Approximate All-Pairs Shortest Paths in the Congested Clique Model

Yi-Jun Chang

National University of Singapore

The talk is based on the paper:

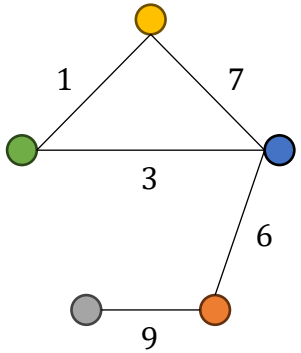
"Improved all-pairs approximate shortest paths in congested clique"

[arXiv:2405.02695, PODC 2024, Distributed Computing 2026]

by Hong Duc Bui, Shashwat Chandra, Yi-Jun Chang, Michal Dory, and Dean Leitersdorf

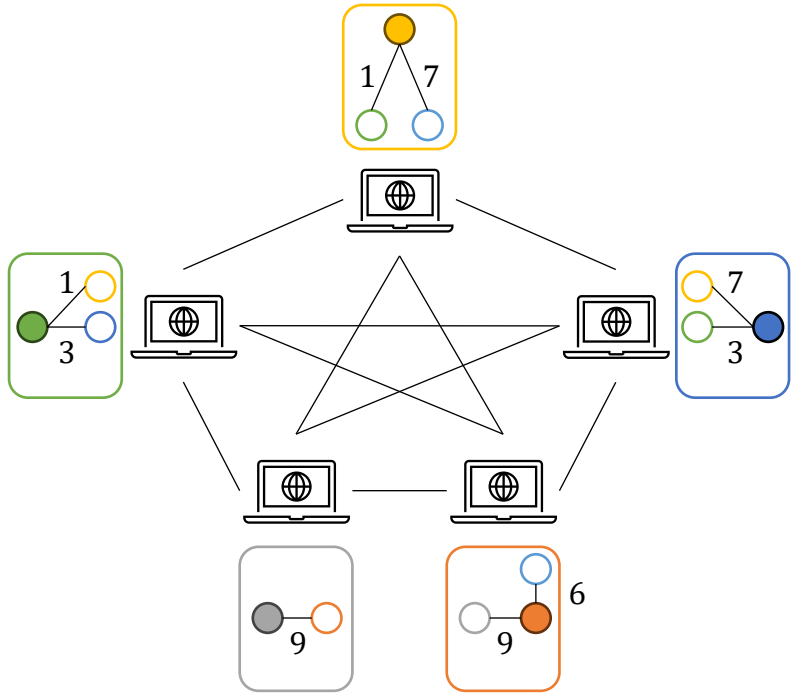
The congested clique model

Input: n-vertex graph $G = (V, E)$.

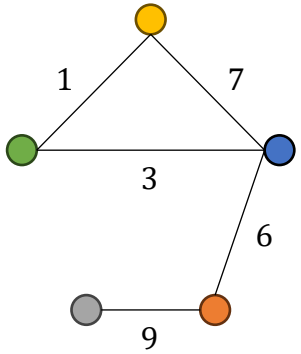


Network:

- Each vertex is a machine.
- Each vertex is initially aware of its incident edges.
- All-to-all communication.



