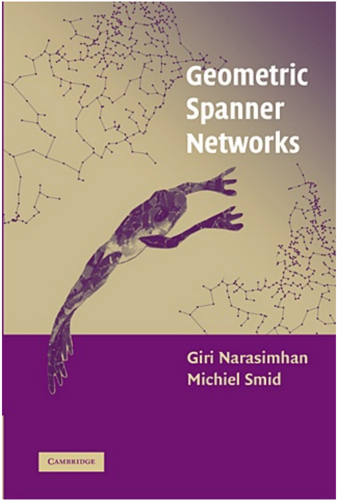
Geometric Spanner Networks

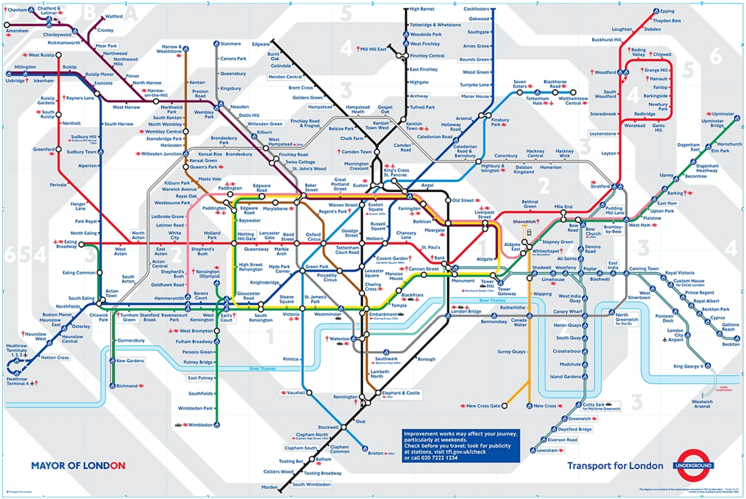
Spring 2014

Textbook:

Giri Narasimhan, Michiel Smid, **Geometric Spanner Networks**, CAMBRIDGE UNIVERSITY PRESS, 2007.

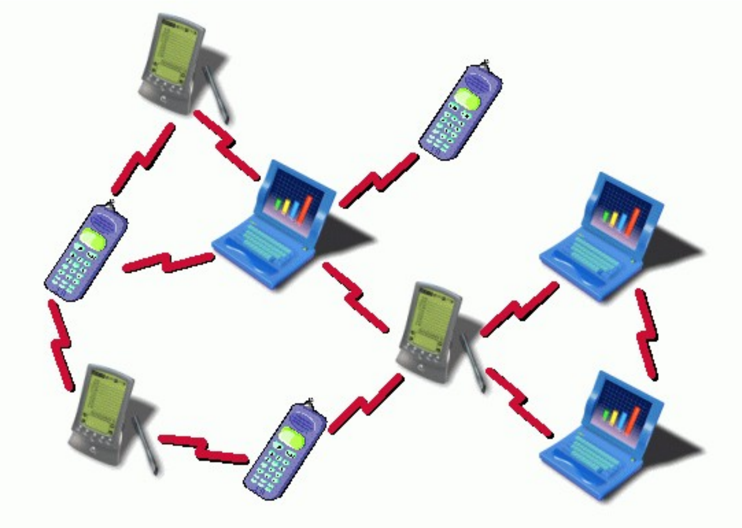


Example of networks



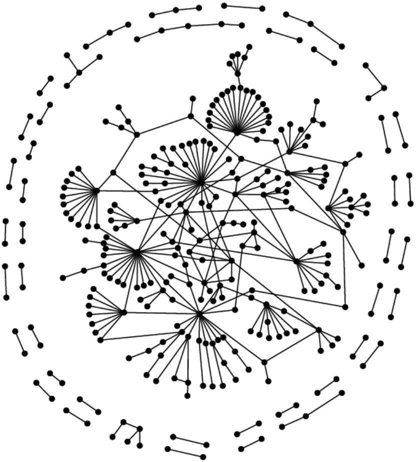
London Underground Network

Example of networks



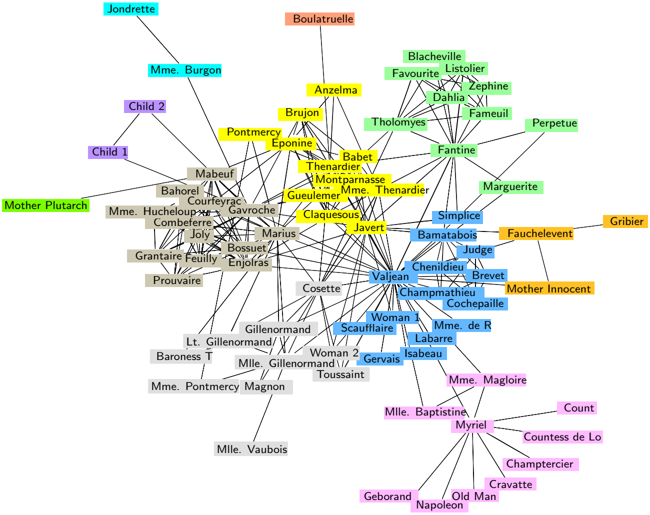
Ad hoc Network

Example of networks



Yeast Protein Interaction Network

Example of networks



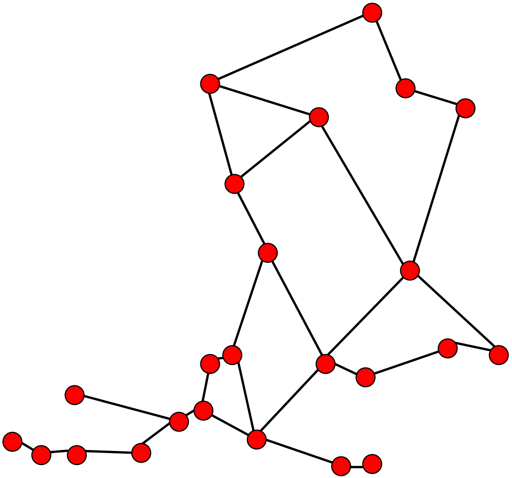
A Social Network (Les Miserables characters)

