Approximation Algorithms for Geometric Proximity Problems: Part I: Approximating Euclidean MSTs

David M. Mount

Department of Computer Science &
Institute for Advanced Computer Studies
University of Maryland, College Park

Joint with: Sunil Arya, Phong Dinh, Jerry Tan

HMI Workshop 2018

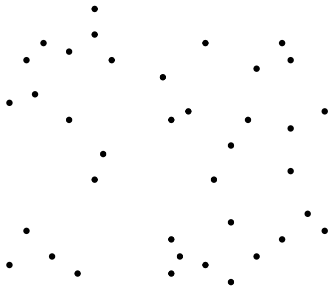
Euclidean Minimum Spanning Tree

Euclidean MST

Given a set P of points in space, compute the minimum spanning tree, where the edge weights are the distances

Assumptions:

- Points in $\mathbb{R}^d$, where d is constant
- Euclidean distance (generalizes to L_p distance)



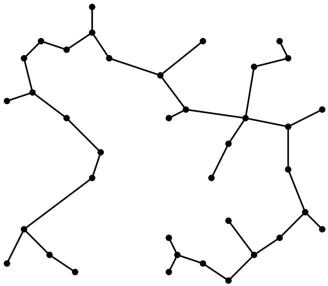
Euclidean Minimum Spanning Tree

Euclidean MST

Given a set P of points in space, compute the minimum spanning tree, where the edge weights are the distances

Assumptions:

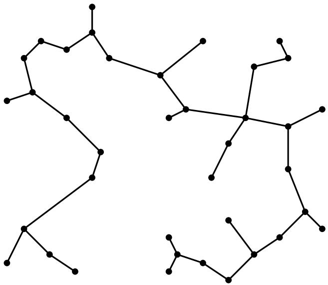
- Points in $\mathbb{R}^d$, where d is constant
- Euclidean distance (generalizes to L_p distance)



Euclidean Minimum Spanning Tree

Euclidean MST

Given a set P of points in space, compute the minimum spanning tree, where the edge weights are the distances



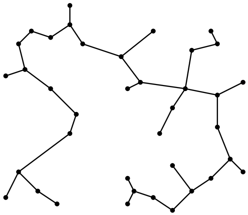
Assumptions:

- Points in $\mathbb{R}^d$, where d is constant
- Euclidean distance (generalizes to L_p distance)

Exact vs. Approximate MST

Exact in $\mathbb{R}^d$:

- Naive: $O(n^2)$ — Run Kruskal on complete graph
- Geometry helps, but not by much:
  Yao (1982) $\widetilde{O}(n^{2-\frac{1}{2^d}})$
  Agarwal *et al.* (1991) $\widetilde{O}(n^{2-\frac{4}{d}})$



Better performance through approximation?

Approximate MST

Return a spanning tree of weight at most $(1+\varepsilon) \cdot \mathrm{wt}(\mathrm{MST}(P))$.

Exact vs. Approximate MST

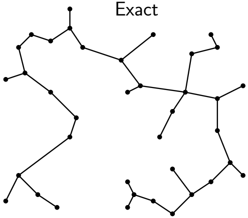
Exact in $\mathbb{R}^d$:

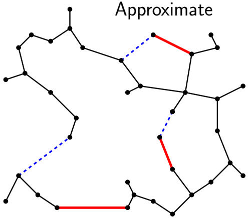
- Naive: $O(n^2)$ — Run Kruskal on complete graph
- Geometry helps, but not by much:
  Yao (1982) $\widetilde{O}(n^{2-\frac{1}{2^d}})$
  Agarwal *et al.* (1991) $\widetilde{O}(n^{2-\frac{4}{d}})$

Better performance through approximation?

Approximate MST

Return a spanning tree of weight at most $(1+\varepsilon) \cdot \mathrm{wt}(\mathrm{MST}(P))$.





Exact vs. Approximate MST

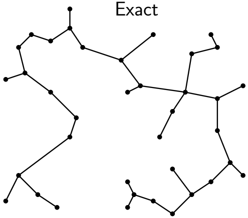
Exact in $\mathbb{R}^d$:

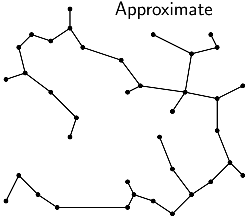
- Naive: $O(n^2)$ — Run Kruskal on complete graph
- Geometry helps, but not by much:
  Yao (1982) $\widetilde{O}(n^{2-\frac{1}{2^d}})$
  Agarwal *et al.* (1991) $\widetilde{O}(n^{2-\frac{4}{d}})$

Better performance through approximation?

Approximate MST

Return a spanning tree of weight at most $(1+\varepsilon)\cdot \mathrm{wt}(\mathrm{MST}(P))$.





Prior Work

- Exact in $\mathbb{R}^d$: Agarwal *et al.* (1991): $\widetilde{O}(n^{2-\frac{4}{d}})$
- ε-Approximate in $\mathbb{R}^d$: (Constant d. Ignoring log factors)

Vaidya (1991)	$\widetilde{O}(n/\varepsilon^d)$	quadtrees
Callahan and Kosaraju (1995)	$\widetilde{O}(n/\varepsilon^{d/2})$	WSPDs
Arya and Chan (2014)	$\widetilde{O}(n/\varepsilon^{d/3})$	DVDs
Arya, Fonseca, Mount (2017)	$\widetilde{O}(n/\varepsilon^{d/4})$	Macbeath regions
Chan (2017)	$\widetilde{O}(n/\varepsilon^{d/4})$	Polynomial method

- Weight approximation in sublinear time: Chazelle *et al.* (2005), Czumaj *et al.* (2005), Czumaj and Sohler (2009)

Talk Overview

- Preliminaries: WSPDs, MSTs, and Fast Lower Bounds
- Theory: ε-approximate MSTs in $\widetilde{O}(n/\varepsilon^2)$ time
- Practice: A more practical approach and implementation

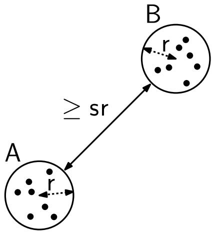
Talk Overview

- Preliminaries: WSPDs, MSTs, and Fast Lower Bounds
- Theory: ε-approximate MSTs in $\widetilde{O}(n/\varepsilon^2)$ time
- Practice: A more practical approach and implementation

Preliminaries - Well-Separated Pairs

WSPDs - Basic Facts: Let $P \subset \mathbb{R}^d$, $|P| = n$

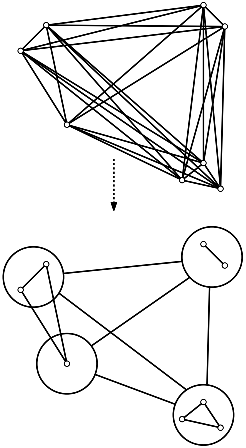
- A and B are s-well separated if they can be enclosed within balls of radius r separated by distance at least $s\,r$
- An s-WSPD of P is a set of s-well separated pairs $\{\{A_i, B_i\}\}_{i=1}^{k}$ that covers all the pairs of P
- Callahan and Kosaraju (1995): Can construct an s-WSPD of size $k = O((s\sqrt{d})^d n)$ in time $O(k + n \log n)$



Preliminaries - Well-Separated Pairs

WSPDs - Basic Facts: Let $P \subset \mathbb{R}^d$, $|P| = n$

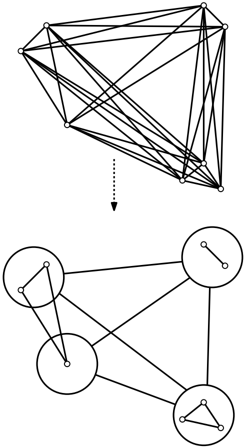
- A and B are s-well separated if they can be enclosed within balls of radius r separated by distance at least $s\,r$
- An s-WSPD of P is a set of s-well separated pairs $\{\{A_i, B_i\}\}_{i=1}^k$ that covers all the pairs of P
- Callahan and Kosaraju (1995): Can construct an s-WSPD of size $k = O((s\sqrt{d})^d n)$ in time $O(k + n \log n)$



Preliminaries - Well-Separated Pairs

WSPDs - Basic Facts: Let $P \subset \mathbb{R}^d$, $|P| = n$

- A and B are s-well separated if they can be enclosed within balls of radius r separated by distance at least $s\,r$
- An s-WSPD of P is a set of s-well separated pairs $\{\{A_i, B_i\}\}_{i=1}^{k}$ that covers all the pairs of P
- Callahan and Kosaraju (1995): Can construct an s-WSPD of size $k = O((s\sqrt{d})^d n)$ in time $O(k + n \log n)$



Well-Separated Pairs

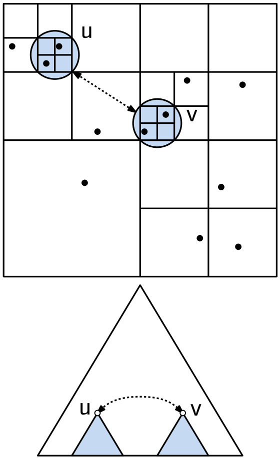
WSPDs - More Facts:

- Each well-separated pair is represented by a pair of nodes $\{u, v\}$ in a quadtree
- Let P_u and P_v denote the associated point sets
- Each well-separated pair also stores a pair of representative points, $p_u \in P_u$ and $p_v \in P_v$

Well-Separated Pairs

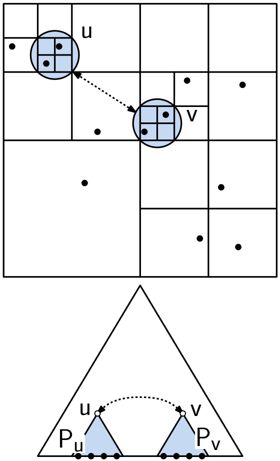
WSPDs - More Facts:

- Each well-separated pair is represented by a pair of nodes $\{u, v\}$ in a quadtree
- Let P_u and P_v denote the associated point sets
- Each well-separated pair also stores a pair of representative points, $p_u \in P_u$ and $p_v \in P_v$



WSPDs - More Facts:

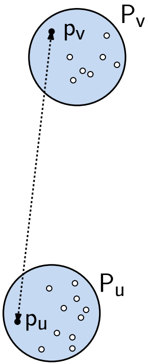
- Each well-separated pair is represented by a pair of nodes $\{u, v\}$ in a quadtree
- Let P_u and P_v denote the associated point sets
- Each well-separated pair also stores a pair of representative points, $p_u \in P_u$ and $p_v \in P_v$



Well-Separated Pairs

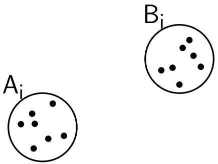
WSPDs - More Facts:

- Each well-separated pair is represented by a pair of nodes $\{u, v\}$ in a quadtree
- Let P_u and P_v denote the associated point sets
- Each well-separated pair also stores a pair of representative points, $p_u \in P_u$ and $p_v \in P_v$



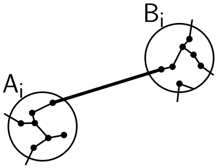
Useful Observations (Callahan and Kosaraju (1995))

