On the Locality of Distributed Sparse Spanner Constructions

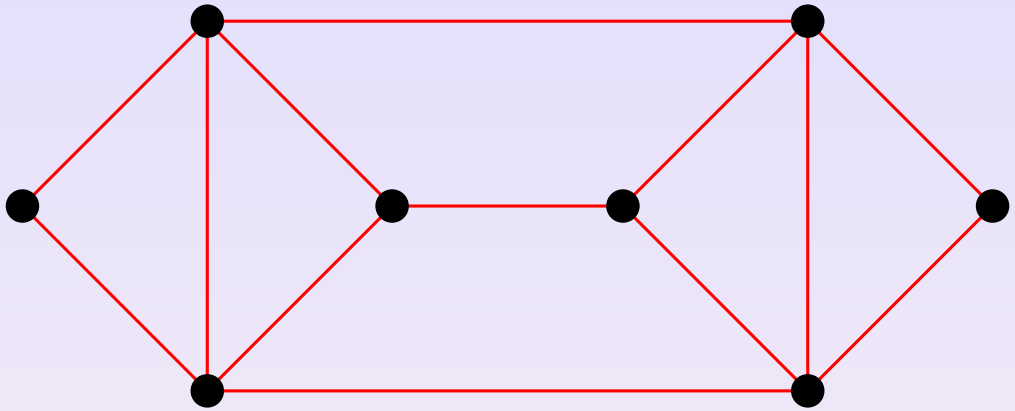
B. Derbel, **C. Gavoille**, D. Peleg, L. Viennot

University of Lille
University of Bordeaux
Weizmann Institute
INRIA, Paris

PODC 2008 – Toronto

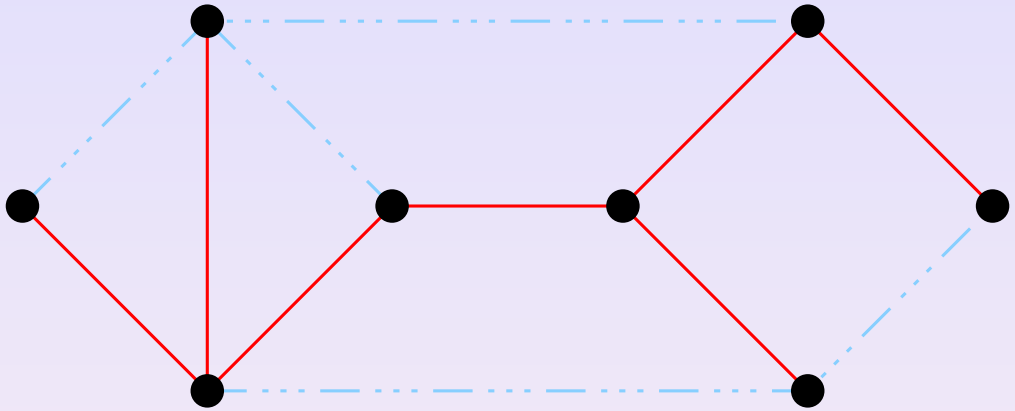
What is a Spanner?

A **spanner** of a graph G is a subgraph spanning $V(G)$



What is a Spanner?

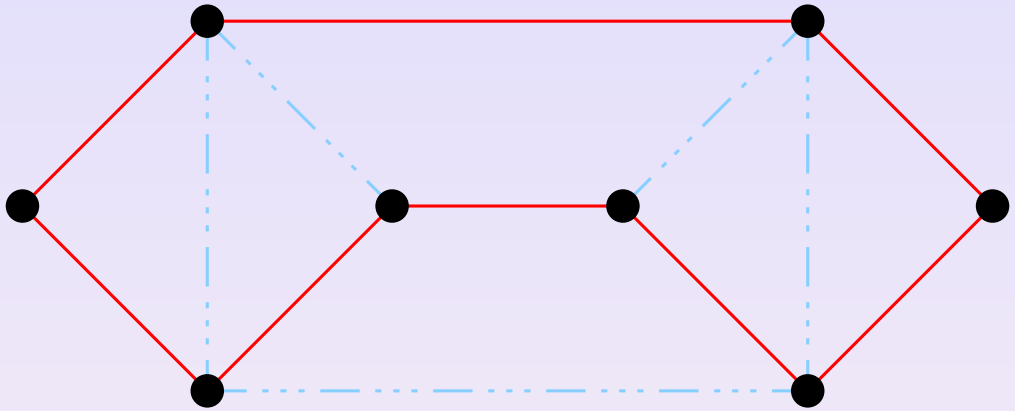
A **spanner** of a graph G is a subgraph spanning $V(G)$



- a spanning tree

What is a Spanner?

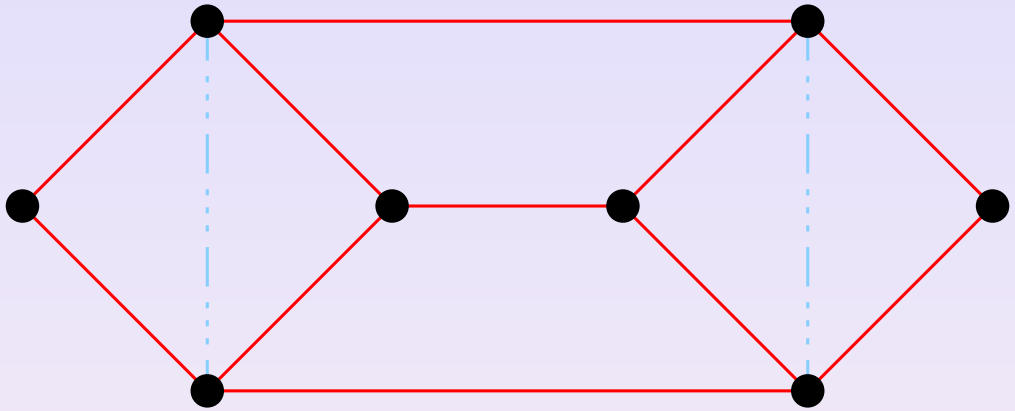
A **spanner** of a graph G is a subgraph spanning $V(G)$



- a spanning tree
- a Hamiltonian cycle

What is a Spanner?

A **spanner** of a graph G is a subgraph spanning $V(G)$



- a spanning tree
- a Hamiltonian cycle
- a maximal bipartite subgraph
- ...

Approximate Distance Spanners

There are two "natural" criteria for a spanner of G:

- **size:** its number of edges.
- **stretch:** its maximum distance distortion from G.

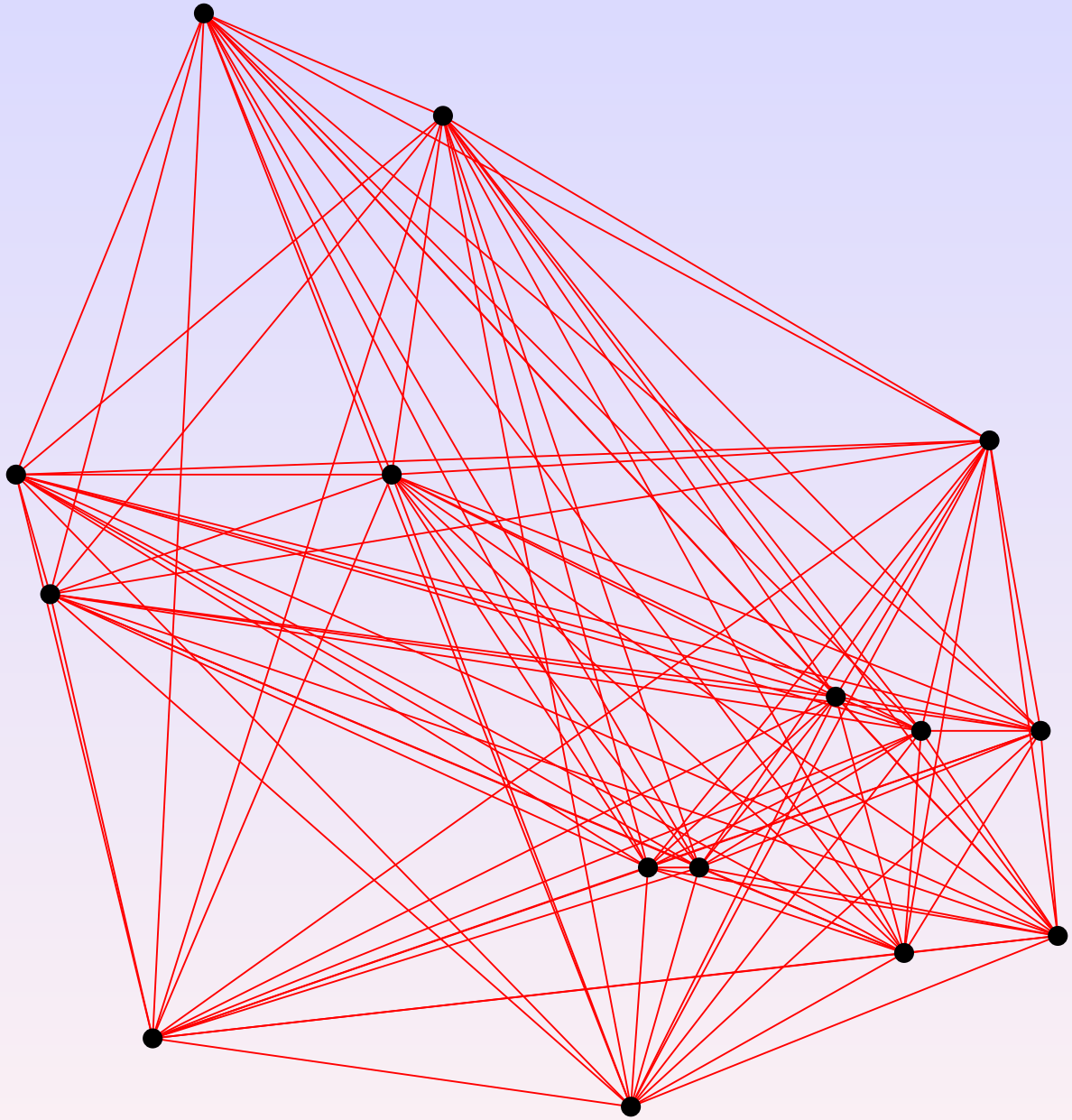
Approximate Distance Spanners

There are two "natural" criteria for a spanner of G:

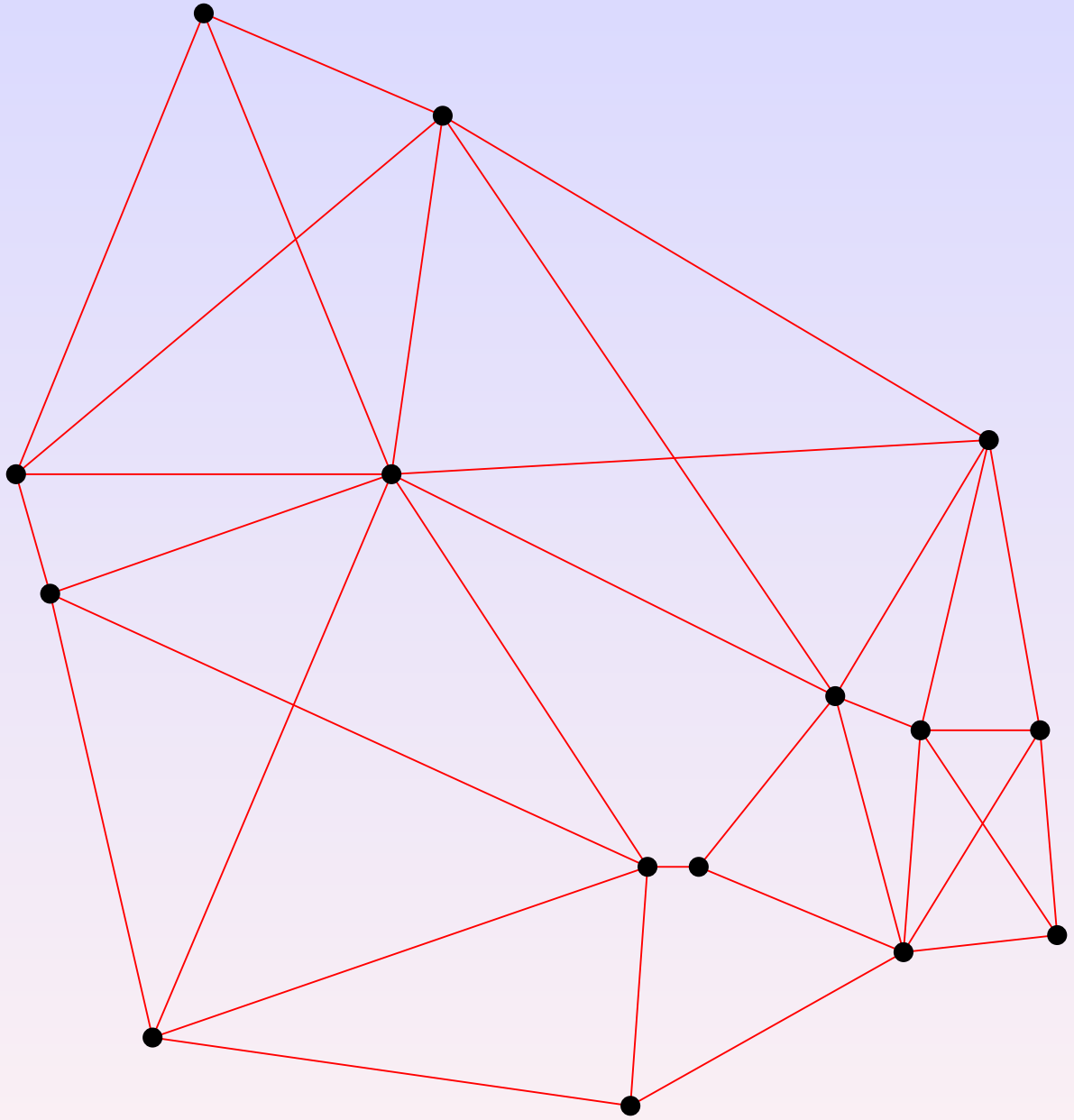
- **size:** its number of edges.
- **stretch:** its maximum distance distortion from G.

Goals:

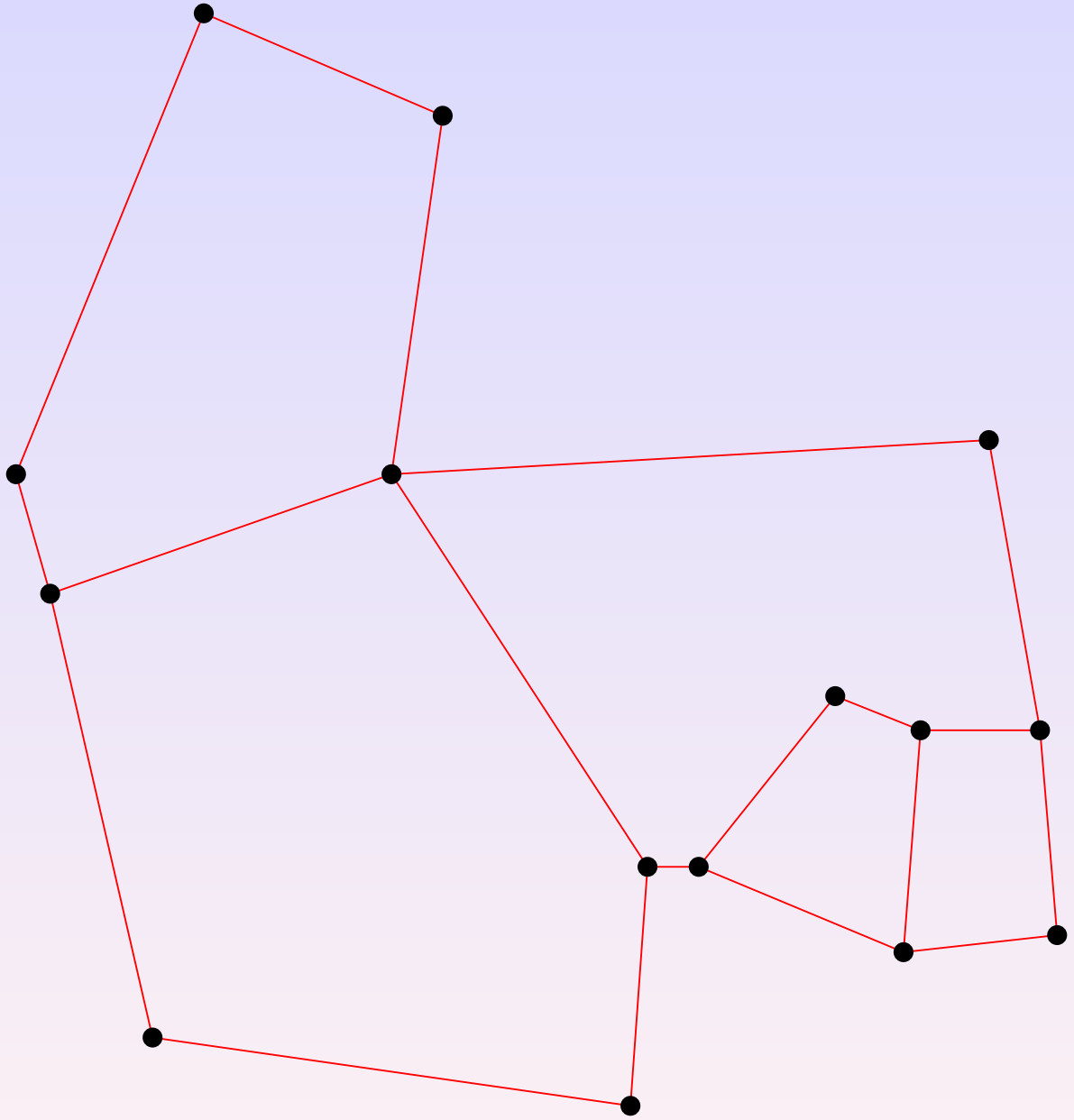
- find a good skeleton of the graph;
- decrease the size of the graph while preserving distances;
- optimize stretch-size tradeoffs.