Geometric Network

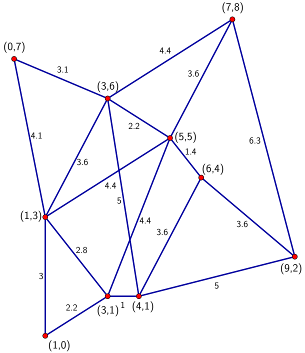
Geometric Network

Geometric Network

Geometric Network



Geometric Network



Geometric Network

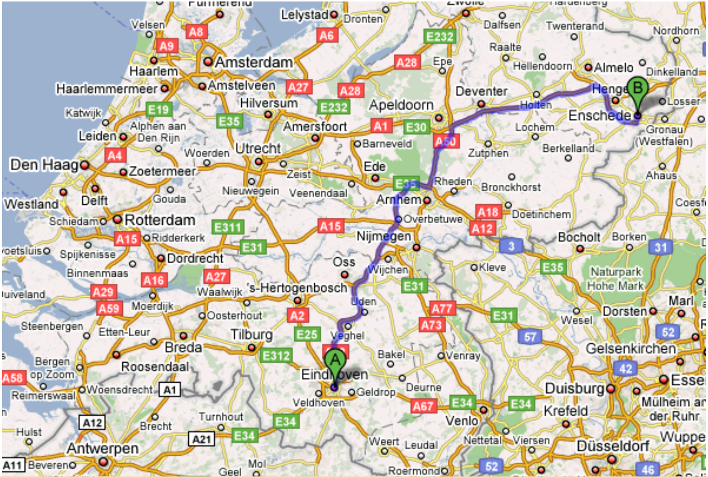
Weighted undirected graph $G(V, E)$ s.t.

- $V \subset \mathbb{R}^d$.
- $\forall e = (u, v) \in E, wt(e) = |uv|$.

Network Quality

- Driving distance: 256 km. Actual distance: 198 km.
- $\dfrac{\text{Driving distance}}{\text{Actual distance}} = 1.27.$

Network Quality

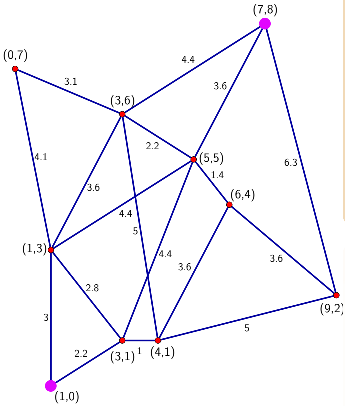


- Driving distance: 180 km.   Actual distance: 136 km.
- $\dfrac{\text{Driving distance}}{\text{Actual distance}} = 1.32.$

Network Quality

- Driving distance: 143 km. Actual distance: 100 km.
- $\dfrac{\text{Driving distance}}{\text{Actual distance}} = 1.43.$

Network Quality



Dilation (stretch factor)

- between a pair of vertices=

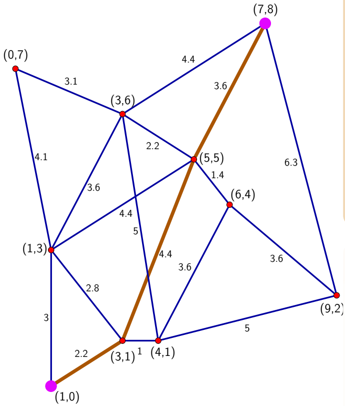
$$\frac{\text{Distance in the graph}}{\text{Euclidean distance}}$$

- of a network= maximum dilation between all pairs.

t-spanner

A network with dilation at most t, or $\forall u, v \in V$, 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=

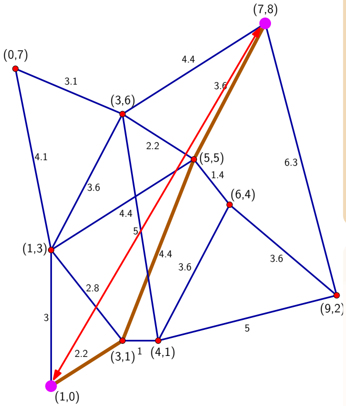
$$\frac{\text{Distance in the graph}}{\text{Euclidean distance}}$$

- of a network= maximum dilation between all pairs.

t-spanner

A network with dilation at most t, or $\forall u, v \in V$, 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=

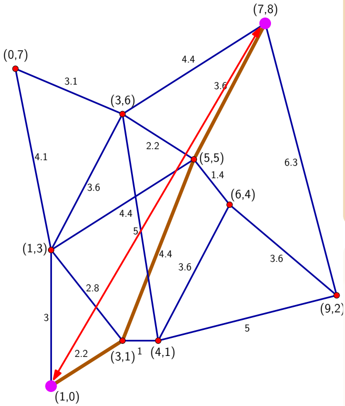
$$\frac{\text{Distance in the graph}}{\text{Euclidean distance}}$$

- of a network= maximum dilation between all pairs.

t-spanner

A network with dilation at most t, or $\forall u, v \in V$, 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=

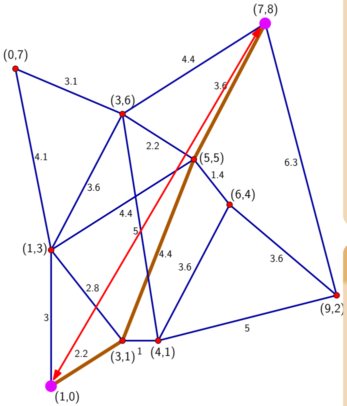
$$\frac{\text{Distance in the graph}}{\text{Euclidean distance}}$$

- of a network= maximum dilation between all pairs.

t-spanner

A network with dilation at most t, or $\forall u, v \in V$, 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=

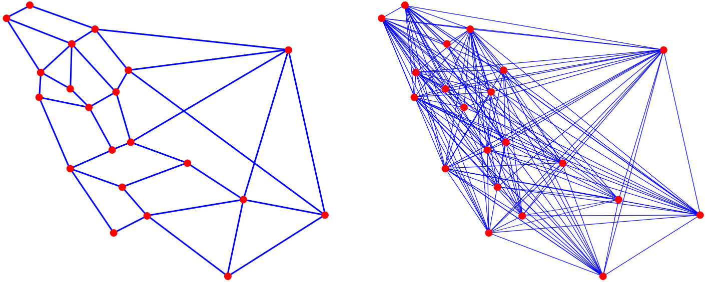
$$\frac{\text{Distance in the graph}}{\text{Euclidean distance}}$$

- of a network= maximum dilation between all pairs.