- A 2-WSPD of size $O(n)$ can be constructed in time $O(n \log n)$
- Each pair of a 2-WSPD contributes at most one edge to the MST
- Given a 2-WSPD for P, form a graph G from the closest pair from each (A_i, B_i)
  - $|G| = O(n)$
  - $\mathrm{MST}(G) \Rightarrow \mathrm{EMST}(P)$
- ε-approximate closest pairs yield an ε-MST



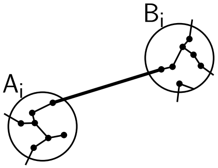
Useful Observations (Callahan and Kosaraju (1995))

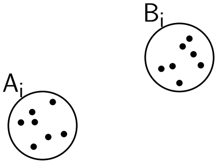
- A 2-WSPD of size $O(n)$ can be constructed in time $O(n \log n)$
- Each pair of a 2-WSPD contributes at most one edge to the MST
- Given a 2-WSPD for P, form a graph G from the closest pair from each (A_i, B_i)
  - $|G| = O(n)$
  - $\mathrm{MST}(G) \Rightarrow \mathrm{EMST}(P)$
- ε-approximate closest pairs yield an ε-MST



Useful Observations (Callahan and Kosaraju (1995))

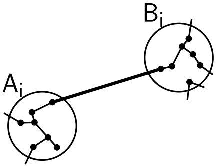
- A 2-WSPD of size $O(n)$ can be constructed in time $O(n \log n)$
- Each pair of a 2-WSPD contributes at most one edge to the MST
- Given a 2-WSPD for P, form a graph G from the closest pair from each (A_i, B_i)
  - $|G| = O(n)$
  - $\mathrm{MST}(G) \Rightarrow \mathrm{EMST}(P)$
- ε-approximate closest pairs yield an ε-MST

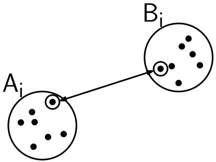




Useful Observations (Callahan and Kosaraju (1995))

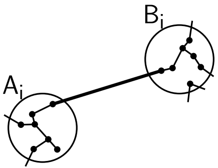
- A 2-WSPD of size $O(n)$ can be constructed in time $O(n \log n)$
- Each pair of a 2-WSPD contributes at most one edge to the MST
- Given a 2-WSPD for P, form a graph G from the closest pair from each (A_i, B_i)
  - $|G| = O(n)$
  - $\mathrm{MST}(G) \Rightarrow \mathrm{EMST}(P)$
- ε-approximate closest pairs yield an ε-MST

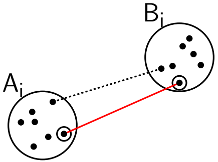




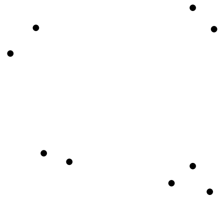
Useful Observations (Callahan and Kosaraju (1995))

- A 2-WSPD of size $O(n)$ can be constructed in time $O(n \log n)$
- Each pair of a 2-WSPD contributes at most one edge to the MST
- Given a 2-WSPD for P, form a graph G from the closest pair from each (A_i, B_i)
  - $|G| = O(n)$
  - $\mathrm{MST}(G) \Rightarrow \mathrm{EMST}(P)$
- ε-approximate closest pairs yield an ε-MST





Simple WSPD-Based Algorithm



ε-Approximate MST Algorithm:

- Compute an s-WSPD, for $s = 4(2+\varepsilon)/\varepsilon$
- For each WSP $\{u, v\}$, add $\{p_u, p_v\}$ to G
- Compute $\mathrm{MST}(G)$ and return

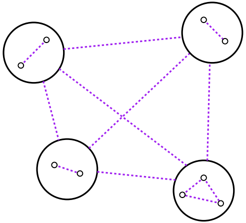
Running time:

- Dominated by MST time: $O((n/\varepsilon^d)\log(n/\varepsilon))$

Simple WSPD-Based Algorithm

ε-Approximate MST Algorithm:

- Compute an s-WSPD, for $s = 4(2+\varepsilon)/\varepsilon$
- For each WSP $\{u, v\}$, add $\{p_u, p_v\}$ to G
- Compute $\mathrm{MST}(G)$ and return



Running time:

- Dominated by MST time: $O((n/\varepsilon^d)\log(n/\varepsilon))$

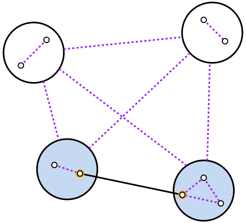
Simple WSPD-Based Algorithm

ε-Approximate MST Algorithm:

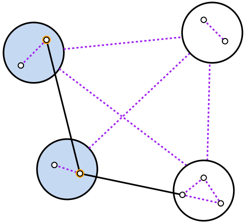
- Compute an s-WSPD, for $s = 4(2+\varepsilon)/\varepsilon$
- For each WSP $\{u, v\}$, add $\{p_u, p_v\}$ to G
- Compute MST(G) and return

Running time:

- Dominated by MST time: $O((n/\varepsilon^d)\log(n/\varepsilon))$



Simple WSPD-Based Algorithm



ε-Approximate MST Algorithm:

- Compute an s-WSPD, for $s = 4(2+\varepsilon)/\varepsilon$
- For each WSP $\{u, v\}$, add $\{p_u, p_v\}$ to G
- Compute $\mathrm{MST}(G)$ and return

Running time:

- Dominated by MST time: $O((n/\varepsilon^d)\log(n/\varepsilon))$

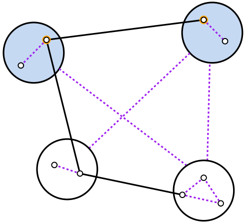
Simple WSPD-Based Algorithm

ε-Approximate MST Algorithm:

- Compute an s-WSPD, for $s = 4(2+\varepsilon)/\varepsilon$
- For each WSP $\{u, v\}$, add $\{p_u, p_v\}$ to G
- Compute MST(G) and return

Running time:

- Dominated by MST time: $O((n/\varepsilon^d)\log(n/\varepsilon))$



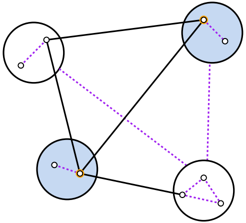
Simple WSPD-Based Algorithm

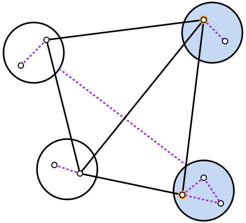
ε-Approximate MST Algorithm:

- Compute an s-WSPD, for $s = 4(2+\varepsilon)/\varepsilon$
- For each WSP $\{u, v\}$, add $\{p_u, p_v\}$ to G
- Compute MST(G) and return

Running time:

- Dominated by MST time: $O((n/\varepsilon^d)\log(n/\varepsilon))$





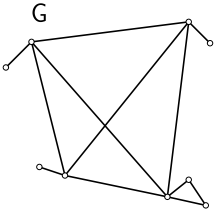
ε-Approximate MST Algorithm:

- Compute an s-WSPD, for $s = 4(2 + \varepsilon)/\varepsilon$
- For each WSP $\{u, v\}$, add $\{p_u, p_v\}$ to G
- Compute MST(G) and return

Running time:

- Dominated by MST time: $O((n/\varepsilon^d)\log(n/\varepsilon))$

Simple WSPD-Based Algorithm



ε-Approximate MST Algorithm:

- Compute an s-WSPD, for $s = 4(2+\varepsilon)/\varepsilon$
- For each WSP $\{u, v\}$, add $\{p_u, p_v\}$ to G
- Compute MST(G) and return

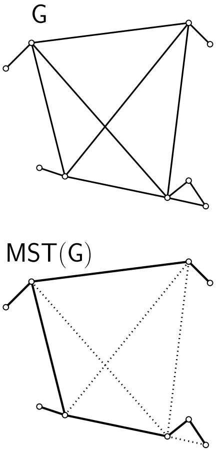
Running time:

- Dominated by MST time: $O((n/\varepsilon^d)\log(n/\varepsilon))$

Simple WSPD-Based Algorithm

ε-Approximate MST Algorithm:

- Compute an s-WSPD, for $s = 4(2+\varepsilon)/\varepsilon$
- For each WSP $\{u, v\}$, add $\{p_u, p_v\}$ to G
- Compute $\mathrm{MST}(G)$ and return



Running time:

- Dominated by MST time: $O((n/\varepsilon^d)\log(n/\varepsilon))$

Simple WSPD-Based Algorithm

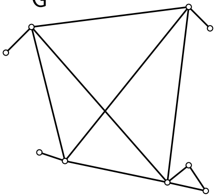
ε-Approximate MST Algorithm:

- Compute an s-WSPD, for $s = 4(2+\varepsilon)/\varepsilon$
- For each WSP $\{u, v\}$, add $\{p_u, p_v\}$ to G
- Compute $\mathrm{MST}(G)$ and return

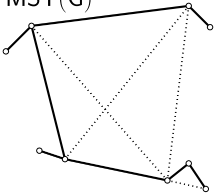
Running time:

- Dominated by MST time: $O((n/\varepsilon^d)\log(n/\varepsilon))$

G



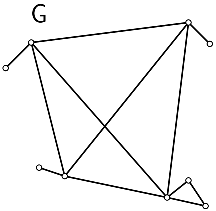
MST(G)

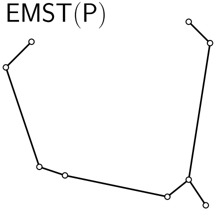


Correctness of the Simple WSPD-Based Algorithm

Correctness:

- Fact: G is a $(1+\varepsilon)$-spanner for $P \Leftrightarrow$
  Each $p, q \in P$ joined by a path of length $\leq (1+\varepsilon)\|pq\|$.
- Replace each edge of EMST(P) with its spanner path
  Total weight increases by at most $(1+\varepsilon)$
- Result G' spans P and has weight $\leq (1+\varepsilon)\text{EMST}(P)$
- The weight of $\text{MST}(G)$ can be no larger

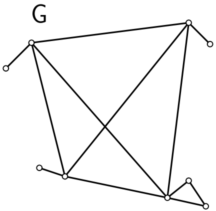


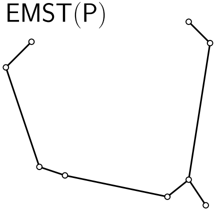


Correctness of the Simple WSPD-Based Algorithm

Correctness:

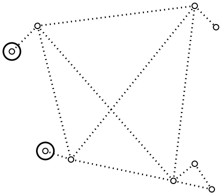
- Fact: G is a $(1+\varepsilon)$-spanner for P $\Leftrightarrow$
  Each $p, q \in P$ joined by a path of length $\leq (1+\varepsilon)\|pq\|$.
- Replace each edge of EMST(P) with its spanner path
  Total weight increases by at most $(1+\varepsilon)$
- Result G' spans P and has weight $\leq (1+\varepsilon)\text{EMST}(P)$
- The weight of $\text{MST}(G)$ can be no larger



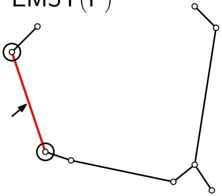


Correctness:

- Fact: G is a $(1+\varepsilon)$-spanner for $P \Leftrightarrow$
  Each $p, q \in P$ joined by a path of length $\leq (1+\varepsilon)\|pq\|$.
- Replace each edge of EMST(P) with its spanner path
  Total weight increases by at most $(1+\varepsilon)$
- Result G' spans P and has weight $\leq (1+\varepsilon)\mathrm{EMST}(P)$
- The weight of $\mathrm{MST}(G)$ can be no larger



