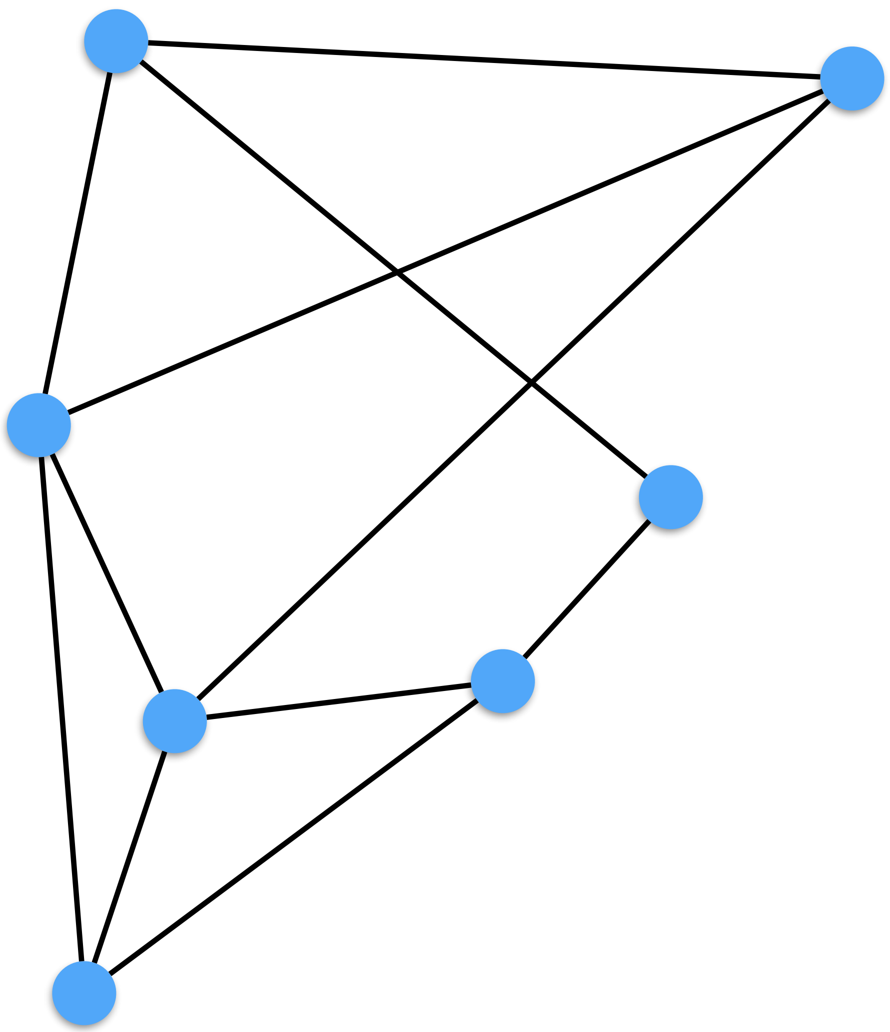
Approximating Graph Spanners

Michael Dinitz
Johns Hopkins University

Joint work with combinations of Robert Krauthgamer (Weizmann), Eden Chlamtáč (Ben Gurion), Ran Raz (Weizmann), Guy Kortsarz (Rutgers-Camden)

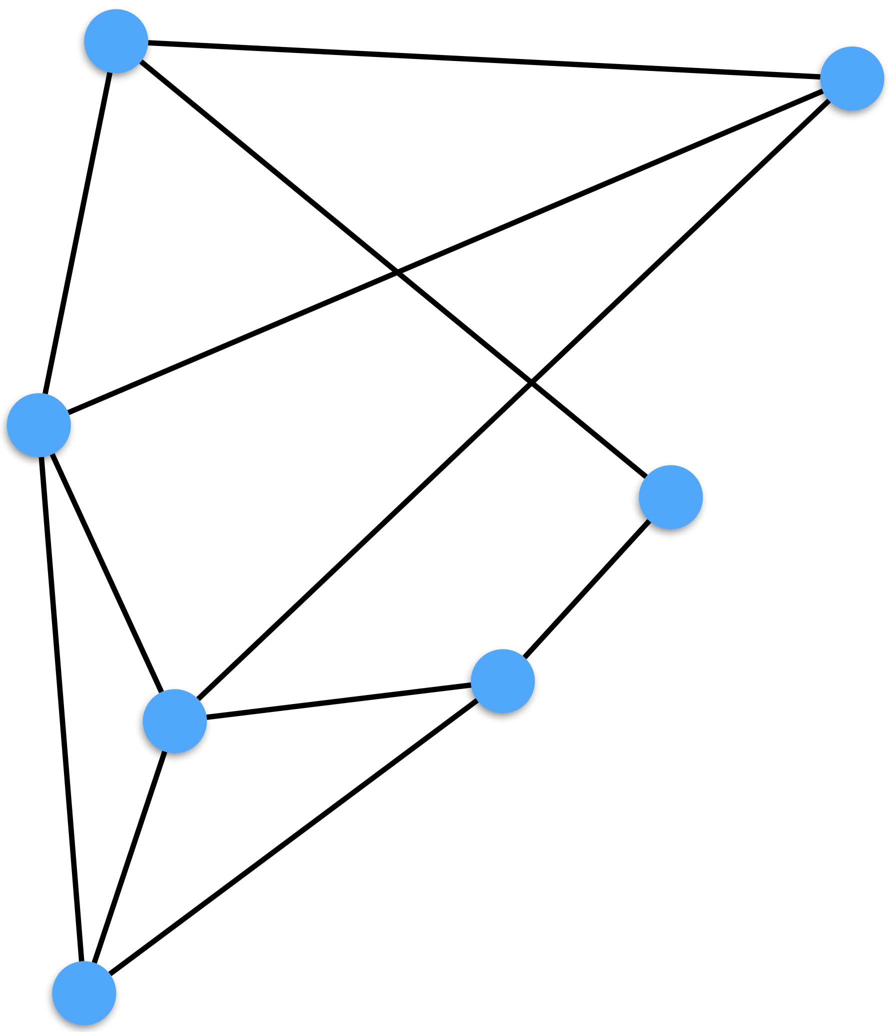
Graph Spanners

Graph Spanners



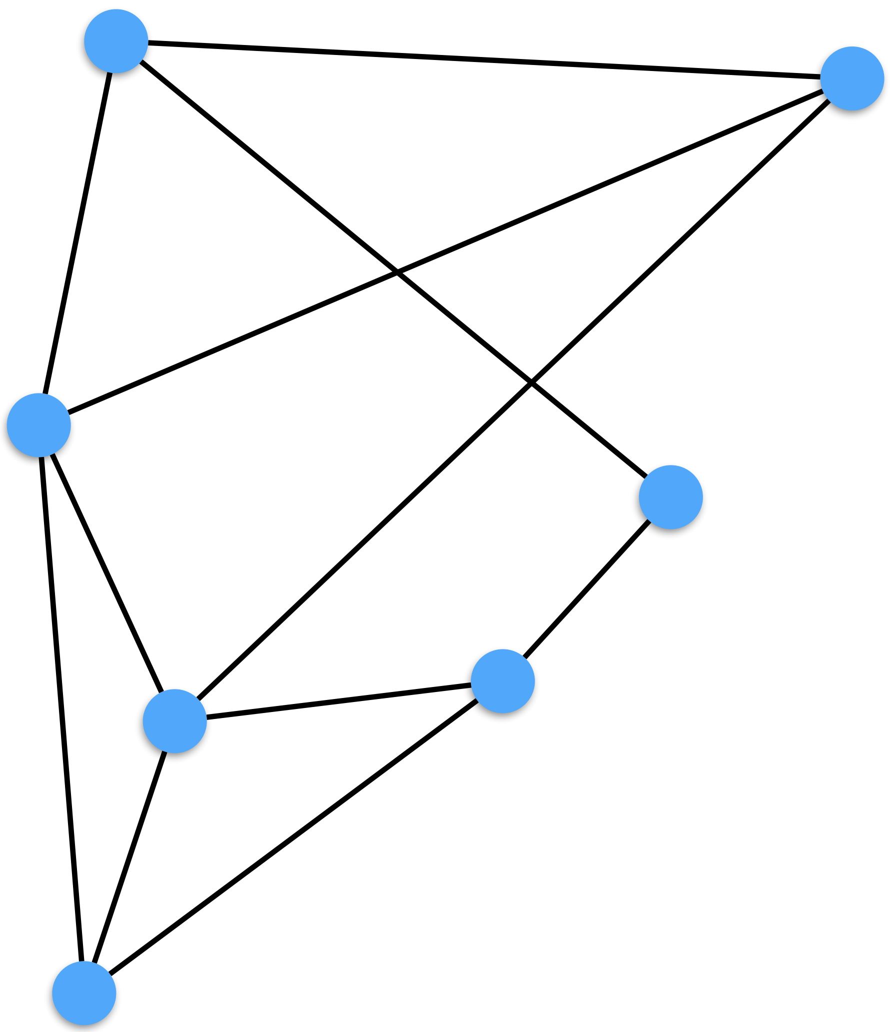
- Graph $G = (V, E)$

Graph Spanners



- Graph $G = (V, E)$
- Can we “compress” the graph while preserving distances?

Graph Spanners

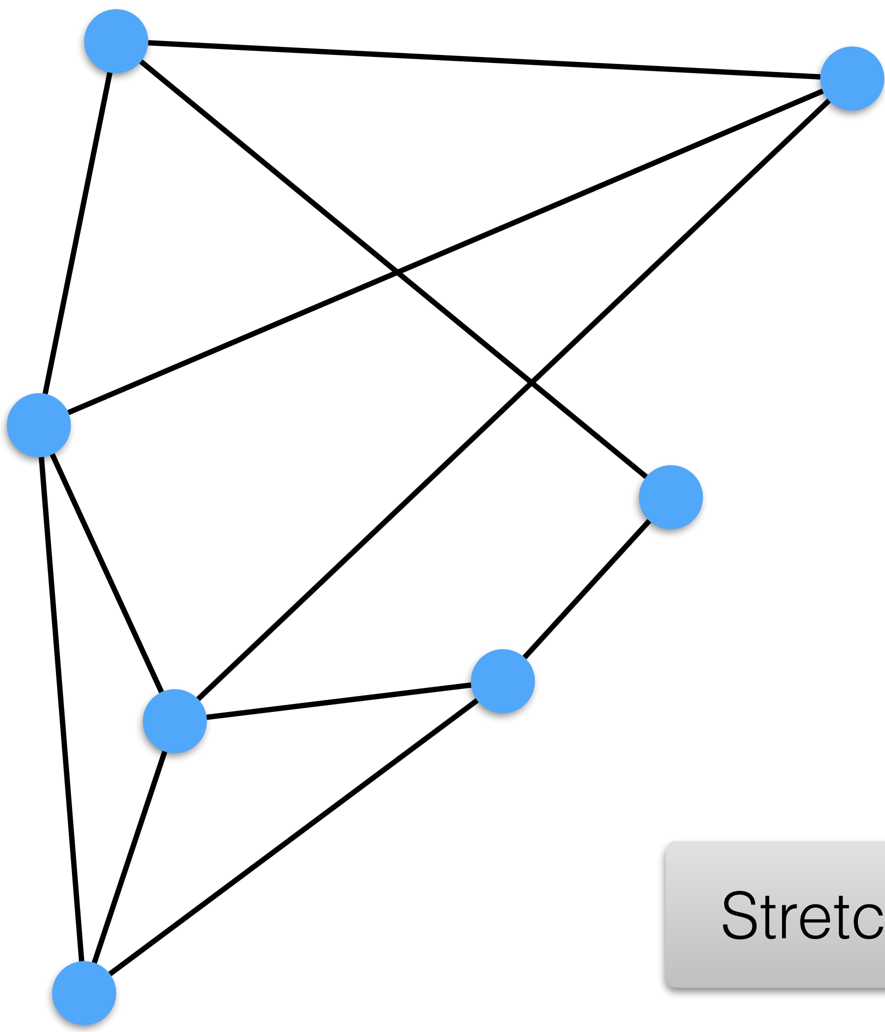


- Graph $G = (V, E)$
- Can we “compress” the graph while preserving distances?
- k-spanner: subgraph G' in which

