Constructing Interference-Minimal Networks

Marc Benkert, Karlsruhe University
Joachim Gudmundsson, Nicta Sydney
Herman Haverkort, TU Eindhoven
Alexander Wolff, Karlsruhe University

January, 2006

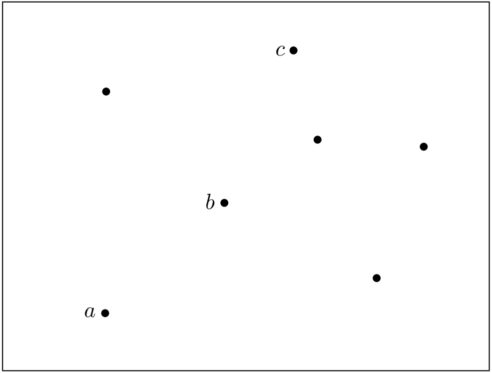
Overview

1. Introduction
   - Motivation
   - What is interference?
   - Properties
2. Exact interference
   - ComputeEdgeSet
   - FulfillsProperty
3. Estimated interference
   - Definition
   - ComputeEdgeSet
   - FulfillsProperty

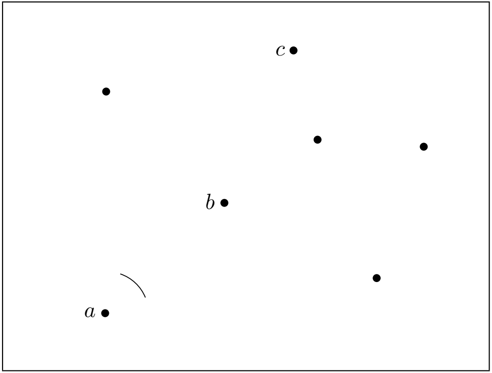
Overview

1. Introduction
   - Motivation
   - What is interference?
   - Properties
2. Exact interference
   - ComputeEdgeSet
   - FulfillsProperty
3. Estimated interference
   - Definition
   - ComputeEdgeSet
   - FulfillsProperty