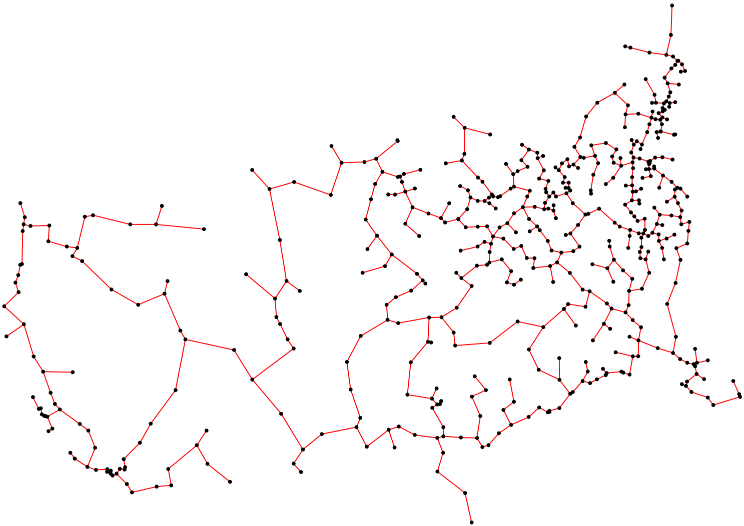
t-spanner

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

$(1+\varepsilon)$-Spanners approximate the complete graphs with error ε.