  $d_{G'}(u,v) \leq k \times d_G(u,v)$

  for all $u,v \in V$

Graph Spanners



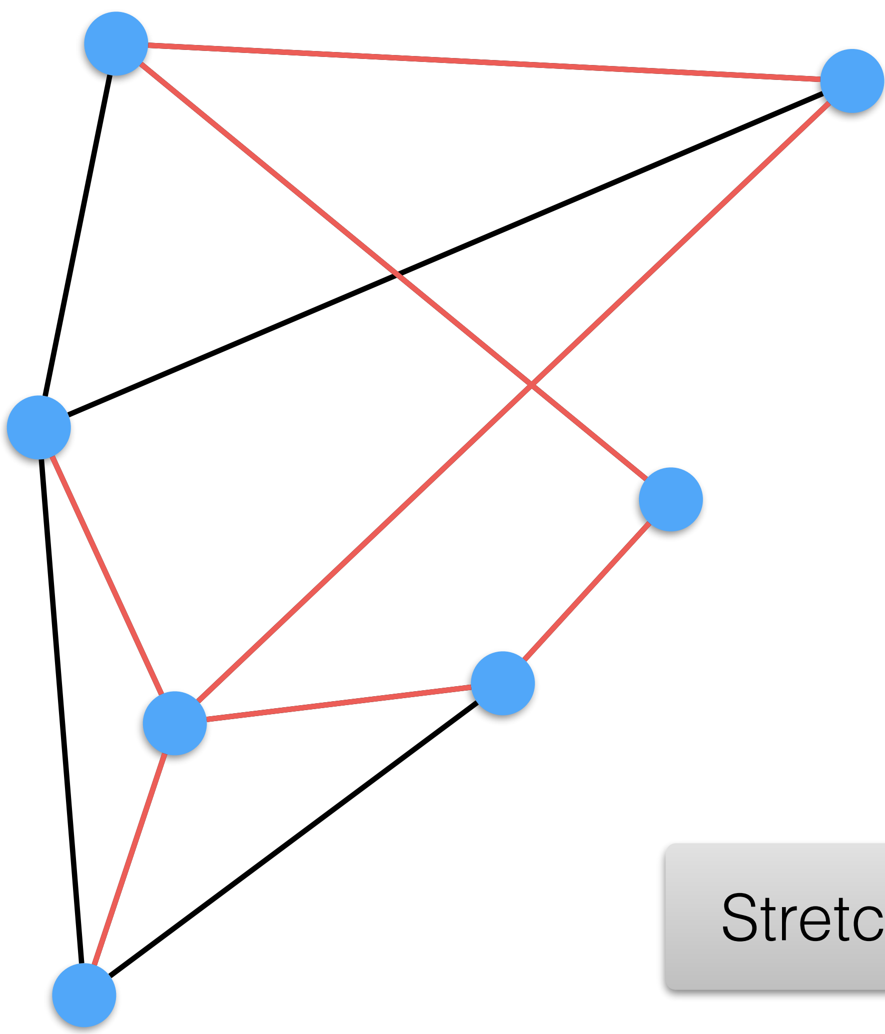
- Graph $G = (V, E)$
- Can we "compress" the graph while preserving distances?
- *k*-spanner: subgraph G' in which

Stretch

$$d_{G'}(u,v) \leq k \times d_G(u,v)$$

for all $u,v \in V$

Graph Spanners



- Graph $G = (V, E)$
- Can we "compress" the graph while preserving distances?
- k-spanner: subgraph G' in which

Stretch

$$d_{G'}(u,v) \leq k \times d_G(u,v)$$

for all $u,v \in V$

History

- First developed in late 80's for distributed computing: Peleg-Schaffer '89, Peleg-Ullman '89

- Many applications:
  - Distance oracles / compact routing
  - Property testing
  - Preprocessing approximation algorithms
  - Maximizing influence spread in social networks
  - Biomedical image segmentation

- Many papers

Fundamental Tradeoff

Theorem [ADDJS '93]: For all $k \geq 1$, every graph G has a $(2k-1)$-spanner with at most $n^{1+1/k}$ edges

Fundamental Tradeoff

Theorem [ADDJS '93]: For all $k \geq 1$, every graph G has a $(2k-1)$-spanner with at most $n^{1+1/k}$ edges

- Very simple greedy algorithm
- Tight (assuming Erdős girth conjecture)
- Lots of followup work extending tradeoff to weight, diameter, hop count, … and proving stronger/different tradeoffs for special classes

What Else?

What Else?

- Great! Tradeoff means applications using spanners can always find sparse spanners

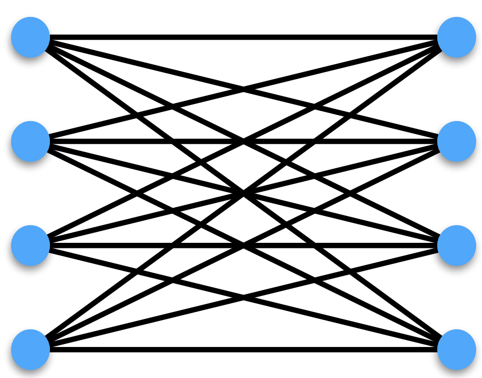
What Else?

- Great! Tradeoff means applications using spanners can always find sparse spanners

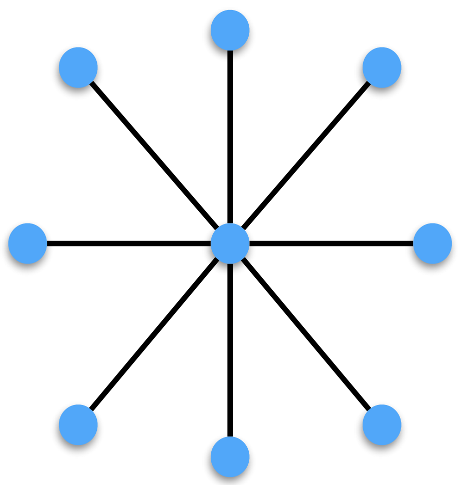
- Not so fast…

What Else?

- Great! Tradeoff means applications using spanners can always find sparse spanners

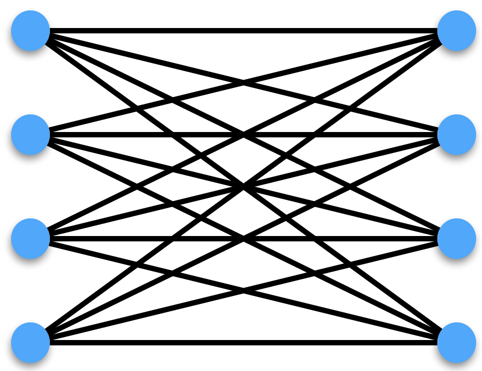


- Not so fast…
  - Tradeoffs don't always exist (2-spanner, directed, max degree)

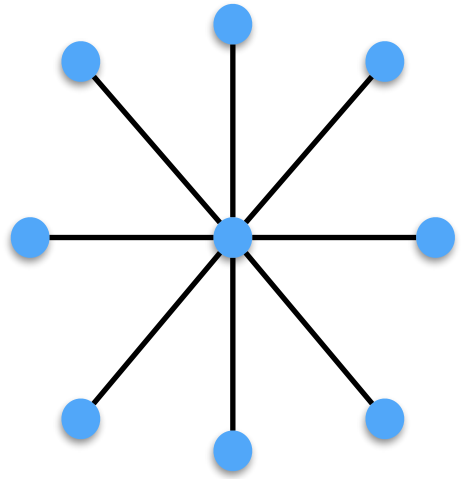


What Else?

- Great! Tradeoff means applications using spanners can always find sparse spanners



- Not so fast…
  - Tradeoffs don't always exist (2-spanner, directed, max degree)
  - Even if they do, still want to find *best* spanner



Optimization

Optimization

- Given graph *G* and value *k*, can we find the best *k*-spanner?

Optimization

- Given graph *G* and value *k*, can we find the best *k*-spanner?
- "Best":
  - Minimum number of edges
  - Minimum total weight
  - Minimum max degree
  - …

Optimization

- Given graph *G* and value *k*, can we find the best *k*-spanner?

- "Best":
  - Minimum number of edges
  - Minimum total weight
  - Minimum max degree
  - …

- This talk: some results, some open questions
  - Still much to do!

Basic k-Spanner

Basic *k*-Spanner

- Most basic version: given undirected G, integer k, find k-spanner with minimum # edges

Basic *k*-Spanner

- Most basic version: given undirected G, integer k, find k-spanner with minimum # edges
  - $k = 2$: Tight $O(\log n)$-approximation [Kortsarz-Peleg '92]
  - $k \geq 3$: Basic tradeoff gives $O(n^{2/(k+1)})$-approximation (odd k) or $O(n^{2/k})$-approximation (even k)
  - $k = 3$: $\tilde{O}(n^{1/3})$-approximation [BBMRY '11]

Basic *k*-Spanner

- Most basic version: given undirected G, integer k, find k-spanner with minimum # edges
  - $k = 2$: Tight $O(\log n)$-approximation [Kortsarz-Peleg '92]
  - $k \geq 3$: Basic tradeoff gives $O(n^{2/(k+1)})$-approximation (odd k) or $O(n^{2/k})$-approximation (even k)
  - $k = 3$: $\tilde{O}(n^{1/3})$-approximation [BBMRY '11]

- Open Question: is it possible to beat the ADDJS bound for stretch values larger than 3?

Strong Hardness

- Can we hope for an *$O(\log n)$*-approx?

Strong Hardness

- Can we hope for an $O(\log n)$-approx?

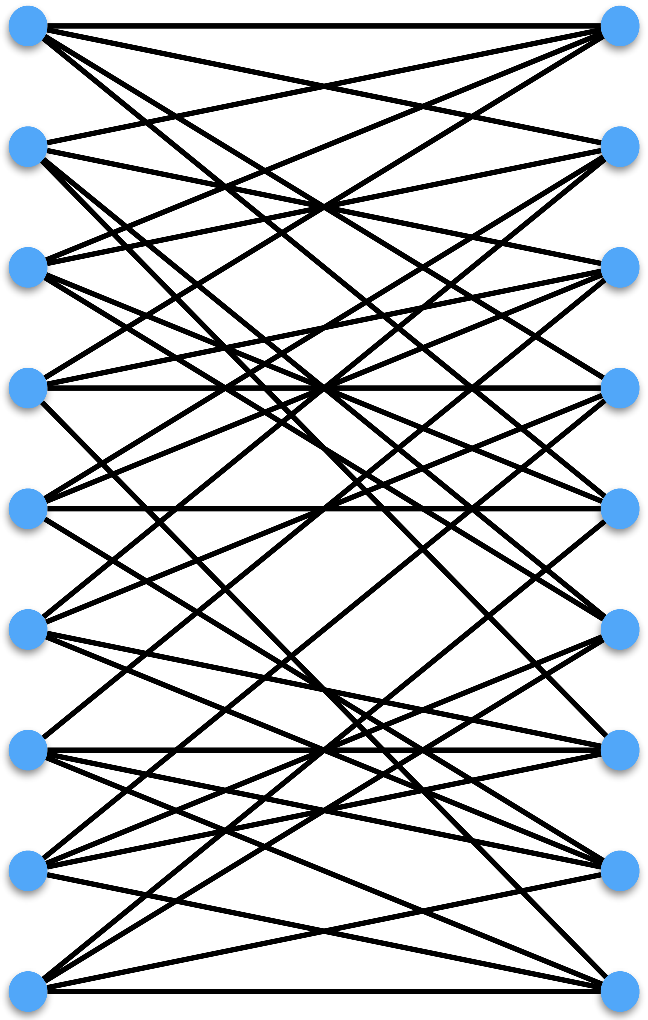
Theorem [D-Kortsarz-Raz ICALP'12]:

There is no polynomial-time algorithm that can approximate Basic *k*-Spanner better than $2^{(\log n)^{(1-\varepsilon)} / k}$ unless $NP \subseteq BPTIME(n^{polylog(n)})$

