Polylog-time and near-linear work approximation scheme for undirected shortest paths

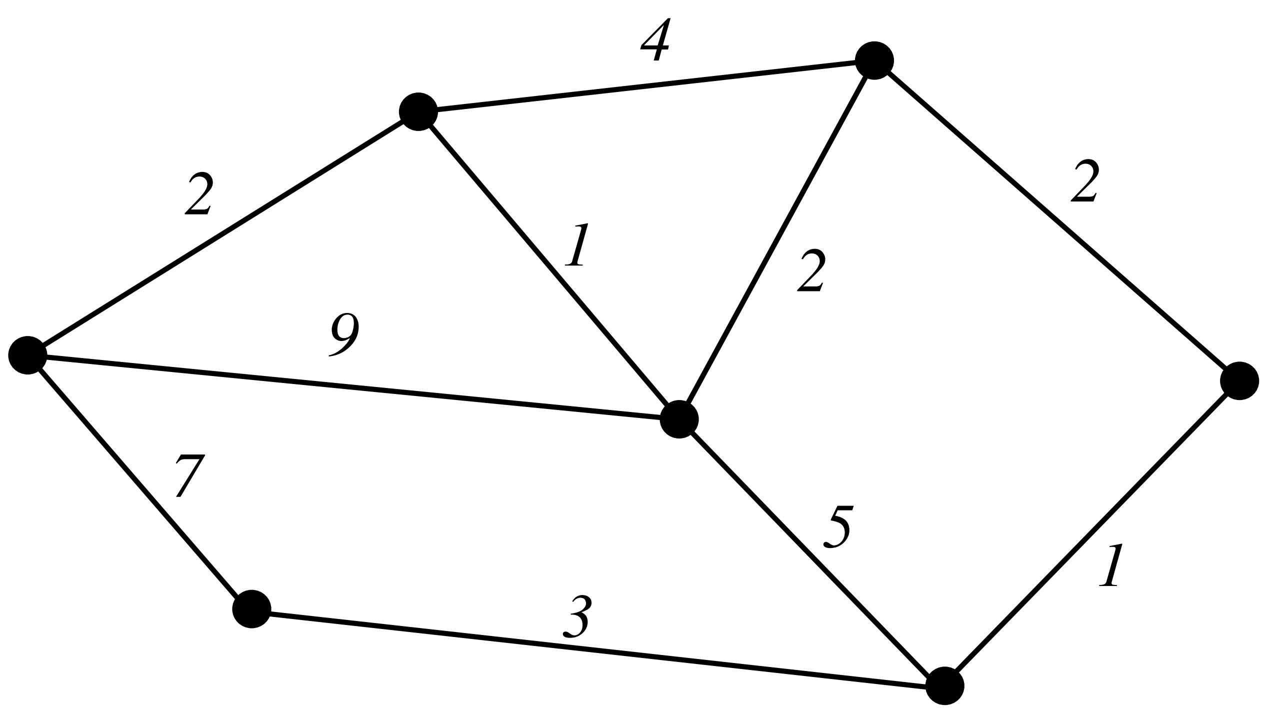
Edith Cohen
AT&T Bell Labs

Shortest-path problem

Network $G = (V, E)$, positive weights $w : E \rightarrow R_+$

$\diamond$ Find minimum-weight paths between:

- designated source node to all other nodes
- all pairs
- specified pairs of nodes



Parallel shortest-paths algorithms
"Transitive-Closure Bottleneck"

s sources, n nodes, m edges

Algorithm	time	work=time×proc.
Dijkstra	$\tilde{O}(n)$	$\tilde{O}(sm)$
Johnson	$\tilde{O}(n)$	$O(nm)$
Floyd-Warshall	polylog(n)	$\tilde{O}(n^3)$
Klein-Sairam**	polylog(n)	$\tilde{O}(sm^2)$ (R)
work/time tradeoffs:		
Spencer*	$\tilde{O}(t)$	$\tilde{O}(s(n^3/t^2 + m))$
Klein-Sairam***	$\tilde{O}(n^{0.5})$	$\tilde{O}(mn^{0.5})$ $(s = n^{0.5})$
C*	$\tilde{O}(t)$	$\tilde{O}(sn^2 + n^3/t^2)$

$*$ if $\max_{e\in E} w(e)/\min_{e\in E} w(e) = O(\text{poly}\, n)$.

otherwise, $(1 + 1/\text{poly}(n))$-approximation

$**$ positive integral polynomial weights

$***$ $(1 + 1/\text{polylog}\, n)$-approximation, randomized

Problem: Faster algorithms perform much more work.

