Sensor Placement for Structural Health Monitoring

Framework and Algorithms

Geethu Joseph

Joint work with

- Prof. Chandra R. Murthy, Indian Institute of Science, Bangalore
- Prof. John V. Mathews, Oregon State University

System Model

Why SHM?

Many infrastructure uses fiber reinforced laminate composite materials

Aircrafts

Oil pipes

✓ Lightweight and strong

✓ Excellent fatigue and corrosion resistance

,

Why SHM?

Many infrastructure uses fiber reinforced laminate composite materials

Aircrafts

Oil pipes

✓ Lightweight and strong

✓ Excellent fatigue and corrosion resistance
,

✗ Damage is invisible to naked eye

Why SHM?

Many infrastructure uses fiber reinforced laminate composite materials

Aircrafts

Oil pipes

✓ Lightweight and strong

✓ Excellent fatigue and corrosion resistance

✗ Damage is invisible to naked eye

Safety and maintenance $\Rightarrow$ SHM!

What is SHM?

Structural health is continuously monitored

Real-time inspection

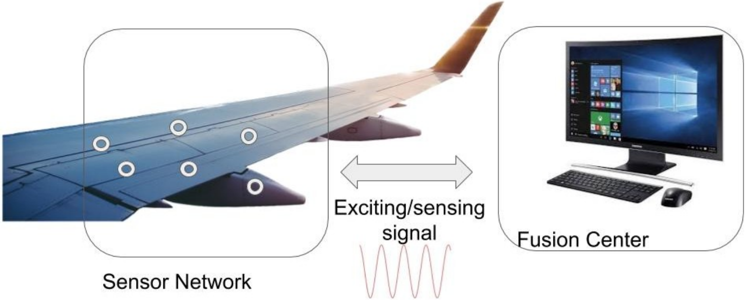
Ground maintenance

What is SHM?

Structural health is continuously monitored

Real-time inspection

Ground maintenance



How is SHM implemented?

Sensor network

Excitation:
Each actuator excites the structure sequentially

↕

Sensing:
Remaining sensors collect the resulting signals

⟹

Fusion Center

Estimation:
Collected signals are sent to fusion center to estimate damage, if any

$$n \text{ sensors} \implies n(n-1) \text{ measurements.}$$

Sensor Coverage

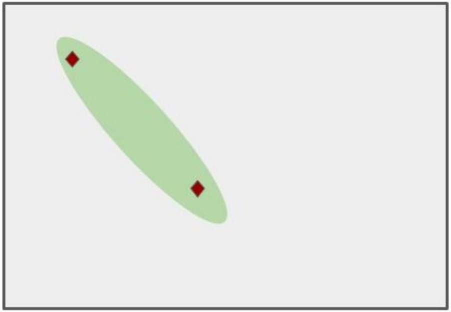
Coverage of (i, j) actuator-sensor pair

$$\text{Area: } \mathcal{R}_{ij} = \left\{ \boldsymbol{x} \in \mathcal{A} : \underbrace{\frac{\|\boldsymbol{x} - \boldsymbol{s}_i\| + \|\boldsymbol{x} - \boldsymbol{s}_j\|}{\|\boldsymbol{s}_j - \boldsymbol{s}_i\|}}_{r_{ij}(\boldsymbol{x})} \leq \beta \right\}$$

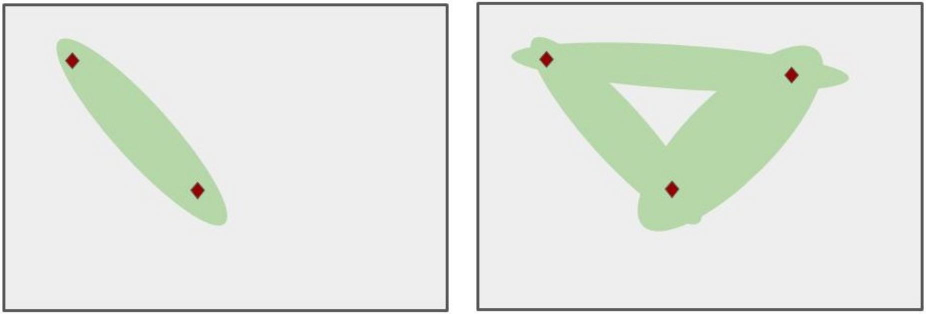
$$\text{Probability: } p_{ij}(\boldsymbol{x}) = \begin{cases} \frac{\beta - r_{ij}(\boldsymbol{x})}{\beta - 1} e^{-\eta \|\boldsymbol{s}_j - \boldsymbol{s}_i\|} & \text{if } \boldsymbol{x} \in \mathcal{R}_{ij} \\ 0 & \text{otherwise.} \end{cases}$$

