Deterministic Distributed Construction of Linear Stretch Spanners in Polylogarithmic Time

Bilel Derbel (LIFL - Univ. Lille 1, France),

Cyril Gavoille (LaBRI - Univ. Bordeaux 1, France),

David Peleg (The Weizmann Institute, Israel)

DISC'07 - Cyprus - Lemosos

24 September 2007

Problem and Goal

- An (α,β)-spanner of G is a subgraph S such that for every two nodes u and v :

$$d_S(u,v) \leq \alpha \cdot d_G(u,v) + \beta.$$

Problem and Goal

- An (α,β)-spanner of G is a subgraph S such that for every two nodes u and v :

$$d_S(u,v) \leq \alpha \cdot d_G(u,v) + \beta.$$

- Quality of a spanner
  - Size and stretch
  - **[Extremal graph theory]** Every graph G has a $(2k-1,0)$-spanner with $O(n^{1+1/k})$ edges.
    - this stretch-size trade-off is believed to be tight (proved for $k = 1, 2, 3, 5$).

Problem and Goal

- An (α,β)-spanner of G is a subgraph S such that for every two nodes u and v :

$$d_S(u,v) \leq \alpha \cdot d_G(u,v) + \beta.$$

- Quality of a spanner
  - Size and stretch
  - **[Extremal graph theory]** Every graph G has a $(2k-1,0)$-spanner with $O(n^{1+1/k})$ edges.
    - this stretch-size trade-off is believed to be tight (proved for $k = 1, 2, 3, 5$).

We want to **efficiently** compute a **high quality** spanner in a **distributed way**.

Motivation

- What kind of spanner can we construct assuming only some local knowledge ?

Motivation

- What kind of spanner can we construct assuming only some local knowledge ?
- $\mathcal{LOCAL}$ model (Linial's model) : unlimited message size.

Motivation

- What kind of spanner can we construct assuming only some local knowledge ?
- $\mathcal{LOCAL}$ model (Linial's model) : unlimited message size.

	(α, β)	size	time
[ABCP'96]	$(4k-3, 0)$	$O(kn^{1+1/k})$	$n^{\epsilon+1/k}$
[DG'06]	$O(k^{\log 5})$	$O(\log k \cdot n^{1+1/k})$	n^{ϵ}

Motivation

- What kind of spanner can we construct assuming only some local knowledge ?
- $\mathcal{LOCAL}$ model (Linial's model) : unlimited message size.

	(α, β)	size	time	expected time
[ABCP'96]	$(4k-3, 0)$	$O(kn^{1+1/k})$	$n^{\epsilon+1/k}$	$n^{1/k} \cdot \log^2 n$
[DG'06]	$O(k^{\log 5})$	$O(\log k \cdot n^{1+1/k})$	n^{ϵ}	$\log n$

Motivation

- What kind of spanner can we construct assuming only some local knowledge ?
- $\mathcal{LOCAL}$ model (Linial's model) : unlimited message size.

	(α, β)	size	time	expected time
[ABCP'96]	$(4k-3, 0)$	$O(kn^{1+1/k})$	$n^{\epsilon+1/k}$	$n^{1/k} \cdot \log^2 n$
[DG'06]	$O(k^{\log 5})$	$O(\log k \cdot n^{1+1/k})$	n^{ϵ}	$\log n$

	(α, β)	expected size	time
[BS'03]	$(2k-1, 0)$	$O(k \cdot n^{1+1/k})$	$O(k^2)$
[DMPRS'03]	$(O(\log n), 0)$	$O(\log n)$	$O(\log^3 n)$

Motivation

- What kind of spanner can we construct assuming only some local knowledge ?
- $\mathcal{LOCAL}$ model (Linial's model) : unlimited message size.

	(α, β)	size	time	expected time
[ABCP'96]	$(4k-3, 0)$	$O(kn^{1+1/k})$	$n^{\epsilon+1/k}$	$n^{1/k} \cdot \log^2 n$
[DG'06]	$O(k^{\log 5})$	$O(\log k \cdot n^{1+1/k})$	n^{ϵ}	$\log n$

	(α, β)	expected size	time
[BS'03]	$(2k-1, 0)$	$O(k \cdot n^{1+1/k})$	$O(k^2)$
[DMPRS'03]	$(O(\log n), 0)$	$O(\log n)$	$O(\log^3 n)$

Is it possible to construct linear stretch spanners deterministically in polylogarithmic time ?

Main Ideas

We add edges in parallel at different regions of the graph

Main Ideas

We add edges in parallel at different regions of the graph

We use an independent ρ-dominating set to break the symmetry efficiently

- $\forall u \in V$, $\exists v \in \mathrm{IDS}(G, \rho)$ such that $d_G(u, v) \leq \rho$.
- $\forall u, v \in \mathrm{IDS}(G, \rho)$, $d_G(u, v) \geq 2$.

Main Ideas

We add edges in parallel at different regions of the graph

We use an independent ρ-dominating set to break the symmetry efficiently

We use the best sequential algorithm to span the interior of a region

We use a new sequential algorithm for bipartite graphs to span the border of a region

Sequential algorithms : $k = 2$

Lemma 1 [Folklore] :
Every graph $G = (V, E)$ has a 3-spanner with $O(|V|^{3/2})$ edges.

Lemma 2 :
Every bipartite graph $B = (W \cup V, E)$ has a 3-spanner with $O(|V| + |W|\sqrt{|V|})$ edges.

3-spanner for Bipartite Graphs

Lemma 2 : Every bipartite graph $B = (W \cup V, E)$ has a 3-spanner with $O(|V| + |W|\sqrt{|V|})$ edges.

3-spanner for Bipartite Graphs

Lemma 2 : Every bipartite graph $B = (W \cup V, E)$ has a 3-spanner with $O(|V| + |W|\sqrt{|V|})$ edges.

Algorithm : Repeat

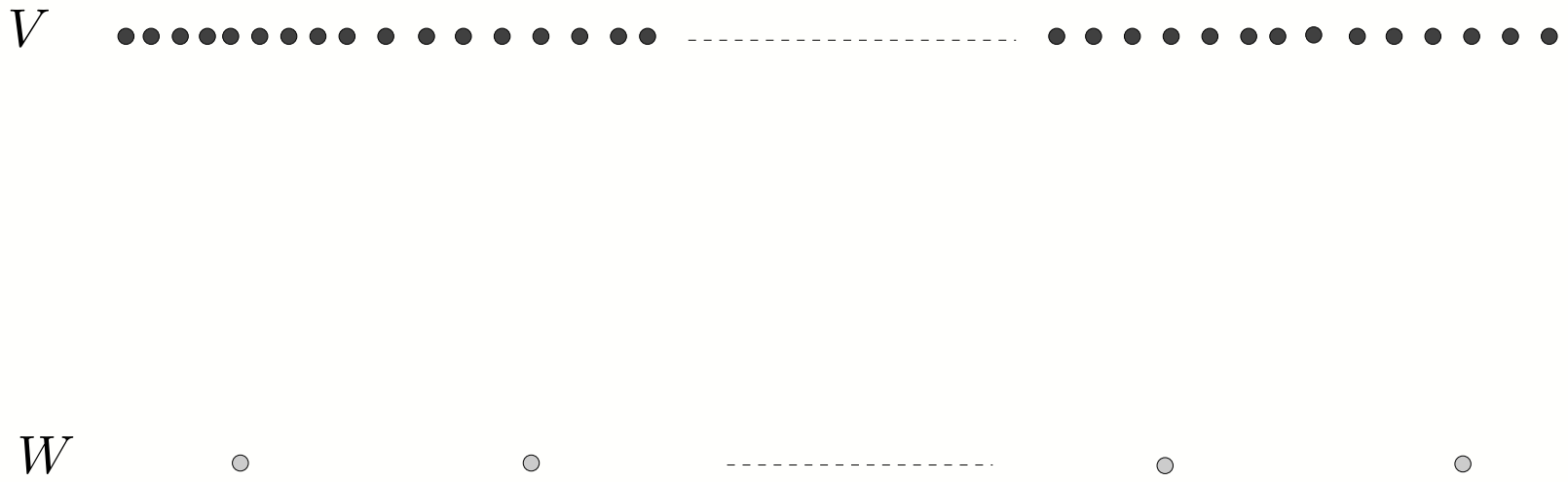
1. chose the highest degree node w in W
2. add the star $\mathcal{N}(w)$ to the spanner
3. connect w to other stars

3-spanner for Bipartite Graphs

Lemma 2 : Every bipartite graph $B = (W \cup V, E)$ has a 3-spanner with $O(|V| + |W|\sqrt{|V|})$ edges.

Algorithm : Repeat

1. chose the highest degree node w in W
2. add the star $\mathcal{N}(w)$ to the spanner
3. connect w to other stars



3-spanner for Bipartite Graphs

Lemma 2 : Every bipartite graph $B = (W \cup V, E)$ has a 3-spanner with $O(|V| + |W|\sqrt{|V|})$ edges.

Algorithm : Repeat

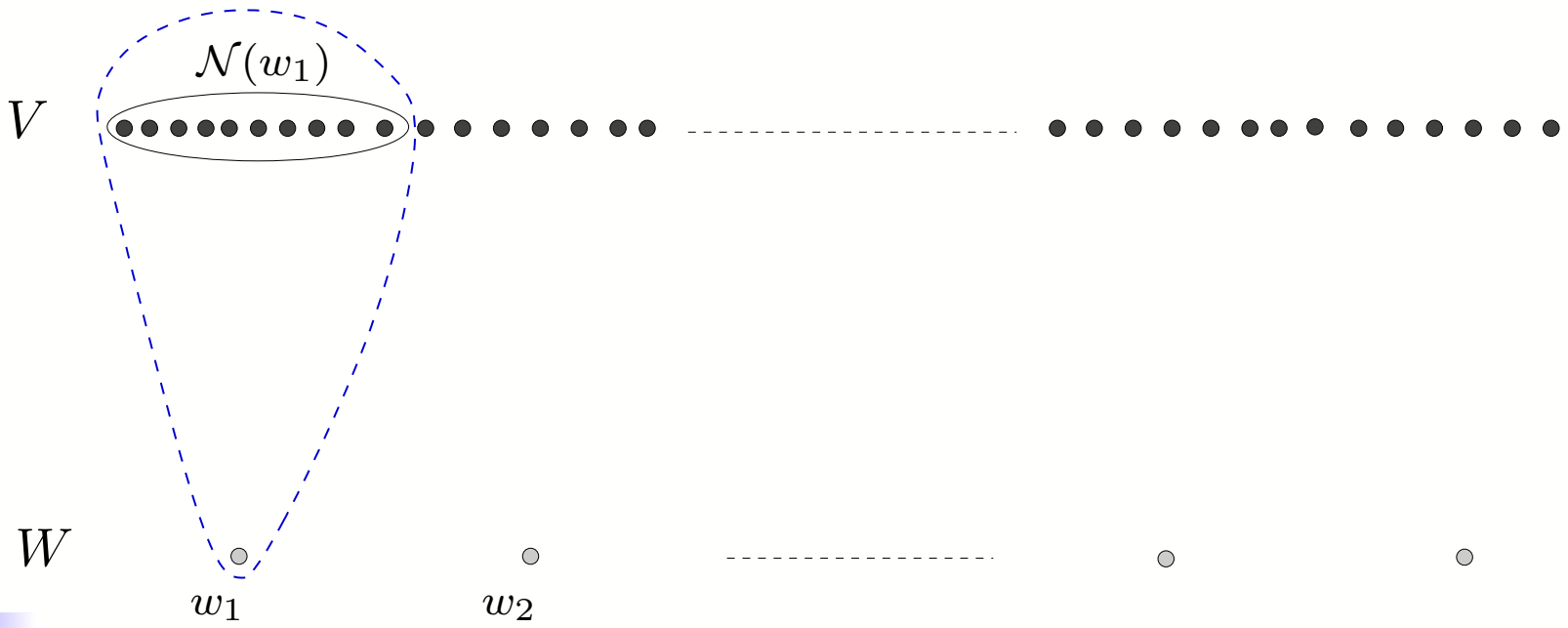
1. chose the highest degree node w in W
2. add the star $\mathcal{N}(w)$ to the spanner
3. connect w to other stars

3-spanner for Bipartite Graphs

Lemma 2 : Every bipartite graph $B = (W \cup V, E)$ has a 3-spanner with $O(|V| + |W|\sqrt{|V|})$ edges.

