Exploiting Sparsity in Sensor Network Localization Problem with the framework of SDP relaxation

Makoto Yamashita [1], Masakazu Kojima [1],
Sunyoung Kim [2], Hayato Waki [3]

[1]Tokyo Institute of Technology,
[2]Ewha Women's university, [3]The University of Electro-Communications

2011.04.13
@ National Cheng Kung University, Taiwan

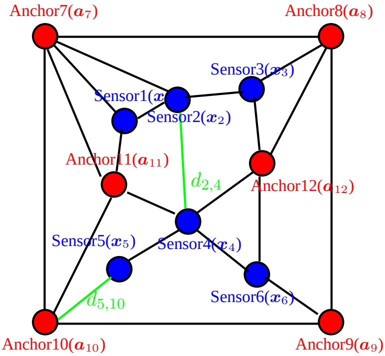
Outline

1. Sensor Network Localization (SNL) Problem has many applications, *e.g.* building check, protein conformation.
2. SNL is NP-complete in general.
3. Biswas-Ye introduced Semidefinite Programming(SDP) relaxation to obtain a good approximation for small SNLs.
4. To solve large-scale SNLs (*i.e.* many sensors) without losing accuracy, we exploit the network sparsity.
5. SFSDP can solve sensor network localization problem with more than 10,000 sensors.

http://www.is.titech.ac.jp/~kojima/SFSDP/SFSDP.html

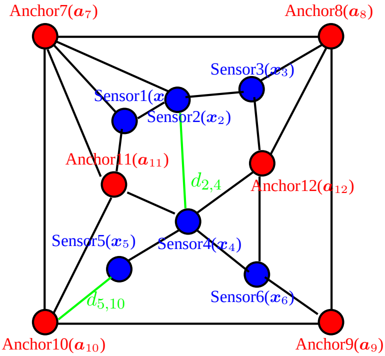
Sensor Network

- In 2D square ($[0,1] \times [0,1]$), anchors and sensors are scattered.



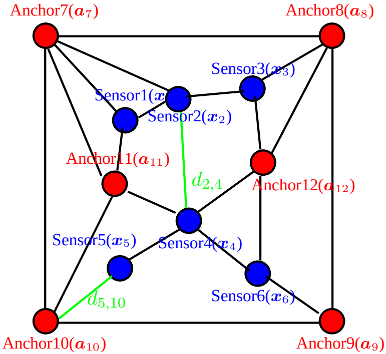
Sensor Network

- In 2D square ($[0,1] \times [0,1]$), anchors and sensors are scattered.
- The exact positions of anchors $\boldsymbol{a}_7, \ldots, \boldsymbol{a}_{12} \in \mathbb{R}^2$ are *known*.
- The exact positions of sensors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_6 \in \mathbb{R}^2$ are *unknown*.



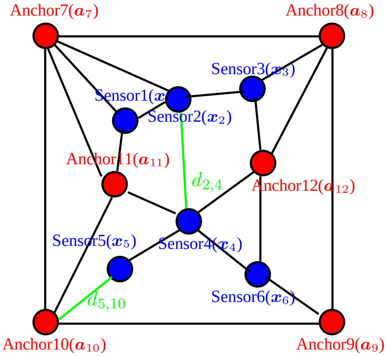
Sensor Network

- In 2D square $([0,1] \times [0,1])$, anchors and sensors are scattered.
- The exact positions of anchors $\boldsymbol{a}_7, \ldots, \boldsymbol{a}_{12} \in \mathbb{R}^2$ are *known*.
- The exact positions of sensors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_6 \in \mathbb{R}^2$ are *unknown*.
- If we know $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_6 \in \mathbb{R}^2$, it is easy to compute the distances, *e.g.* $d_{2,4} = ||\boldsymbol{x}_2 - \boldsymbol{x}_4||$, $d_{5,10} = ||\boldsymbol{x}_5 - \boldsymbol{a}_{10}||$.
  $||\boldsymbol{x}_2 - \boldsymbol{x}_4|| = \sqrt{(x_{21} - x_{41})^2 + (x_{22} - x_{42})^2)}$



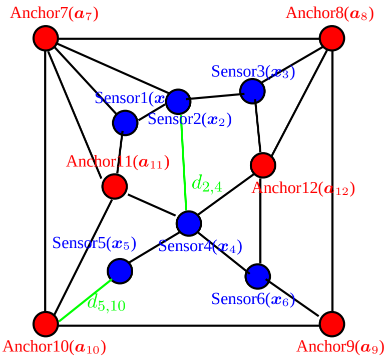
Sensor Network

- In 2D square $([0,1] \times [0,1])$, anchors and sensors are scattered.
- The exact positions of anchors $\boldsymbol{a}_7, \ldots, \boldsymbol{a}_{12} \in \mathbb{R}^2$ are *known*.
- The exact positions of sensors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_6 \in \mathbb{R}^2$ are *unknown*.
- If we know $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_6 \in \mathbb{R}^2$, it is easy to compute the distances, *e.g.* $d_{2,4} = ||\boldsymbol{x}_2 - \boldsymbol{x}_4||$, $d_{5,10} = ||\boldsymbol{x}_5 - \boldsymbol{a}_{10}||$. $||\boldsymbol{x}_2 - \boldsymbol{x}_4|| = \sqrt{(x_{21} - x_{41})^2 + (x_{22} - x_{42})^2)}$
- The reverse is SNL.



Sensor Network

- In 2D square $([0,1] \times [0,1])$, anchors and sensors are scattered.
- The exact positions of anchors $\boldsymbol{a}_7, \ldots, \boldsymbol{a}_{12} \in \mathbb{R}^2$ are *known*.
- The exact positions of sensors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_6 \in \mathbb{R}^2$ are *unknown*.
- If we know $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_6 \in \mathbb{R}^2$, it is easy to compute the distances, *e.g.* $d_{2,4} = ||\boldsymbol{x}_2 - \boldsymbol{x}_4||$, $d_{5,10} = ||\boldsymbol{x}_5 - \boldsymbol{a}_{10}||$. $||\boldsymbol{x}_2 - \boldsymbol{x}_4|| = \sqrt{(x_{21} - x_{41})^2 + (x_{22} - x_{42})^2)}$
- The reverse is SNL.
- Given distance, compute the locations.
- However, the reverse is difficult (NP-complete).



Example of SNL computation

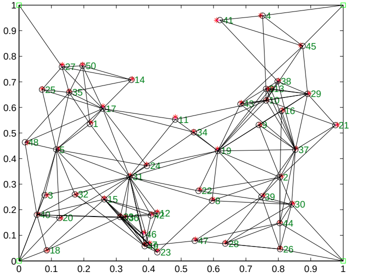


Figure: Input

50 Sensors, 4 Anchors(at corners), Radiorange = 0.3, Noise = 0.02,

Example of SNL computation

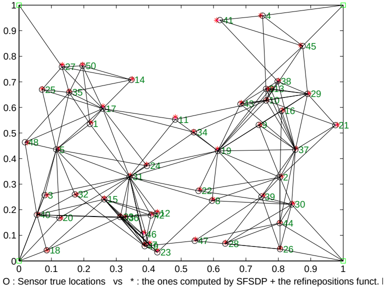


Figure: Input

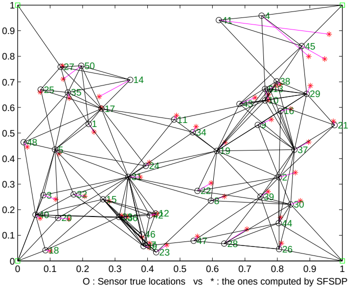
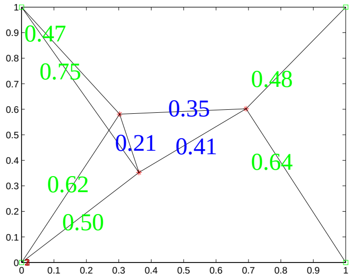


Figure: Output

50 Sensors, 4 Anchors(at corners), Radiorange = 0.3, Noise = 0.02,

Usage of SFSDP

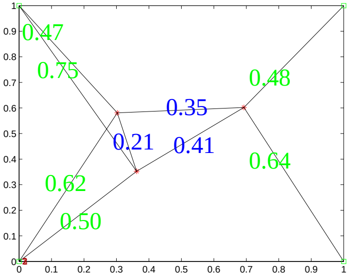


* : the sensor locations computed by SFSDP + the refinepositions funct. by K. Toh

Input: sDim $= 2$, noOfSensors $= 3$, noOfAnchors $= 4$, xMatrix0 $= \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}$,

$$\text{ditanceMatrix0} = \left(\begin{array}{ccc|cccc} 0 & 0.21 & 0.41 & 0.50 & 0.75 & 0 & 0 \\ 0 & 0 & 0.35 & 0.62 & 0.47 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.64 & 0.48 \end{array}\right)$$

Usage of SFSDP



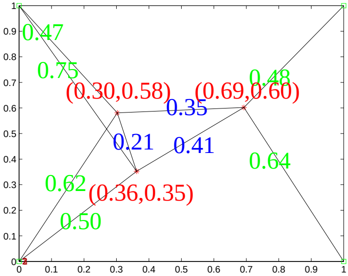
* : the sensor locations computed by SFSDP + the refinepositions funct. by K. Toh

Input: sDim = 2, noOfSensors = 3, noOfAnchors = 4, xMatrix0 = $\begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}$,

ditanceMatrix0 = $\left(\begin{array}{ccc|cccc} 0 & 0.21 & 0.41 & 0.50 & 0.75 & 0 & 0 \\ 0 & 0 & 0.35 & 0.62 & 0.47 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.64 & 0.48 \end{array}\right)$

Execute: [xMatrix] =
SFSDPplus(sDim,noOfSensors,noOfAnchors,xMatrix0,distanceMatrix0,pars);

Usage of SFSDP



* : the sensor locations computed by SFSDP + the refinepositions funct. by K. Toh

Input: sDim $= 2$, noOfSensors $= 3$, noOfAnchors $= 4$, xMatrix0 $= \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}$,

$$\text{ditanceMatrix0} = \left(\begin{array}{ccc|cccc} 0 & 0.21 & 0.41 & 0.50 & 0.75 & 0 & 0 \\ 0 & 0 & 0.35 & 0.62 & 0.47 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.64 & 0.48 \end{array}\right)$$

Execute: [xMatrix] = SFSDPplus(sDim,noOfSensors,noOfAnchors,xMatrix0,distanceMatrix0,pars);

Output: xMatrix(:,noOfSensors) $= \begin{pmatrix} 0.36 & 0.30 & 0.69 \\ 0.35 & 0.58 & 0.60 \end{pmatrix}$

Applications of SNL

There are many applications of SNL.

- Follow-up studies of wild animals
- Structural investigations
  - High buildings after long years.
  - Subway tunnel in U.K.
- Molecule and protein conformation (in 3D and noisy input)

Applications of SNL

There are many applications of SNL.

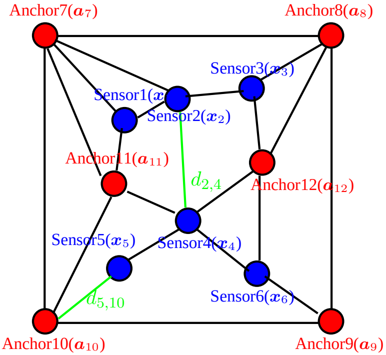
- Follow-up studies of wild animals
- Structural investigations
  - High buildings after long years.
  - Subway tunnel in U.K.
- Molecule and protein conformation (in 3D and noisy input)

Various approaches:

- Multi Dimensional Scaling (a method like principal component analysis)
- SDP relaxation
- Divide and conquer
- Shortest-path approach (onging work)

Basic Formulation

Given edge sets $\mathcal{N}_x$ and $\mathcal{N}_a$, distances $d_{pq}^2((p,q) \in \mathcal{N}_x)$ and $d_{pr}^2((p,r) \in \mathcal{N}_a)$, and anchor locations $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m \in \mathbb{R}^d$,

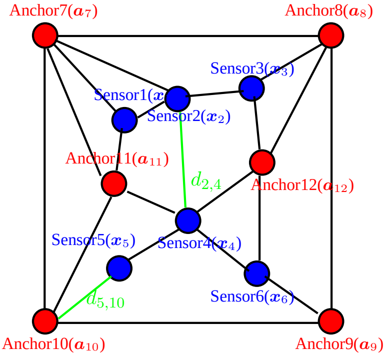


Basic Formulation

Given edge sets $\mathcal{N}_x$ and $\mathcal{N}_a$, distances $d_{pq}^2((p,q) \in \mathcal{N}_x)$ and $d_{pr}^2((p,r) \in \mathcal{N}_a)$, and anchor locations $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m \in \mathbb{R}^d$,

Find solution $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$

$$\begin{cases} \|x_p - x_q\|^2 = d_{pq}^2 & (p,q) \in \mathcal{N}_x \\ \|x_p - a_r\|^2 = d_{pr}^2 & (p,r) \in \mathcal{N}_a \end{cases}$$



Basic Formulation

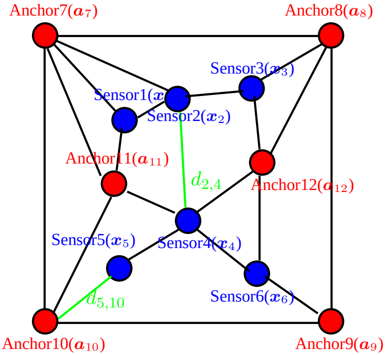
Given edge sets $\mathcal{N}_x$ and $\mathcal{N}_a$, distances $d_{pq}^2((p,q) \in \mathcal{N}_x)$ and $d_{pr}^2((p,r) \in \mathcal{N}_a)$, and anchor locations $\boldsymbol{a}_1, \dots, \boldsymbol{a}_m \in \mathbb{R}^d$,

Find solution $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n \in \mathbb{R}^d$

$$\begin{cases} ||x_p - x_q||^2 = d_{pq}^2 & (p,q) \in \mathcal{N}_x \\ ||x_p - a_r||^2 = d_{pr}^2 & (p,r) \in \mathcal{N}_a \end{cases}$$

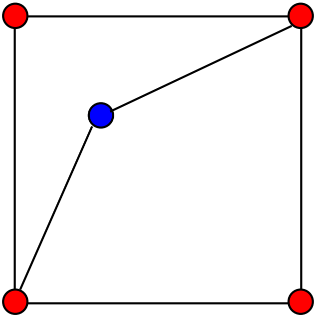
For $\mathcal{N}_x$ and $\mathcal{N}_a$, the *radio range* ρ is usually used.

$$\begin{aligned} \mathcal{N}_x^\rho &= \{(p,q) : ||x_p - x_q|| \le \rho\} \\ \mathcal{N}_a^\rho &= \{(p,r) : ||x_p - a_r|| \le \rho\} \end{aligned}$$



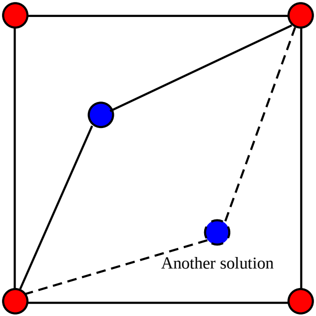
Why NP-complete?

- Solutions may NOT be unique. (Late, we will discuss when SNL has unique solution.)



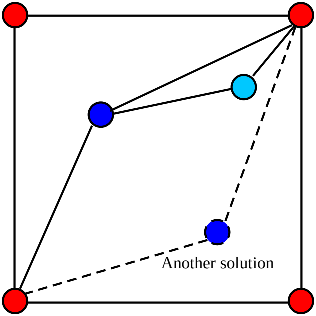
Why NP-complete?