New parallel shortest paths algorithms for weighted undirected networks:

(randomized algorithm)

For any fixed integer k and $\epsilon_0 > 0$:
Paths within $(1+O(1/\log^k n))$ of shortest, from s sources to all other nodes are computed in:
$\diamond$ polylog time using
$\diamond$ $O(mn^{\epsilon_0} + s(m + n^{1+\epsilon_0}))$ work

Improvements:

- Previous polylog-time algorithms require $\min\{O(n^3), \tilde{O}(sm^2)\}$ work.
- Previous near-linear work algorithms require near-$O(n)$ time.
- Best-known sequential time is $\tilde{O}(sm)$.

Faster sequential shortest paths

Paths from s source nodes to all other nodes:

- Upper bound $\tilde{O}(sm)$ (Dijkstra)
- Lower bound $O(m + sn)$
- stretch-t paths ($\leq t\times$shortest):

ABCP	$\tilde{O}(mn^{64/t} + sn^{1+32/t})$
C	$\tilde{O}(mn^{(2+\epsilon)/t} + sn^{1+(2+\epsilon)/t})$

New Algorithm: For any fixed $\epsilon_0 > 0$:
In $O((m + sn)n^{\epsilon_0})$ time computes paths s.t.:

- **Nearby** ($O(w_{\max} \operatorname{polylog} n)$) pairs of nodes: weight $O(w_{\max} \operatorname{polylog} n)$

- **Distant** ($\Omega(w_{\max} \operatorname{polylog} n)$) pairs of nodes: paths within $(1 + 1/\operatorname{polylog} n)$ of shortest.

($w_{\max}$ – maximum edge-weight)

- **In parallel:** randomized polylog time; $O((m + sn)n^{\epsilon_0})$ work

Outline

- **Main result:**
  Parallel shortest paths algorithm

- **Another result:**
  Near-optimal sequential algorithm for "distant" pairs of nodes

