Explicit Sensor Network Localization using Semidefinite Representations and Clique Reductions

Nathan Krislock, Henry Wolkowicz

Department of Combinatorics & Optimization
University of Waterloo

ISMP, Chicago
August 25, 2009

Introduction

The Sensor Network Localization (SNL) Problem

Given:

- Distances between sensors within a fixed radio range
- Positions of some fixed sensors (called anchors)

Goal:

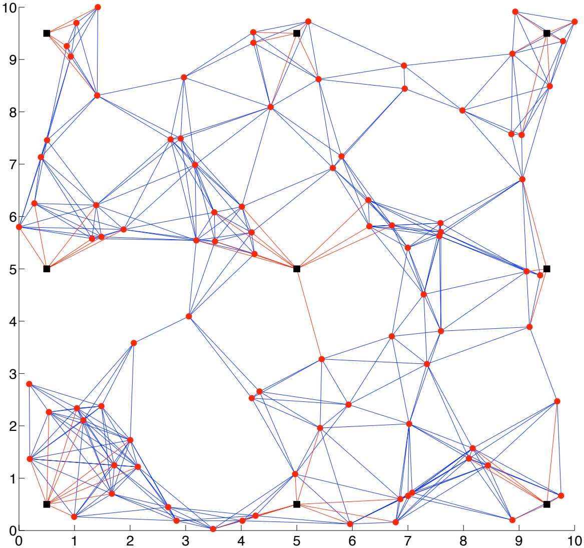
- Determine locations of sensors

Motivation

Many applications use wireless sensor networks:

- natural habitat monitoring, weather monitoring, tracking of goods, random deployment in inaccessible terrains, surveillance, ...

Introduction



n = 100, m = 9, R = 2

Outline

1. Sensor Network Localization (SNL)
   - Introduction
   - Euclidean Distance Matrices and Semidefinite Matrices
2. Clique Reductions of SNL
   - Clique Reductions
   - Computing Sensor Positions
3. Algorithm
   - Clique Unions and Node Absorptions
   - Results

Outline

1. Sensor Network Localization (SNL)
   - Introduction
   - Euclidean Distance Matrices and Semidefinite Matrices
2. Clique Reductions of SNL
   - Clique Reductions
   - Computing Sensor Positions
3. Algorithm
   - Clique Unions and Node Absorptions
   - Results

Outline

1. Sensor Network Localization (SNL)
   - Introduction
   - Euclidean Distance Matrices and Semidefinite Matrices
2. Clique Reductions of SNL
   - Clique Reductions
   - Computing Sensor Positions
3. Algorithm
   - Clique Unions and Node Absorptions
   - Results

Outline

1 Sensor Network Localization (SNL)
- Introduction
- Euclidean Distance Matrices and Semidefinite Matrices

2 Clique Reductions of SNL
- Clique Reductions
- Computing Sensor Positions

3 Algorithm
- Clique Unions and Node Absorptions
- Results

Introduction

Notation

- $p_1, \ldots, p_{n-m} \in \mathbb{R}^r$ - unknown points (sensors)
- $a_1, \ldots, a_m \in \mathbb{R}^r$ - known points (anchors)
  - anchors also labeled $p_{n-m+1}, \ldots, p_n$

$$P = \begin{bmatrix} p_1^T \\ \vdots \\ p_n^T \end{bmatrix} = \begin{bmatrix} X \\ A \end{bmatrix} \in \mathbb{R}^{n \times r}$$

- r - embedding dimension (usually 2 or 3)
- $R > 0$ - radio range

Graph Realization

- $G = (N, E, w)$ - underlying weighted graph
  - $N = \{1, \ldots, n\}$
  - $(i,j) \in E$ if $w_{ij} = \|p_i - p_j\| < R$
- SNL problem $\equiv$ find realization of graph in $\mathbb{R}^r$

Euclidean Distance Matrix (EDM) Completion

- $D_p \in \mathcal{S}^n$ - partial EDM:

$$(D_p)_{ij} = \begin{cases} \|p_i - p_j\|^2 & \text{if } (i,j) \in E \\ ? & \text{otherwise} \end{cases}$$