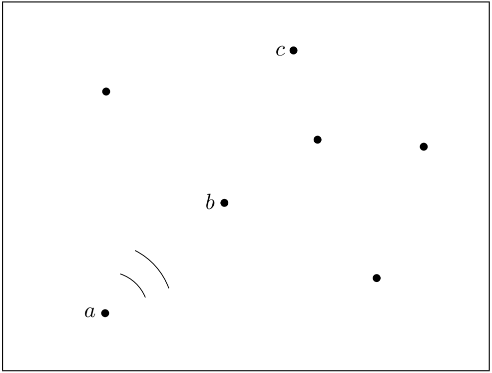
Setting up communication networks



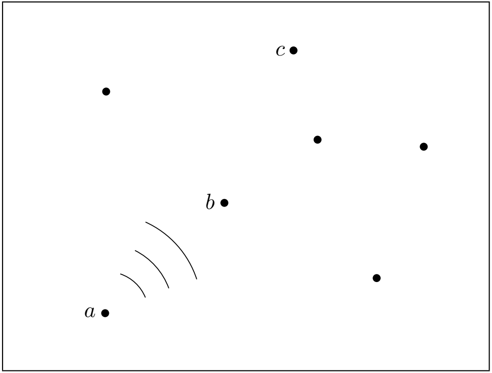
Setting up communication networks



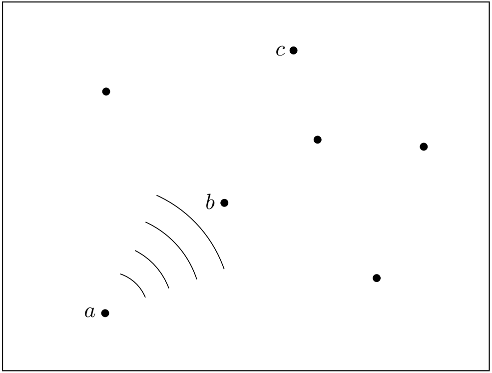
Setting up communication networks



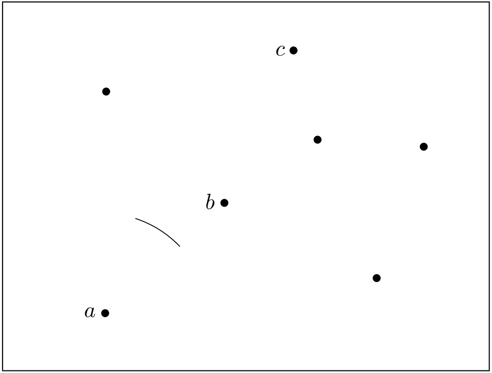
Setting up communication networks



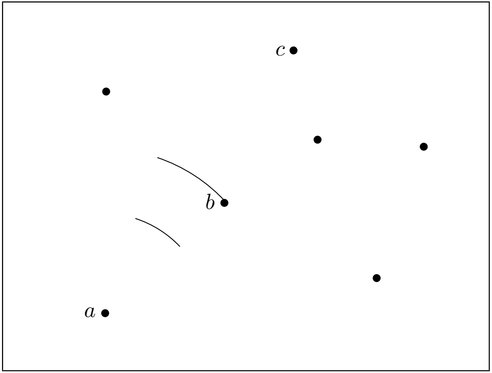
Setting up communication networks



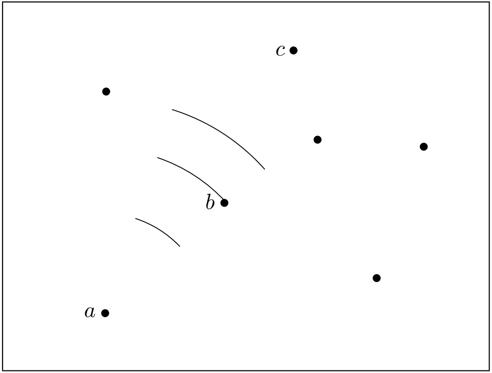
Setting up communication networks



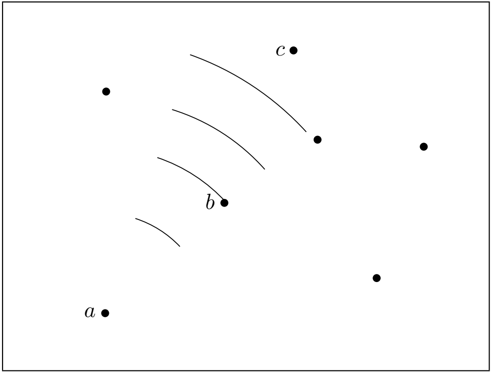
Setting up communication networks



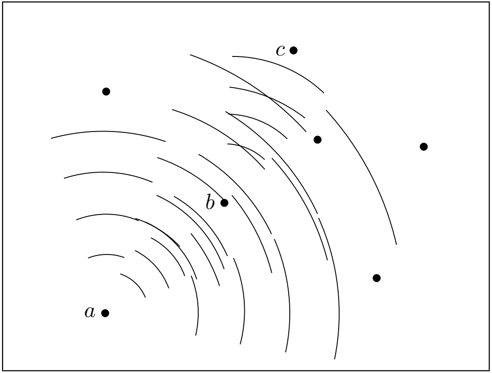
Setting up communication networks



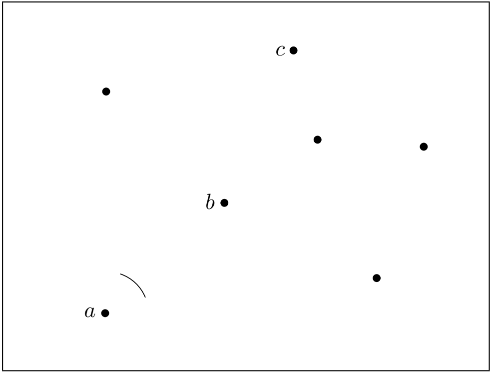
Setting up communication networks



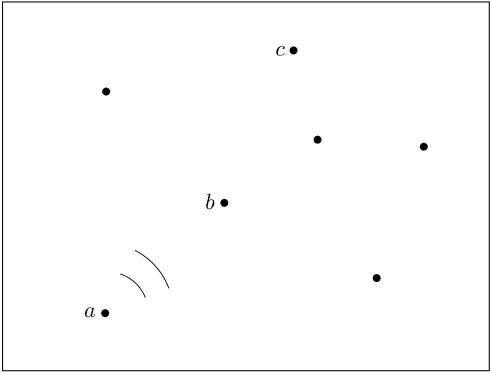
Setting up communication networks



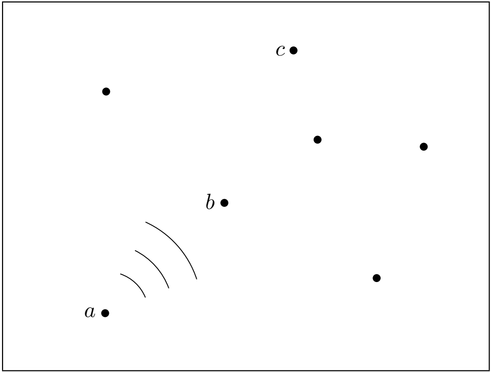
Setting up communication networks



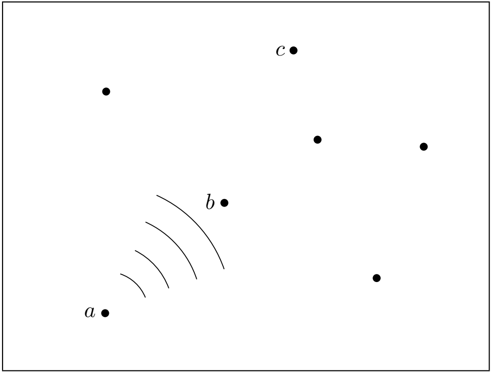
Setting up communication networks



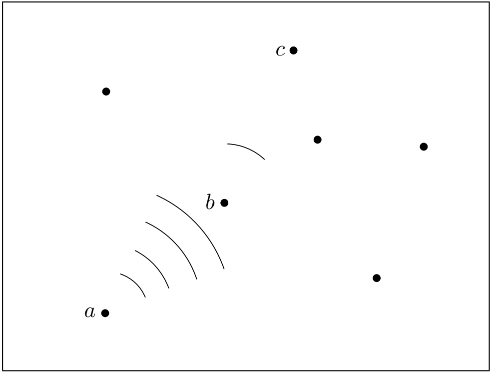
Setting up communication networks



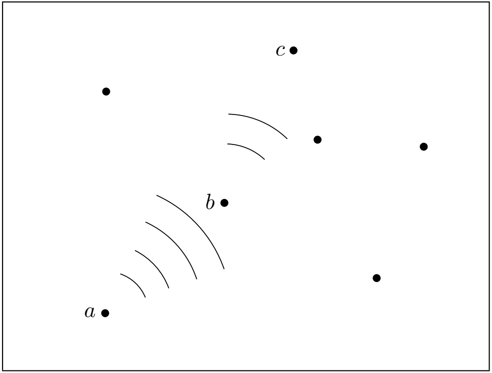
Setting up communication networks



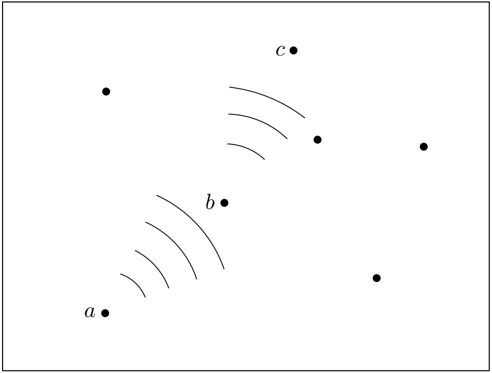
Setting up communication networks



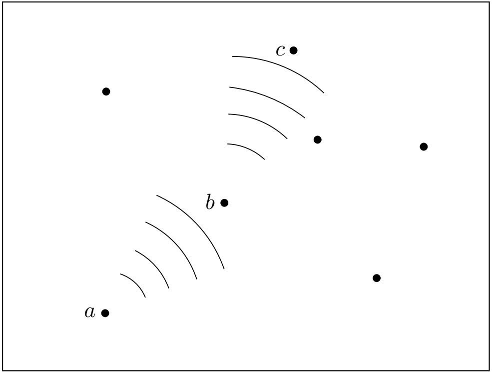
Setting up communication networks



Setting up communication networks

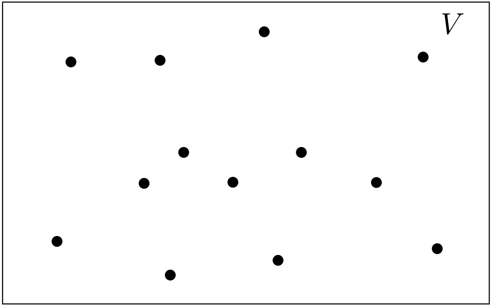


Setting up communication networks



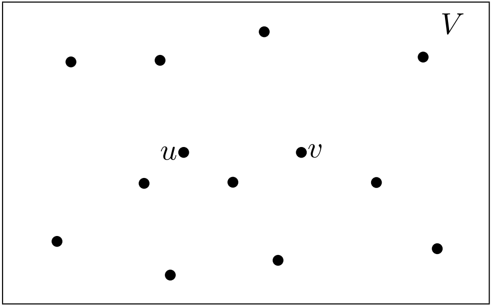
Definition of *interference*

[Burkhart et al., 2004]



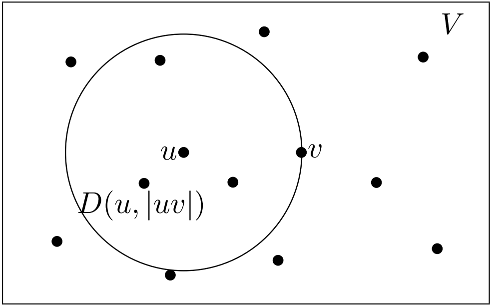
Definition of *interference*

[Burkhart et al., 2004]



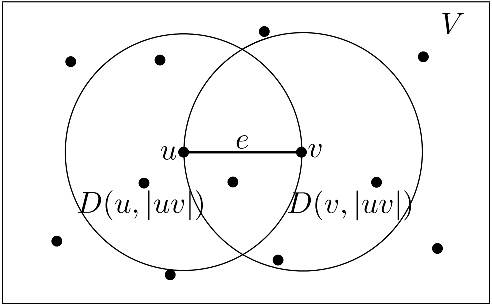
Definition of *interference*

[Burkhart et al., 2004]



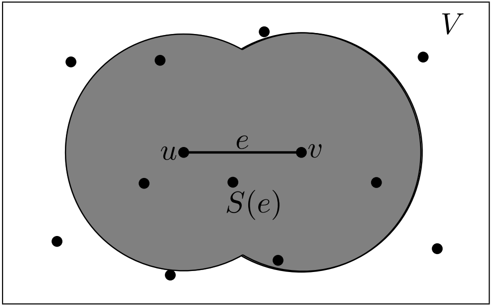
Definition of *interference*

[Burkhart et al., 2004]



Definition of *interference*

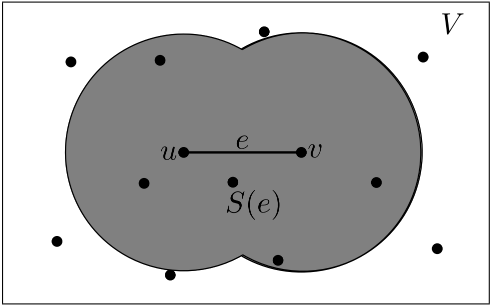
[Burkhart et al., 2004]



- $S(e) := D(u, |uv|) \cup D(v, |uv|)$.