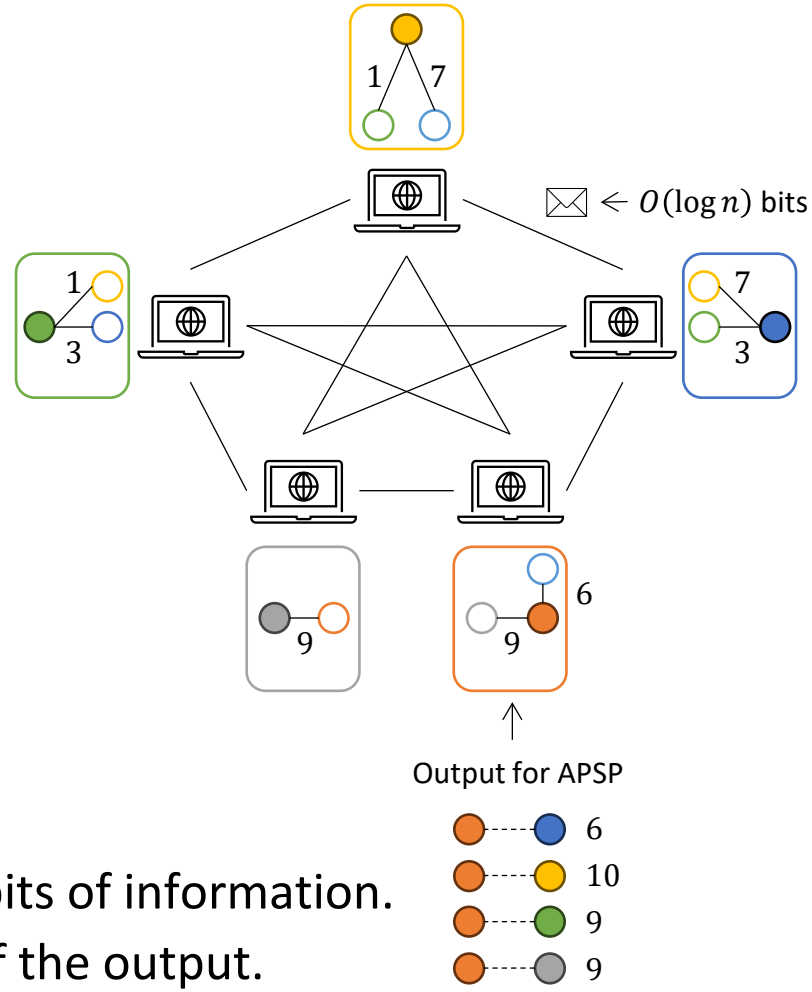
The congested clique model



Input: n-vertex graph $G = (V, E)$.

Network:

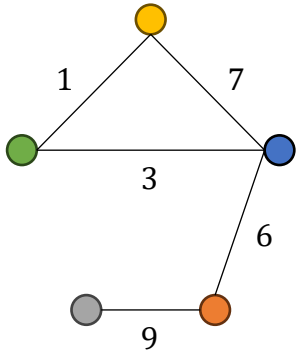
- Each vertex is a machine.
- Each vertex is initially aware of its incident edges.
- All-to-all communication.



Communication:

- In each round, every pair of vertices can exchange $O(\log n)$ bits of information.
- At the end of the computation, each vertex returns its part of the output.

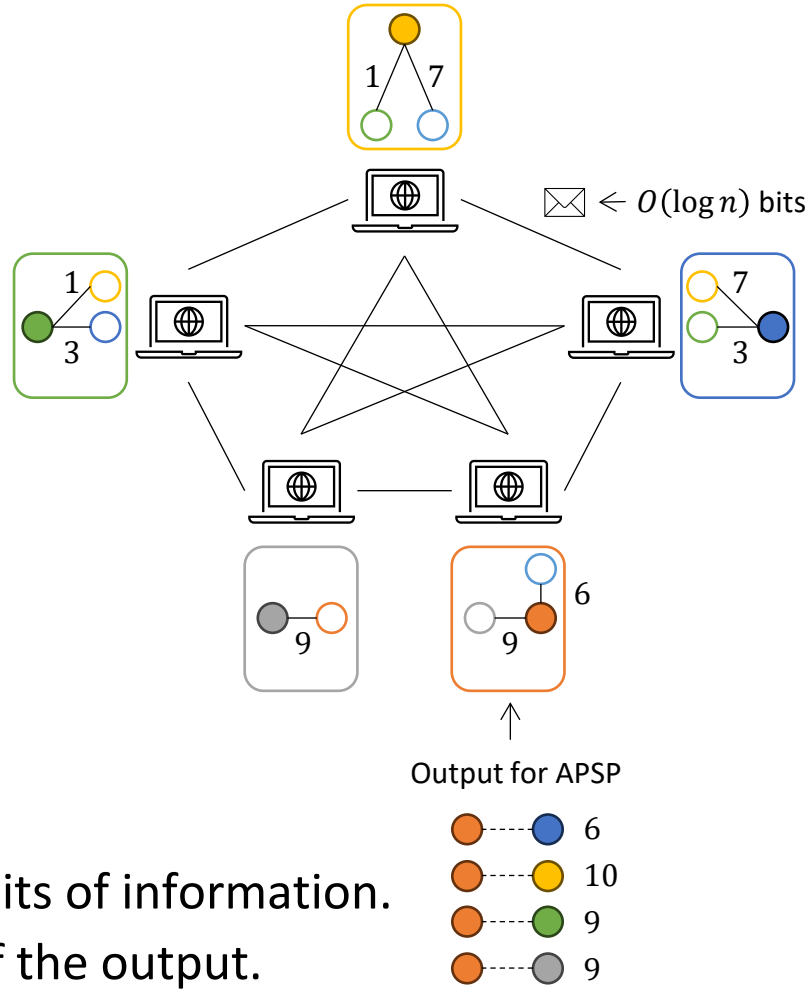
The congested clique model



Input: n-vertex graph $G = (V, E)$.