- $\boldsymbol{s}_i, \boldsymbol{s}_j$: Sensor locations
- β, η : Known constants

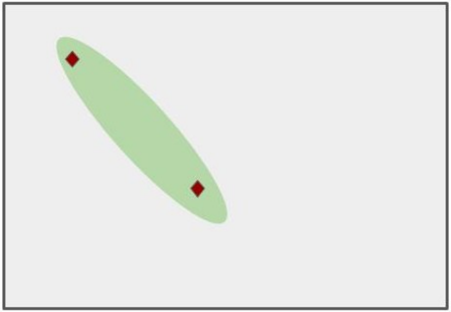
Illustrative Example

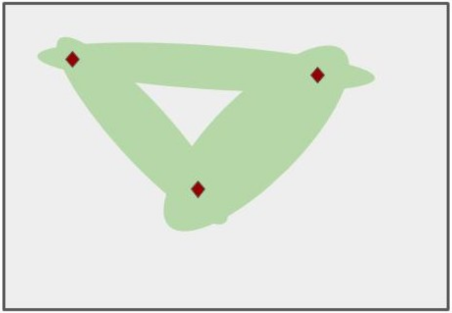


Illustrative Example



Illustrative Example

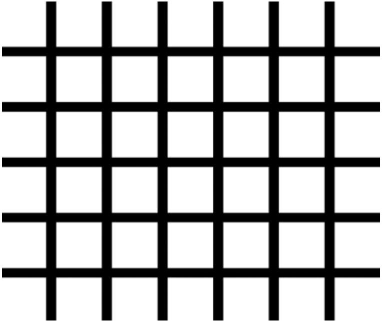




Goal:	Every anomaly point in $\mathcal{A}$ is detected with probability at least $p_{\min}$
Problem:	Find **minimum number** of sensors and their placement

Framework

Grid Based Approach



$\mathcal{A}$ is discretized to N grid points

Grid locations: $\boldsymbol{x}_m, m = 1, 2 \ldots, N$

Goal: Every **grid point** is detected with probability at least p_{min}

Problem: Find minimum number of sensors and their placement on **grid points**

Formulation

Constraint

A grid point is missed if all the actuator-sensor pairs miss it

$$1 - \prod_{i,j=1}^{N_s} [1 - p_{ij}(\boldsymbol{x}_m)] \geq p_{\min}, \quad \text{for } m = 1, 2, \ldots, N.$$

N_s : Number of sensors placed

Equivalent constraint

$$\sum_{i,j=1}^{N_s} \log [1 - p_{ij}(\boldsymbol{x}_m)] \leq \log(1 - p_{\min}), \quad \text{for } m = 1, 2, \ldots, N.$$

N_s : Number of sensors placed

Formulation

$$\sum_{i,j=1}^{N_s} \log\left[1 - p_{ij}(\boldsymbol{x}_m)\right] \leq \log(1 - p_{\min}), \quad \text{for } m = 1, 2, \ldots, N.$$

Indicator vector for sensor placement: $\boldsymbol{z} \in \{0,1\}^N$

Sensor-sensor pair mapping function: $g : \{0,1\}^N \to \{0,1\}^{\binom{N}{2}}$

Sensing matrix: $\boldsymbol{A}^{\text{cover}} \in \mathbb{R}^{N \times \binom{N}{2}}$

$$\boldsymbol{A}^{\text{cover}}_{m,n} = \frac{\log\left[1 - p_{f^{-1}(n)}\left(\boldsymbol{x}_m\right)\right]}{\log(1 - p_{\min})}.$$

Equivalent Constraint

$$\boldsymbol{A}^{\text{cover}} g(z) \geq \mathbf{1}$$

Final Problem

$$\boldsymbol{A}^{\text{cover}} g(\boldsymbol{z}) \geq \mathbf{1}$$

If we replace $\boldsymbol{A}^{\text{cover}}$ with $\boldsymbol{A}$, constraint remains unchanged

$$\boldsymbol{A}_{m,n} = \min\left\{1, \frac{\log\left[1 - p_{f^{-1}(n)}(\boldsymbol{x}_m)\right]}{\log(1 - p_{\min})}\right\}.$$

Optimization Problem

$$\boldsymbol{z}^* = \operatorname*{arg\,min}_{\boldsymbol{z} \in \{0,1\}^N} \|\boldsymbol{z}\|_0 \text{ subject to } \boldsymbol{A} g(\boldsymbol{z}) \geq \mathbf{1}.$$

Some Observations

1. The size of the smallest anomaly area that can be detected depends on the grid size
2. The framework does not depend on the model
3. The feasible set is always nonempty as it always contains the all-one vector.