- SNL problem $\equiv$ find EDM completion with embed. dim. $= r$

Outline

1. Sensor Network Localization (SNL)
   - Introduction
   - Euclidean Distance Matrices and Semidefinite Matrices
2. Clique Reductions of SNL
   - Clique Reductions
   - Computing Sensor Positions
3. Algorithm
   - Clique Unions and Node Absorptions
   - Results

EDMs and Semidefinite Matrices

Linear Transformation $\mathcal{K}$

- If D is an EDM with embed. dim. r given by $P \in \mathbb{R}^{n\times r}$, then:

$$\begin{aligned} D_{ij} = \|p_i - p_j\|^2 &= p_i^T p_i + p_j^T p_j - 2p_i^T p_j \\ &= \left(\operatorname{diag}(PP^T)e^T + e\operatorname{diag}(PP^T)^T - 2PP^T\right)_{ij} \\ &= \mathcal{K}(PP^T)_{ij} \end{aligned}$$

- Thus $D = \mathcal{K}(Y)$, where:

$$\boxed{\mathcal{K}(Y) := \operatorname{diag}(Y)e^T + e\operatorname{diag}(Y)^T - 2Y} \quad \text{and} \quad Y := PP^T$$

- $Y = PP^T$ is positive semidefinite, $\operatorname{rank}(Y) = r$
- $\mathcal{K}$ maps the semidefinite cone, $\mathcal{S}_+^n$, onto the EDM cone, $\mathcal{E}^n$

EDMs and Semidefinite Matrices

Linear Transformation $\mathcal{K}$

- If D is an EDM with embed. dim. r given by $P \in \mathbb{R}^{n\times r}$, then:

$$\begin{aligned} D_{ij} = \|p_i - p_j\|^2 &= p_i^T p_i + p_j^T p_j - 2p_i^T p_j \\ &= \left(\mathrm{diag}(PP^T)e^T + e\,\mathrm{diag}(PP^T)^T - 2PP^T\right)_{ij} \\ &= \mathcal{K}(PP^T)_{ij} \end{aligned}$$

- Thus $D = \mathcal{K}(Y)$, where:

$$\boxed{\mathcal{K}(Y) := \mathrm{diag}(Y)e^T + e\,\mathrm{diag}(Y)^T - 2Y} \quad \text{and} \quad Y := PP^T$$

- $Y = PP^T$ is positive semidefinite, $\mathrm{rank}(Y) = r$
- $\mathcal{K}$ maps the semidefinite cone, $\mathcal{S}_+^n$, onto the EDM cone, $\mathcal{E}^n$

EDMs and Semidefinite Matrices

Linear Transformation $\mathcal{K}$

- If D is an EDM with embed. dim. r given by $P \in \mathbb{R}^{n\times r}$, then:

$$\begin{aligned} D_{ij} = \|p_i - p_j\|^2 &= p_i^T p_i + p_j^T p_j - 2 p_i^T p_j \\ &= \left(\operatorname{diag}(PP^T)e^T + e\operatorname{diag}(PP^T)^T - 2PP^T\right)_{ij} \\ &= \mathcal{K}(PP^T)_{ij} \end{aligned}$$

- Thus $D = \mathcal{K}(Y)$, where:

$$\boxed{\mathcal{K}(Y) := \operatorname{diag}(Y)e^T + e\operatorname{diag}(Y)^T - 2Y} \quad \text{and} \quad Y := PP^T$$

- $Y = PP^T$ is positive semidefinite, $\operatorname{rank}(Y) = r$
- $\mathcal{K}$ maps the semidefinite cone, $\mathcal{S}^n_+$, onto the EDM cone, $\mathcal{E}^n$

EDMs and Semidefinite Matrices

Vector Formulation

Find $p_1, \dots, p_n \in \mathbb{R}^r$ such that $\left\{ \begin{array}{ll} \|p_i - p_j\|^2 = (D_p)_{ij}, & \forall (i,j) \in E \\ \|p_i - p_j\|^2 \geq R^2, & \forall (i,j) \notin E \end{array} \right\}$