Definition of *interference*

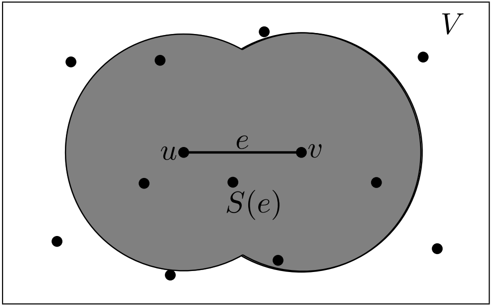
[Burkhart et al., 2004]



- $S(e) := D(u, |uv|) \cup D(v, |uv|)$.
- $\mathrm{Int}(e) := \big|V \cap S(e)\big| - 2$.

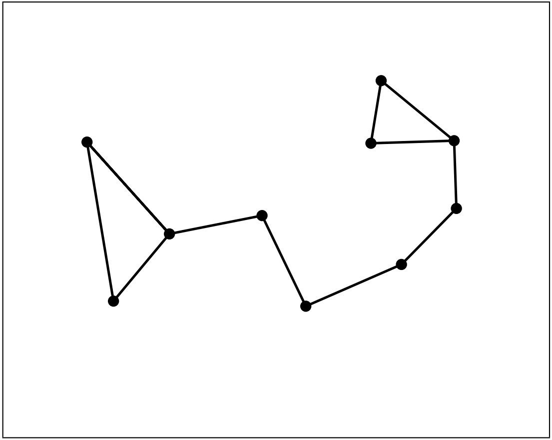
Definition of *interference*

[Burkhart et al., 2004]



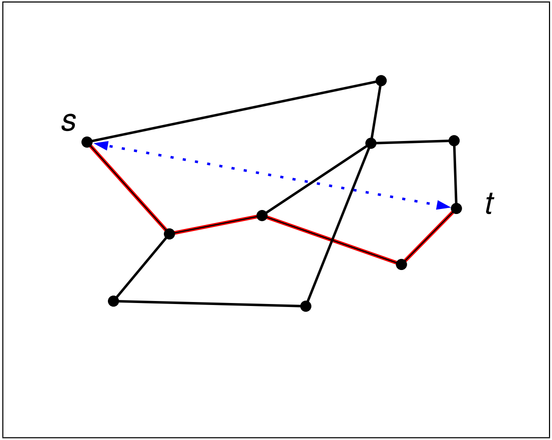
- $S(e) := D(u, |uv|) \cup D(v, |uv|)$.
- $\mathrm{Int}(e) := \big|V \cap S(e)\big| - 2$.
- $G = (V, E)$ communication graph.
  $\mathrm{Int}(G) := \max_{e \in E} \mathrm{Int}(e)$.

Properties of graphs that we construct



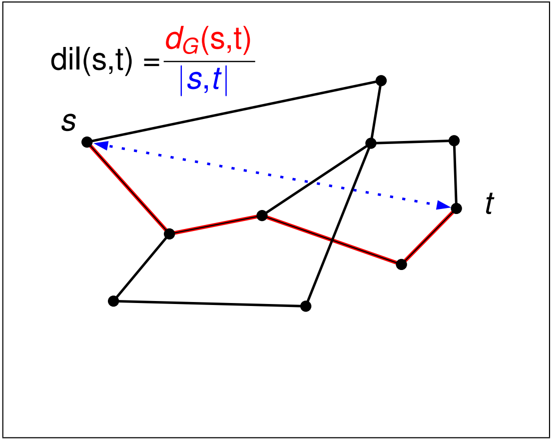
connectivity

Properties of graphs that we construct



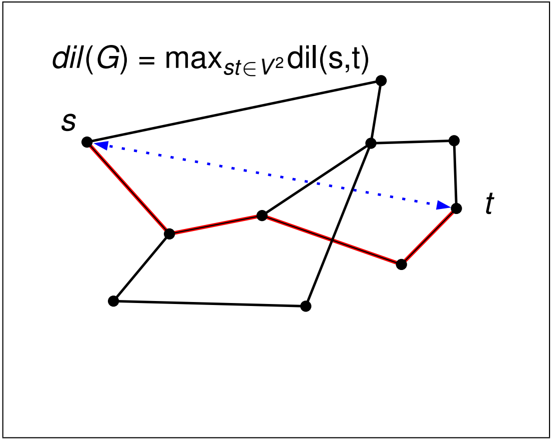
t-spanner

Properties of graphs that we construct



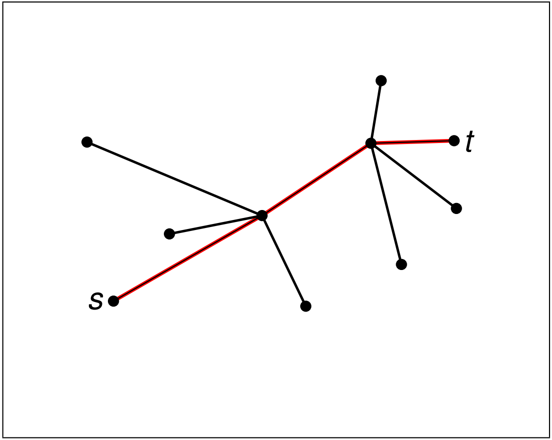
t-spanner

Properties of graphs that we construct



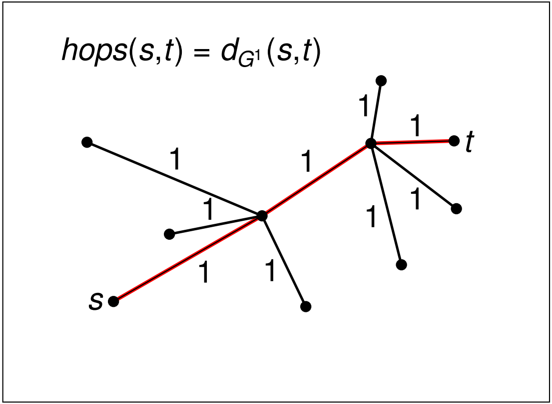
t-spanner

Properties of graphs that we construct



d-hop network

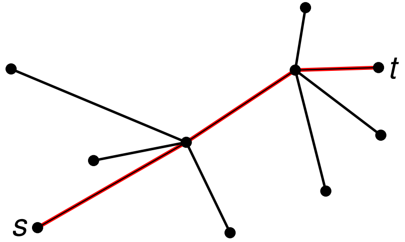
Properties of graphs that we construct



d-hop network

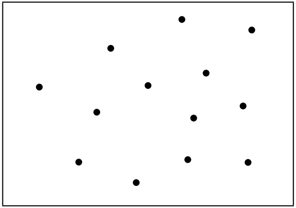
Properties of graphs that we construct

$$hops(G) = \max_{st \in V^2} hops(s,t)$$



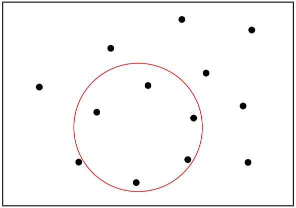
d-hop network

Previous results



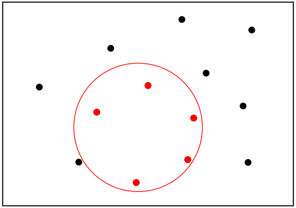
- Circular range searching [M93] $\Longrightarrow$
  interference values of all edges in $O(n^{\frac{9}{4}} \text{polylog } n)$ time.

