Geometric Spanner Networks

Spring 2014
Textbook:


---

Course Outline

Textbook

Introduction

Algorithms Review

- Greedy Algorithm (Org. and Imp.)
- Apx. Greedy Algorithm (Ordered) $\Theta$-Graph
- Algorithm (Sink and Skip-list spanner)
- Sink Spanner
- WSPD-based Algorithm

Theoretical bounds

Applications

- Designing approximation algorithms with spanners
- Metric space searching
- Protein Visualization

Research Topics
Example of networks

London Underground Network
Geometric Spanner Networks

Course Outline

Textbook

Introduction

Algorithms Review

Greedy Algorithm (Org. and Imp.)
Apx. Greedy Algorithm (Ordered) $\Theta$-Graph Algorithm (Sink and Skip-list spanner)
Sink Spanner
WSPD-based Algorithm

Theoretical bounds

Applications

Designing approximation algorithms with spanners
Metric space searching
Protein Visualization

Research Topics

Example of networks

Ad hoc Network
Example of networks

Yeast Protein Interaction Network

Geometric Spanner Networks

Course Outline

Introduction

Algorithms Review

Greedy Algorithm (Org. and Imp.)
Apx. Greedy Algorithm (Ordered) \( \Theta \)-Graph Algorithm (Sink and Skip-list spanner)
Sink Spanner
WSPD-based Algorithm

Theoretical bounds

Applications

Designing approximation algorithms with spanners
Metric space searching
Protein Visualization

Research Topics
Example of networks

A Social Network (Les Miserables characters)
Geometric Network

Course Outline
Textbook

Introduction

Algorithms Review
- Greedy Algorithm (Org. and Imp.)
- Apx. Greedy Algorithm (Ordered) $\Theta$-Graph Algorithm (Sink and Skip-list spanner)
- Sink Spanner
- WSPD-based Algorithm

Theoretical bounds

Applications
- Designing approximation algorithms with spanners
- Metric space searching
- Protein Visualization

Research Topics
Geometric Network

Course Outline

Textbook

Introduction

Algorithms Review

Greedy Algorithm (Org. and Imp.)
Apx. Greedy Algorithm (Ordered) \( \Theta \)-Graph Algorithm (Sink and Skip-list spanner)
Sink Spanner
WSPD-based Algorithm

Theoretical bounds

Applications

Designing approximation algorithms with spanners
Metric space searching
Protein Visualization

Research Topics
Geometric Network

Course Outline

Textbook

Introduction

Algorithms Review

Greedy Algorithm (Org. and Imp.)

Apx. Greedy Algorithm (Ordered) $\Theta$-Graph Algorithm (Sink and Skip-list spanner)

Sink Spanner

WSPD-based Algorithm

Theoretical bounds

Applications

Designing approximation algorithms with spanners

Metric space searching

Protein Visualization

Research Topics
Geometric Network

Course Outline

Textbook

Introduction

Algorithms Review
- Greedy Algorithm (Org. and Imp.)
- Apx. Greedy Algorithm (Ordered $\Theta$-Graph Algorithm (Sink and Skip-list spanner))
- Sink Spanner
- WSPD-based Algorithm

Theoretical bounds

Applications
- Designing approximation algorithms with spanners
- Metric space searching
- Protein Visualization

Research Topics
**Geometric Network**

Weighted undirected graph \( G(V, E) \) s.t.

\[
\begin{align*}
V & \subset \mathbb{R}^d. \\
\forall e = (u, v) \in E, \ wt(e) & = |uv|.
\end{align*}
\]
Network Quality

Driving distance: 256 km. Actual distance: 198 km.

Driving distance \[\frac{256}{198} \approx 1.27\].
Driving distance: 180 km.  Actual distance: 136 km.

\[
\frac{\text{Driving distance}}{\text{Actual distance}} = 1.32.
\]
Network Quality

Driving distance: 143 km. Actual distance: 100 km.

\[ \frac{\text{Driving distance}}{\text{Actual distance}} = 1.43. \]
Dilation (stretch factor)

- between a pair of vertices = 
  \[
  \frac{\text{Distance in the graph}}{\text{Euclidean distance}}
  \]

- of a network = maximum dilation between all pairs.

\textit{t-spanner}

A network with dilation at most \( t \), or

\[ \forall u, v \in V, \text{ there is a path between } u \text{ and } v \text{ of length} \leq t \times |uv|. \] (\textit{t-path})
Network Quality

Dilation (stretch factor)
- between a pair of vertices = Distance in the graph
  - Euclidean distance
- of a network = maximum dilation between all pairs.

\( t \)-spanner
A network with dilation at most \( t \), or
\( \forall u, v \in V \), there is a path between \( u \) and \( v \) of length
\( \leq t \times |uv| \).
(t-path)
Network Quality

Dilation (stretch factor)

- between a pair of vertices = Distance in the graph / Euclidean distance
- of a network = maximum dilation between all pairs.

\[ t \text{-spanner} \]
A network with dilation at most \( t \), or
\[ \forall u, v \in V, \text{ there is a path between } u \text{ and } v \text{ of length } \leq t \times |uv|. \] (\( t \)-path)
Dilation (stretch factor)

- between a pair of vertices = Distance in the graph
  Euclidean distance
- of a network = maximum dilation between all pairs.

\[ t\text{-spanner} \]

A network with dilation at most \( t \), or
\[ \forall u, v \in V, \text{ there is a path between } u \text{ and } v \text{ of length } \leq t \times |uv|. \]
Network Quality

Dilation (stretch factor)

- between a pair of vertices = \[
\frac{\text{Distance in the graph}}{\text{Euclidean distance}}
\]

- of a network = maximum dilation between all pairs.