- Solutions may NOT be unique. (Late, we will discuss when SNL has unique solution.)



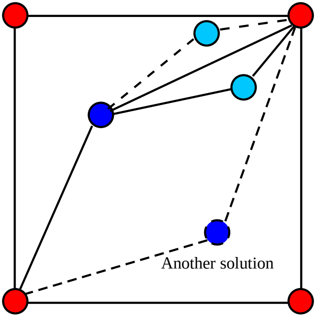
Why NP-complete?

- Solutions may NOT be unique. (Late, we will discuss when SNL has unique solution.)



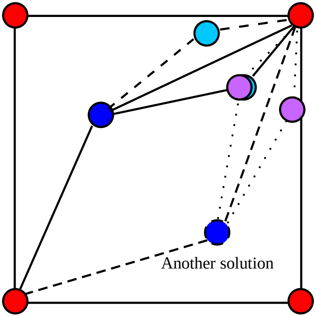
Why NP-complete?

- Solutions may NOT be unique. (Late, we will discuss when SNL has unique solution.)



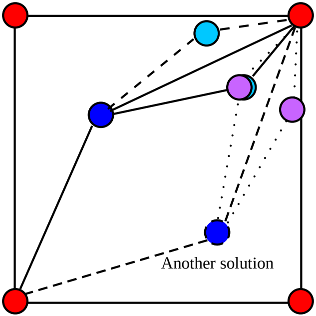
Why NP-complete?

- Solutions may NOT be unique. (Late, we will discuss when SNL has unique solution.)



Why NP-complete?

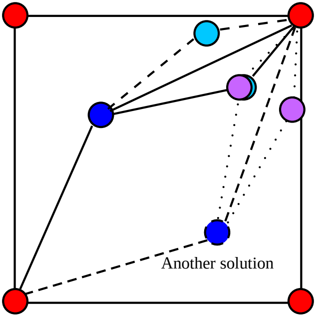
- Solutions may NOT be unique. (Late, we will discuss when SNL has unique solution.)
- In right figure, at least $O(2^n)$ solutions.
- The number of feasible solutions can be large in *exponential* order.



Why NP-complete?

- Solutions may NOT be unique. (Late, we will discuss when SNL has unique solution.)
- In right figure, at least $O(2^n)$ solutions.
- The number of feasible solutions can be large in *exponential* order.

Enumerating all the solutions are im-possible for large number of sensors.



Nonlinear Optimization

SNL

$$\begin{cases} ||x_p - x_q||^2 = d_{pq}^2 & \forall (p,q) \in \mathcal{N}_x \\ ||x_p - a_r||^2 = d_{pr}^2 & \forall (p,r) \in \mathcal{N}_a \end{cases}$$

Nonlinear Optimization

SNL

$$\begin{cases} ||x_p - x_q||^2 = d_{pq}^2 & \forall (p,q) \in \mathcal{N}_x \\ ||x_p - a_r||^2 = d_{pr}^2 & \forall (p,r) \in \mathcal{N}_a \end{cases}$$

can be formulated as

Nonlinear Optimization Formulation

$$\min \quad : \quad \sum_{(p,q)\in\mathcal{N}_x} \left| ||x_p - x_q||^2 - d_{pq}^2 \right| + \sum_{(p,r)\in\mathcal{N}_a} \left| ||x_p - a_r||^2 - d_{pr}^2 \right|$$

- This objective function is nonconvex. (Multiple solutions.)
- At feasible solution of SNL, the objective value is *zero*.

Noisy Input

- Some input distances in real applications may have small noise.
- There may be no feasible points.

$$\begin{cases} ||x_p - x_q||^2 = d_{pq}^2 & \forall (p,q) \in \mathcal{N}_x \\ ||x_p - a_r||^2 = d_{pr}^2 & \forall (p,r) \in \mathcal{N}_a \end{cases}$$

Noisy Input

- Some input distances in real applications may have small noise.
- There may be no feasible points.

$$\begin{cases} \|x_p - x_q\|^2 = d_{pq}^2 & \forall (p,q) \in \mathcal{N}_x \\ \|x_p - a_r\|^2 = d_{pr}^2 & \forall (p,r) \in \mathcal{N}_a \end{cases}$$

- Instead, minimize the differences

$$\min \quad : \quad \sum_{(p,q)\in\mathcal{N}_x} \left| \|x_p - x_q\|^2 - d_{pq}^2 \right| + \sum_{(p,r)\in\mathcal{N}_a} \left| \|x_p - a_r\|^2 - d_{pr}^2 \right|$$

2. SDP relaxations

- Standard form of SDP
- Goemans-Williamson's relaxation for Max-cut problems
- Biswas-Ye's relaxation for SNL

Standard form of SDP

Standard form of SDP has primal-dual from.

$$\text{(Primal)} \quad : \quad \min \boldsymbol{C} \bullet \boldsymbol{X} \quad \text{s.t.} \boldsymbol{A}_k \bullet \boldsymbol{X} = b_k (k = 1, \ldots, m), \boldsymbol{X} \succeq \boldsymbol{O}$$

$$\text{(Dual)} \quad : \quad \max \sum_{k=1}^{m} b_k z_k \quad \text{s.t.} \sum_{k=1}^{m} \boldsymbol{A}_k z_k + \boldsymbol{Y} = \boldsymbol{C}, \boldsymbol{Y} \succeq \boldsymbol{O}$$

Notations:

- $\mathbb{S}^n$: the space of $n \times n$ symmetric matrices
- $\mathbb{S}^n_+$: the space of $n \times n$ positive semidefinite symmetric matrices
- $\boldsymbol{X} \succeq \boldsymbol{O}$: $\boldsymbol{X} \in \mathbb{S}^n_+$
  all the eigenvalues of $\boldsymbol{X}$ are non-negative
  $\Leftrightarrow \quad \boldsymbol{u}^T \boldsymbol{X} \boldsymbol{u} \geq 0 \quad \text{for } \forall \boldsymbol{u} \in \mathbb{R}^n.$
- $\boldsymbol{X} \bullet \boldsymbol{Y}$: Inner product in $\mathbb{S}^n$, $\boldsymbol{X} \bullet \boldsymbol{Y} = \operatorname{tr}(\boldsymbol{X}^T \boldsymbol{Y}) = \sum_{i=1}^{n} \sum_{j=1}^{n} X_{ij} Y_{ij}$

Characteristics of SDP

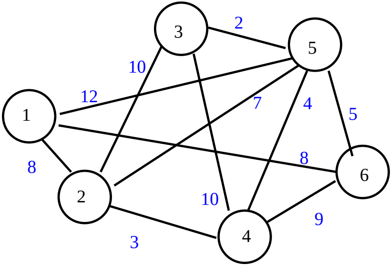
- Extension of Linear Programs to Hilbert space
- Solvable in polynomial time (owing to Primal-Dual Interior-Point Methods)
- Eigenvalue is restricted to non-negative

Max-cut problem

Goemans and Williams first applied SDP to Max-cut problems.
This opened the new researches called SDP relaxations.

Max-cut problem

Given a graph $G(V, W)$ (V is the vertex set, W is the edge weights), find the cut with the maximum weight.



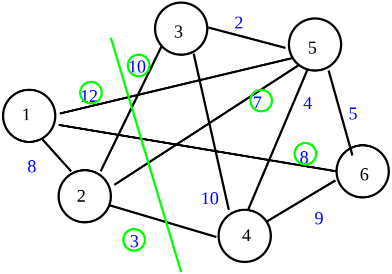
$V = \{1, 2, 3, 4, 5, 6\}$

Max-cut problem

Goemans and Williams first applied SDP to Max-cut problems.
This opened the new researches called SDP relaxations.

Max-cut problem

Given a graph $G(V, W)$ (V is the vertex set, W is the edge weights), find the cut with the maximum weight.



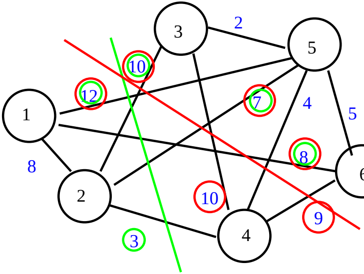
$V = \{1, 2, 3, 4, 5, 6\}$
$Cut(\{1, 2\}, \{3, 4, 5, 6\}) = 12 + 10 + 7 + 8 + 3 = 40$

Max-cut problem

Goemans and Williams first applied SDP to Max-cut problems.
This opened the new researches called SDP relaxations.

Max-cut problem

Given a graph $G(V, W)$ (V is the vertex set, W is the edge weights), find the cut with the maximum weight.



$V = \{1, 2, 3, 4, 5, 6\}$

$Cut(\{1, 2\}, \{3, 4, 5, 6\}) = 12 + 10 + 7 + 8 + 3 = 40$

$Cut(\{1, 2, 4\}, \{3, 5, 6\}) = 12 + 10 + 10 + 7 + 8 + 9 = 56$ (Max-cut)

Mathematical formulation of Max-cut

A cut separates the vertex set $V = \{v_1, \ldots, v_n\}$ to two disjoint sets Left L and Right R.

$$V = L \cup R \qquad L \cap R = \emptyset$$

Mathematical formulation of Max-cut

A cut separates the vertex set $V = \{v_1, \ldots, v_n\}$ to two disjoint sets Left L and Right R.

$$V = L \cup R \qquad L \cap R = \emptyset$$

The weight of a cut is

$$W(L, R) = \sum_{v_i \in L, v_j \in R} w_{ij} + \sum_{v_i \in R, v_j \in L} w_{ij}.$$

Mathematical formulation of Max-cut

A cut separates the vertex set $V = \{v_1, \dots, v_n\}$
to two disjoint sets Left L and Right R.

$$V = L \cup R \qquad L \cap R = \emptyset$$

The weight of a cut is

$$W(L,R) = \sum_{v_i \in L, v_j \in R} w_{ij} + \sum_{v_i \in R, v_j \in L} w_{ij}.$$

They introduced the variables $x_1, \dots, x_n$

$$x_i = 1 \text{ if } v_i \in L, \qquad x_i = -1 \text{ if } v_i \in R.$$

Mathematical formulation of Max-cut

A cut separates the vertex set $V = \{v_1, \ldots, v_n\}$
to two disjoint sets Left L and Right R.

$$V = L \cup R \qquad L \cap R = \emptyset$$

The weight of a cut is

$$W(L, R) = \sum_{v_i \in L, v_j \in R} w_{ij} + \sum_{v_i \in R, v_j \in L} w_{ij}.$$

They introduced the variables $x_1, \ldots, x_n$

$$x_i = 1 \text{ if } v_i \in L, \qquad x_i = -1 \text{ if } v_i \in R.$$

then

$$1 - x_i x_j \quad = \quad \begin{cases} 2 & \text{if } [v_i \in L, v_j \in R] \text{ or } [v_j \in L, v_i \in R] \\ 0 & \text{otherwise} \end{cases}$$

Mathematical formulation of Max-cut

A cut separates the vertex set $V = \{v_1, \ldots, v_n\}$ to two disjoint sets Left L and Right R.

$$V = L \cup R \qquad L \cap R = \emptyset$$

The weight of a cut is

$$W(L,R) = \sum_{v_i \in L, v_j \in R} w_{ij} + \sum_{v_i \in R, v_j \in L} w_{ij}.$$

They introduced the variables $x_1, \ldots, x_n$

$$x_i = 1 \text{ if } v_i \in L, \qquad x_i = -1 \text{ if } v_i \in R.$$

then

$$1 - x_i x_j \quad = \quad \begin{cases} 2 & \text{if } [v_i \in L, v_j \in R] \text{ or } [v_j \in L, v_i \in R] \\ 0 & \text{otherwise} \end{cases}$$

Then the Max-cut problem can be expressed as

$$\max \ W(L,R) = \frac{1}{4} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij}(1 - x_i x_j) \qquad \text{s.t.} \quad x_i \in \{-1, 1\} \ (i = 1, \ldots, n)$$

This problem is also NP-hard (The number of feasible solution is 2^n).

Some matrices

They introduce the matrix $\boldsymbol{X} \in \mathbb{S}^n$ by $\boldsymbol{X} = \boldsymbol{x}\boldsymbol{x}^T$, that is, $X_{ij} = x_i x_j$.
Then $X_{ii} = 1$, because $x_i \in \{-1, 1\}$.

Some matrices

They introduce the matrix $\boldsymbol{X} \in \mathbb{S}^n$ by $\boldsymbol{X} = \boldsymbol{x}\boldsymbol{x}^T$, that is, $X_{ij} = x_i x_j$.
Then $X_{ii} = 1$, because $x_i \in \{-1, 1\}$.
In addition, we define the matrix $\boldsymbol{C}$ by

$$C_{ij} = \begin{cases} \frac{1}{4}\left(\sum_{k=1}^{n} w_{ik} - w_{ii}\right) & \text{if } i = j \\ \frac{1}{4}\left(-w_{ij}\right) & \text{if } i \neq j \end{cases}$$

Some matrices

They introduce the matrix $\boldsymbol{X} \in \mathbb{S}^n$ by $\boldsymbol{X} = \boldsymbol{x}\boldsymbol{x}^T$, that is, $X_{ij} = x_i x_j$.
Then $X_{ii} = 1$, because $x_i \in \{-1, 1\}$.
In addition, we define the matrix $\boldsymbol{C}$ by

$$C_{ij} = \begin{cases} \frac{1}{4}\left(\sum_{k=1}^n w_{ik} - w_{ii}\right) & \text{if } i = j \\ \frac{1}{4}\left(-w_{ij}\right) & \text{if } i \neq j \end{cases}$$

Then

$$\begin{aligned} \boldsymbol{C} \bullet \boldsymbol{X} &= \sum_{i=1}^n \sum_{j=1}^n C_{ij} X_{ij} = \sum_{i=1}^n C_{ii} X_{ii} + \sum_{i=1}^n \sum_{j \neq i} C_{ij} X_{ij} \\ &= \frac{1}{4} \sum_{i=1}^n \left(\sum_{k=1}^n w_{ik} - w_{ii} \right) X_{ii} - \frac{1}{4} \sum_{i=1}^n \sum_{j \neq i} w_{ij} X_{ij} \\ &= \frac{1}{4} \sum_{i=1}^n \sum_{k=1}^n w_{ik} X_{ii} - \frac{1}{4} \sum_{i=1}^n \sum_{j=1}^n w_{ij} X_{ij} \\ &= \frac{1}{4} \sum_{i=1}^n \sum_{j=1}^n w_{ij} (1 - x_i x_j) = W(L, R) \end{aligned}$$

Matrix Form

Rank-1 property

We can recover the condition $x_i \in \{-1, 1\}$ $(i = 1, \ldots, n)$ if

$$\boldsymbol{X} \succeq \boldsymbol{O}, X_{ii} = 1 \ (i = 1, \ldots, n), \mathrm{rank}(\boldsymbol{X}) = 1$$

by decomposing $\boldsymbol{X} = \boldsymbol{x}\boldsymbol{x}^T$.

Matrix Form

Rank-1 property

We can recover the condition $x_i \in \{-1, 1\}$ $(i = 1, \dots, n)$ if

$$\boldsymbol{X} \succeq \boldsymbol{O}, X_{ii} = 1 \ (i = 1, \dots, n), \mathrm{rank}(\boldsymbol{X}) = 1$$

by decomposing $\boldsymbol{X} = \boldsymbol{x}\boldsymbol{x}^T$.