Algorithm : Repeat

1. chose the highest degree node w in W
2. add the star $\mathcal{N}(w)$ to the spanner
3. connect w to other stars

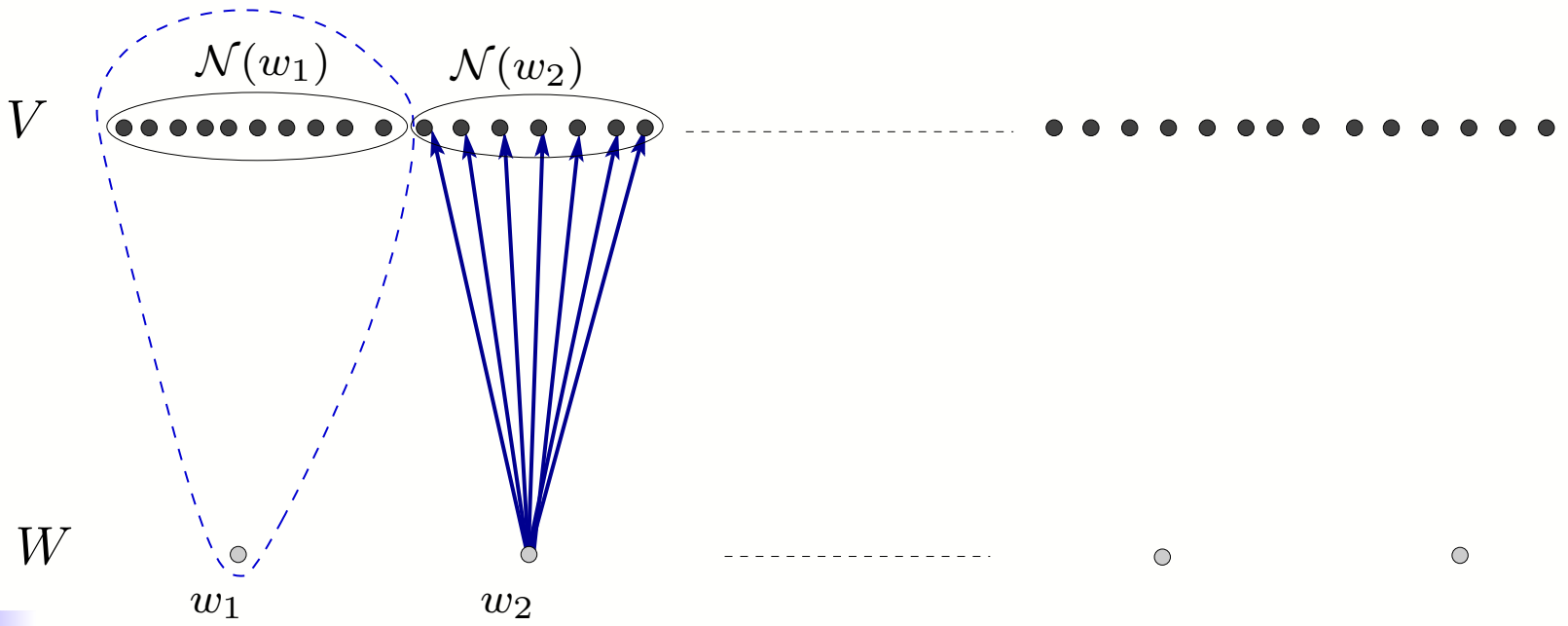


3-spanner for Bipartite Graphs

Lemma 2 : Every bipartite graph $B = (W \cup V, E)$ has a 3-spanner with $O(|V| + |W|\sqrt{|V|})$ edges.

Algorithm : Repeat

1. chose the highest degree node w in W
2. add the star $\mathcal{N}(w)$ to the spanner
3. connect w to other stars

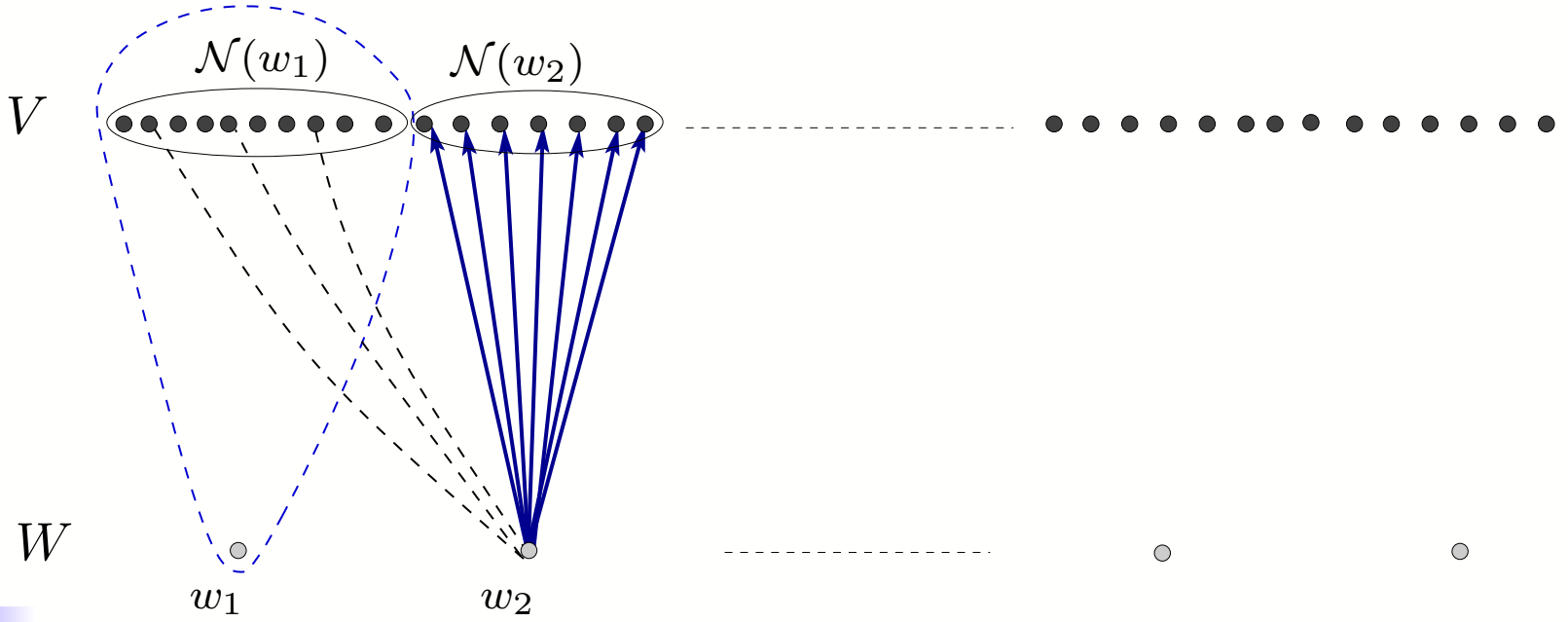


3-spanner for Bipartite Graphs

Lemma 2 : Every bipartite graph $B = (W \cup V, E)$ has a 3-spanner with $O(|V| + |W|\sqrt{|V|})$ edges.

Algorithm : Repeat

1. chose the highest degree node w in W
2. add the star $\mathcal{N}(w)$ to the spanner
3. connect w to other stars

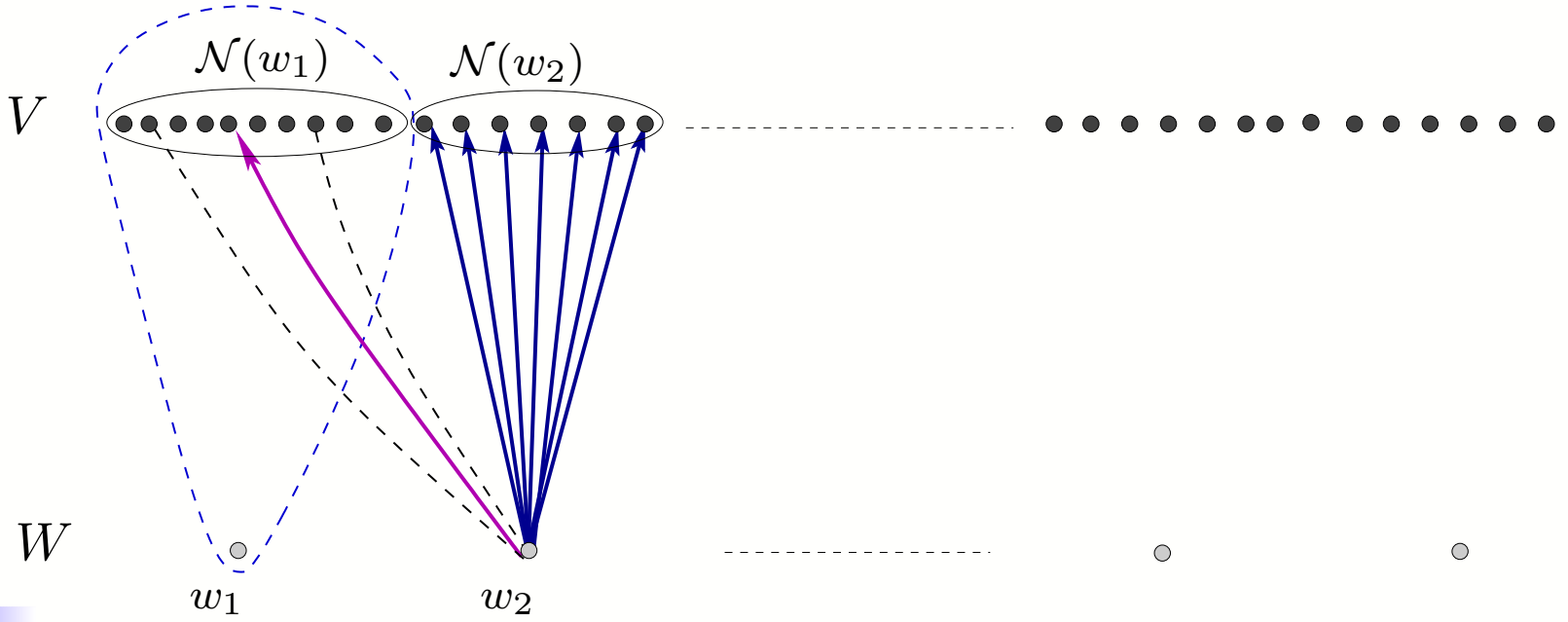


3-spanner for Bipartite Graphs

Lemma 2 : Every bipartite graph $B = (W \cup V, E)$ has a 3-spanner with $O(|V| + |W|\sqrt{|V|})$ edges.

Algorithm : Repeat

1. chose the highest degree node w in W
2. add the star $\mathcal{N}(w)$ to the spanner
3. connect w to other stars

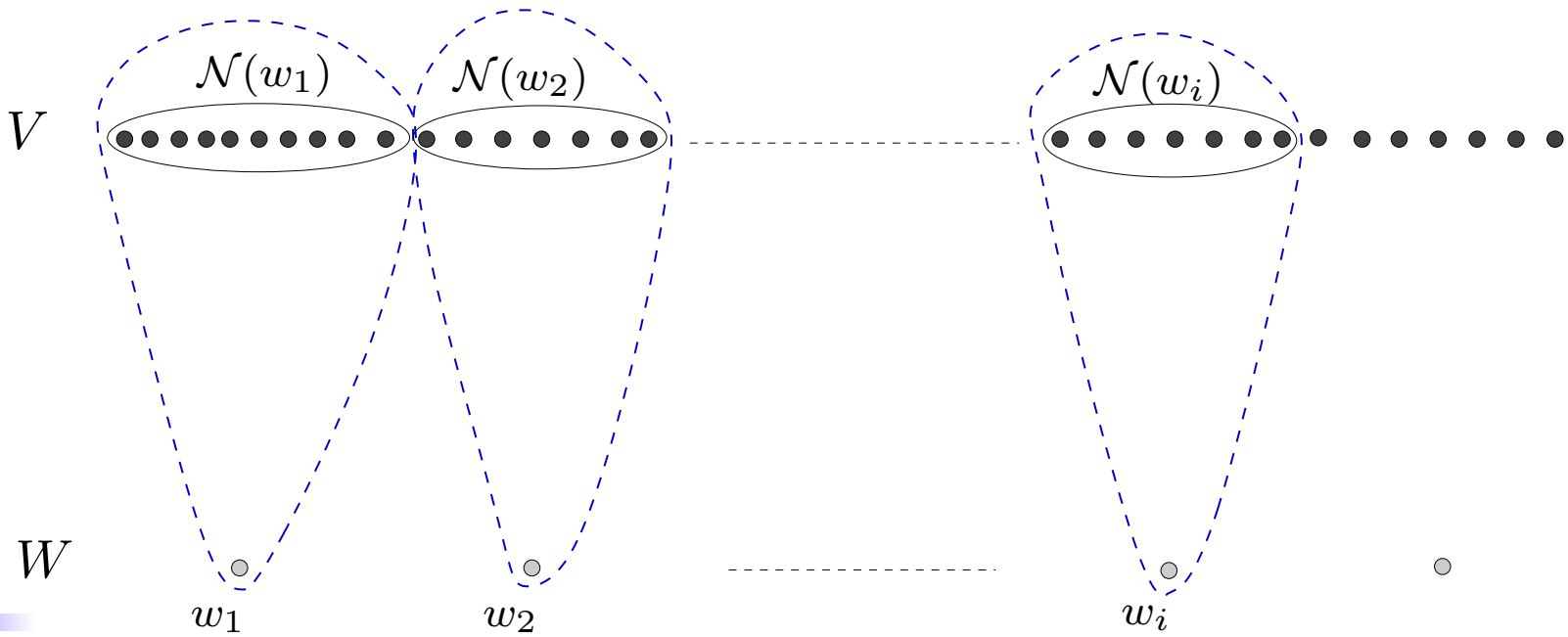


3-spanner for Bipartite Graphs

Lemma 2 : Every bipartite graph $B = (W \cup V, E)$ has a 3-spanner with $O(|V| + |W|\sqrt{|V|})$ edges.