Matrix Formulation

Find $P \in \mathbb{R}^{n \times r}$ such that $\left\{ \begin{array}{l} W \circ \mathcal{K}(Y) = D_p \\ H \circ \mathcal{K}(Y) \geq R^2 \end{array} \right\}$, where $Y = PP^T$

Semidefinite Programming (SDP) Relaxation

Find $Y \in \mathcal{S}^n_+ \cap \mathcal{S}_C$ such that $\left\{ \begin{array}{l} W \circ \mathcal{K}(Y) = D_p \\ H \circ \mathcal{K}(Y) \geq R^2 \end{array} \right\}$

- Vector/Matrix Formulation is non-convex and NP-HARD
- SDP Relaxation is convex, but degenerate (strict feasibility fails)

EDMs and Semidefinite Matrices

Vector Formulation

Find $p_1, \ldots, p_n \in \mathbb{R}^r$ such that $\left\{ \begin{array}{ll} \|p_i - p_j\|^2 = (D_p)_{ij}, & \forall (i,j) \in E \\ \|p_i - p_j\|^2 \geq R^2, & \forall (i,j) \notin E \end{array} \right\}$

Matrix Formulation

Find $P \in \mathbb{R}^{n \times r}$ such that $\left\{ \begin{array}{l} W \circ \mathcal{K}(Y) = D_p \\ H \circ \mathcal{K}(Y) \geq R^2 \end{array} \right\}$, where $Y = PP^T$

Semidefinite Programming (SDP) Relaxation

Find $Y \in \mathcal{S}^n_+ \cap \mathcal{S}_C$ such that $\left\{ \begin{array}{l} W \circ \mathcal{K}(Y) = D_p \\ H \circ \mathcal{K}(Y) \geq R^2 \end{array} \right\}$

- Vector/Matrix Formulation is non-convex and NP-HARD
- SDP Relaxation is convex, but degenerate (strict feasibility fails)

EDMs and Semidefinite Matrices

Vector Formulation

Find $p_1, \ldots, p_n \in \mathbb{R}^r$ such that $\left\{ \begin{array}{ll} \|p_i - p_j\|^2 = (D_p)_{ij}, & \forall (i,j) \in E \\ \|p_i - p_j\|^2 \geq R^2, & \forall (i,j) \notin E \end{array} \right\}$

Matrix Formulation

Find $P \in \mathbb{R}^{n \times r}$ such that $\left\{ \begin{array}{l} W \circ \mathcal{K}(Y) = D_p \\ H \circ \mathcal{K}(Y) \geq R^2 \end{array} \right\}$, where $Y = PP^T$

Semidefinite Programming (SDP) Relaxation

Find $Y \in \mathcal{S}_+^n \cap \mathcal{S}_C$ such that $\left\{ \begin{array}{l} W \circ \mathcal{K}(Y) = D_p \\ H \circ \mathcal{K}(Y) \geq R^2 \end{array} \right\}$

- Vector/Matrix Formulation is non-convex and NP-HARD
- SDP Relaxation is convex, but degenerate (strict feasibility fails)

EDMs and Semidefinite Matrices

Vector Formulation

Find $p_1, \ldots, p_n \in \mathbb{R}^r$ such that $\left\{ \begin{array}{ll} \|p_i - p_j\|^2 = (D_p)_{ij}, & \forall (i,j) \in E \\ \|p_i - p_j\|^2 \geq R^2, & \forall (i,j) \notin E \end{array} \right\}$

Matrix Formulation

Find $P \in \mathbb{R}^{n \times r}$ such that $\left\{ \begin{array}{l} W \circ \mathcal{K}(Y) = D_p \\ H \circ \mathcal{K}(Y) \geq R^2 \end{array} \right\}$, where $Y = PP^T$

Semidefinite Programming (SDP) Relaxation

Find $Y \in \mathcal{S}^n_+ \cap \mathcal{S}_C$ such that $\left\{ \begin{array}{l} W \circ \mathcal{K}(Y) = D_p \\ H \circ \mathcal{K}(Y) \geq R^2 \end{array} \right\}$

- Vector/Matrix Formulation is non-convex and NP-HARD
- SDP Relaxation is convex, but degenerate (strict feasibility fails)