Therefore

$$\max \ W(L, R) = \frac{1}{4} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij}(1 - x_i x_j) \qquad \text{s.t.} \quad x_i \in \{-1, 1\} \ (i = 1, \dots, n)$$

is equivalent to

$$\max \ W(L, R) = \boldsymbol{C} \bullet \boldsymbol{X} \qquad \text{s.t.} \quad \boldsymbol{X} \succeq \boldsymbol{O}, X_{ii} = 1 \ (i = 1, \dots, n), \mathrm{rank}(\boldsymbol{X}) = 1$$

Matrix Form

Rank-1 property

We can recover the condition $x_i \in \{-1, 1\}$ $(i = 1, \ldots, n)$ if

$$\boldsymbol{X} \succeq \boldsymbol{O}, X_{ii} = 1 \ (i = 1, \ldots, n), \mathrm{rank}(\boldsymbol{X}) = 1$$

by decomposing $\boldsymbol{X} = \boldsymbol{x}\boldsymbol{x}^T$.

Therefore

$$\max \ W(L, R) = \frac{1}{4}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(1 - x_i x_j) \qquad \text{s.t.} \quad x_i \in \{-1, 1\} \ (i = 1, \ldots, n)$$

is equivalent to

$$\max \ W(L, R) = \boldsymbol{C} \bullet \boldsymbol{X} \qquad \text{s.t.} \quad \boldsymbol{X} \succeq \boldsymbol{O}, X_{ii} = 1 \ (i = 1, \ldots, n), \mathrm{rank}(\boldsymbol{X}) = 1$$

The NP-hard difficulty is now embedded into the constraint $\mathrm{rank}(\boldsymbol{X}) = 1$.

SDP relaxation

Goemans and Williamson relaxed the constraints $\mathrm{rank}(\boldsymbol{X}) = 1$.

Max-cut problem

$$W^* = \max\ W(L,R) = \boldsymbol{C} \bullet \boldsymbol{X} \qquad \text{s.t.} \quad \boldsymbol{X} \succeq \boldsymbol{O}, X_{ii} = 1\ (i = 1,\ldots,n), \mathrm{rank}(\boldsymbol{X}) = 1$$

SDP relaxation

$$\widehat{W} = \max\ W(L,R) = \boldsymbol{C} \bullet \boldsymbol{X} \qquad \text{s.t.} \quad \boldsymbol{X} \succeq \boldsymbol{O}, X_{ii} = 1\ (i = 1,\ldots,n)$$

SDP relaxation

Goemans and Williamson relaxed the constraints $\mathrm{rank}(\boldsymbol{X}) = 1$.

Max-cut problem

$$W^* = \max \; W(L,R) = \boldsymbol{C} \bullet \boldsymbol{X} \qquad \text{s.t.} \quad \boldsymbol{X} \succeq \boldsymbol{O}, X_{ii} = 1 \; (i = 1, \ldots, n), \mathrm{rank}(\boldsymbol{X}) = 1$$

SDP relaxation

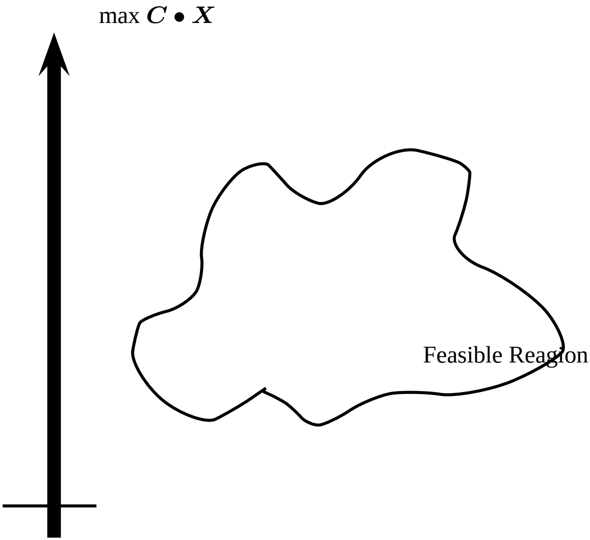
$$\widehat{W} = \max \; W(L,R) = \boldsymbol{C} \bullet \boldsymbol{X} \qquad \text{s.t.} \quad \boldsymbol{X} \succeq \boldsymbol{O}, X_{ii} = 1 \; (i = 1, \ldots, n)$$

From the relaxation solution $\widehat{\boldsymbol{X}}$, they generated a rank-1 feasible solution $\widetilde{\boldsymbol{X}}$ such that

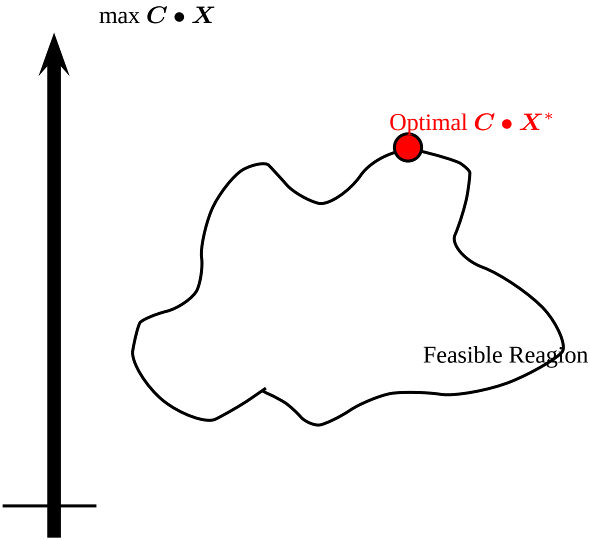
$$\boldsymbol{C} \bullet \widetilde{\boldsymbol{X}} \geq 0.878 \times W^*$$

in average.

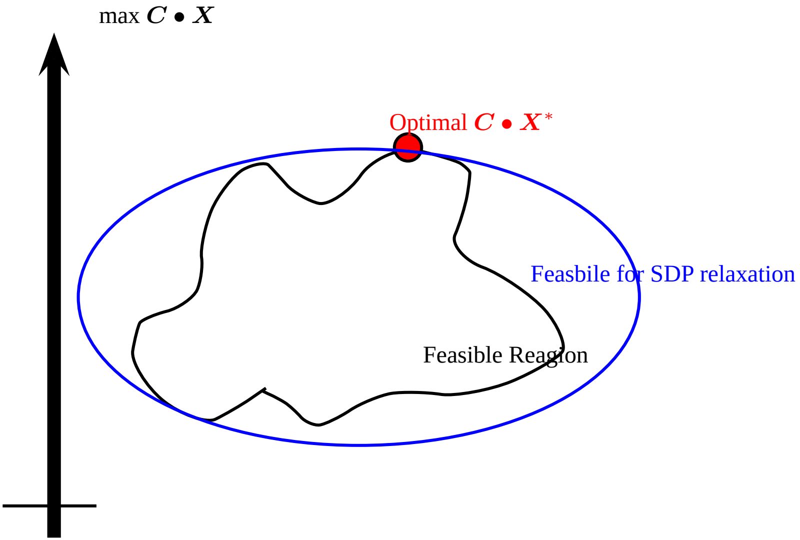
Quality of SDP relaxation



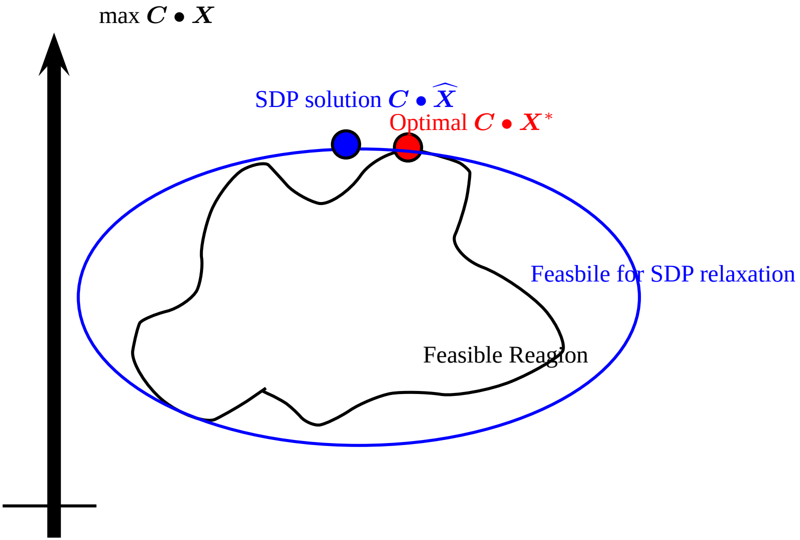
Quality of SDP relaxation



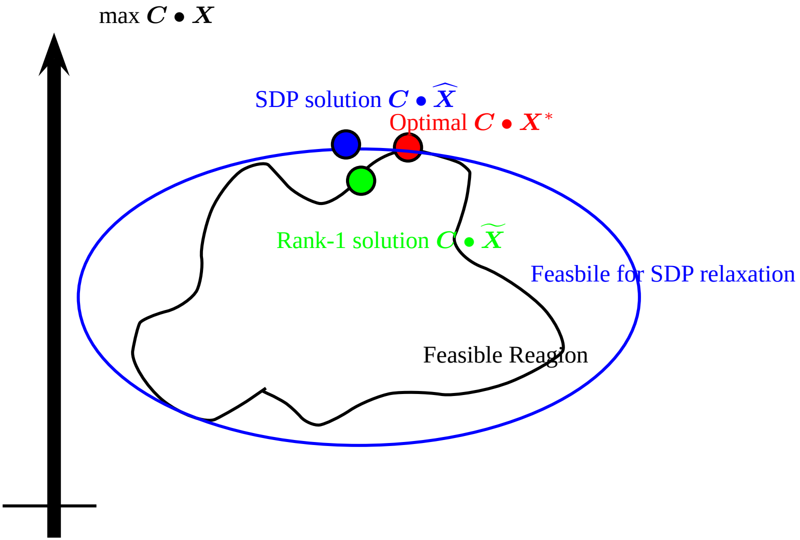
Quality of SDP relaxation



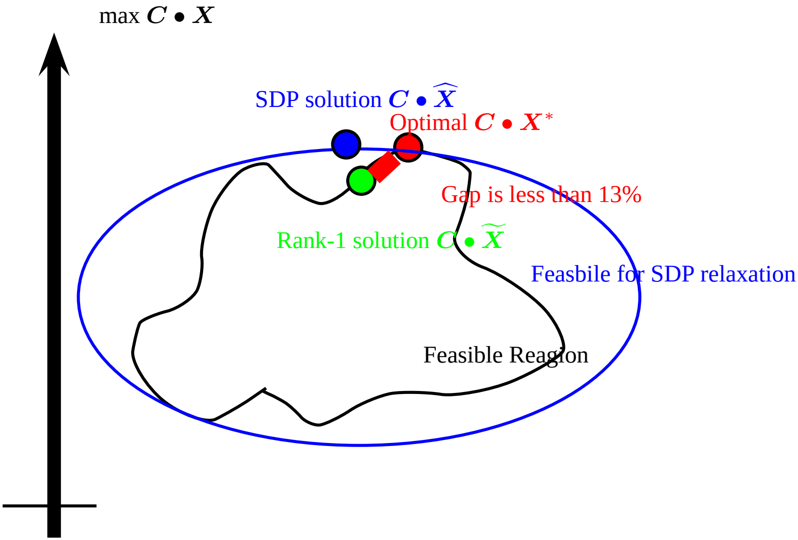
Quality of SDP relaxation



Quality of SDP relaxation



Quality of SDP relaxation



3. Biswas-Ye SDP relaxation

Biswas-Ye Approach to SNL

The main idea of Biswas-Ye Approach

- Lift the variable space of SNL to a higher dimension
- Then apply SDP relaxation

Biswas-Ye Approach to SNL

The main idea of Biswas-Ye Approach

- Lift the variable space of SNL to a higher dimension
- Then apply SDP relaxation

At the first step, for sensor locations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$ in d-dimensional space, they introduce the matrix $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in \mathbb{R}^{d \times n}$. They also use $\boldsymbol{Y} = \boldsymbol{X}^T \boldsymbol{X}$. Then, $Y_{pq} = [\boldsymbol{X}^T \boldsymbol{X}]_{pq} = \boldsymbol{x}_p^T \boldsymbol{x}_q$.

Biswas-Ye Approach to SNL

The main idea of Biswas-Ye Approach

- Lift the variable space of SNL to a higher dimension
- Then apply SDP relaxation

At the first step, for sensor locations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$ in d-dimensional space, they introduce the matrix $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in \mathbb{R}^{d \times n}$. They also use $\boldsymbol{Y} = \boldsymbol{X}^T \boldsymbol{X}$. Then, $Y_{pq} = [\boldsymbol{X}^T \boldsymbol{X}]_{pq} = \boldsymbol{x}_p^T \boldsymbol{x}_q$.
Hence,

$$
\begin{aligned}
||\boldsymbol{x}_p - \boldsymbol{x}_q||^2 &= (\boldsymbol{x}_p - \boldsymbol{x}_q)^T (\boldsymbol{x}_p - \boldsymbol{x}_q) \\
&= \boldsymbol{x}_p^T \boldsymbol{x}_p - 2\boldsymbol{x}_p^T \boldsymbol{x}_q + \boldsymbol{x}_q^T \boldsymbol{x}_q \\
&= Y_{pp} - 2Y_{pq} + Y_{qq} \\
||\boldsymbol{x}_p - \boldsymbol{a}_r||^2 &= (\boldsymbol{x}_p - \boldsymbol{a}_r)^T (\boldsymbol{x}_p - \boldsymbol{a}_r) \\
&= \boldsymbol{x}_p^T \boldsymbol{x}_p - 2\boldsymbol{x}_p^T \boldsymbol{a}_r + \boldsymbol{a}_r^T \boldsymbol{a}_r \\
&= Y_{pp} - 2\boldsymbol{a}_r^T [\boldsymbol{X}]_{*p} + ||\boldsymbol{a}_r||^2
\end{aligned}
$$

Matrix Formulation

Original Formulation

$$\min \quad : \quad 0 \qquad \text{s.t.} : \begin{array}{ll} ||x_p - x_q||^2 = d_{pq}^2 & \forall (p,q) \in \mathcal{N}_x \\ ||x_p - a_r||^2 = d_{pr}^2 & \forall (p,r) \in \mathcal{N}_a \end{array}$$

Matrix Formulation

$$\min \quad : \quad 0 \qquad \text{s.t.} : \begin{array}{ll} Y_{pp} - 2Y_{pq} + Y_{qq} = d_{pq}^2 & \forall (p,q) \in \mathcal{N}_x \\ Y_{pp} - 2\boldsymbol{a}_r^T[\boldsymbol{X}]_{*p} + ||\boldsymbol{a}_r||^2 = d_{pr}^2 & \forall (p,r) \in \mathcal{N}_a \\ \boldsymbol{Y} = \boldsymbol{X}^T\boldsymbol{X} & \end{array}$$

Matrix Formulation

Original Formulation

$$\min \quad : \quad 0 \qquad \text{s.t. :} \quad \begin{array}{ll} ||x_p - x_q||^2 = d_{pq}^2 & \forall (p,q) \in \mathcal{N}_x \\ ||x_p - a_r||^2 = d_{pr}^2 & \forall (p,r) \in \mathcal{N}_a \end{array}$$

Matrix Formulation

$$\min \quad : \quad 0 \qquad \text{s.t. :} \quad \begin{array}{ll} Y_{pp} - 2Y_{pq} + Y_{qq} = d_{pq}^2 & \forall (p,q) \in \mathcal{N}_x \\ Y_{pp} - 2\boldsymbol{a}_r^T [\boldsymbol{X}]_{*p} + ||\boldsymbol{a}_r||^2 = d_{pr}^2 & \forall (p,r) \in \mathcal{N}_a \\ \boldsymbol{Y} = \boldsymbol{X}^T \boldsymbol{X} & \end{array}$$

The NP-hard difficulty is now summarized into $\boldsymbol{Y} = \boldsymbol{X}^T \boldsymbol{X}$.