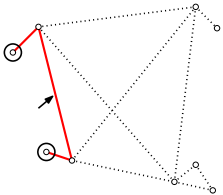
EMST(P)



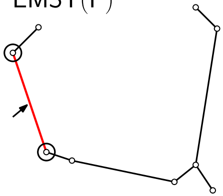
Correctness of the Simple WSPD-Based Algorithm

Correctness:

- Fact: G is a $(1+\varepsilon)$-spanner for $P \Leftrightarrow$
  Each $p, q \in P$ joined by a path of length $\leq (1+\varepsilon)\|pq\|$.
- Replace each edge of EMST(P) with its spanner path
  Total weight increases by at most $(1+\varepsilon)$
- Result G' spans P and has weight $\leq (1+\varepsilon)\text{EMST}(P)$
- The weight of $\text{MST}(G)$ can be no larger



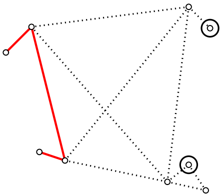
EMST(P)



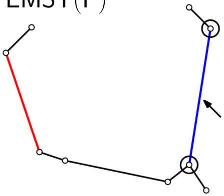
Correctness of the Simple WSPD-Based Algorithm

Correctness:

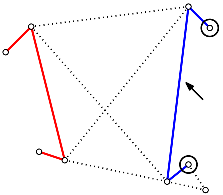
- Fact: G is a $(1+\varepsilon)$-spanner for $P \Leftrightarrow$
  Each $p, q \in P$ joined by a path of length $\leq (1+\varepsilon)\|pq\|$.
- Replace each edge of EMST(P) with its spanner path
  Total weight increases by at most $(1+\varepsilon)$
- Result G' spans P and has weight $\leq (1+\varepsilon)\text{EMST}(P)$
- The weight of $\text{MST}(G)$ can be no larger



EMST(P)



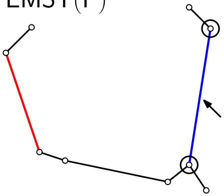
Correctness of the Simple WSPD-Based Algorithm



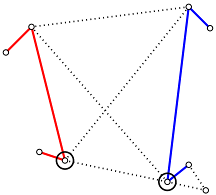
Correctness:

- Fact: G is a $(1+\varepsilon)$-spanner for $P \Leftrightarrow$
  Each $p, q \in P$ joined by a path of length $\leq (1+\varepsilon)\|pq\|$.
- Replace each edge of EMST(P) with its spanner path
  Total weight increases by at most $(1+\varepsilon)$
- Result G' spans P and has weight $\leq (1+\varepsilon)\mathrm{EMST}(P)$
- The weight of $\mathrm{MST}(G)$ can be no larger

EMST(P)

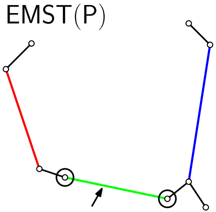


Correctness of the Simple WSPD-Based Algorithm

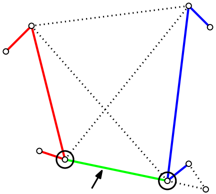


Correctness:

- Fact: G is a $(1+\varepsilon)$-spanner for $P \Leftrightarrow$
  Each $p, q \in P$ joined by a path of length $\leq (1+\varepsilon)\|pq\|$.
- Replace each edge of EMST(P) with its spanner path
  Total weight increases by at most $(1+\varepsilon)$
- Result G' spans P and has weight $\leq (1+\varepsilon)\text{EMST}(P)$
- The weight of $\text{MST}(G)$ can be no larger



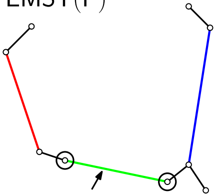
Correctness of the Simple WSPD-Based Algorithm



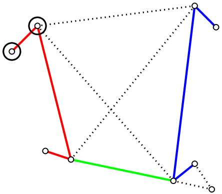
Correctness:

- Fact: G is a $(1+\varepsilon)$-spanner for $P \Leftrightarrow$
  Each $p, q \in P$ joined by a path of length $\leq (1+\varepsilon)\|pq\|$.
- Replace each edge of EMST(P) with its spanner path
  Total weight increases by at most $(1+\varepsilon)$
- Result G' spans P and has weight $\leq (1+\varepsilon)\mathrm{EMST}(P)$
- The weight of $\mathrm{MST}(G)$ can be no larger

EMST(P)



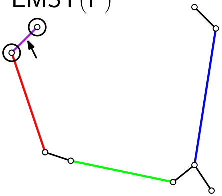
Correctness of the Simple WSPD-Based Algorithm



Correctness:

- Fact: G is a $(1+\varepsilon)$-spanner for $P \Leftrightarrow$
  Each $p, q \in P$ joined by a path of length $\leq (1+\varepsilon)\|pq\|$.
- Replace each edge of EMST(P) with its spanner path
  Total weight increases by at most $(1+\varepsilon)$
- Result G' spans P and has weight $\leq (1+\varepsilon)\mathrm{EMST}(P)$
- The weight of $\mathrm{MST}(G)$ can be no larger

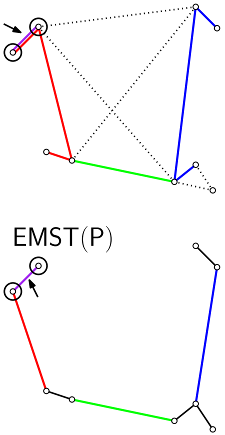
EMST(P)



Correctness of the Simple WSPD-Based Algorithm

Correctness:

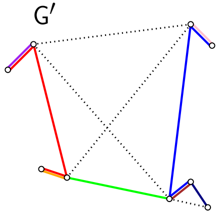
- Fact: G is a $(1+\varepsilon)$-spanner for $P \Leftrightarrow$
  Each $p, q \in P$ joined by a path of length $\leq (1+\varepsilon)\|pq\|$.
- Replace each edge of EMST(P) with its spanner path
  Total weight increases by at most $(1+\varepsilon)$
- Result G' spans P and has weight $\leq (1+\varepsilon)\mathrm{EMST}(P)$
- The weight of $\mathrm{MST}(G)$ can be no larger

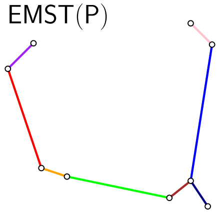


Correctness of the Simple WSPD-Based Algorithm

Correctness:

- Fact: G is a $(1+\varepsilon)$-spanner for $P \Leftrightarrow$
  Each $p, q \in P$ joined by a path of length $\leq (1+\varepsilon)\|pq\|$.
- Replace each edge of EMST(P) with its spanner path
  Total weight increases by at most $(1+\varepsilon)$
- Result G' spans P and has weight $\leq (1+\varepsilon)\mathrm{EMST}(P)$
- The weight of $\mathrm{MST}(G)$ can be no larger

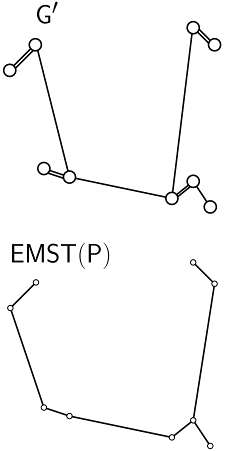




Correctness of the Simple WSPD-Based Algorithm

Correctness:

- Fact: G is a $(1+\varepsilon)$-spanner for $P \Leftrightarrow$
  Each $p, q \in P$ joined by a path of length $\leq (1+\varepsilon)\|pq\|$.
- Replace each edge of EMST(P) with its spanner path
  Total weight increases by at most $(1+\varepsilon)$
- Result G' spans P and has weight $\leq (1+\varepsilon)\mathrm{EMST}(P)$
- The weight of $\mathrm{MST}(G)$ can be no larger



Speeding This Up

$(1/\varepsilon)$-WSPD has too many pairs ($O(n/\varepsilon^d)$)

- Build a 2-WSPD $\Rightarrow$ Only $O(n)$ pairs
- Be smarter about computing ε-approximate closest pairs?
  $\Rightarrow \widetilde{O}(n/\varepsilon^{d/2})$ [CK95], $\widetilde{O}(n/\varepsilon^{d/3})$ [AC14], $\widetilde{O}(n/\varepsilon^{d/4})$ [Ch17, AFM17] $\leftarrow$ tough!
- Use lower bounds on MST weight more judiciously $\leftarrow$ our approach

Speeding This Up

$(1/\varepsilon)$-WSPD has too many pairs ($O(n/\varepsilon^d)$)

- Build a 2-WSPD $\Rightarrow$ Only $O(n)$ pairs
- Be smarter about computing ε-approximate closest pairs?
  $\Rightarrow$ $\widetilde{O}(n/\varepsilon^{d/2})$ [CK95], $\widetilde{O}(n/\varepsilon^{d/3})$ [AC14], $\widetilde{O}(n/\varepsilon^{d/4})$ [Ch17, AFM17] $\longleftarrow$ tough!
- Use lower bounds on MST weight more judiciously $\longleftarrow$ our approach

Speeding This Up

$(1/\varepsilon)$-WSPD has too many pairs ($O(n/\varepsilon^d)$)

- Build a 2-WSPD $\Rightarrow$ Only $O(n)$ pairs
- Be smarter about computing ε-approximate closest pairs?
  $\Rightarrow \widetilde{O}(n/\varepsilon^{d/2})$ [CK95], $\widetilde{O}(n/\varepsilon^{d/3})$ [AC14], $\widetilde{O}(n/\varepsilon^{d/4})$ [Ch17, AFM17] $\longleftarrow$ tough!
- Use lower bounds on MST weight more judiciously $\longleftarrow$ our approach

Speeding This Up

$(1/\varepsilon)$-WSPD has too many pairs ($O(n/\varepsilon^d)$)

- Build a 2-WSPD $\Rightarrow$ Only $O(n)$ pairs
- Be smarter about computing ε-approximate closest pairs?
  $\Rightarrow \widetilde{O}(n/\varepsilon^{d/2})$ [CK95], $\widetilde{O}(n/\varepsilon^{d/3})$ [AC14], $\widetilde{O}(n/\varepsilon^{d/4})$ [Ch17, AFM17] $\leftarrow$ tough!
- Use lower bounds on MST weight more judiciously $\leftarrow$ our approach

Speeding This Up

$(1/\varepsilon)$-WSPD has too many pairs ($O(n/\varepsilon^d)$)

- Build a 2-WSPD $\Rightarrow$ Only $O(n)$ pairs
- Be smarter about computing ε-approximate closest pairs?
  $\Rightarrow \widetilde{O}(n/\varepsilon^{d/2})$ [CK95], $\widetilde{O}(n/\varepsilon^{d/3})$ [AC14], $\widetilde{O}(n/\varepsilon^{d/4})$ [Ch17, AFM17] $\leftarrow$ tough!
- Use lower bounds on MST weight more judiciously $\leftarrow$ our approach

Speeding This Up

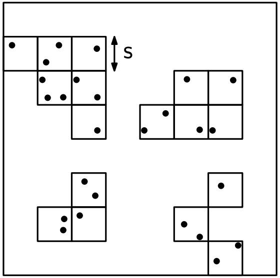
$(1/\varepsilon)$-WSPD has too many pairs ($O(n/\varepsilon^d)$)

- Build a 2-WSPD $\Rightarrow$ Only $O(n)$ pairs
- Be smarter about computing ε-approximate closest pairs?
  $\Rightarrow$ $\widetilde{O}(n/\varepsilon^{d/2})$ [CK95], $\widetilde{O}(n/\varepsilon^{d/3})$ [AC14], $\widetilde{O}(n/\varepsilon^{d/4})$ [Ch17, AFM17] $\leftarrow$ tough!
- Use lower bounds on MST weight more judiciously $\leftarrow$ our approach

Fast MST Lower Bound

Lemma [Czumaj *et al.* 2005]

Consider a grid of side length s in $\mathbb{R}^d$. Let m be the number of grid boxes containing at least one point of P. Then there is a constant c (depending on d) such that $\mathrm{wt}(\mathrm{MST}(P)) \geq s\,m/c$.