Outline

1 Sensor Network Localization (SNL)
- Introduction
- Euclidean Distance Matrices and Semidefinite Matrices

2 Clique Reductions of SNL
- Clique Reductions
- Computing Sensor Positions

3 Algorithm
- Clique Unions and Node Absorptions
- Results

Clique Reductions

Theorem: Single Clique Reduction

Let:

- D_p be a partial EDM such that

$$D_p = \left[\begin{array}{c|c} \bar{D} & \cdot \\ \hline \cdot & \cdot \end{array}\right], \quad \text{for some } \bar{D} \in \mathcal{E}^k \text{ with embed. dim. } t \leq r$$

- $F := \left\{ Y \in \mathcal{S}^n_+ \cap \mathcal{S}_C : \mathcal{K}(Y[1:k]) = \bar{D} \right\}$ (contains SDP feas. set)

Then:

$$\boxed{\operatorname{face}(F) = \left(U \mathcal{S}_+^{n-k+t+1} U^T \right) \cap \mathcal{S}_C}$$

where $U := \left[\begin{array}{c|c} \bar{U} & 0 \\ \hline 0 & I_{n-k} \end{array}\right]$, $\bar{U} \in \mathbb{R}^{k \times t}$ eigenvectors of $B := \mathcal{K}^\dagger(\bar{D})$

Clique Reductions

Theorem: Single Clique Reduction

Let:

- D_p be a partial EDM such that

$$D_p = \left[\begin{array}{c|c} \bar{D} & \cdot \\ \hline \cdot & \cdot \end{array}\right], \quad \text{for some } \bar{D} \in \mathcal{E}^k \text{ with embed. dim. } t \leq r$$

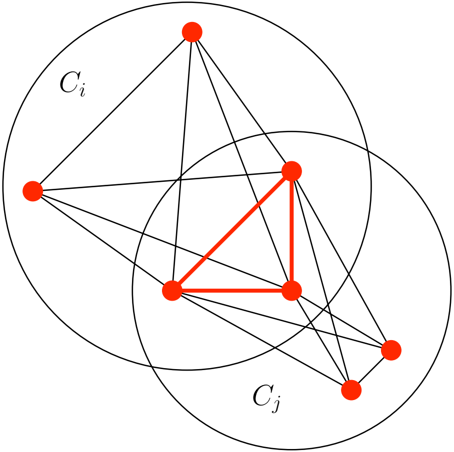
- $F := \left\{ Y \in \mathcal{S}^n_+ \cap \mathcal{S}_C : \mathcal{K}(Y[1:k]) = \bar{D} \right\}$ (contains SDP feas. set)

Then:

$$\boxed{\mathrm{face}(F) = \left(U \mathcal{S}^{n-k+t+1}_+ U^T \right) \cap \mathcal{S}_C}$$

where $U := \left[\begin{array}{c|c} \bar{U} & 0 \\ \hline 0 & I_{n-k} \end{array}\right]$, $\bar{U} \in \mathbb{R}^{k \times t}$ eigenvectors of $B := \mathcal{K}^\dagger(\bar{D})$

Clique Reductions



Clique Reductions

Theorem: Two Clique Reduction

Let $D \in \mathcal{E}^n$ with embed. dim. r. Let $\alpha_1, \alpha_2 \subseteq 1:n$ and $k := |\alpha_1 \cup \alpha_2|$.
For $i = 1, 2$ let:

- $t_i :=$ embed. dim. of $D[\alpha_i] \in \mathcal{E}^{k_i}$
- $F_i := \left\{ Y \in \mathcal{S}^n_+ \cap \mathcal{S}_C : \mathcal{K}(Y[\alpha_i]) = D[\alpha_i] \right\}$ (contains SDP feas. set)
- $\mathrm{face}(F_i) =: \left(U_i \mathcal{S}_+^{n-k_i+t_i+1} U_i^T \right) \cap \mathcal{S}_C$