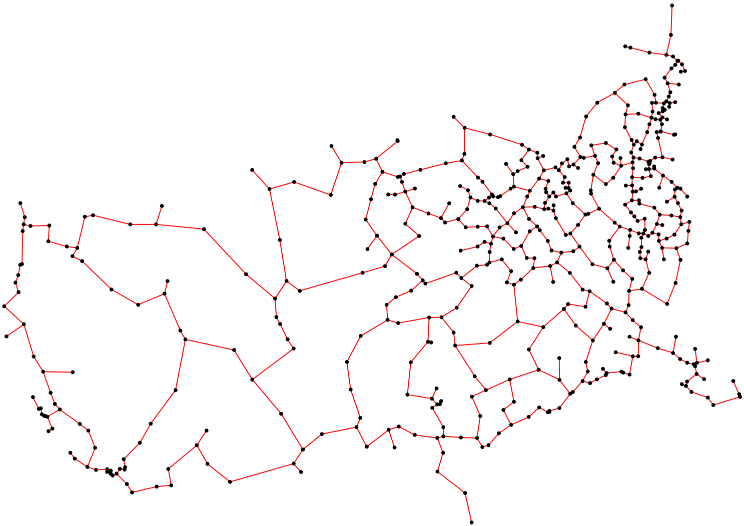


Example



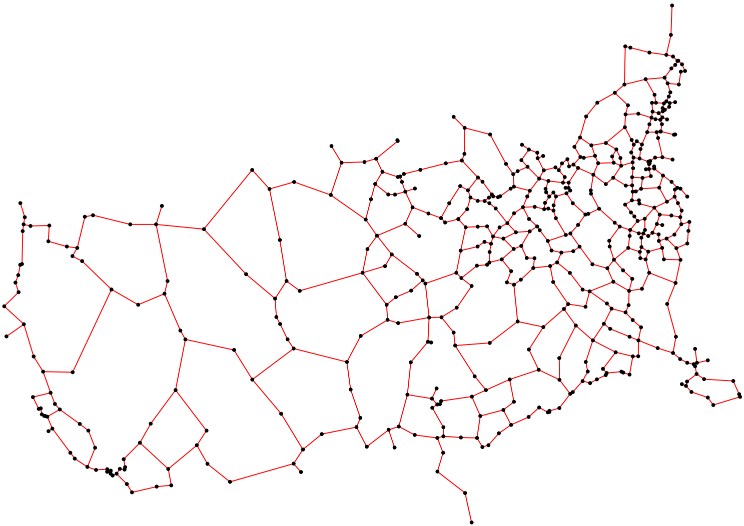
10-spanner for 532 US-cities

Example



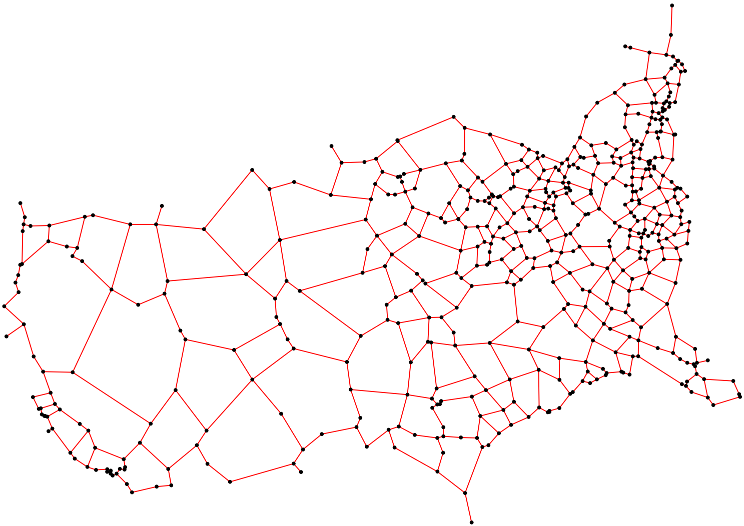
5-spanner for 532 US-cities

Example



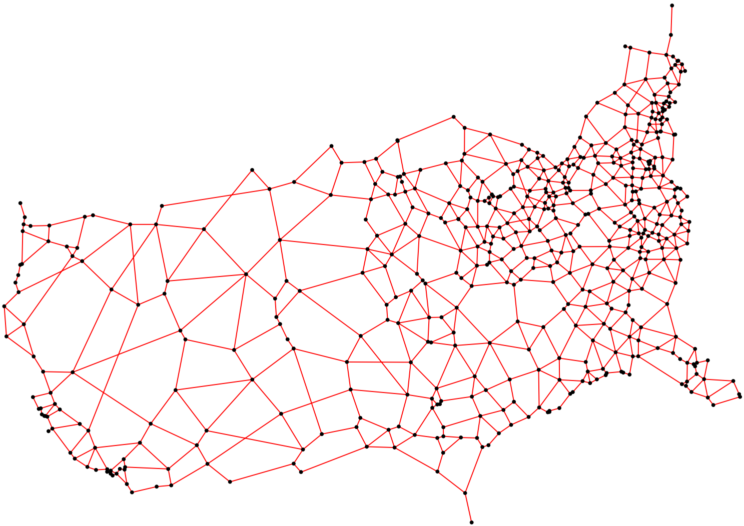
3-spanner for 532 US-cities

Example



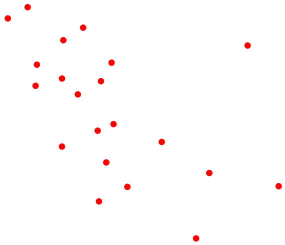
2-spanner for 532 US-cities

Example

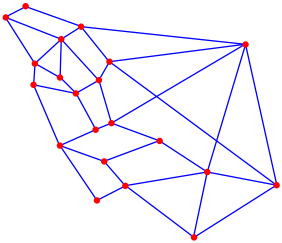


1.5-spanner for 532 US-cities

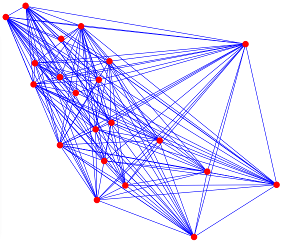
How to compute a good spanner?



Given a set V and $t > 1$



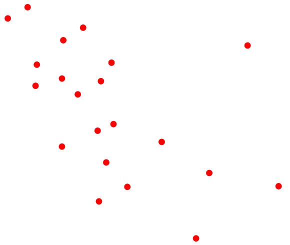
Sparse t-Spanner



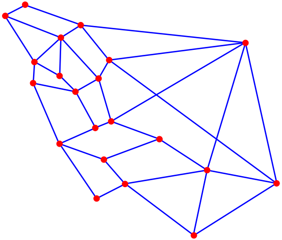
Quality measurement:

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

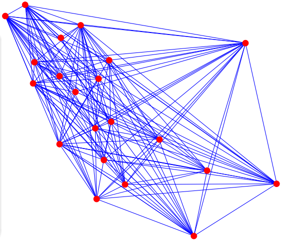
How to compute a good spanner?



Given a set V and $t > 1$



Sparse t-Spanner



Quality measurement:

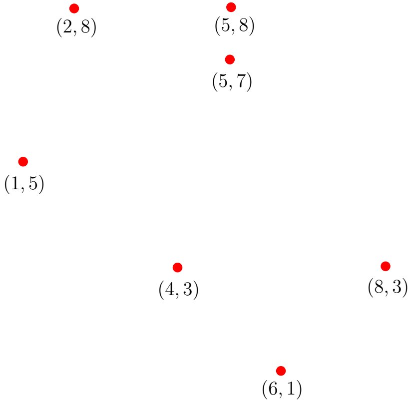
- Number of edges (size)
- Weight (compared with MST)
- Maximum degree
- Diameter

How to compute a good spanner?

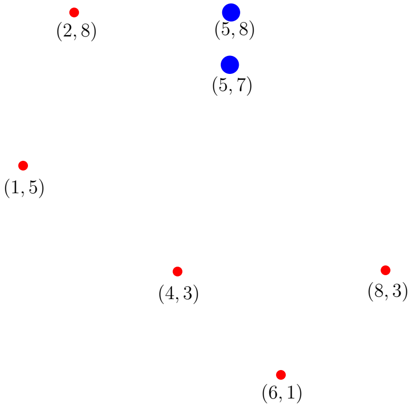
Constructing sparse t-spanners:

- 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)).