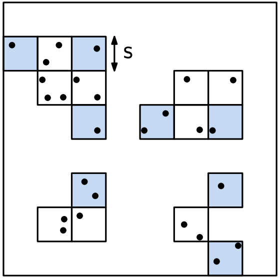


Proof:

- Color the grid with 2^d colors. Boxes of the same color are separated by distance $\geq s$
- Some color class has at least $m/2^d$ boxes
- The cost of connecting these boxes is $\Omega(s\,m)$

Lemma [Czumaj *et al.* 2005]

Consider a grid of side length s in $\mathbb{R}^d$. Let m be the number of grid boxes containing at least one point of P. Then there is a constant c (depending on d) such that $\text{wt}(\text{MST}(P)) \geq s\,m/c$.



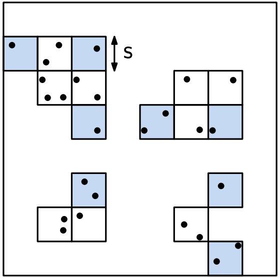
Proof:

- Color the grid with 2^d colors. Boxes of the same color are separated by distance $\geq s$
- Some color class has at least $m/2^d$ boxes
- The cost of connecting these boxes is $\Omega(s\,m)$

Fast MST Lower Bound

Lemma [Czumaj *et al.* 2005]

Consider a grid of side length s in $\mathbb{R}^d$. Let m be the number of grid boxes containing at least one point of P. Then there is a constant c (depending on d) such that $\mathrm{wt}(\mathrm{MST}(P)) \geq s\,m/c$.



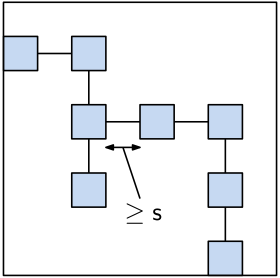
Proof:

- Color the grid with 2^d colors. Boxes of the same color are separated by distance $\geq s$
- Some color class has at least $m/2^d$ boxes
- The cost of connecting these boxes is $\Omega(s\,m)$

Fast MST Lower Bound

Lemma [Czumaj *et al.* 2005]

Consider a grid of side length s in $\mathbb{R}^d$. Let m be the number of grid boxes containing at least one point of P. Then there is a constant c (depending on d) such that $\mathrm{wt}(\mathrm{MST}(P)) \geq s\,m/c$.



Proof:

- Color the grid with 2^d colors. Boxes of the same color are separated by distance $\geq s$
- Some color class has at least $m/2^d$ boxes
- The cost of connecting these boxes is $\Omega(s\,m)$

Talk Overview

- Preliminaries: WSPDs, MSTs, and Fast Lower Bounds
- Theory: ε-approximate MSTs in $\widetilde{O}(n/\varepsilon^2)$ time (joint with Sunil Arya)
- Practice: A more practical approach and implementation

Talk Overview

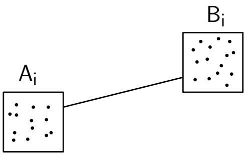
- Preliminaries: WSPDs, MSTs, and Fast Lower Bounds
- Theory: ε-approximate MSTs in $\widetilde{O}(n/\varepsilon^2)$ time (joint with Sunil Arya)
- Practice: A more practical approach and implementation

Talk Overview

- Preliminaries: WSPDs, MSTs, and Fast Lower Bounds
- Theory: ε-approximate MSTs in $\widetilde{O}(n/\varepsilon^2)$ time (joint with Sunil Arya)
  - Simple, deterministic, using standard data structures
  - Novel amortized cost analysis
  - The $1/\varepsilon^2$ factor is independent of dimension
- Practice: A more practical approach and implementation

Simple (Slow) Algorithm

- Compute a 2-WSPD for P
- Each box stores a representative point
- For each WSP (A_i, B_i):
  - Let s be the box size. Subdivide A_i and B_i until the box diameter $\leq \varepsilon s$
  - $(p_i, q_i) \leftarrow$ closest pair of box representatives
- $G \leftarrow$ closest pairs. Return MST(G)



Slow! $O(n/(\varepsilon^d)^2) = O(n/\varepsilon^{2d})$.

Worst case arises when pairs have many boxes.

But, this only occurs when MST cost is high.

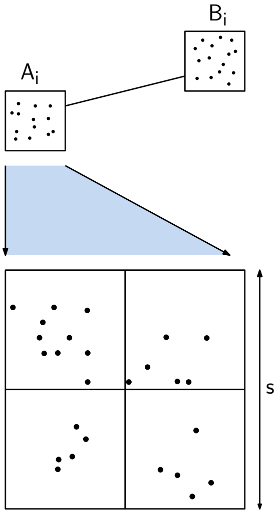
Simple (Slow) Algorithm

- Compute a 2-WSPD for P
- Each box stores a representative point
- For each WSP (A_i, B_i):
  - Let s be the box size. Subdivide A_i and B_i until the box diameter $\leq \varepsilon s$
  - $(p_i, q_i) \leftarrow$ closest pair of box representatives
- $G \leftarrow$ closest pairs. Return MST(G)

Slow! $O(n/(\varepsilon^d)^2) = O(n/\varepsilon^{2d})$.

Worst case arises when pairs have many boxes.

But, this only occurs when MST cost is high.



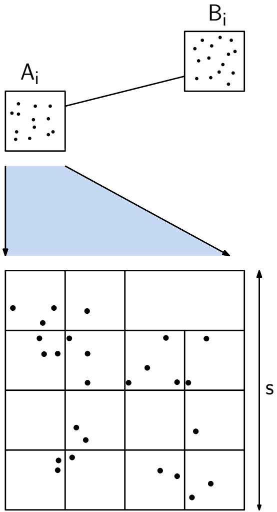
Simple (Slow) Algorithm

- Compute a 2-WSPD for P
- Each box stores a representative point
- For each WSP (A_i, B_i):
  - Let s be the box size. Subdivide A_i and B_i until the box diameter $\leq \varepsilon s$
  - $(p_i, q_i) \leftarrow$ closest pair of box representatives
- $G \leftarrow$ closest pairs. Return MST(G)

Slow! $O(n/(\varepsilon^d)^2) = O(n/\varepsilon^{2d})$.

Worst case arises when pairs have many boxes.

But, this only occurs when MST cost is high.



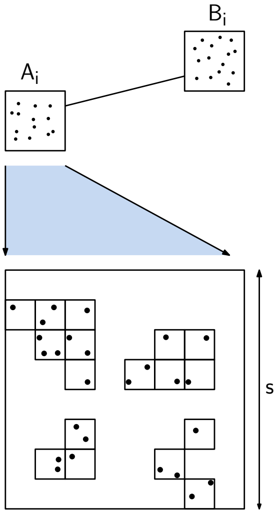
Simple (Slow) Algorithm

- Compute a 2-WSPD for P
- Each box stores a representative point
- For each WSP (A_i, B_i):
  - Let s be the box size. Subdivide A_i and B_i until the box diameter $\leq \varepsilon s$
  - $(p_i, q_i) \leftarrow$ closest pair of box representatives
- $G \leftarrow$ closest pairs. Return MST(G)

Slow! $O(n/(\varepsilon^d)^2) = O(n/\varepsilon^{2d})$.

Worst case arises when pairs have many boxes.

But, this only occurs when MST cost is high.



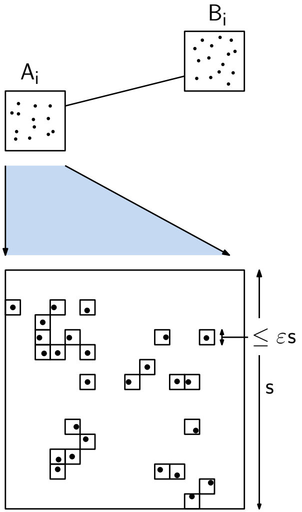
Simple (Slow) Algorithm

- Compute a 2-WSPD for P
- Each box stores a representative point
- For each WSP (A_i, B_i):
  - Let s be the box size. Subdivide A_i and B_i until the box diameter $\leq \varepsilon s$
  - $(p_i, q_i) \leftarrow$ closest pair of box representatives
- $G \leftarrow$ closest pairs. Return MST(G)

Slow! $O(n/(\varepsilon^d)^2) = O(n/\varepsilon^{2d})$.

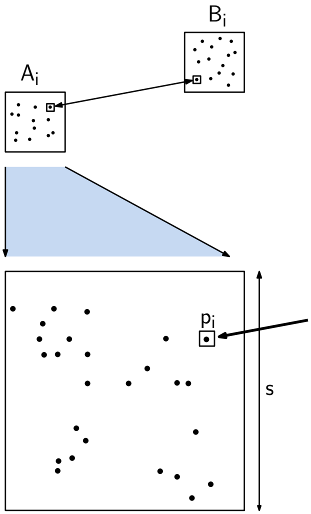
Worst case arises when pairs have many boxes.

But, this only occurs when MST cost is high.



Simple (Slow) Algorithm

- Compute a 2-WSPD for P
- Each box stores a representative point
- For each WSP (A_i, B_i):
  - Let s be the box size. Subdivide A_i and B_i until the box diameter $\leq \varepsilon s$
  - $(p_i, q_i) \leftarrow$ closest pair of box representatives
- $G \leftarrow$ closest pairs. Return $\mathrm{MST}(G)$



Slow! $O(n/(\varepsilon^d)^2) = O(n/\varepsilon^{2d})$.

Worst case arises when pairs have many boxes.

But, this only occurs when MST cost is high.

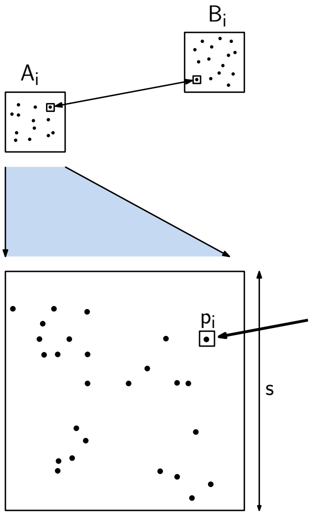
Simple (Slow) Algorithm

- Compute a 2-WSPD for P
- Each box stores a representative point
- For each WSP (A_i, B_i):
  - Let s be the box size. Subdivide A_i and B_i until the box diameter $\leq \varepsilon s$
  - $(p_i, q_i) \leftarrow$ closest pair of box representatives
- $G \leftarrow$ closest pairs. Return $\mathrm{MST}(G)$

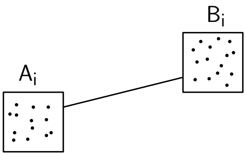
Slow! $O(n/(\varepsilon^d)^2) = O(n/\varepsilon^{2d})$.