Previous results



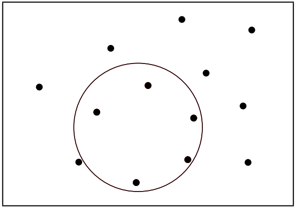
- Circular range searching [M93] $\Longrightarrow$ interference values of all edges in $O(n^{\frac{9}{4}} \text{polylog } n)$ time.

Previous results



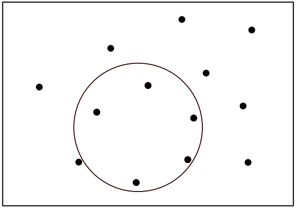
- Circular range searching [M93] $\Longrightarrow$ interference values of all edges in $O(n^{\frac{9}{4}} \text{polylog } n)$ time.

Previous results



- Circular range searching [M93] $\Longrightarrow$ interference values of all edges in $O(n^{\frac{9}{4}} \text{polylog}\, n)$ time.
- Interference-minimal t-spanner
  - Burkhart et al., 2004 $O(n^4)$
  - Moaveni-Nejad and Li, 2004 $O(n^3 \log n)$
  - Our observation: CRS + binary search $O(n^{\frac{9}{4}} \text{polylog}\, n)$ exp.

Previous results



- Circular range searching [M93] $\Longrightarrow$ interference values of all edges in $O(n^{\frac{9}{4}} \operatorname{polylog} n)$ time.
- Interference-minimal t-spanner
  - Burkhart et al., 2004 $O(n^4)$
  - Moaveni-Nejad and Li, 2004 $O(n^3 \log n)$
  - Our observation: CRS + binary search $O(n^{\frac{9}{4}} \operatorname{polylog} n)$ exp.
- ! All interference values were known a priori.

Our approach

- $E_\ell := \{e \mid \mathrm{Int}(e) \leq \ell\}$, $G_\ell = (V, E_\ell)$.

Our approach

- $E_\ell := \{e \mid \mathrm{Int}(e) \leq \ell\}$, $G_\ell = (V, E_\ell)$.
- k the smallest integer such that G_k has property $\mathcal{P}$.

Our approach

- $E_\ell := \{e \mid \mathrm{Int}(e) \leq \ell\}$, $G_\ell = (V, E_\ell)$.
- k the smallest integer such that G_k has property $\mathcal{P}$.
- Our aim: Faster algorithms for small values of k.

Our approach

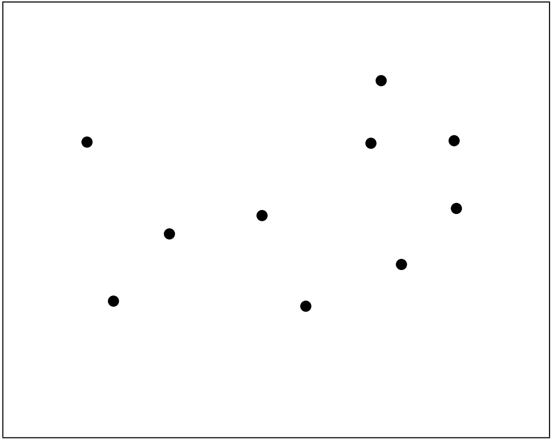
- $E_\ell := \{e \mid \mathrm{Int}(e) \leq \ell\}$, $G_\ell = (V, E_\ell)$.
- k the smallest integer such that G_k has property $\mathcal{P}$.
- Our aim: Faster algorithms for small values of k.
- Our runtime for the t-spanner: $O(\log k(nk^2 + n\log n))$ exp.

Our approach

- $E_\ell := \{e \mid \mathrm{Int}(e) \leq \ell\}$, $G_\ell = (V, E_\ell)$.
- k the smallest integer such that G_k has property $\mathcal{P}$.
- Our aim: Faster algorithms for small values of k.
- Our runtime for the t-spanner: $O(\log k(nk^2 + n\log n))$ exp.

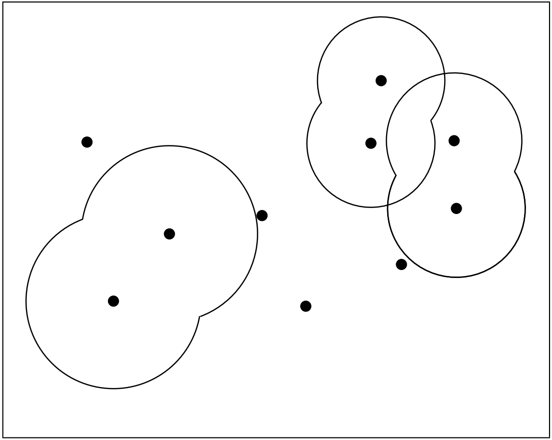
  $\Longrightarrow$ better than CRS $O(n^{\frac{9}{4}}\text{polylog } n)$ for $k \in O(n^{\frac{5}{8}})$.

Incremental approach (connectivity)



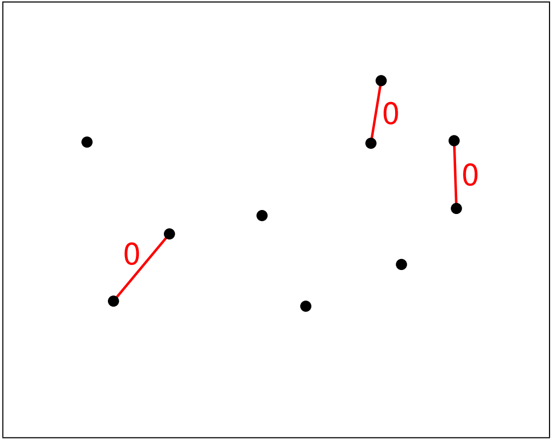
Input hosts

Incremental approach (connectivity)



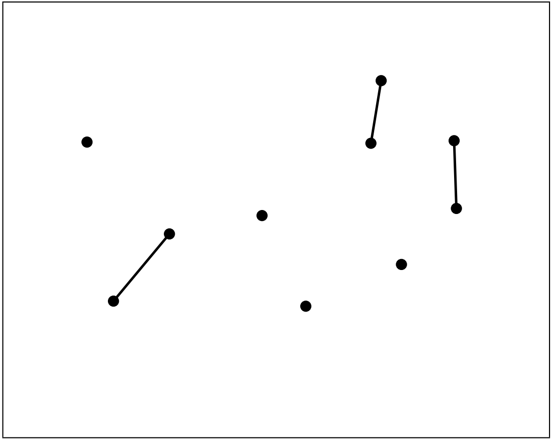
Spheres of edges e with $\mathrm{Int}(e) = 0$.