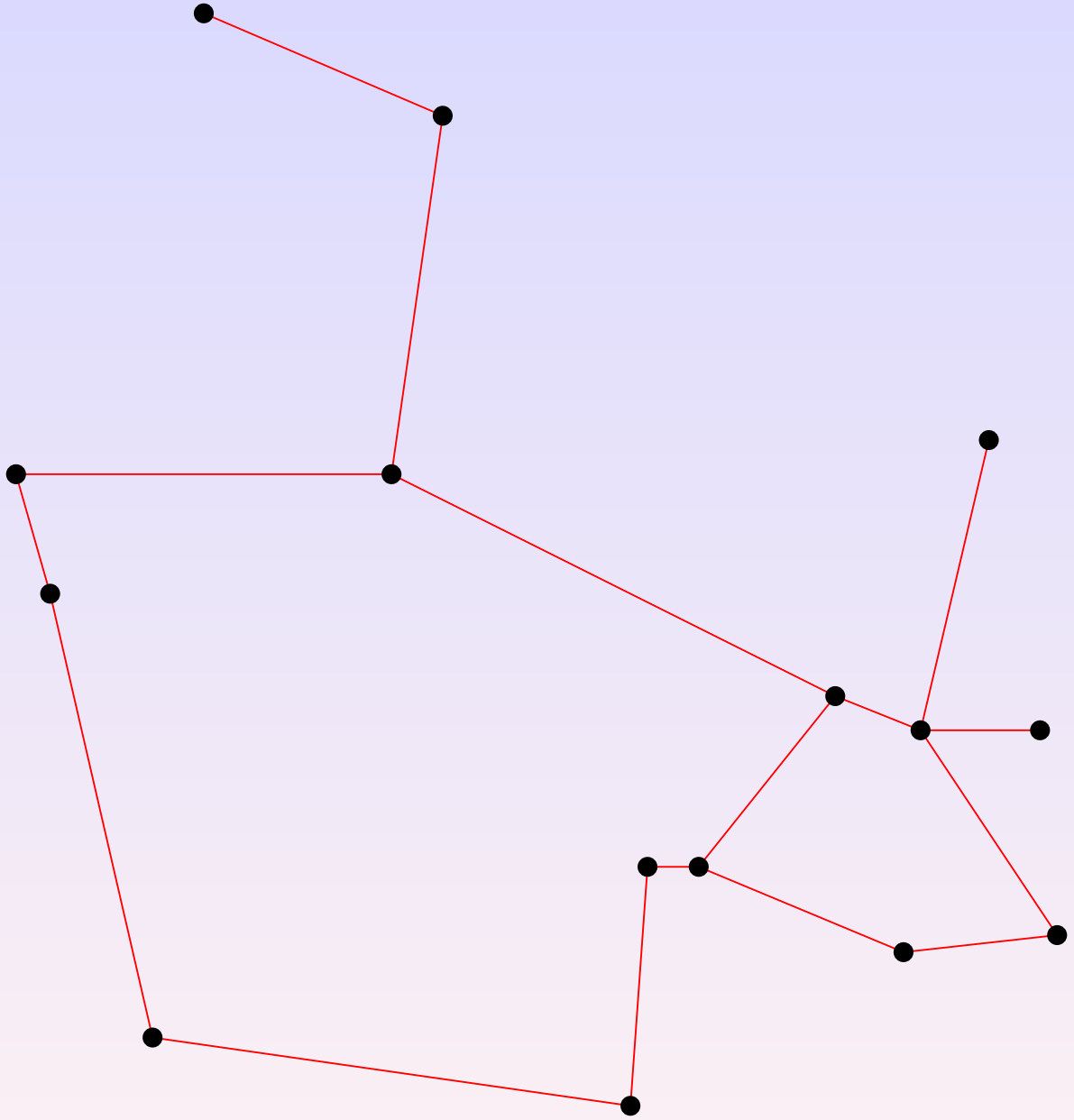
A complete Euclidian graph on 15 nodes



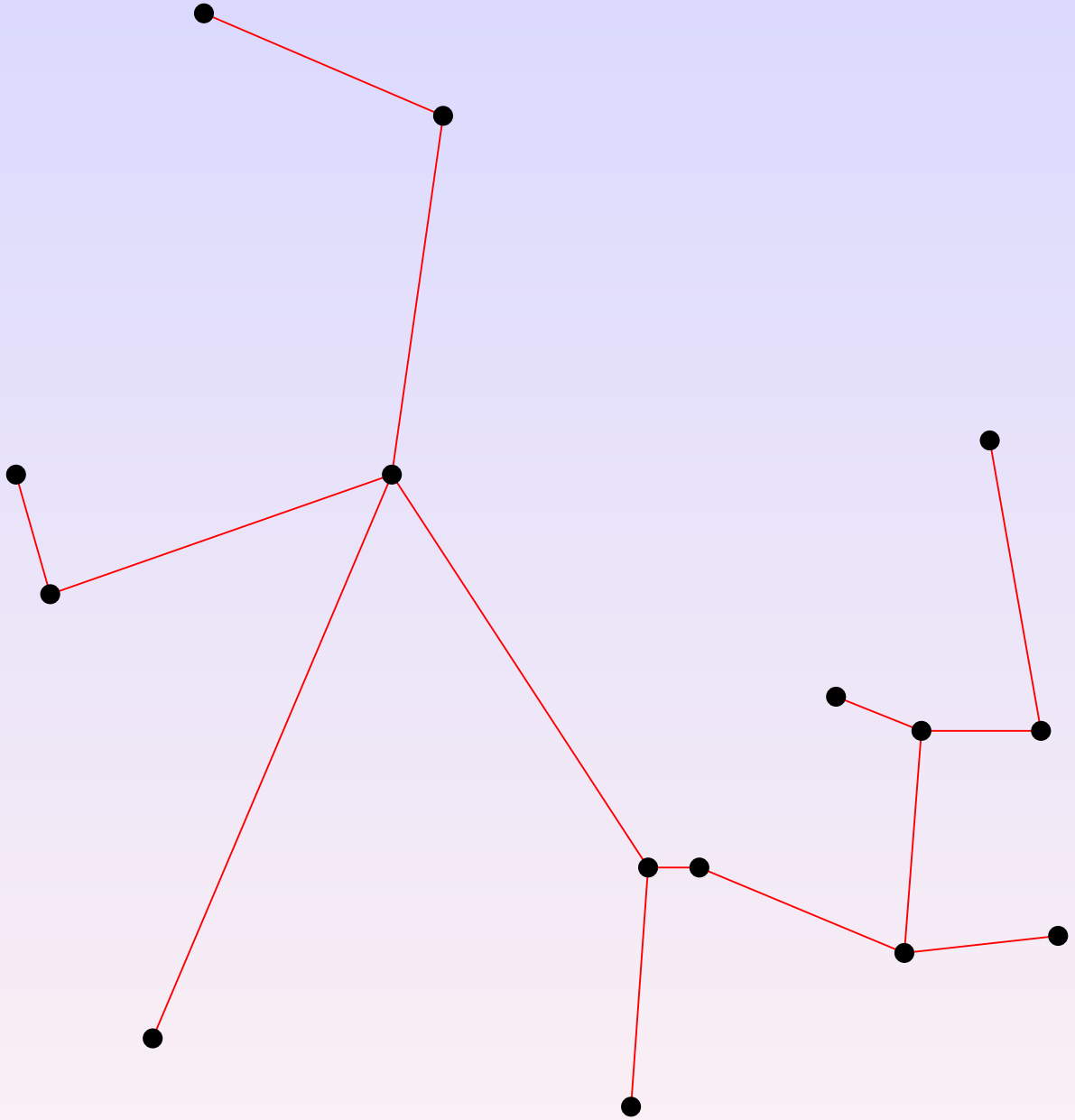
A (minimum cost) spanner with stretch 1.2



A (minimum cost) spanner with stretch 1.7



A (minimum cost) spanner with stretch 2.0



A (minimum cost) spanner with stretch 3.0

More Formally

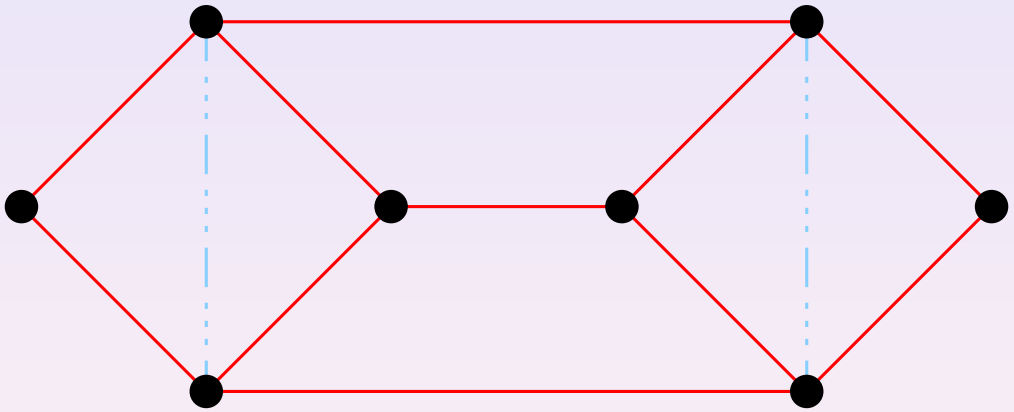
Definition

An (α, β)-spanner S of G is spanner of G satisfying $d_S(x, y) \leqslant \alpha \cdot d_G(x, y) + \beta$ for all $x, y \in V(G)$.

More Formally

Definition

An (α, β)-spanner S of G is spanner of G satisfying $d_S(x, y) \leqslant \alpha \cdot d_G(x, y) + \beta$ for all $x, y \in V(G)$.



A $(2, 0)$-spanner of size 11 which is a $(1, 1)$-spanner as well.

Spanners to do What?

[Peleg-Ullman '87]: *"An optimal synchronizer for the Hypercube"* (440 Google hits)

Used for:

- communication networks
- distributed systems
- network design

Spanners to do What?