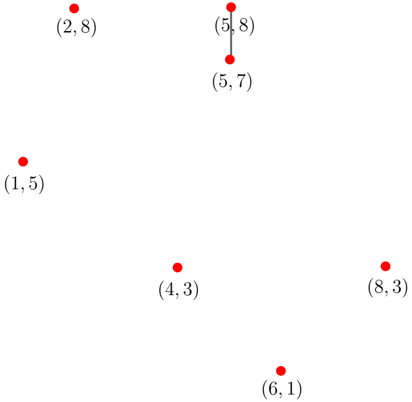
(Org.) Greedy Algorithm



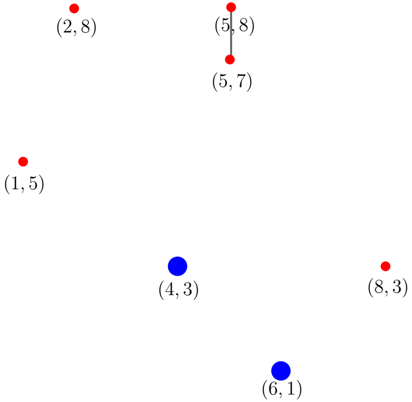
(Org.) Greedy Algorithm



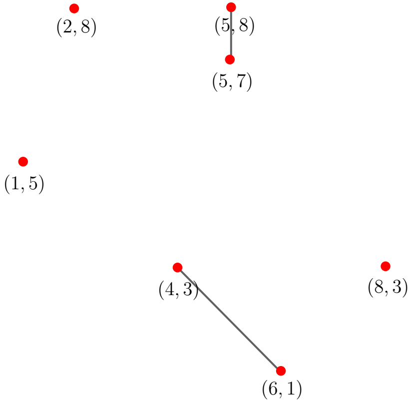
(Org.) Greedy Algorithm



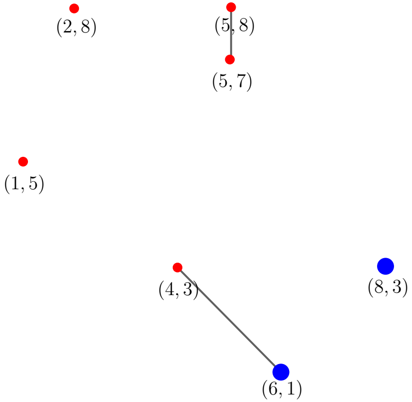
(Org.) Greedy Algorithm



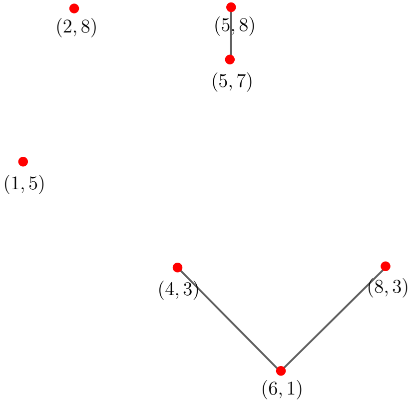
(Org.) Greedy Algorithm



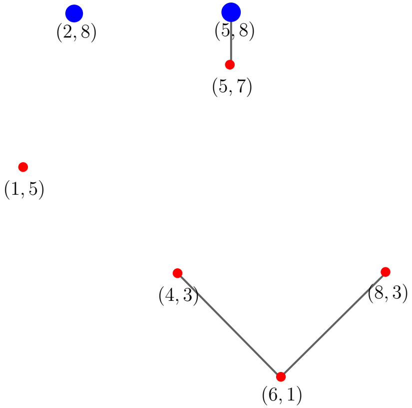
(Org.) Greedy Algorithm



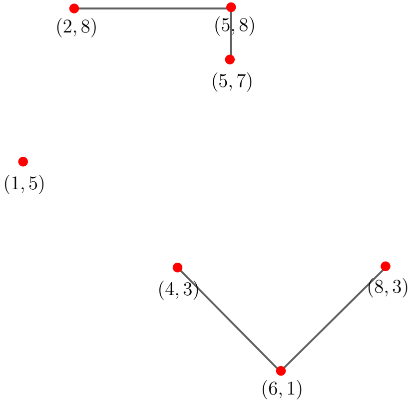
(Org.) Greedy Algorithm



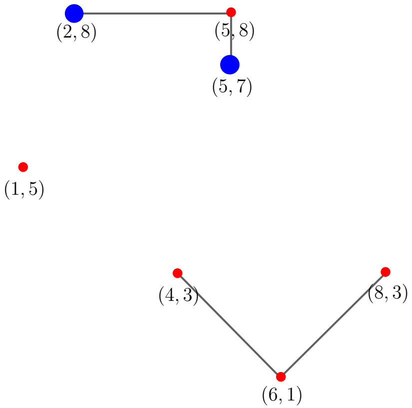
(Org.) Greedy Algorithm



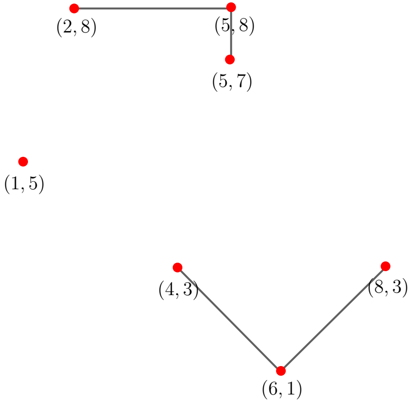
(Org.) Greedy Algorithm



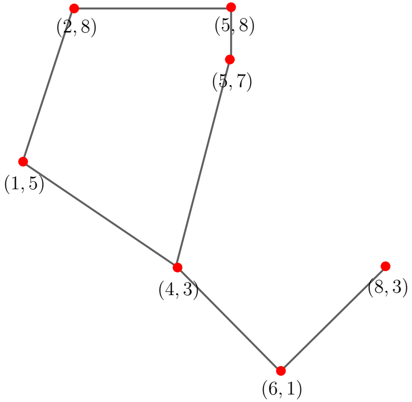
(Org.) Greedy Algorithm



(Org.) Greedy Algorithm



(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 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: $\mathcal{O}(n^2)$.

(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 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 SHORTESTPATH$(G, u, v) > t \cdot |uv|$ **then**
        Add (u, v) to E;
    end
end
return $G(V, E)$;

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

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 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 $\mathcal{O}(n)$ times.

Imp. Greedy Algorithm

Imp. Greedy

Input: V and $t > 1$
Output: t-spanner $G(V, E)$
for *each pair* $(u, v) \in V^2$ **do** Set *Weight*$(u, v) := \infty$;
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 *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 *Weight*(u, w) and *Weight*(w, u);
        if *Weight*$(u, v) \leq t \cdot |uv|$ **then** Skip (u, v);
        else Add (u, v) to E;
    end
end
return $G(V, E)$;

Imp. Greedy Algorithm

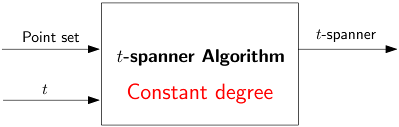
Conjecture:

The running time of Imp. Greedy is $\mathcal{O}(n^2 \log n)$.

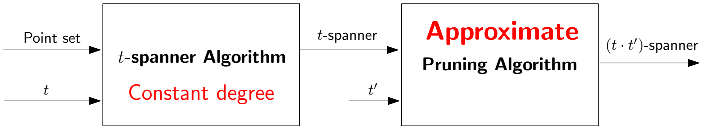
Bose, Carmi, Farshi, Maheshvari and Smid (2008)

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

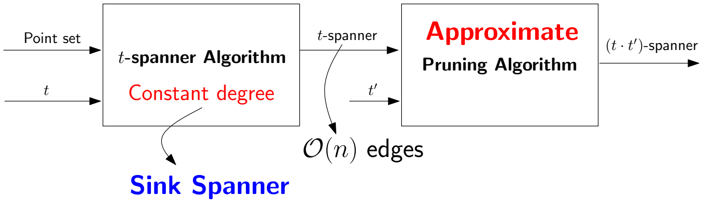
Apx. Greedy Algorithm

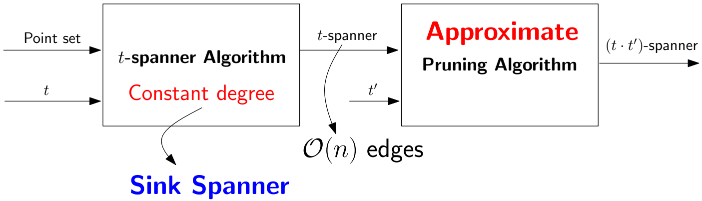


Apx. Greedy Algorithm



Apx. Greedy Algorithm





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

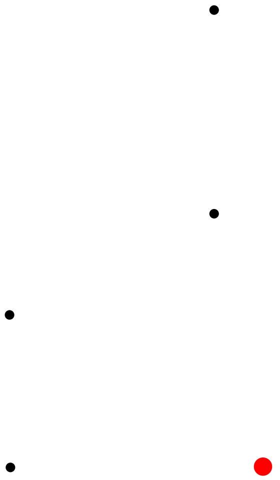
Storage Complexity: $\mathcal{O}(n)$.