Algorithm : Repeat

1. chose the highest degree node w in W
2. add the star $\mathcal{N}(w)$ to the spanner
3. connect w to other stars

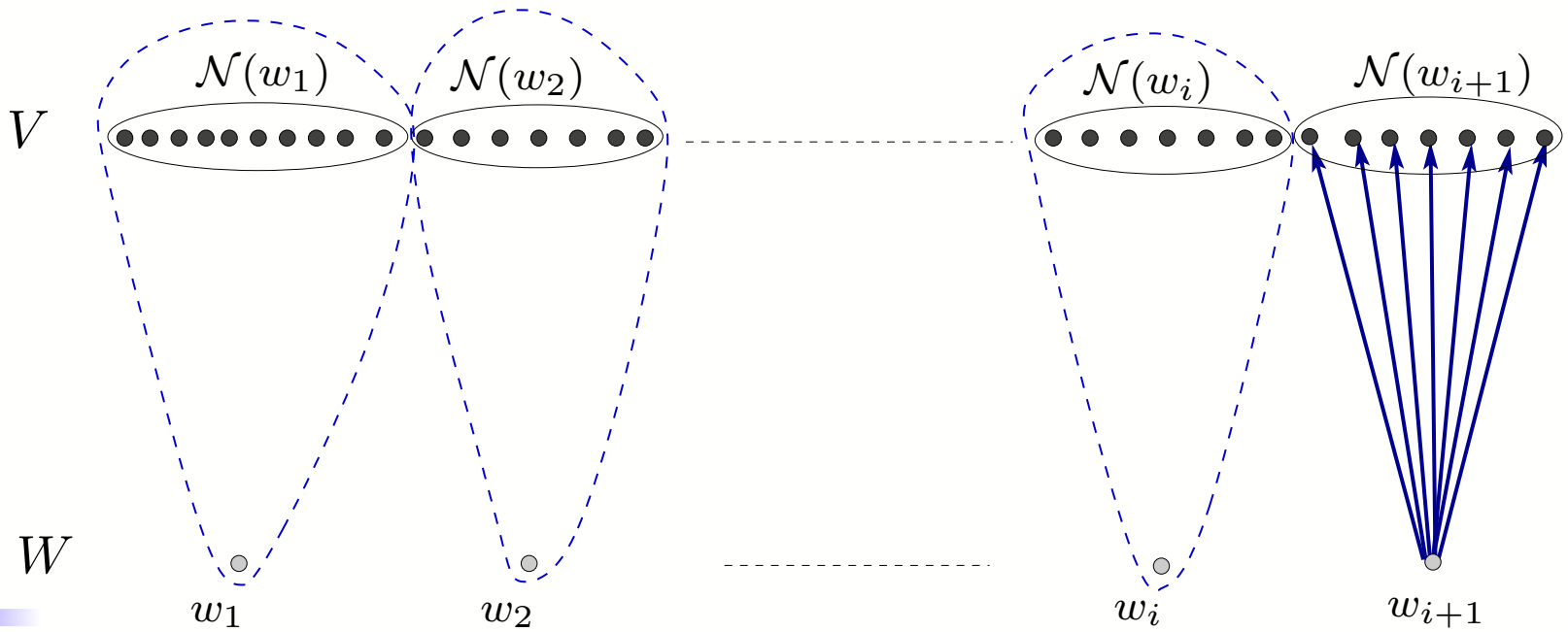


3-spanner for Bipartite Graphs

Lemma 2 : Every bipartite graph $B = (W \cup V, E)$ has a 3-spanner with $O(|V| + |W|\sqrt{|V|})$ edges.

Algorithm : Repeat

1. chose the highest degree node w in W
2. add the star $\mathcal{N}(w)$ to the spanner
3. connect w to other stars

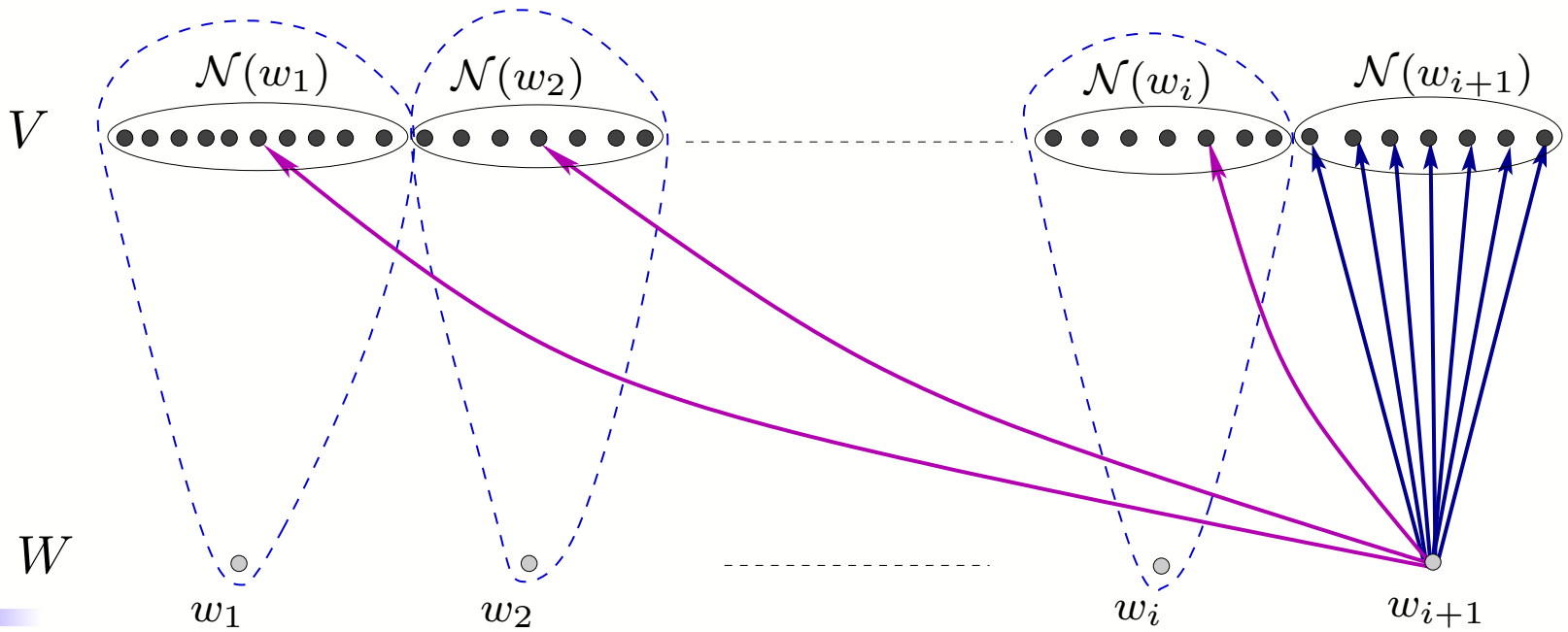


3-spanner for Bipartite Graphs

Lemma 2 : Every bipartite graph $B = (W \cup V, E)$ has a 3-spanner with $O(|V| + |W|\sqrt{|V|})$ edges.

Algorithm : Repeat

1. chose the highest degree node w in W
2. add the star $\mathcal{N}(w)$ to the spanner
3. connect w to other stars

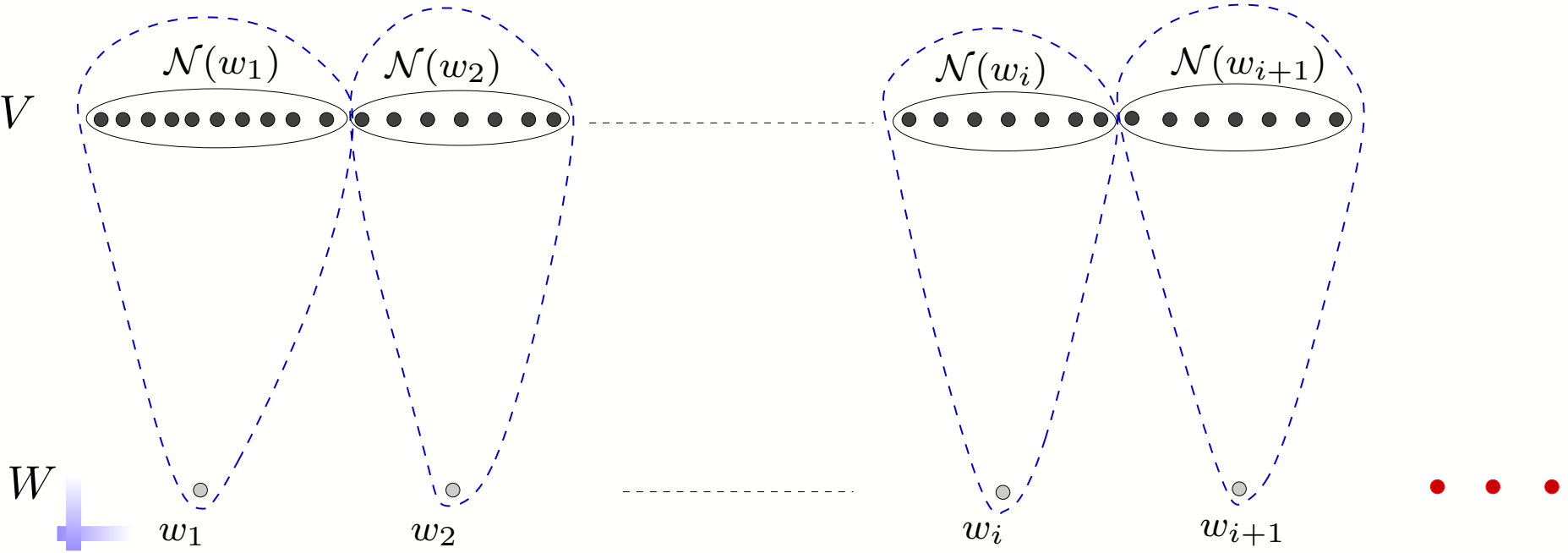


3-spanner for Bipartite Graphs

Lemma 2 : Every bipartite graph $B = (W \cup V, E)$ has a 3-spanner with $O(|V| + |W|\sqrt{|V|})$ edges.