Worst case arises when pairs have many boxes.

But, this only occurs when MST cost is high.



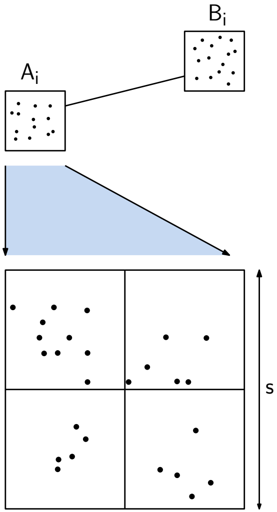
A Smarter/Sloppier Algorithm



- Compute a 2-WSPD for P
- Each box stores a representative point
- For each (A_i, B_i) approximate the closest pair:
  - Let s be the box size. Subdivide A_i and B_i until either:
    - Box diameter $\leq \varepsilon s$ — or —
    - The number of nonempty boxes $\geq c/\varepsilon$ (for some constant c)
  - $(p_i, q_i) \leftarrow$ closest pair of box representatives
- $G \leftarrow$ closest pairs. Return $\mathrm{MST}(G)$

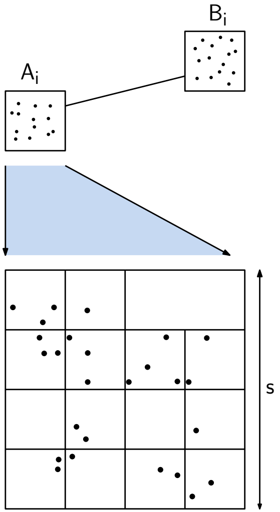
A Smarter/Sloppier Algorithm

- Compute a 2-WSPD for P
- Each box stores a representative point
- For each (A_i, B_i) approximate the closest pair:
  - Let s be the box size. Subdivide A_i and B_i until either:
    - Box diameter $\leq \varepsilon s$ — or —
    - The number of nonempty boxes $\geq c/\varepsilon$ (for some constant c)
  - $(p_i, q_i) \leftarrow$ closest pair of box representatives
- $G \leftarrow$ closest pairs. Return $\mathrm{MST}(G)$



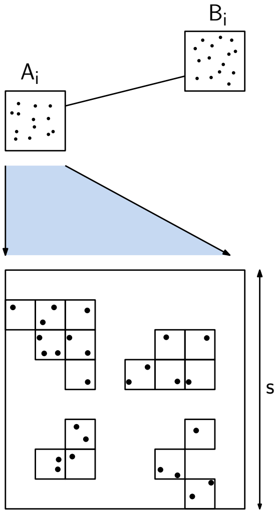
A Smarter/Sloppier Algorithm

- Compute a 2-WSPD for P
- Each box stores a representative point
- For each (A_i, B_i) approximate the closest pair:
  - Let s be the box size. Subdivide A_i and B_i until either:
    - Box diameter $\leq \varepsilon s$ — or —
    - The number of nonempty boxes $\geq c/\varepsilon$ (for some constant c)
  - $(p_i, q_i) \leftarrow$ closest pair of box representatives
- $G \leftarrow$ closest pairs. Return $\mathrm{MST}(G)$



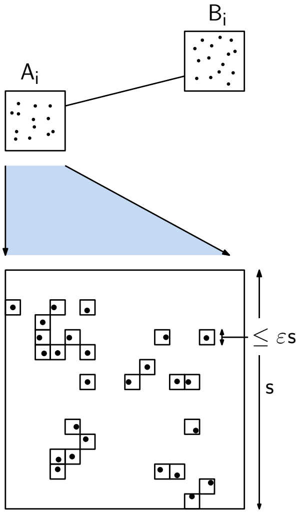
A Smarter/Sloppier Algorithm

- Compute a 2-WSPD for P
- Each box stores a representative point
- For each (A_i, B_i) approximate the closest pair:
  - Let s be the box size. Subdivide A_i and B_i until either:
    - Box diameter $\leq \varepsilon s$ — or —
    - The number of nonempty boxes $\geq c/\varepsilon$ (for some constant c)
  - $(p_i, q_i) \leftarrow$ closest pair of box representatives
- $G \leftarrow$ closest pairs. Return $\mathrm{MST}(G)$



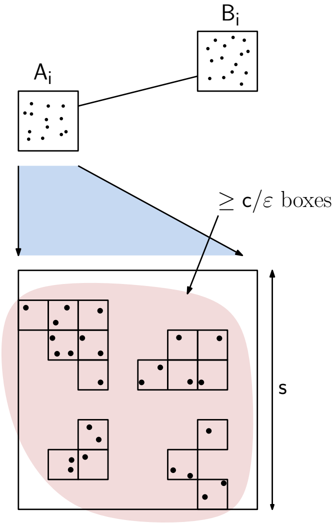
A Smarter/Sloppier Algorithm

- Compute a 2-WSPD for P
- Each box stores a representative point
- For each (A_i, B_i) approximate the closest pair:
  - Let s be the box size. Subdivide A_i and B_i until either:
    - Box diameter $\leq \varepsilon s$ — or —
    - The number of nonempty boxes $\geq c/\varepsilon$ (for some constant c)
  - $(p_i, q_i) \leftarrow$ closest pair of box representatives
- $G \leftarrow$ closest pairs. Return MST(G)



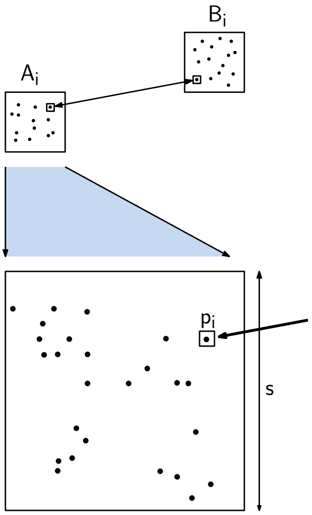
A Smarter/Sloppier Algorithm

- Compute a 2-WSPD for P
- Each box stores a representative point
- For each (A_i, B_i) approximate the closest pair:
  - Let s be the box size. Subdivide A_i and B_i until either:
    - Box diameter $\leq \varepsilon s$ — or —
    - The number of nonempty boxes $\geq c/\varepsilon$ (for some constant c)
  - $(p_i, q_i) \leftarrow$ closest pair of box representatives
- $G \leftarrow$ closest pairs. Return MST(G)



A Smarter/Sloppier Algorithm

- Compute a 2-WSPD for P
- Each box stores a representative point
- For each (A_i, B_i) approximate the closest pair:
  - Let s be the box size. Subdivide A_i and B_i until either:
    - Box diameter $\leq \varepsilon s$ — or —
    - The number of nonempty boxes $\geq c/\varepsilon$ (for some constant c)
  - $(p_i, q_i) \leftarrow$ closest pair of box representatives
- $G \leftarrow$ closest pairs. Return $\mathrm{MST}(G)$



Approximation Analysis (First Attempt)

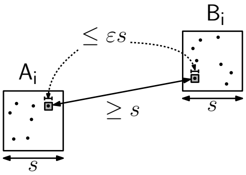
- Case 1: Box diameters $\leq \varepsilon s$:
  - Absolute error $\leq 2\varepsilon s \lesssim \varepsilon \cdot \mathrm{dist}(A_i, B_i)$
  - Relative error $\lesssim \varepsilon$
- Case 2: Number of nonempty boxes $\geq c/\varepsilon$:
  - Let δ be the diameters of the boxes
  - Absolute error $\lesssim \delta$
  - By Lower-Bound Lemma, weight of MST restricted to A_i or B_i is $\geq \delta(c/\varepsilon)/c = \delta/\varepsilon$
  - Relative error is $\lesssim \varepsilon$ (amortized over the box)

…hey, aren't you multiply charging?

Approximation Analysis (First Attempt)

- Case 1: Box diameters $\leq \varepsilon s$:
  - Absolute error $\leq 2\varepsilon s \lesssim \varepsilon \cdot \mathrm{dist}(A_i, B_i)$
  - Relative error $\lesssim \varepsilon$
- Case 2: Number of nonempty boxes $\geq c/\varepsilon$:
  - Let δ be the diameters of the boxes
  - Absolute error $\lesssim \delta$
  - By Lower-Bound Lemma, weight of MST restricted to A_i or B_i is $\geq \delta(c/\varepsilon)/c = \delta/\varepsilon$
  - Relative error is $\lesssim \varepsilon$ (amortized over the box)

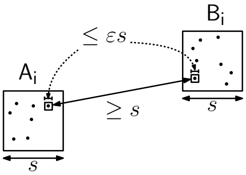
…hey, aren't you multiply charging?

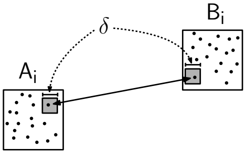


Approximation Analysis (First Attempt)

- Case 1: Box diameters $\leq \varepsilon s$:
  - Absolute error $\leq 2\varepsilon s \lesssim \varepsilon \cdot \text{dist}(A_i, B_i)$
  - Relative error $\lesssim \varepsilon$
- Case 2: Number of nonempty boxes $\geq c/\varepsilon$:
  - Let δ be the diameters of the boxes
  - Absolute error $\lesssim \delta$
  - By Lower-Bound Lemma, weight of MST restricted to A_i or B_i is $\geq \delta(c/\varepsilon)/c = \delta/\varepsilon$
  - Relative error is $\lesssim \varepsilon$ (amortized over the box)

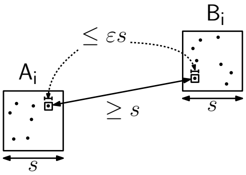
...hey, aren't you multiply charging?



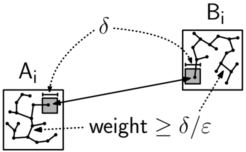


Approximation Analysis (First Attempt)

- Case 1: Box diameters $\leq \varepsilon s$:
  - Absolute error $\leq 2\varepsilon s \lesssim \varepsilon \cdot \mathrm{dist}(A_i, B_i)$
  - Relative error $\lesssim \varepsilon$



- Case 2: Number of nonempty boxes $\geq c/\varepsilon$:
  - Let δ be the diameters of the boxes
  - Absolute error $\lesssim \delta$
  - By Lower-Bound Lemma, weight of MST restricted to A_i or B_i is $\geq \delta(c/\varepsilon)/c = \delta/\varepsilon$
  - Relative error is $\lesssim \varepsilon$ (amortized over the box)

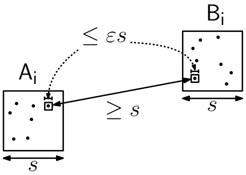


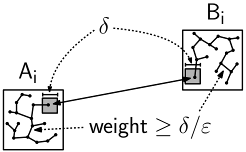
...hey, aren't you multiply charging?

Approximation Analysis (First Attempt)

- Case 1: Box diameters $\leq \varepsilon s$:
  - Absolute error $\leq 2\varepsilon s \lesssim \varepsilon \cdot \text{dist}(A_i, B_i)$
  - Relative error $\lesssim \varepsilon$
- Case 2: Number of nonempty boxes $\geq c/\varepsilon$:
  - Let δ be the diameters of the boxes
  - Absolute error $\lesssim \delta$
  - By Lower-Bound Lemma, weight of MST restricted to A_i or B_i is $\geq \delta(c/\varepsilon)/c = \delta/\varepsilon$
  - Relative error is $\lesssim \varepsilon$ (amortized over the box)

...hey, aren't you multiply charging?