\(t\)-spanner

A network with dilation at most \(t\), or
\[\forall u, v \in V, \text{ there is a path between } u \text{ and } v \text{ of length } \leq t \times |uv|. \]

\(t\)-path
(1 + \varepsilon)-Spanners approximate the complete graphs with error \varepsilon.
Example

10-spanner for 532 US-cities
Example

5-spanner for 532 US-cities

Geometric Spanner Networks

Course Outline
Textbook

Introduction

Algorithms Review
Greedy Algorithm (Org. and Imp.)
Apx. Greedy Algorithm
(Ordered) $\Theta$-Graph Algorithm (Sink and Skip-list spanner)
Sink Spanner
WSPD-based Algorithm

Theoretical bounds

Applications
Designing approximation algorithms with spanners
Metric space searching
Protein Visualization

Research Topics
Geometric Spanner Networks

Course Outline
Textbook
Introduction
Algorithms Review
Greedy Algorithm (Org. and Imp.)
Apx. Greedy Algorithm (Ordered) $\Theta$-Graph Algorithm (Sink and Skip-list spanner)
Sink Spanner
WSPD-based Algorithm
Theoretical bounds
Applications
Designing approximation algorithms with spanners
Metric space searching
Protein Visualization
Research Topics

Example

3-spanner for 532 US-cities
Example

2-spanner for 532 US-cities
Example

1.5-spanner for 532 US-cities
How to compute a good spanner?

Given a set $V$ and $t > 1$

Quality measurement:
- Number of edges (size)
- Weight (compared with MST)
- Maximum degree
- Diameter

Sparse $t$-Spanner
How to compute a good spanner?

Given a set $V$ and $t > 1$

Quality measurement:
- Number of edges (size)
- Weight (compared with MST)
- Maximum degree
- Diameter
How to compute a good spanner?

Constructing sparse t-spanners:

- **Greedy** (Bern (1989) and Althöfer et al. (1993)).
- **Θ-graph** (Clarkson (1987) and Keil (1988)).
- **Ordered Θ-graph** (Bose et. al. (2004)).
- **Well-Separated Pair Decomposition** (Arya et. al. (1995)).
(Org.) Greedy Algorithm

(2, 8) (5, 8)

(5, 7)

(1, 5)

(4, 3) (8, 3)

(6, 1)
(Org.) Greedy Algorithm

\((1, 5)\)

\((2, 8)\) (5, 8)

\((4, 3)\) \((8, 3)\)

\((6, 1)\)
(Org.) Greedy Algorithm

(2, 8)   (5, 8)

(5, 7)

(1, 5)   (4, 3)   (8, 3)

(6, 1)

Geometric Spanner Networks

Course Outline

Textbook

Introduction

Algorithms Review

Greedy Algorithm (Org. and Imp.)

Apx. Greedy Algorithm

(Ordered) \( \Theta \)-Graph Algorithm (Sink and Skip-list spanner)

Sink Spanner

WSPD-based Algorithm

Theoretical bounds

Applications

Designing approximation algorithms with spanners

Metric space searching

Protein Visualization

Research Topics
Geometric Spanner Networks

Course Outline

Textbook

Introduction

Algorithms Review

Greedy Algorithm (Org. and Imp.)
Apx. Greedy Algorithm
(Ordered) $\Theta$-Graph Algorithm (Sink and Skip-list spanner)
Sink Spanner
WSPD-based Algorithm

Theoretical bounds

Applications

Designing approximation algorithms with spanners
Metric space searching
Protein Visualization

Research Topics

(Org.) Greedy Algorithm

(2, 8)  (5, 8)
(5, 7)

(1, 5)

(4, 3)  (6, 1)
(8, 3)

(5, 8)
(Org.) Greedy Algorithm

(2, 8) - (5, 7) - (5, 8)

(1, 5) - (4, 3) - (6, 1) - (8, 3)

Course Outline
Textbook
Introduction
Algorithms Review
Greedy Algorithm (Org. and Imp.)
Apx. Greedy Algorithm
(Ordered) $\Theta$-Graph Algorithm (Sink and Skip-list spanner)
Sink Spanner
WSPD-based Algorithm
Theoretical bounds
Applications
Designing approximation algorithms with spanners
Metric space searching
Protein Visualization
Research Topics
(Org.) Greedy Algorithm

(2, 8) (5, 8)
(5, 7)
(1, 5)
(4, 3) (8, 3)
(6, 1)
(Org.) Greedy Algorithm

Introduction

Algorithms Review

Greedy Algorithm (Org. and Imp.)