Techniques for Hardness

Techniques for Hardness

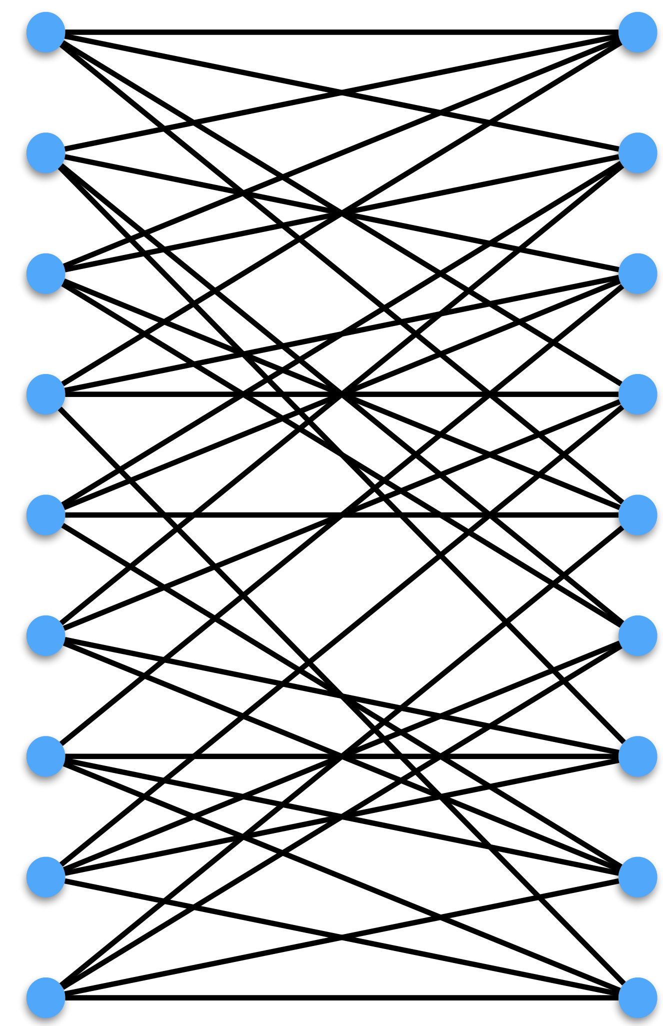
- [Elkin-Peleg ICALP'00] framework: Label Cover → Label Cover with girth ≥ k → Basic k-Spanner



Techniques for Hardness

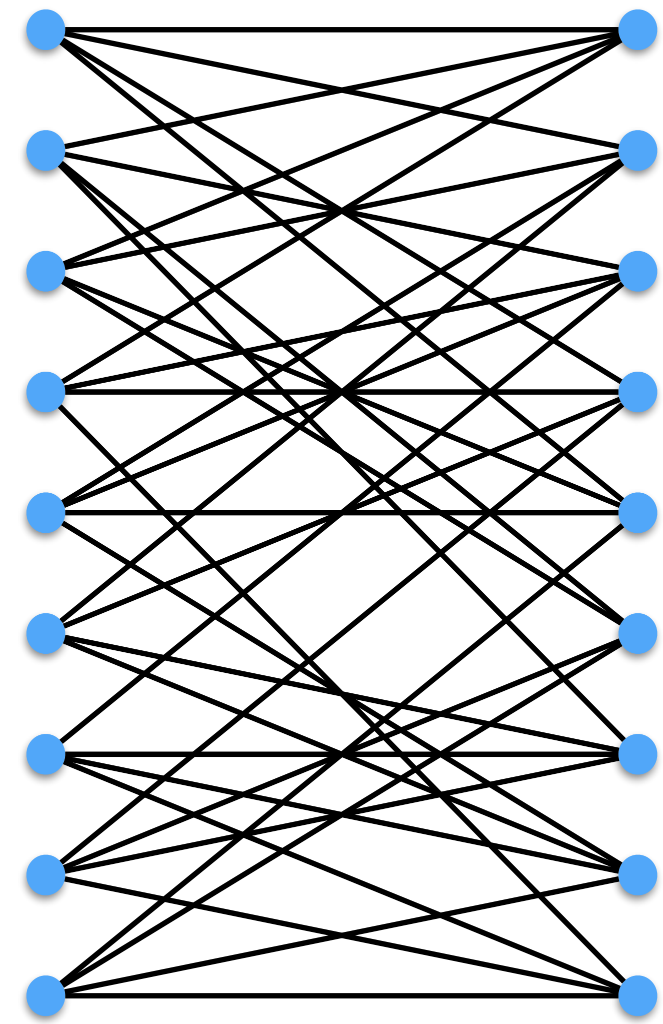
- [Elkin-Peleg ICALP'00] framework: Label Cover → Label Cover with girth ≥ k → Basic k-Spanner

- Second reduction straightforward: how to do first?



Techniques for Hardness

- [Elkin-Peleg ICALP'00] framework: Label Cover → Label Cover with girth ≥ k → Basic k-Spanner
- Second reduction straightforward: how to do first?
- Key idea: random subsampling

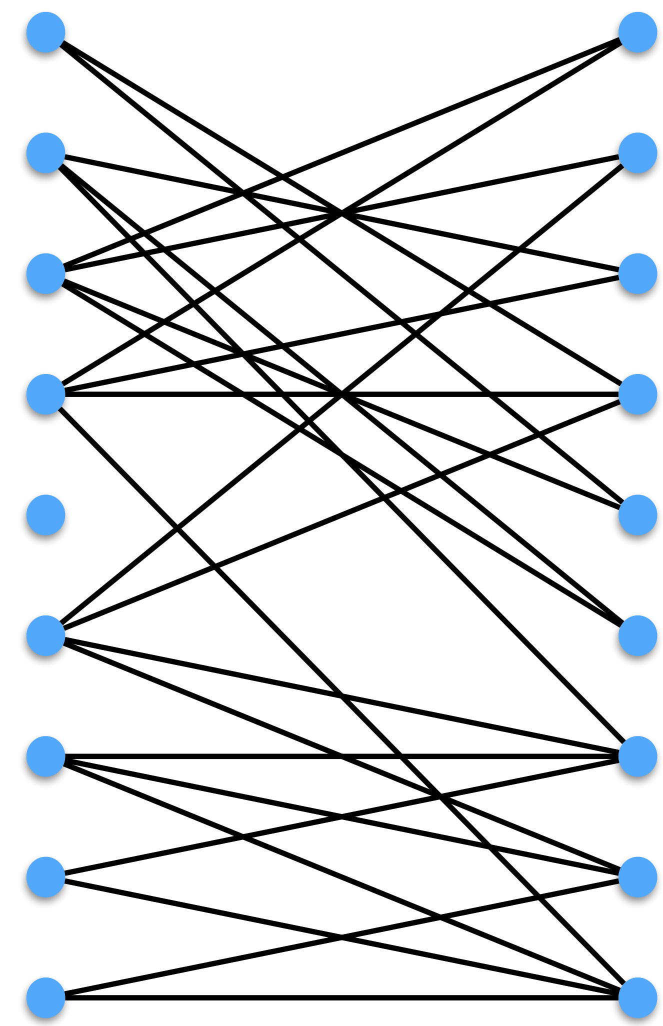


Techniques for Hardness

- [Elkin-Peleg ICALP'00] framework: Label Cover → Label Cover with girth ≥ k → Basic k-Spanner

- Second reduction straightforward: how to do first?

- Key idea: random subsampling



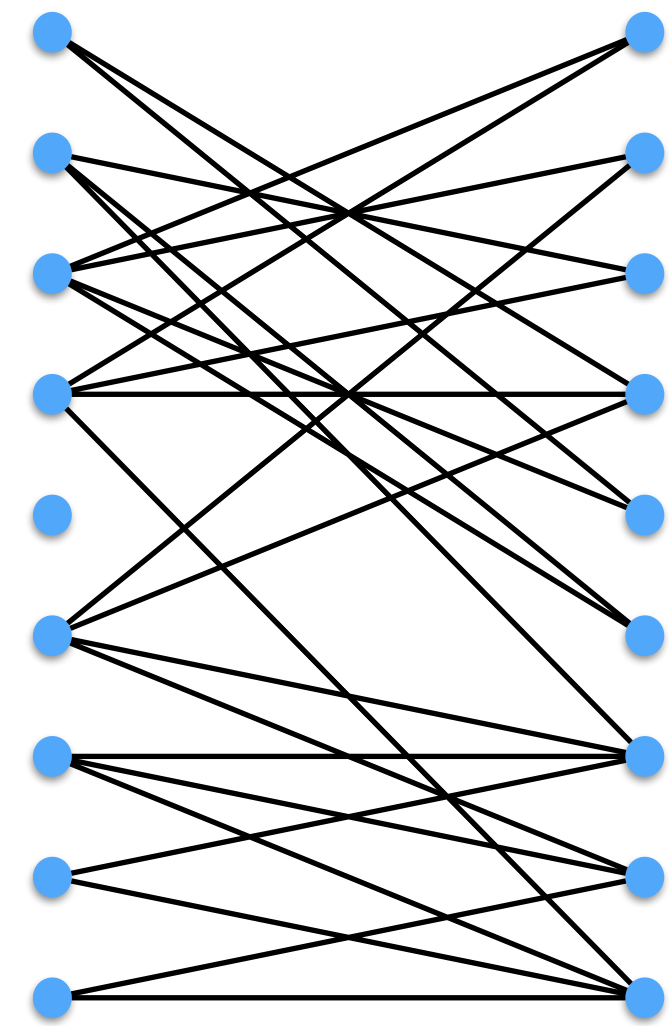
Techniques for Hardness

- [Elkin-Peleg ICALP'00] framework: Label Cover → Label Cover with girth ≥ k → Basic k-Spanner

- Second reduction straightforward: how to do first?

- Key idea: random subsampling

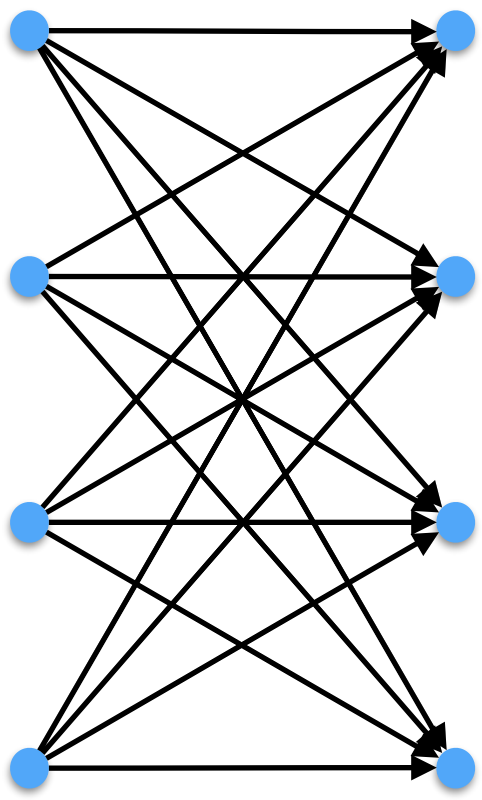
- Takeaway: hardness ≈ degree
  - "Improvement" over parallel repetition: apply repetition, then sample
  - Same hardness, smaller degree/fewer edges



Directed Spanners

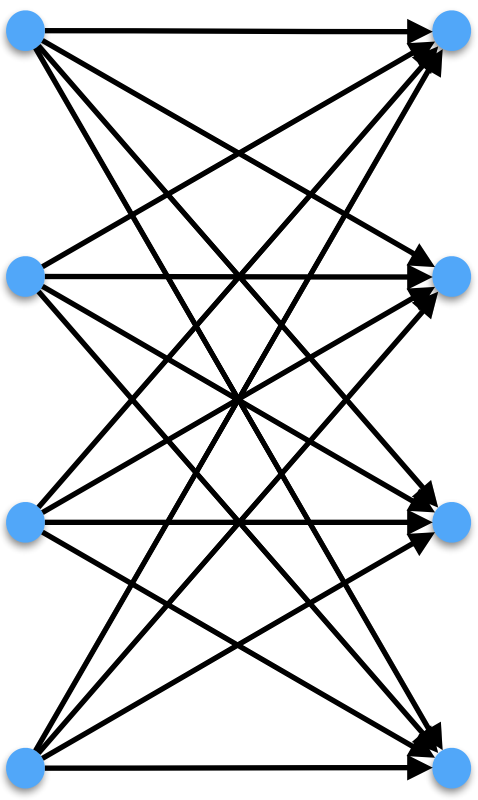
Directed Spanners

- No tradeoff possible



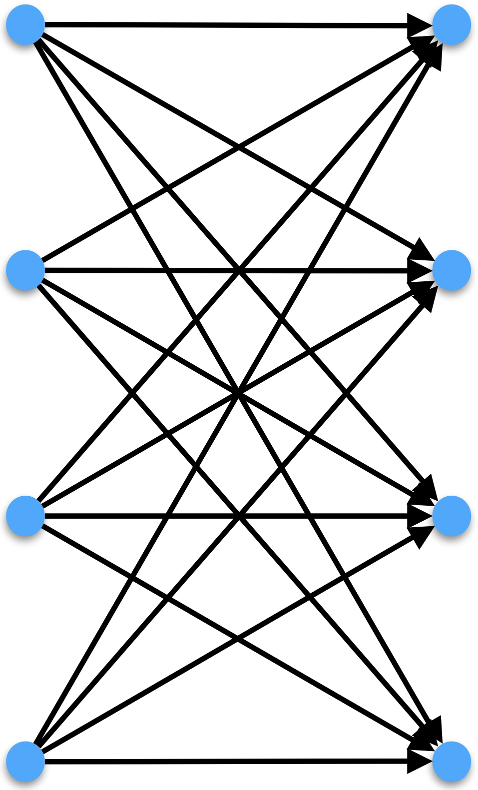
Directed Spanners

- No tradeoff possible
- Hard to approximate better than $2^{(\log n)^{(1-\varepsilon)}}$ [Elkin-Peleg STACS'00]