Θ-Graph Algorithm

$t = 3, \Theta = \pi/6$

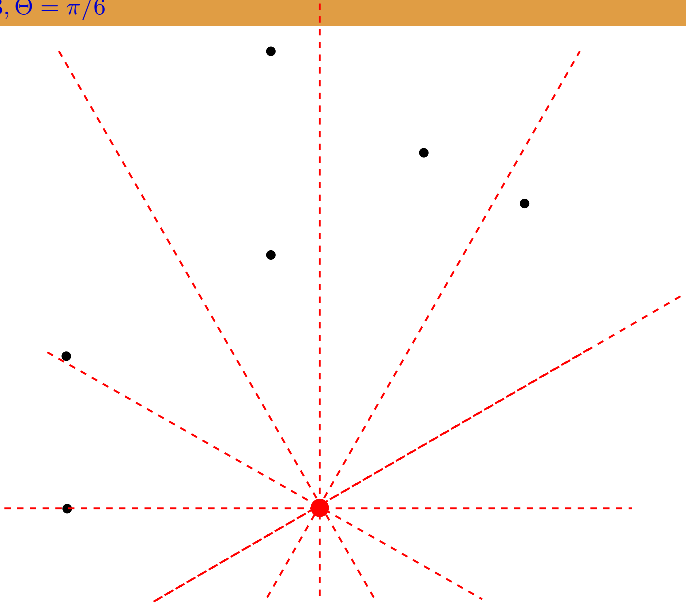
Θ-Graph Algorithm

$t = 3, \Theta = \pi/6$



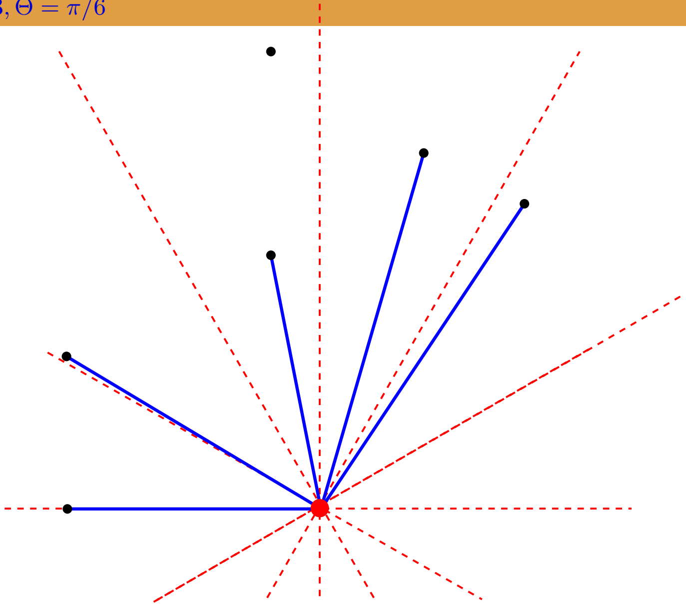
Θ-Graph Algorithm

$t = 3, \Theta = \pi/6$



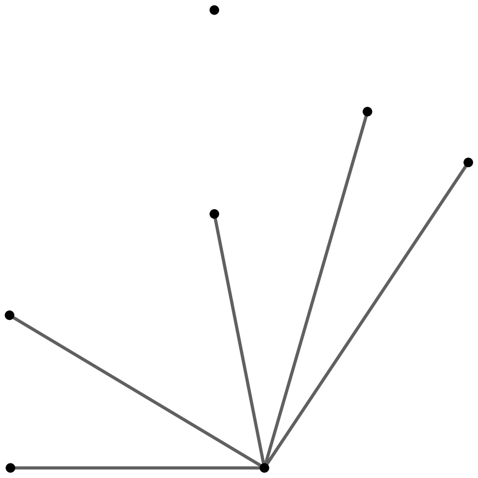
Θ-Graph Algorithm

$t = 3, \Theta = \pi/6$



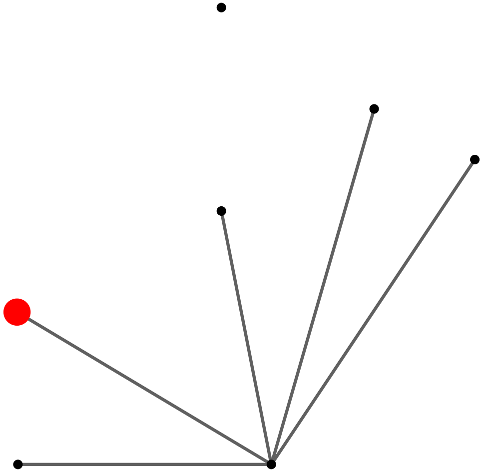
Θ-Graph Algorithm

$t = 3, \Theta = \pi/6$



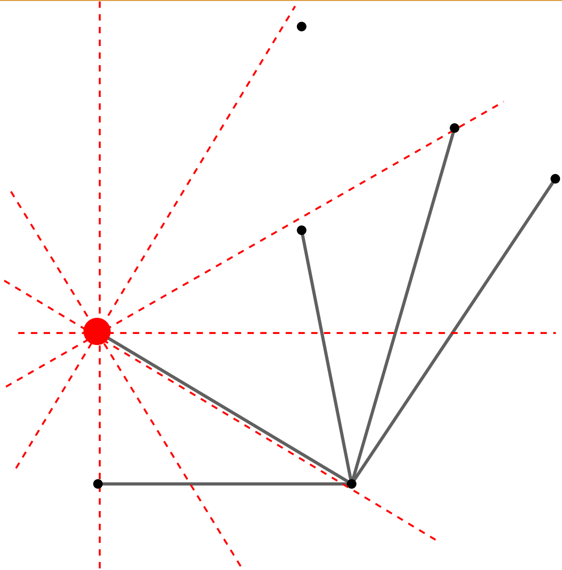
Θ-Graph Algorithm

$t = 3, \Theta = \pi/6$



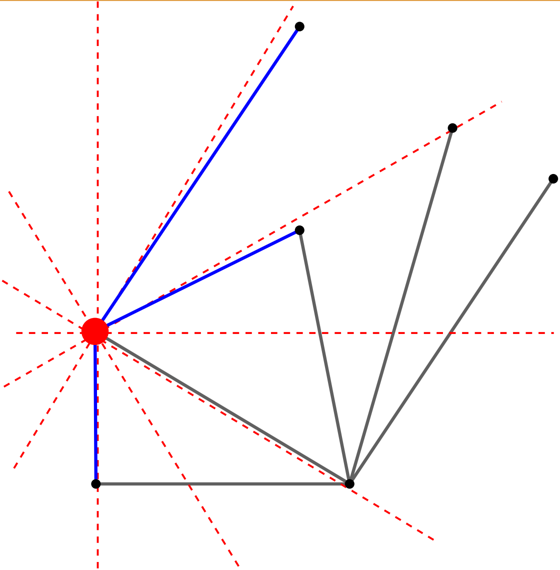
Θ-Graph Algorithm

$t = 3, \Theta = \pi/6$



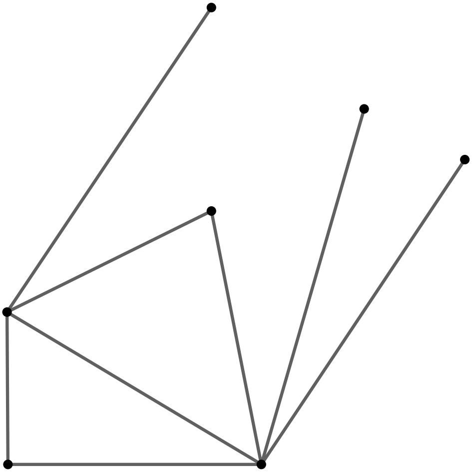
Θ-Graph Algorithm

$t = 3, \Theta = \pi/6$



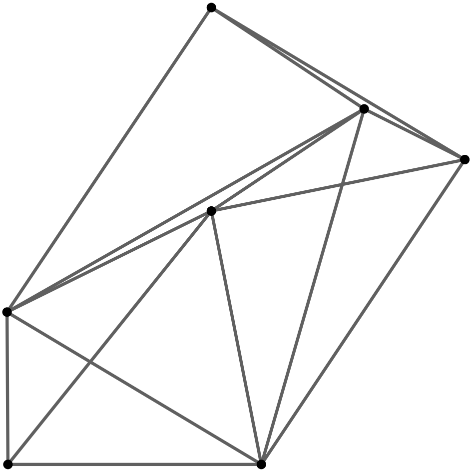
Θ-Graph Algorithm

$t = 3, \Theta = \pi/6$



Θ-Graph Algorithm

$t = 3, \Theta = \pi/6$



Θ-Graph Algorithm

Θ-Graph

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 $\theta = 2\pi/k$;

$E := \emptyset$;

for *each point* $u \in V$ **do**

  $C_1, \ldots, C_k :=$ non-overlapping cones with angle θ 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: $\mathcal{O}(n \log n)$.

Storage Complexity: $\mathcal{O}(n)$.

Θ-Graph Algorithm

Θ-Graph

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 $\theta = 2\pi/k$;

$E := \emptyset$;

for *each point* $u \in V$ **do**

  $C_1, \ldots, C_k :=$ non-overlapping cones with angle θ 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: $\mathcal{O}(n \log n)$.

Storage Complexity: $\mathcal{O}(n)$.

Variants of Θ-Graph Algorithm

Ordered Θ-Graph– $\mathcal{O}(\log n)$ maximum degree

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

Random Ordered Θ-Graph– $\mathcal{O}(\log n)$ 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– $\mathcal{O}(\log n)$ spanner diameter

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

Variants of Θ-Graph Algorithm

Ordered Θ-Graph– $\mathcal{O}(\log n)$ maximum degree

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

