Distributed Optimization for Sensor Placement: Exploring Strongly Connected, Periodically Connected, and Gossip Algorithms

Shayan Malekpour

January 13, 2024

Problem Description

You have N sensors deployed in a 2D space, aiming to optimize their positions for accurate coverage of five points. The goal is to minimize a cost function based on distances between the sensors and these points. Additionally, each point must be covered by at least three sensors, and there is a threshold constraint for the detection range of sensors.

Distance Calculation

The distance (Distance_{ik}) from sensor k to point i is calculated using the Euclidean distance formula:

$$\text{Distance}_{ik} = \sqrt{(x_i - x_k)^2 + (y_i - y_k)^2}$$

Cost Function

The cost function (J) to be minimized is the summation of the exponential of the negative distances for all points and sensors:

$$J = \sum_{i=1}^{5} \sum_{k=1}^{N} e^{-\sqrt{(x_i - x_k)^2 + (y_i - y_k)^2}}$$

Constraints

- **Coverage Constraint:** Each point should be covered by at least three sensors:

  $$\sum_{k=1}^{N} \text{BinaryVar}_{ik} \geq 3$$

  where BinaryVar_{ik} is a binary variable indicating whether sensor k covers point i.

- **Threshold Distance Constraint:** Ensure that the distance between each sensor and each point is within the threshold:

  $$\text{Distance}_{ik} \leq \text{Threshold}$$

  for all i and k.

- **Sensor Binary Variable:** Binary variable indicating whether a sensor is active or not:

  $$\text{SensorVar}_k = \{0, 1\}$$

  where $\text{SensorVar}_k = 1$ if sensor k is active.

Objective

Minimize the cost function J subject to the coverage and threshold distance constraints.

Optimization Algorithm Steps

Step 1: Initialization

- Initialize each agent's position randomly in the 2D space.
- Establish a strongly connected network graph representing communication links between agents.

Step 2: Distributed Consensus

For each connected pair of agents:

$$\text{if i \& j are connected:} \quad \text{AssumedPosition}_{ij}^{(t+1)} = \text{ActualPosition}_{i}^{(t)}$$

$$\text{else:} \quad \text{AssumedPosition}_{ij}^{(t+1)} = \text{AssumedPosition}_{i}^{(t)}$$

where $\text{AssumedPosition}_{ij}^{(t)}$ is the assumed position of agent i by agent j at iteration t, and $\text{ActualPosition}_{i}^{(t)}$ is the actual position of agent i at iteration t.

Step 3: Cost Function Optimization

Each agent optimizes its own cost function based on the updated assumptions:

$$\text{Minimize } J_i = -\sum_{k=1}^{N} e^{-\sqrt{(x_i - x_k)^2 + (y_i - y_k)^2}}$$

Using a suitable optimization algorithm (e.g., gradient descent):

$$\text{ActualPosition}_{i}^{(t+1)} = \text{ActualPosition}_{i}^{(t)} - \gamma \cdot \nabla J_i$$

where γ is the step size for optimization.

Step 4: Constraint Satisfaction

Check if the coverage and distance constraints are satisfied. If violated:

$$\text{ActualPosition}_{i}^{(t+1)} = \text{NearestFeasiblePosition}(\text{ActualPosition}_{i}^{(t+1)})$$

Step 5: Convergence Check

Check for convergence. If not converged, go back to Step 2.

0.1 results

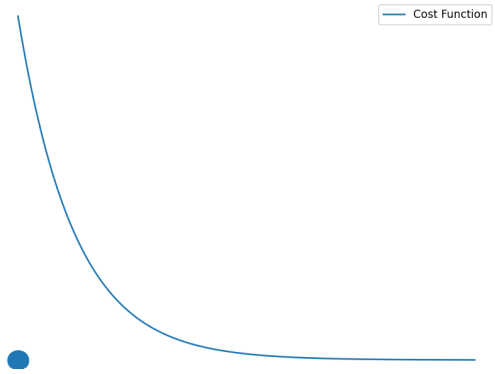

Figure 1: cost function

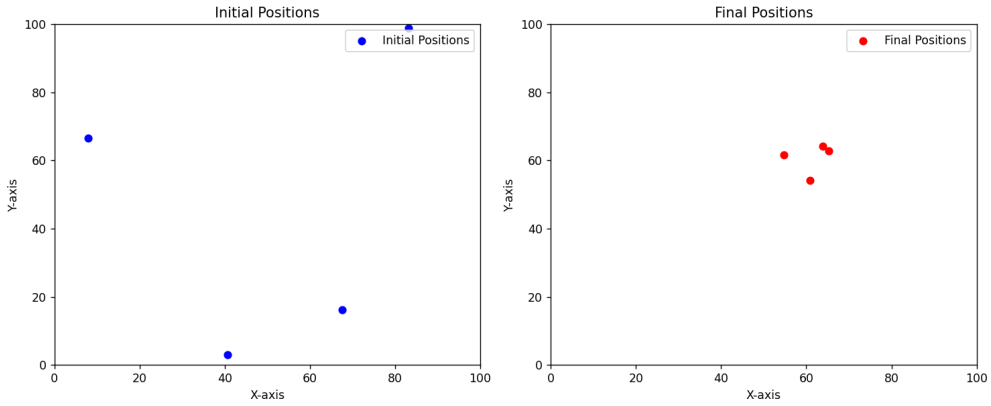

Figure 2: cost function

1 Periodically Connected Network Graph

The network graph is designed to be periodically connected, creating a structure where agents have connections not only with their immediate neighbors but also with agents on the opposite side of the grid. This periodicity is achieved by generating a 2D grid graph using the `grid_2d_graph` function from the `networkx` library with periodic connections enabled.