Matrix Formulation

Original Formulation

$$\min \;:\; 0 \qquad \text{s.t.}: \begin{array}{ll} ||x_p - x_q||^2 = d_{pq}^2 & \forall (p,q) \in \mathcal{N}_x \\ ||x_p - a_r||^2 = d_{pr}^2 & \forall (p,r) \in \mathcal{N}_a \end{array}$$

Matrix Formulation

$$\min \;:\; 0 \qquad \text{s.t.}: \begin{array}{ll} Y_{pp} - 2Y_{pq} + Y_{qq} = d_{pq}^2 & \forall (p,q) \in \mathcal{N}_x \\ Y_{pp} - 2\boldsymbol{a}_r^T[\boldsymbol{X}]_{*p} + ||\boldsymbol{a}_r||^2 = d_{pr}^2 & \forall (p,r) \in \mathcal{N}_a \\ \boldsymbol{Y} = \boldsymbol{X}^T\boldsymbol{X} & \end{array}$$

The NP-hard difficulty is now summarized into $\boldsymbol{Y} = \boldsymbol{X}^T\boldsymbol{X}$.
Biswas-Ye replaced this constraint by SDP relaxation $\boldsymbol{Y} \succeq \boldsymbol{X}^T\boldsymbol{X}$
($\Leftrightarrow \boldsymbol{Y} - \boldsymbol{X}^T\boldsymbol{X} \succeq \boldsymbol{O}$).

Matrix Formulation

Original Formulation

$$\min \quad : \quad 0 \qquad \text{s.t.} : \begin{array}{ll} ||x_p - x_q||^2 = d_{pq}^2 & \forall (p,q) \in \mathcal{N}_x \\ ||x_p - a_r||^2 = d_{pr}^2 & \forall (p,r) \in \mathcal{N}_a \end{array}$$

Matrix Formulation

$$\min \quad : \quad 0 \qquad \text{s.t.} : \begin{array}{ll} Y_{pp} - 2Y_{pq} + Y_{qq} = d_{pq}^2 & \forall (p,q) \in \mathcal{N}_x \\ Y_{pp} - 2\boldsymbol{a}_r^T [\boldsymbol{X}]_{*p} + ||\boldsymbol{a}_r||^2 = d_{pr}^2 & \forall (p,r) \in \mathcal{N}_a \\ \boldsymbol{Y} = \boldsymbol{X}^T \boldsymbol{X} & \end{array}$$

The NP-hard difficulty is now summarized into $\boldsymbol{Y} = \boldsymbol{X}^T \boldsymbol{X}$.
Biswas-Ye replaced this constraint by SDP relaxation $\boldsymbol{Y} \succeq \boldsymbol{X}^T \boldsymbol{X}$
($\Leftrightarrow \boldsymbol{Y} - \boldsymbol{X}^T \boldsymbol{X} \succeq \boldsymbol{O}$).
Note that $\boldsymbol{Y} = \boldsymbol{X}^T \boldsymbol{X} \Leftrightarrow \boldsymbol{Y} \succeq \boldsymbol{X}^T \boldsymbol{X}, \text{rank}(\boldsymbol{Y}) = d$.

Schur complement

Due to Schur complement,

$$\boldsymbol{Y} - \boldsymbol{X}^T\boldsymbol{X} \succeq \boldsymbol{O} \quad \Leftrightarrow \quad \begin{pmatrix} \boldsymbol{I}_d & \boldsymbol{X} \\ \boldsymbol{X}^T & \boldsymbol{Y} \end{pmatrix} \succeq \boldsymbol{O}$$

Schur complement

Due to Schur complement,

$$\boldsymbol{Y} - \boldsymbol{X}^T\boldsymbol{X} \succeq \boldsymbol{O} \quad \Leftrightarrow \quad \begin{pmatrix} \boldsymbol{I}_d & \boldsymbol{X} \\ \boldsymbol{X}^T & \boldsymbol{Y} \end{pmatrix} \succeq \boldsymbol{O}$$

Proof ($\Rightarrow$) Assume $\boldsymbol{Y} - \boldsymbol{X}^T\boldsymbol{X} \succeq \boldsymbol{O}$, then $\boldsymbol{u}^T(\boldsymbol{Y} - \boldsymbol{X}^T\boldsymbol{X})\boldsymbol{u} \geq 0$ for any vector $\boldsymbol{u}$. For any vectors $\boldsymbol{v}, \boldsymbol{w}$,

$$\begin{aligned} & \begin{pmatrix} \boldsymbol{v}^T & \boldsymbol{w}^T \end{pmatrix}^T \begin{pmatrix} \boldsymbol{I}_d & \boldsymbol{X} \\ \boldsymbol{X}^T & \boldsymbol{Y} \end{pmatrix} \begin{pmatrix} \boldsymbol{v} \\ \boldsymbol{w} \end{pmatrix} \\ = \quad & \boldsymbol{v}^T\boldsymbol{v} + 2\boldsymbol{v}^T\boldsymbol{X}\boldsymbol{w} + \boldsymbol{w}^T\boldsymbol{Y}\boldsymbol{w} \\ = \quad & (\boldsymbol{v} + \boldsymbol{X}\boldsymbol{w})^T(\boldsymbol{v} + \boldsymbol{X}\boldsymbol{w}) - (\boldsymbol{X}\boldsymbol{w})^T(\boldsymbol{X}\boldsymbol{w}) + \boldsymbol{w}^T\boldsymbol{Y}\boldsymbol{w} \\ = \quad & ||\boldsymbol{v} + \boldsymbol{X}\boldsymbol{w}||^2 + \boldsymbol{w}^T(\boldsymbol{Y} - \boldsymbol{X}^T\boldsymbol{X})\boldsymbol{w} \geq 0 \end{aligned}$$

($\Leftarrow$) Assume there exists a vector $\boldsymbol{w}$ s.t. $\boldsymbol{w}^T(\boldsymbol{Y} - \boldsymbol{X}^T\boldsymbol{X})\boldsymbol{w} < 0$. Let $\boldsymbol{v} = -\boldsymbol{X}\boldsymbol{w}$. Then

$$\begin{aligned} & \begin{pmatrix} \boldsymbol{v}^T & \boldsymbol{w}^T \end{pmatrix}^T \begin{pmatrix} \boldsymbol{I}_d & \boldsymbol{X} \\ \boldsymbol{X}^T & \boldsymbol{Y} \end{pmatrix} \begin{pmatrix} \boldsymbol{v} \\ \boldsymbol{w} \end{pmatrix} \\ = \quad & ||\boldsymbol{v} + \boldsymbol{X}\boldsymbol{w}||^2 + \boldsymbol{w}^T(\boldsymbol{Y} - \boldsymbol{X}^T\boldsymbol{X})\boldsymbol{w} < 0 \end{aligned}$$

Lift up of the variable space

They prepared the large variable matrix $\boldsymbol{Z} \in \mathbb{R}^{(d+n)\times(d+n)}$:

$$\boldsymbol{Z} = \begin{pmatrix} \boldsymbol{I}_d & \boldsymbol{X} \\ \boldsymbol{X}^T & \boldsymbol{Y} \end{pmatrix}$$

Biswas-Ye SDP relaxation for SNL

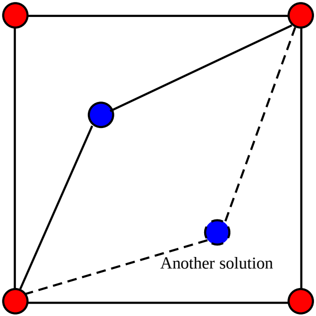
$$\min\ \boldsymbol{O} \bullet \boldsymbol{Z} \qquad \text{s.t.} \quad \boldsymbol{A}_k \bullet \boldsymbol{Z} = b_k \ (k \in \Lambda), \quad \boldsymbol{Z} \succeq \boldsymbol{O}$$

The set Λ includes all the linear constraints, for example,

$$\begin{aligned} & ||\boldsymbol{x}_p - \boldsymbol{a}_r||^2 = d_{pr}^2 \\ \Leftrightarrow \quad & Y_{pp} - 2\boldsymbol{a}_r^T[X]_{*p} + ||\boldsymbol{a}_r||^2 = d_{pr}^2 \\ \Leftrightarrow \quad & \begin{pmatrix} \boldsymbol{O} & \begin{pmatrix} -\boldsymbol{a}_r \end{pmatrix} \\ \begin{pmatrix} -\boldsymbol{a}_r^T \\ \\ \end{pmatrix} & \begin{pmatrix} 1 \\ \\ \end{pmatrix} \end{pmatrix} \bullet \begin{pmatrix} \boldsymbol{I}_d & \boldsymbol{X} \\ \boldsymbol{X}^T & \boldsymbol{Y} \end{pmatrix} = d_{pr}^2 - ||a_r||^2 \end{aligned}$$

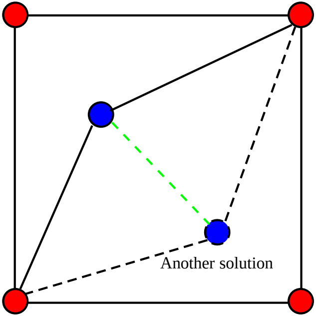
The effect of Biswas-Ye relaxation

In SNL, multiple solutions are discrete. The SDP relaxation connects them.



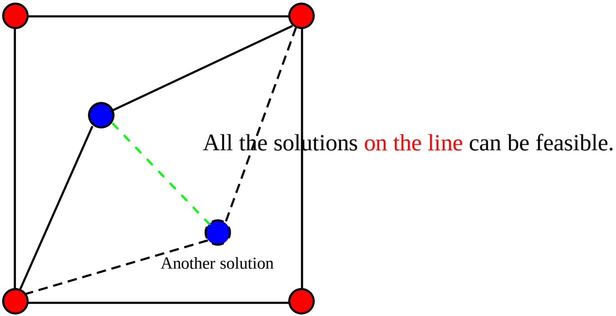
The effect of Biswas-Ye relaxation

In SNL, multiple solutions are discrete. The SDP relaxation connects them.



The effect of Biswas-Ye relaxation

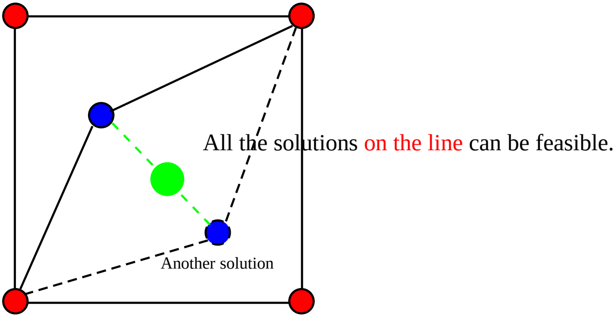
In SNL, multiple solutions are discrete. The SDP relaxation connects them.



Therefore, the feasible region of SDP relaxation is convex.

The effect of Biswas-Ye relaxation

In SNL, multiple solutions are discrete. The SDP relaxation connects them.



Therefore, the feasible region of SDP relaxation is convex.
When we apply *Primal-Dual Interior-Point Methods*, the obtained solution is usually a max-rank solution. In SNL, max-rank corresponds to the center of the feasible region.

Uniquely Localizable and SDP solution

In SDP relaxation, we obtain the solution $(\boldsymbol{X}^*, \boldsymbol{Y}^*)$ with the property $\boldsymbol{Y}^* \succeq (\boldsymbol{X}^*)^T(\boldsymbol{X}^*)$.

Uniquely Localizable and SDP solution

In SDP relaxation, we obtain the solution $(\boldsymbol{X}^*, \boldsymbol{Y}^*)$ with the property $\boldsymbol{Y}^* \succeq (\boldsymbol{X}^*)^T(\boldsymbol{X}^*)$.

So *et al* showed that

If $\boldsymbol{Y}^* = (\boldsymbol{X}^*)^T(\boldsymbol{X}^*)$ holds, the SNL has the unique solution. There is no solution for the SNL except $\boldsymbol{X}^*$.

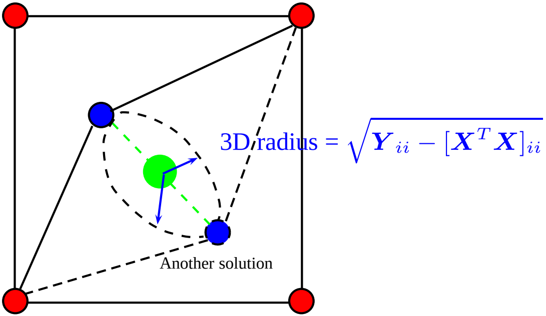
Uniquely Localizable and SDP solution

In SDP relaxation, we obtain the solution $(\boldsymbol{X}^*, \boldsymbol{Y}^*)$ with the property $\boldsymbol{Y}^* \succeq (\boldsymbol{X}^*)^T(\boldsymbol{X}^*)$.

So *et al* showed that

If $\boldsymbol{Y}^* = (\boldsymbol{X}^*)^T(\boldsymbol{X}^*)$ holds, the SNL has the unique solution. There is no solution for the SNL except $\boldsymbol{X}^*$.

This fact is very strong in the sense that without heavy computation cost, we can know whether the SNL has a unique solution or multiple solutions.

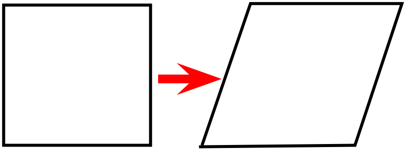


Uniquely Localizable

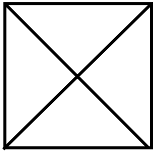
(Roughly speaking),
So *et al* implied for 2D that if the base graph satisfies the three conditions,

1. Algebraically independent
   (No three vertices do not exist on a unique line)
2. Rigid
   (Continuous deformation is not allowed)
3. Three-connected
   (The graph is still connected even when any two vertices and their edges are removed)

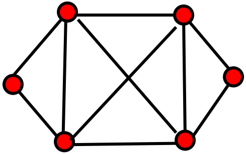
then the SNL has the unique solution. The result for 3D or higher dimension is unknown.



Non -Rigid



Rigid



Three-connected

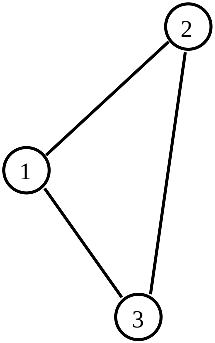
Trilateration Graph

For example, a trilateration graph satisfies the three conditions.

Trilateration graph in d dimension

There is an order of vertices such that

- The first $d+1$ vertices connect to each other.
- The pth vertex ($p \geq d+2$) has at least $d+1$ edges from the vertex $1, \ldots, p-1$.



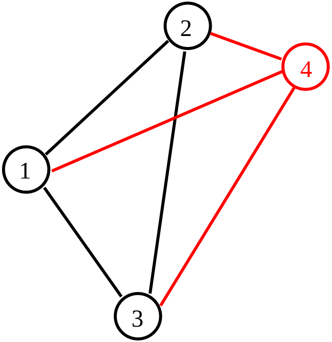
Trilateration Graph

For example, a trilateration graph satisfies the three conditions.

Trilateration graph in d dimension

There is an order of vertices such that

- The first $d+1$ vertices connect to each other.
- The pth vertex ($p \geq d+2$) has at least $d+1$ edges from the vertex $1, \ldots, p-1$.



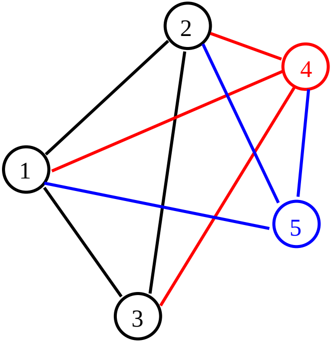
Trilateration Graph

For example, a trilateration graph satisfies the three conditions.