Random Ordered Θ-Graph– $\mathcal{O}(\log n)$ 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– $\mathcal{O}(\log n)$ spanner diameter

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

Variants of Θ-Graph Algorithm

Ordered Θ-Graph– $\mathcal{O}(\log n)$ maximum degree

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

Random Ordered Θ-Graph– $\mathcal{O}(\log n)$ 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– $\mathcal{O}(\log n)$ spanner diameter

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

Variants of Θ-Graph Algorithm

Ordered Θ-Graph– $\mathcal{O}(\log n)$ maximum degree

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

Random Ordered Θ-Graph– $\mathcal{O}(\log n)$ 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– $\mathcal{O}(\log n)$ spanner diameter

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

Sink Spanner

A variant of Θ-graph with bounded degree

Input: V and $t > 1$
Output: t-spanner $G(V, E)$
Construct a directed $\sqrt{t}$-spanner $\overrightarrow{G}$ 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 Θ-graph with bounded degree

Input: V and $t > 1$
Output: t-spanner $G(V, E)$
Construct a directed $\sqrt{t}$-spanner $\overrightarrow{G}$ 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 Θ-graph with $\mathcal{O}(\log n)$ 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: $\mathcal{O}(n \log n)$ Storage Complexity: $\mathcal{O}(n)$.

Skip-List Spanner

A variant of Θ-graph with $\mathcal{O}(\log n)$ 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: $\mathcal{O}(n \log n)$ Storage Complexity: $\mathcal{O}(n)$.

Well Separated Pair Decomposition (WSPD)

Well Separated Pair:

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

-
-

Geometric Spanner Networks

Well Separated Pair Decomposition (WSPD)

Well Separated Pair:

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

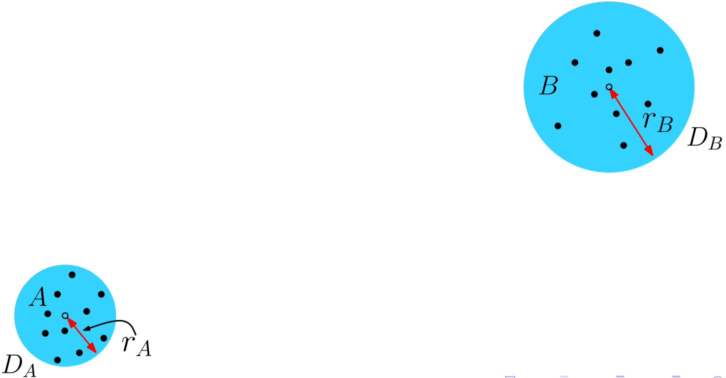
-
-

Well Separated Pair Decomposition (WSPD)