[Peleg-Ullman '87]: *"An optimal synchronizer for the Hypercube"* (440 Google hits)

Used for:

- communication networks
- distributed systems
- network design

Synchronizers [Awerbuch JACM '85]

Links with: Sparse Partition [Awerbuch et al. FOCS'90]; Distance Oracle [Thorup-Zwick STOC'01, Baswana et al. SODA'04]; Compact Routing [Peleg-Upfal STOC'89, Thorup-Zwick SPAA'01];

Variant: Geometric Spanners used for TSP (minimize $\sum_{e \in E(S)} \omega(e)$ of within a given stretch)

Some Basic Facts

Theorem (Althöfer et al. '93)

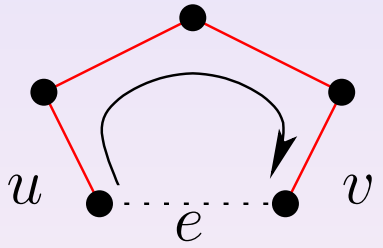
Every graph has a $(2k-1,0)$-spanner with $O(n^{1+1/k})$ edges.

Some Basic Facts

Theorem (Althöfer et al. '93)

Every graph has a $(2k-1, 0)$-spanner with $O(n^{1+1/k})$ edges.

1. $S := \varnothing$ (the empty graph)
2. While $\exists e \in E(G)$ with stretch in S is $> 2k-1$, $S := S \cup \{e\}$



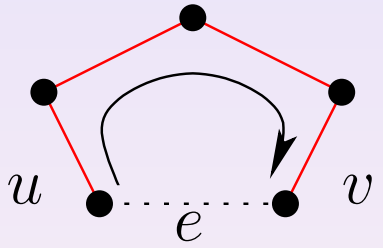
$k = 2$

Some Basic Facts

Theorem (Althöfer et al. '93)

Every graph has a $(2k-1, 0)$-spanner with $O(n^{1+1/k})$ edges.

1. $S := \varnothing$ (the empty graph)
2. While $\exists e \in E(G)$ with stretch in S is $> 2k-1$, $S := S \cup \{e\}$



$k = 2$

Properties:

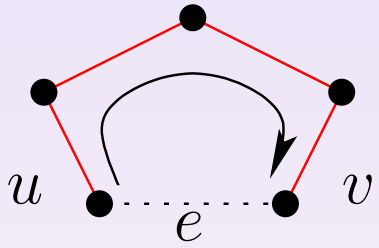
- Stretch of S is $\leqslant 2k-1$.

Some Basic Facts

Theorem (Althöfer et al. '93)

Every graph has a $(2k-1, 0)$-spanner with $O(n^{1+1/k})$ edges.

1. $S := \varnothing$ (the empty graph)
2. While $\exists e \in E(G)$ with stretch in S is $> 2k-1$, $S := S \cup \{e\}$



$k = 2$

Properties:

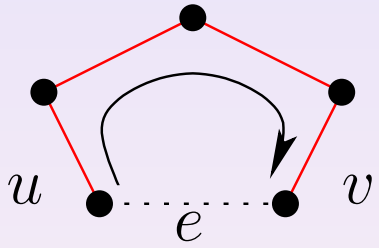
- Stretch of S is $\leqslant 2k-1$.
- Whenever e is added to S, one cannot create any cycle of length $\leqslant 2k$.

Some Basic Facts

Theorem (Althöfer et al. '93)

Every graph has a $(2k-1, 0)$-spanner with $O(n^{1+1/k})$ edges.

1. $S := \varnothing$ (the empty graph)
2. While $\exists e \in E(G)$ with stretch in S is $> 2k-1$, $S := S \cup \{e\}$



$k = 2$

Properties:

- Stretch of S is $\leqslant 2k-1$.
- Whenever e is added to S, one cannot create any cycle of length $\leqslant 2k$. Theorem [Folk] $\Rightarrow S$ has $\leqslant n^{1+1/k}$ edges.

Erdös-Simonovits Conjecture

"For each $k \geqslant 1$, there is a n-node graph with no cycle of length $\leqslant 2k$ with $\Omega(n^{1+1/k})$ edges."

Erdös-Simonovits Conjecture

"For each $k \geqslant 1$, there is a n-node graph with no cycle of length $\leqslant 2k$ with $\Omega(n^{1+1/k})$ edges."

$\Rightarrow$

Conjecture (Proved for $k = 1, 2, 3, 5$)

Every (α, β)-spanner such that $\alpha + \beta < 2k + 1$ requires $\Omega(n^{1+1/k})$ edges for some worst-case graph.

Erdös-Simonovits Conjecture

"For each $k \geqslant 1$, there is a n-node graph with no cycle of length $\leqslant 2k$ with $\Omega(n^{1+1/k})$ edges."

$\Rightarrow$

Conjecture (Proved for $k = 1, 2, 3, 5$)

Every (α, β)-spanner such that $\alpha + \beta < 2k + 1$ requires $\Omega(n^{1+1/k})$ edges for some worst-case graph.

For $k = 2$: a $(3, 0)$-spanner, or a $(1, 2)$-spanner has $\Omega(n^{3/2})$ edges in the worst-case graph.

Distributed Algorithms
(uniform edge-cost)

stretch	size	time	
Deterministic			
$2\log n$	$O(n)$	$2^{O(\sqrt{\log n})}$	[Panconesi et al. '05]
$4k-5$	$O(kn^{1+1/k})$	$2^{O(k)}\log^{k-1} n$	[Derbel et al. '07]
Randomized (expected size)			
$2^{O(\log^* n)}\log n$	$O(n)$	$\log^{1+o(1)} n$	[Pettie '08]
$2k-1$	$O(kn^{1+1/k})$	k	[Baswana et al. '05]
$(1+\varepsilon,\beta)$	$O(\beta n^{1+1/k})$	$O(\beta)$	[Elkin et al. '06] $\beta=(k/\varepsilon)^{O(k\log k)}$

Distributed Algorithms for $k = 2$

	stretch	size	time	
DET.	3	$O(n^{3/2})$	$O(\log n)$	[Derbel et al. '07]
RAND.	3	$O(n^{3/2})$	2	[Baswana et al. '05]

What the Locality of the Problem?

What is the smallest t such that if each node u of a graph knows $B(u,t)$, then u can **deterministically** decide alone which incident edges to keep to form a $(3,0)$-spanner of size $O(n^{3/2})$?

(and more generally a $(2k-1,0)$-spanner of size $O(n^{1+1/k})$?)

The Model

LOCAL model: (a.k.a. Free model, or Linial's model)

- synchrone
- unique IDs
- no size limit messages
- no failures
- simultaneous wake-up
- arbitrary computational power at nodes

Time complexity: number of rounds
(1 round = messages sent/received between all neighbors)

Our Results

Theorem (1)

There is a deterministic distributed algorithm that for every n-node graph computes a $(2k-1,0)$-spanner of size at most $kn^{1+1/k}$ in time k.

Our Results

Theorem (1)

There is a deterministic distributed algorithm that for every n-node graph computes a $(2k-1,0)$-spanner of size at most $kn^{1+1/k}$ in time k.

The time bound "k" is best possible (under the E.-S. Conjecture), even for randomized algorithms (expected time).

Our Results

Theorem (1)

There is a deterministic distributed algorithm that for every n-node graph computes a $(2k-1,0)$-spanner of size at most $kn^{1+1/k}$ in time k.

The time bound "k" is best possible (under the E.-S. Conjecture), even for randomized algorithms (expected time).

If n is unkown (important in practice!), then the algorithm still requires time $O(k)$.

Our Results

Theorem (2)

For every $\varepsilon > 0$, there is a deterministic distributed algorithm that for every n-node graph (where n is unkown to the nodes) computes a $(1+\varepsilon, 2)$-spanner of size $O(\varepsilon^{-1} n^{3/2})$ in $O(\varepsilon^{-1})$ time.

Our Results

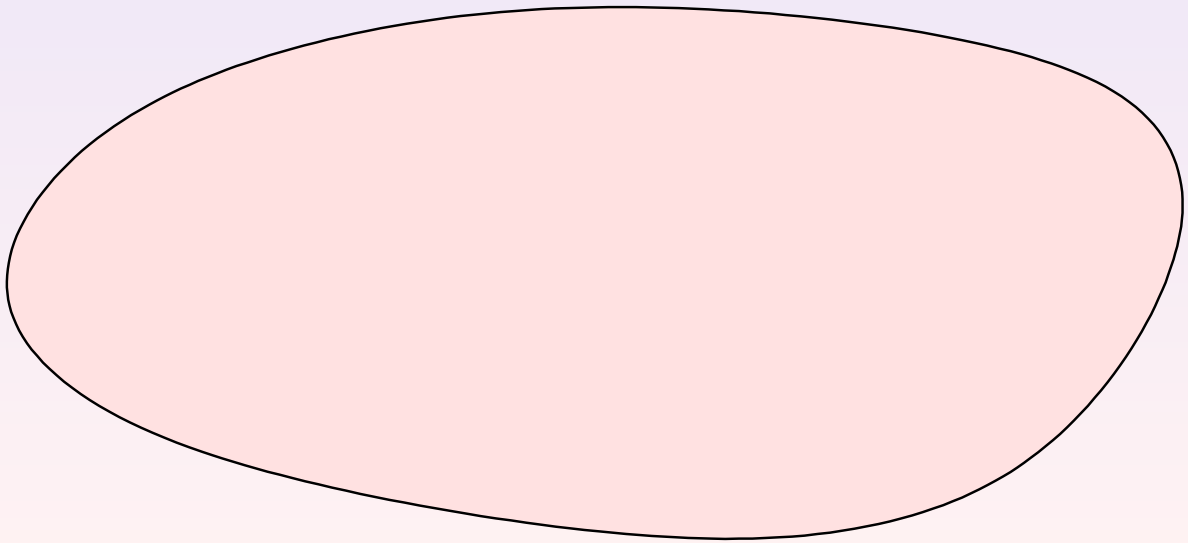
Theorem (2)

For every $\varepsilon > 0$, there is a deterministic distributed algorithm that for every n-node graph (where n is unkown to the nodes) computes a $(1+\varepsilon, 2)$-spanner of size $O(\varepsilon^{-1} n^{3/2})$ in $O(\varepsilon^{-1})$ time.

