18, 14, 12, 19, 9
5	0.8	0.6	0.3	15;18	13, 12, 8, 18, 17, 7, 15, 9
5	0.8	0.6	0.3	17;24	13, 12, 14, 8, 17, 19, 7
6	0.8	0.6	0.2	10;16	21, 15, 22, 20, 23, 9, 17, 27, 14, 11
6	0.8	0.6	0.2	22;29	16, 21, 15, 17, 27, 9, 20, 10, 14, 23, 28
6	0.8	0.6	0.3	10;16	21, 15, 22, 20, 23, 9, 17
6	0.8	0.6	0.3	22;29	16, 21, 15, 17, 27, 9, 20, 10, 14
6	0.8	0.6	0.3	8;15	16, 21, 22, 10, 27, 23, 20, 17, 14
6	0.8	0.6	0.4	10;16	21, 15, 22, 20, 23, 9
6	0.8	0.6	0.4	22;29	16, 21, 15, 17, 27, 9, 20
6	0.8	0.6	0.4	21;26	22, 15, 16, 28, 9, 17
6	0.6	0.5	0.2	x	16, 21, 15, 22, 14, 23
6	0.6	0.5	0.25	x	16, 21, 15, 22, 14
6	0.6	0.5	0.3	x	16, 21, 15, 22
6	0.6	0.5	0.4	x	16, 21, 15, 22
6	0.8	0.5	0.2	x	16, 21, 15, 22, 17, 20, 28, 9
6	0.8	0.6	0.2	21;26	22, 15, 16, 28, 9, 17, 14, 23, 10, 21, 26
6	0.8	0.6	0.2	8;15	21, 16, 22, 10, 27, 23, 20, 17, 14, 9
6	0.8	0.5	0.25	x	16, 21, 15, 22, 17, 20, 28, 9
6	0.8	0.5	0.3	x	16, 21, 15, 22, 17, 20, 28
6	0.8	0.6	0.3	21;26	22, 15, 16, 28, 9, 17, 14, 23
6	0.8	0.5	0.4	x	16, 21, 15, 22, 17

feature space based on conditions such as "if $n < 6$ and obstacles are present between nodes i and j, then place sensors in a certain place." This segmentation approach enables the model to better accommodate diverse patterns in the data.

TABLE II: Machine Learning Models Evaluation

Model	Percentage of Correctly Positioned Sensor Nodes
Linear Regression	30.22%
Polynomial Regression	33.20%
Ridge Regression	24.50%
Lasso Regression	23.19
Decision Tree Regressor	78.94%

VI. Conclusion

In this paper, we present a data-driven methodology to predict sensor deployment outcomes across varied scenarios. In particular, we develop a tool that allows IoT designers to simulate and generate datasets for multiple sensor deployment options. We also introduce an approach to generate datasets that serve as a foundation for training ML models to optimize sensor placement in WIoTs. The dataset is instrumental in evaluating the effectiveness of the proposed algorithms and lays the groundwork for developing advanced AI models to enhance coverage strategies in diverse deployment settings. By generating simulated data and ML predictions, our approach offers scalable and adaptable solutions for WIoT deployments. These innovations have the potential to reduce setup costs, extend sensor lifespan, and ensure consistent achievement of coverage objectives across different environments.

As a future work, we will work on explaining results with XAI tools. Moreover, we aim to generate complex datasets with additional input features such as field dimensions, area type (urban, rural, underwater), and maximum node count. Furthermore, we aim to use an advanced complex criterion during ML models training such as Fitness function that includes several terms like Coverage Area, Node Proximity Penalty, Coverage Hole Penalty and Deployment Cost.