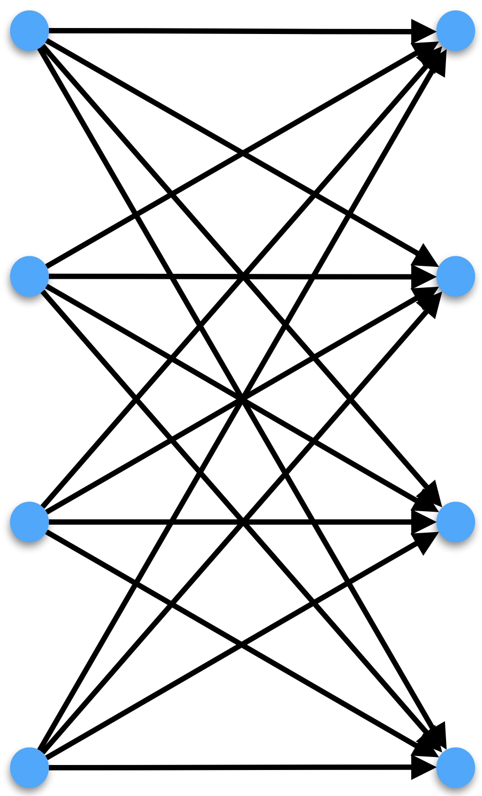
Directed Spanners

- No tradeoff possible
- Hard to approximate better than $2^{(\log n)^{(1-\varepsilon)}}$ [Elkin-Peleg STACS'00]
- Upper bounds:
    - $\tilde{O}(n^{1-1/k})$ [BGJRW SODA'09]
    - $\tilde{O}(n^{2/3})$ [D-Krauthgamer STOC'11]
    - $\tilde{O}(n^{1/2})$ [BBMRY ICALP'11]



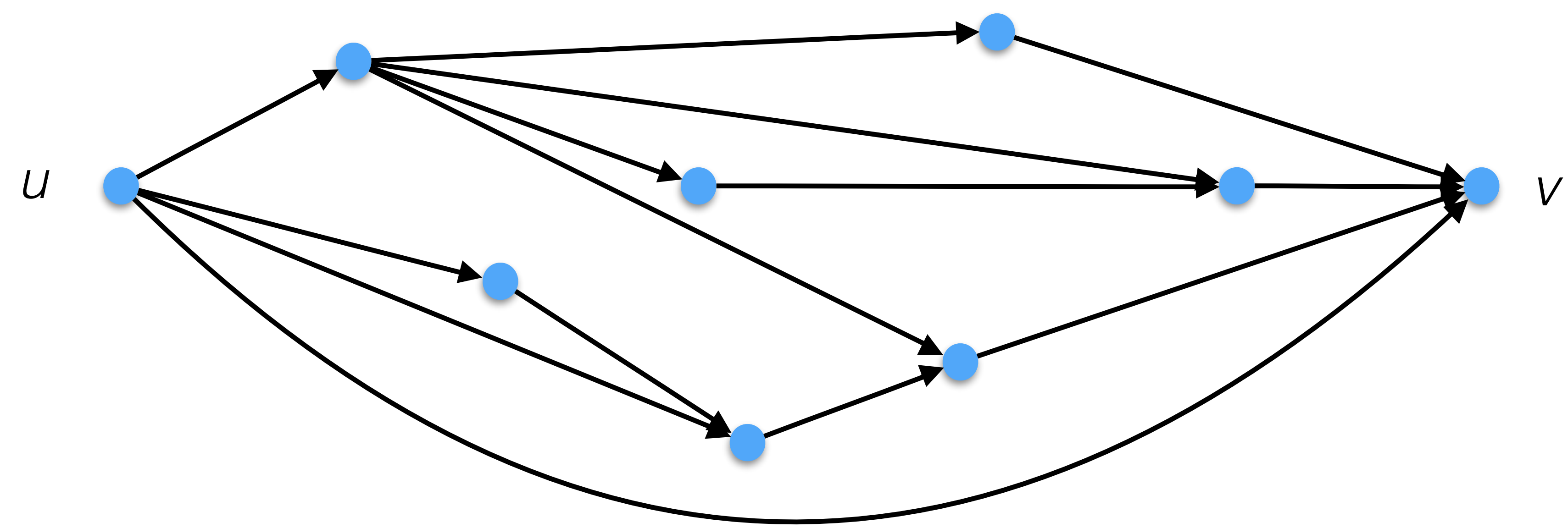
Directed Spanners

- No tradeoff possible
- Hard to approximate better than $2^{(\log n)^{(1-\varepsilon)}}$ [Elkin-Peleg STACS'00]
- Upper bounds:
  - $\tilde{O}(n^{1-1/k})$ [BGJRW SODA'09]
  - $\tilde{O}(n^{2/3})$ [D-Krauthgamer STOC'11]
  - $\tilde{O}(n^{1/2})$ [BBMRY ICALP'11]
  - Stretch 3: $\tilde{O}(n^{1/2})$ [DK STOC'11], $\tilde{O}(n^{1/3})$ [BBMRY ICALP'11]



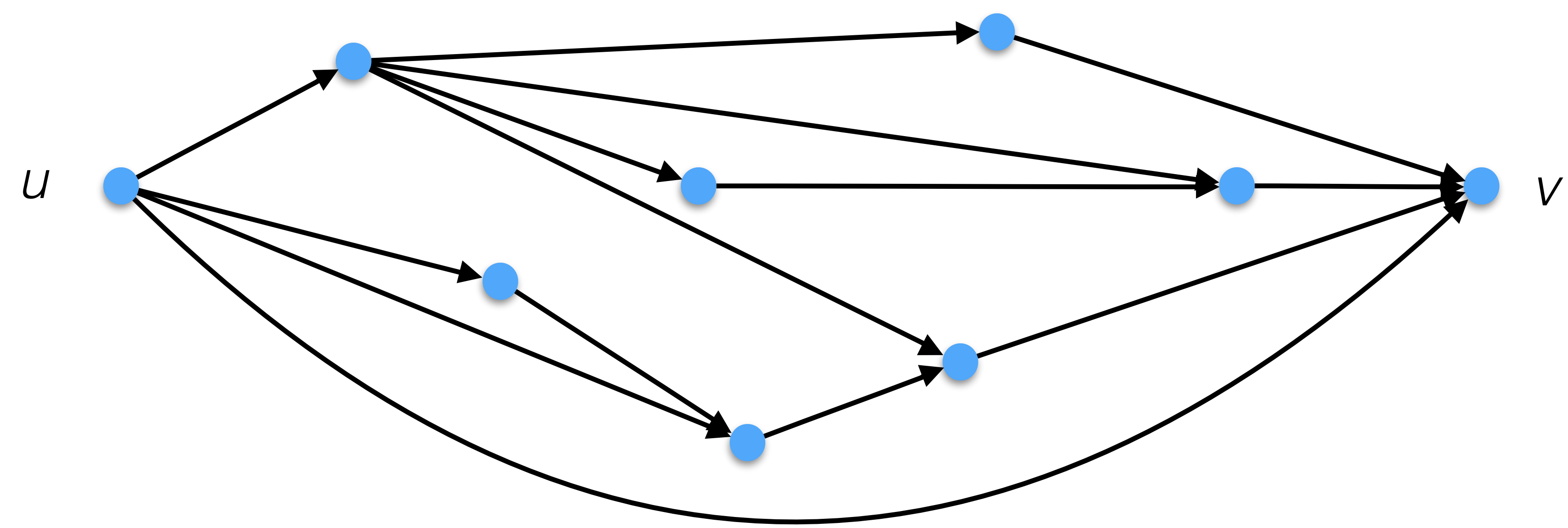
High-Level Framework [BGJRW '09]

k=3



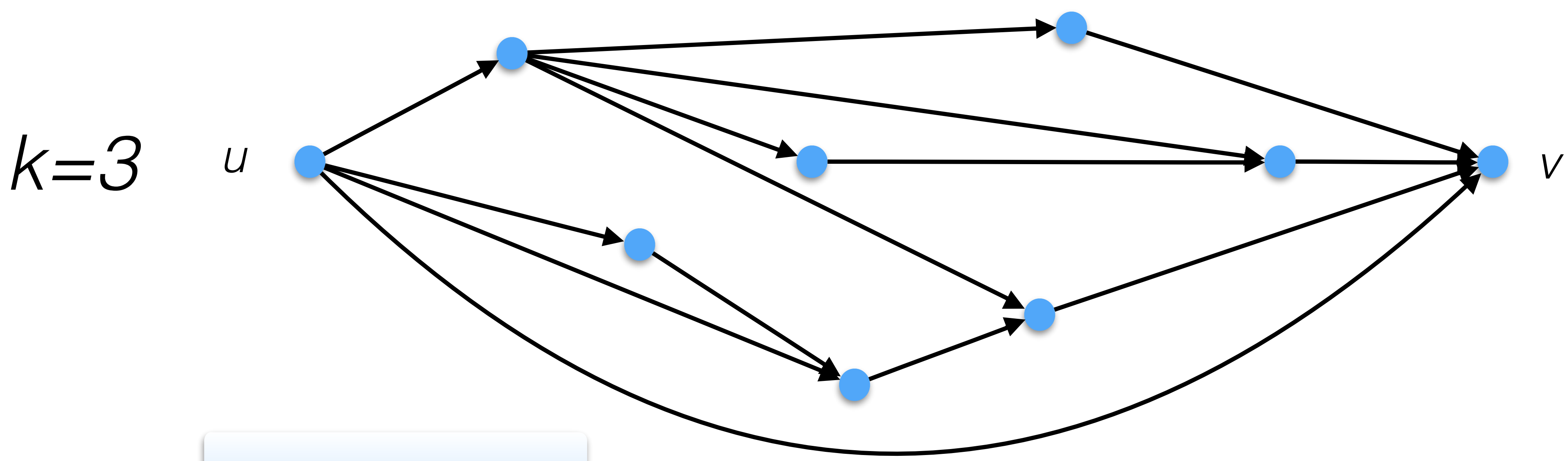
High-Level Framework [BGJRW '09]

k=3



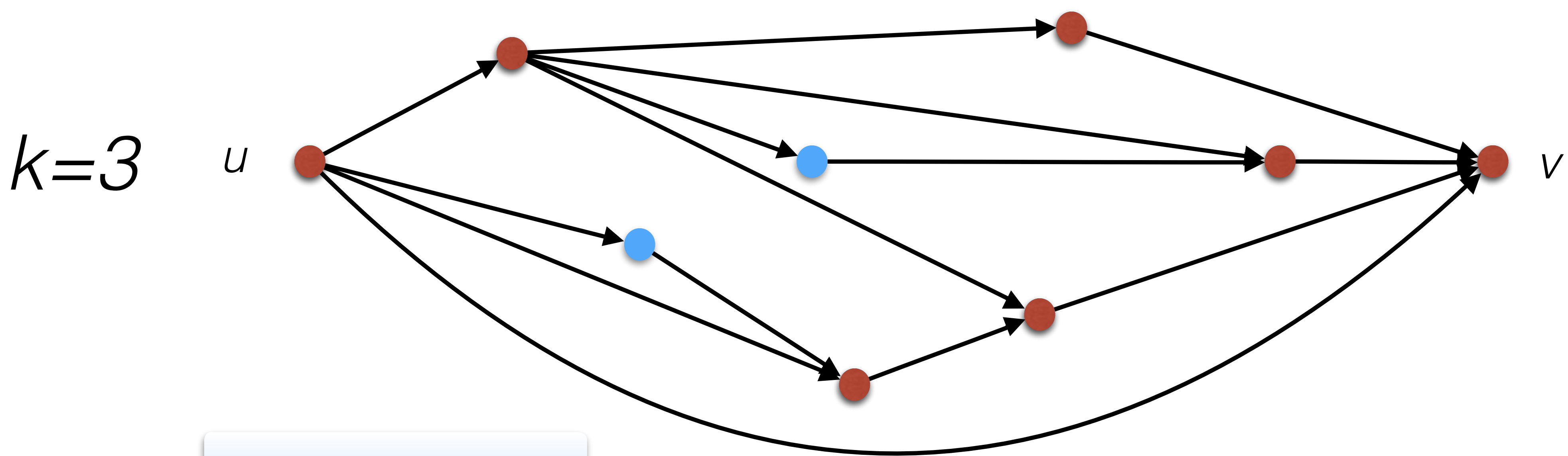
- N(u,v) = {w : ∃ stretch-*k* *u*-*v* path containing *w*}