We also prove that $(1+\varepsilon, 2)$-spanner of size $O(n^{3/2})$ **cannot** be computed in less than $\Omega(\varepsilon^{-1})$ expected time.

Ideas for Previous DETERMINISTIC Algorithms

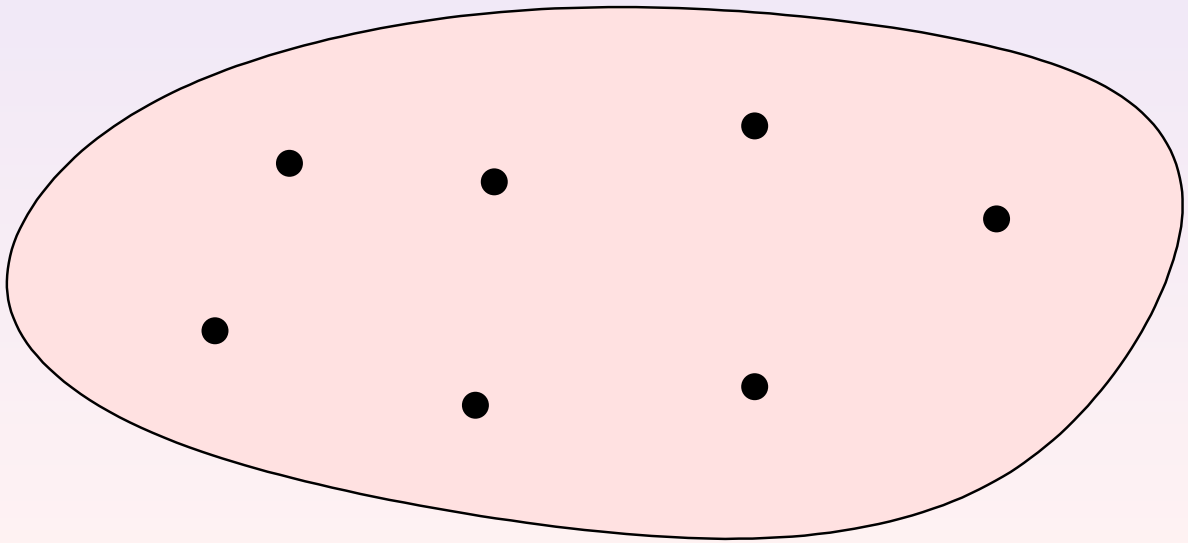
"Sparse Partition" like techniques, here for $k = 2$:



Ideas for Previous Deterministic Algorithms

"Sparse Partition" like techniques, here for $k = 2$:

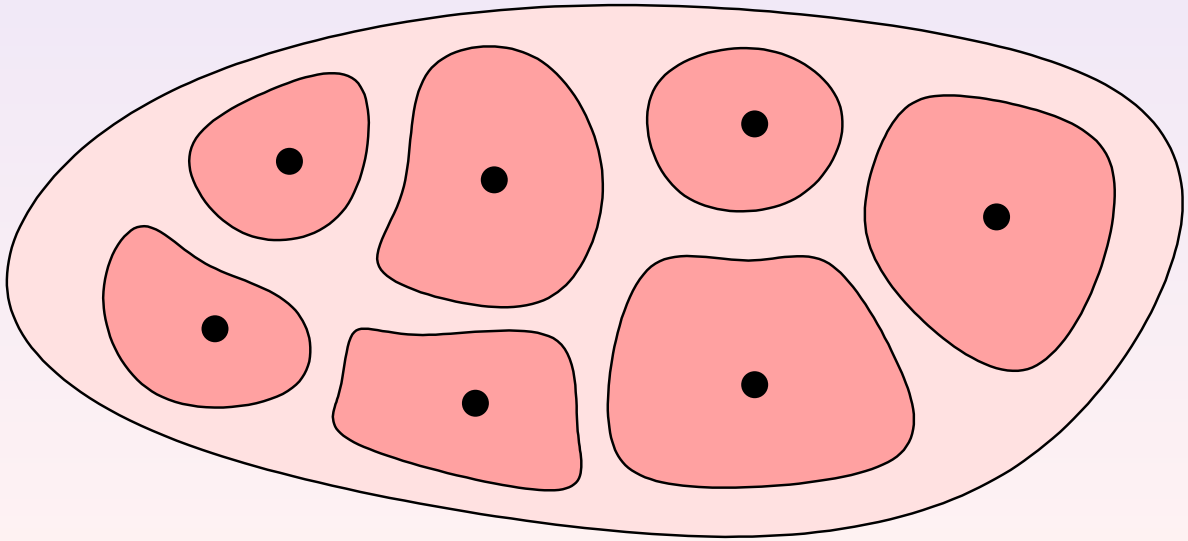
1. Compute an maximal indendent set (MIS) X in G^2
   $\Rightarrow$ points pairwise at distance $\geqslant 3$ and $\leqslant 5$



Ideas for Previous DETERMINISTIC Algorithms

"Sparse Partition" like techniques, here for $k = 2$:

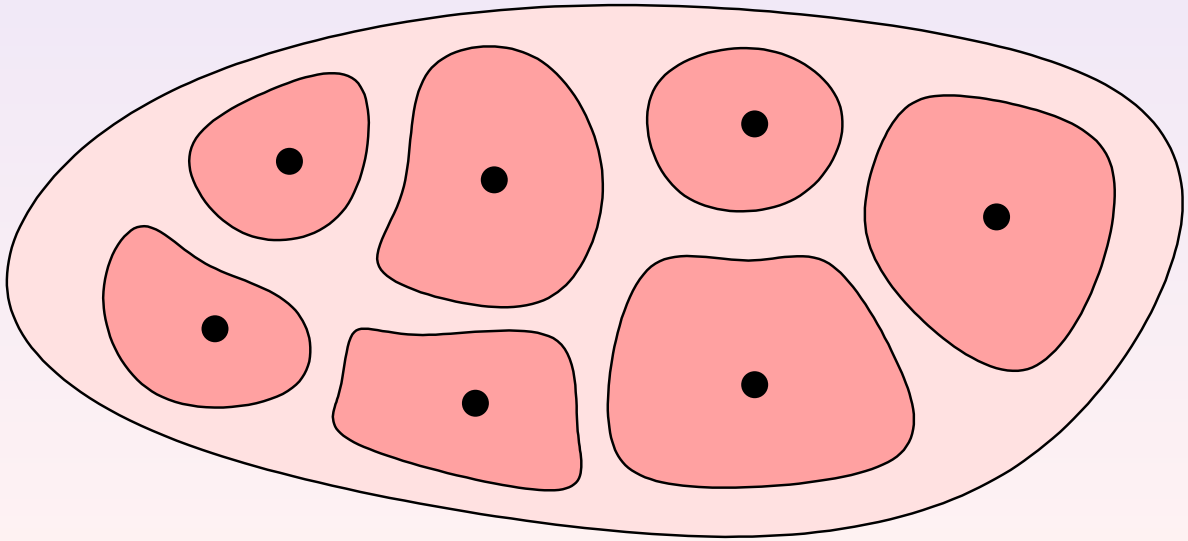
1. Compute an maximal indendent set (MIS) X in G^2
   $\Rightarrow$ points pairwise at distance $\geqslant 3$ and $\leqslant 5$
2. Create independent regions with centers in $X|_{\deg \geqslant \sqrt{n}}$



Ideas for Previous DETERMINISTIC Algorithms

"Sparse Partition" like techniques, here for $k = 2$:

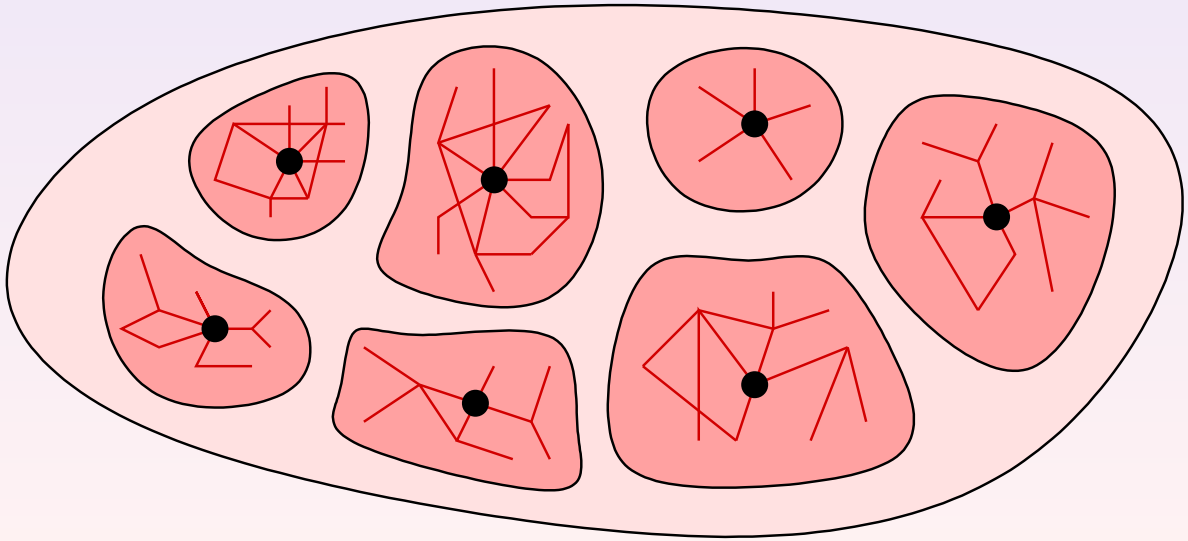
1. Compute an maximal indendent set (MIS) X in G^2
   $\Rightarrow$ points pairwise at distance $\geqslant 3$ and $\leqslant 5$
2. Create independent regions with centers in $X|_{\deg \geqslant \sqrt{n}}$
   $\Rightarrow$ regions of radius $\leqslant 2$ and number of dense regions $\leqslant \sqrt{n}$



Ideas for Previous DETERMINISTIC Algorithms

"Sparse Partition" like techniques, here for $k = 2$:

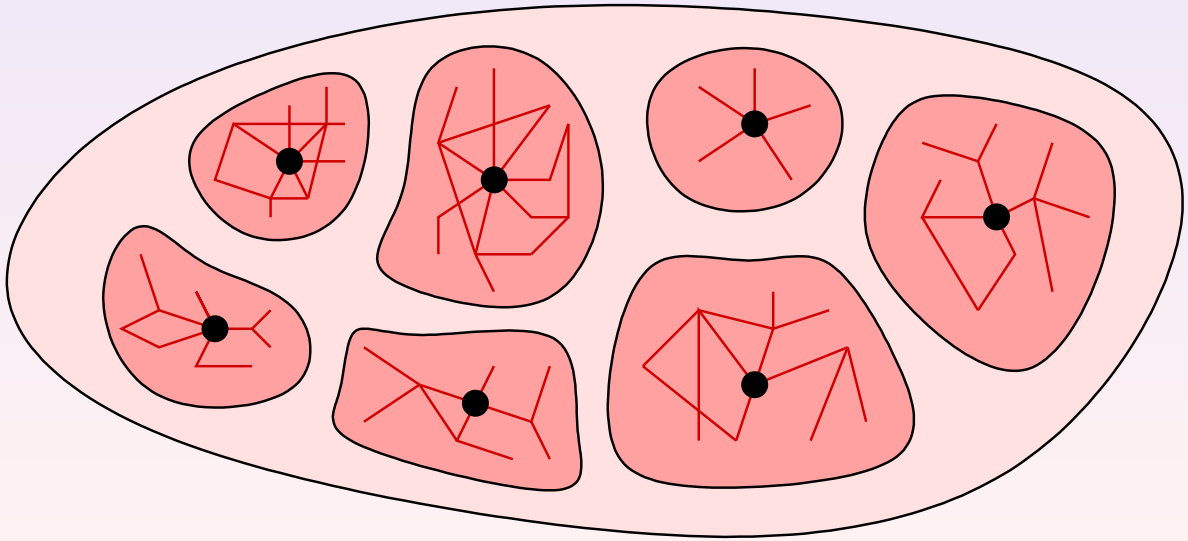
1. Compute an maximal indendent set (MIS) X in G^2
   $\Rightarrow$ points pairwise at distance $\geqslant 3$ and $\leqslant 5$
2. Create independent regions with centers in $X|_{\deg \geqslant \sqrt{n}}$
   $\Rightarrow$ regions of radius $\leqslant 2$ and number of dense regions $\leqslant \sqrt{n}$
3. In parallel compute a "good" spanner in each region



Ideas for Previous DETERMINISTIC Algorithms

"Sparse Partition" like techniques, here for $k = 2$:

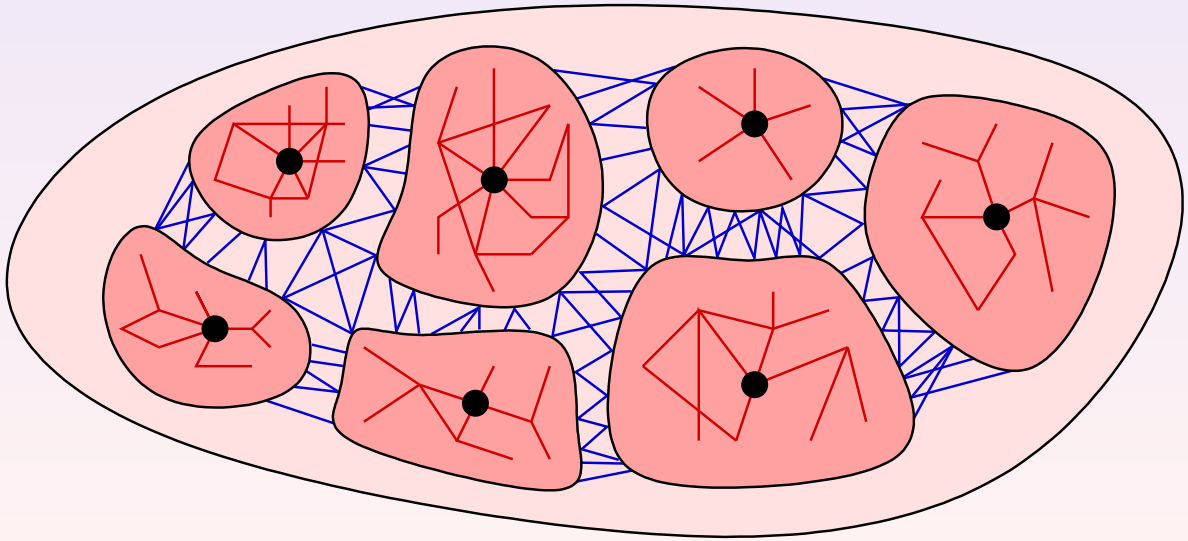
1. Compute an maximal indendent set (MIS) X in G^2
   $\Rightarrow$ points pairwise at distance $\geqslant 3$ and $\leqslant 5$
2. Create independent regions with centers in $X|_{\deg \geqslant \sqrt{n}}$
   $\Rightarrow$ regions of radius $\leqslant 2$ and number of dense regions $\leqslant \sqrt{n}$
3. In parallel compute a "good" spanner in each region
   $\Rightarrow$ optimal stretch-size tradeoff in 2 rounds



Ideas for Previous DETERMINISTIC Algorithms

"Sparse Partition" like techniques, here for $k = 2$:

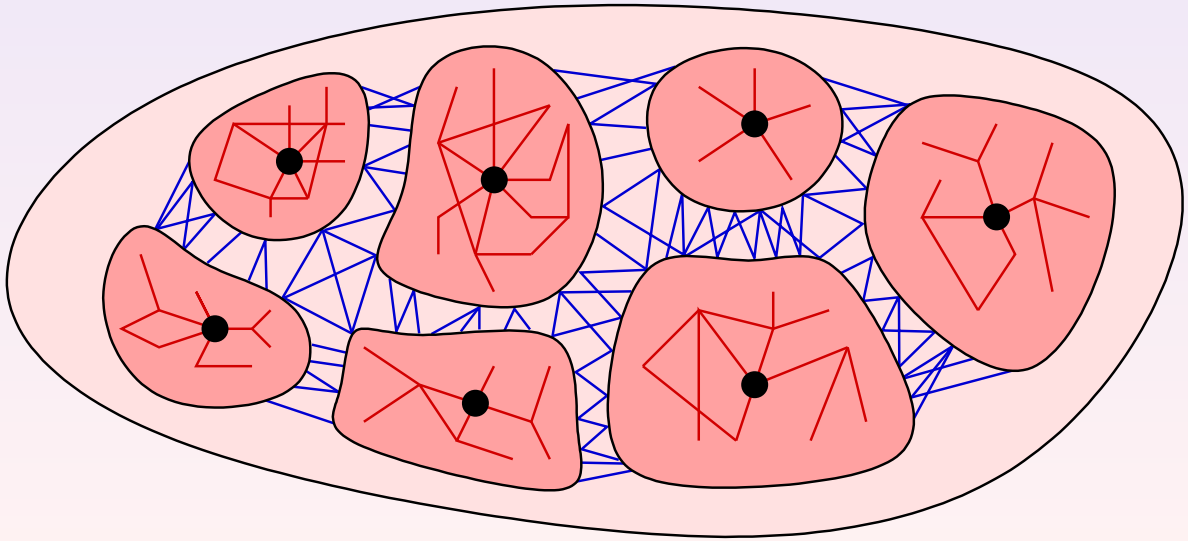
1. Compute an maximal indendent set (MIS) X in G^2
   $\Rightarrow$ points pairwise at distance $\geqslant 3$ and $\leqslant 5$
2. Create independent regions with centers in $X|_{\deg \geqslant \sqrt{n}}$
   $\Rightarrow$ regions of radius $\leqslant 2$ and number of dense regions $\leqslant \sqrt{n}$
3. In parallel compute a "good" spanner in each region
   $\Rightarrow$ optimal stretch-size tradeoff in 2 rounds
4. Cover inter-region edges (bipartite) with length-3 paths