Network:

- Each vertex is a machine.
- Each vertex is initially aware of its incident edges.
- All-to-all communication.



Communication:

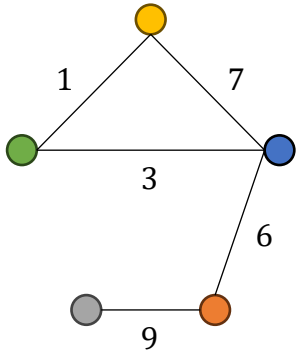
- In each round, every pair of vertices can exchange $O(\log n)$ bits of information.
- At the end of the computation, each vertex returns its part of the output.

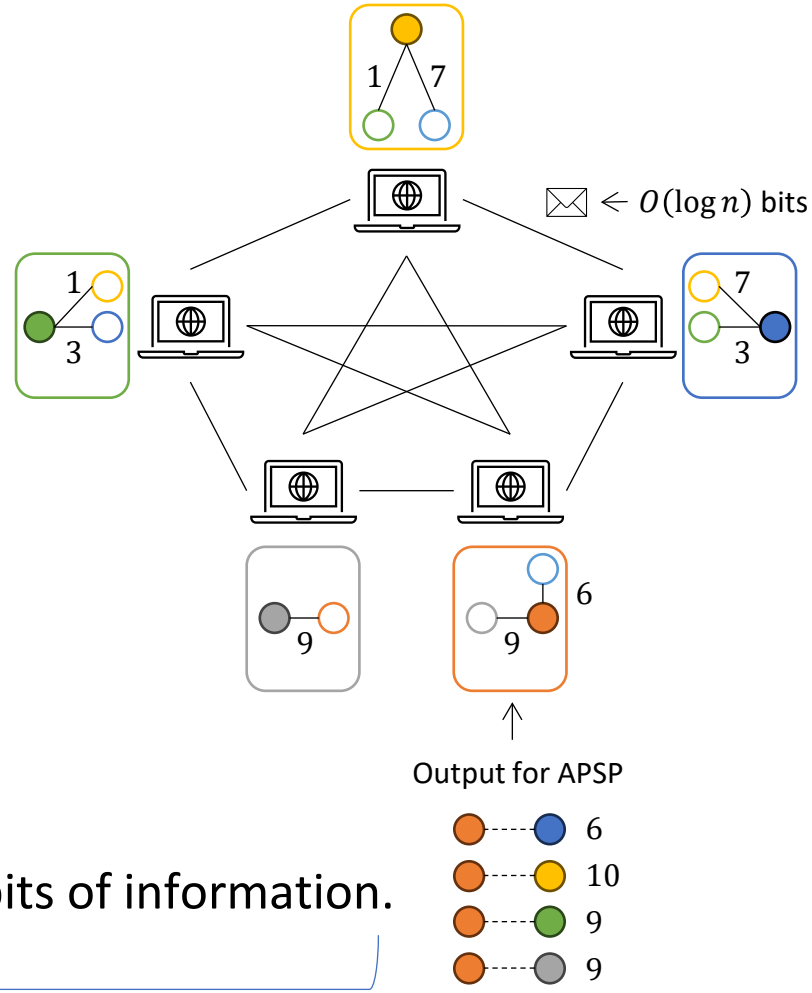
Complexity measure: The number of rounds.

What is the round complexity for All-Pairs Shortest Paths (APSP)?

The congested clique model

Input: n-vertex graph $G = (V, E)$.





Network:

- Each vertex is a machine.
- Each vertex is initially aware of its incident edges.
- All-to-all communication.

Communication:

- In each round, every pair of vertices can exchange $O(\log n)$ bits of information.

Equivalent definition: Each vertex can send and receive $O(n)$ messages of $O(\log n)$ bits.

by Lenzen's routing

Prior work

Weighted APSP in $n^{O(1)}$ rounds

Censor-Hillel, Kaski, Korhonen, Lenzen, Paz, and Suomela (PODC 2015, Distributed Computing 2019)

- $\tilde{O}(n^{1/3})$ rounds for weighted APSP.
- $O(n^{0.158})$ rounds for unweighted APSP.