Incremental approach (connectivity)



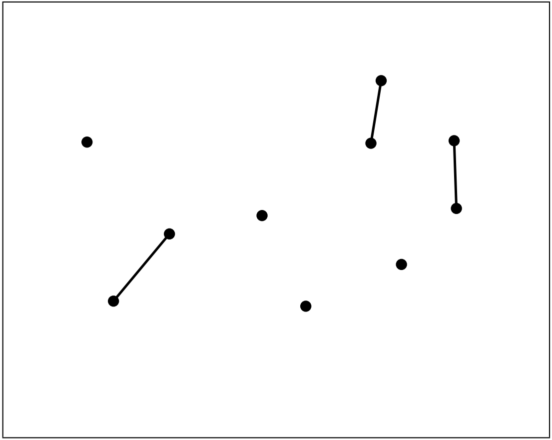
ComputeEdgeSet E_0

Incremental approach (connectivity)



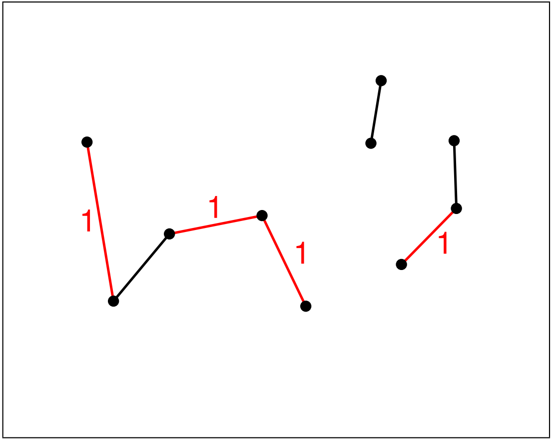
FulfillsProperty $\mathcal{P}$ (connected)?

Incremental approach (connectivity)



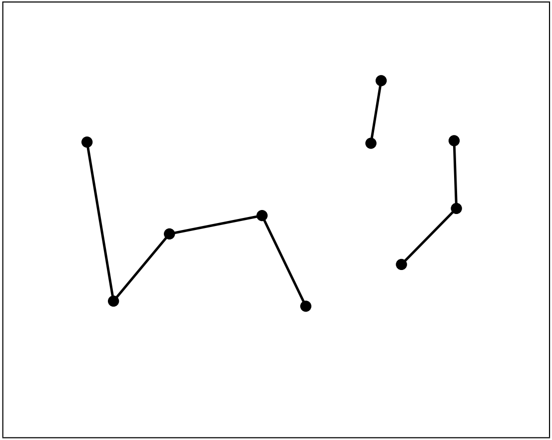
No.

Incremental approach (connectivity)



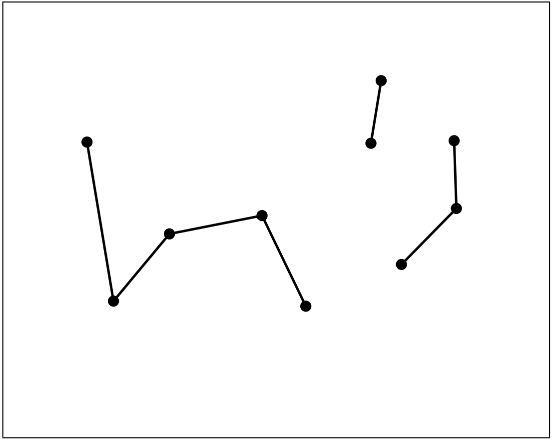
ComputeEdgeSet E_1

Incremental approach (connectivity)



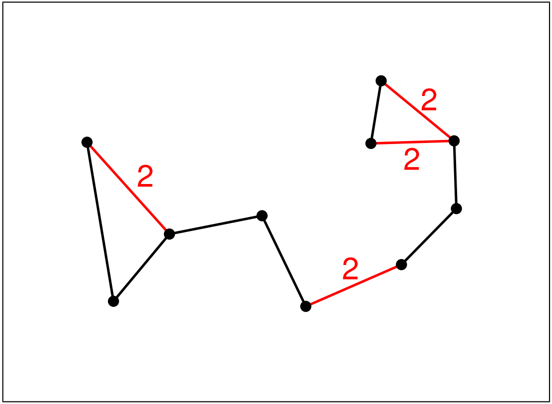
FulfillsProperty $\mathcal{P}$ (connected)?

Incremental approach (connectivity)



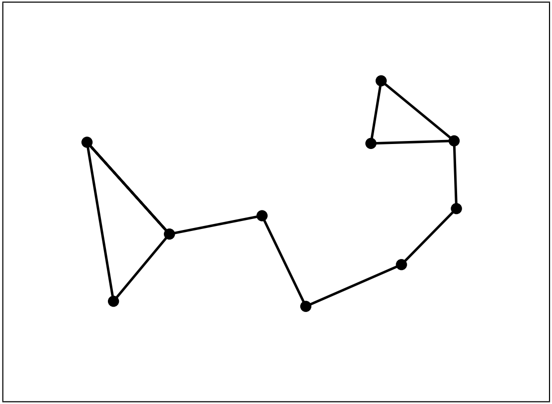
No.

Incremental approach (connectivity)



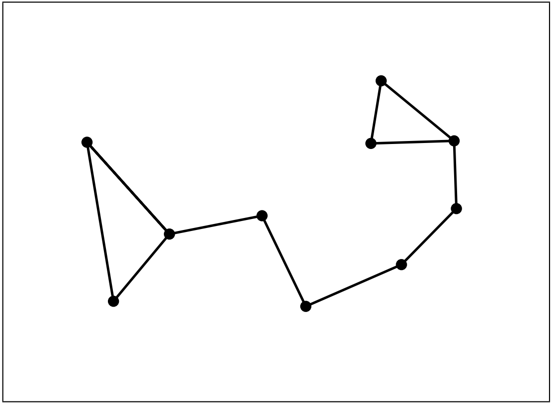
ComputeEdgeSet E_2

Incremental approach (connectivity)



FulfillsProperty $\mathcal{P}$ (connected)?

Incremental approach (connectivity)



Yes.

Overall proceeding

Given: Hosts V and a graph property $\mathcal{P}$, $n = |V|$.

Overall proceeding

Given: Hosts V and a graph property $\mathcal{P}$, $n = |V|$.
Find: k, the smallest integer such that G_k has property $\mathcal{P}$.

Overall proceeding

Given: Hosts V and a graph property $\mathcal{P}$, $n = |V|$.
Find: k, the smallest integer such that G_k has property $\mathcal{P}$.

Naive **Algorithm**

1. For $\ell = 0$ to n do
2. ComputeEdgeSet E_ℓ
3. FulfillsProperty G_ℓ

k=27, n=128: 0,1,2,3,...,27 $\rightarrow O(k)$ steps.

Overall proceeding

Given: Hosts V and a graph property $\mathcal{P}$, $n = |V|$.
Find: k, the smallest integer such that G_k has property $\mathcal{P}$.

Better **Algorithm**

1. Apply binary search to find k
2. ComputeEdgeSet E_ℓ
3. FulfillsProperty G_ℓ

k=27, n=128: 0,128,64,32,16,24,28,26,27 $\rightarrow O(\log n)$ steps.

Overall proceeding

Given: Hosts V and a graph property $\mathcal{P}$, $n = |V|$.
Find: k, the smallest integer such that G_k has property $\mathcal{P}$.

Best **Algorithm**

1. Apply exponential and binary search to find k
2. ComputeEdgeSet E_ℓ
3. FulfillsProperty G_ℓ

k=27, n=128: 0,1,2,4,8,16,32,24,28,26,27 $\rightarrow O(\log k)$ steps.

Overall proceeding

Given: Hosts V and a graph property $\mathcal{P}$, $n = |V|$.
Find: k, the smallest integer such that G_k has property $\mathcal{P}$.

Best **Algorithm**

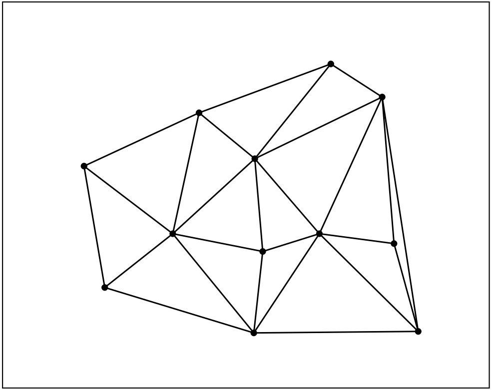
1. Apply exponential and binary search to find k
2. ComputeEdgeSet E_ℓ
3. FulfillsProperty G_ℓ

k=27, n=128: 0,1,2,4,8,16,32,24,28,26,27 $\rightarrow O(\log k)$ steps.