- **HopSets** and their use for parallel shortest paths computations

- Flavor of our HopSet constructions

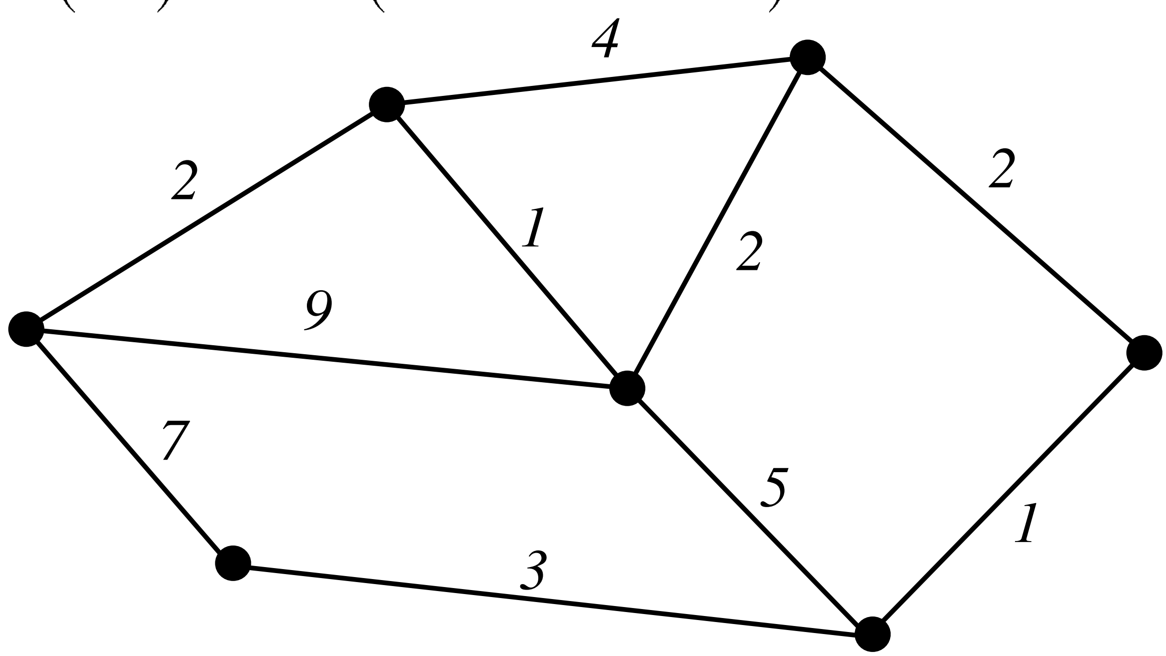
- Open problems

“Reduction” to d-edge shortest paths

d-edge shortest paths are minimum weight paths among paths containing at most d edges.

In parallel, d-edge SP’s can be computed in: $\tilde{O}(d)$ time using

- $O(mds)$ work (parallel Bellman-Ford)
- for $(1 + 1/\operatorname{polylog} n)$-approximation: $\tilde{O}(ms)$ work (Klein-Sairam)



Idea: We compute a sparse collection of new edges E^* (HopSet) such that $(\operatorname{polylog} n)$-edge distances in $E \cup E^*$ are within $(1 + \epsilon)$ of original distances.

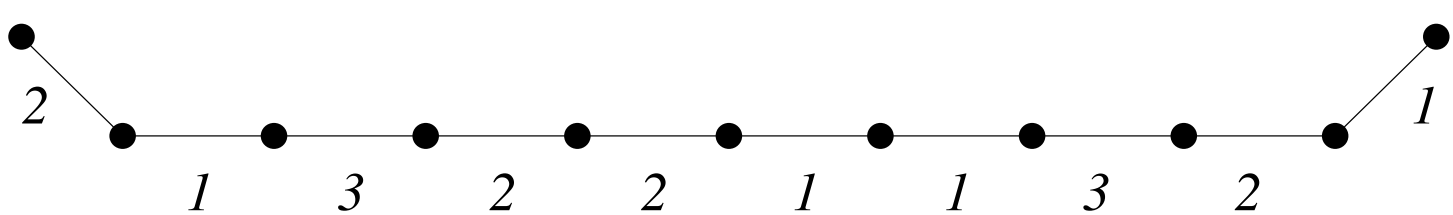
(d, ϵ)- HopSet

Network $G = (V, E)$, integer $d \geq 1$, scalar $\epsilon > 0$

A (d, ϵ)- HopSet of G is a set E^* of weighted edges such that:

1. $\text{dist}_{E \cup E^*}(u_1, u_2) = \text{dist}_E(u_1, u_2)$

2. $\text{dist}^d_{E \cup E^*}(u_1, u_2) \leq (1 + \epsilon)\text{dist}_E(u_1, u_2)$



Good HopSets

We want:

1. A sparse hopset (close to $O(m)$ edges)

2. Small diameter ($d = O(\text{polylog}\, n)$)

3. Good approximation ($\epsilon \leq 1\%$)

• The **existence** of hopsets with some specified attributes is of independent interest.

• We also want **efficient** constructions.