Solution

Optimization Problem

$$\boldsymbol{z}^* = \underset{\boldsymbol{z}\in\{0,1\}^N}{\arg\min} \|\boldsymbol{z}\|_0 \text{ subject to } \boldsymbol{A}g(\boldsymbol{z}) \geq \mathbf{1}.$$

Key challenges:

- $\boldsymbol{z}$ is constrained to have only 0 or 1 as entries
- $g(\boldsymbol{z})$ is a non-convex function of $\boldsymbol{z}$

Solution

Optimization Problem

$$\boldsymbol{z}^* = \arg\min_{\boldsymbol{z}\in\{0,1\}^N} \|\boldsymbol{z}\|_0 \text{ subject to } \boldsymbol{A}g(\boldsymbol{z}) \geq \mathbf{1}.$$

Key challenges:

- $\boldsymbol{z}$ is constrained to have only 0 or 1 as entries
- $g(\boldsymbol{z})$ is a non-convex function of $\boldsymbol{z}$

What Next?

Algorithms

Our Options

Optimization Problem

$$\boldsymbol{z}^* = \underset{\boldsymbol{z} \in \{0,1\}^N}{\arg\min} \|\boldsymbol{z}\|_0 \text{ subject to } \boldsymbol{A}g(\boldsymbol{z}) \geq \mathbf{1}.$$

1 Convex Relaxation   Gives poor results ✗

Our Options

Optimization Problem

$$\boldsymbol{z}^* = \underset{\boldsymbol{z}\in\{0,1\}^N}{\arg\min} \|\boldsymbol{z}\|_0 \text{ subject to } \boldsymbol{A}g(\boldsymbol{z}) \geq \mathbf{1}.$$

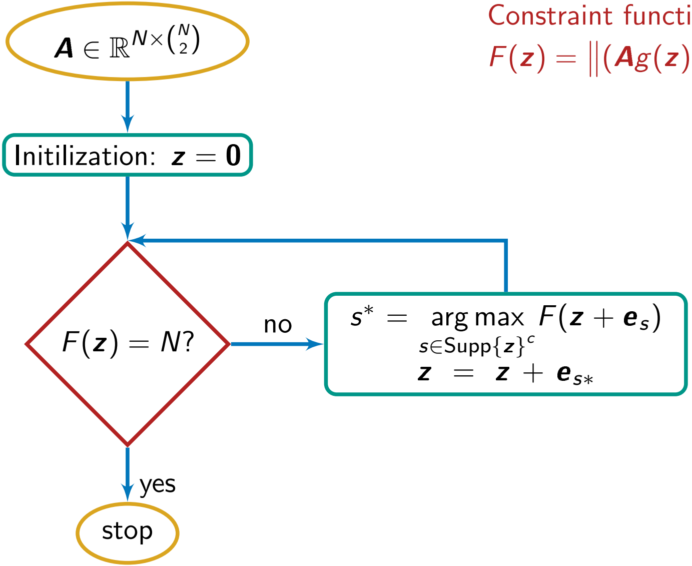
1 Convex Relaxation — Gives poor results ✗

2 Greedy Algorithm — Gives near optimal results! ✓ (Surprising?)

Greedy Algorithm

Constraint function:

$F(\boldsymbol{z}) = \left\|(\boldsymbol{A}g(\boldsymbol{z}) - \mathbf{1})_+\right\|_0$



Computational Complexity

Assume that the algorithm stops after N_S^* iterations

Greedy Algorithm $\mathcal{O}(N_s^* N^4)$ ✓

Exhaustive search $\mathcal{O}(2^N)$ ✗

Computational Complexity

Assume that the algorithm stops after N_S^* iterations

Greedy Algorithm $\mathcal{O}(N_s^* N^4)$ ✓

Exhaustive search $\mathcal{O}(2^N)$ ✗

Even faster?