Prior work

Weighted APSP in $n^{O(1)}$ rounds

Censor-Hillel, Kaski, Korhonen, Lenzen, Paz, and Suomela (PODC 2015, Distributed Computing 2019)

- $\tilde{O}(n^{1/3})$ rounds for weighted APSP.
- $O(n^{0.158})$ rounds for unweighted APSP.

Technique: Reduce to min-plus or standard matrix multiplication.

Prior work

Weighted APSP in $n^{O(1)}$ rounds

Censor-Hillel, Kaski, Korhonen, Lenzen, Paz, and Suomela (PODC 2015, Distributed Computing 2019)

▽

$O(1)$-approximate weighted APSP in $\log^{O(1)} n$ rounds

Censor-Hillel, Dory, Korhonen, and Leitersdorf (PODC 2019 best student paper, Distributed Computing 2021)

Definition of α-approximate APSP:

- Cannot underestimate distances.
- Can overestimate distances by a factor of at most α.

Prior work

Weighted APSP in $n^{O(1)}$ rounds

Censor-Hillel, Kaski, Korhonen, Lenzen, Paz, and Suomela (PODC 2015, Distributed Computing 2019)

▽

$O(1)$-approximate weighted APSP in $\log^{O(1)} n$ rounds

Censor-Hillel, Dory, Korhonen, and Leitersdorf (PODC 2019 best student paper, Distributed Computing 2021)

Definition of α-approximate APSP:

- Cannot underestimate distances.
- Can overestimate distances by a factor of at most α.

Technique: Build a **hopset**.

Adding edges in such a way that $O(1)$-approximate distances can be learned within $O(\log n)$ hops.

+ **sparse** matrix multiplication

Prior work

Weighted APSP in $n^{O(1)}$ rounds

Censor-Hillel, Kaski, Korhonen, Lenzen, Paz, and Suomela (PODC 2015, Distributed Computing 2019)

▽

$O(1)$-approximate weighted APSP in $\log^{O(1)} n$ rounds

Censor-Hillel, Dory, Korhonen, and Leitersdorf (PODC 2019 best student paper, Distributed Computing 2021)

▽

$O(1)$-approximate unweighted APSP in $\log^{O(1)} \log n$ rounds

Dory and Parter (PODC 2020 best paper, JACM 2022)

Prior work

Weighted APSP in $n^{O(1)}$ rounds

Censor-Hillel, Kaski, Korhonen, Lenzen, Paz, and Suomela (PODC 2015, Distributed Computing 2019)

▽

$O(1)$-approximate weighted APSP in $\log^{O(1)} n$ rounds

Censor-Hillel, Dory, Korhonen, and Leitersdorf (PODC 2019 best student paper, Distributed Computing 2021)

▽

$O(1)$-approximate unweighted APSP in $\log^{O(1)} \log n$ rounds

Dory and Parter (PODC 2020 best paper, JACM 2022)

▽

$O(\log n)$-approximate weighted APSP in $O(1)$ rounds.

- via efficient construction of multiplicative spanners

Dory, Fischer, Khoury, and Leitersdorf (PODC 2021)

Chechik and Zhang (PODC 2022)

Prior work

Weighted APSP in $n^{O(1)}$ rounds

Censor-Hillel, Kaski, Korhonen, Lenzen, Paz, and Suomela (PODC 2015, Distributed Computing 2019)

▽

$O(1)$-approximate weighted APSP in $\log^{O(1)} n$ rounds

Censor-Hillel, Dory, Korhonen, and Leitersdorf (PODC 2019 best student paper, Distributed Computing 2021)

▽

$O(1)$-approximate unweighted APSP in $\log^{O(1)} \log n$ rounds

Dory and Parter (PODC 2020 best paper, JACM 2022)

▽

$O(\log n)$-approximate weighted APSP in $O(1)$ rounds.

- via efficient construction of multiplicative spanners

Dory, Fischer, Khoury, and Leitersdorf (PODC 2021)

Chechik and Zhang (PODC 2022)

↑

A spanner with $O(\log n)$ stretch has $O(n)$ edges, so it can be broadcast to the entire network in $O(1)$ rounds.

Our results

New: $O(1)$-approximate weighted APSP in $O(\log\log\log n)$ rounds.