Our HopSets

For any fixed integer k and $\epsilon_0 > 0$:

size	ϵ (approx.)	d (diameter)
$O(n^{1+\epsilon_0})$	$O(1/\log^k n)$	$\text{polylog}\, n$

◇ In: $O(mn^{\epsilon_0})$ time

◇ In: polylog time using $O(mn^{\epsilon_0})$ work

Outline:
Flavor of our HopSet constructions

- Review of "pairwise covers"
  (used in our HopSets constructions)

- A simple, sequential, construction of
  $(O(\epsilon^{-1}\log n),\epsilon)$-HopSets of size $\tilde{O}(n^{4/3})$
  In time: $\tilde{O}(mn^{2/3})$

Sketch of further ideas:

- Sequential constructions of sparser HopSets, faster

- Parallel HopSet constructions:
  The parallel cover constructions of [C93] are instrumental.
  ⋄ using limited covers to obtain limited HopSets
  ⋄ using limited HopSets to obtain HopSets

Pairwise covers

Network with weights $w : E \to \mathcal{R}_+$, scalar $W \geq 1$

$\diamond$ A W**-cover** of G is a collection of:

- subsets of nodes $X_1, \ldots, X_k$ (clusters), and
- nodes $v_1, \ldots, v_k$ where $v_i \in X_i$ (centers)

such that:

1. For every path p such that $w(p) \leq W$, $\exists i$ such that $p \subset X_i$
2. $\forall i$, $\forall\ u \in X_i$, $\text{dist}\{v_i, u\} \leq W \lceil \log n \rceil$
3. $\Sigma_i |X_i| = \tilde{O}(n)$.
4. $\Sigma_i |E \cap X_i \times X_i| = \tilde{O}(m)$.

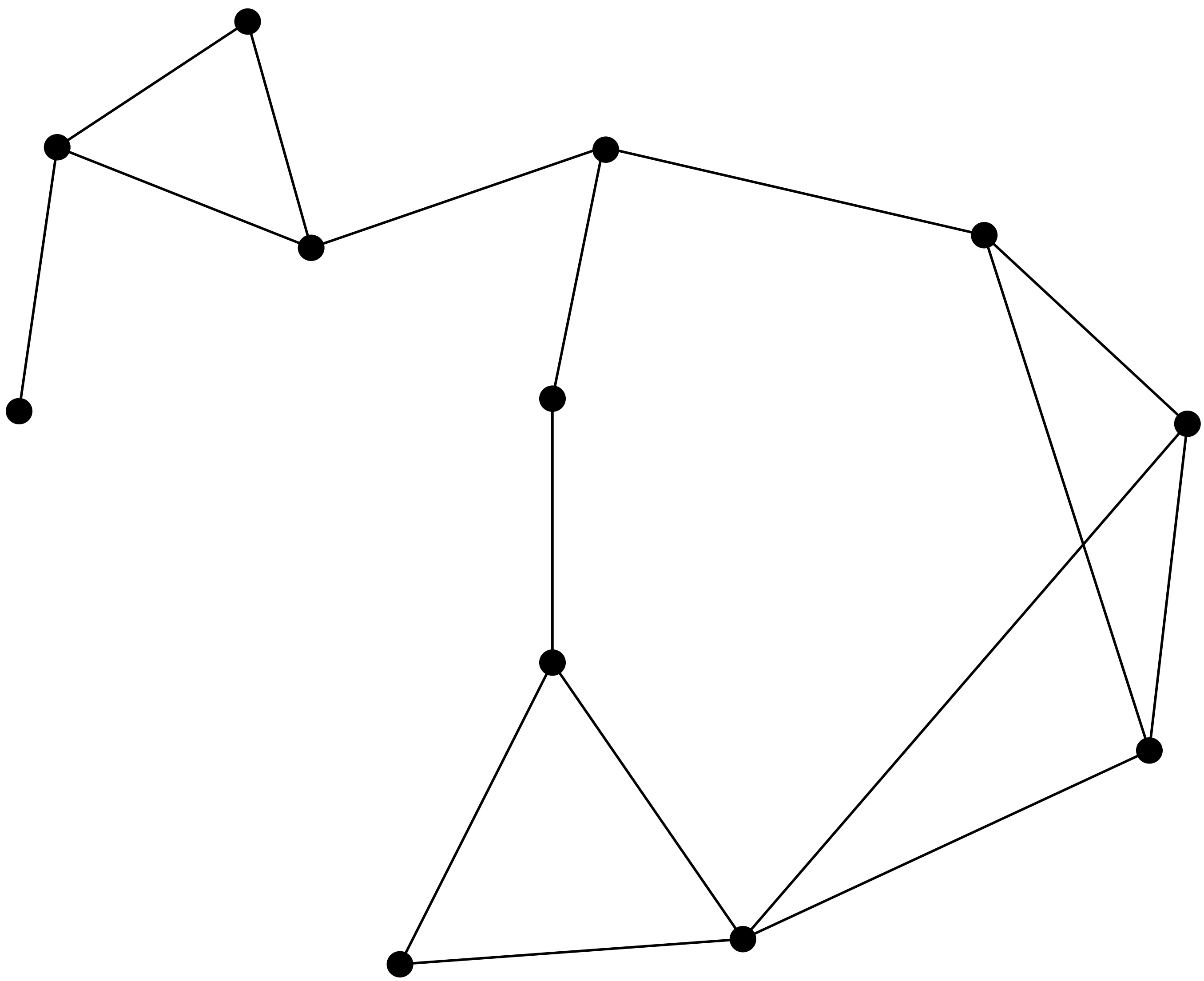
Complexity:

- Sequentially: $\tilde{O}(m)$ time [ABCP93] [C93]
- In parallel: (ℓ-limited covers)

$\tilde{O}(\ell)$ expected time $\tilde{O}(m)$ work [C93]

Example: 1-cover

3 clusters: X_1, X_2, X_3, $W = 1$, radius $= 2$



$n = 12$, $m = 16$

$\Sigma_i \, |X_i| = 5 + 4 + 7 = 16$, $\Sigma_i \, |E \cap X_i \times X_i| = 17$

Simple HopSet algorithm

- In time: $\tilde{O}(mn^{2/3})$
- Computes $(O(\epsilon^{-1}\log n), \epsilon)$-HopSet
- of size $\tilde{O}(n^{4/3})$.

Algorithm: HopSet for distances in $[R, 2R]$:

1. $W = \epsilon R/(4\lceil\log n\rceil)$. Construct a W-cover χ.
   Big clusters: $X \in \chi$ such that $|X| > n^{1/3}$
   Small clusters: $X \in \chi$ such that $|X| \leq n^{1/3}$

2. For each small cluster:
   a complete set of edges

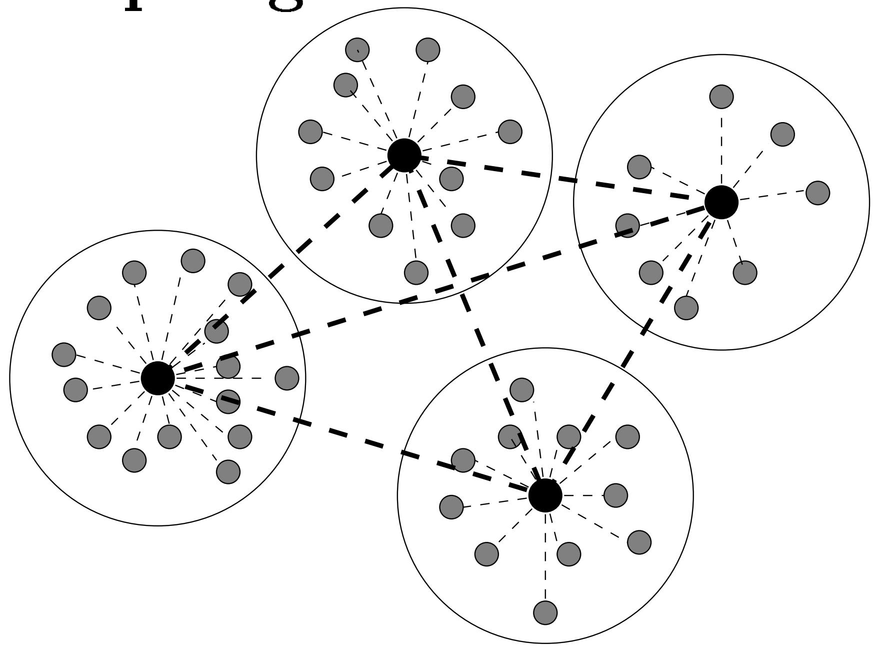
3. For each big cluster:
   star graph rooted at the center