Ideas for Previous DETERMINISTIC Algorithms

"Sparse Partition" like techniques, here for $k = 2$:

1. Compute an maximal indendent set (MIS) X in G^2
   $\Rightarrow$ points pairwise at distance $\geqslant 3$ and $\leqslant 5$
2. Create independent regions with centers in $X|_{\deg \geqslant \sqrt{n}}$
   $\Rightarrow$ regions of radius $\leqslant 2$ and number of dense regions $\leqslant \sqrt{n}$
3. In parallel compute a "good" spanner in each region
   $\Rightarrow$ optimal stretch-size tradeoff in 2 rounds
4. Cover inter-region edges (bipartite) with length-3 paths
   $\Rightarrow$ doable with size $n + 2|R_x|\sqrt{n}$ for dense region R_x



Bottleneck: Sparse Partition Construction

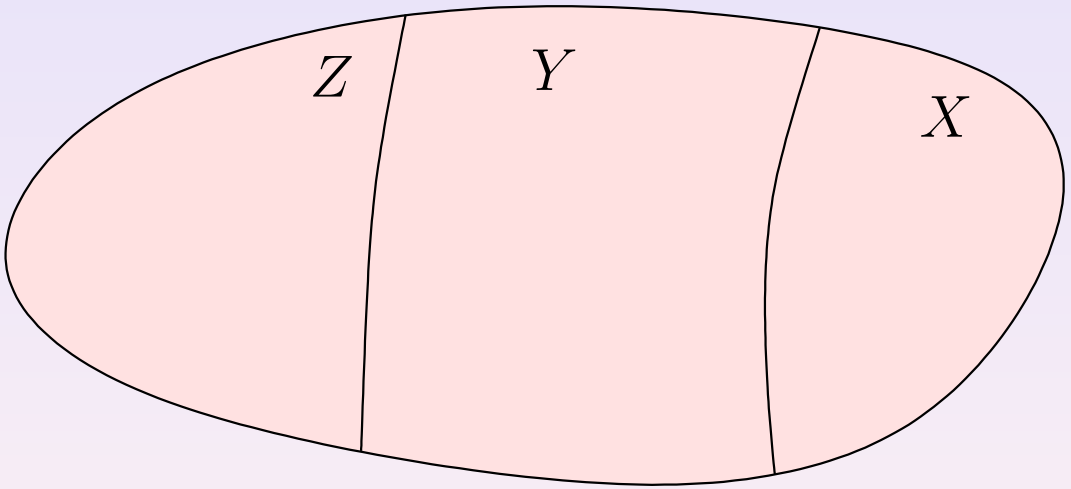
Computing quickly an MIS is difficult.
It is solved sequentially by (inherently?) greedy algorithms.

- Upper bound: $2^{O(\sqrt{\log n})}$ [Panconesi et al. STOC'96]
- Lower bound: $\Omega(\sqrt{\log n / \log\log n})$ [Kuhn et al. PODC'06]

Ideas for RANDOMIZED Algorithms

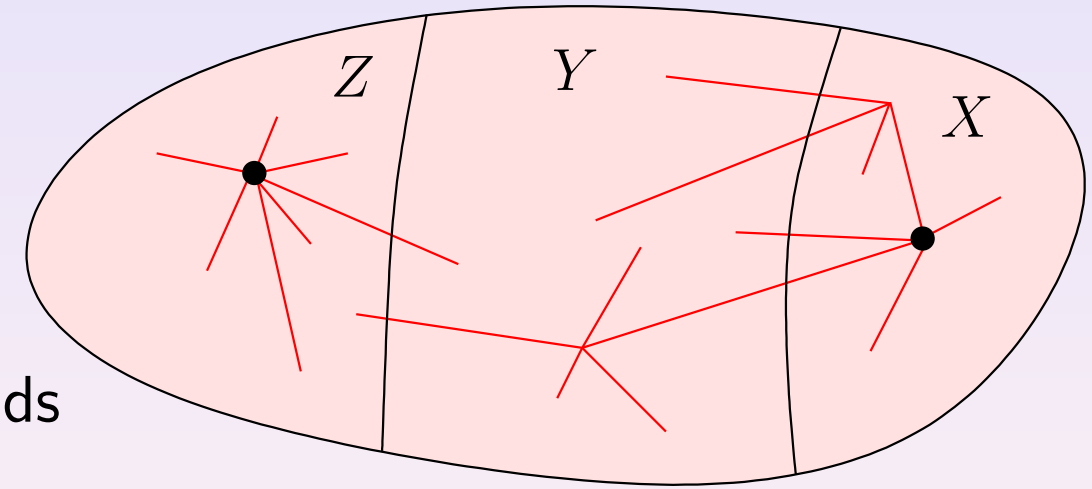
[Baswana-Pettie '05, here for $k = 2$]

1. $b_u := 1|0$ with proba $1/\sqrt{n}$. Let $X := \{u \mid b_u = 1\}$
2. $[u \in Z]$ if $B(u,1) \cap X = \varnothing$, then $S_u := B(u,1)$
3. $[u \in X]$ if $b_u = 1$, then $S_u := \mathrm{BFS}(u, B(u,2))$



Ideas for RANDOMIZED Algorithms
[Baswana-Pettie '05, here for $k = 2$]

1. $b_u := 1|0$ with proba $1/\sqrt{n}$. Let $X := \{u \mid b_u = 1\}$
2. $[u \in Z]$ if $B(u,1) \cap X = \varnothing$, then $S_u := B(u,1)$
3. $[u \in X]$ if $b_u = 1$, then $S_u := \mathrm{BFS}(u, B(u,2))$



Time: 2 rounds
Stretch: 3
Size: $2n^{3/2}$ in expectation
(In expectation: $|X| = \sqrt{n}$, and if $u \in Z$, then $\deg(u) \leqslant \sqrt{n}$)

Observations

In all previous algorithms:

1. Need to distinguish some independent set of nodes (X).
2. Knowledge of n is required.
3. Performences not garanteed for randomized algorithms.

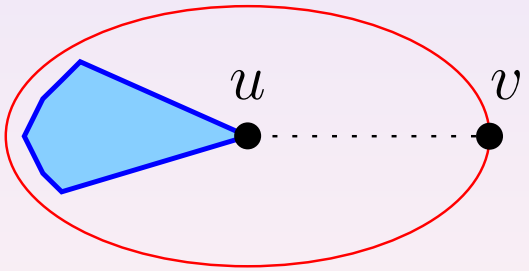
The New Algorithm (for $k = 2$)

For every node u do:

The New Algorithm (for $k = 2$)

For every node u do:

1. $R_u := \{u\} \cup \{$ **any** selection of $|B(u,3)|^{1/2}$ neighbors $\}$
2. send R_u and receive R_v to/from its neighbors
3. $W := B(u,1) \setminus \{v \mid u \in R_v\} \setminus R_u$ and $C := \varnothing$
4. while $\exists w \in W$:
   1. $C := C \cup \{w\}$
   2. $W := W \setminus \{v \in W \mid R_v \cap R_w \neq \varnothing\}$
5. Select edges from u to $R_u \cup C$

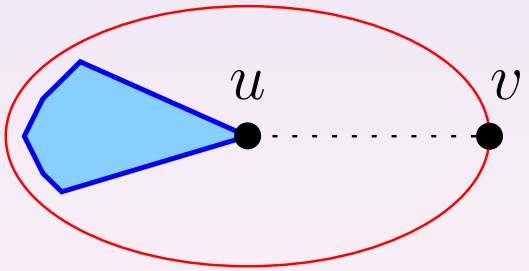


The New Algorithm (for $k = 2$)

For every node u do:

1. $R_u := \{u\} \cup \{$ **any** selection of $|B(u,3)|^{1/2}$ neighbors $\}$
2. send R_u and receive R_v to/from its neighbors
3. $W := B(u,1) \setminus \{v \mid u \in R_v\} \setminus R_u$ and $C := \varnothing$
4. while $\exists w \in W$:
   1. $C := C \cup \{w\}$
   2. $W := W \setminus \{v \in W \mid R_v \cap R_w \neq \varnothing\}$
5. Select edges from u to $R_u \cup C$

Stretch: if $(u,v) \notin S$, then v removed from W in 4.2.

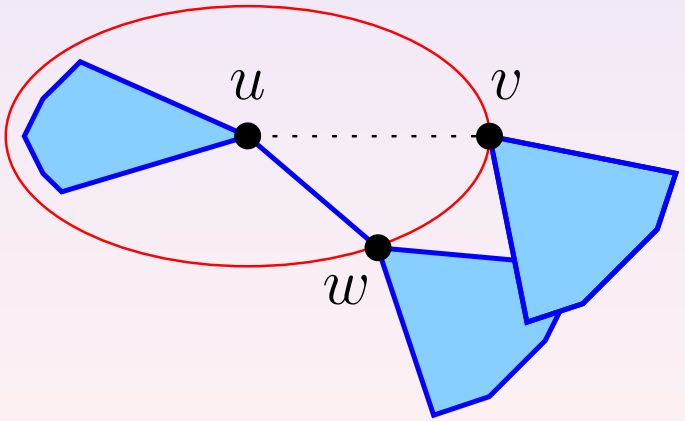


The New Algorithm (for $k = 2$)

For every node u do:

1. $R_u := \{u\} \cup \{$ **any** selection of $|B(u,3)|^{1/2}$ neighbors $\}$
2. send R_u and receive R_v to/from its neighbors
3. $W := B(u,1) \setminus \{v \mid u \in R_v\} \setminus R_u$ and $C := \varnothing$
4. while $\exists w \in W$:
   1. $C := C \cup \{w\}$
   2. $W := W \setminus \{v \in W \mid R_v \cap R_w \neq \varnothing\}$
5. Select edges from u to $R_u \cup C$

Stretch: if $(u,v) \notin S$, then v removed from W in 4.2. Thus $\exists w \in C$ with $R_v \cap R_w \neq \varnothing$. Hence, stretch $\leqslant 3$.

