Group 3

Smart Building Sensor Placement PSO + SA Algorithm Implementation Detail

Clarification on PSO Hybrid combination (PSO + SA): We intend to use **PSO** to perform global exploration of the entire search space in combination with **Simulated Annealing (SA)** for better local convergence. As the PSO algorithm explores the search space and determines the best sensor placement, if it converges too quickly or misses more optimal solutions, SA comes into play to focus on local refinement. Using SA, the algorithm can be programmed to make small changes and occasionally accept worse solutions on purpose so that the PSO algorithm can spend more time optimizing for an even better solution rather than settling for a sub-optimal solution early. Therefore, the reason we chose SA is to improve local search and escape local minima which is a hybrid solution that makes sense for our use case.

Clarification on Sensor Data Simulation and Implementation Plan:
We are planning to use Python as our tool of choice.

The goal is to maximize the area covered by the sensor, minimize the number of sensors to reduce cost as well as unnecessary overlaps between sensors.

Step 1: Building the grid and assigning room types and weights
We plan to simulate the building using a 2D grid where each cell in the grid corresponds to a specific location inside the building from a side-view perspective. The goal is to create a simplistic model of the interior structure and simulate how sensors can be best placed within this space.

- Create an object to store grid details.
- Defining the overall grid dimensions including height and width to represent the shape and scale of the building and each room.
- Each room type (living areas, hallways, basements) will be identified within the object, so that weights can be assigned based on room type to determine which room requires more sensor coverage.
- Define the coordinates along the border of each grid and cell in terms of hard surfaces vs open space.

Step 2: Defining sensor objects and placement constraints

- Initial x (row) and y (column) position on the grid.
- Fixed radius to determine the range of detection for each sensor.
- Define placement constraints such that valid placement of sensors only include coordinates that fall under hard surfaces (not open spaces).

Step 3: Simulating coverage and overlap
Apply the initial sensor position to the pre-defined grid layout, and determine:

- How much area of each cell (room) is covered by each sensor saved to a coverage map.
- Number of sensors covering each cell saved to an overlap map.
- Assume hard surfaces will not block signals and signal strength remains constant within defined radius to keep the simulation more simplistic.

Step 4: Defining the fitness function

Determine a fitness function that evaluates the quality of sensor arrangement on grid border with a goal of minimizing the score:

- Penalizes important cells based on room weights that are not accounted for in the coverage map. (penalty_1)
- Penalizes cells in the overlap map that are covered by more than one sensor. (penalty_2)
- Penalizes the total number of sensors used. (penalty_3)

Eg. Fitness = weight_1 * penalty_1 + weight_2 * penalty_2 + weight_3 * penalty_3

***Step 5:* Define Code for Particle Swarm Optimization (PSO)**

- Initialize a population of particles to represent each individual sensor
- Iteratively update sensor positions based on best solution versus swarm's global best solution
- Explore the search space to maximize coverage, minimize overlaps, and reduce the total number of sensors.
- Apply the fitness function in step 4 to evaluate quality of sensor arrangement on grid border
- Define a maximum number of iterations and/or maximum threshold for improvement after some iterations

***Step 6:* Define Code for Simulated Annealing (SA)**

- After the PSO algorithm finds an optimal sensor layout, run SA to fine-tune layouts.
- Ensure SA makes small random moves to sensor position within allowed placements only.
- Program SA to accept some changes that make the layout a little worse at first.
- Program SA to stop accepting as many sub-optimal solutions over time to gradually improve the layout by optimizing the local minima.
- Define

Step 7: Final evaluation and confirmation

- After SA run is completed, evaluate the layout's fitness again
- Plot appropriate graphs for simulation
- Log final coverage, overlap, sensor count, and fitness score to console
- Multiple runs can be conducted to select the best placement layout generated