Then:

$$\boxed{\mathrm{face}(F_1 \cap F_2) = \left(U \mathcal{S}_+^{n-k+t+1} U^T \right) \cap \mathcal{S}_C}$$

where $U \in \mathbb{R}^{n \times t}$ full column rank s.t. $\mathrm{col}(U) = \mathrm{col}(U_1) \cap \mathrm{col}(U_2)$

Clique Reductions

Theorem: Two Clique Reduction

Let $D \in \mathcal{E}^n$ with embed. dim. r. Let $\alpha_1, \alpha_2 \subseteq 1:n$ and $k := |\alpha_1 \cup \alpha_2|$.
For $i = 1, 2$ let:

- $t_i :=$ embed. dim. of $D[\alpha_i] \in \mathcal{E}^{k_i}$
- $F_i := \left\{ Y \in \mathcal{S}_+^n \cap \mathcal{S}_C : \mathcal{K}(Y[\alpha_i]) = D[\alpha_i] \right\}$ (contains SDP feas. set)
- $\operatorname{face}(F_i) =: \left(U_i \mathcal{S}_+^{n-k_i+t_i+1} U_i^T \right) \cap \mathcal{S}_C$

Then:

$$\boxed{\operatorname{face}(F_1 \cap F_2) = \left(U \mathcal{S}_+^{n-k+t+1} U^T \right) \cap \mathcal{S}_C}$$

where $U \in \mathbb{R}^{n \times t}$ full column rank s.t. $\operatorname{col}(U) = \operatorname{col}(U_1) \cap \operatorname{col}(U_2)$

Subspace Intersection for Two Intersecting Cliques

Suppose:

$$U_1 = \begin{bmatrix} U_1' & 0 \\ U_1'' & 0 \\ 0 & I \end{bmatrix} \quad \text{and} \quad U_2 = \begin{bmatrix} I & 0 \\ 0 & U_2'' \\ 0 & U_2' \end{bmatrix}$$

Then:

$$U := \begin{bmatrix} U_1' \\ U_1'' \\ U_2'(U_2'')^\dagger U_1'' \end{bmatrix} \quad \text{or} \quad U := \begin{bmatrix} U_1'(U_1'')^\dagger U_2'' \\ U_2'' \\ U_2' \end{bmatrix}$$

Satisfies:

$$\mathrm{col}(U) = \mathrm{col}(U_1) \cap \mathrm{col}(U_2)$$

Outline

1. Sensor Network Localization (SNL)
   - Introduction
   - Euclidean Distance Matrices and Semidefinite Matrices
2. Clique Reductions of SNL
   - Clique Reductions
   - Computing Sensor Positions
3. Algorithm
   - Clique Unions and Node Absorptions
   - Results

Computing Sensor Positions

Corollary: Computing Sensor Positions

Let:

- $D \in \mathcal{E}^n$ with embed. dim. r
- $D_p := W \circ D$ be a partial EDM (for some 0–1 matrix W)
- $F := \left\{ Y \in \mathcal{S}^n_+ \cap \mathcal{S}_C : W \circ \mathcal{K}(Y) = D_p \right\}$ and let $Y \in F$
- $\mathrm{face}(F) =: \left(U\mathcal{S}^{r+1}_+ U^T \right) \cap \mathcal{S}_C = (UV)\mathcal{S}^r_+(UV)^T$

If $D_p[\beta]$ is complete with embed. dim. r then:

- $\mathcal{K}(Y[\beta]) = D_p[\beta]$
- $Y = (UV)Z(UV)^T$, for some $Z \in \mathcal{S}^r_+$
- $(JU[\beta,:]V)Z(JU[\beta,:]V)^T = \mathcal{K}^\dagger(D_p[\beta])$ has a unique solution Z
- $D = \mathcal{K}\left(PP^T\right)$ where $\boxed{P := UVZ^{\frac{1}{2}} \in \mathbb{R}^{n\times r}}$

Computing Sensor Positions

Corollary: Computing Sensor Positions

Let:

- $D \in \mathcal{E}^n$ with embed. dim. r
- $D_p := W \circ D$ be a partial EDM (for some 0–1 matrix W)
- $F := \left\{ Y \in \mathcal{S}^n_+ \cap \mathcal{S}_C : W \circ \mathcal{K}(Y) = D_p \right\}$ and let $Y \in F$
- $\mathrm{face}(F) =: \left(U \mathcal{S}^{r+1}_+ U^T \right) \cap \mathcal{S}_C = (UV) \mathcal{S}^r_+ (UV)^T$