Trilateration graph in d dimension

There is an order of vertices such that

- The first $d+1$ vertices connect to each other.
- The pth vertex ($p \geq d+2$) has at least $d+1$ edges from the vertex $1, \ldots, p-1$.



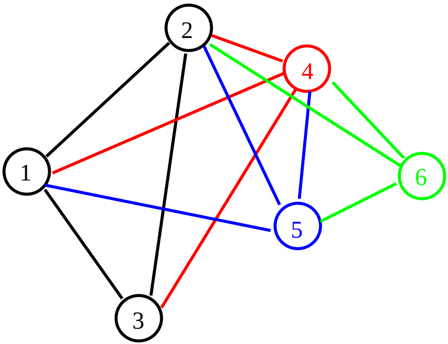
Trilateration Graph

For example, a trilateration graph satisfies the three conditions.

Trilateration graph in d dimension

There is an order of vertices such that

- The first $d+1$ vertices connect to each other.
- The pth vertex ($p \geq d+2$) has at least $d+1$ edges from the vertex $1, \ldots, p-1$.



4. SFSDP – Exploiting Sparsity

- Aggregate Sparsity
- Chordal Graph
- Matrix Completion
- Apply Matrix Completion to SNL

SFSDP

The main idea of SFSDP

Exploiting sparsity in the SDP to solve large-scale SNLs. The mathematical tool is Matrix completion.

SFSDP

The main idea of SFSDP

Exploiting sparsity in the SDP to solve large-scale SNLs. The mathematical tool is Matrix completion.

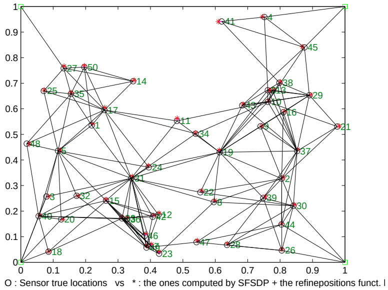
Without exploiting sparsity, the number of sensors n is limited to $n \leq 3,000$.
We want to solve $n \geq 15,000$.

SFSDP

The main idea of SFSDP

Exploiting sparsity in the SDP to solve large-scale SNLs. The mathematical tool is Matrix completion.

Without exploiting sparsity, the number of sensors n is limited to $n \leq 3,000$.
We want to solve $n \geq 15,000$.
The network of SNL is usually sparse.



In most applications,

- The sensors with short distances are connected.
- But, the pairs with far distance are NOT connected.

Input data matrices in SNL are very sparse

$$
\begin{aligned}
& ||\boldsymbol{x}_p - \boldsymbol{x}_q||^2 = d_{pq}^2 \Leftrightarrow \boldsymbol{x}_p^T \boldsymbol{x}_p - 2\boldsymbol{x}_p^T \boldsymbol{x}_q + \boldsymbol{x}_q^T \boldsymbol{x}_q = d_{pq}^2 \\
\Leftrightarrow \quad & Y_{pp} - 2Y_{pq} + Y_{qq} = d_{pq}^2 \\
\Rightarrow \quad & \left(\begin{array}{cc} \boldsymbol{O} & \boldsymbol{O} \\ \boldsymbol{O} & \left(\begin{array}{cc} 1 & -1 \\ -1 & 1 \end{array} \right) \end{array} \right) \bullet \left(\begin{array}{cc} \boldsymbol{I}_d & \boldsymbol{X} \\ \boldsymbol{X}^T & \boldsymbol{Y} \end{array} \right) = d_{pq}^2
\end{aligned}
$$

$$
\begin{aligned}
& ||\boldsymbol{x}_p - \boldsymbol{a}_r||^2 = d_{pr}^2 \Leftrightarrow \boldsymbol{x}_p^T \boldsymbol{x}_p - 2\boldsymbol{x}_p^T \boldsymbol{a}_r + \boldsymbol{a}_r^T \boldsymbol{a}_r = d_{pr}^2 \\
\Leftrightarrow \quad & Y_{pp} - 2\boldsymbol{a}_r^T [\boldsymbol{X}]_{*p} + Y_{qq} = d_{pq}^2 - ||\boldsymbol{a}_r||^2 \\
\Rightarrow \quad & \left(\begin{array}{cc} \boldsymbol{O} & \left(\begin{array}{c} -\boldsymbol{a}_r \end{array} \right) \\ \left(\begin{array}{c} -\boldsymbol{a}_r^T \end{array} \right) & \left(\begin{array}{c} 1 \end{array} \right) \end{array} \right) \bullet \left(\begin{array}{cc} \boldsymbol{I}_d & \boldsymbol{X} \\ \boldsymbol{X}^T & \boldsymbol{Y} \end{array} \right) = d_{pr}^2 - ||a_r||^2
\end{aligned}
$$

Only 4 or $2d + 1$ non-zero elements in $(d + n) \times (d + n)$ positions.

Only partial elements are necessary

To understand matrix completion, we start from a simple example.

$$\min \quad : \quad \begin{pmatrix} 2 & -1 & 0 & -2 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 4 \\ -2 & 0 & 4 & 0 \end{pmatrix} \bullet \begin{pmatrix} X_{11} & X_{12} & X_{13} & X_{14} \\ X_{12} & X_{22} & X_{23} & X_{24} \\ X_{13} & X_{23} & X_{33} & X_{34} \\ X_{14} & X_{24} & X_{34} & X_{44} \end{pmatrix}$$

$$= 2X_{11} - 2X_{12} - 4X_{14} - X_{33} + 8X_{34}$$

$$\text{s.t.} \quad : \quad \begin{pmatrix} 0 & -2 & 0 & 0 \\ -2 & -1 & 0 & 0 \\ 0 & 0 & 2 & -3 \\ 0 & 0 & -3 & 1 \end{pmatrix} \bullet \begin{pmatrix} X_{11} & X_{12} & X_{13} & X_{14} \\ X_{12} & X_{22} & X_{23} & X_{24} \\ X_{13} & X_{23} & X_{33} & X_{34} \\ X_{14} & X_{24} & X_{34} & X_{44} \end{pmatrix} = 1, \qquad \boldsymbol{X} \succeq \boldsymbol{O}$$

$$= -4X_{12} - X_{22} + 2X_{33} - 6X_{34} + X_{44}$$

Only partial elements are necessary

To understand matrix completion, we start from a simple example.

$$\min \quad : \begin{pmatrix} 2 & -1 & 0 & -2 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 4 \\ -2 & 0 & 4 & 0 \end{pmatrix} \bullet \begin{pmatrix} X_{11} & X_{12} & X_{13} & X_{14} \\ X_{12} & X_{22} & X_{23} & X_{24} \\ X_{13} & X_{23} & X_{33} & X_{34} \\ X_{14} & X_{24} & X_{34} & X_{44} \end{pmatrix}$$

$$= 2X_{11} - 2X_{12} - 4X_{14} - X_{33} + 8X_{34}$$

$$\text{s.t.} \quad : \begin{pmatrix} 0 & -2 & 0 & 0 \\ -2 & -1 & 0 & 0 \\ 0 & 0 & 2 & -3 \\ 0 & 0 & -3 & 1 \end{pmatrix} \bullet \begin{pmatrix} X_{11} & X_{12} & X_{13} & X_{14} \\ X_{12} & X_{22} & X_{23} & X_{24} \\ X_{13} & X_{23} & X_{33} & X_{34} \\ X_{14} & X_{24} & X_{34} & X_{44} \end{pmatrix} = 1, \qquad \boldsymbol{X} \succeq \boldsymbol{O}$$

$$= -4X_{12} - X_{22} + 2X_{33} - 6X_{34} + X_{44}$$

Only blue elements are enough to evaluate the objective function and the equality constraint.

Only partial elements are necessary

To understand matrix completion, we start from a simple example.

$$\min \quad : \quad \begin{pmatrix} 2 & -1 & 0 & -2 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 4 \\ -2 & 0 & 4 & 0 \end{pmatrix} \bullet \begin{pmatrix} X_{11} & X_{12} & X_{13} & X_{14} \\ X_{12} & X_{22} & X_{23} & X_{24} \\ X_{13} & X_{23} & X_{33} & X_{34} \\ X_{14} & X_{24} & X_{34} & X_{44} \end{pmatrix}$$

$$= 2X_{11} - 2X_{12} - 4X_{14} - X_{33} + 8X_{34}$$

$$\text{s.t.} \quad : \quad \begin{pmatrix} 0 & -2 & 0 & 0 \\ -2 & -1 & 0 & 0 \\ 0 & 0 & 2 & -3 \\ 0 & 0 & -3 & 1 \end{pmatrix} \bullet \begin{pmatrix} X_{11} & X_{12} & X_{13} & X_{14} \\ X_{12} & X_{22} & X_{23} & X_{24} \\ X_{13} & X_{23} & X_{33} & X_{34} \\ X_{14} & X_{24} & X_{34} & X_{44} \end{pmatrix} = 1, \qquad \boldsymbol{X} \succeq \boldsymbol{O}$$

$$= -4X_{12} - X_{22} + 2X_{33} - 6X_{34} + X_{44}$$

Only blue elements are enough to evaluate the objective function and the equality constraint. However, for positive semidefinite conditions,

$$\begin{pmatrix} 5 & -4 & 0 & -1 \\ -4 & 4 & 1 & 0 \\ 0 & 1 & 3 & 2 \\ -1 & 0 & 2 & 3 \end{pmatrix} \not\succeq \boldsymbol{O} \qquad \text{but} \qquad \begin{pmatrix} 5 & -4 & 0 & -1 \\ -4 & 4 & 1 & 2 \\ 0 & 1 & 3 & 2 \\ -1 & 2 & 2 & 3 \end{pmatrix} \succeq \boldsymbol{O}$$

Only partial elements are necessary

To understand matrix completion, we start from a simple example.

$$\min \quad : \quad \begin{pmatrix} 2 & -1 & 0 & -2 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 4 \\ -2 & 0 & 4 & 0 \end{pmatrix} \bullet \begin{pmatrix} X_{11} & X_{12} & X_{13} & X_{14} \\ X_{12} & X_{22} & X_{23} & X_{24} \\ X_{13} & X_{23} & X_{33} & X_{34} \\ X_{14} & X_{24} & X_{34} & X_{44} \end{pmatrix}$$

$$= 2X_{11} - 2X_{12} - 4X_{14} - X_{33} + 8X_{34}$$

$$\text{s.t.} \quad : \quad \begin{pmatrix} 0 & -2 & 0 & 0 \\ -2 & -1 & 0 & 0 \\ 0 & 0 & 2 & -3 \\ 0 & 0 & -3 & 1 \end{pmatrix} \bullet \begin{pmatrix} X_{11} & X_{12} & X_{13} & X_{14} \\ X_{12} & X_{22} & X_{23} & X_{24} \\ X_{13} & X_{23} & X_{33} & X_{34} \\ X_{14} & X_{24} & X_{34} & X_{44} \end{pmatrix} = 1, \qquad \boldsymbol{X} \succeq \boldsymbol{O}$$

$$= -4X_{12} - X_{22} + 2X_{33} - 6X_{34} + X_{44}$$

Only blue elements are enough to evaluate the objective function and the equality constraint. However, for positive semidefinite conditions,

$$\begin{pmatrix} 5 & -4 & 0 & -1 \\ -4 & 4 & 1 & 0 \\ 0 & 1 & 3 & 2 \\ -1 & 0 & 2 & 3 \end{pmatrix} \not\succeq \boldsymbol{O} \qquad \text{but} \qquad \begin{pmatrix} 5 & -4 & 0 & -1 \\ -4 & 4 & 1 & 2 \\ 0 & 1 & 3 & 2 \\ -1 & 2 & 2 & 3 \end{pmatrix} \succeq \boldsymbol{O}$$

Q: How many elements are necessary?

Aggregate Sparsity Pattern

Aggregate Sparsity Pattern

For an SDP

$$\min \boldsymbol{C} \bullet \boldsymbol{X} \quad \text{s.t. } \boldsymbol{A}_k \bullet \boldsymbol{X} = b_k (k = 1, \ldots, m), \boldsymbol{X} \succeq \boldsymbol{O}$$

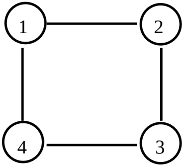
We define

$$\mathcal{E} = \{(i,j) : i \neq j, C_{ij} \neq 0 \text{ or } [\boldsymbol{A}_k]_{ij} \neq 0 \text{ for some } k = 1, \ldots, m\}$$

$$\begin{pmatrix} X_{11} & X_{12} & X_{13} & X_{14} \\ X_{12} & X_{22} & X_{23} & X_{24} \\ X_{13} & X_{23} & X_{33} & X_{34} \\ X_{14} & X_{24} & X_{34} & X_{44} \end{pmatrix}$$

$$\Rightarrow \quad \mathcal{E} = \{(1,2), (1,4), (2,3), (3,4)\}$$

Corresponding graph is



Aggregate Sparsity Pattern

Aggregate Sparsity Pattern

For an SDP

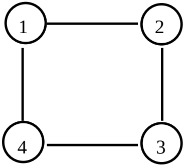
$$\min \boldsymbol{C} \bullet \boldsymbol{X} \quad \text{s.t. } \boldsymbol{A}_k \bullet \boldsymbol{X} = b_k (k = 1, \ldots, m), \boldsymbol{X} \succeq \boldsymbol{O}$$

We define

$$\mathcal{E} = \{(i, j) : i \neq j, C_{ij} \neq 0 \text{ or } [\boldsymbol{A}_k]_{ij} \neq 0 \text{ for some } k = 1, \ldots, m\}$$

Corresponding graph is

$$\begin{pmatrix} X_{11} & X_{12} & X_{13} & X_{14} \\ X_{12} & X_{22} & X_{23} & X_{24} \\ X_{13} & X_{23} & X_{33} & X_{34} \\ X_{14} & X_{24} & X_{34} & X_{44} \end{pmatrix}$$



$$\Rightarrow \quad \mathcal{E} = \{(1, 2), (1, 4), (2, 3), (3, 4)\}$$

We have already known that at least Aggregate Sparsity Pattern are necessary.
Positive semidefiniteness is related to *chordal graph*.

Chordal Graph

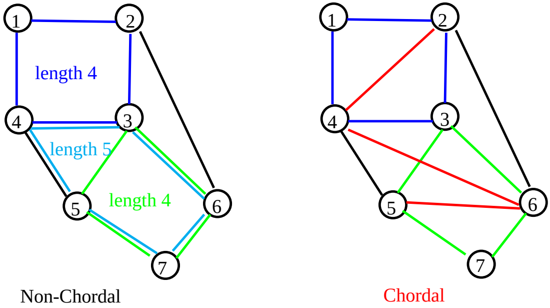
Chordal Graph

A graph is *chordal* if every cycle whose length is larger than 3 has a chord.

Chordal Graph

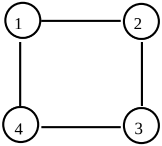
Chordal Graph

A graph is *chordal* if every cycle whose length is larger than 3 has a chord.



Chordal Extension

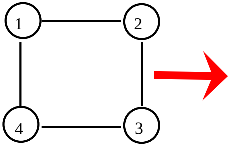
To satisfy positive semidefiniteness of entire $\boldsymbol{X}$, Aggregate sparsity patter is not enough, but Chordal extension is enough.



$$\begin{pmatrix} X_{11} & X_{12} & & X_{14} \\ X_{12} & X_{22} & X_{23} & \\ & X_{23} & X_{33} & X_{34} \\ X_{14} & & X_{34} & X_{44} \end{pmatrix}$$

Chordal Extension

To satisfy positive semidefiniteness of entire $\boldsymbol{X}$, Aggregate sparsity patter is not enough, but Chordal extension is enough.