Apx. Greedy Algorithm
(Ordered) Θ-Graph Algorithm (Sink and Skip-list spanner)

Sink Spanner

WSPD-based Algorithm

Theoretical bounds

Applications

Designing approximation algorithms with spanners

Metric space searching

Protein Visualization

Research Topics
(Org.) Greedy Algorithm

(2, 8)  (5, 8)

(5, 7)

(1, 5)

(4, 3)  (8, 3)

(6, 1)
(Org.) Greedy Algorithm
(Org.) Greedy Algorithm

(2, 8) — (5, 8)
(5, 7)
(1, 5)
(4, 3) — (8, 3)
(6, 1)
Introduction
Algorithms Review
Greedy Algorithm (Org. and Imp.)
Apx. Greedy Algorithm (Ordered) \( \Theta \)-Graph Algorithm (Sink and Skip-list spanner)
Sink Spanner
WSPD-based Algorithm

Theoretical bounds
Applications
Designing approximation algorithms with spanners
Metric space searching
Protein Visualization

Research Topics
(Org.) Greedy Algorithm

[Diagram showing a geometric spanner network with vertices labeled (1, 5), (2, 8), (4, 3), (5, 7), (5, 8), (6, 1), (8, 3).]
(Org.) Greedy Algorithm

**Org. Greedy**

**Input:** $V$ and $t > 1$

**Output:** $t$-spanner $G(V, E)$

Sort pairs of points by non-decreasing order of distance;

$E := \emptyset$;

$G := (V, E)$;

for each pair $(u, v)$ of points (in sorted order) do

    if $\text{ShortestPath}(G, u, v) > t \cdot |uv|$ then

        Add $(u, v)$ to $E$;

    end

end

return $G(V, E)$;

**Time Complexity:** $O(n^3 \log n)$.

**Storage Complexity:** $O(n^2)$. 
(Org.) Greedy Algorithm

**Org. Greedy**

**Input:** \( V \) and \( t > 1 \)

**Output:** \( t \)-spanner \( G(V, E) \)

Sort pairs of points by non-decreasing order of distance;

\( E := \emptyset \);

\( G := (V, E) \);

for each pair \((u, v)\) of points (in sorted order) do

    if \( \text{SHORTESTPATH}(G, u, v) > t \cdot |uv| \) then
        Add \((u, v)\) to \( E \);
    end

end

return \( G(V, E) \);

**Time Complexity:** \( \mathcal{O}(n^3 \log n) \).

**Storage Complexity:** \( \mathcal{O}(n^2) \).
Imp. Greedy Algorithm

**Org. Greedy**

**Input:** $V$ and $t > 1$

**Output:** $t$-spanner $G(V, E)$

Sort pairs of points by non-decreasing order of distance;

$$E := \emptyset; \ G := (V, E);$$

for each pair $(u, v)$ of points (in sorted order) do

```
if \text{SHORTESTPATH}(G, u, v) > t \cdot |uv| \text{ then}
   Add (u, v) to $E$;
```

end

end

return $G(V, E)$;

Number of shortest path queries: $\Theta(n^2)$. 
Imp. Greedy Algorithm

**Org. Greedy**

**Input:** $V$ and $t > 1$

**Output:** $t$-spanner $G(V, E)$

Sort pairs of points by non-decreasing order of distance;

$E := \emptyset; \ G := (V, E)$;

for each pair $(u,v)$ of points (in sorted order) do
  if $\text{SHORTESTPATH}(G, u, v) > t \cdot |uv|$ then
    Add $(u,v)$ to $E$;
  end
end

return $G(V, E)$;

Number of shortest path queries: $\Theta(n^2)$.

**Observations:**

- We only want to know if there is a $t$-path between $u$ and $v$.
- The graph is only updated $O(n)$ times.
Imp. Greedy Algorithm

**Imp. GREEDY**

**Input:** \( V \) and \( t > 1 \)

**Output:** \( t \)-spanner \( G(V, E) \)

1. For each pair \( (u, v) \in V^2 \) do
   - Set \( \text{Weight}(u, v) := \infty \);
   - Sort pairs of points by non-decreasing order of distance;
   - \( E := \emptyset; \ G := (V, E); \)

2. For each pair \( (u, v) \) of points (in sorted order) do
   - If \( \text{Weight}(u, v) \leq t \cdot |uv| \) then
     - Skip \( (u, v) \);
   - Else
     - Compute single source shortest path with source \( u \);
     - For each \( w \) do update \( \text{Weight}(u, w) \) and \( \text{Weight}(w, u) \);
     - If \( \text{Weight}(u, v) \leq t \cdot |uv| \) then Skip \( (u, v) \);
     - Else Add \( (u, v) \) to \( E \);

3. Return \( G(V, E) \);
Imp. Greedy Algorithm

Conjecture:

The running time of IMP. GREEDY is $O(n^2 \log n)$.

Bose, Carmi, Farshi, Maheshvari and Smid (2008)

- The conjecture is wrong!
- They presented an algorithm which computes the greedy spanner in $O(n^2 \log n)$ time (even for points from some metric spaces).
Apx. Greedy Algorithm

- **Point set** → **$t$-spanner Algorithm**
  - **$t$-spanner** → **Constant degree**

Theoretical bounds

Applications
- Designing approximation algorithms with spanners
- Metric space searching
- Protein Visualization

Research Topics
Apx. Greedy Algorithm

Point set $t$ $t$-spanner Algorithm $t$-spanner $t$-spanner $t'$ t-spanner Algorithm Pruning Algorithm $(t \cdot t')$-spanner

Introduction

Algorithms Review
Greedy Algorithm (Org. and Imp.)
Apx. Greedy Algorithm (Ordered) $\Theta$-Graph Algorithm (Sink and Skip-list spanner)
Sink Spanner
WSPD-based Algorithm

Theoretical bounds

Applications
Designing approximation algorithms with spanners
Metric space searching
Protein Visualization

Research Topics
Apx. Greedy Algorithm

- Point set
  - $t$-spanner Algorithm
    - Constant degree
    - $t$-spanner
    - $O(n)$ edges
  - Sink Spanner
- Approximate Pruning Algorithm
  - $(t \cdot t')$-spanner
  - $t'$-spanner
  - $t$-spanner