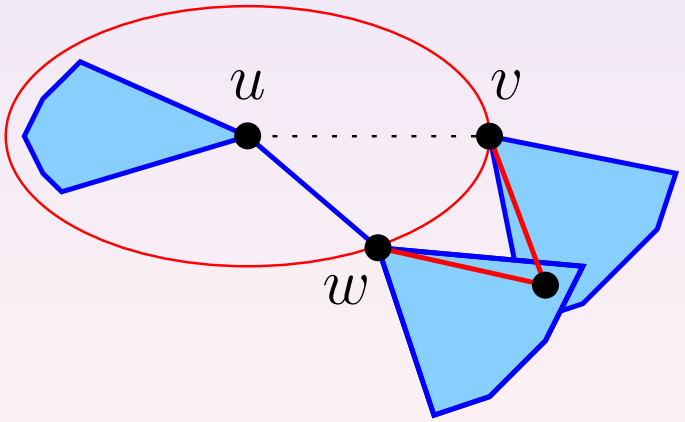


The New Algorithm (for $k = 2$)

For every node u do:

1. $R_u := \{u\} \cup \{$ **any** selection of $|B(u,3)|^{1/2}$ neighbors $\}$
2. send R_u and receive R_v to/from its neighbors
3. $W := B(u,1) \setminus \{v \mid u \in R_v\} \setminus R_u$ and $C := \varnothing$
4. while $\exists w \in W$:
   1. $C := C \cup \{w\}$
   2. $W := W \setminus \{v \in W \mid R_v \cap R_w \neq \varnothing\}$
5. Select edges from u to $R_u \cup C$

Stretch: if $(u,v) \notin S$, then v removed from W in 4.2. Thus $\exists w \in C$ with $R_v \cap R_w \neq \varnothing$. Hence, stretch $\leqslant 3$.

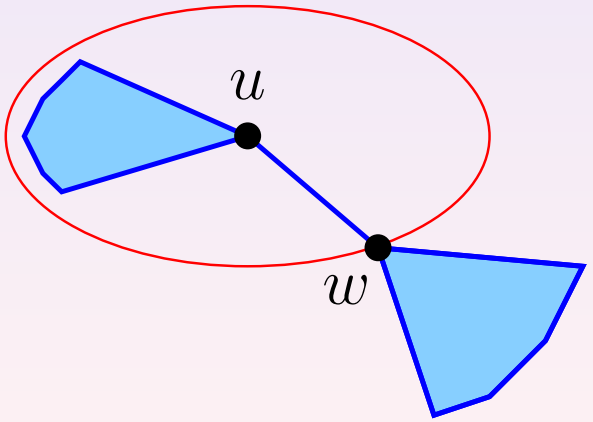


The New Algorithm (for $k = 2$)

For every node u do:

1. $R_u := \{u\} \cup \{$ **any** selection of $|B(u,3)|^{1/2}$ neighbors $\}$
2. send R_u and receive R_v to/from its neighbors
3. $W := B(u,1) \setminus \{v \mid u \in R_v\} \setminus R_u$ and $C := \varnothing$
4. while $\exists w \in W$:
   1. $C := C \cup \{w\}$
   2. $W := W \setminus \{v \in W \mid R_v \cap R_w \neq \varnothing\}$
5. Select edges from u to $R_u \cup C$

Size: $\forall w \in W$ after $\boxed{3}$ has degree $\geqslant |B(w,3)|^{1/2}$.

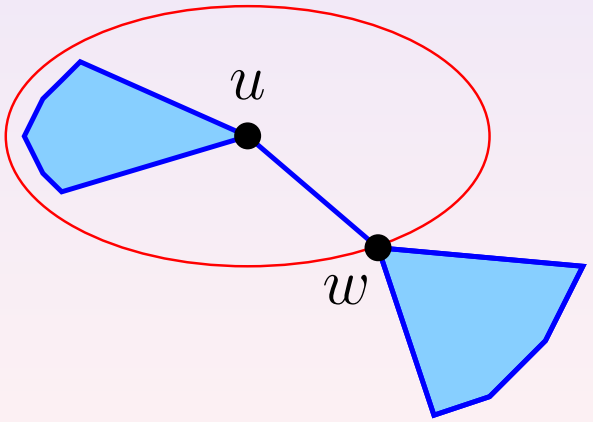


The New Algorithm (for $k = 2$)

For every node u do:

1. $R_u := \{u\} \cup \{$ **any** selection of $|B(u,3)|^{1/2}$ neighbors $\}$
2. send R_u and receive R_v to/from its neighbors
3. $W := B(u,1) \setminus \{v \mid u \in R_v\} \setminus R_u$ and $C := \varnothing$
4. while $\exists w \in W$:
   1. $C := C \cup \{w\}$
   2. $W := W \setminus \{v \in W \mid R_v \cap R_w \neq \varnothing\}$
5. Select edges from u to $R_u \cup C$

Size: $\forall w \in W$ after $\boxed{3}$ has degree $\geqslant |B(w,3)|^{1/2}$. Thus, $\deg(w) \geqslant |B(u,2)|^{1/2}$ since $B(u,2) \subseteq B(w,3)$.

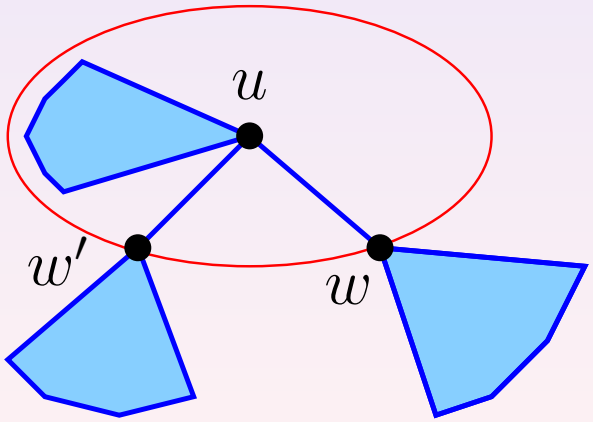


The New Algorithm (for $k = 2$)

For every node u do:

1. $R_u := \{u\} \cup \{$ **any** selection of $|B(u,3)|^{1/2}$ neighbors $\}$
2. send R_u and receive R_v to/from its neighbors
3. $W := B(u,1) \setminus \{v \mid u \in R_v\} \setminus R_u$ and $C := \varnothing$
4. while $\exists w \in W$:
   1. $C := C \cup \{w\}$
   2. $W := W \setminus \{v \in W \mid R_v \cap R_w \neq \varnothing\}$
5. Select edges from u to $R_u \cup C$

Size: $\forall w \in W$ after $\boxed{3}$ has degree $\geqslant |B(w,3)|^{1/2}$. Thus, $\deg(w) \geqslant |B(u,2)|^{1/2}$ since $B(u,2) \subseteq B(w,3)$. R_w taken in C are disjoint. #loops is $|C| \leqslant |B(u,2)|/|B(u,2)|^{1/2} \leqslant \sqrt{n}$. Size: $n \times 2\sqrt{n}$.



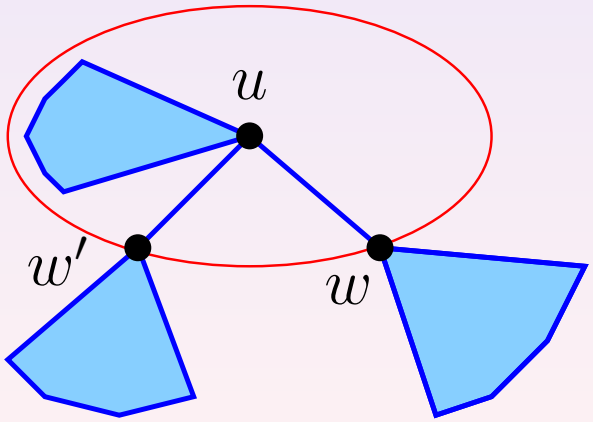
The New Algorithm (for $k = 2$)

For every node u do:

1. $R_u := \{u\} \cup \{$ **any** selection of $|B(u,3)|^{1/2}$ neighbors $\}$
2. send R_u and receive R_v to/from its neighbors
3. $W := B(u,1) \setminus \{v \mid u \in R_v\} \setminus R_u$ and $C := \varnothing$
4. while $\exists w \in W$:
   1. $C := C \cup \{w\}$
   2. $W := W \setminus \{v \in W \mid R_v \cap R_w \neq \varnothing\}$
5. Select edges from u to $R_u \cup C$

Conclusion:

- Stretch: 3
- Size: $2n^{3/2}$
- Time: 3 (or 2 if n known)



Early registration: August 25!!!

Thank You
for your attention