If $D_p[\beta]$ is complete with embed. dim. r then:

- $\mathcal{K}(Y[\beta]) = D_p[\beta]$
- $Y = (UV)Z(UV)^T$, for some $Z \in \mathcal{S}^r_+$
- $(JU[\beta,:]V)Z(JU[\beta,:]V)^T = \mathcal{K}^\dagger(D_p[\beta])$ has a unique solution Z
- $D = \mathcal{K}\left(PP^T\right)$ where $\boxed{P := UVZ^{\frac{1}{2}} \in \mathbb{R}^{n \times r}}$

Computing Sensor Positions

Corollary: Computing Sensor Positions

Let:

- $D \in \mathcal{E}^n$ with embed. dim. r
- $D_p := W \circ D$ be a partial EDM (for some 0–1 matrix W)
- $F := \left\{ Y \in \mathcal{S}_+^n \cap \mathcal{S}_C : W \circ \mathcal{K}(Y) = D_p \right\}$ and let $Y \in F$
- $\text{face}(F) =: \left(U \mathcal{S}_+^{r+1} U^T \right) \cap \mathcal{S}_C = (UV)\mathcal{S}_+^r (UV)^T$

If $D_p[\beta]$ is complete with embed. dim. r then:

- $\mathcal{K}(Y[\beta]) = D_p[\beta]$
- $Y = (UV)Z(UV)^T$, for some $Z \in \mathcal{S}_+^r$
- $(JU[\beta,:]V)Z(JU[\beta,:]V)^T = \mathcal{K}^\dagger(D_p[\beta])$ has a unique solution Z
- $D = \mathcal{K}\left(PP^T\right)$ where $\boxed{P := UVZ^{\frac{1}{2}} \in \mathbb{R}^{n \times r}}$

Computing Sensor Positions

Corollary: Computing Sensor Positions

Let:

- $D \in \mathcal{E}^n$ with embed. dim. r
- $D_p := W \circ D$ be a partial EDM (for some 0–1 matrix W)
- $F := \left\{ Y \in \mathcal{S}_+^n \cap \mathcal{S}_C : W \circ \mathcal{K}(Y) = D_p \right\}$ and let $Y \in F$
- $\mathrm{face}(F) =: \left(U \mathcal{S}_+^{r+1} U^T \right) \cap \mathcal{S}_C = (UV)\mathcal{S}_+^r (UV)^T$

If $D_p[\beta]$ is complete with embed. dim. r then:

- $\mathcal{K}(Y[\beta]) = D_p[\beta]$
- $Y = (UV)Z(UV)^T$, for some $Z \in \mathcal{S}_+^r$
- $(JU[\beta,:]V)Z(JU[\beta,:]V)^T = \mathcal{K}^\dagger(D_p[\beta])$ has a unique solution Z
- $D = \mathcal{K}\left(PP^T\right)$ where $\boxed{P := UVZ^{\frac{1}{2}} \in \mathbb{R}^{n \times r}}$

Computing Sensor Positions

Corollary: Computing Sensor Positions

Let:

- $D \in \mathcal{E}^n$ with embed. dim. r
- $D_p := W \circ D$ be a partial EDM (for some 0–1 matrix W)
- $F := \left\{ Y \in \mathcal{S}^n_+ \cap \mathcal{S}_C : W \circ \mathcal{K}(Y) = D_p \right\}$ and let $Y \in F$
- $\mathrm{face}(F) =: \left(U \mathcal{S}^{r+1}_+ U^T \right) \cap \mathcal{S}_C = (UV)\mathcal{S}^r_+(UV)^T$

If $D_p[\beta]$ is complete with embed. dim. r then:

- $\mathcal{K}(Y[\beta]) = D_p[\beta]$
- $Y = (UV)Z(UV)^T$, for some $Z \in \mathcal{S}^r_+$
- $(JU[\beta,:]V)Z(JU[\beta,:]V)^T = \mathcal{K}^\dagger(D_p[\beta])$ has a unique solution Z
- $D = \mathcal{K}\left(PP^T\right)$ where $\boxed{P := UVZ^{\frac{1}{2}} \in \mathbb{R}^{n \times r}}$

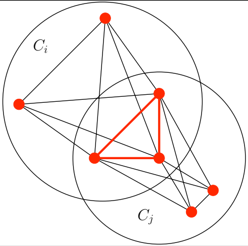
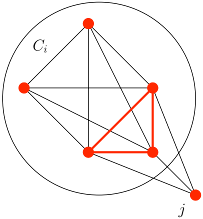
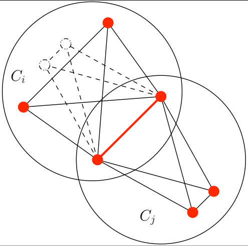
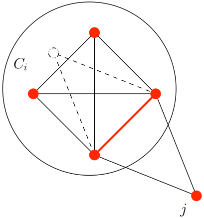
Outline

1 Sensor Network Localization (SNL)
- Introduction
- Euclidean Distance Matrices and Semidefinite Matrices

2 Clique Reductions of SNL
- Clique Reductions
- Computing Sensor Positions

3 Algorithm
- Clique Unions and Node Absorptions
- Results

Algorithm

	Clique Union	Node Absorption
Rigid		
Non-rigid		

Algorithm

Initialize

$$C_i := \left\{ j : (D_p)_{ij} < (R/2)^2 \right\}, \quad \text{for } i = 1, \ldots, n$$

Iterate

- For $|C_i \cap C_j| \geq r + 1$, do Rigid Clique Union
- For $|C_i \cap \mathcal{N}(j)| \geq r + 1$, do Rigid Node Absorption
- For $|C_i \cap C_j| = r$, do Non-Rigid Clique Union (lower bounds)
- For $|C_i \cap \mathcal{N}(j)| = r$, do Non-Rigid Node Absorption (lower bounds)

Finalize

When $\exists$ a clique containing all the anchors, use the computed facial representation and the positions of the anchors to locate the sensors

Algorithm

Initialize

$$C_i := \left\{ j : (D_p)_{ij} < (R/2)^2 \right\}, \quad \text{for } i = 1, \ldots, n$$

Iterate

- For $|C_i \cap C_j| \geq r + 1$, do Rigid Clique Union
- For $|C_i \cap \mathcal{N}(j)| \geq r + 1$, do Rigid Node Absorption
- For $|C_i \cap C_j| = r$, do Non-Rigid Clique Union (lower bounds)
- For $|C_i \cap \mathcal{N}(j)| = r$, do Non-Rigid Node Absorption (lower bounds)

Finalize

When $\exists$ a clique containing all the anchors, use the computed facial representation and the positions of the anchors to locate the sensors

Algorithm

Initialize

$$C_i := \left\{ j : (D_p)_{ij} < (R/2)^2 \right\}, \quad \text{for } i = 1, \ldots, n$$

Iterate

- For $|C_i \cap C_j| \geq r + 1$, do Rigid Clique Union
- For $|C_i \cap \mathcal{N}(j)| \geq r + 1$, do Rigid Node Absorption
- For $|C_i \cap C_j| = r$, do Non-Rigid Clique Union (lower bounds)
- For $|C_i \cap \mathcal{N}(j)| = r$, do Non-Rigid Node Absorption (lower bounds)

Finalize

When $\exists$ a clique containing all the anchors, use the computed facial representation and the positions of the anchors to locate the sensors

Outline

1. Sensor Network Localization (SNL)
   - Introduction
   - Euclidean Distance Matrices and Semidefinite Matrices
2. Clique Reductions of SNL
   - Clique Reductions
   - Computing Sensor Positions
3. Algorithm
   - Clique Unions and Node Absorptions
   - Results

Results

- Random *noiseless* problems
- Dimension $r = 2$
- Square region: $[0, 1] \times [0, 1]$
- $m = 9$ anchors
- Using only Rigid Clique Union and Rigid Node Absorption
- Error measure: Root Mean Square Deviation

$$\text{RMSD} = \left(\frac{1}{n} \sum_{i=1}^{n} \| p_i - p_i^{\text{true}} \|^2 \right)^{1/2}$$