Algorithm : Repeat

1. chose the highest degree node w in W
2. add the star $\mathcal{N}(w)$ to the spanner
3. connect w to other stars

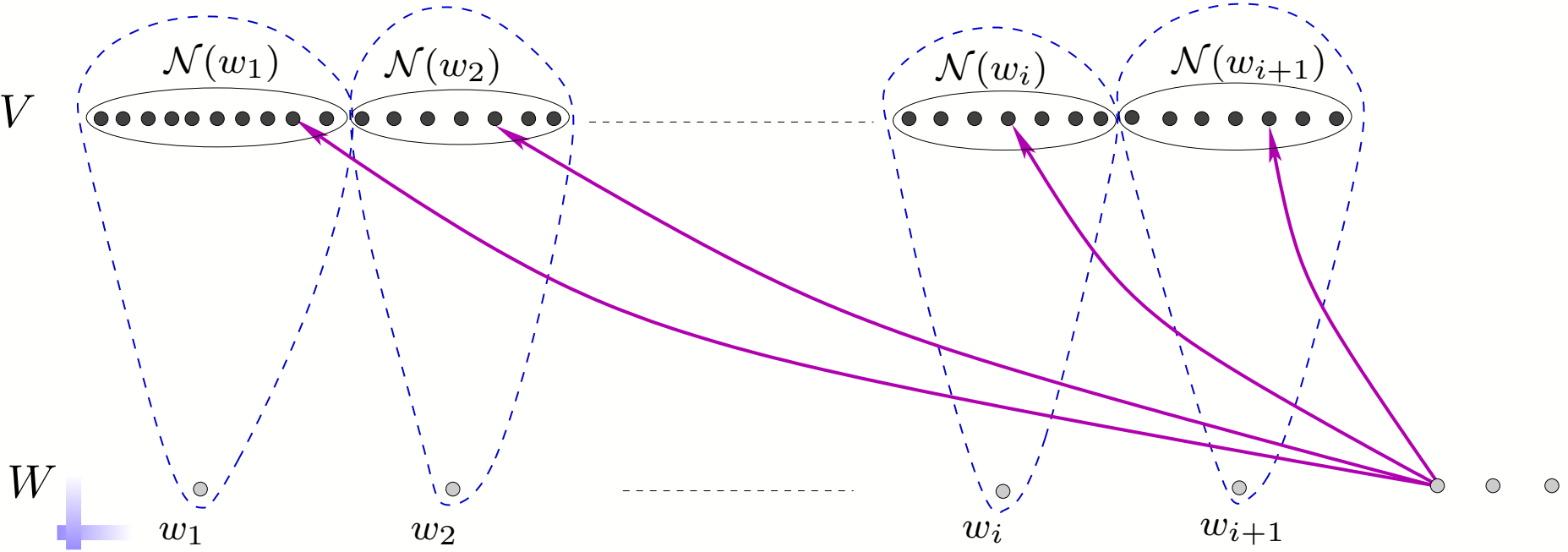


3-spanner for Bipartite Graphs

Lemma 2 : Every bipartite graph $B = (W \cup V, E)$ has a 3-spanner with $O(|V| + |W|\sqrt{|V|})$ edges.

Algorithm : Repeat

1. chose the highest degree node w in W
2. add the star $\mathcal{N}(w)$ to the spanner
3. connect w to other stars



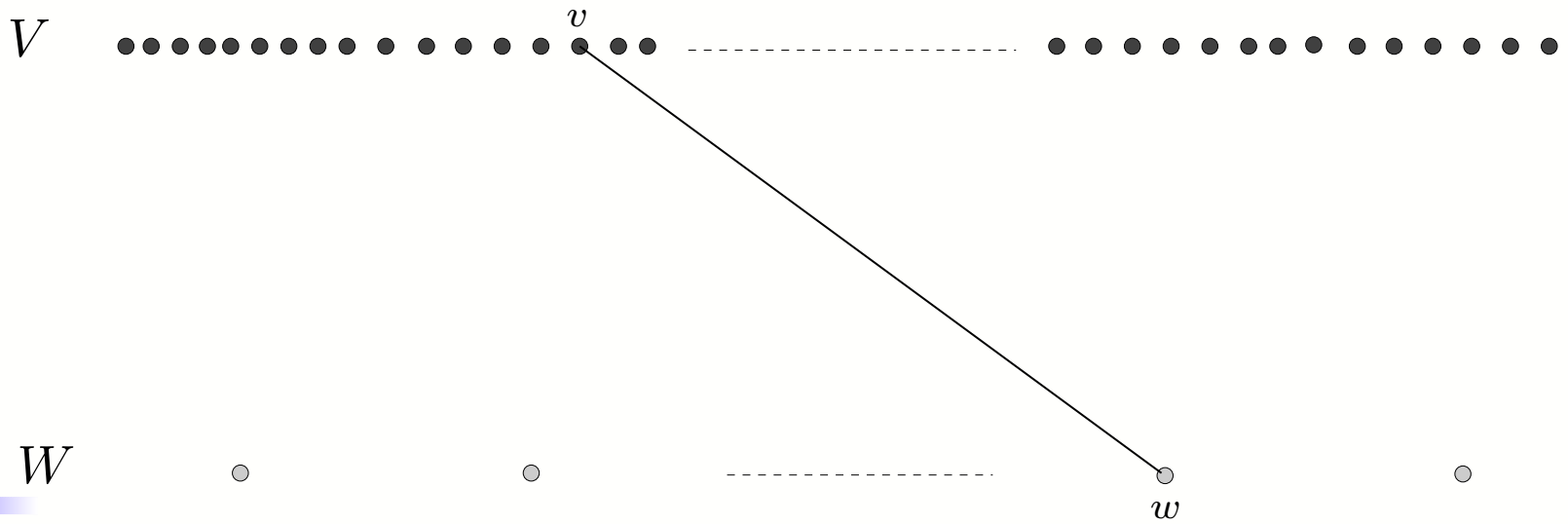
3-spanner for Bipartite Graphs

Lemma 2 : Every bipartite graph $B = (W \cup V, E)$ has a 3-spanner with $O(|V| + |W|\sqrt{|V|})$ edges.

Algorithm : Repeat

1. chose the highest degree node w in W
2. add the star $\mathcal{N}(w)$ to the spanner
3. connect w to other stars

Stretch Analysis



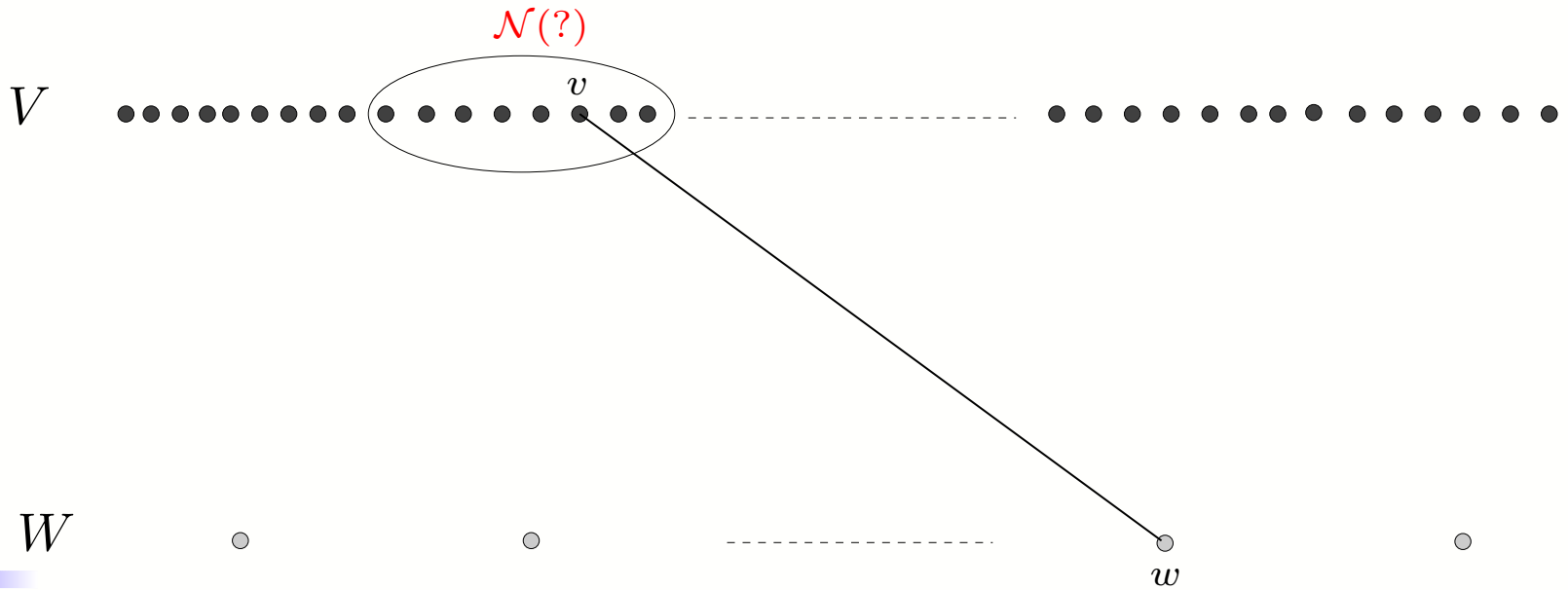
3-spanner for Bipartite Graphs

Lemma 2 : Every bipartite graph $B = (W \cup V, E)$ has a 3-spanner with $O(|V| + |W|\sqrt{|V|})$ edges.

Algorithm : Repeat

1. chose the highest degree node w in W
2. add the star $\mathcal{N}(w)$ to the spanner
3. connect w to other stars

Stretch Analysis



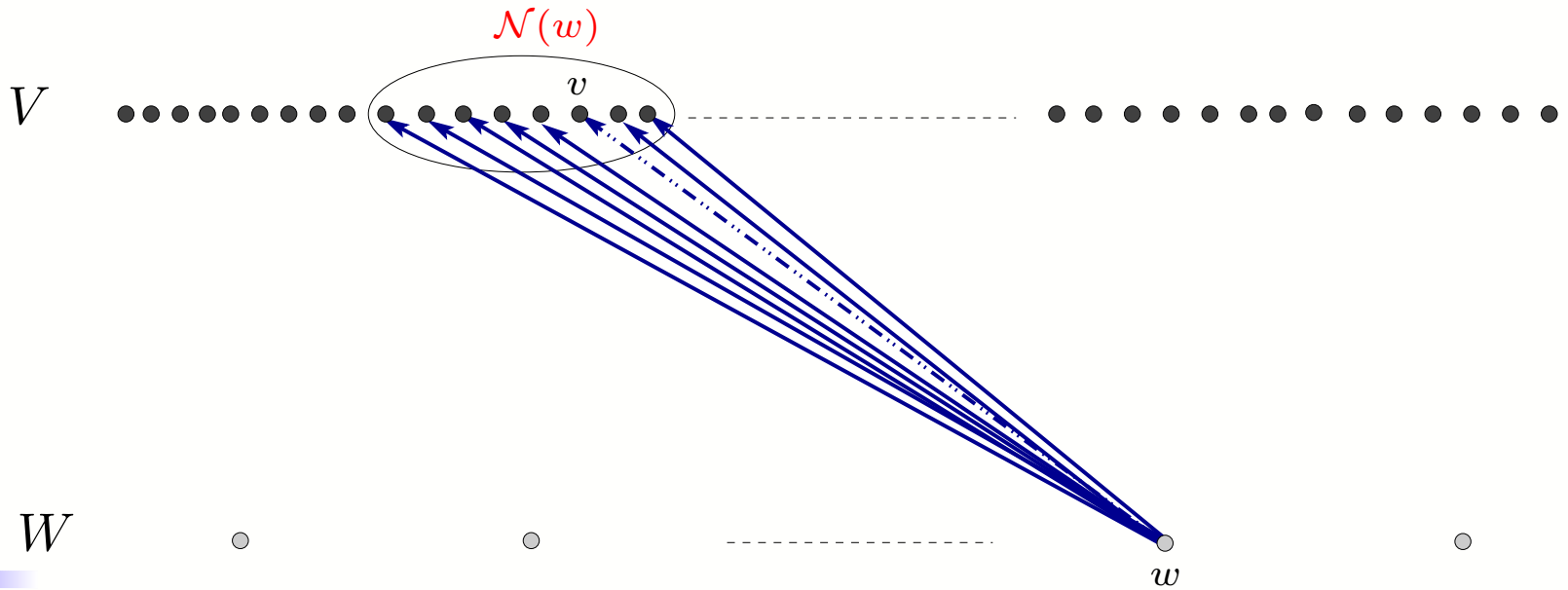
3-spanner for Bipartite Graphs

Lemma 2 : Every bipartite graph $B = (W \cup V, E)$ has a 3-spanner with $O(|V| + |W|\sqrt{|V|})$ edges.