References

[1] R. Yus, G. Bouloukakis, S. Mehrotra, and N. Venkatasubramanian, "Abstracting interactions with iot devices towards a semantic vision of smart spaces," in *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, ser. BuildSys '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 91–100. [Online]. Available: https://doi.org/10.1145/3360322.3360859

[2] H. H. Hassan, J. Ma, G. Bouloukakis, R. Yus, and A. Kattepur, "Efficient scheduling of smart building energy systems using AI planning," in *International Conference on ICT for Sustainability (ICT4S)*. Stockholm, Sweden: IEEE, Jun. 2024. [Online]. Available: https://inria.hal.science/hal-04605818

[3] Z. Ahmed, D. Gui, G. Murtaza, L. Yunfei, and S. Ali, "An overview of smart irrigation management for improving water productivity under climate change in drylands," *Agronomy*, vol. 13, no. 8, 2023. [Online]. Available: https://www.mdpi.com/2073-4395/13/8/2113

[4] E. F. Orumwense and K. Abo-Al-Ez, "Applications, challenges and future trends towards enabling internet of things for smart energy systems," in *2022 IEEE Nigeria 4th International Conference on Disruptive Technologies for Sustainable Development (NIGERCON)*, 2022, pp. 1–7.

[5] K. Lawal and H. N. Rafsanjani, "Trends, benefits, risks, and challenges of iot implementation in residential and commercial buildings," *Energy and Built Environment*, vol. 3, no. 3, pp. 251–266, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666123321000167

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

[7] D. Múnera, D. V., J. Aguirre, and N. Gaviria, "Iot-based air quality monitoring systems for smart cities: A systematic mapping study," *International Journal of Electrical and Computer Engineering*, vol. 11, p. 34703482, 08 2021.

[8] A. Chio, D. Jiang, P. Gupta, G. Bouloukakis, R. Yus, S. Mehrotra, and N. Venkatasubramanian, "Smartspec: Customizable smart space datasets via event-driven simulations," in *2022 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2022, pp. 152–162.

[9] A.-E. Marcu, G. Suciu, E. Olteanu, D. Miu, A. Drosu, and I. Marcu, "Iot system for forest monitoring," in *2019 42nd International Conference on Telecommunications and Signal Processing (TSP)*, 2019, pp. 629–632.

[10] H. Shahab, M. Iqbal, A. Sohaib, F. Ullah Khan, and M. Waqas, "Iot-based agriculture management techniques for sustainable farming: A comprehensive review," *Computers and Electronics in Agriculture*, vol. 220, p. 108851, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169924002424

[11] K. Zaimen, M.-E.-A. Brahmia, L. Moalic, A. Abouaissa, and L. Idoumghar, "A survey of artificial intelligence based wsns deployment techniques and related objectives modeling," *IEEE Access*, vol. 10, pp. 113 294–113 329, 2022.

[12] N. Bakir, A. Samrouth, and K. Samrouth, "Pso-ga-based federated learning for predicting energy consumption in smart buildings," in *2023 International Conference on Microelectronics (ICM)*, 2023, pp. 234–238.

[13] K. V. N. A. Bhargavi, G. P. S. Varma, I. Hemalatha, and R. Dilli, "An enhanced particle swarm optimization-based node deployment and coverage in sensor networks," *Sensors*, vol. 24, no. 19, 2024. [Online]. Available: https://www.mdpi.com/1424-8220/24/19/6238

[14] L. Dai and B. Wang, "Sensor placement based on delaunay triangulation for complete confident information coverage in an area with obstacles," 12 2015, pp. 1–8.

[15] Q. Zhao, C. Li, D. Zhu, and C. Xie, "Coverage optimization of wireless sensor networks using combinations of pso and chaos optimization," *Electronics*, vol. 11, no. 6, 2022. [Online]. Available: https://www.mdpi.com/2079-9292/11/6/853

[16] B. Poggi, C. Babatounde, E. Vittori, and T. Antoine-Santoni, "Efficient wsn node placement by coupling knn machine learning for signal estimations and i-hbia metaheuristic algorithm for node position optimization," *Sensors*, vol. 22, no. 24, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/24/9927