4. Complete graph on centers of big clusters

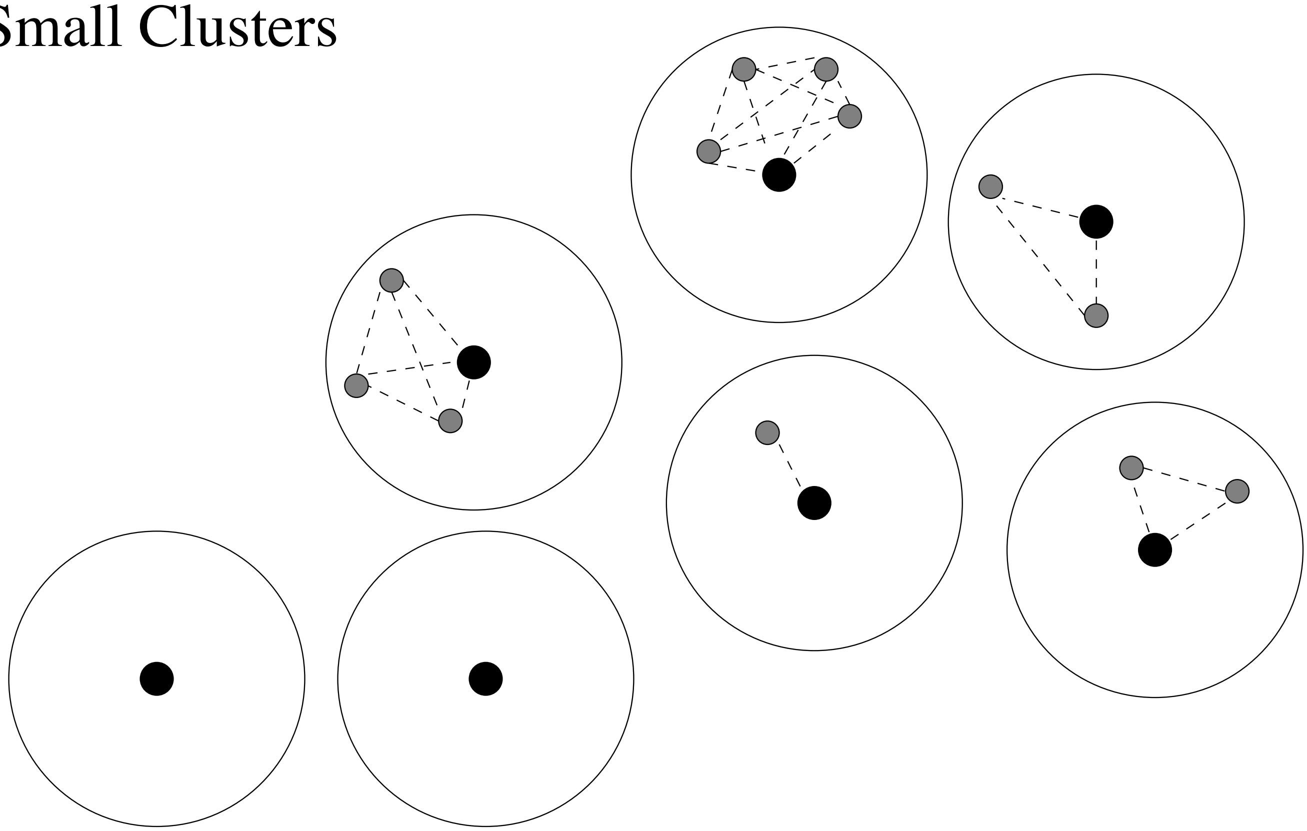
The assigned edge weights are the distances.

Hop edges

Big Clusters



Small Clusters



Size of the HopSet

We bound the number of hop edges produced.