Algorithm : Repeat

1. chose the highest degree node w in W
2. add the star $\mathcal{N}(w)$ to the spanner
3. connect w to other stars

Stretch Analysis



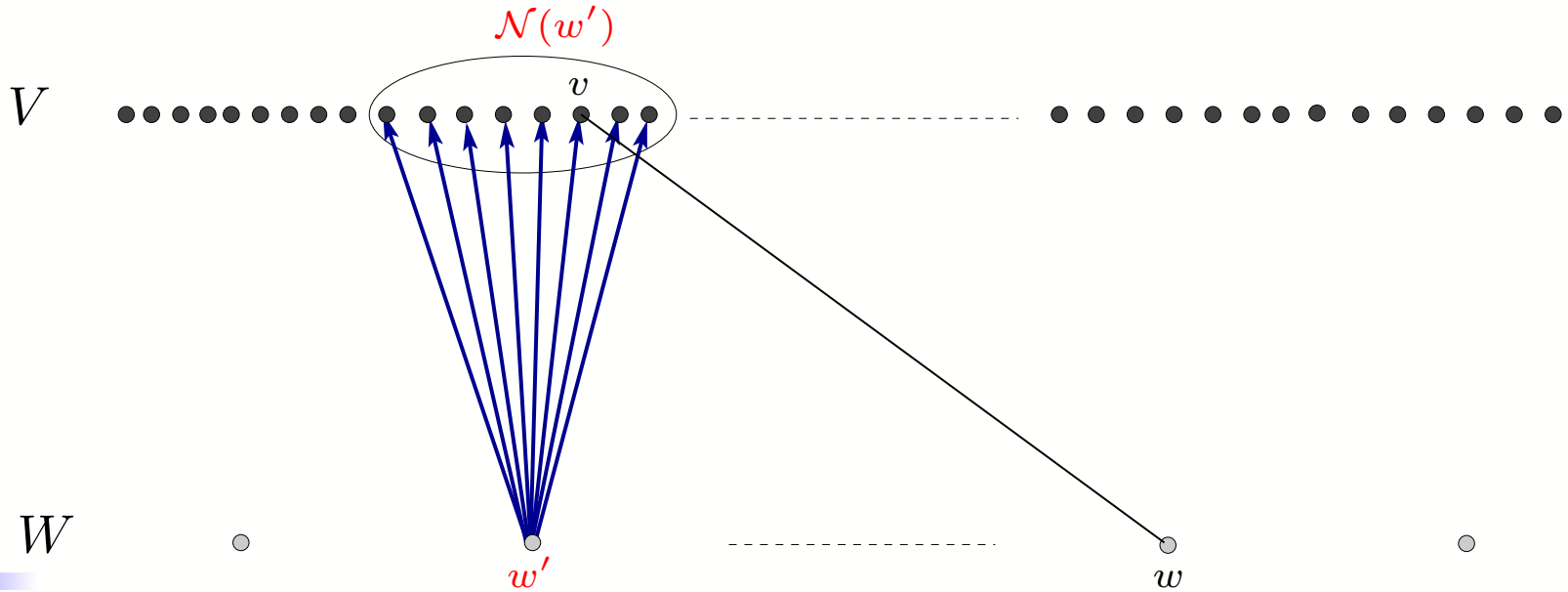
3-spanner for Bipartite Graphs

Lemma 2 : Every bipartite graph $B = (W \cup V, E)$ has a 3-spanner with $O(|V| + |W|\sqrt{|V|})$ edges.

Algorithm : Repeat

1. chose the highest degree node w in W
2. add the star $\mathcal{N}(w)$ to the spanner
3. connect w to other stars

Stretch Analysis



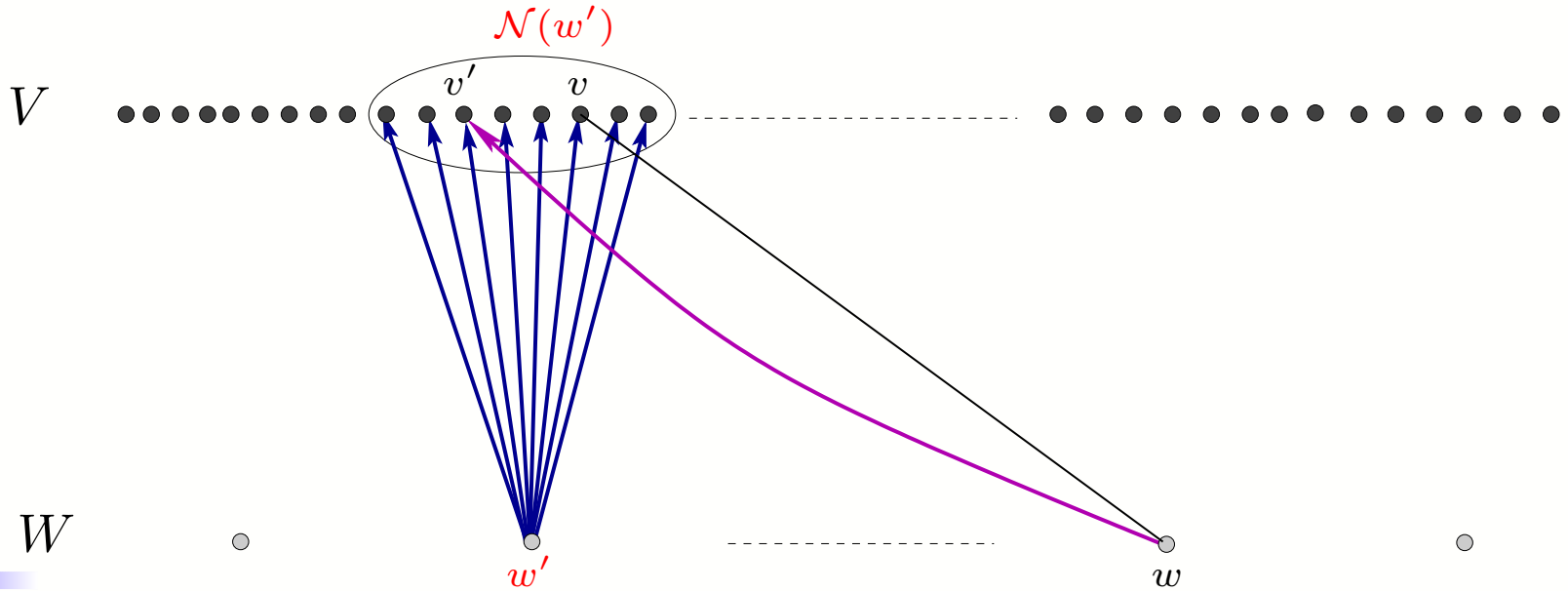
3-spanner for Bipartite Graphs

Lemma 2 : Every bipartite graph $B = (W \cup V, E)$ has a 3-spanner with $O(|V| + |W|\sqrt{|V|})$ edges.

Algorithm : Repeat

1. chose the highest degree node w in W
2. add the star $\mathcal{N}(w)$ to the spanner
3. connect w to other stars

Stretch Analysis



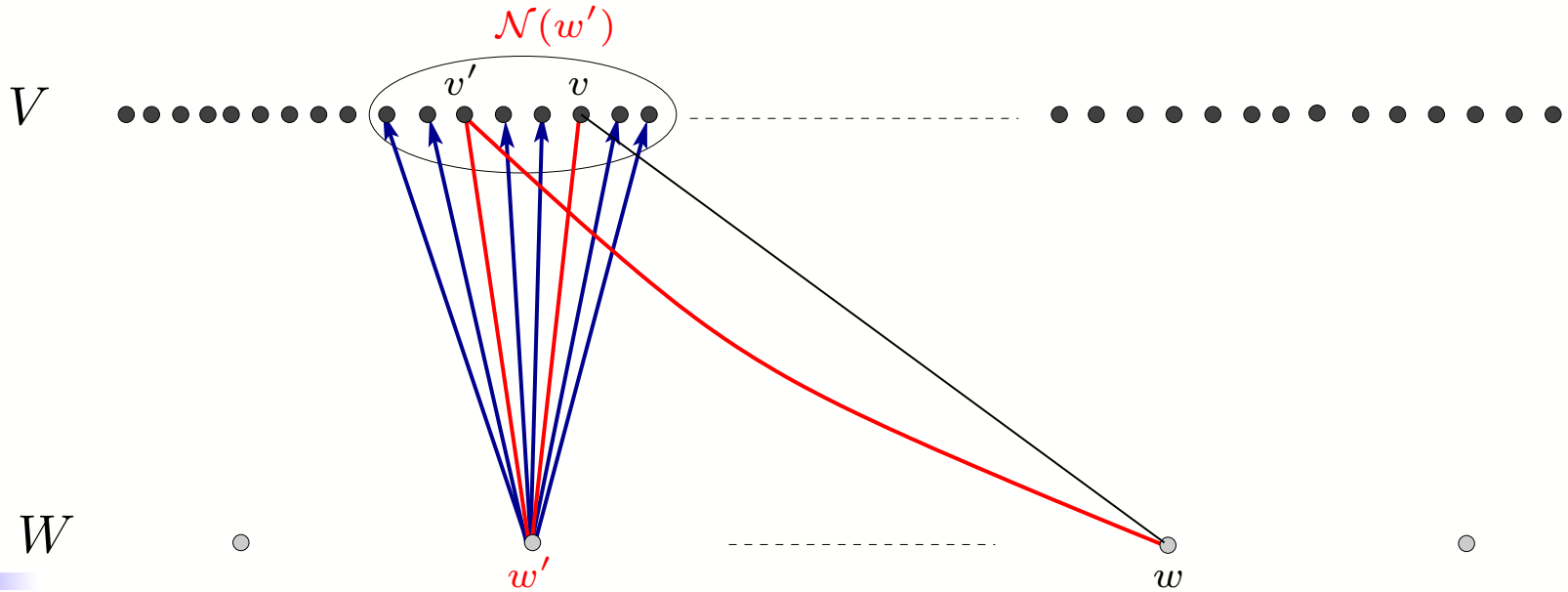
3-spanner for Bipartite Graphs

Lemma 2 : Every bipartite graph $B = (W \cup V, E)$ has a 3-spanner with $O(|V| + |W|\sqrt{|V|})$ edges.

Algorithm : Repeat

1. chose the highest degree node w in W
2. add the star $\mathcal{N}(w)$ to the spanner
3. connect w to other stars

Stretch Analysis



3-spanner for Bipartite Graphs

Lemma 2 : Every bipartite graph $B = (W \cup V, E)$ has a 3-spanner with $O(|V| + |W|\sqrt{|V|})$ edges.

Algorithm : Repeat

1. chose the highest degree node w in W
2. add the star $\mathcal{N}(w)$ to the spanner
3. connect w to other stars

Stretch Analysis

Stretch ≤ 3

$$\forall v, w \in V \cup W, \quad d_S(v,w) \leq \begin{cases} 2 \cdot d_B(v,w) + 1 & \text{if } d_B(v,w) \text{ is odd} \\ 2 \cdot d_B(v,w) + 2 & \text{otherwise.} \end{cases}$$

3-spanner for Bipartite Graphs

Lemma 2 : Every bipartite graph $B = (W \cup V, E)$ has a 3-spanner with $O(|V| + |W|\sqrt{|V|})$ edges.

Algorithm : Repeat

1. chose the highest degree node w in W
2. add the star $\mathcal{N}(w)$ to the spanner
3. connect w to other stars

Stretch Analysis

Stretch ≤ 3

Size Analysis

- at most $|V|$ edges added by the stars
- Each node in $w \in W$ is connected to the previous stars by at most $2\sqrt{|V|}$ edges (proof by contradiction).

3-spanner for Bipartite Graphs

Lemma 2 : Every bipartite graph $B = (W \cup V, E)$ has a 3-spanner with $O(|V| + |W|\sqrt{|V|})$ edges.