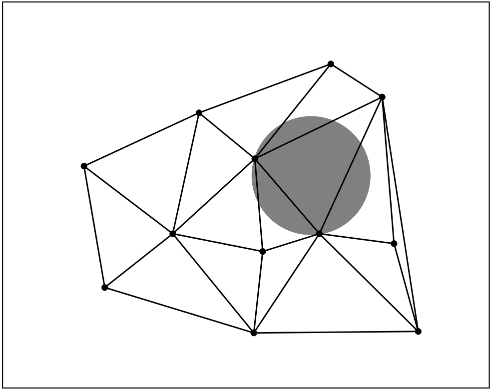
It remains to implement ComputeEdgeSet E_ℓ and FulfillsProperty G_ℓ !

Overview

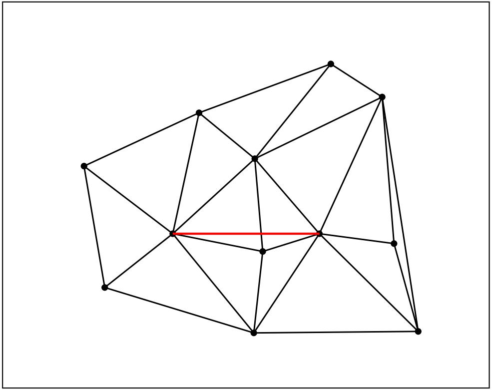
ComputeEdgeSet E_ℓ : Excursion Delaunay Edge.



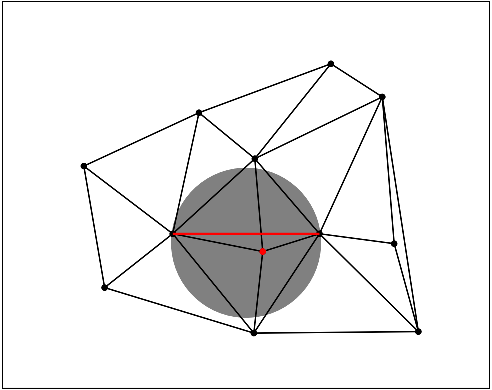
ComputeEdgeSet E_ℓ : Excursion Delaunay Edge.



ComputeEdgeSet E_ℓ : Excursion Delaunay Edge.



ComputeEdgeSet E_ℓ : Excursion Delaunay Edge.



ComputeEdgeSet E_ℓ.

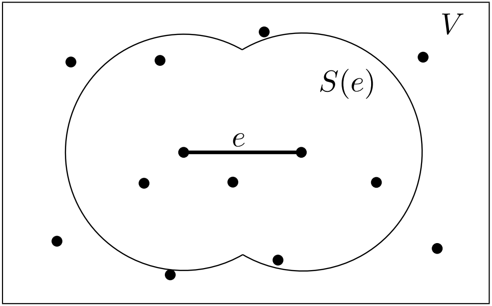
- Order-ℓ Delaunay graph DG_ℓ

ComputeEdgeSet E_ℓ.

- Order-ℓ Delaunay graph DG_ℓ
- Our edge set $E_\ell := \{e \mid \mathrm{Int}(e) \leq \ell\}$

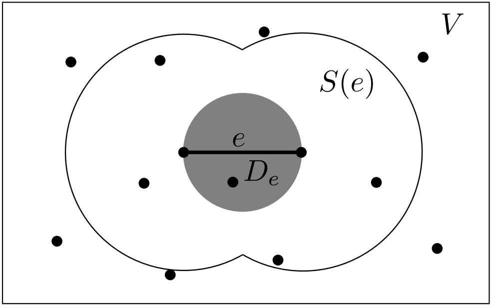
ComputeEdgeSet E_ℓ.

- Order-ℓ Delaunay graph DG_ℓ
- Our edge set $E_\ell := \{e \mid \mathrm{Int}(e) \leq \ell\}$



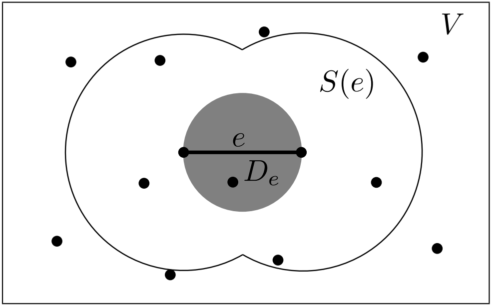
ComputeEdgeSet E_ℓ.

- Order-ℓ Delaunay graph DG_ℓ
- Our edge set $E_\ell := \{e \mid \mathrm{Int}(e) \leq \ell\}$



ComputeEdgeSet E_ℓ.

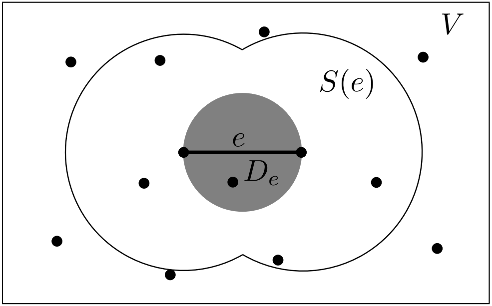
- Order-ℓ Delaunay graph DG_ℓ
- Our edge set $E_\ell := \{e \mid \mathrm{Int}(e) \leq \ell\}$



- $D_e \subset S(e) \implies E_\ell \subset DG_\ell$.

ComputeEdgeSet E_ℓ.

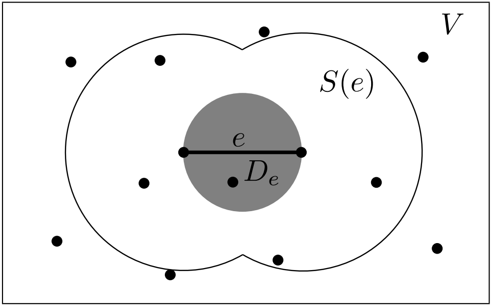
- Order-ℓ Delaunay graph DG_ℓ
- Our edge set $E_\ell := \{e \mid \mathrm{Int}(e) \leq \ell\}$



- $D_e \subset S(e) \implies E_\ell \subset DG_\ell$.
- DG_ℓ can be computed in $O(n\ell \log \ell + n \log n)$ exp. time. [GHK02]

ComputeEdgeSet E_ℓ.

- Order-ℓ Delaunay graph DG_ℓ
- Our edge set $E_\ell := \{e \mid \mathrm{Int}(e) \leq \ell\}$



- $D_e \subset S(e) \implies E_\ell \subset DG_\ell$.
- DG_ℓ can be computed in $O(n\ell \log \ell + n \log n)$ exp. time. [GHK02]
- $|DG_\ell| = O(n\ell)$. $\implies$ can compute E_ℓ in $O(n\ell^2 + n \log n)$.

FulfillsProperty G_ℓ

Is G_ℓ connected?

FulfillsProperty G_ℓ

Is G_ℓ connected?

- breadth first search
- check if all hosts were reached
- $O(n\ell)$ time

FulfillsProperty G_ℓ

Is G_ℓ connected?

- breadth first search
- check if all hosts were reached
- $O(n\ell)$ time

Is G_ℓ a t-spanner?

FulfillsProperty G_ℓ

Is G_ℓ connected?