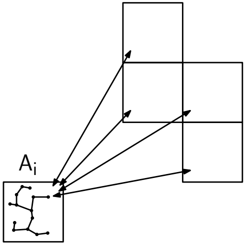




Approximation Analysis (Finer Points)

We charge the same MST edge multiple times:

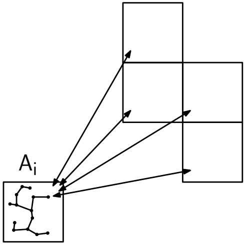
- Multiple WSPs share the same quadtree box
  — each box is in $O\left(\sqrt{d}\right)^d = O(1)$ WSPs
  $\rightarrow$ increase c by this constant
- Multiple tree levels charge the same edge
  $\rightarrow$ further increase c by tree height
  — $\times\ O\left(\log \frac{n}{\varepsilon}\right)$ [Arora (1998)]
- Reducing the log factor
  — A more refined analysis reduces the log factor to $O\left(\log \frac{1}{\varepsilon}\right)$

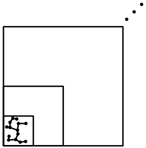


Approximation Analysis (Finer Points)

We charge the same MST edge multiple times:

- Multiple WSPs share the same quadtree box
  — each box is in $O(\sqrt{d})^d = O(1)$ WSPs
  $\rightarrow$ increase c by this constant
- Multiple tree levels charge the same edge
  $\rightarrow$ further increase c by tree height
  — $\times\ O\left(\log \frac{n}{\varepsilon}\right)$ [Arora (1998)]
- Reducing the log factor
  — A more refined analysis reduces the log factor to $O\left(\log \frac{1}{\varepsilon}\right)$

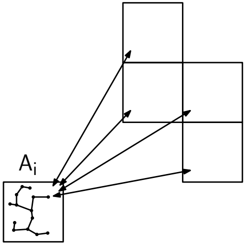


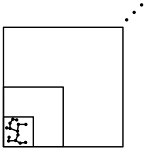


Approximation Analysis (Finer Points)

We charge the same MST edge multiple times:

- Multiple WSPs share the same quadtree box
  — each box is in $O(\sqrt{d})^d = O(1)$ WSPs
  $\rightarrow$ increase c by this constant
- Multiple tree levels charge the same edge
  $\rightarrow$ further increase c by tree height
  — $\times\ O\left(\log \frac{n}{\varepsilon}\right)$ [Arora (1998)]
- Reducing the log factor
  — A more refined analysis reduces the log factor to $O\left(\log \frac{1}{\varepsilon}\right)$





Execution Time

- Build the quadtree and WSPD: $O(n \log n)$
- Find the approximate closest pair for each WSP:
  - $O(n)$ WSPs
  - $O(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})$ boxes per WSP
  - $O((\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})^2)$ representative pairs per WSP
- Compute the MST of G: $O(n \log n)$
- Total time: $O\left(n \log n + (\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})^2 n\right) = \widetilde{O}(n/\varepsilon^2)$

Execution Time

- Build the quadtree and WSPD: $O(n \log n)$
- Find the approximate closest pair for each WSP:
  - $O(n)$ WSPs
  - $O(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})$ boxes per WSP
  - $O((\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})^2)$ representative pairs per WSP
- Compute the MST of G: $O(n \log n)$
- Total time: $O\left(n \log n + (\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})^2 n\right) = \widetilde{O}(n/\varepsilon^2)$

Execution Time

- Build the quadtree and WSPD: $O(n \log n)$
- Find the approximate closest pair for each WSP:
  - $O(n)$ WSPs
  - $O(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})$ boxes per WSP
  - $O((\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})^2)$ representative pairs per WSP
- Compute the MST of G: $O(n \log n)$
- Total time: $O\left(n \log n + (\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})^2 n\right) = \widetilde{O}(n/\varepsilon^2)$

Execution Time

- Build the quadtree and WSPD: $O(n \log n)$
- Find the approximate closest pair for each WSP:
  - $O(n)$ WSPs
  - $O(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})$ boxes per WSP
  - $O((\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})^2)$ representative pairs per WSP
- Compute the MST of G: $O(n \log n)$
- Total time: $O\left(n \log n + (\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})^2 n\right) = \widetilde{O}(n/\varepsilon^2)$

Execution Time

- Build the quadtree and WSPD: $O(n \log n)$
- Find the approximate closest pair for each WSP:
  - $O(n)$ WSPs
  - $O(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})$ boxes per WSP
  - $O((\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})^2)$ representative pairs per WSP
- Compute the MST of G: $O(n \log n)$
- Total time: $O\left(n \log n + (\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})^2 n\right) = \widetilde{O}(n/\varepsilon^2)$

Execution Time

- Build the quadtree and WSPD: $O(n \log n)$
- Find the approximate closest pair for each WSP:
  - $O(n)$ WSPs
  - $O(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})$ boxes per WSP
  - $O((\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})^2)$ representative pairs per WSP
- Compute the MST of G: $O(n \log n)$
- Total time: $O\left(n \log n + (\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})^2 n\right) = \widetilde{O}(n/\varepsilon^2)$

Talk Overview

- Preliminaries: WSPDs, MSTs, and Fast Lower Bounds
- Theory: ε-approximate MSTs in $\widetilde{O}(n/\varepsilon^2)$ time
- Practice: A more practical approach and implementation

Talk Overview

- Preliminaries: WSPDs, MSTs, and Fast Lower Bounds
- Theory: ε-approximate MSTs in $\widetilde{O}(n/\varepsilon^2)$ time
- Practice: A more practical approach and implementation

Practical Solutions

Practical solutions: (for exact MSTs)

- Bentley and Friedman (1978): kd-trees + Prim
- Narasimhan and Zachariasen (2001): WSPDs + Kruskal: GeoMST, GeoMST2
- Chatterjee, Connor, and Kumar (2010): WSPDs + Kruskal: GeoFilterKruskal
- March, Ram, Gray (2010): WSPD + Borůvka
- Here: Adding approximation (Joint with Phong Dinh and Jerry Tan)

GeoMST (Narasimhan and Zachariasen (2001))

GeoMST(P):

- $\Psi \leftarrow$ a 2-WSPD for P
- For each $(A_i, B_i) \in \Psi$, compute the bichromatic closest pair, $\mathrm{BCP}(A_i, B_i)$
- Run Kruskal on the resulting edges

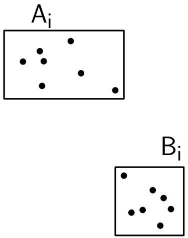
$\mathrm{BCP}(A_i, B_i)$:

- If $|A_i| = |B_i| = 1$ return this pair
- Else, split the larger cell (kd-tree children) A_i', A_i''
- Compute $\delta' \leftarrow \mathrm{BCP}(A_i', B_i)$ (closer child)
- If $(\text{cell-dist}(A_i'', B_i) < \delta'/(1+\epsilon))$ $\delta'' \leftarrow \mathrm{BCP}(A_i', B_i)$
- return $\min(\delta', \delta'')$

GeoMST (Narasimhan and Zachariasen (2001))

GeoMST(P):

- $\Psi \leftarrow$ a 2-WSPD for P
- For each $(A_i, B_i) \in \Psi$, compute the bichromatic closest pair, BCP(A_i, B_i)
- Run Kruskal on the resulting edges



BCP(A_i, B_i):

- If $|A_i| = |B_i| = 1$ return this pair
- Else, split the larger cell (kd-tree children) A_i', A_i''
- Compute $\delta' \leftarrow$ BCP(A_i', B_i) (closer child)
- If (cell-dist$(A_i'', B_i) < \delta'/(1+\varepsilon)$) $\delta'' \leftarrow$ BCP(A_i', B_i)
- return $\min(\delta', \delta'')$

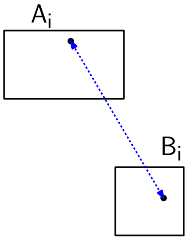
GeoMST (Narasimhan and Zachariasen (2001))

GeoMST(P):

- $\Psi \leftarrow$ a 2-WSPD for P
- For each $(A_i, B_i) \in \Psi$, compute the bichromatic closest pair, BCP(A_i, B_i)
- Run Kruskal on the resulting edges

BCP(A_i, B_i):

- If $|A_i| = |B_i| = 1$ return this pair
- Else, split the larger cell (kd-tree children) A_i', A_i''
- Compute $\delta' \leftarrow$ BCP(A_i', B_i) (closer child)
- If (cell-dist$(A_i'', B_i) < \delta' / (1+\varepsilon)$) $\delta'' \leftarrow$ BCP(A_i', B_i)
- return min(δ', δ'')



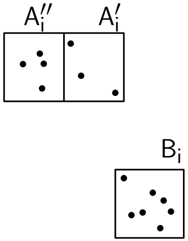
GeoMST (Narasimhan and Zachariasen (2001))

GeoMST(P):

- $\Psi \leftarrow$ a 2-WSPD for P
- For each $(A_i, B_i) \in \Psi$, compute the bichromatic closest pair, BCP(A_i, B_i)
- Run Kruskal on the resulting edges

BCP(A_i, B_i):

- If $|A_i| = |B_i| = 1$ return this pair
- Else, split the larger cell (kd-tree children) A_i', A_i''
- Compute $\delta' \leftarrow$ BCP(A_i', B_i) (closer child)
- If (cell-dist$(A_i'', B_i) < \delta'/(1+\varepsilon)$) $\delta'' \leftarrow$ BCP(A_i', B_i)
- return $\min(\delta', \delta'')$



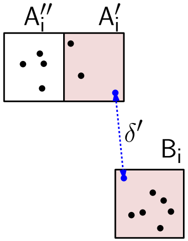
GeoMST (Narasimhan and Zachariasen (2001))

GeoMST(P):

- $\Psi \leftarrow$ a 2-WSPD for P
- For each $(A_i, B_i) \in \Psi$, compute the bichromatic closest pair, $\mathrm{BCP}(A_i, B_i)$
- Run Kruskal on the resulting edges

BCP(A_i, B_i):

- If $|A_i| = |B_i| = 1$ return this pair
- Else, split the larger cell (kd-tree children) A_i', A_i''
- Compute $\delta' \leftarrow \mathrm{BCP}(A_i', B_i)$ (closer child)
- If (cell-dist$(A_i'', B_i) < \delta'/(1+\varepsilon)$) $\delta'' \leftarrow \mathrm{BCP}(A_i', B_i)$
- return $\min(\delta', \delta'')$



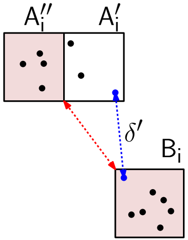
GeoMST (Narasimhan and Zachariasen (2001))

GeoMST(P):

- $\Psi \leftarrow$ a 2-WSPD for P
- For each $(A_i, B_i) \in \Psi$, compute the bichromatic closest pair, $\text{BCP}(A_i, B_i)$
- Run Kruskal on the resulting edges

BCP(A_i, B_i):