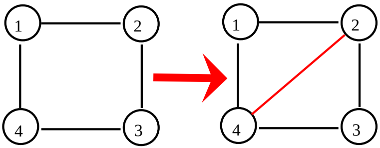


Chordal Extension

$$\begin{pmatrix} X_{11} & X_{12} & & X_{14} \\ X_{12} & X_{22} & X_{23} & \\ & X_{23} & X_{33} & X_{34} \\ X_{14} & & X_{34} & X_{44} \end{pmatrix}$$

Chordal Extension

To satisfy positive semidefiniteness of entire $\boldsymbol{X}$, Aggregate sparsity patter is not enough, but Chordal extension is enough.

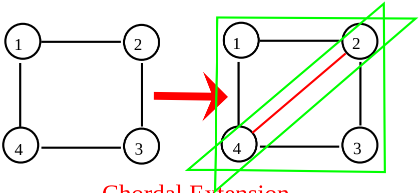


Chordal Extension

$$\begin{pmatrix} X_{11} & X_{12} & & X_{14} \\ X_{12} & X_{22} & X_{23} & \\ & X_{23} & X_{33} & X_{34} \\ X_{14} & & X_{34} & X_{44} \end{pmatrix} \Rightarrow \begin{pmatrix} X_{11} & X_{12} & & X_{14} \\ X_{12} & X_{22} & X_{23} & X_{24} \\ & X_{23} & X_{33} & X_{34} \\ X_{14} & X_{24} & X_{34} & X_{44} \end{pmatrix} \succeq \boldsymbol{O}$$

Chordal Extension

To satisfy positive semidefiniteness of entire $\boldsymbol{X}$, Aggregate sparsity patter is not enough, but Chordal extension is enough.



Chordal Extension

$$\begin{pmatrix} X_{11} & X_{12} & & X_{14} \\ X_{12} & X_{22} & X_{23} & \\ & X_{23} & X_{33} & X_{34} \\ X_{14} & & X_{34} & X_{44} \end{pmatrix} \Rightarrow \begin{pmatrix} X_{11} & X_{12} & & X_{14} \\ X_{12} & X_{22} & X_{23} & X_{24} \\ & X_{23} & X_{33} & X_{34} \\ X_{14} & X_{24} & X_{34} & X_{44} \end{pmatrix} \succeq \boldsymbol{O}$$

$$\Leftrightarrow \begin{pmatrix} X_{11} & X_{12} & X_{14} \\ X_{12} & X_{22} & X_{24} \\ X_{14} & X_{24} & X_{44} \end{pmatrix} \succeq \boldsymbol{O}, \begin{pmatrix} X_{22} & X_{23} & X_{24} \\ X_{23} & X_{33} & X_{34} \\ X_{24} & X_{34} & X_{44} \end{pmatrix} \succeq \boldsymbol{O}$$

Submatrix

In the following, we use many sub-matrices. Here, we introduce a simple expression of the sub-matrices.

Submatrix based on the subset C

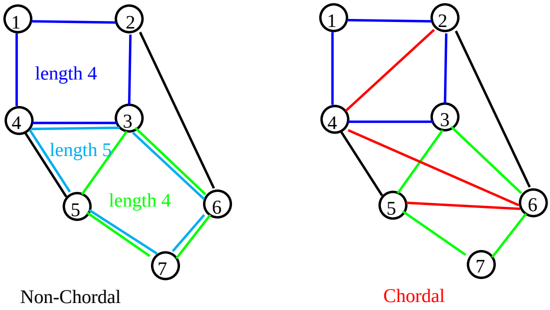
For an index set $C = \{i_1, \ldots, i_l\}$, we define

$$\boldsymbol{X}(C) = \mathrm{matrix}(X_{ij} : i, j \in C)$$

For example,

$$\boldsymbol{X}(\{1,2,4\}) = \begin{pmatrix} X_{11} & X_{12} & X_{14} \\ X_{12} & X_{22} & X_{24} \\ X_{14} & X_{24} & X_{44} \end{pmatrix}, \boldsymbol{X}(\{2,3,4\}) = \begin{pmatrix} X_{22} & X_{23} & X_{24} \\ X_{23} & X_{33} & X_{34} \\ X_{24} & X_{34} & X_{44} \end{pmatrix}$$

Another example for Matrix completion



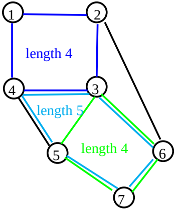
$$
\begin{aligned}
& \boldsymbol{X} \in \mathbb{S}_{+}^{7\times 7} \\
\Leftrightarrow \quad & \boldsymbol{X}(\{1,2,4\}) \in \mathbb{S}_{+}^{3\times 3}, \boldsymbol{X}(\{2,3,4,6\}) \in \mathbb{S}_{+}^{4\times 4}, \\
& \boldsymbol{X}(\{3,4,5,6\}) \in \mathbb{S}_{+}^{4\times 4}, \boldsymbol{X}(\{5,6,7\}) \in \mathbb{S}_{+}^{3\times 3},
\end{aligned}
$$

In this graph, we have four maximal cliques; $\{1,2,4\}, \{2,3,4,6\}, \{3,4,5,6\}, \{5,6,7\}$.

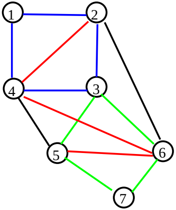
How to obtain maximal cliques

Sparse Cholesky $\boldsymbol{X} = \boldsymbol{U}^T\boldsymbol{U}$

$$\begin{pmatrix} n & 1 & & 1 & & & \\ 1 & n & 1 & & & 1 & \\ & 1 & n & 1 & 1 & 1 & \\ 1 & & 1 & n & 1 & & \\ & & 1 & 1 & n & & 1 \\ & 1 & 1 & & & n & 1 \\ & & & & 1 & 1 & n \end{pmatrix}$$



Non-Chordal



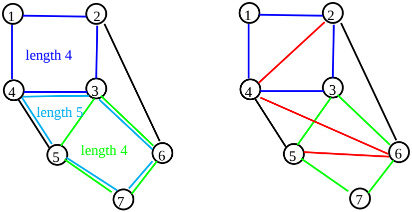
Chordal

How to obtain maximal cliques

Sparse Cholesky $\boldsymbol{X} = \boldsymbol{U}^T\boldsymbol{U}$

$$\begin{pmatrix} n & 1 & & 1 & & & \\ 1 & n & 1 & & & 1 & \\ & 1 & n & 1 & 1 & 1 & \\ 1 & & 1 & n & 1 & & \\ & & 1 & 1 & n & & 1 \\ & 1 & 1 & & & n & 1 \\ & & & & 1 & 1 & n \end{pmatrix}$$

$$= \begin{pmatrix} 2.6 & 0.4 & & 0.4 & & & \\ & 2.6 & 0.4 & \color{red}{-0.1} & & 0.4 & \\ & & 2.6 & 0.4 & 0.4 & 0.3 & \\ & & & 2.6 & 0.3 & \color{red}{-0.1} & \\ & & & & 2.6 & \color{red}{-0.1} & 0.4 \\ & & & & & 2.6 & 0.4 \\ & & & & & & 2.6 \end{pmatrix}^T \begin{pmatrix} 2.6 & 0.4 & & 0.4 & & & \\ & 2.6 & 0.4 & \color{red}{-0.1} & & 0.4 & \\ & & 2.6 & 0.4 & 0.4 & 0.3 & \\ & & & 2.6 & 0.3 & \color{red}{-0.1} & \\ & & & & 2.6 & \color{red}{-0.1} & 0.4 \\ & & & & & 2.6 & 0.4 \\ & & & & & & 2.6 \end{pmatrix}$$



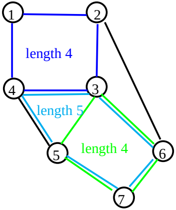
Non-Chordal   Chordal

How to obtain maximal cliques

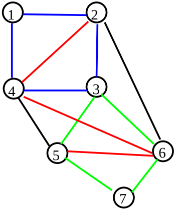
Sparse Cholesky $\boldsymbol{X} = \boldsymbol{U}^T\boldsymbol{U}$

$$\begin{pmatrix} n & 1 & & 1 & & & \\ 1 & n & 1 & & & 1 & \\ & 1 & n & 1 & 1 & 1 & \\ 1 & & 1 & n & 1 & & \\ & & 1 & 1 & n & & 1 \\ & 1 & 1 & & & n & 1 \\ & & & & 1 & 1 & n \end{pmatrix}$$

$$= \begin{pmatrix} 2.6 & 0.4 & & 0.4 & & & \\ & 2.6 & 0.4 & {\color{red}-0.1} & & 0.4 & \\ & & 2.6 & 0.4 & 0.4 & 0.3 & \\ & & & 2.6 & 0.3 & {\color{red}-0.1} & \\ & & & & 2.6 & {\color{red}-0.1} & 0.4 \\ & & & & & 2.6 & 0.4 \\ & & & & & & 2.6 \end{pmatrix}^T \begin{pmatrix} 2.6 & 0.4 & & 0.4 & & & \\ & 2.6 & 0.4 & {\color{red}-0.1} & & 0.4 & \\ & & 2.6 & 0.4 & 0.4 & 0.3 & \\ & & & 2.6 & 0.3 & {\color{red}-0.1} & \\ & & & & 2.6 & {\color{red}-0.1} & 0.4 \\ & & & & & 2.6 & 0.4 \\ & & & & & & 2.6 \end{pmatrix}$$



Non-Chordal



Chordal

1. $C_1 = \{1, 2, 4\}$
2. $C_2 = \{2, 3, 4, 6\}$
3. $C_3 = \{3, 4, 5, 6\}$
4. $\{4, 5, 6\}$ is a subset of C_3 (not maximal)
5. $C_4 = \{5, 6, 7\}$
6. $\{6, 7\}$ is a subset of C_4, $\{7\}$ is a subset of C_4

Matrix Completion

Grone et al, 1984

Assume that the graph is chordal and its maximal cliques are $C_1, \ldots, C_\ell$, and the cliques satisfy the running intersection property.
If $\boldsymbol{X}(C_r) \succeq \boldsymbol{O}$ $(r = 1, \ldots, \ell)$, then we can recover the enter matrix $\boldsymbol{X} \succeq \boldsymbol{O}$ by assigning appropriate values to X_{ij} for $(i, j) \notin \cup_{r=1}^{\ell} C_r \times C_r$.

Instead of handling large matrix $\boldsymbol{X}$, we can solve the SDPs with smaller matrices.
This can save computation time.

Very Effective Case

When the graph is decomposed into smaller cliques, the computation cost to solve the SDP will be smaller.

Very Effective Case

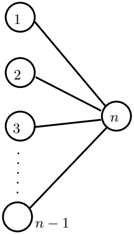
When the graph is decomposed into smaller cliques, the computation cost to solve the SDP will be smaller.

$$\begin{pmatrix} X_{11} & & & & X_{1n} \\ & X_{22} & & & X_{2n} \\ & & \ddots & & \vdots \\ & & & X_{n-1,n-1} & X_{n-1,n} \\ X_{1n} & X_{2n} & \cdots & X_{n-1,n-1} & X_{n-1,n} \end{pmatrix} \succeq \boldsymbol{O}$$

Very Effective Case

When the graph is decomposed into smaller cliques, the computation cost to solve the SDP will be smaller.

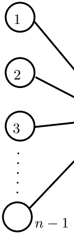
$$\begin{pmatrix} X_{11} & & & & X_{1n} \\ & X_{22} & & & X_{2n} \\ & & \ddots & & \vdots \\ & & & X_{n-1,n-1} & X_{n-1,n} \\ X_{1n} & X_{2n} & \cdots & X_{n-1,n-1} & X_{n-1,n} \end{pmatrix} \succeq \boldsymbol{O}$$



Very Effective Case

When the graph is decomposed into smaller cliques, the computation cost to solve the SDP will be smaller.

$$\begin{pmatrix} X_{11} & & & & X_{1n} \\ & X_{22} & & & X_{2n} \\ & & \ddots & & \vdots \\ & & & X_{n-1,n-1} & X_{n-1,n} \\ X_{1n} & X_{2n} & \cdots & X_{n-1,n-1} & X_{n-1,n} \end{pmatrix} \succeq \boldsymbol{O}$$



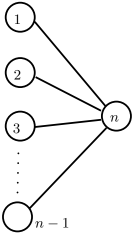
Aggregate sparsity pattern is already chordal extension.
The positive semidefinite condition will be decomposed into

$$\begin{pmatrix} X_{11} & X_{1n} \\ X_{1n} & X_{nn} \end{pmatrix} \succeq \boldsymbol{O}, \begin{pmatrix} X_{22} & X_{2n} \\ X_{2n} & X_{nn} \end{pmatrix} \succeq \boldsymbol{O}, \ldots, \begin{pmatrix} X_{n-1,n-1} & X_{n-1,n} \\ X_{n-1,n} & X_{nn} \end{pmatrix} \succeq \boldsymbol{O}.$$

Very Effective Case

When the graph is decomposed into smaller cliques, the computation cost to solve the SDP will be smaller.

$$\begin{pmatrix} X_{11} & & & & X_{1n} \\ & X_{22} & & & X_{2n} \\ & & \ddots & & \vdots \\ & & & X_{n-1,n-1} & X_{n-1,n} \\ X_{1n} & X_{2n} & \cdots & X_{n-1,n-1} & X_{n-1,n} \end{pmatrix} \succeq \boldsymbol{O}$$



Aggregate sparsity pattern is already chordal extension.
The positive semidefinite condition will be decomposed into

$$\begin{pmatrix} X_{11} & X_{1n} \\ X_{1n} & X_{nn} \end{pmatrix} \succeq \boldsymbol{O}, \begin{pmatrix} X_{22} & X_{2n} \\ X_{2n} & X_{nn} \end{pmatrix} \succeq \boldsymbol{O}, \ldots, \begin{pmatrix} X_{n-1,n-1} & X_{n-1,n} \\ X_{n-1,n} & X_{nn} \end{pmatrix} \succeq \boldsymbol{O}.$$

The cost of PDIPM is roughly $O(m^2n^2 + mn^3 + m^3)$.
This will be reduced to $O(m^2(2^2n) + m(2^3n) + m^3)$.
If $n \geq 1000$, the effect is prominent.