Algorithm : Repeat

1. chose the highest degree node w in W
2. add the star $\mathcal{N}(w)$ to the spanner
3. connect w to other stars

Stretch Analysis

Stretch ≤ 3

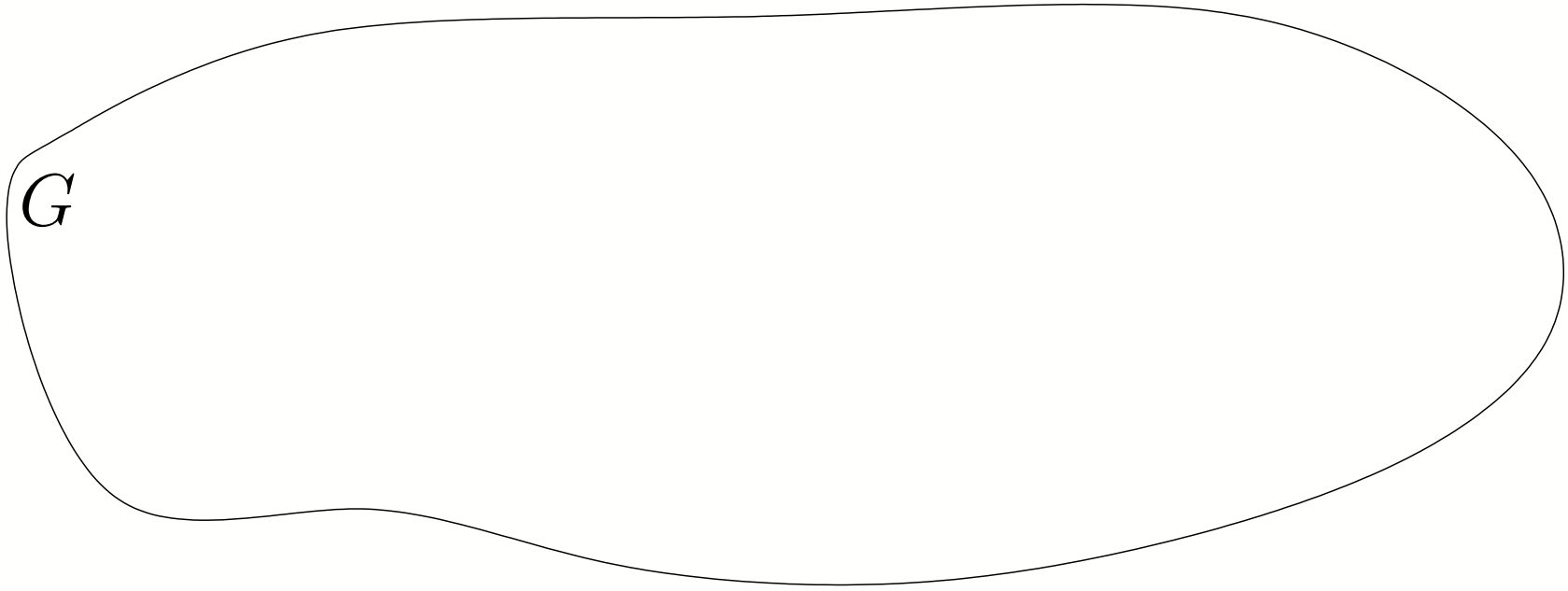
Size Analysis

- at most $|V|$ edges added by the stars
- Each node in $w \in W$ is connected to the previous stars by at most $2\sqrt{|V|}$ edges (proof by contradiction).

Size $= O(|V| + |W| \cdot \sqrt{|V|})$

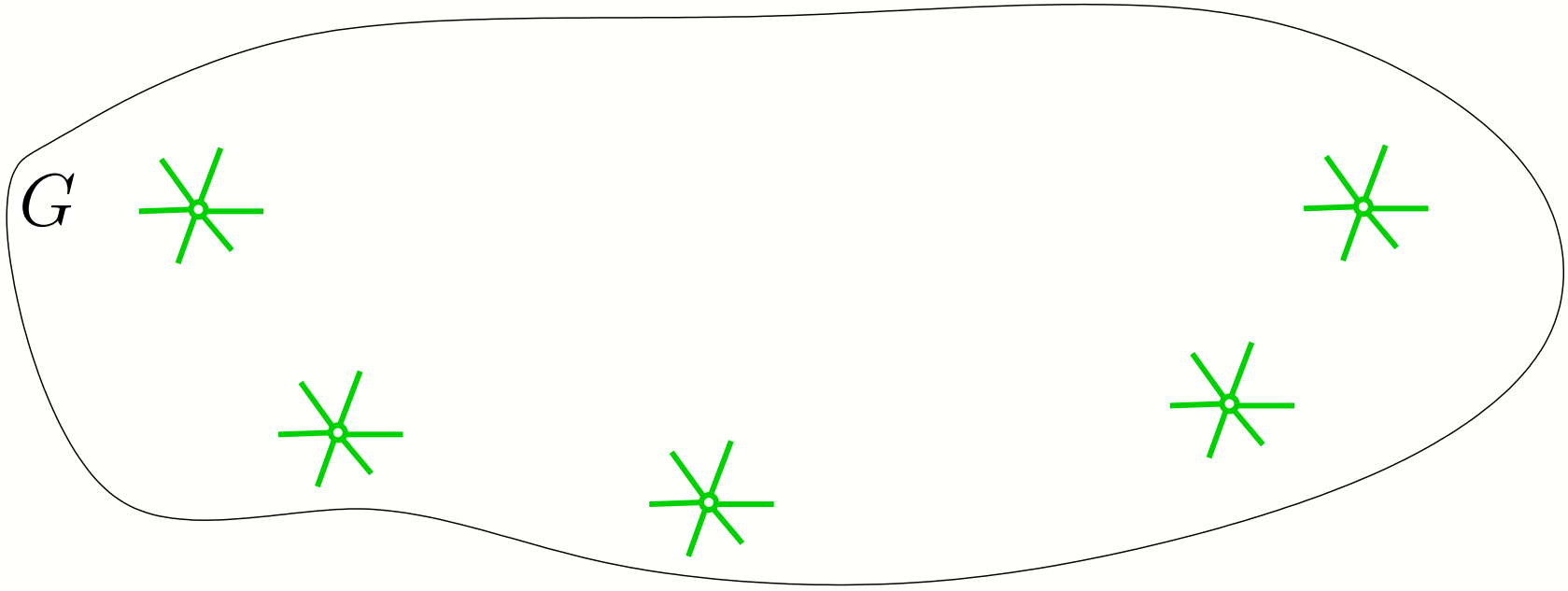
The algorithm : $k = 2$

- If the degree of a node v is at most $\sqrt{n}$, then add the star $\mathcal{N}(v)$ to the spanner and delete v from G.



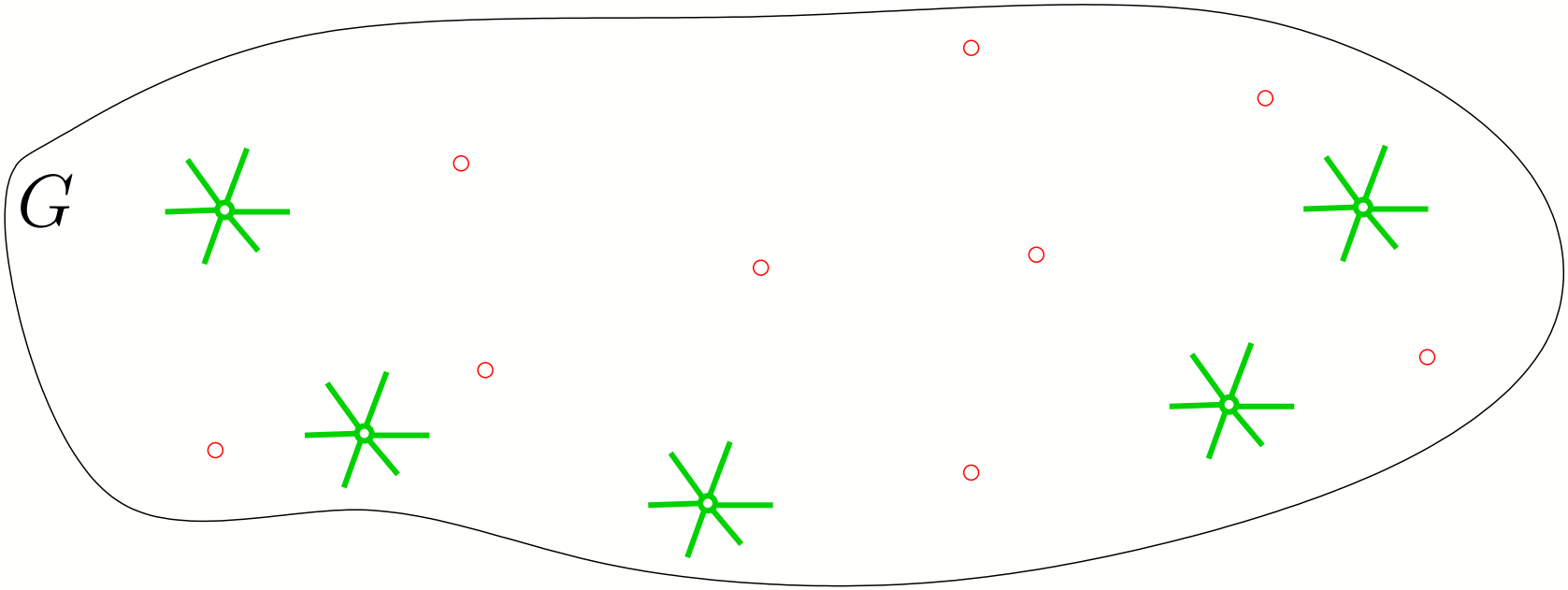
The algorithm : $k = 2$

- If the degree of a node v is at most $\sqrt{n}$, then add the star $\mathcal{N}(v)$ to the spanner and delete v from G.



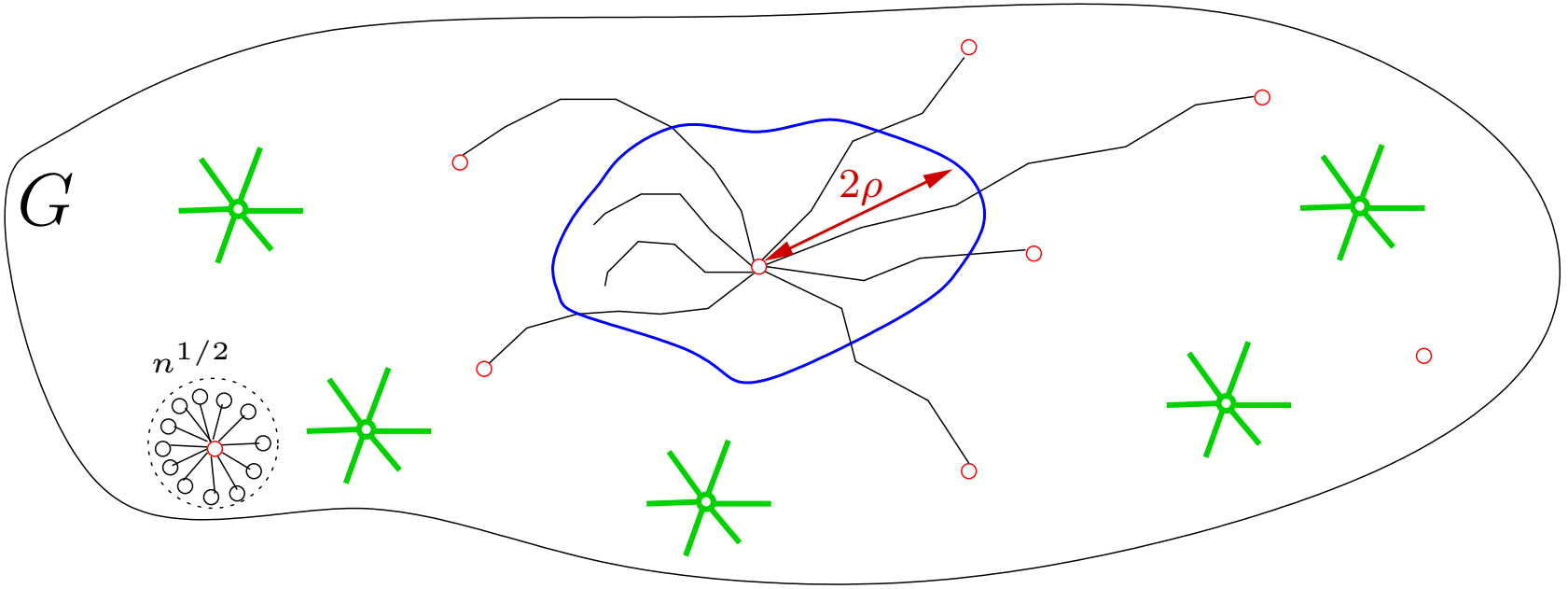
The algorithm : $k = 2$

- If the degree of a node v is at most $\sqrt{n}$, then add the star $\mathcal{N}(v)$ to the spanner and delete v from G.
- Find an independent ρ-dominating set X of G^2.