Fast Greedy Algorithm

1 Elimination of rows of A

- Once a grid point is covered, it will remain covered
- Eliminate rows corresponding to the covered grid points

2 Recursive update of constraint function

- When a new sensor is placed, compute the change in the coverage function

Comparison

Assume that the algorithm stops after N_S^* iterations

Fast Greedy Algorithm	$\mathcal{O}(N_s^* N^3)$ ✓
Greedy Algorithm	$\mathcal{O}(N_s^* N^4)$ ✗
Exhaustive search	$\mathcal{O}(2^N)$ ✗

Comparison

Assume that the algorithm stops after N_S^* iterations

Fast Greedy Algorithm	$\mathcal{O}(N_s^* N^3)$ ✓	Suboptimal
Greedy Algorithm	$\mathcal{O}(N_s^* N^4)$ ✗	Suboptimal
Exhaustive search	$\mathcal{O}(2^N)$ ✗	Optimal ✓

Comparison

Assume that the algorithm stops after N_S^* iterations

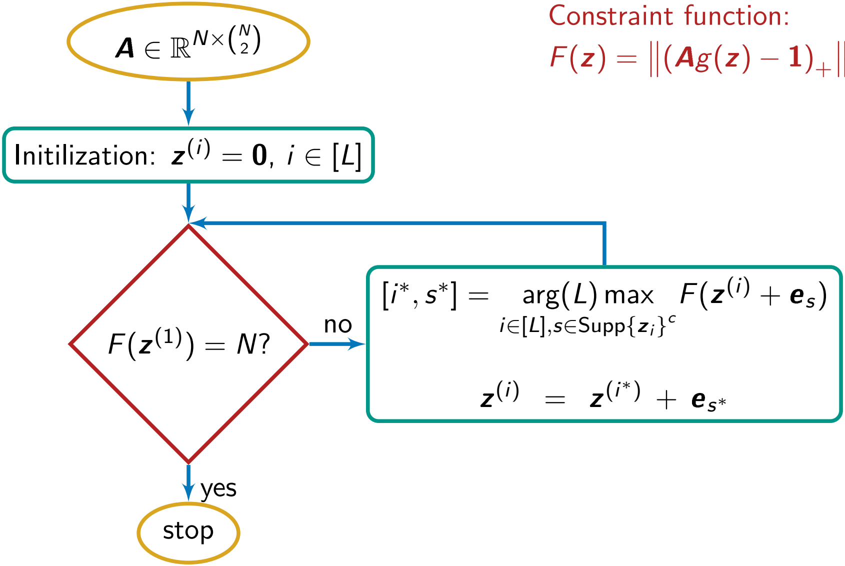
Fast Greedy Algorithm	$\mathcal{O}(N_s^* N^3)$ ✓	Suboptimal
Greedy Algorithm	$\mathcal{O}(N_s^* N^4)$ ✗	Suboptimal
Exhaustive search	$\mathcal{O}(2^N)$ ✗	Optimal ✓

Improve performance?

Group Greedy Algorithm

Constraint function:

$F(\boldsymbol{z}) = \left\| (\boldsymbol{A}g(\boldsymbol{z}) - \mathbf{1})_+ \right\|_0$



C. Jiang, et al., Group Greedy Method for Sensor Placement, TSP, May, '19.

Example: $N = 5, L = 2$

	Choices	$\{1\}, \{2\}, \{3\}, \{4\}, \{5\}$
$k = 1$	Best two	$\{1\}, \{3\}$
	Selection	$\{1\}$

	Choices	$\{1,2\}, \{1,3\}\{1,4\}, \{1,5\}, \{3,2\}, \{3,4\}, \{3,5\}$
$k = 2$	Best two	$\{1,2\}, \{1,3\}$
	Selection	$\{1,2\}$

	Choices	$\{1,2,3\}, \{1,2,4\}, \{1,2,5\}, \{1,3,4\}, \{1,3,5\}$
$k = 3$	Best two	$\{1,3,5\}, \{1,2,4\}$
	Selection	$\{1,3,5\}$

Final Comparison

Assume that the algorithm stops after N_S^* iterations

Group Greedy Algorithm	$\mathcal{O}(N_s^* N^3 \underbrace{L}_{\text{Extra}})$
Fast Greedy Algorithm	$\mathcal{O}(N_s^* N^3)$
Greedy Algorithm	$\mathcal{O}(N_s^* N^4)$

Summary

Problem Coverage of an area using sensor pairs

Framework Grid-based approach

Algorithms

1. Greedy algorithm
2. Fast greedy algorithm **Least computations** ✓
3. Group greedy algorithm **Best performance** ✓

Thank You!