Results

of Sensors Located

# sensors \ R	0.07	0.06	0.05	0.04
2000	2000	2000	1956	1375
6000	6000	6000	6000	6000
10000	10000	10000	10000	10000

CPU Seconds

# sensors \ R	0.07	0.06	0.05	0.04
2000	1	1	1	3
6000	6	5	5	5
10000	16	13	12	12

RMSD (over located sensors)

# sensors \ R	0.07	0.06	0.05	0.04
2000	$4e{-16}$	$9e{-16}$	$4e{-16}$	$4e{-16}$
6000	$6e{-16}$	$4e{-16}$	$3e{-16}$	$6e{-16}$
10000	$4e{-16}$	$4e{-16}$	$6e{-16}$	$6e{-16}$

Results

of Sensors Located

# sensors \ R	0.07	0.06	0.05	0.04
2000	2000	2000	1956	1375
6000	6000	6000	6000	6000
10000	10000	10000	10000	10000

CPU Seconds

# sensors \ R	0.07	0.06	0.05	0.04
2000	1	1	1	3
6000	6	5	5	5
10000	16	13	12	12

RMSD (over located sensors)

# sensors \ R	0.07	0.06	0.05	0.04
2000	$4e{-}16$	$9e{-}16$	$4e{-}16$	$4e{-}16$
6000	$6e{-}16$	$4e{-}16$	$3e{-}16$	$6e{-}16$
10000	$4e{-}16$	$4e{-}16$	$6e{-}16$	$6e{-}16$

Results

of Sensors Located

# sensors \ R	0.07	0.06	0.05	0.04
2000	2000	2000	1956	1375
6000	6000	6000	6000	6000
10000	10000	10000	10000	10000

CPU Seconds

# sensors \ R	0.07	0.06	0.05	0.04
2000	1	1	1	3
6000	6	5	5	5
10000	16	13	12	12

RMSD (over located sensors)

# sensors \ R	0.07	0.06	0.05	0.04
2000	$4e{-}16$	$9e{-}16$	$4e{-}16$	$4e{-}16$
6000	$6e{-}16$	$4e{-}16$	$3e{-}16$	$6e{-}16$
10000	$4e{-}16$	$4e{-}16$	$6e{-}16$	$6e{-}16$

Results

Large-Scale Problems

# sensors	# anchors	radio range	RMSD	Time
20000	9	.02	$5e{-}16$	35s
40000	9	.015	$7e{-}16$	2m 15s
60000	9	.01	$1e{-}15$	5m 21s
100000	9	.01	$8e{-}16$	14m 14s

Summary

- SDP relaxation of SNL is highly degenerate: The feasible set of this SDP is restricted to a low dimensional face of the SDP cone, causing the Slater constraint qualification (strict feasibility) to fail
- We take advantage of this degeneracy by finding explicit representations of the faces of the SDP cone corresponding to unions of intersecting cliques
- Without using an SDP-solver (eg. SeDuMi, SDPA, SDPT3), we quickly compute the exact solution to the large SDP relaxations

Summary

- SDP relaxation of SNL is highly degenerate: The feasible set of this SDP is restricted to a low dimensional face of the SDP cone, causing the Slater constraint qualification (strict feasibility) to fail
- We take advantage of this degeneracy by finding explicit representations of the faces of the SDP cone corresponding to unions of intersecting cliques
- Without using an SDP-solver (eg. SeDuMi, SDPA, SDPT3), we quickly compute the exact solution to the large SDP relaxations

Summary

- SDP relaxation of SNL is highly degenerate: The feasible set of this SDP is restricted to a low dimensional face of the SDP cone, causing the Slater constraint qualification (strict feasibility) to fail
- We take advantage of this degeneracy by finding explicit representations of the faces of the SDP cone corresponding to unions of intersecting cliques
- Without using an SDP-solver (eg. SeDuMi, SDPA, SDPT3), we quickly compute the exact solution to the large SDP relaxations

Thank you!