- If $|A_i| = |B_i| = 1$ return this pair
- Else, split the larger cell (kd-tree children) A_i', A_i''
- Compute $\delta' \leftarrow \text{BCP}(A_i', B_i)$ (closer child)
- If $(\text{cell-dist}(A_i'', B_i) < \delta'/(1+\varepsilon))$ $\quad \delta'' \leftarrow \text{BCP}(A_i', B_i)$
- return $\min(\delta', \delta'')$



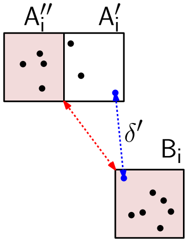
GeoMST (Narasimhan and Zachariasen (2001))

GeoMST(P):

- $\Psi \leftarrow$ a 2-WSPD for P
- For each $(A_i, B_i) \in \Psi$, compute the bichromatic closest pair, BCP(A_i, B_i)
- Run Kruskal on the resulting edges

BCP(A_i, B_i):

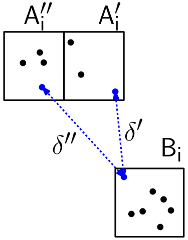
- If $|A_i| = |B_i| = 1$ return this pair
- Else, split the larger cell (kd-tree children) A_i', A_i''
- Compute $\delta' \leftarrow$ BCP(A_i', B_i) (closer child)
- If (cell-dist$(A_i'', B_i) < \delta'/(1+\varepsilon)$) $\delta'' \leftarrow$ BCP(A_i', B_i)
- return min(δ', δ'')



GeoMST (Narasimhan and Zachariasen (2001))

GeoMST(P):

- $\Psi \leftarrow$ a 2-WSPD for P
- For each $(A_i, B_i) \in \Psi$, compute the bichromatic closest pair, $\mathrm{BCP}(A_i, B_i)$
- Run Kruskal on the resulting edges

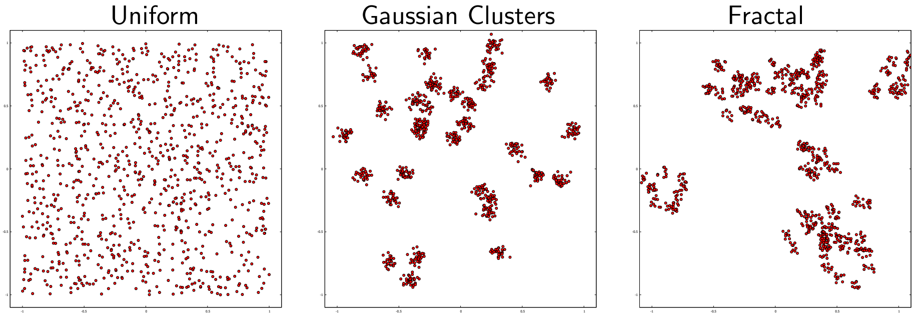


BCP(A_i, B_i):

- If $|A_i| = |B_i| = 1$ return this pair
- Else, split the larger cell (kd-tree children) A_i', A_i''
- Compute $\delta' \leftarrow \mathrm{BCP}(A_i', B_i)$ (closer child)
- If $(\text{cell-dist}(A_i'', B_i) < \delta'/(1+\varepsilon)) \quad \delta'' \leftarrow \mathrm{BCP}(A_i', B_i)$
- return $\min(\delta', \delta'')$

The Problem with WSPDs

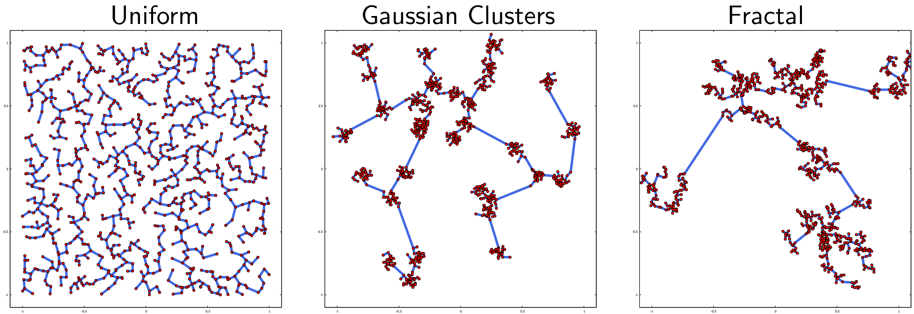
Practical Limitation — Too many WSPs in higher dimensions



Point set ($n = 1000$)

The Problem with WSPDs

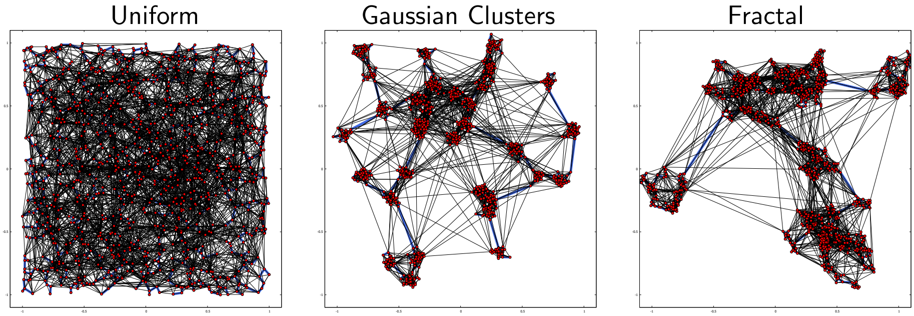
Practical Limitation — Too many WSPs in higher dimensions



MST ($n = 1000$)

The Problem with WSPDs

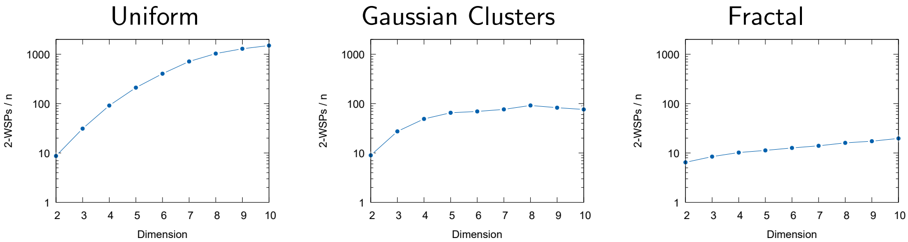
Practical Limitation — Too many WSPs in higher dimensions



WSPs joining reps ($n = 1000$)

The Problem with WSPDs

Practical Limitation — Too many WSPs in higher dimensions



Number of 2-WSPs vs. Dimension ($n = 1000$)

GeoMST2 (Narasimhan and Zachariasen (2001))

Want a WSP-based approach, but without the WSPD!

- Build WSPs only as needed
- Run Kruskal in parallel with WSP construction

GeoMST2: (Narasimhan and Zachariasen, 2001)

- Process WSPs in increasing order of distance
- Add edge to MST whenever it is safe: $\mathrm{BCP}(A_i, B_i) \leq$ distance between next WSP
- No postprocessing. The safe edges form the MST

GeoMST2 (Narasimhan and Zachariasen (2001))

Want a WSP-based approach, but without the WSPD!

- Build WSPs only as needed
- Run Kruskal in parallel with WSP construction

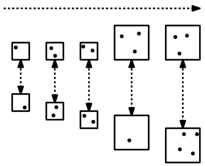
GeoMST2: (Narasimhan and Zachariasen, 2001)

- Process WSPs in increasing order of distance
- Add edge to MST whenever it is safe: $\mathrm{BCP}(A_i, B_i) \leq$ distance between next WSP
- No postprocessing. The safe edges form the MST

GeoMST2 (Narasimhan and Zachariasen (2001))

Want a WSP-based approach, but without the WSPD!

- Build WSPs only as needed
- Run Kruskal in parallel with WSP construction



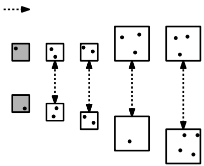
GeoMST2: (Narasimhan and Zachariasen, 2001)

- Process WSPs in increasing order of distance
- Add edge to MST whenever it is safe:
  $BCP(A_i, B_i) \leq$ distance between next WSP
- No postprocessing. The safe edges form the MST

GeoMST2 (Narasimhan and Zachariasen (2001))

Want a WSP-based approach, but without the WSPD!

- Build WSPs only as needed
- Run Kruskal in parallel with WSP construction



GeoMST2: (Narasimhan and Zachariasen, 2001)

- Process WSPs in increasing order of distance
- Add edge to MST whenever it is safe:
  $\mathrm{BCP}(A_i, B_i) \leq$ distance between next WSP
- No postprocessing. The safe edges form the MST

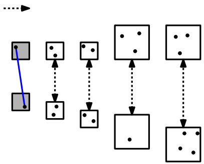
GeoMST2 (Narasimhan and Zachariasen (2001))

Want a WSP-based approach, but without the WSPD!

- Build WSPs only as needed
- Run Kruskal in parallel with WSP construction

GeoMST2: (Narasimhan and Zachariasen, 2001)

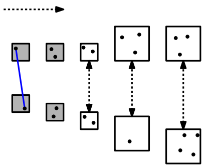
- Process WSPs in increasing order of distance
- Add edge to MST whenever it is safe:
  $\mathrm{BCP}(A_i, B_i) \leq$ distance between next WSP
- No postprocessing. The safe edges form the MST



GeoMST2 (Narasimhan and Zachariasen (2001))

Want a WSP-based approach, but without the WSPD!

- Build WSPs only as needed
- Run Kruskal in parallel with WSP construction



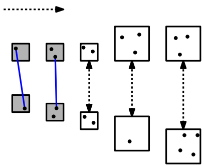
GeoMST2: (Narasimhan and Zachariasen, 2001)

- Process WSPs in increasing order of distance
- Add edge to MST whenever it is safe:
  $\mathrm{BCP}(A_i, B_i) \leq$ distance between next WSP
- No postprocessing. The safe edges form the MST

GeoMST2 (Narasimhan and Zachariasen (2001))

Want a WSP-based approach, but without the WSPD!

- Build WSPs only as needed
- Run Kruskal in parallel with WSP construction



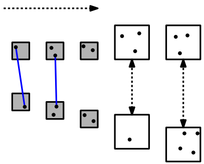
GeoMST2: (Narasimhan and Zachariasen, 2001)

- Process WSPs in increasing order of distance
- Add edge to MST whenever it is safe:
  BCP$(A_i, B_i) \leq$ distance between next WSP
- No postprocessing. The safe edges form the MST

GeoMST2 (Narasimhan and Zachariasen (2001))

Want a WSP-based approach, but without the WSPD!

- Build WSPs only as needed
- Run Kruskal in parallel with WSP construction



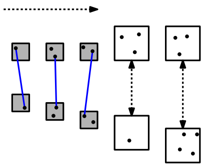
GeoMST2: (Narasimhan and Zachariasen, 2001)

- Process WSPs in increasing order of distance
- Add edge to MST whenever it is safe:
  $\text{BCP}(A_i, B_i) \leq$ distance between next WSP
- No postprocessing. The safe edges form the MST

GeoMST2 (Narasimhan and Zachariasen (2001))

Want a WSP-based approach, but without the WSPD!