- Complete graphs on small clusters:
  For each small cluster X, $O(|X|^2)$ edges.
  We have $|X| \leq n^{1/3}$ and $\sum_{X \in \chi} |X| = \tilde{O}(n)$.
  Hence,

$$\sum_{X \text{ is small}} |X|^2 \leq \tilde{O}(n^{2/3}) n^{2/3} = \tilde{O}(n^{4/3})$$

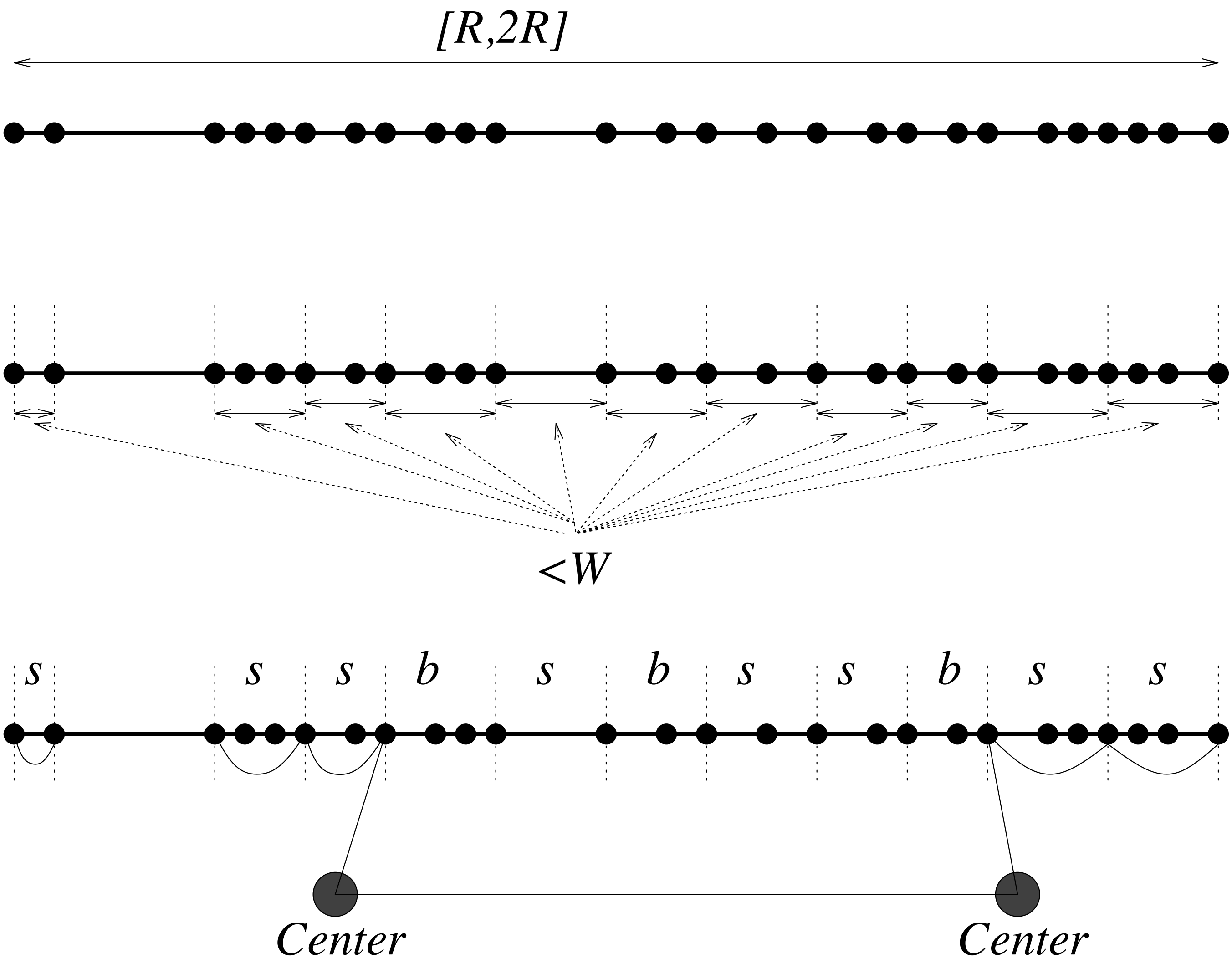
- Star graphs on big clusters: $\tilde{O}(n)$

- Complete graph on centers of big clusters:
  There are $\tilde{O}(n^{2/3})$ big clusters. Hence:
  $\tilde{O}(n^{4/3})$

Total number of hop edges: $\tilde{O}(n^{4/3})$

Correctness

We show that the algorithm produces a $(O(\epsilon^{-1} \log n), \epsilon)$-HopSet.
Consider a path of weight in $[R, 2R]$.
$W = \epsilon R/(4\lceil \log n \rceil)$.