- breadth first search
- check if all hosts were reached
- $O(n\ell)$ time

Is G_ℓ a t-spanner?

- all pairs shortest path computation, weight$(u, v) = |uv|$
- check for all pairs $d_{G_\ell}(u, v) \leq t|uv|$
- n times Dijkstra's shortest path algorithm $\rightarrow\ O(n^2(k + \log n))$

FulfillsProperty G_ℓ

Is G_ℓ connected?

- breadth first search
- check if all hosts were reached
- $O(n\ell)$ time

Is G_ℓ a t-spanner?

- all pairs shortest path computation, weight$(u, v) = |uv|$
- check for all pairs $d_{G_\ell}(u, v) \leq t|uv|$
- n times Dijkstra's shortest path algorithm $\rightarrow$ $O(n^2(k + \log n))$

Is G_ℓ a d-hop network?

FulfillsProperty G_ℓ

Is G_ℓ connected?

- breadth first search
- check if all hosts were reached
- $O(n\ell)$ time

Is G_ℓ a t-spanner?

- all pairs shortest path computation, weight$(u, v) = |uv|$
- check for all pairs $d_{G_\ell}(u, v) \leq t|uv|$
- n times Dijkstra's shortest path algorithm $\rightarrow\ O(n^2(k + \log n))$

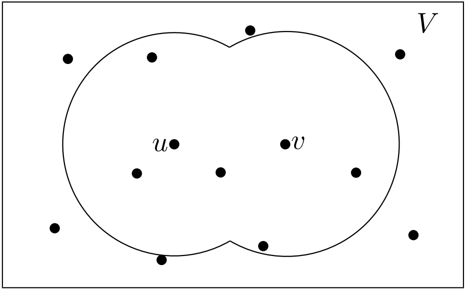
Is G_ℓ a d-hop network?

- all pairs shortest path computation, weight$(u, v) = 1$
- check for all pairs $d_{G_\ell}(u, v) \leq d$
- $O(n^2 \log n)$ exp. time [MT87]

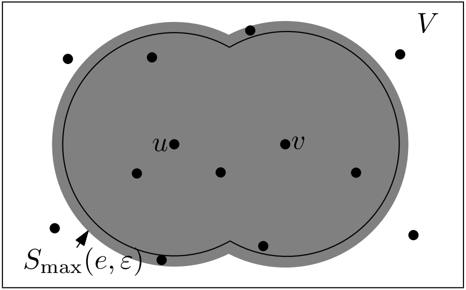
Overview

1. Introduction
   - Motivation
   - What is interference?
   - Properties
2. Exact interference
   - ComputeEdgeSet
   - FulfillsProperty
3. Estimated interference
   - Definition
   - ComputeEdgeSet
   - FulfillsProperty

Definition of *estimated* interference

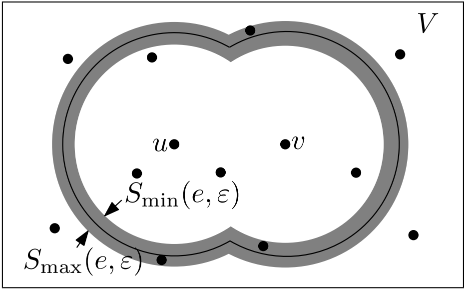


Definition of *estimated* interference



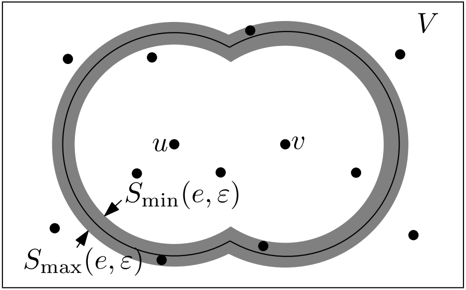
- $S_{\max}(e, \varepsilon) =$ "$(1 + \varepsilon) \cdot S(e)$"

Definition of *estimated* interference



- $S_{\max}(e,\varepsilon) =$ "$(1+\varepsilon)\cdot S(e)$"
- $S_{\min}(e,\varepsilon) =$ "$(1-\varepsilon)\cdot S(e)$"

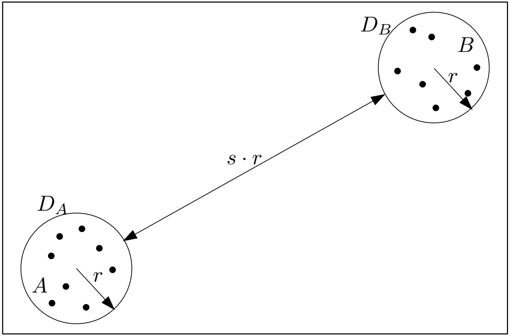
Definition of *estimated* interference



- $S_{\max}(e, \varepsilon) =$ "$(1 + \varepsilon) \cdot S(e)$"
- $S_{\min}(e, \varepsilon) =$ "$(1 - \varepsilon) \cdot S(e)$"
- $\tilde{\mathrm{Int}}$ is an ε-valid *interference estimation* of e if

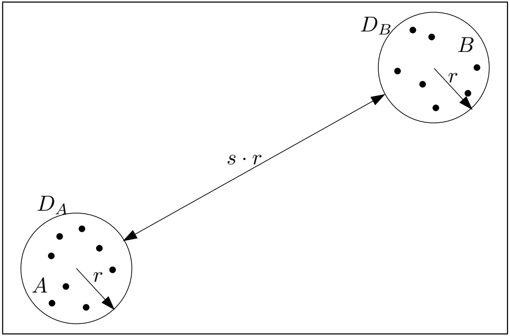
$$|S_{\max}(e, \varepsilon) \cap V| - 2 \;\geq\; \tilde{\mathrm{Int}} \geq |S_{\min}(e, \varepsilon) \cap V| - 2.$$

Well Separated Pair Decomposition [CK95]



A *well-separated* pair $\{A, B\}$ of point sets.

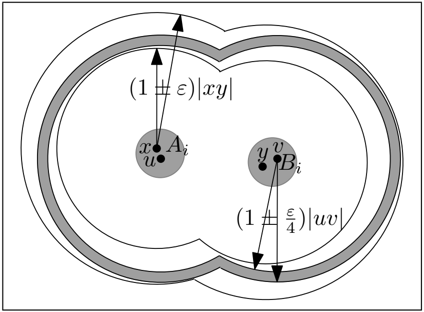
Well Separated Pair Decomposition [CK95]



A *well-separated* pair $\{A, B\}$ of point sets.

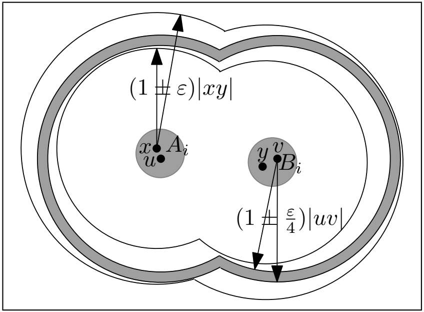
There is a WSPD $\{A_1, B_1\}, \ldots, \{A_k, B_k\}$ with $k = O(s^2 n)$ well-sep. pairs such that every point pair is separated by exactly one $\{A_i, B_i\}$.

ComputeEdgeSet E_ℓ (estimated!)



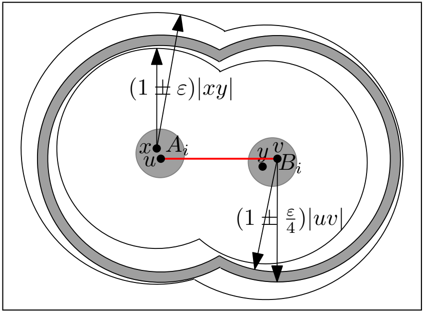
Goal: Faster computation than in the exact case. $O(n\ell^2)$

ComputeEdgeSet E_ℓ (estimated!)