The process of creating the periodically connected graph involves the following steps:

1. **Grid Graph Generation:** The initial step is to generate a 2D grid graph with a specified number of rows and columns. This graph represents the spatial arrangement of agents in the 2D space.

2. **Periodic Connections:** To introduce periodic connections, the `grid_2d_graph` function is used with the parameter `periodic=True`. This setting ensures that connections wrap around the edges of the grid, creating a toroidal structure.

3. **Probability of Connection:** Optionally, a probability parameter can be introduced to control the likelihood of connections between agents. For instance, the `p` parameter in the `grid_2d_graph` function allows for introducing randomness in the connections. Higher values of `p` increase the probability of having long-range connections, enhancing the periodicity of the graph.

4. **Adjusting Grid Size:** The size of the grid, determined by the number of rows and columns, influences the overall structure of the periodically connected graph. Experimenting with the grid size allows for tuning the balance between local and periodic connections.

1.1 results

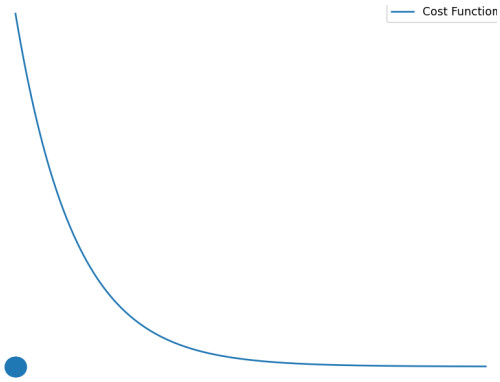

Figure 3: cost function

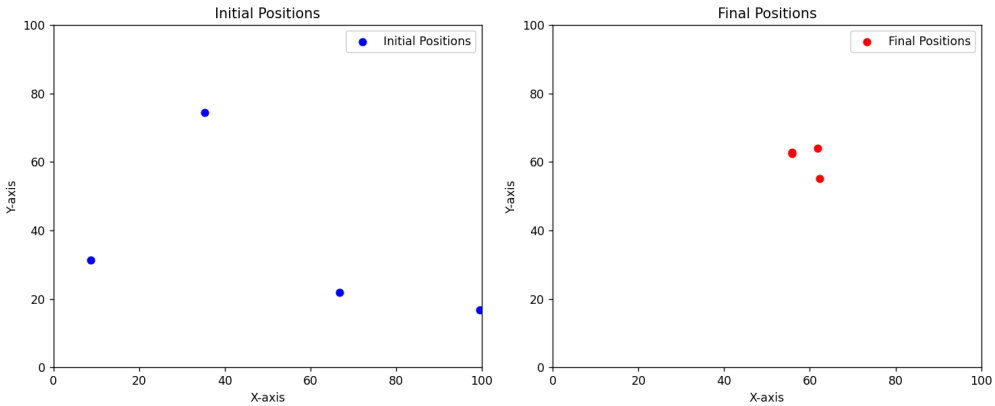

Figure 4: cost function

2 Gossip Algorithm

The gossip algorithm is employed to enable information exchange among sensors in a decentralized manner, facilitating collaborative optimization towards a global objective function. The algorithm involves the following steps:

1. **Initialization:** Each sensor initializes its position randomly in the 2D space.
2. **Network Graph Creation:** Establish a periodically connected network graph to represent communication links between sensors. Introduce a probability parameter to control the likelihood of connections between sensors.
3. **Gossip Interaction:** At each iteration, sensors select random neighbors to exchange information with. The information can include local positions or gradients.
4. **Objective Function Update:** Update the local objective function for each sensor based on the exchanged information.
5. **Convergence Check:** Monitor the change in the global objective function or the positions of sensors to check for convergence.

Detailed Implementation

Initialization

Initialize each sensor's position randomly in the 2D space.

Network Graph Creation

Establish a periodically connected network graph using the `new_net` function. The probability of connections can be controlled by adjusting parameters in the graph generation process.

Gossip Interaction

At each iteration:

- Sensors randomly select a neighbor to exchange information with.
- Information exchange can involve sharing local positions, distances, or gradients.

Objective Function Update

Update the local objective function for each sensor based on the exchanged information. For example, if using a cost function J:

$$J_i = \sum_{k=1}^{N} e^{-\sqrt{(x_i - x_k)^2 + (y_i - y_k)^2}}$$