Index Number

SDP relaxation for SNL

$$\min \ \boldsymbol{O} \bullet \boldsymbol{Z} \qquad \text{s.t.} \quad \boldsymbol{A}_k \bullet \boldsymbol{Z} = b_k \ (k \in \Lambda), \quad \boldsymbol{Z} \succeq \boldsymbol{O}$$

Index Number

SDP relaxation for SNL

$$\min\ \boldsymbol{O} \bullet \boldsymbol{Z} \quad \text{s.t.} \quad \boldsymbol{A}_k \bullet \boldsymbol{Z} = b_k\ (k \in \Lambda), \quad \boldsymbol{Z} \succeq \boldsymbol{O}$$

First, we add the index number to the rows/columns of the variable matrix $\boldsymbol{Z}$.

$$\boldsymbol{Z} = \left(\begin{array}{c|c} \boldsymbol{I}_d & \boldsymbol{X} \\ \hline \boldsymbol{X}^T & \boldsymbol{Y} \end{array}\right) = \begin{array}{c} \\ 10 \\ 20 \\ \vdots \\ d0 \\ - \\ *1 \\ *2 \\ \vdots \\ *n \end{array} \begin{array}{c} \begin{array}{cccccccccc} 10 & 20 & \ldots & d0 & | & *1 & *2 & \ldots & *n \end{array} \\ \left(\begin{array}{cccccccccc} 1 & & & & | & X_{11} & X_{21} & \ldots & X_{n1} \\ & 1 & & & | & X_{12} & X_{22} & \ldots & X_{n2} \\ & & \ddots & & | & \vdots & \vdots & \ddots & \vdots \\ & & & 1 & | & X_{1d} & X_{2d} & \ldots & X_{nd} \\ - & - & - & - & - & - & - & - & - \\ X_{11} & X_{12} & \cdots & X_{1d} & | & Y_{11} & Y_{12} & \cdots & Y_{1n} \\ X_{21} & X_{22} & \cdots & X_{2d} & | & Y_{12} & Y_{22} & \cdots & Y_{2n} \\ \vdots & \vdots & \ddots & \vdots & | & \vdots & \vdots & \ddots & \vdots \\ X_{n1} & X_{n2} & \cdots & X_{nd} & | & Y_{1n} & Y_{2n} & \cdots & Y_{nn} \end{array}\right) \end{array}$$

Index Number

SDP relaxation for SNL

$$\min\ \boldsymbol{O} \bullet \boldsymbol{Z} \qquad \text{s.t.} \quad \boldsymbol{A}_k \bullet \boldsymbol{Z} = b_k \ (k \in \Lambda), \quad \boldsymbol{Z} \succeq \boldsymbol{O}$$

First, we add the index number to the rows/columns of the variable matrix $\boldsymbol{Z}$.

$$\boldsymbol{Z} = \left(\begin{array}{c|c} \boldsymbol{I}_d & \boldsymbol{X} \\ \hline \boldsymbol{X}^T & \boldsymbol{Y} \end{array}\right) = \begin{array}{cc} & \begin{array}{cccc|cccc} 10 & 20 & \ldots & d0 & *1 & *2 & \ldots & *n \end{array} \\ \begin{array}{c} 10 \\ 20 \\ \vdots \\ d0 \\ - \\ *1 \\ *2 \\ \vdots \\ *n \end{array} & \left(\begin{array}{cccc|cccc} 1 & & & & X_{11} & X_{21} & \ldots & X_{n1} \\ & 1 & & & X_{12} & X_{22} & \ldots & X_{n2} \\ & & \ddots & & \vdots & \vdots & \ddots & \vdots \\ & & & 1 & X_{1d} & X_{2d} & \ldots & X_{nd} \\ - & - & - & - & - & - & - & - \\ X_{11} & X_{12} & \cdots & X_{1d} & Y_{11} & Y_{12} & \cdots & Y_{1n} \\ X_{21} & X_{22} & \cdots & X_{2d} & Y_{12} & Y_{22} & \cdots & Y_{2n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ X_{n1} & X_{n2} & \cdots & X_{nd} & Y_{1n} & Y_{2n} & \cdots & Y_{nn} \end{array}\right) \end{array}$$

We focus on the aggregated sparsity of the submatrix $\boldsymbol{Y}$.

The Chordal extension of SNL

The aggregated sparsity of the submatrix $\boldsymbol{Y}$ corresponds to the base network of SNL.

The Chordal extension of SNL

The aggregated sparsity of the submatrix $\boldsymbol{Y}$ corresponds to the base network of SNL.
If sensors p and q are connected, only Y_{pp}, Y_{pq}, Y_{qq} appear in the aggregated sparsity.

$$||\boldsymbol{x}_p - \boldsymbol{x}_q||^2 = d_{pq}^2 \Rightarrow \begin{pmatrix} \boldsymbol{O} & \boldsymbol{O} \\ \boldsymbol{O} & \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \end{pmatrix} \bullet \begin{pmatrix} \boldsymbol{I}_d & \boldsymbol{X} \\ \boldsymbol{X}^T & \boldsymbol{Y} \end{pmatrix} = d_{pq}^2$$

The Chordal extension of SNL

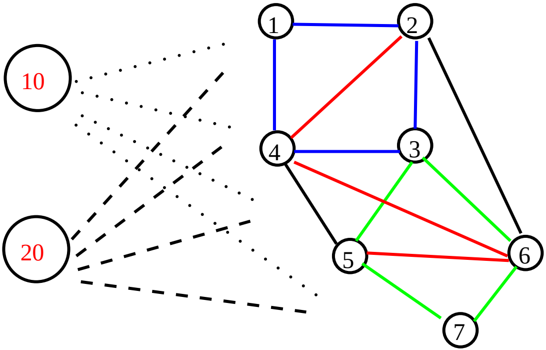
The aggregated sparsity of the submatrix $\boldsymbol{Y}$ corresponds to the base network of SNL.
If sensors p and q are connected, only Y_{pp}, Y_{pq}, Y_{qq} appear in the aggregated sparsity.

$$||\boldsymbol{x}_p - \boldsymbol{x}_q||^2 = d_{pq}^2 \Rightarrow \begin{pmatrix} \boldsymbol{O} & \boldsymbol{O} \\ \boldsymbol{O} & \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \end{pmatrix} \bullet \begin{pmatrix} \boldsymbol{I}_d & \boldsymbol{X} \\ \boldsymbol{X}^T & \boldsymbol{Y} \end{pmatrix} = d_{pq}^2$$

The maximal cliques of $\boldsymbol{Y}$ are $C_1, \ldots, C_\ell$, then the maximal cliques of $\boldsymbol{Z}$ are $\widetilde{C}_1, \ldots, \widetilde{C}_\ell$ where

$$\widetilde{C}_r = \underbrace{\{10, \ldots, d0\}}_{\text{from } \boldsymbol{X}} \cup \underbrace{\{*p : p \in C_r\}}_{\text{from } \boldsymbol{Y}}$$

An example of the chordal extension



The maximal cliques

$$\{1,2,4\},\{2,3,4,6\},\{3,4,5,6\},\{5,6,7\}$$

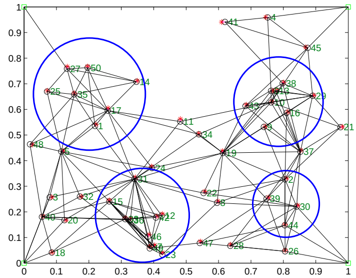
in the base network are extended to

$$\{10,20,*1,*2,*4\},\{10,20,*2,*3,*4,*6\},\{10,20,*3,*4,*5,*6\},\{10,20,*5,*6,*7\},$$

in the converted SDP.

Cliques and quasi-cliques in SNL

If the base network of SNL is almost chordal graph, the matrix completion works well.
This happens in many applications, because the sensors in the short range are often connected and such connections involve many cliques and quasi-cliques.



O : Sensor true locations vs * : the ones computed by SFSDP + the refinepositions funct. l

Decomposed SDP

Original SDP

$$\min \;:\; 0 \quad \text{s.t.}: \begin{array}{ll} Y_{pp} - 2Y_{pq} + Y_{qq} = d_{pq}^2 & \forall (p,q) \in \mathcal{N}_x \\ Y_{pp} - 2\boldsymbol{a}_r^T [\boldsymbol{X}]_{*p} + \|\boldsymbol{a}_r\|^2 = d_{pr}^2 & \forall (p,r) \in \mathcal{N}_a \\ \begin{pmatrix} \boldsymbol{I}_d & \boldsymbol{X} \\ \boldsymbol{X}^T & \boldsymbol{Y} \end{pmatrix} \succeq \boldsymbol{O} & \end{array}$$

Decomposed SDP

$$\min \;:\; 0 \quad \text{s.t.}: \begin{array}{ll} Y_{pp} - 2Y_{pq} + Y_{qq} = d_{pq}^2 & \forall (p,q) \in \mathcal{N}_x \\ Y_{pp} - 2\boldsymbol{a}_r^T [\boldsymbol{X}]_{*p} + \|\boldsymbol{a}_r\|^2 = d_{pr}^2 & \forall (p,r) \in \mathcal{N}_a \\ \begin{pmatrix} \boldsymbol{I}_d & \boldsymbol{X}(:, C_r) \\ \boldsymbol{X}^T(:, C_r) & \boldsymbol{Y}(C_r) \end{pmatrix} \succeq \boldsymbol{O} & r = 1, \ldots, \ell \end{array}$$

If the sizes of $C_1, \ldots, C_\ell$ are remarkably smaller than n, the decomposed SDP can be solved in a shorter time.

Decomposed SDP from Noisy SNL

Noisy SNL

$$
\begin{array}{rcl}
\min & : & \Sigma_{(p,q)\in\mathcal{N}_x}(\xi_{pq}^+ + \xi_{pq}^-) + \Sigma_{(p,r)\in\mathcal{N}_a}(\xi_{pr}^+ + \xi_{pr}^-) \\
\text{s.t.} & : & d_{pq}^2 = ||x_p - x_q||^2 + \xi_{pq}^+ - \xi_{pq}^- \ \forall (p,q)\in\mathcal{N}_x \\
& & d_{pr}^2 = ||x_p - a_r||^2 + \xi_{pr}^+ - \xi_{pr}^- \ \forall (p,r)\in\mathcal{N}_a \\
& & \xi_{pq}^+, \xi_{pq}^- \geq 0 \ \forall (p,q)\in\mathcal{N}_x \\
& & \xi_{pr}^+, \xi_{pr}^- \geq 0 \ \forall (p,r)\in\mathcal{N}_a
\end{array}
$$

Decomposed SDP from Noisy SNL

$$
\begin{array}{rcll}
\min & : & \Sigma_{(p,q)\in\mathcal{N}_x}(\xi_{pq}^+ + \xi_{pq}^-) + \Sigma_{(p,r)\in\mathcal{N}_a}(\xi_{pr}^+ + \xi_{pr}^-) & \\
& & Y_{pp} - 2Y_{pq} + Y_{qq} + \xi_{pq}^+ - \xi_{pq}^- = d_{pq}^2 & \forall (p,q)\in\mathcal{N}_x \\
& & Y_{pp} - 2\boldsymbol{a}_r^T[\boldsymbol{X}]_{*p} + ||\boldsymbol{a}_r||^2 + \xi_{pr}^+ - \xi_{pr}^- = d_{pr}^2 & \forall (p,r)\in\mathcal{N}_a \\
\text{s.t.} & : & \xi_{pq}^+, \xi_{pq}^- \geq 0 & \forall (p,q)\in\mathcal{N}_x \\
& & \xi_{pr}^+, \xi_{pr}^- \geq 0 & \forall (p,r)\in\mathcal{N}_a \\
& & \begin{pmatrix} \boldsymbol{I}_d & \boldsymbol{X}(:,C_r) \\ \boldsymbol{X}^T(:,C_r) & \boldsymbol{Y}(C_r) \end{pmatrix} \succeq \boldsymbol{O} & r = 1,\ldots,\ell
\end{array}
$$

5. Numerical Results

Setting 1 of 3

- Computing Environment
  - (for *small* SNL) 2.8GHz Quad-Core Intel Xeon with 4GB memory
  - (for *large* SNL) 2.8GHz Quad-Core Intel Core i7 with 16GB memory
- SDP Solver
  SDPA 7.3.1 [http://sdpa.sourceforge.net]
- Accuracy Measure
  RMSD (root mean square distance)

$$\left(\frac{1}{n}\sum_{p=1}^{n}||\boldsymbol{x}_p - \boldsymbol{a}_p||^2\right)^{1/2}$$

  where $\boldsymbol{x}_p$ is the computed location of pth sensor and $\boldsymbol{a}_p$ is its true location.
- Computation time unit in the results is *second*.

Setting 2 of 3

- Sensors (n)
  $n = 1000, 3000, 5000$
- Anchors (m)
  - $m = 4$ or 8 : 4 corners in $[0,1]^2$ or 8 corners in $[0,1]^3$.
  - $m = 5\% \times n$ or $10\% \times n$: randomly generated locations
- Radiorange ρ for $[0,1]^2$ in 2D
  - $\rho = 0.1$
  - $\rho = \sqrt{10/n}$ (the square $\rho \times \rho$ contains 10 sensor on average)

  For $[0,1]^3$ in 3D, $\rho = 0.25$ or $\rho = (15/n)^{1/3}$
- For noisy case, noisy factor σ is chosen from 0.0, 0.1 and 0.2. For the true locations $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$, the input distances are

$$d_{pq} = \max\{(1 + \sigma\epsilon_{pq}), 0.1\}||\boldsymbol{a}_p - \boldsymbol{a}_q|| \quad ((p,q) \in \mathcal{N}_x^\rho$$
$$d_{pr} = \max\{(1 + \sigma\epsilon_{pr}), 0.1\}||\boldsymbol{a}_p - \boldsymbol{a}_r|| \quad ((p,r) \in \mathcal{N}_a^\rho$$

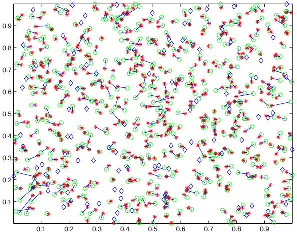
where $\epsilon_{pq}, \epsilon_{pr}$ follow the standard normal distribution $N(0,1)$.

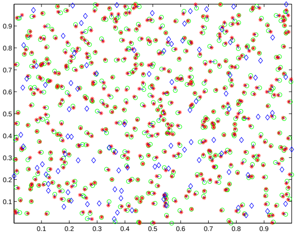
Setting 3 of 3

- Refinement
  We apply the gradient method developed by Toh (a local method) by setting the solution of SDP relaxation as its initial point.

$$\min \quad : \quad \sum_{(p,q)\in\mathcal{N}_x} \left|\|x_p - x_q\|^2 - d_{pq}^2\right| + \sum_{(p,r)\in\mathcal{N}_a} \left|\|x_p - a_r\|^2 - d_{pr}^2\right|$$

SDP solution $\Rightarrow$ with Gradient method





Comparison between three SDP formulations

- FSDP (Full SDP; without exploiting sparsity) Biswas-Ye
- ESDP (Edge Based SDP) Wang-Zheng-Boyd-Ye
- SFSDP (Sparse variant Full SDP)

Numerical Results on 2D

n	m	ρ	σ	Form	RMSD SDP	RMSD w.Grad	Time SDPA	Time Grad	Time Total
1000	4	0.1	0.0	FSDP	5.5e-5	2.1e-5	45.4	0.3	48.8
				SFSDP	5.8e-5	2.5e-5	12.6	0.3	17.8
			0.1	FSDP	4.5e-2	1.0e-2	157.2	14.9	179.5
				SFSDP	4.5e-2	1.0e-2	18.3	12.5	39.8
1000	100	0.1	0.0	FSDP	5.0e-4	1.0e-5	53.8	0.2	56.9
				SFSDP	4.9e-5	7.1e-6	2.3	0.3	7.3
			0.1	FSDP	1.7e-2	7.1e-3	308.6	1.5	317.3
				SFSDP	1.7e-2	7.1e-3	5.1	5.0	18.9

- SFSDP is much faster than FSDP.
- The RMSDs of FSDP and SFSDP are almost same.
- Exact distance information shortens computation time.
  (Objective function is 0; hence the SDP is decomposed into smaller cliques.)
- In SFSDP, less anchors leads to longer computation.
  (The freedom in choice of the edges is reduced, and smaller cliques is difficult to expect.)