- Build WSPs only as needed
- Run Kruskal in parallel with WSP construction



GeoMST2: (Narasimhan and Zachariasen, 2001)

- Process WSPs in increasing order of distance
- Add edge to MST whenever it is safe: $\mathrm{BCP}(A_i, B_i) \leq$ distance between next WSP
- No postprocessing. The safe edges form the MST

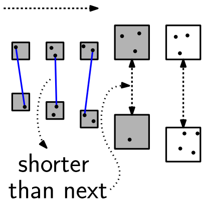
GeoMST2 (Narasimhan and Zachariasen (2001))

Want a WSP-based approach, but without the WSPD!

- Build WSPs only as needed
- Run Kruskal in parallel with WSP construction

GeoMST2: (Narasimhan and Zachariasen, 2001)

- Process WSPs in increasing order of distance
- Add edge to MST whenever it is safe: $\mathrm{BCP}(A_i, B_i) \leq$ distance between next WSP
- No postprocessing. The safe edges form the MST



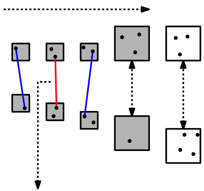
GeoMST2 (Narasimhan and Zachariasen (2001))

Want a WSP-based approach, but without the WSPD!

- Build WSPs only as needed
- Run Kruskal in parallel with WSP construction

GeoMST2: (Narasimhan and Zachariasen, 2001)

- Process WSPs in increasing order of distance
- Add edge to MST whenever it is safe:
  $\mathrm{BCP}(A_i, B_i) \leq$ distance between next WSP
- No postprocessing. The safe edges form the MST



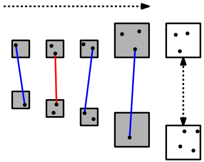
GeoMST2 (Narasimhan and Zachariasen (2001))

Want a WSP-based approach, but without the WSPD!

- Build WSPs only as needed
- Run Kruskal in parallel with WSP construction

GeoMST2: (Narasimhan and Zachariasen, 2001)

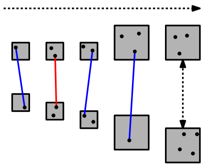
- Process WSPs in increasing order of distance
- Add edge to MST whenever it is safe:
  $\mathrm{BCP}(A_i, B_i) \leq$ distance between next WSP
- No postprocessing. The safe edges form the MST



GeoMST2 (Narasimhan and Zachariasen (2001))

Want a WSP-based approach, but without the WSPD!

- Build WSPs only as needed
- Run Kruskal in parallel with WSP construction



GeoMST2: (Narasimhan and Zachariasen, 2001)

- Process WSPs in increasing order of distance
- Add edge to MST whenever it is safe:
  $\mathrm{BCP}(A_i, B_i) \leq$ distance between next WSP
- No postprocessing. The safe edges form the MST

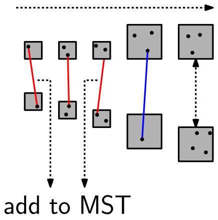
GeoMST2 (Narasimhan and Zachariasen (2001))

Want a WSP-based approach, but without the WSPD!

- Build WSPs only as needed
- Run Kruskal in parallel with WSP construction

GeoMST2: (Narasimhan and Zachariasen, 2001)

- Process WSPs in increasing order of distance
- Add edge to MST whenever it is safe:
  $\mathrm{BCP}(A_i, B_i) \leq$ distance between next WSP
- No postprocessing. The safe edges form the MST



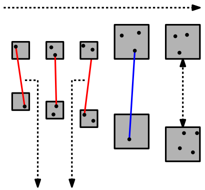
GeoMST2 (Narasimhan and Zachariasen (2001))

Want a WSP-based approach, but without the WSPD!

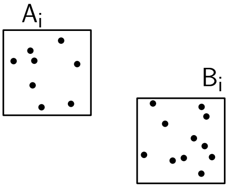
- Build WSPs only as needed
- Run Kruskal in parallel with WSP construction

GeoMST2: (Narasimhan and Zachariasen, 2001)

- Process WSPs in increasing order of distance
- Add edge to MST whenever it is safe:
  $\mathrm{BCP}(A_i, B_i) \leq$ distance between next WSP
- No postprocessing. The safe edges form the MST



Component-First



A further refinement to GeoMST2

Key modification: Process pairs (A_i, B_i) even if they are not well-separated

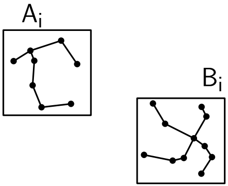
- If (all of A_i in one component and all of B_i in one component) then
  - If (same component) then discard the pair (A_i, B_i)
  - else compute BCP(A_i, B_i)

Component-First

A further refinement to GeoMST2

Key modification: Process pairs (A_i, B_i) even if they are not well-separated

- If (all of A_i in one component and all of B_i in one component) then
  - If (same component) then discard the pair (A_i, B_i)
  - else compute BCP(A_i, B_i)

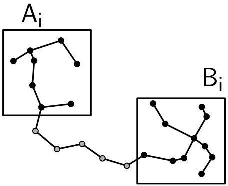


Component-First

A further refinement to GeoMST2

Key modification: Process pairs (A_i, B_i) even if they are not well-separated

- If (all of A_i in one component and all of B_i in one component) then
  - If (same component) then discard the pair (A_i, B_i)
  - else compute BCP(A_i, B_i)

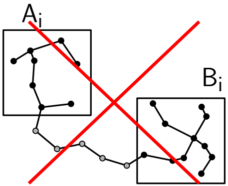


Component-First

A further refinement to GeoMST2

Key modification: Process pairs (A_i, B_i) even if they are not well-separated

- If (all of A_i in one component and all of B_i in one component) then
  - If (same component) then discard the pair (A_i, B_i)
  - else compute BCP(A_i, B_i)

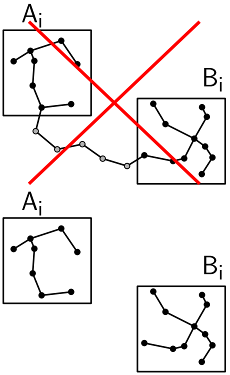


Component-First

A further refinement to GeoMST2

Key modification: Process pairs (A_i, B_i) even if they are not well-separated

- If (all of A_i in one component and all of B_i in one component) then
  - If (same component) then discard the pair (A_i, B_i)
  - else compute BCP(A_i, B_i)

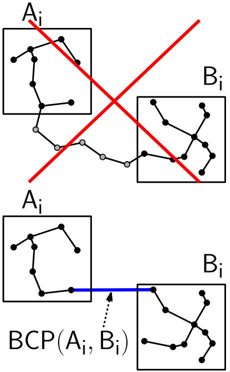


Component-First

A further refinement to GeoMST2

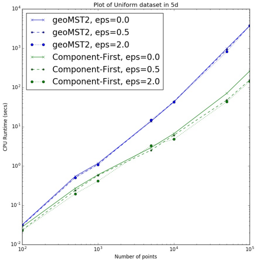
Key modification: Process pairs (A_i, B_i) even if they are not well-separated

- If (all of A_i in one component and all of B_i in one component) then
  - If (same component) then discard the pair (A_i, B_i)
  - else compute BCP(A_i, B_i)

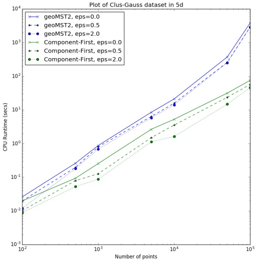


GeoMST2 and Component-First Performance

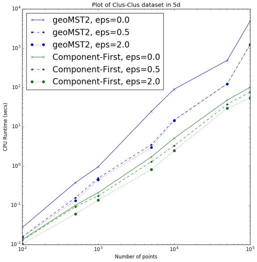
Uniform



Gaussian Clusters



Fractal



These algorithms are quite practical, and further improvements may be possible

Concluding Remarks

- Summary:
  - ε-approximate EMSTs in $\mathbb{R}^d$ in $\widetilde{O}(n/\varepsilon^2)$ time
  - Simple, deterministic algorithm (quadtrees, well-separated pairs)
  - Not really practical, but ideas can be applied to improve implementations
- Caveats:
  - EMST minimizes the bottleneck (max) edge cost — ours does not
  - Big-O hides factors that grow exponentially with dimension
- Further Work:
  - Approximate minimum bottleneck spanning tree in similar time?
  - Reduce extraneous factors $1/\varepsilon^2 \to 1/\varepsilon$? $\log^2(1/\varepsilon) \to O(1)$?
  - Further engineering of practical approaches

Concluding Remarks

- Summary:
  - ε-approximate EMSTs in $\mathbb{R}^d$ in $\widetilde{O}(n/\varepsilon^2)$ time
  - Simple, deterministic algorithm (quadtrees, well-separated pairs)
  - Not really practical, but ideas can be applied to improve implementations
- Caveats:
  - EMST minimizes the bottleneck (max) edge cost — ours does not
  - Big-O hides factors that grow exponentially with dimension
- Further Work:
  - Approximate minimum bottleneck spanning tree in similar time?
  - Reduce extraneous factors $1/\varepsilon^2 \to 1/\varepsilon$? $\log^2(1/\varepsilon) \to O(1)$?
  - Further engineering of practical approaches

Thank you for your attention!

Bibliography

- P. K. Agarwal, H. Edelsbrunner, O. Schwartzkopf, and E. Welzl, Euclidean minimum spanning trees and bichromatic closest pairs, Discr. and Comp. Geom., 6, 1991, 407–422
- S. Arora, Polynomial Time Approximation Schemes for Euclidean Traveling Salesman and Other Geometric Problems, J. of the ACM, 45, 1998, 753–782
- S. Arya and T. M. Chan, Better ε-dependencies for offline approximate nearest neighbor search, Euclidean minimum spanning trees, and ε-kernels, Proc. 30th SoCG, 2014, 416–425
- S. Arya, G. D. da Fonseca, and D. M. Mount, Near-optimal ε-kernel construction and related problems, Proc. 33rd Internat. Sympos. Comput. Geom., 2017, 10:1–15
- T. M. Chan, Applications of Chebyshev polynomials to low-dimensional computational geometry, Proc. 33rd Internat. Sympos. Comput. Geom., 2017, 26:1–15
- P. B. Callahan and S. R. Kosaraju, A Decomposition of Multidimensional Point Sets with Applications to k-Nearest-Neighbors and n-Body Potential Fields, J. of the ACM, 42, 1995, 67–90
- S. Chatterjee, M. Connor, and P. Kumar, Geometric Minimum Spanning Trees with GeoFilterKruskal, Proc. 9th Internat. Sympos. on Exper. Alg., 2010, 486-500
- B. Chazelle, R. Rubinfeld, and L. Trevisan, Approximating the minimum spanning tree weight in sublinear time, SIAM J. Comput., 2005, 34:1370–1379

Bibliography

- A. Czumaj, F. Ergün, L. Fortnow, A. Magen, I. Newman, R. Rubinfeld, and C. Sohler, Approximating the Weight of the Euclidean Minimum Spanning Tree in Sublinear Time, SIAM J. Comput., 35, 2005, 91–109
- A. Czumaj and C. Sohler, Estimating the Weight of Metric Minimum Spanning Trees in Sublinear Time, SIAM J. Comput., 39, 2009, 904–922
- G. Narasimhan and M. Zachariasen, Geometric Minimum Spanning Trees via Well-Separated Pair Decompositions, ACM J. Exper. Algorithms, 6, 2001
- A. C. Yao, On Constructing Minimum Spanning Trees in k-Dimensional Spaces and Related Problems, SIAM J. Comput., 11, 1982, 721–736