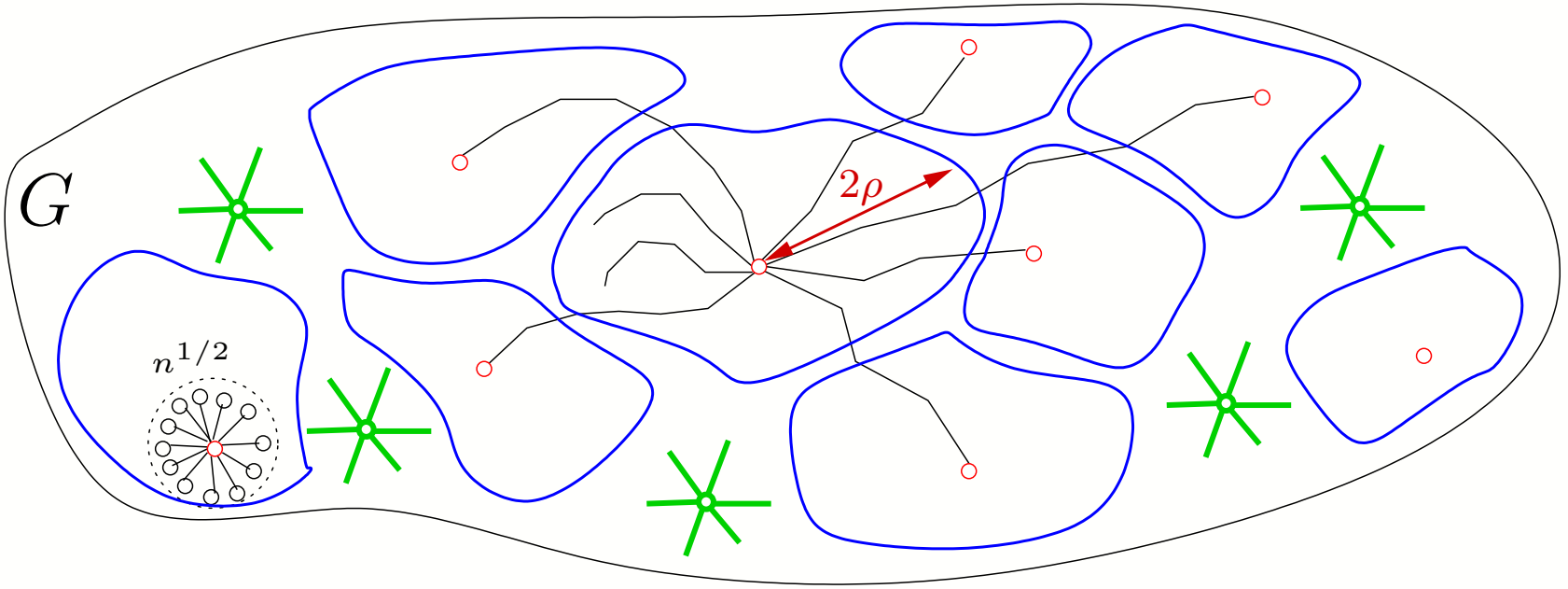
The algorithm : $k = 2$

- If the degree of a node v is at most $\sqrt{n}$, then add the star $\mathcal{N}(v)$ to the spanner and delete v from G.
- Find an independent ρ-dominating set X of G^2.
- Form a region $R(v)$ around each node v of X.
  - There are at most $\sqrt{n}$ disjoint regions
  - The radius of each region is at most $O(\rho)$



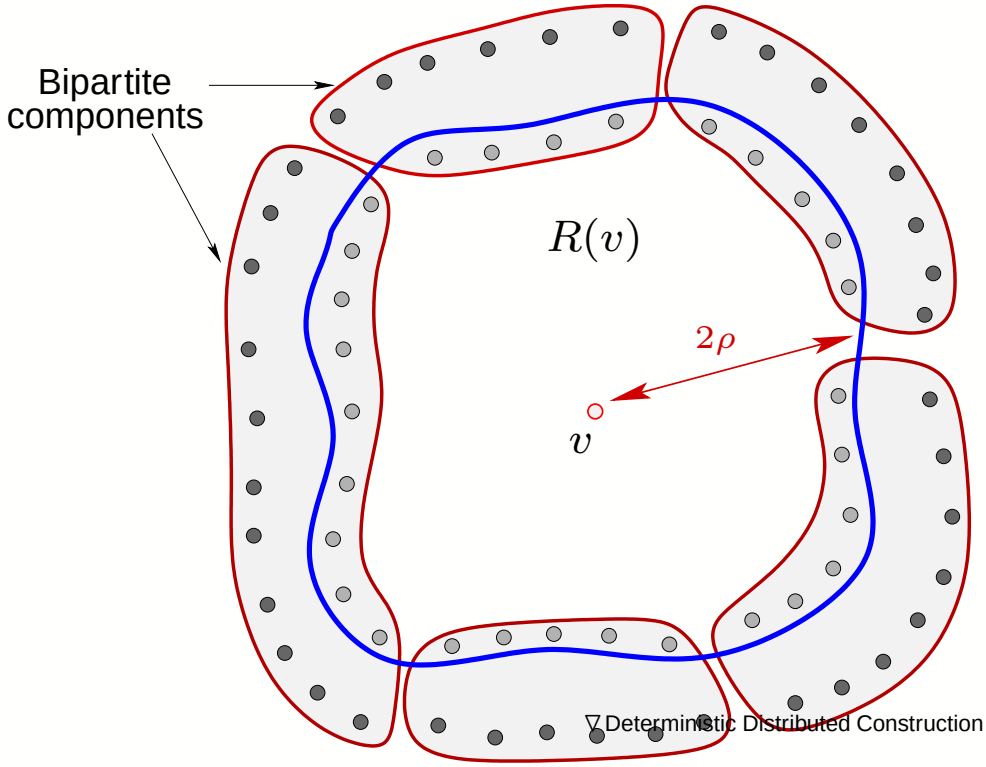
The algorithm : $k = 2$

- If the degree of a node v is at most $\sqrt{n}$, then add the star $\mathcal{N}(v)$ to the spanner and delete v from G.
- Find an independent ρ-dominating set X of G^2.
- Form a region $R(v)$ around each node v of X.
  - There are at most $\sqrt{n}$ disjoint regions
  - The radius of each region is at most $O(\rho)$



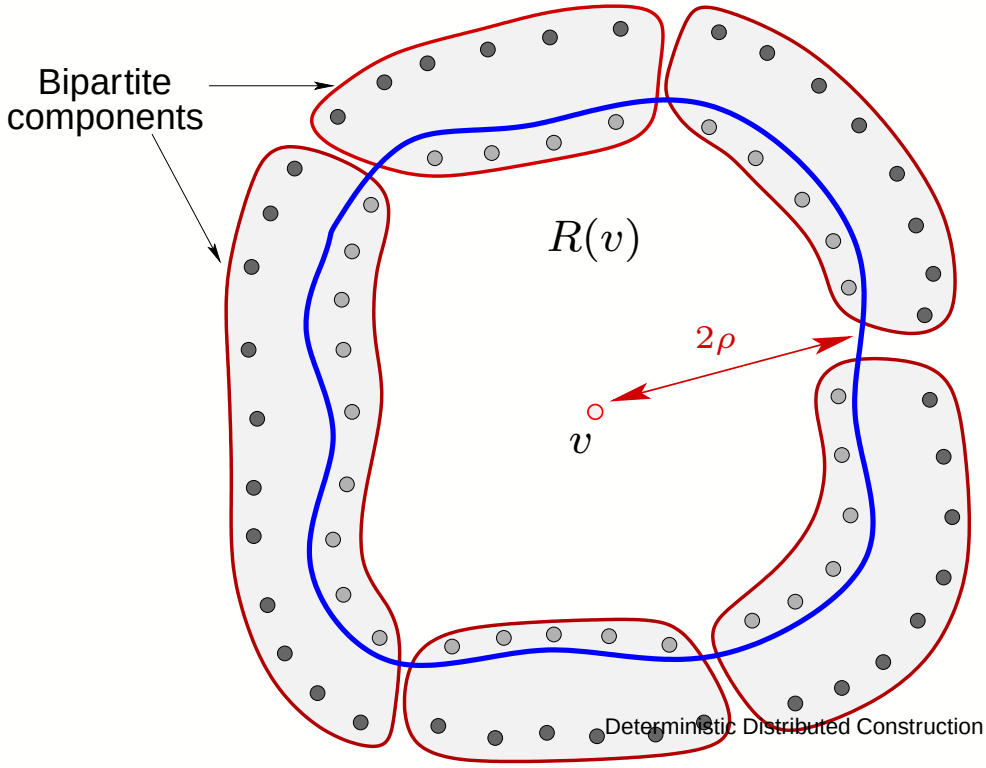
The algorithm : $k = 2$

- If the degree of a node v is at most $\sqrt{n}$, then add the star $\mathcal{N}(v)$ to the spanner and delete v from G.
- Find an independent ρ-dominating set X of G^2.
- Form a region $R(v)$ around each node v of X.
  - There are at most $\sqrt{n}$ disjoint regions
  - The radius of each region is at most $O(\rho)$



The algorithm : $k = 2$

- If the degree of a node v is at most $\sqrt{n}$, then add the star $\mathcal{N}(v)$ to the spanner and delete v from G.
- Find an independent ρ-dominating set X of G^2.
- Form a region $R(v)$ around each node v of X.
- Span the interior of a region $R(v)$ using Lemma 1.
- Span the border of a region $R(v)$ using Lemma 2.



Analysis : $k = 2$

Size analysis :

Analysis : $k = 2$

Size analysis :

- Spanning the sparse stars :
  - number of edges $\leq n \cdot \sqrt{n}$

Analysis : $k = 2$

Size analysis :

- Spanning the sparse stars :
  - number of edges $\leq n \cdot \sqrt{n}$
- Spanning the interior of the regions (Lemma 1) :
  - number of edges added per region $\leq n_v \cdot \sqrt{n_v} \leq n_v \cdot \sqrt{n}$

  $\Longrightarrow$ Sum over the (disjoint) regions $\leq n \cdot \sqrt{n}$

Analysis : $k = 2$

Size analysis :

- Spanning the sparse stars :
  - number of edges $\leq n \cdot \sqrt{n}$
- Spanning the interior of the regions (Lemma 1) :
  - number of edges added per region $\leq n_v \cdot \sqrt{n_v} \leq n_v \cdot \sqrt{n}$

  $\Longrightarrow$ Sum over the (disjoint) regions $\leq n \cdot \sqrt{n}$
- Spanning the border of the regions (Lemma 2) :
  - number of edges added per region $\leq n + n_v \cdot \sqrt{n}$

  $\Longrightarrow$ Sum over the (disjoint) $\sqrt{n}$ regions $\leq \sqrt{n} \cdot n + n \cdot \sqrt{n}$

Analysis : $k = 2$

Size analysis :

- Spanning the sparse stars :
  - number of edges $\leq n \cdot \sqrt{n}$
- Spanning the interior of the regions (Lemma 1) :
  - number of edges added per region $\leq n_v \cdot \sqrt{n_v} \leq n_v \cdot \sqrt{n}$

  $\Longrightarrow$ Sum over the (disjoint) regions $\leq n \cdot \sqrt{n}$
- Spanning the border of the regions (Lemma 2) :
  - number of edges added per region $\leq n + n_v \cdot \sqrt{n}$

  $\Longrightarrow$ Sum over the (disjoint) $\sqrt{n}$ regions $\leq \sqrt{n} \cdot n + n \cdot \sqrt{n}$

at most $O(n^{3/2})$ edges

Analysis : $k = 2$

Stretch analysis :

- Every edge (u, v) is either :
  - incident to a sparse star
  - inside a region (Lemma 1)
  - in the border of a region (Lemma 2)

$$\text{stretch} \leq 3$$

Analysis : $k = 2$

Stretch analysis :

- Every edge (u, v) is either :
  - incident to a sparse star
  - inside a region (Lemma 1)
  - in the border of a region (Lemma 2)

$$\text{stretch} \leq 3$$

Time analysis :

- Independent ρ dominating set $\Longrightarrow \mathrm{IDS}(n, \rho)$ time
- spanning the interior and the border of regions $\Longrightarrow O(\rho)$ time.

$$\text{Time} = O(\mathrm{IDS}(n, \rho) + \rho)$$

Analysis : $k = 2$

Stretch analysis :

- Every edge (u, v) is either :
  - incident to a sparse star
  - inside a region (Lemma 1)
  - in the border of a region (Lemma 2)

$$\text{stretch} \leq 3$$

Time analysis :

- Independent ρ dominating set $\Longrightarrow \mathrm{IDS}(n, \rho)$ time
- spanning the interior and the border of regions $\Longrightarrow O(\rho)$ time.

$$\text{Time} = O(\mathrm{IDS}(n, \rho) + \rho)$$

$\rho = O(\log n) \longrightarrow$ There exists a deterministic distributed algorithm computing an independent $(\log n)$-dominating set in $O(\log n)$ time.

Results

Theorem 1 : There exists a distributed algorithm that given an n-node graph constructs a 3-spanner with $O(n^{3/2})$ edges in $O(\log n)$ deterministic time.

Results

Theorem 1 : There exists a distributed algorithm that given an n-node graph constructs a 3-spanner with $O(n^{3/2})$ edges in $O(\log n)$ deterministic time.