Results on Middle Size SNL ($n = 3000$)

$m = 300$

Form	RMSD	Time SDPA	Time Total
$\rho = 0.1, \sigma = 0.0$			
ESDP	1.7e-3	68.0	359.2
SFSDP	2.3e-7	6.2	36.9
$\rho = 0.1, \sigma = 0.2$			
ESDP	8.1e-3	70.1	348.0
SFSDP	4.8e-3	15.1	98.8
$\rho = \sqrt{10/n} \sim 0.058, \sigma = 0.0$			
ESDP	8.1e-4	69.3	241.2
SFSDP	4.8e-3	21.5	88.9
$\rho = \sqrt{10/n} \sim 0.058, \sigma = 0.2$			
ESDP	7.9e-3	47.0	216.2
SFSDP	4.4e-3	21.5	88.9

$m = 4$ at corners

Form	RMSD	Time SDPA	Time Total
$\rho = 0.1, \sigma = 0.0$			
ESDP	1.2e-2	84.6	274.0
SFSDP	6.7e-6	36.8	74.2
$\rho = 0.1, \sigma = 0.2$			
ESDP	3.3e-2	76.3	263.4
SFSDP	9.3e-3	63.3	176.7
$\rho = \sqrt{10/n} \sim 0.058, \sigma = 0.0$			
ESDP	7.2e-3	258.3	71.1
SFSDP	7.6e-5	167.5	206.6
$\rho = \sqrt{10/n} \sim 0.058, \sigma = 0.2$			
ESDP	3.3e-2	77.2	265.4
SFSDP	1.1e-2	176.4	312.4

- SFSDP attains smaller RMSD compared to ESDP.
- SFSDP is faster than ESDP in most cases.
- In the case $m = 4$, $\rho = \sqrt{10/n}$ and $\sigma = 0.2$, SFSDP becomes slower than ESDP. (The situation with less anchors and noise requires long computation for numerical accuracy.)

Results on Large Size SNL

n	m	ρ	σ	RMSD SDPA	RMSD w.Grad	Time SDPA	Time Grad	Time Total
20000	2000	0.1	0.0	1.4e-6	3.0e-7	93.4	0.7	326.9
			0.2	1.9e-2	4.4e-3	148.2	23.5	882.1
		$\sqrt{10/n}$	0.0	1.1e-4	4.0e-6	466.2	4.4	708.0
		~ 0.022	0.2	6.2e-3	1.5e-3	237.6	41.4	773.1
20000	4	0.1	0.0	4.0e-5	6.9e-6	182.9	2.0	469.2
			0.2	6.6e-2	1.0e-2	402.6	146.0	1150.5
		$\sqrt{10/n}$	0.0	Out of memory				
		~ 0.022	0.2	Out of memory				

- SFSDP can handle large size SNL ($n \geq 10000$).

SFSDP can handle 3D data

Mathematical formulation is same as 2D case. But, its computational cost becomes huge.

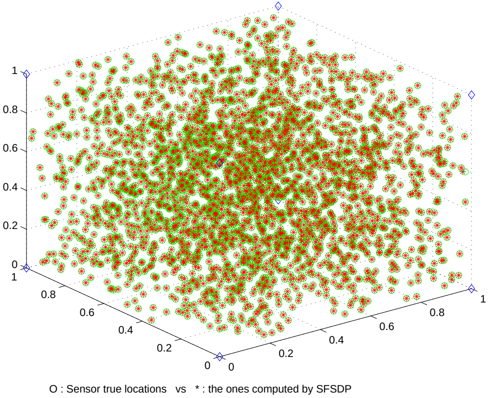


Figure: 3000 sensors, 8 anchors (corners), noise 0.0

Results on 3D SNL

n	m	ρ	σ	RMSD SDPA	RMSD w.Grad	Time SDPA	Time Grad	Time Total
3000	300	0.25	0.0	6.2e-6	9.8e-7	12.1	0.4	48.1
			0.2	6.1e-2	1.7e-2	20.3	26.3	144.7
		$(10/n)^{1/3}$	0.0	1.0e-4	2.9e-6	49.3	1.2	87.3
		~ 0.171	0.2	4.9e-2	1.6e-2	51.6	25.3	162.0
3000	8	0.25	0.0	1.0e-4	7.5e-6	368.4	1.2	413.0
			0.2	1.4e-1	2.6e-2	422.2	45.9	563.6
		$\sqrt{10/n}$	0.0	Out of memory				
		~ 0.171	0.2	Out of memory				
5000	500	0.25	0.0	1.4e-6	4.2e-7	17.5	0.5	112.2
			0.2	6.1e-2	1.6e-2	37.5	31.8	331.8
		$(10/n)^{1/3}$	0.0	8.8e-5	2.1e-6	194.5	2.8	295.4
		~ 0.144	0.2	4.2e-2	1.2e-2	170.1	54.4	452.2
5000	250	0.25	0.0	1.8e-5	7.7e-7	18.7	1.2	117.3
			0.2	6.3e-2	1.7e-2	39.3	42.6	348.9
		$\sqrt{10/n}$	0.0	Out of memory				
		~ 0.144	0.2	Out of memory				

Same tendency as 2D can be observed in 3D case.

- Less anchor is difficult.
- Less edge (due to short ρ) requires more computation resources.

Anchor-free 3D SNL

Some applications (like protein conformation) do not have any anchors.
We solve SNLs of this type by the following steps.

1. Fix $d + 1$ sensors which are connected each other, as anchors.
2. Apply SFSDP and obtain SDP solution.
3. Apply Gradient method.
4. Apply *parallel translation*, *reflection* and *rotation* if the true locations are known.

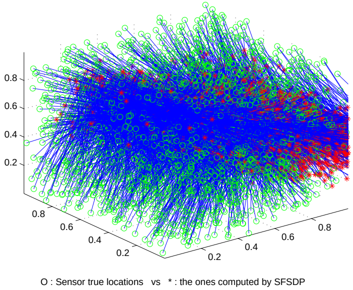


Figure: SDP solution

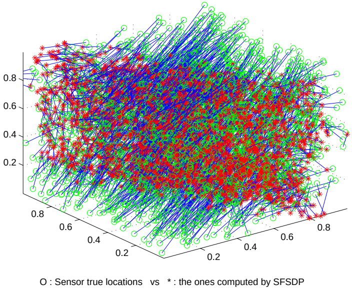


Figure: After Gradient method

Parallel Translation, Reflection and Rotation

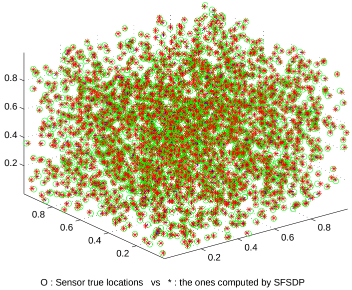
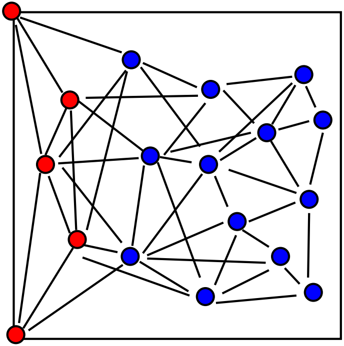


Figure: After Parallel Translation, Reflection and Rotation

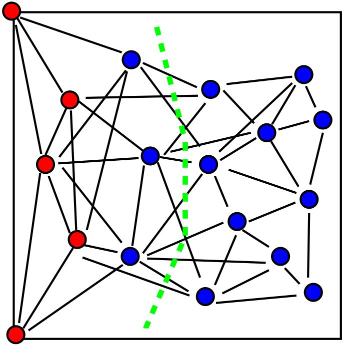
- We apply procusters.m developed by Toh.
  This function find the linear transformation T which minimizes
  $$\sum_{p=1}^{n} ||T(\boldsymbol{x}_p) - \boldsymbol{a}_p||^2$$
  where $\boldsymbol{x}_p$ is the SDP solution and $\boldsymbol{a}_p$ is the true location.
- We can obtain enough accuracy.

6. Future Works (Shortest-Path Propagation Method)

Basic Idea of Different Approach

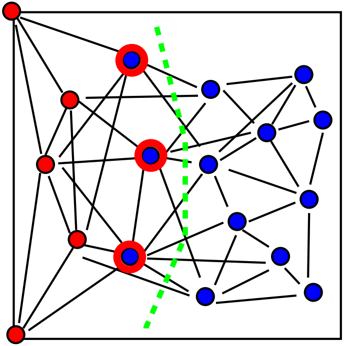


Basic Idea of Different Approach



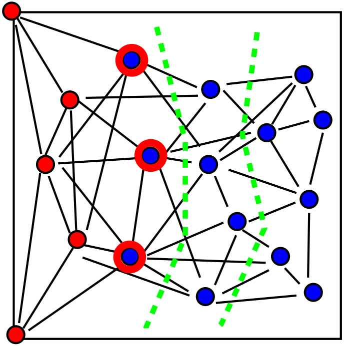
1. We fix only the three sensors by the gradient method or SDP relaxation.

Basic Idea of Different Approach



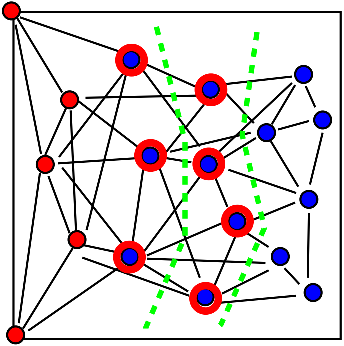
1. We fix only the three sensors by the gradient method or SDP relaxation.
2. The three sensors are changed as new anchors.
   (We propagate the region of anchors.)

Basic Idea of Different Approach



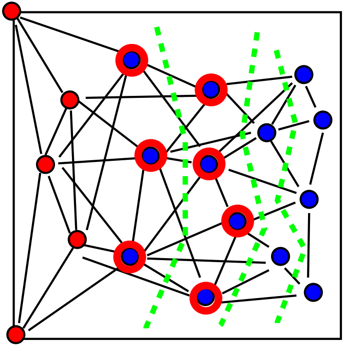
1. We fix only the three sensors by the gradient method or SDP relaxation.
2. The three sensors are changed as new anchors.
   (We propagate the region of anchors.)
3. We iterate this procedure.

Basic Idea of Different Approach



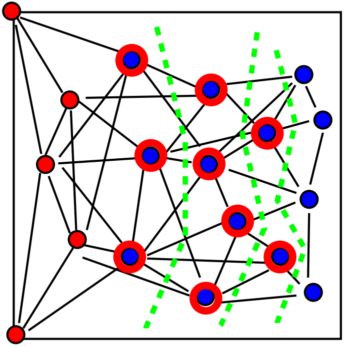
1. We fix only the three sensors by the gradient method or SDP relaxation.
2. The three sensors are changed as new anchors.
   (We propagate the region of anchors.)
3. We iterate this procedure.

Basic Idea of Different Approach



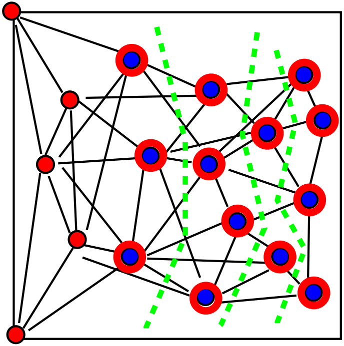
1. We fix only the three sensors by the gradient method or SDP relaxation.
2. The three sensors are changed as new anchors.
   (We propagate the region of anchors.)
3. We iterate this procedure.

Basic Idea of Different Approach



1. We fix only the three sensors by the gradient method or SDP relaxation.
2. The three sensors are changed as new anchors.
   (We propagate the region of anchors.)
3. We iterate this procedure.

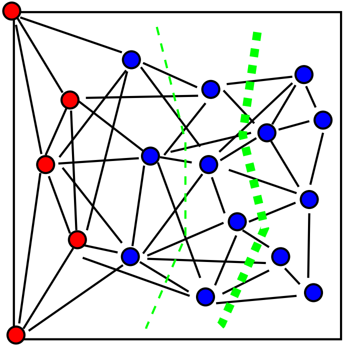
Basic Idea of Different Approach



1. We fix only the three sensors by the gradient method or SDP relaxation.
2. The three sensors are changed as new anchors.
   (We propagate the region of anchors.)
3. We iterate this procedure.

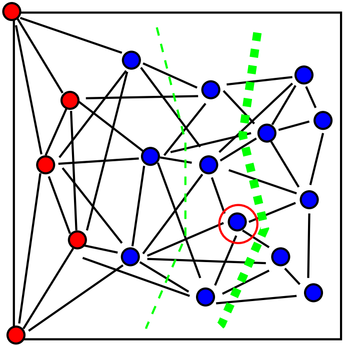
Distance Estimation by Shortest Path

- To reduce the iteration number, we fix the sensors with in two or three steps.
- We use the shortest path to infer the distances.



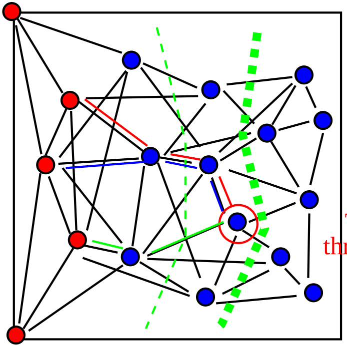
Distance Estimation by Shortest Path

- To reduce the iteration number, we fix the sensors with in two or three steps.
- We use the shortest path to infer the distances.



Distance Estimation by Shortest Path

- To reduce the iteration number, we fix the sensors with in two or three steps.
- We use the shortest path to infer the distances.

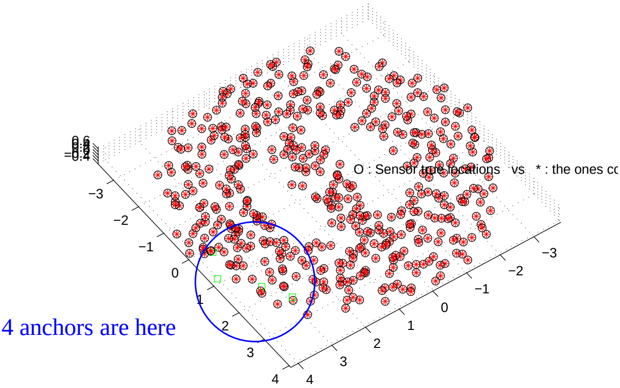


To infer this senser
three shortest paths are used.

Test Sample (Donut)

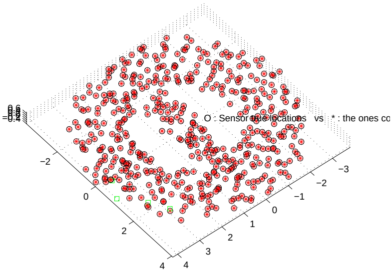
A donut image is illustrated in 3D.
(This sample was made by Cucuringu).

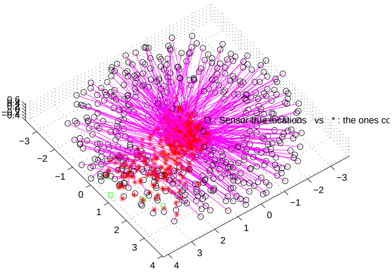
735 sensors + 4 anchors



Numerical Results

We can obtain good accuracy for proteins. But there are hard problems.





	SFSDP	Shortest Path
RMSD	3.54e-4	2.49
Time	322.36 (SDPA 320.97 + Grad 0.09)	12.84 (Shortest 0.04 + Grad 9.70)

- Shortest Path Propagation is too inaccurate in this example.
- All the anchors are located near one corner.
- Distance Estimation is not well.
- (Computation time is short; there is a room to improve.)

For further improvements

- Select good step-number.
- Select more stable shortest path.
- Estimate better shortest path.
  (Need not to be exact, because it is NOT straight.)

Conclusion & Future works

1. Sensor Network Localization Problem
2. Biswas-Ye's SDP relaxation
3. Exploiting Sparsity and SFSDP
4. SFSDP can solve large-scale SNLs

SFSDP is available at

http://www.is.titech.ac.jp/~kojima/SFSDP/SFSDP.html

- Further improvements on Shortest Path Propagation.

Thank you very much for your attention.
謝謝