Course Outline
- Textbook
- Introduction
- Algorithms Review
  - Greedy Algorithm (Orig. and Imp.)
  - Apx. Greedy Algorithm (Ordered) $\Theta$-Graph Algorithm (Sink and Skip-list spanner)
  - Sink Spanner
  - WSPD-based Algorithm
- Theoretical bounds
- Applications
  - Designing approximation algorithms with spanners
  - Metric space searching
  - Protein Visualization
- Research Topics
Apx. Greedy Algorithm

Point set → $t$-spanner Algorithm → Approximate Pruning Algorithm

$t$-spanner → $t$-spanner

$t$-spanner → (t · t')-spanner

Constant degree

$O(n)$ edges

Sink Spanner

Time Complexity: $O(n \log^2 n)$

Storage Complexity: $O(n)$. 
\( \Theta \)-Graph Algorithm

\[ t = 3, \Theta = \pi/6 \]
Course Outline

Textbook

Introduction

Algorithms Review
- Greedy Algorithm (Org. and Imp.)
- Apx. Greedy Algorithm (Ordered) \( \Theta \)-Graph Algorithm (Sink and Skip-list spanner)
- Sink Spanner
- WSPD-based Algorithm

Theoretical bounds

Applications
- Designing approximation algorithms with spanners
- Metric space searching
- Protein Visualization

Research Topics
$\Theta$-Graph Algorithm

$t = 3, \Theta = \frac{\pi}{6}$
Course Outline
Textbook
Introduction
Algorithms Review
Greedy Algorithm (Org. and Imp.)
Apx. Greedy Algorithm (Ordered) \( \Theta \)-Graph Algorithm (Sink and Skip-list spanner)
Sink Spanner
WSPD-based Algorithm
Theoretical bounds
Applications
Designing approximation algorithms with spanners
Metric space searching
Protein Visualization
Research Topics

\( \Theta \)-Graph Algorithm
\( t = 3, \Theta = \pi/6 \)
Geometric Spanner Networks

Course Outline

Textbook

Introduction

Algorithms Review

Greedy Algorithm (Org. and Imp.)

Apx. Greedy Algorithm (Ordered) \( \Theta \)-Graph Algorithm (Sink and Skip-list spanner)

Sink Spanner

WSPD-based Algorithm

Theoretical bounds

Applications

Designing approximation algorithms with spanners

Metric space searching

Protein Visualization

Research Topics

\( \Theta \)-Graph Algorithm

\( t = 3, \Theta = \pi / 6 \)
θ-Graph Algorithm

\( t = 3, \theta = \pi/6 \)
Course Outline

Textbook

Introduction

Algorithms Review
- Greedy Algorithm (Org. and Imp.)
- Apx. Greedy Algorithm (Ordered) \( \Theta \)-Graph Algorithm (Sink and Skip-list spanner)
- Sink Spanner
- WSPD-based Algorithm

Theoretical bounds

Applications
- Designing approximation algorithms with spanners
- Metric space searching
- Protein Visualization

Research Topics

\( \Theta \)-Graph Algorithm

\( t = 3, \Theta = \pi/6 \)
Geometric Spanner Networks

Course Outline

Textbook

Introduction

Algorithms Review

Greedy Algorithm (Org. and Imp.)

Apx. Greedy Algorithm (Ordered) \( \Theta \)-Graph Algorithm (Sink and Skip-list spanner)

Sink Spanner

WSPD-based Algorithm

Theoretical bounds

Applications

Designing approximation algorithms with spanners

Metric space searching

Protein Visualization

Research Topics

\( \Theta \)-Graph Algorithm

\( t = 3, \Theta = \pi/6 \)
$\Theta$-Graph Algorithm

$t = 3, \Theta = \pi/6$
$\Theta$-Graph Algorithm

\[ t = 3, \Theta = \pi/6 \]
Θ-Graph Algorithm

**Input:** V and t > 1

**Output:** t-spanner G(V, E)

Set \( k := \) the smallest integer such that \( t = \frac{1}{\cos \theta - \sin \theta} \) for \( \theta = \frac{2\pi}{k} \);

\( E := \emptyset \);

for each point \( u \in V \) do

\( C_1, \ldots, C_k := \) non-overlapping cones with angle \( \theta \) and with apex at \( u \);

for each cone \( C_i \) do

Connect \( u \) to the closest point in \( C_i \);

end

end

return \( G(V, E) \);

Time Complexity: \( O(n \log n) \).

Storage Complexity: \( O(n) \).
Θ-Graph Algorithm

**Input:** \( V \) and \( t > 1 \)
**Output:** \( t \)-spanner \( G(V, E) \)

Set \( k := \) the smallest integer such that \( t = \frac{1}{\cos \theta - \sin \theta} \) for \( \theta = \frac{2\pi}{k} \);
\( E := \emptyset \);

for each point \( u \in V \) do

\( C_1, \ldots, C_k := \) non-overlapping cones with angle \( \theta \) and with apex at \( u \);

for each cone \( C_i \) do

Connect \( u \) to the closest point in \( C_i \);

end

end

return \( G(V, E) \);

**Time Complexity:** \( O(n \log n) \).
**Storage Complexity:** \( O(n) \).
Variants of $\Theta$-Graph Algorithm

**Ordered $\Theta$-Graph—$O(\log n)$ maximum degree**

Same as the $\Theta$-graph algorithm, except we add points one by one in a special order.

**Random Ordered $\Theta$-Graph—$O(\log n)$ spanner diameter**

We add points one by one in a random order.