High-Level Framework [BGJRW '09]



- N(u,v) = {w : ∃ stretch-*k* *u*-*v* path containing *w*}

High-Level Framework [BGJRW '09]

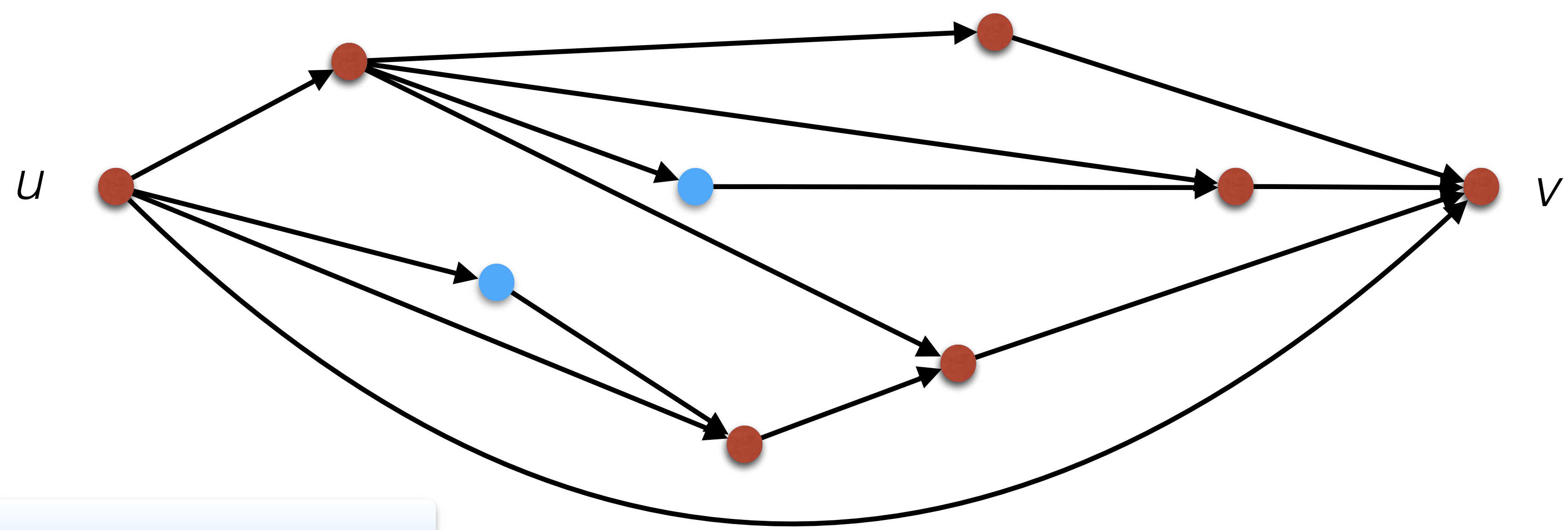


Local Neighborhood

- N(u,v) = {w : ∃ stretch-*k* *u*-*v* path containing *w*}

High-Level Framework [BGJRW ’09]

k=3



Local Neighborhood

- N(u,v) = {w : ∃ stretch-*k* *u*-*v* path containing *w*}
- Two algorithms: one for small |N(u,v)|, one for large

Small $|N(u,v)|$

Small |N(u,v)|

- Linear Program [DK ’11]:

Small |N(u,v)|

- Linear Program [DK '11]:

$$
\begin{aligned}
\min \quad & \sum_{e \in E} x_e \\
\text{s.t.} \quad & \sum_{P \in \mathcal{P}_{u,v} : e \in P} f_P \leq x_e & \forall (u,v) \in E,\ \forall e \in E \\
& \sum_{P \in \mathcal{P}_{u,v}} f_P \geq 1 & \forall (u,v) \in E \\
& x_e \geq 0 & \forall e \in E \\
& f_P \geq 0 & \forall (u,v) \in E,\ \forall P \in \mathcal{P}_{u,v}
\end{aligned}
$$

Small |N(u,v)|

- Linear Program [DK '11]:

$$
\begin{aligned}
\min \quad & \sum_{e \in E} x_e & \\
\text{s.t.} \quad & \sum_{P \in \mathcal{P}_{u,v} : e \in P} f_P \le x_e & \forall (u,v) \in E,\ \forall e \in E \\
& \sum_{P \in \mathcal{P}_{u,v}} f_P \ge 1 & \forall (u,v) \in E \\
& x_e \ge 0 & \forall e \in E \\
& f_P \ge 0 & \forall (u,v) \in E,\ \forall P \in \mathcal{P}_{u,v}
\end{aligned}
$$

Stretch *k* *u*-*v* paths

Small |N(u,v)|

- Linear Program [DK '11]:

$$
\begin{aligned}
\min \quad & \sum_{e \in E} x_e \\
\text{s.t.} \quad & \sum_{P \in \mathcal{P}_{u,v} : e \in P} f_P \le x_e & \forall (u,v) \in E,\ \forall e \in E \\
& \sum_{P \in \mathcal{P}_{u,v}} f_P \ge 1 & \forall (u,v) \in E \\
& x_e \ge 0 & \forall e \in E \\
& f_P \ge 0 & \forall (u,v) \in E,\ \forall P \in \mathcal{P}_{u,v}
\end{aligned}
$$

Stretch *k* *u*-*v* paths

- Take *e* with probability t^*x_e [BBRMY '11]: spans all (u,v) where $|N(u,v)| \le t$
  - Analysis: union over all "cuts"

Small |N(u,v)|

- Linear Program [DK '11]:

$$
\begin{aligned}
\min \quad & \sum_{e \in E} x_e \\
\text{s.t.} \quad & \sum_{P \in \mathcal{P}_{u,v} : e \in P} f_P \le x_e && \forall (u,v) \in E,\ \forall e \in E \\
& \sum_{P \in \mathcal{P}_{u,v}} f_P \ge 1 && \forall (u,v) \in E \\
& x_e \ge 0 && \forall e \in E \\
& f_P \ge 0 && \forall (u,v) \in E,\ \forall P \in \mathcal{P}_{u,v}
\end{aligned}
$$

Stretch *k* *u*-*v* paths

- Take *e* with probability $t^* x_e$ [BBRMY '11]: spans all (u,v) where $|N(u,v)| \le t$
  - Analysis: union over all "cuts"
- Improves over threshold rounding [DK '11]

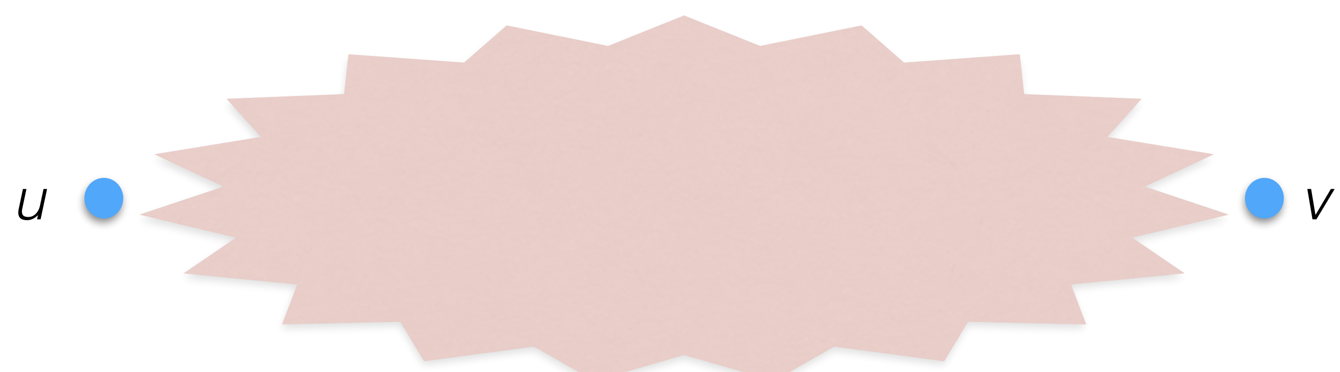
Large $|N(u,v)|$

Large |N(u,v)|

- β times: randomly choose *w*, build shortest-path in- and out-arborescences
  - If $w \in N(u,v)$ we win

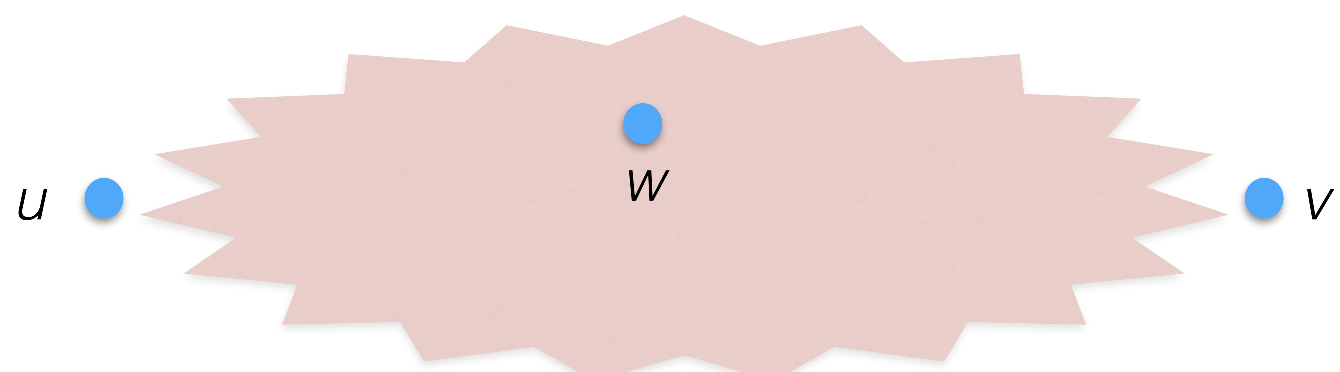
Large |N(u,v)|

- β times: randomly choose *w*, build shortest-path in- and out-arborescences
  - If $w \in N(u,v)$ we win



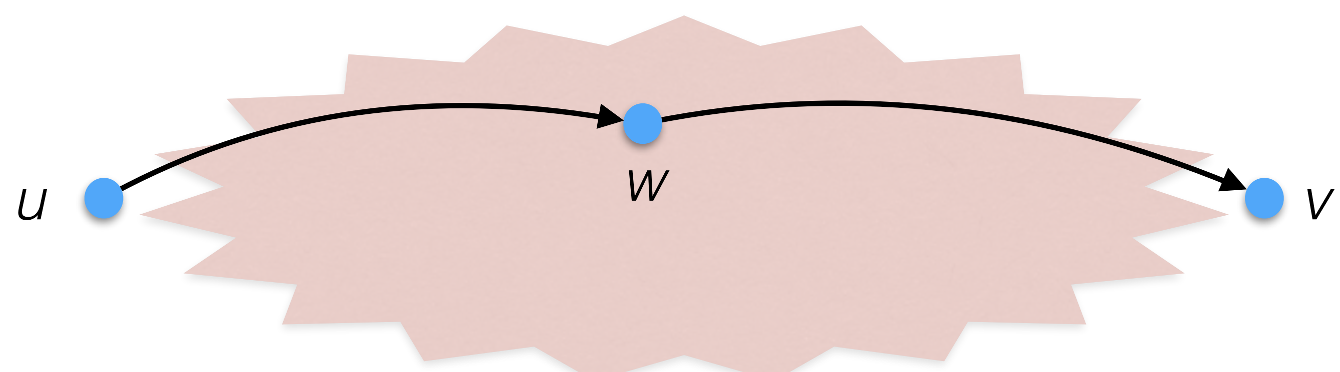
Large |N(u,v)|

- β times: randomly choose *w*, build shortest-path in- and out-arborescences
  - If $w \in N(u,v)$ we win