Size of the path is $O(R/W) = O(\epsilon^{-1} \log n)$
Weight is larger by at most $4W\lceil \log n \rceil \leq \epsilon R$

Computing better HopSets

To produce sparser HopSets (size $O(n^{1+\epsilon_0})$) more efficiently (time $O(mn^{\epsilon_0})$) we use recursive version of the algorithm.

Sketch:

Big Clusters: size $> n^{1-\epsilon_0}$
Small clusters: size $\leq n^{1-\epsilon_0}$

- Big clusters are treated the same.
- For small clusters, instead of a complete graph (and all-pairs shortest paths computations), we apply the algorithm recursively.

Computing HopSets in parallel

- **limited covers** can be computed efficiently in parallel [C]
- using limited covers in the HopSet algorithm produces **limited HopSets**
- HopSets can be obtained by applying $O(\log n)$ times a limited HopSet algorithm

ℓ-limited covers in parallel

Network $G = (V, E)$ with weights $w : E \to \mathcal{R}_+$, a scalar $W \geq 1$, an integer $\ell > 1$

$\diamond$ An **ℓ-limited W-cover** of G is a collection of:

- subsets of nodes $X_1, \ldots, X_k$ (clusters), and
- nodes $v_1, \ldots, v_k$ where $v_i \in X_i$ (centers)

such that:

1. Every path p such that $|p| \leq \ell$ and $w(p) \leq W$, $\exists i$ such that $p \subset X_i$

2. $\forall i$, $\forall\ u \in X_i$, $\text{dist}\{v_i, u\} \leq W \lceil \log n \rceil$

3. $\sum_i |X_i| = \tilde{O}(n)$.

4. $\sum_i |E \cap X_i^2| = \tilde{O}(m)$.

Complexity: $\tilde{O}(\ell)$ expected time $\tilde{O}(m)$ work [C93]

ℓ-limited (d, ϵ)- HopSet

Integers ℓ, d, scalar $\epsilon > 0$
An ℓ-limited (d, ϵ)- HopSet of $G = (V, E)$ is a set E^* of weighted edges such that:

1. $\text{dist}_{E \cup E^*}(u_1, u_2) = \text{dist}_E(u_1, u_2)$

2. $\text{dist}^d_{E \cup E^*}(u_1, u_2) \leq (1 + \epsilon)\text{dist}^\ell_E(u_1, u_2)$

- The algorithm is such that d is independent of our choice of ℓ.

- The running time is linear in ℓ.

We will use $\ell = 2d$

HopSets in parallel

Consider a weighted network (V, E)

- $\ell \leftarrow 2d,\ E^{**} = \emptyset$
- For $i = 1, \ldots, \log n$:
  1. Compute ℓ-limited (d, ϵ)-HopSet E^{*} for $(V, E \cup E^{**})$
  2. $E^{**} \leftarrow E^{**} \cup E^{*}$

Correctness:
After iteration i, E^{**} constitutes:
a $2^{i}d$-limited (d, ϵ_i)-HopSet of (V, E),
where $(1 + \epsilon_i) = (1 + \epsilon)^{i}$.

We choose $\epsilon << 1/\log^2 n$ hence $\epsilon_i = O(1/\log n)$.

Open Problems

- Overcoming the transitive closure bottleneck for directed networks

- Existence of sparse HopSets for:
  - ⋄ exact distances?
  - ⋄ directed networks?

- Better sequential ($(1 + \epsilon)$-approx) shortest paths:
  Upper bound: $O(sm)$ for s sources
  Lower bound: $O(m + sn)$ for s sources