**Sink Spanner—bounded degree**

Decrease the degree of nodes by replacing some edges by paths within other nodes.

**Skip-List Spanner—$O(\log n)$ spanner diameter**

Decrease the diameter of $\Theta$-graph by adding some extra edges.
Geometric Spanner Networks

Course Outline

Textbook

Introduction

Algorithms Review

Greedy Algorithm (Org. and Imp.)

Apx. Greedy Algorithm (Ordered) Θ-Graph Algorithm (Sink and Skip-list spanner)

Sink Spanner

WSPD-based Algorithm

Theoretical bounds

Applications

Designing approximation algorithms with spanners

Metric space searching

Protein Visualization

Research Topics

Variants of Θ-Graph Algorithm

Ordered Θ-Graph—$O(\log n)$ maximum degree

Same as the Θ-graph algorithm, except we add points one by one in a special order.

Random Ordered Θ-Graph—$O(\log n)$ spanner diameter

We add points one by one in a random order.

Sink Spanner—bounded degree

Decrease the degree of nodes by replacing some edges by paths within other nodes.

Skip-List Spanner—$O(\log n)$ spanner diameter

Decrease the diameter of Θ-graph by adding some extra edges.
Variants of Θ-Graph Algorithm

**Ordered Θ-Graph** – $O(\log n)$ maximum degree

Same as the Θ-graph algorithm, except we add points one by one in a special order.

**Random Ordered Θ-Graph** – $O(\log n)$ spanner diameter

We add points one by one in a random order.

**Sink Spanner** – bounded degree

Decrease the degree of nodes by replacing some edges by paths within other nodes.

**Skip-List Spanner** – $O(\log n)$ spanner diameter

Decrease the diameter of Θ-graph by adding some extra edges.
Variants of $\Theta$-Graph Algorithm

**Ordered $\Theta$-Graph** – $O(\log n)$ maximum degree

Same as the $\Theta$-graph algorithm, except we add points one by one in a special order.

**Random Ordered $\Theta$-Graph** – $O(\log n)$ spanner diameter

We add points one by one in a random order.

**Sink Spanner** – bounded degree

Decrease the degree of nodes by replacing some edges by paths within other nodes.

**Skip-List Spanner** – $O(\log n)$ spanner diameter

Decrease the diameter of $\Theta$-graph by adding some extra edges.
Sink Spanner
A variant of $\Theta$-graph with bounded degree

Input: $V$ and $t > 1$
Output: $t$-spanner $G(V, E)$

Construct a directed $\sqrt{t}$-spanner $\vec{G}$ with bounded out-degree;
for each point $q \in V$ do
  | Replace the “star” pointing to $q$ by a $\sqrt{t}$-$q$-sink spanner
end
return $G(V, E)$;

Time Complexity: $\mathcal{O}(n \log n)$  Storage Complexity: $\mathcal{O}(n)$. 
Sink Spanner
A variant of $\Theta$-graph with bounded degree

**Input:** $V$ and $t > 1$

**Output:** $t$-spanner $G(V, E)$

Construct a directed $\sqrt{t}$-spanner $\vec{G}$ with bounded out-degree;

for each point $q \in V$ do
   Replace the “star” pointing to $q$ by a $\sqrt{t}$-$q$-sink spanner
end

return $G(V, E)$;

**Time Complexity:** $\mathcal{O}(n \log n)$  
**Storage Complexity:** $\mathcal{O}(n)$. 
Skip-List Spanner
A variant of $\Theta$-graph with $O(\log n)$ spanner diameter

**Input:** $V$ and $t > 1$

**Output:** $t$-spanner $G(V, E)$

Set $V_0 := V$; $i := 1$; 
while $V_{i-1} \neq \emptyset$ do 
  $V_i$ contains each points of $V_{i-1}$ with probability $1/2$; 
end 
for each $i$ do 
  Construct a $t$-spanner $G_i(V_i, E_i)$ using the $\Theta$-graph algorithm; 
end 
$E = \bigcup_i E_i$; 
return $G(V, E)$;

Time Complexity: $O(n \log n)$  Storage Complexity: $O(n)$.
Skip-List Spanner
A variant of Θ-graph with $O(\log n)$ spanner diameter

**Input:** $V$ and $t > 1$

**Output:** $t$-spanner $G(V, E)$

Set $V_0 := V$; $i := 1$

while $V_{i-1} \neq \emptyset$ do

  $V_i$ contains each points of $V_{i-1}$ with probability $1/2$;

end

for each $i$ do

  Construct a $t$-spanner $G_i(V_i, E_i)$ using the Θ-graph algorithm;

end

$E = \cup_i E_i$

return $G(V, E)$;

Time Complexity: $O(n \log n)$

Storage Complexity: $O(n)$.
Well Separated Pair: 

\[ A, B \subset \mathbb{R}^d \text{ are } s\text{-well separated } (s > 0), \text{ if } \exists \text{ disjoint balls, } D_A \text{ and } D_B \text{ such that} \]

- 

-
Well Separated Pair Decomposition (WSPD)

**Well Separated Pair:**

\[ A, B \subset \mathbb{R}^d \text{ are } s\text{-well separated } (s > 0), \text{ if } \exists \text{ disjoint balls, } D_A \text{ and } D_B \text{ such that} \]

- \( A \)
- \( B \)

\[ A \cap B = \emptyset \]
Well Separated Pair Decomposition (WSPD)