Well Separated Pair:

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

- $A \subseteq D_A$ and $B \subseteq D_B$.
-

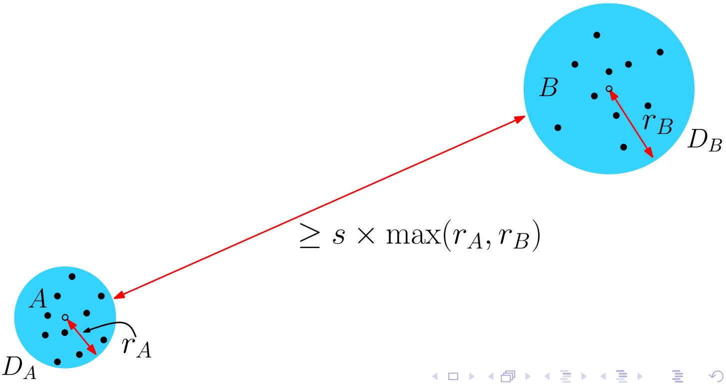


Well Separated Pair Decomposition (WSPD)

Well Separated Pair:

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

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



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 $O(s^d \cdot n)$ 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 : 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 : 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:

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 $\mathcal{O}(s^d \cdot n)$ 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: $\mathcal{O}(n)$.

-	Size	Weight	Degree	Time
Greedy spanner	$\mathcal{O}(n)$	$\mathcal{O}(wt(\mathrm{MST}))$	$\mathcal{O}(1)$	$\mathcal{O}(n^2 \log n)$
Apx. greedy spanner	$\mathcal{O}(n)$	$\mathcal{O}(wt(\mathrm{MST}))$	$\mathcal{O}(1)$	$\mathcal{O}(n \log n)$
Θ-graph	$\mathcal{O}(n)$	$\Theta(n \cdot wt(\mathrm{MST}))$	$\Theta(n)$	$\mathcal{O}(n \log n)$
O. Θ-graph	$\mathcal{O}(n)$	$\mathcal{O}(n \cdot wt(\mathrm{MST}))$	$\mathcal{O}(\log n)$	$\mathcal{O}(n \log n)$
WSPD spanner	$\mathcal{O}(n)$	$\mathcal{O}(\log n \cdot wt(\mathrm{MST}))$	$\Theta(n)$	$\mathcal{O}(n \log n)$
Sink-spanner	$\mathcal{O}(n)$	$\mathcal{O}(n \cdot wt(\mathrm{MST}))$	$\mathcal{O}(1)$	$\mathcal{O}(n \log n)$
Skip-list spanner	$\mathcal{O}(n)^*$	$\Theta(n \cdot wt(\mathrm{MST}))^*$	$\Theta(n)$	$\mathcal{O}(n \log n)^*$

(*): Expected with high probability

Applications

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:

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 2-approximation algorithm for metric spaces by Rosenkrantz *et al.* (1977).
- A 1.5-approximation algorithm by Christofides *et al.* (1976).
- A PTAS ($(1+\varepsilon)$-approx. Alg.) for geometric case by Arora (1998) and Mitchell (1999).
- A PTAS for geometric case using spanners by Rao and Smith (1998).

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 $t \cdot wt(\mathrm{TSP}(P))$.

Theorem (Rao and Smith, 1998)

Given a $(1+\varepsilon)$-spanner of a set of n points with $\mathcal{O}(n)$ size and $\mathcal{O}(wt(\mathrm{MST}))$ weight, we can compute a $(1+\varepsilon)$-approximation of $\mathrm{TSP}(P)$ in $\mathcal{O}(n \log n)$ time.

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 $t \cdot wt(\mathrm{TSP}(P))$.

Theorem (Rao and Smith, 1998)

Given a $(1+\varepsilon)$-spanner of a set of n points with $\mathcal{O}(n)$ size and $\mathcal{O}(wt(\mathrm{MST}))$ weight, we can compute a $(1+\varepsilon)$-approximation of $\mathrm{TSP}(P)$ in $\mathcal{O}(n \log n)$ time.

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:

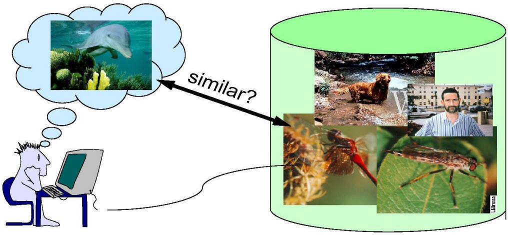
For any t-spanner G for P, there is a tour of P in G, whose weight is at most $t \cdot wt(\mathrm{TSP}(P))$.

Theorem (Rao and Smith, 1998)

Given a $(1+\varepsilon)$-spanner of a set of n points with $\mathcal{O}(n)$ size and $\mathcal{O}(wt(\mathrm{MST}))$ weight, we can compute a $(1+\varepsilon)$-approximation of $\mathrm{TSP}(P)$ in $\mathcal{O}(n \log n)$ time.

S. B. Rao and W. D. Smith, **Approximating Geometrical Graphs via "Spanners" and "Banyans"**, STOC'98, pp. 540–550, 1998.

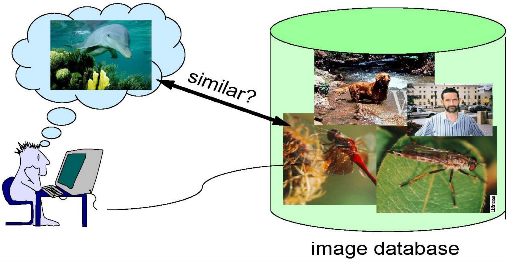
Applications
Metric space searching



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 $\mathcal{O}(n^2)$ 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 $\mathcal{O}(n^2)$ 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 $\mathcal{O}(n^2)$ 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 $\mathcal{O}(n^2)$ space (AESA).
- A t-spanner can be used as a sparse data structure to reduce the space.

G. Navarro, R. Paredes, and E. Chávez, **t-Spanners for metric space searching**, Data & Knowledge Engineering, pp. 820-854, 2007.

D. Russel and L. Guibas, **Exploring Protein Folding Trajectories Using Geometric Spanners**, Pacific Symposium on Biocomputing, pp. 40-51, 2005.

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 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.