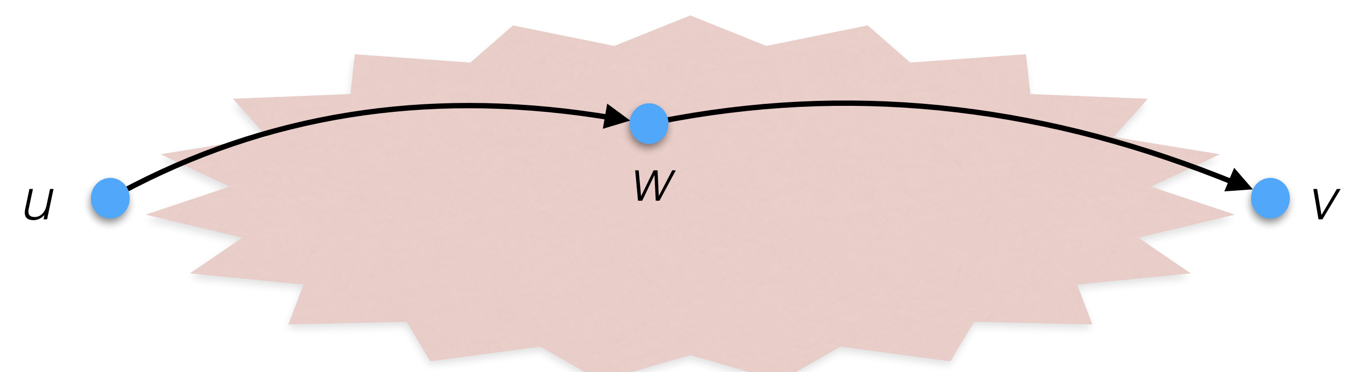
Large |N(u,v)|

- β times: randomly choose *w*, build shortest-path in- and out-arborescences
  - If $w \in N(u,v)$ we win



Large |N(u,v)|

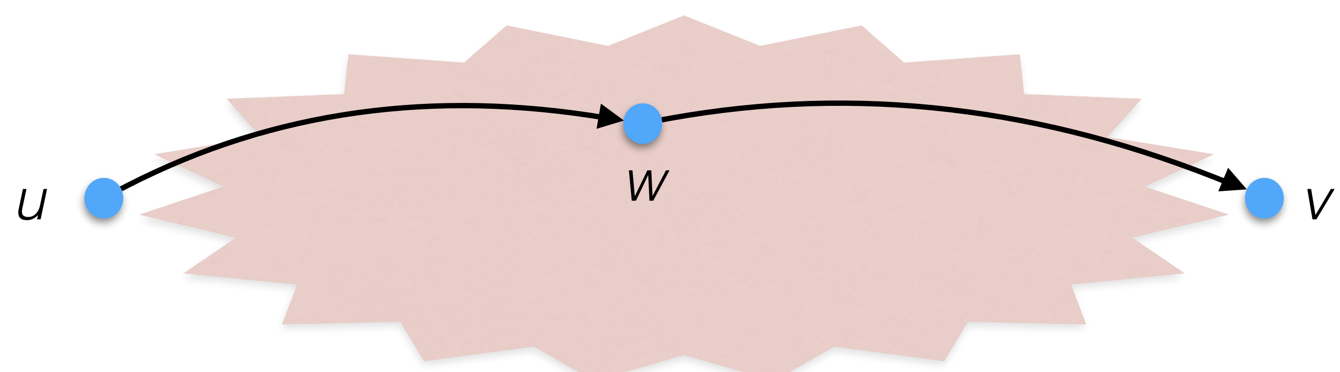
- β times: randomly choose *w*, build shortest-path in- and out-arborescences
  - If $w \in N(u,v)$ we win



- If $|N(u,v)| \leq n/\beta$, span (u,v) w.h.p.

Large |N(u,v)|

- β times: randomly choose *w*, build shortest-path in- and out-arborescences
  - If $w \in N(u,v)$ we win



- If $|N(u,v)| \leq n/\beta$, span (u,v) w.h.p.

- Cost at most $\beta * 2n \leq O(\beta * OPT)$

Combined

Combined

- Tradeoff between LP rounding and arborescence sampling: rounding for $|N(u,v)| \leq \sqrt{n}$, arborescence sampling for $|N(u,v)| \geq \sqrt{n}$

- $\tilde{O}(n^{1/2})$-approx

Combined

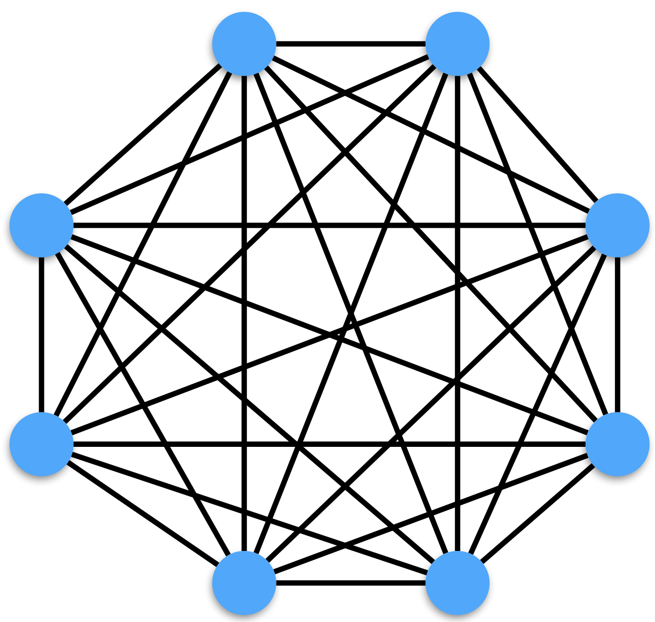
- Tradeoff between LP rounding and arborescence sampling: rounding for $|N(u,v)| \leq \sqrt{n}$, arborescence sampling for $|N(u,v)| \geq \sqrt{n}$
- $\tilde{O}(n^{1/2})$-approx
- Stretch *3*: LP rounding to to handle $|N(u,v)| \leq t$ with cost only $O(t^{1/2})$
    - $O(n^{1/2})$ by $t=n$ (without arborescence sampling) [DK '11]
    - $O(n^{1/3})$ with arborescence sampling [BBMRY '11]

Open Questions

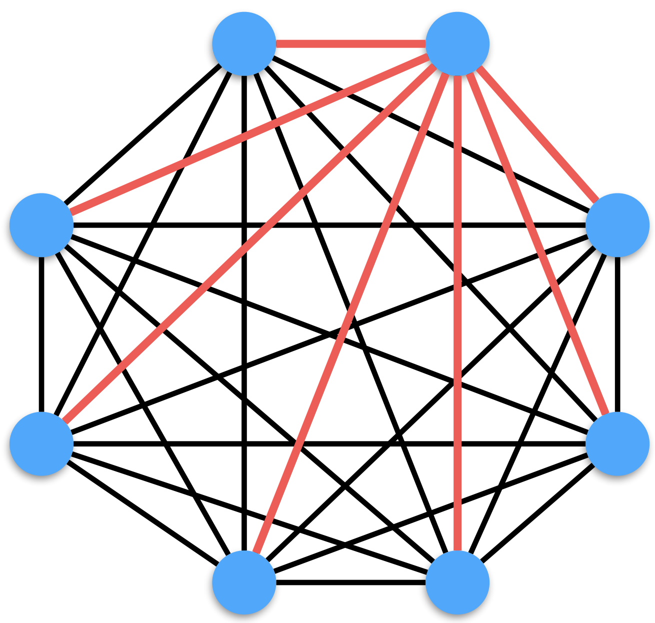
- Improved bounds?
- Arborescence sampling is terrible!
  - Uses a trivial lower bound on OPT
  - Can’t handle weights, not as flexible as LP
  - Any way to remove/reduce use of arborescence sampling?
- Back to undirected *k*-spanner:
  - Directed 3-spanner approx also best for undirected
  - Any way to use LP for larger stretch?

Fault Tolerance

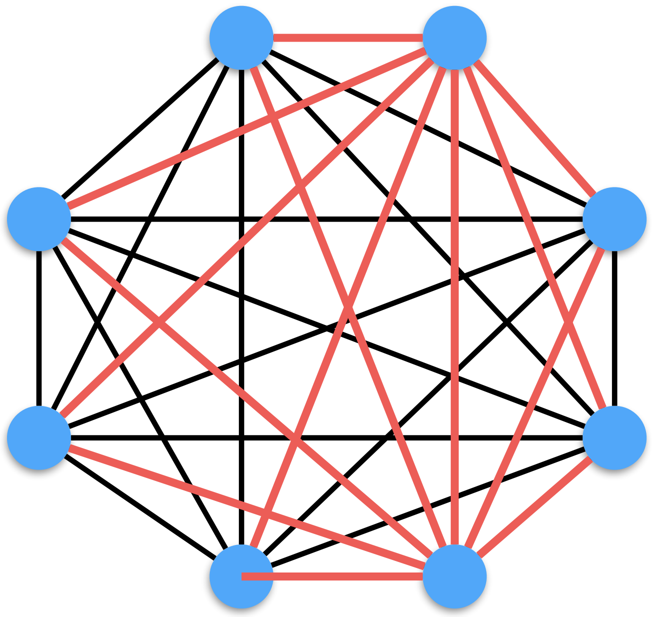
Fault Tolerance



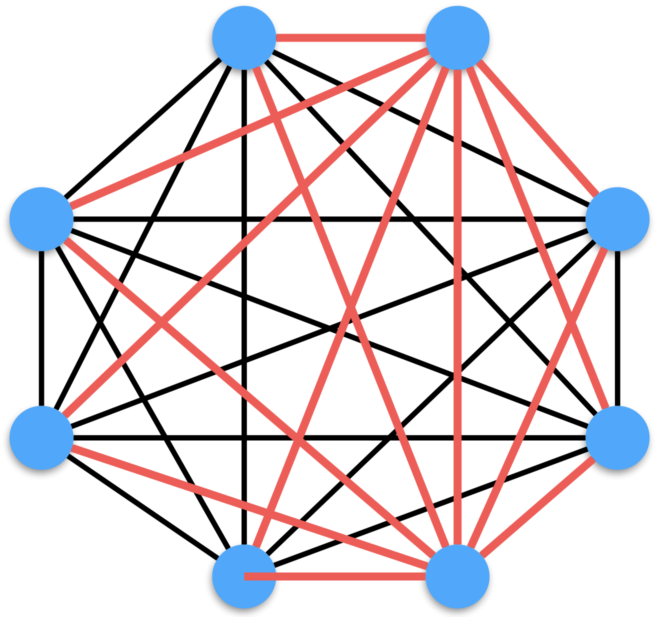
Fault Tolerance



Fault Tolerance

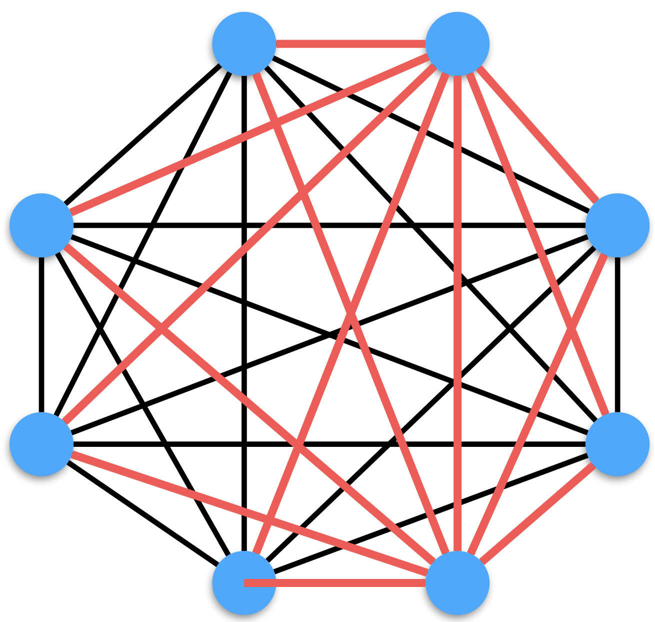


Fault Tolerance



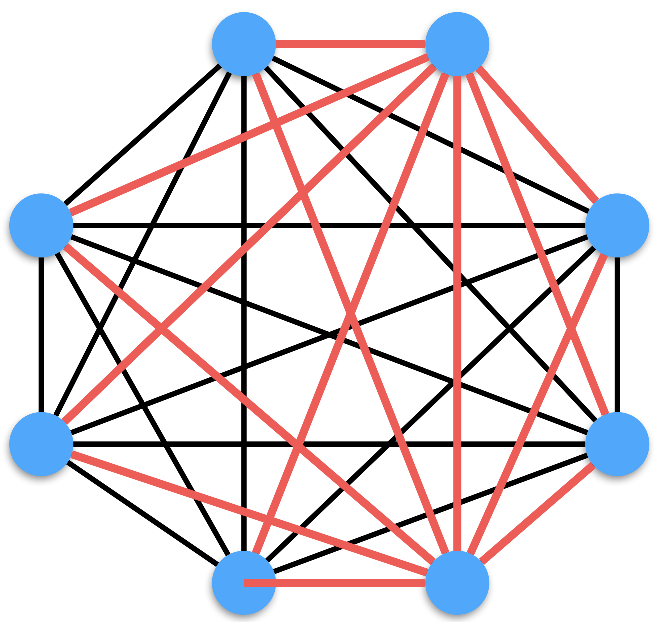
- Def [CLPR STOC'09]: H is an f-fault-tolerant k-spanner if H - F is a k-spanner of G - F for all $F \subseteq V$ with $|F| \leq f$