**Well Separated Pair:**

\[ A, B \subset \mathbb{R}^d \text{ are } s\text{-well separated } (s > 0), \text{ if } \exists \text{ disjoint balls, } D_A \text{ and } D_B \text{ such that} \]

- \( A \subseteq D_A \text{ and } B \subseteq D_B. \)
Well Separated Pair Decomposition (WSPD)

Well Separated Pair:

\( A, B \subseteq \mathbb{R}^d \) are \( s \)-well separated \((s > 0)\), if \( \exists \) disjoint balls, \( D_A \) and \( D_B \) such that

- \( A \subseteq D_A \) and \( B \subseteq D_B \).
- \( d(D_A, D_B) \geq s \times \max(\text{radius}(D_A), \text{radius}(D_B)) \).

\[ r_A \]  \[ r_B \]  \[ A \]  \[ B \]  \[ D_A \]  \[ D_B \]  \[ \geq s \times \max(r_A, r_B) \]
Well Separated Pair Decomposition (WSPD)

Well Separated Pair Decomposition:

Let $V \subset \mathbb{R}^d$ and $s > 0$. A WSPD for $V$ with respect to $s$ is a set $\{(A_i, B_i)\}_{i=1}^m$ of pairs of non-empty subsets of $V$ such that

- $\forall i$, $A_i$ and $B_i$ are $s$-well separated,
- $\forall p, q \in V$, there is exactly one index $i$ s.t.
  - $p \in A_i$ and $q \in B_i$ or
  - $q \in A_i$ and $p \in B_i$.

$m$ : Size of WSPD.

Callahan & Kosaraju (1995)

For each set of $n$ points, we can construct a WSPD of size $\mathcal{O}(s^d \cdot n)$ in $\mathcal{O}(n \log n)$ time using $\mathcal{O}(s^d \cdot n)$ space.
Well Separated Pair Decomposition (WSPD)

Well Separated Pair Decomposition:

Let $V \subset \mathbb{R}^d$ and $s > 0$. A WSPD for $V$ with respect to $s$ is a set $\{(A_i, B_i)\}_{i=1}^m$ of pairs of non-empty subsets of $V$ such that

- $\forall i$, $A_i$ and $B_i$ are $s$-well separated,
- $\forall p, q \in V$, there is exactly one index $i$ s. t. $p \in A_i$ and $q \in B_i$ or $q \in A_i$ and $p \in B_i$.

$m$ : Size of WSPD.

Callahan & Kosaraju (1995)

For each set of $n$ points, we can construct a WSPD of size $O(s^d \cdot n)$ in $O(n \log n)$ time using $O(s^d \cdot n)$ space.
Well Separated Pair Decomposition (WSPD)

Well Separated Pair Decomposition:

Let $V \subset \mathbb{R}^d$ and $s > 0$. A WSPD for $V$ with respect to $s$ is a set $\{(A_i, B_i)\}_{i=1}^m$ of pairs of non-empty subsets of $V$ such that

- $\forall i$, $A_i$ and $B_i$ are $s$-well separated,
- $\forall p, q \in V$, there is exactly one index $i$ s. t. $p \in A_i$ and $q \in B_i$ or $q \in A_i$ and $p \in B_i$.

$m : \text{Size of WSPD.}$

Callahan & Kosaraju (1995)

For each set of $n$ points, we can construct a WSPD of size $\mathcal{O}(s^d \cdot n)$ in $\mathcal{O}(n \log n)$ time using $\mathcal{O}(s^d \cdot n)$ space.
Well Separated Pair Decomposition (WSPD)

Well Separated Pair Decomposition:

Let \( V \subset \mathbb{R}^d \) and \( s > 0 \). A WSPD for \( V \) with respect to \( s \) is a set \( \{(A_i, B_i)\}_{i=1}^{m} \) of pairs of non-empty subsets of \( V \) such that

- \( \forall i, A_i \text{ and } B_i \text{ are } s\text{-well separated}, \)
- \( \forall p, q \in V, \text{ there is exactly one index } i \text{ s. t.} \)
  - \( p \in A_i \text{ and } q \in B_i \) or
  - \( q \in A_i \text{ and } p \in B_i \).

\( m \): Size of WSPD.

Callahan & Kosaraju (1995)

For each set of \( n \) points, we can construct a WSPD of size \( \mathcal{O}(s^d \cdot n) \) in \( \mathcal{O}(n \log n) \) time using \( \mathcal{O}(s^d \cdot n) \) space.
Well Separated Pair Decomposition (WSPD)

Well Separated Pair Decomposition:

Let \( V \subseteq \mathbb{R}^d \) and \( s > 0 \). A WSPD for \( V \) with respect to \( s \) is a set \( \{(A_i, B_i)\}_{i=1}^{m} \) of pairs of non-empty subsets of \( V \) such that

- \( \forall i, A_i \) and \( B_i \) are \( s \)-well separated,
- \( \forall p, q \in V \), there is exactly one index \( i \) s. t.
  - \( p \in A_i \) and \( q \in B_i \) or
  - \( q \in A_i \) and \( p \in B_i \).

\( m \) : Size of WSPD.