Lemma 3 : Every bipartite graph $B = (W \cup V, E)$ has a $(4k-5)$-spanner with $O(|V+W| + |W|\sqrt{|V+W|})$ edges.

Theorem 2 : There exists a distributed algorithm that given an n-node graph constructs a $(4k-5)$-spanner with $O(k \cdot n^{1+1/k})$ edges in $2^{O(k)} \log^{k-1} n$ deterministic time.

Almost pure additive spanners

Existing results

	(α, β)	size	time	expected time
[EP'04]	$(1, 2)$	$n^{3/2}$	?	?
[BKMP'05]	$(1, 6)$	$n^{4/3}$	?	?
[EP'04,EZ'04]*	$(1+\epsilon, \beta)$	$n^{1+\delta}$	$O(n^{1+\delta})$	?

$* : \beta = \beta(\delta^{-1}, \epsilon^{-1})$

Almost pure additive spanners

Existing results

	(α, β)	size	time	expected time
[EP'04]	$(1, 2)$	$n^{3/2}$	?	?
[BKMP'05]	$(1, 6)$	$n^{4/3}$	?	?
[EP'04,EZ'04]*	$(1+\epsilon, \beta)$	$n^{1+\delta}$	$O(n^{1+\delta})$	?

$* : \beta = \beta(\delta^{-1}, \epsilon^{-1})$

This paper

(α, β)	size	time	expected time
$(1+\epsilon, 4)$	$n^{3/2}$	$n^{O(1/\sqrt{\log n})} + O(1/\epsilon)$	$O(\log n + 1/\epsilon)$
$(1+\epsilon, 8\log n)$	$n^{3/2}$	$O(\log n/\epsilon)$	$--$

Almost pure additive spanners

Algorithm :

1. run our 3-spanner algorithm (parameter ρ) :
   - compute S : a 3-spanner with $O(n^{3/2})$ edges.
   - compute X : an independent ρ-dominating set of G^2.
2. $\forall v \in X$, add to S a BFS tree rooted at v up to distance $2\rho + \gamma$.

Almost pure additive spanners

Algorithm :

1. run our 3-spanner algorithm (parameter ρ) :
   - compute S : a 3-spanner with $O(n^{3/2})$ edges.
   - compute X : an independent ρ-dominating set of G^2.
2. $\forall v \in X$, add to S a BFS tree rooted at v up to distance $2\rho + \gamma$.

Size Analysis

$$O(n^{3/2}) + |X| \cdot n = O(n^{3/2})$$

Almost pure additive spanners

Algorithm :

1. run our 3-spanner algorithm (parameter ρ) :
   - compute S : a 3-spanner with $O(n^{3/2})$ edges.
   - compute X : an independent ρ-dominating set of G^2.
2. $\forall v \in X$, add to S a BFS tree rooted at v up to distance $2\rho + \gamma$.

Time Analysis

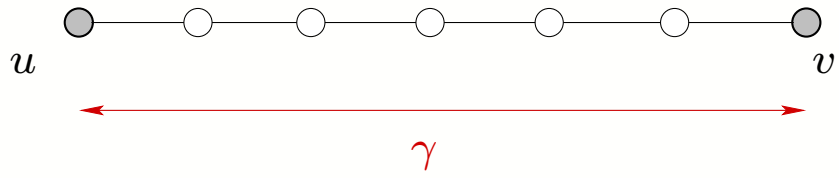
$$\mathrm{IDS}(n, \rho) + O(\rho + \gamma)$$

Almost pure additive spanners

Algorithm :

1. run our 3-spanner algorithm (parameter ρ) :
   - compute S : a 3-spanner with $O(n^{3/2})$ edges.
   - compute X : an independent ρ-dominating set of G^2.
2. $\forall v \in X$, add to S a BFS tree rooted at v up to distance $2\rho + \gamma$.

Stretch Analysis

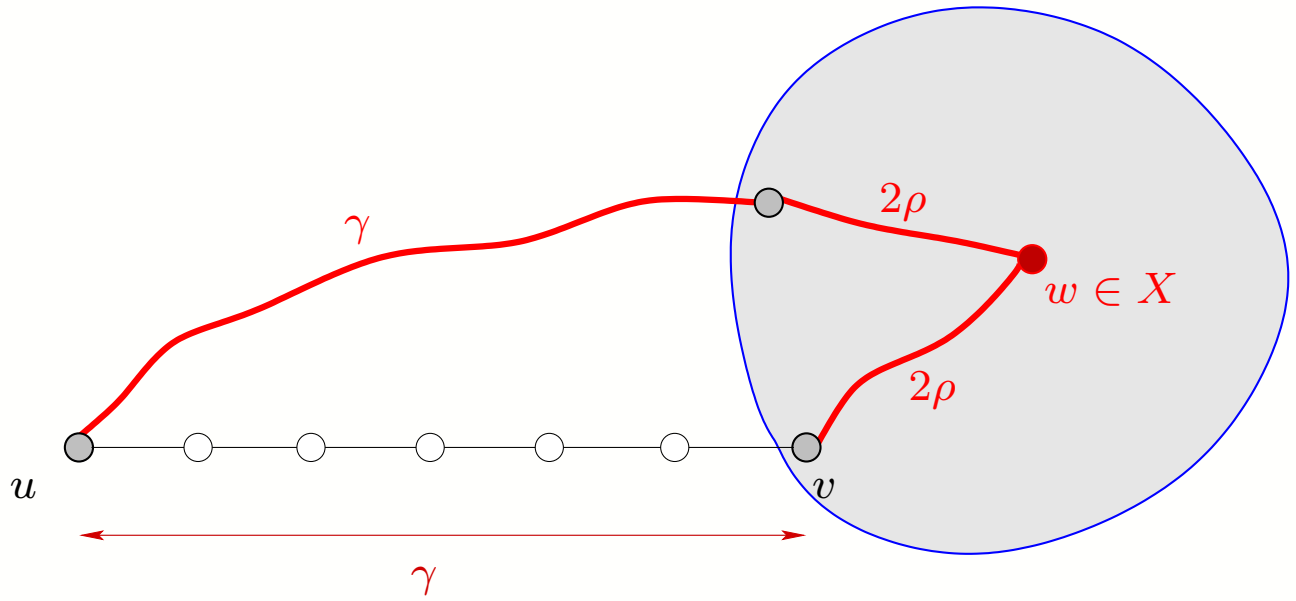


Almost pure additive spanners

Algorithm :

1. run our 3-spanner algorithm (parameter ρ) :
   - compute S : a 3-spanner with $O(n^{3/2})$ edges.
   - compute X : an independent ρ-dominating set of G^2.
2. $\forall v \in X$, add to S a BFS tree rooted at v up to distance $2\rho + \gamma$.

Stretch Analysis



Almost pure additive spanners

Algorithm :

1. run our 3-spanner algorithm (parameter ρ) :
   - compute S : a 3-spanner with $O(n^{3/2})$ edges.
   - compute X : an independent ρ-dominating set of G^2.
2. $\forall v \in X$, add to S a BFS tree rooted at v up to distance $2\rho + \gamma$.

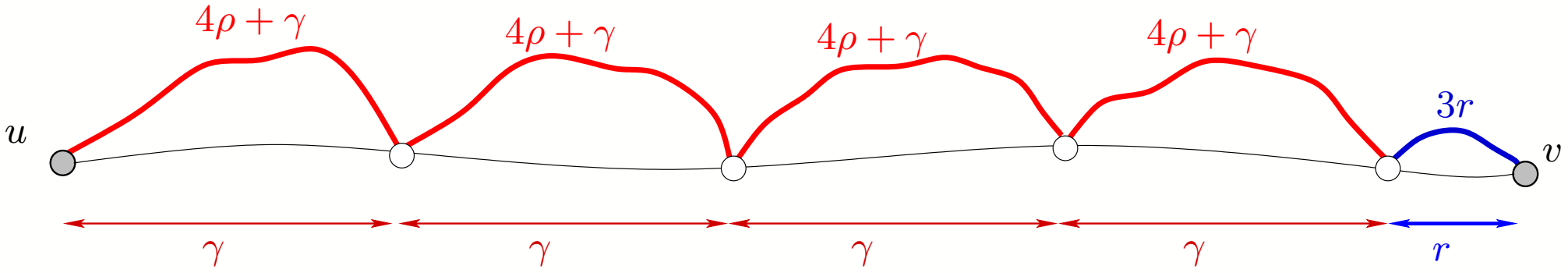
Stretch Analysis

Almost pure additive spanners

Algorithm :

1. run our 3-spanner algorithm (parameter ρ) :
   - compute S : a 3-spanner with $O(n^{3/2})$ edges.
   - compute X : an independent ρ-dominating set of G^2.
2. $\forall v \in X$, add to S a BFS tree rooted at v up to distance $2\rho + \gamma$.

Stretch Analysis

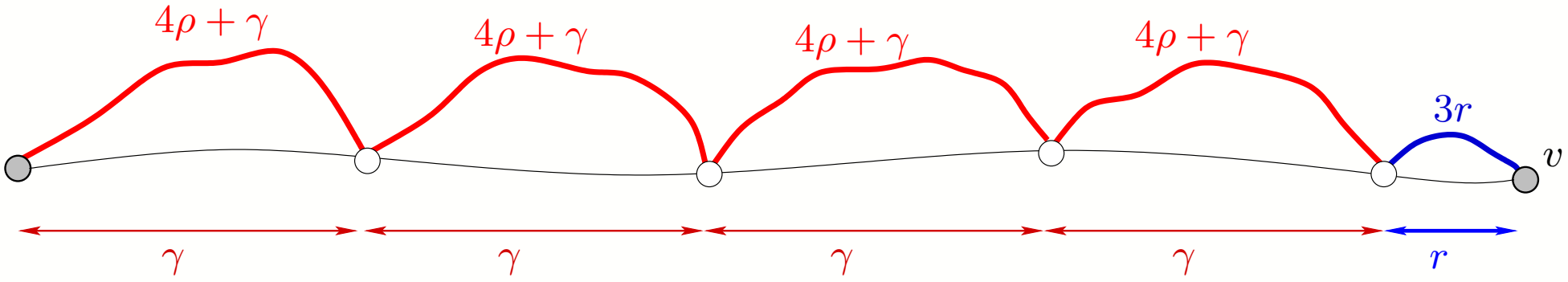


Almost pure additive spanners

Algorithm :

1. run our 3-spanner algorithm (parameter ρ) :
   - compute S : a 3-spanner with $O(n^{3/2})$ edges.
   - compute X : an independent ρ-dominating set of G^2.
2. $\forall v \in X$, add to S a BFS tree rooted at v up to distance $2\rho + \gamma$.

Stretch Analysis



$$\text{Stretch} = \left(1 + O(\rho/\gamma) \, , \, O(\rho) \right)$$

Almost pure additive spanners

Algorithm :

1. run our 3-spanner algorithm (parameter ρ) :
   - compute S : a 3-spanner with $O(n^{3/2})$ edges.
   - compute X : an independent ρ-dominating set of G^2.
2. $\forall v \in X$, add to S a BFS tree rooted at v up to distance $2\rho + \gamma$.

Stretch	Time	Size
$(1 + O(\rho/\gamma), O(\rho))$	$\mathrm{IDS}(n, \rho) + O(\rho + \gamma)$	$O(n^{3/2})$

Almost pure additive spanners

Algorithm :

1. run our 3-spanner algorithm (parameter ρ) :
   - compute S : a 3-spanner with $O(n^{3/2})$ edges.
   - compute X : an independent ρ-dominating set of G^2.
2. $\forall v \in X$, add to S a BFS tree rooted at v up to distance $2\rho + \gamma$.

Stretch	Time	Size
$(1 + O(\rho/\gamma), O(\rho))$	$\mathrm{IDS}(n, \rho) + O(\rho + \gamma)$	$O(n^{3/2})$
$(1 + \epsilon, O(1))$	$n^{O(1/\sqrt{\log n})} + O(\epsilon^{-1})$	$O(n^{3/2})$
$(1 + \epsilon, O(\log n))$	$O(\epsilon^{-1} \log n)$	$O(n^{3/2})$

ρ	$\mathrm{IDS}(n, \rho)$	γ
1	$n^{O(1/\sqrt{\log n})}$	$\Theta(\epsilon^{-1})$
$2 \log n$	$O(\log n)$	$\Theta(\epsilon^{-1} \log n)$

Conclusion

The locality of constructing graph spanners

Some open questions :

- Can we improve the Stretch and/or the Time.
- Do there exist $(1, f(k))$-spanners with $O(n^{1+1/k})$ edges ?

THANK YOU