Fault Tolerance



- Def [CLPR STOC'09]: H is an f-fault-tolerant k-spanner if $H - F$ is a k-spanner of $G - F$ for all $F \subseteq V$ with $|F| \leq f$
- Note: might not be particularly well connected!
  - k-spanner relative to $G - F$, not G
  - If G a tree, $H = G$ is n-fault tolerant

Fault Tolerance



- Def [CLPR STOC'09]: H is an f-fault-tolerant k-spanner if H - F is a k-spanner of G - F for all $F \subseteq V$ with $|F| \leq f$
- Note: might not be particularly well connected!
  - k-spanner relative to G - F, not G
  - If G a tree, $H = G$ is n-fault tolerant
- Reasonable: can't be more fault-tolerant than G

Fault Tolerance: State of the Art

Theorem [D-Krauthgamer PODC'11]:
For every k, f, every graph G admits an f-fault tolerant $(2k-1)$-spanner with $O(f^2\, n^{1+1/k})$ edges

Fault Tolerance: State of the Art

Theorem [D-Krauthgamer PODC'11]:
For every k, f, every graph G admits an f-fault tolerant $(2k-1)$-spanner with $O(f^2 n^{1+1/k})$ edges

- f^2 times:
  - Sample each node with probability $1/f$
  - Build $(2k-1)$-spanner of size at most $n^{1+1/k}$ on sample

Fault Tolerance: State of the Art

Theorem [D-Krauthgamer PODC'11]:
For every k, f, every graph G admits an f-fault tolerant $(2k-1)$-spanner with $O(f^2 n^{1+1/k})$ edges

- f^2 times:
  - Sample each node with probability $1/f$
  - Build $(2k-1)$-spanner of size at most $n^{1+1/k}$ on sample
- Very simple to analyze

Fault-Tolerant: Stretch 2

Fault-Tolerant: Stretch 2

- Non-fault tolerant: *O(log n)*-approx [Kortsarz-Peleg JALG'94]
- *f*-fault tolerant:
  - *O(f log n)*-approx [D-Krauthgamer STOC'11]
  - *O(log n)*-approx [D-Krauthgamer PODC'11]
- Both based on rounding LP

Fault-Tolerant: Stretch 2

- Non-fault tolerant: *O(log n)*-approx [Kortsarz-Peleg JALG'94]

- *f*-fault tolerant:
  - *O(f log n)*-approx [D-Krauthgamer STOC'11]
  - *O(log n)*-approx [D-Krauthgamer PODC'11]

- Both based on rounding LP

- Open Question: approximate *f*-fault-tolerant *k*-spanner with no dependence on *f* ?
  - Tradeoff gives $O(f\, n^{1/k})$-approx

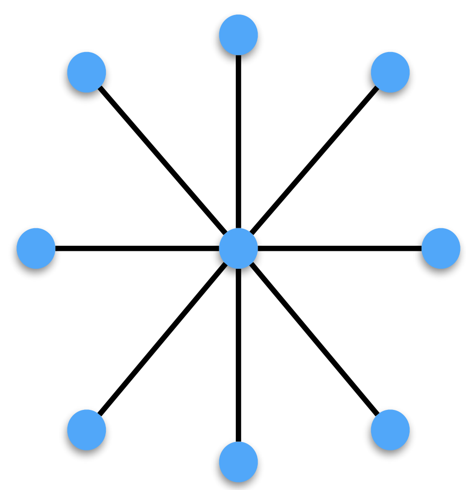
Maximum Degree

Maximum Degree

- Max degree instead of number of edges (average degree)

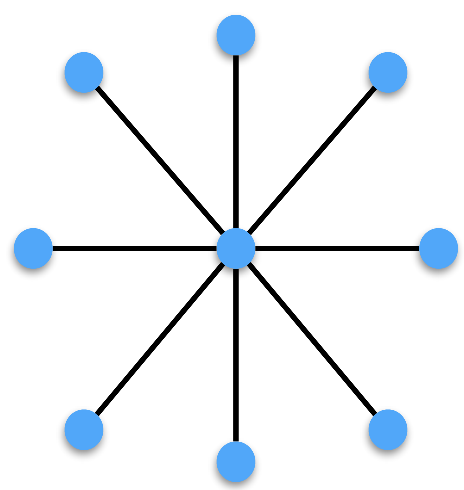
Maximum Degree

- Max degree instead of number of edges (average degree)
- No tradeoff:



Maximum Degree

- Max degree instead of number of edges (average degree)
- No tradeoff:



- Lowest Degree *k*-Spanner (LDkS): find *k*-spanner that minimizes maximum degree

LDkS vs Basic *k*-Spanner

- Very different, much more difficult!
- Stretch 2:
  - Basic 2-spanner: $O(\log n)$-approx [Kortsarz-Peleg TALG'94]
  - LD2S: $O(\Delta^{1/4})$-approx [Kortsarz-Peleg SICOMP'98]
- Larger stretch:
  - Basic *k*-Spanner: $O(n^{2/(k+1)})$-approx
  - LDkS: $\Omega(\log n)$ hardness

LDkS vs Basic *k*-Spanner

- Very different, much more difficult!
- Stretch 2:
  - Basic 2-spanner: $O(\log n)$-approx [Kortsarz-Peleg TALG'94]
  - LD2S: $O(\Delta^{1/4})$-approx [Kortsarz-Peleg SICOMP'98]

Δ = max degree

- Larger stretch:
  - Basic *k*-Spanner: $O(n^{2/(k+1)})$-approx
  - LDkS: $\Omega(\log n)$ hardness

Our Results:

- LD2S: $\tilde{O}(\Delta^{3-2\sqrt{2}+\varepsilon}) \approx \tilde{O}(\Delta^{0.172})$-approx [Chlamtac-D-Krauthgamer FOCS'12]
  - Improvement over $O(\Delta^{1/4})$
- LDkS: $\tilde{O}(\Delta^{(1-1/k)^2})$-approx, $\Omega(\Delta^{1/k})$ hardness [Chlamtac-D APPROX'14]
  - Improvement over $\Omega(\log n)$ hardness

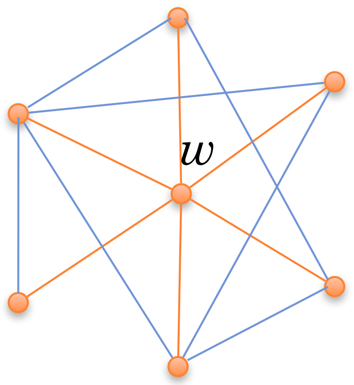
LD2S

LD2S

- Reduce to Smallest *m*-Edge Subgraph (SmES / min-DkS)

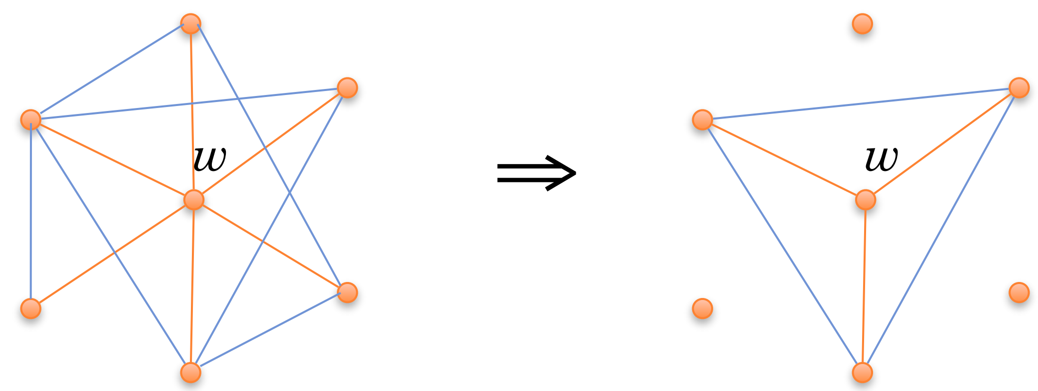
LD2S

- Reduce to Smallest *m*-Edge Subgraph (SmES / min-DkS)



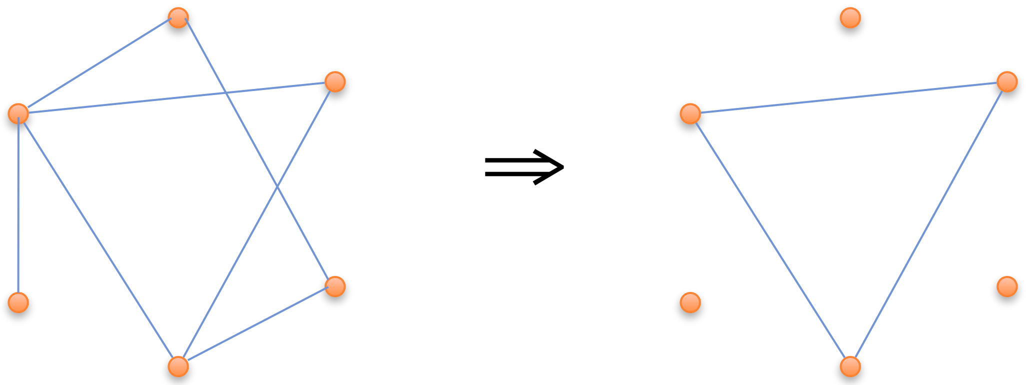
LD2S

- Reduce to Smallest *m*-Edge Subgraph (SmES / min-DkS)



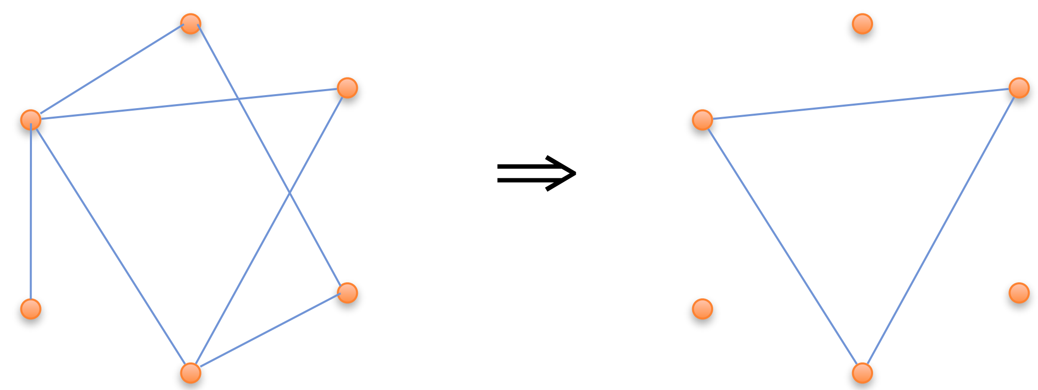
LD2S

- Reduce to Smallest *m*-Edge Subgraph (SmES / min-DkS)



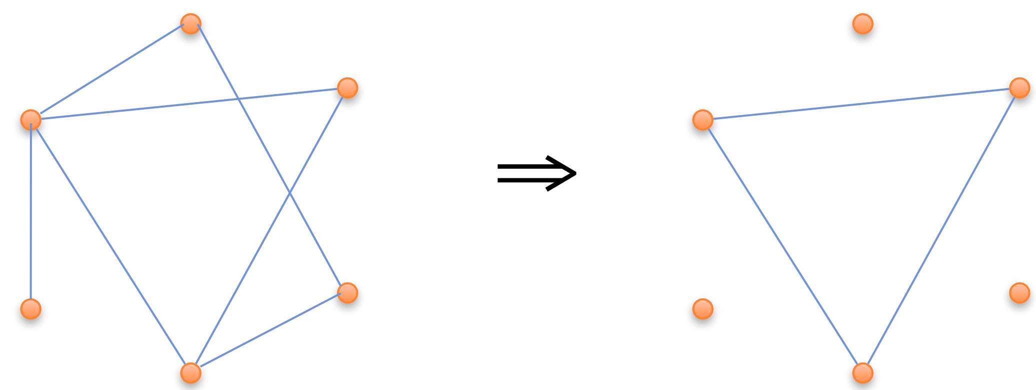
LD2S

- Reduce to Smallest *m*-Edge Subgraph (SmES / min-DkS)
  - Not black box: LD2S LP contains *n* SmES LPs
  - Need special “faithful rounding” SmES algorithm