Callahan & Kosaraju (1995)

For each set of \( n \) points, we can construct a WSPD of size \( \mathcal{O}(s^d \cdot n) \) in \( \mathcal{O}(n \log n) \) time using \( \mathcal{O}(s^d \cdot n) \) space.
WSPD-based Algorithm

**WSPD Algorithm**

**Input**: $V$ and $t > 1$

**Output**: $t$-spanner $G(V, E)$

Set $\mathcal{W} :=$ WSPD of $V$ w.r.t. $s := \frac{4(t+1)}{t-1}$;

Set $E = \emptyset$

**for each** $(A_i, B_i) \in \mathcal{W}$ **do**

- Select an arbitrary node $u \in A_i$ and an arbitrary node $v \in B_i$
- Add edge $(u, v)$ to $E$

**end**

**return** $G(V, E)$.

**Time Complexity**: $\mathcal{O}(n \log n)$.

**Storage Complexity**: $\mathcal{O}(n)$. 
**WSPD-based Algorithm**

**WSPD Algorithm**

- **Input:** $V$ and $t > 1$
- **Output:** $t$-spanner $G(V, E)$
- Set $\mathcal{W} :=$ WSPD of $V$ w.r.t. $s := \frac{4(t+1)}{t-1}$;
- Set $E = \emptyset$;
- **for each** $(A_i, B_i) \in \mathcal{W}$ **do**
  - Select an arbitrary node $u \in A_i$ and an arbitrary node $v \in B_i$;
  - Add edge $(u, v)$ to $E$.
- **end**
- **return** $G(V, E)$.

**Time Complexity:** $\mathcal{O}(n \log n)$.
**Storage Complexity:** $\mathcal{O}(n)$. 
### Theoretical bounds

<table>
<thead>
<tr>
<th></th>
<th>Size</th>
<th>Weight</th>
<th>Degree</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greedy spanner</td>
<td>$O(n)$</td>
<td>$O(wt(MST))$</td>
<td>$O(1)$</td>
<td>$O(n^2 \log n)$</td>
</tr>
<tr>
<td>Apx. greedy spanner</td>
<td>$O(n)$</td>
<td>$O(wt(MST))$</td>
<td>$O(1)$</td>
<td>$O(n \log n)$</td>
</tr>
<tr>
<td>$\Theta$-graph</td>
<td>$O(n)$</td>
<td>$\Theta(n \cdot wt(MST))$</td>
<td>$\Theta(n)$</td>
<td>$O(n \log n)$</td>
</tr>
<tr>
<td>O. $\Theta$-graph</td>
<td>$O(n)$</td>
<td>$O(n \cdot wt(MST))$</td>
<td>$O(\log n)$</td>
<td>$O(n \log n)$</td>
</tr>
<tr>
<td>WSPD spanner</td>
<td>$O(n)$</td>
<td>$O(\log n \cdot wt(MST))$</td>
<td>$\Theta(n)$</td>
<td>$O(n \log n)$</td>
</tr>
<tr>
<td>Sink-spanner</td>
<td>$O(n)$</td>
<td>$O(n \cdot wt(MST))$</td>
<td>$O(1)$</td>
<td>$O(n \log n)$</td>
</tr>
<tr>
<td>Skip-list spanner</td>
<td>$O(n)^*$</td>
<td>$\Theta(n \cdot wt(MST))^*$</td>
<td>$\Theta(n)$</td>
<td>$O(n \log n)^*$</td>
</tr>
</tbody>
</table>

(*): Expected with high probability
Applications

Course Outline
Textbook
Introduction
Algorithms Review
Greedy Algorithm (Org. and Imp.)
Apx. Greedy Algorithm (Ordered) $\Theta$-Graph Algorithm (Sink and Skip-list spanner)
Sink Spanner
WSPD-based Algorithm
Theoretical bounds
Applications
Designing approximation algorithms with spanners
Metric space searching
Protein Visualization
Research Topics
Traveling Salesperson Problem (TSP)

Find the **shortest** tour that visits each point exactly once and return to the starting point.

Known results:

- The problem is NP-hard even in $\mathbb{R}^d$.
- A $1.5$-approximation algorithm by Christofides et al. (1976).
Applications
Designing approximation algorithms with spanners

Traveling Salesperson Problem (TSP)
Find the **shortest** tour that visits each point exactly once and return to the starting point.

Known results:

- The problem is NP-hard even in $\mathbb{R}^d$.
- A PTAS ($(1 + \varepsilon)$-approx. Alg.) for geometric case by Arora (1998) and Mitchell (1999).
Applications
Designing approximation algorithms with spanners

Definition:
If $G$ is a graph with vertex set $P$, then a tour of $P$ in $G$ is a (possibly non-simple) cycle in $G$ that visits each point of $P$ at least once.

Observation:
For any $t$-spanner $G$ for $P$, there is a tour of $P$ in $G$, whose weight is at most $t \cdot wt(TSP(P))$.

Theorem (Rao and Smith, 1998)
Given a $(1 + \varepsilon)$-spanner of a set of $n$ points with $O(n)$ size and $O(wt(MST))$ weight, we can compute a $(1 + \varepsilon)$-approximation of $TSP(P)$ in $O(n \log n)$ time.
Applications
Designing approximation algorithms with spanners

Definition:
If $G$ is a graph with vertex set $P$, then a tour of $P$ in $G$ is a (possibly non-simple) cycle in $G$ that visits each point of $P$ at least once.

Observation:
For any $t$-spanner $G$ for $P$, there is a tour of $P$ in $G$, whose weight is at most $t \cdot wt(TSP(P))$.

Theorem (Rao and Smith, 1998)
Given a $(1 + \varepsilon)$-spanner of a set of $n$ points with $O(n)$ size and $O(wt(MST))$ weight, we can compute a $(1 + \varepsilon)$-approximation of $TSP(P)$ in $O(n \log n)$ time.
Applications
Designing approximation algorithms with spanners

**Definition:**
If $G$ is a graph with vertex set $P$, then a tour of $P$ in $G$ is a (possibly non-simple) cycle in $G$ that visits each point of $P$ at least once.