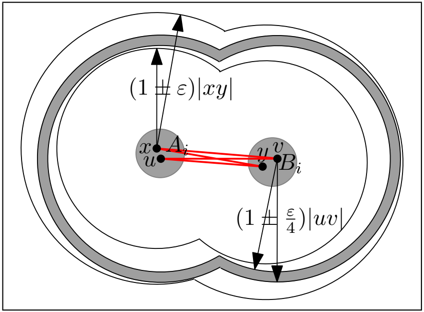
Approach: Preprocess ε-valid interference estimations for all edges.

ComputeEdgeSet E_ℓ (estimated!)



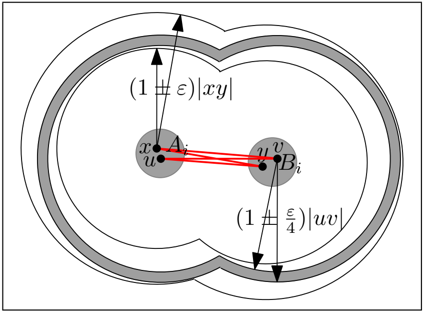
Idea: For each $\{A_i, B_i\}$ compute only one $\frac{\varepsilon}{4}$-valid interference estimations for an arbitrary edge $\{u, v\}$ ...

ComputeEdgeSet E_ℓ (estimated!)



Idea: For each $\{A_i, B_i\}$ compute only one $\frac{\varepsilon}{4}$-valid interference estimations for an arbitrary edge $\{u, v\}$ ...and assign its value to all edges that connect $\{A_i, B_i\}$.

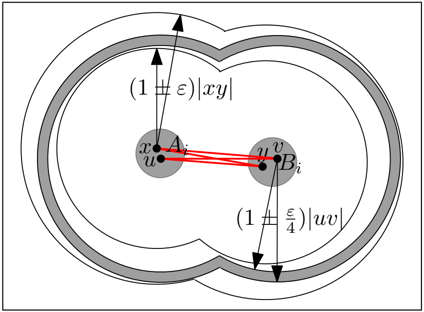
ComputeEdgeSet E_ℓ (estimated!)



Running time: $O(\frac{n}{\varepsilon^2} \cdot (\frac{1}{\varepsilon} + \log n))$ preprocessing.

approximate range counting [AM00]

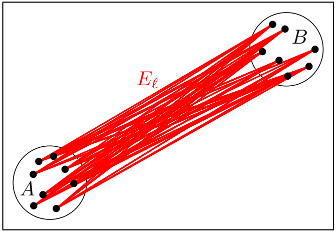
ComputeEdgeSet E_ℓ (estimated!)



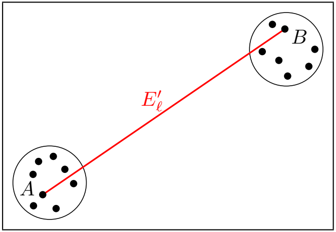
Running time: $O(\frac{n}{\varepsilon^2} \cdot (\frac{1}{\varepsilon} + \log n))$ preprocessing.

$\Longrightarrow O(\frac{n}{\varepsilon^2})$ to compute one E_ℓ exact: $O(n\ell^2)$.

FulfillsProperty G_ℓ (estimated)

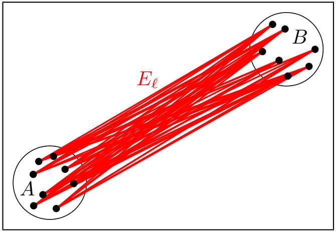


$$|E_\ell| = O(n^2)$$

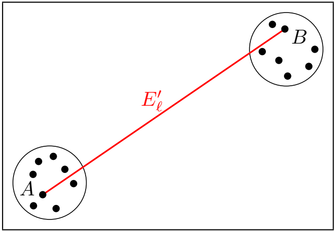


$$|E'_\ell| = O(\frac{n}{\varepsilon^2})$$

FulfillsProperty G_ℓ (estimated)



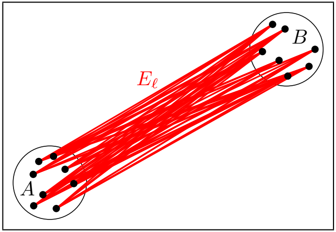
$|E_\ell| = O(n^2)$



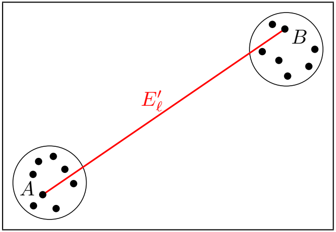
$|E'_\ell| = O(\frac{n}{\varepsilon^2})$

- Property testing (connectivity) in $G'_\ell = (V, E'_\ell)$.

FulfillsProperty G_ℓ (estimated)



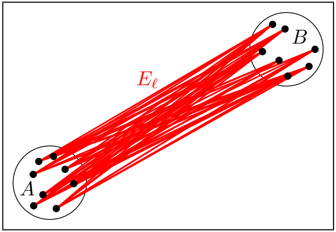
$|E_\ell| = O(n^2)$

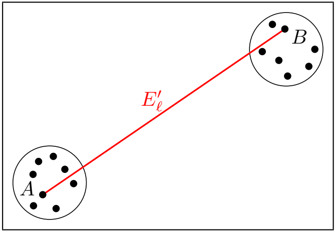


$|E'_\ell| = O(\frac{n}{\varepsilon^2})$

- Property testing (connectivity) in $G'_\ell = (V, E'_\ell)$.
- **Theorem:** G_ℓ connected $\Longleftrightarrow$ G'_ℓ connected.

FulfillsProperty G_ℓ (estimated)





$|E_\ell| = O(n^2)$ $\qquad$ $|E'_\ell| = O(\frac{n}{\varepsilon^2})$

- Property testing (connectivity) in $G'_\ell = (V, E'_\ell)$.
- **Theorem:** G_ℓ connected $\Longleftrightarrow$ G'_ℓ connected.
- $\Longrightarrow$ BFS in G'_ℓ runs in $O(\frac{n}{\varepsilon^2})$ time.

Tool boxes

Exact case

	connectivity	t-spanner	d-hop
CompEdgeSet E_ℓ	order-ℓ Delaunay triangulation		
FulfillsProperty G_ℓ	BFS	all pairs shortest path	

Estimated case

	connectivity	t-spanner	d-hop
CompEdgeSet E'_ℓ	Well-separated pair decomposition		
FulfillsProperty G_ℓ	BFS	approx.dilation	implicitBFS

Running times

Exact case

	connectivity	t-spanner	d-hop
CompEdgeSet E_ℓ	$O(n\ell^2 + n\log n)$		
FulfillsProperty G_ℓ	$O(n\ell)$	$O(n^2 \log n)$	

Estimated case

	connectivity	t-spanner	d-hop
CompEdgeSet E'_ℓ	preproc. $O(\frac{n}{\varepsilon^2}[\frac{1}{\varepsilon} + \log n])$, query $O(\frac{n}{\varepsilon^2})$		
FulfillsProperty G_ℓ	$O(\frac{n}{\varepsilon^2})$	$O(\frac{n^2}{\varepsilon^4} \log n)$	$O(\frac{n^2}{\varepsilon^2})$

Challenges

- Find better algorithms for $k \in \Omega(n^{\frac{5}{8}})$, i.e. faster than $O(n^{\frac{9}{4}} \text{polylog } n)$. (CRS)
- Find other desirable properties for communication graphs.