LD2S

- Reduce to Smallest *m*-Edge Subgraph (SmES / min-DkS)
  - Not black box: LD2S LP contains *n* SmES LPs
  - Need special “faithful rounding” SmES algorithm



- Improved algorithm for SmES using Sherali-Adams hierarchy
  - $O(n^{1/4})$ directly from DkS
  - SmES used in past, first time improvement over DkS shown

LDkS: Upper Bound

- Straightforward LP:

$$
\begin{aligned}
\min \quad & d \\
\text{s.t.} \quad & \sum_{(u,v)\in E} x_{(u,v)} \le d && \forall u \in V \\
& \sum_{P\in\mathcal{P}_{u,v}: e\in P} f_P \le x_e && \forall (u,v)\in E,\ \forall e \in E \\
& \sum_{P\in\mathcal{P}_{u,v}} f_P \ge 1 && \forall (u,v)\in E \\
& x_e \ge 0 && \forall e \in E \\
& f_P \ge 0 && \forall (u,v)\in E,\ \forall P \in \mathcal{P}_{u,v}
\end{aligned}
$$

LDkS: Upper Bound

- Straightforward LP:

$$
\begin{aligned}
\min \quad & d \\
\text{s.t.} \quad & \sum_{(u,v)\in E} x_{(u,v)} \le d && \forall u \in V \\
& \sum_{P\in\mathcal{P}_{u,v}:e\in P} f_P \le x_e && \forall (u,v)\in E,\ \forall e \in E \\
& \sum_{P\in\mathcal{P}_{u,v}} f_P \ge 1 && \forall (u,v)\in E \\
& x_e \ge 0 && \forall e \in E \\
& f_P \ge 0 && \forall (u,v)\in E,\ \forall P \in \mathcal{P}_{u,v}
\end{aligned}
$$

- Rounding: include *e* with probability $x_e^{1/k}$

LDkS: Upper Bound

- Straightforward LP:

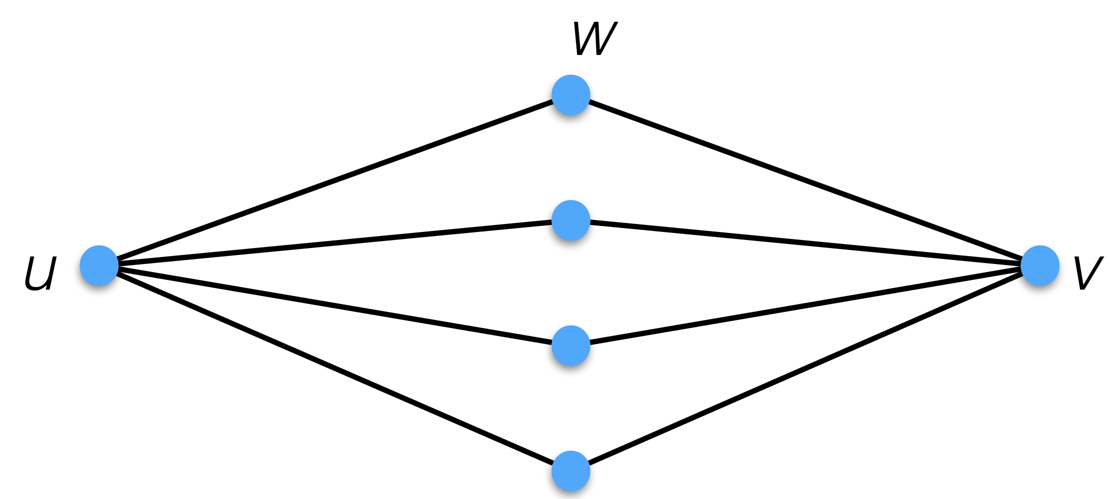
$$
\begin{array}{lll}
\min & d & \\
\text{s.t.} & \displaystyle\sum_{(u,v)\in E} x_{(u,v)} \leq d & \forall u \in V \\
& \displaystyle\sum_{P\in\mathcal{P}_{u,v}:e\in P} f_P \leq x_e & \forall (u,v)\in E,\ \forall e \in E \\
& \displaystyle\sum_{P\in\mathcal{P}_{u,v}} f_P \geq 1 & \forall (u,v) \in E \\
& x_e \geq 0 & \forall e \in E \\
& f_P \geq 0 & \forall (u,v)\in E,\ \forall P \in \mathcal{P}_{u,v}
\end{array}
$$

- Rounding: include *e* with probability $x_e^{1/k}$

- Not hard to analyze cost: trick is proving rounded solution is a *k*-spanner

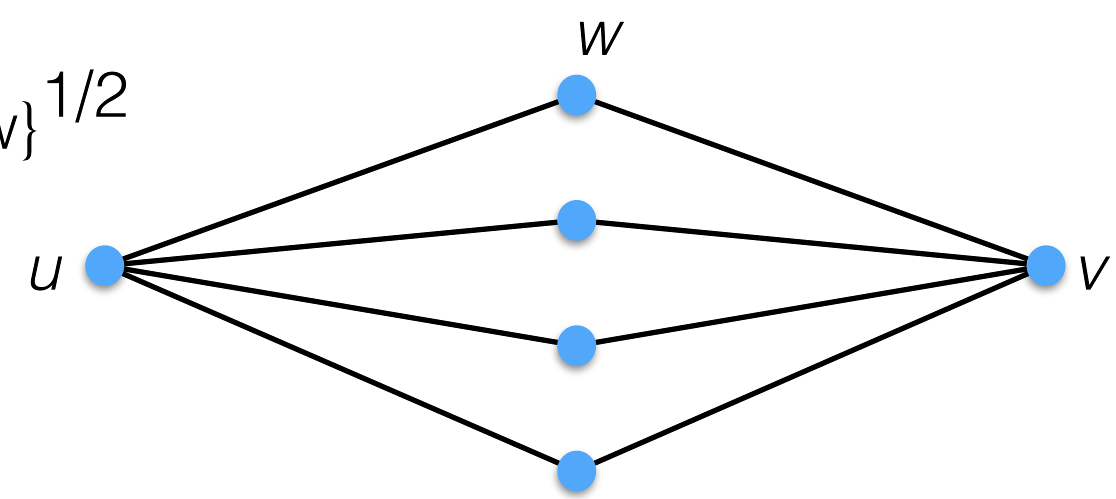
LDkS: Upper Bound (II)

LDkS: Upper Bound (II)



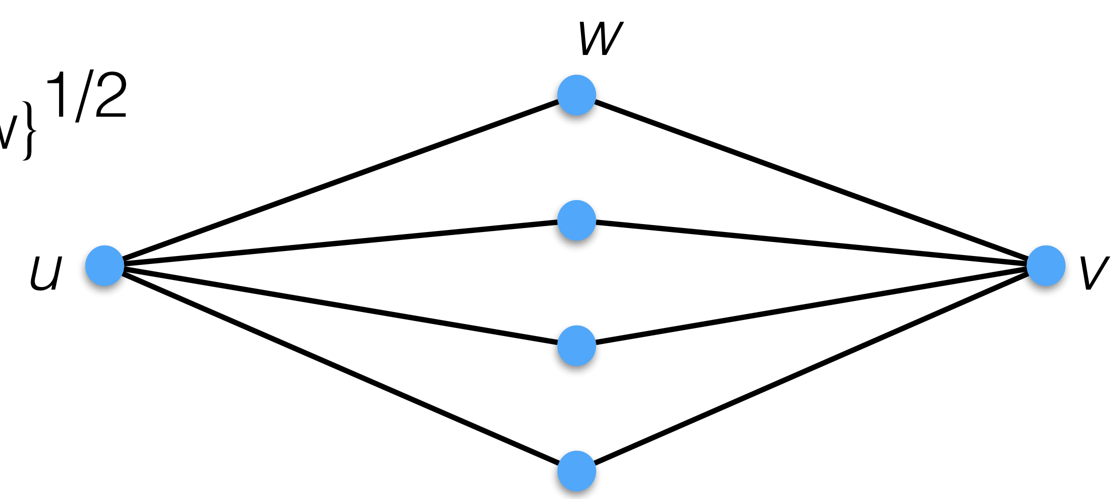
LDkS: Upper Bound (II)

- Pr[get u-w-v] = $x_{\{u,w\}}^{1/2}\, x_{\{v,w\}}^{1/2}$
  $\geq f_{(u,w,v)}$



LDkS: Upper Bound (II)

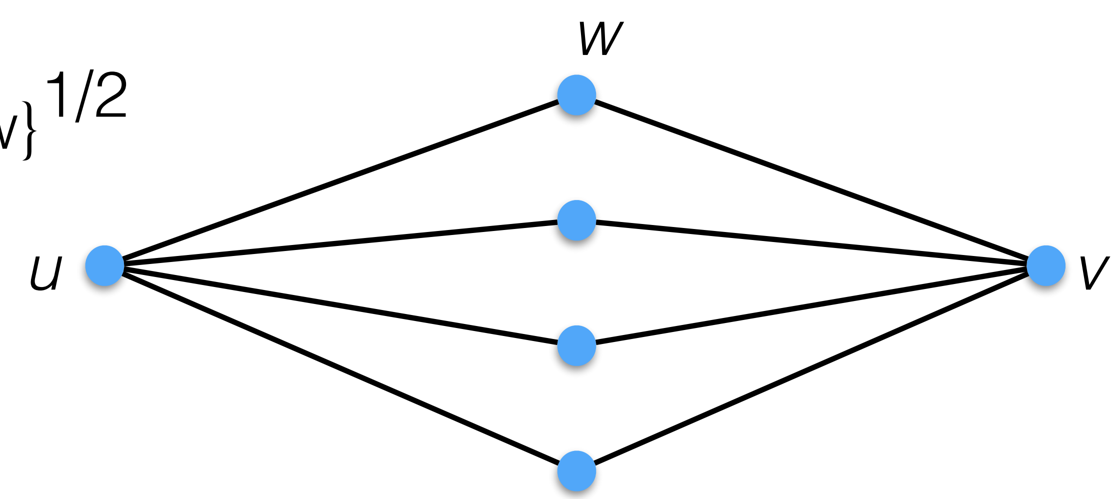
- Pr[get u-w-v] = $x_{\{u,w\}}^{1/2}\, x_{\{v,w\}}^{1/2}$
  $\geq f_{(u,w,v)}$



- If paths disjoint, get each path independently so Pr[span u-v] $\geq 1-\prod_w(1-f_{(u,w,v)}) \geq 1 - 1/e$

LDkS: Upper Bound (II)

- Pr[get u-w-v] = $x_{\{u,w\}}^{1/2} \, x_{\{v,w\}}^{1/2}$
  $\geq f_{(u,w,v)}$



- If paths disjoint, get each path independently so Pr[span u-v] $\geq 1-\prod_w(1-f_{(u,w,v)}) \geq 1 - 1/e$

- But for stretch > 2 paths not disjoint
  - Complicated bucketing of paths to argue lots of flow must be on nearly-disjoint paths

LDkS: Lower Bound

- Based on lower bound for Basic k-Spanner
  - Interlacing of sampling with reduction
  - Start with hard Label Cover instance, subsample edges, apply reduction, subsample edges
- Need girth of Label Cover graph and reduction graph to be larger than k
  - Leads to hardness that gets worse with k

LDkS: Open Question

- What is the right approximation?
- Upper bound $\tilde{O}(\Delta^{(1-1/k)^2})$ gets larger with k, lower bound $\Omega(\Delta^{1/k})$ gets smaller with k

Conclusion

- Tons of work on graph spanners, very little on optimizing/approximating spanners
- We know a fair amount, but still many very basic open questions
  - Beating trivial bound from tradeoff for basic k-spanner?
  - Is any dependence on f necessary for approximating fault-tolerant spanners?
  - Does Lowest Degree k-Spanner get easier or harder as k increases?
  - Approximating weight?
  - Approximating geometric spanners?

Thanks!

(Work on spanners with me!)