**Observation:**
For any $t$-spanner $G$ for $P$, there is a tour of $P$ in $G$, whose weight is at most $t \cdot wt(TSP(P))$.

**Theorem (Rao and Smith, 1998)**
Given a $(1 + \varepsilon)$-spanner of a set of $n$ points with $O(n)$ size and $O(wt(MST))$ weight, we can compute a $(1 + \varepsilon)$-approximation of $TSP(P)$ in $O(n \log n)$ time.

Applications
Metric space searching

Approximate proximity searching:
- Multimedia information retrieval,
- Data mining,
- Pattern recognition,
- Machine learning,
- Computer vision and
- Biomedical databases.
Applications
Metric space searching

Approximate proximity searching:
- Multimedia information retrieval,
- Data mining,
- Pattern recognition,
- Machine learning,
- Computer vision and
- Biomedical databases.
What is the role of spanners?

- A meter show the similarity between any two objects.
- But evaluating the distances are expensive.
- One way to speedup is computing the distance between any two objects and save them, but it need $O(n^2)$ space (AESA).
- A $t$-spanner can be used as a sparse data structure to reduce the space.
What is the role of spanners?

- A meter show the similarity between any two objects.
- But evaluating the distances are expensive.
- One way to speedup is computing the distance between any two objects and save them, but it need $O(n^2)$ space (AESA).
- A $t$-spanner can be used as a sparse data structure to reduce the space.
What is the role of spanners?

- A meter shows the similarity between any two objects.
- But evaluating the distances are expensive.
- One way to speed up is computing the distance between any two objects and save them, but it needs $O(n^2)$ space (AESA).
- A $t$-spanner can be used as a sparse data structure to reduce the space.
Applications

Metric space searching

What is the role of spanners?

- A meter show the similarity between any two objects.
- But evaluating the distances are expensive.
- One way to speedup is computing the distance between any two objects and save them, but it need $\mathcal{O}(n^2)$ space (AESA).
- A $t$-spanner can be used as a sparse data structure to reduce the space.

Applications
Protein Visualization

Current and Future Works:

- Dynamic spanners (insert and remove nodes).
- Kinetic spanners (when points move and we want to maintain an spanner all the time).
- Fault-tolerant spanners (vertex/edge fault tolerant or region fault tolerant).
- Spanners among obstacles.
- Optimization problems.
- External memory (I/O efficient) algorithms for generating spanners.
- Experimental works on spanner algorithms.
Research Topics

Current and Future Works:

- Dynamic spanners (insert and remove nodes).
- Kinetic spanners (when points move and we want to maintain an spanner all the time).
- Fault-tolerant spanners (vertex/edge fault tolerant or region fault tolerant).
- Spanners among obstacles.
- Optimization problems.
- External memory (I/O efficient) algorithms for generating spanners.
- Experimental works on spanner algorithms.
Research Topics

Current and Future Works:

- Dynamic spanners (insert and remove nodes).
- Kinetic spanners (when points move and we want to maintain an spanner all the time).
- Fault-tolerant spanners (vertex/edge fault tolerant or region fault tolerant).
- Spanners among obstacles.
- Optimization problems.
- External memory (I/O efficient) algorithms for generating spanners.
- Experimental works on spanner algorithms.
Research Topics

Current and Future Works:

- Dynamic spanners (insert and remove nodes).
- Kinetic spanners (when points move and we want to maintain an spanner all the time).
- Fault-tolerant spanners (vertex/edge fault tolerant or region fault tolerant).
- Spanners among obstacles.
- Optimization problems.
- External memory (I/O efficient) algorithms for generating spanners.
- Experimental works on spanner algorithms.
Research Topics

Current and Future Works:

- Dynamic spanners (insert and remove nodes).
- Kinetic spanners (when points move and we want to maintain a spanner all the time).
- Fault-tolerant spanners (vertex/edge fault tolerant or region fault tolerant).
- Spanners among obstacles.
- Optimization problems.
  - External memory (I/O efficient) algorithms for generating spanners.
  - Experimental works on spanner algorithms.
Research Topics

Current and Future Works:

- Dynamic spanners (insert and remove nodes).
- Kinetic spanners (when points move and we want to maintain an spanner all the time).
- Fault-tolerant spanners (vertex/edge fault tolerant or region fault tolerant).
- Spanners among obstacles.
- Optimization problems.
- External memory (I/O efficient) algorithms for generating spanners.
- Experimental works on spanner algorithms.
Research Topics

Current and Future Works:
- Dynamic spanners (insert and remove nodes).
- Kinetic spanners (when points move and we want to maintain an spanner all the time).
- Fault-tolerant spanners (vertex/edge fault tolerant or region fault tolerant).
- Spanners among obstacles.
- Optimization problems.
- External memory (I/O efficient) algorithms for generating spanners.
- Experimental works on spanner algorithms.
Geometric Spanner Networks

Course Outline

Textbook

Introduction

Algorithms Review

Greedy Algorithm (Org. and Imp.)
Apx. Greedy Algorithm (Ordered) $\Theta$-Graph Algorithm (Sink and Skip-list spanner)
Sink Spanner
WSPD-based Algorithm

Theoretical bounds

Applications

Designing approximation algorithms with spanners
Metric space searching
Protein Visualization

Research Topics