Weighted APSP in $n^{O(1)}$ rounds

Censor-Hillel, Kaski, Korhonen, Lenzen, Paz, and Suomela (PODC 2015, Distributed Computing 2019)

▽

$O(1)$-approximate weighted APSP in $\log^{O(1)} n$ rounds

Censor-Hillel, Dory, Korhonen, and Leitersdorf (PODC 2019 best student paper, Distributed Computing 2021)

▽

$O(1)$-approximate unweighted APSP in $\log^{O(1)} \log n$ rounds

Dory and Parter (PODC 2020 best paper, JACM 2022)

▽

$O(\log n)$-approximate weighted APSP in $O(1)$ rounds.

- via efficient construction of multiplicative spanners

Dory, Fischer, Khoury, and Leitersdorf (PODC 2021)

Chechik and Zhang (PODC 2022)

Our results

New: $O(1)$-approximate weighted APSP in $O(\log\log\log n)$ rounds.

Doubly exponential improvement

Exponential improvement

$O(1)$-approximate weighted APSP in $\log^{O(1)} n$ rounds

Censor-Hillel, Dory, Korhonen, and Leitersdorf (PODC 2019 best student paper, Distributed Computing 2021)

▽

$O(1)$-approximate unweighted APSP in $\log^{O(1)} \log n$ rounds

Dory and Parter (PODC 2020 best paper, JACM 2022)

▽

$O(\log n)$-approximate weighted APSP in $O(1)$ rounds.

- via efficient construction of multiplicative spanners

Dory, Fischer, Khoury, and Leitersdorf (PODC 2021)

Chechik and Zhang (PODC 2022)

Our results

New: $O(1)$-approximate weighted APSP in $O(\log\log\log n)$ rounds.

New: $O\left(\log^{2^{-t}} n\right)$-approximate weighted APSP in $O(t)$ rounds.

$O(\log n)$-approximate weighted APSP in $O(1)$ rounds.

- via efficient construction of multiplicative spanners

Dory, Fischer, Khoury, and Leitersdorf (PODC 2021)

Chechik and Zhang (PODC 2022)

Our results

New: $O(1)$-approximate weighted APSP in $O(\log\log\log n)$ rounds.

New: $O\left(\log^{2^{-t}} n\right)$-approximate weighted APSP in $O(t)$ rounds.

- $O(\log^{\epsilon} n)$-approximation in $O(1)$ rounds, for any constant $\epsilon > 0$.



Polynomial improvement

$O(\log n)$-approximate weighted APSP in $O(1)$ rounds.

- via efficient construction of multiplicative spanners

Dory, Fischer, Khoury, and Leitersdorf (PODC 2021)

Chechik and Zhang (PODC 2022)

For simplicity, here we assume the underlying graph is **unweighted**.

Skeleton graph

A standard technique for parallel shortest paths:

- **Centers:** Sample $\widetilde{\Theta}\left(\frac{n}{k}\right)$ vertices.
- **Skeleton graph:**
  - For any two centers u and v with $d(u,v) \leq k$, add an edge $\{u,v\}$ of weight $d(u,v)$.
- How to approximate $d(u,v)$?

For simplicity, here we assume the underlying graph is **unweighted**.

Skeleton graph

A standard technique for parallel shortest paths:

- **Centers:** Sample $\widetilde{\Theta}\left(\frac{n}{k}\right)$ vertices.
- **Skeleton graph:**
  - For any two centers u and v with $d(u,v) \leq k$, add an edge $\{u,v\}$ of weight $d(u,v)$.
- How to approximate $d(u,v)$?
  - Case: $d(u,v) \leq k$.
    - Still need to compute it from scratch...
  - Case: $d(u,v) > k$.
    - $d\big(u,c(u)\big) + d\big(c(u),c(v)\big) + d(c(v),v)$ is an $O(1)$–approximation.

Distance in the skeleton graph

$c(v)$ is the center closest to vertex v.

Cost of implementation

- **Issue:** Need to compute $d(u, v)$ for any u and v with $d(u, v) \leq k$.
  - This is very costly.
  - E.g., this technique is useless for small-diameter graphs, where APSP computation is still challenging.

Cost of implementation

- **Issue:** Need to compute $d(u, v)$ for any u and v with $d(u, v) \leq k$.
  - This is very costly.
  - E.g., this technique is useless for small-diameter graphs, where APSP computation is still challenging.

- **Solution:** A variant of skeleton graph.
  - Here, all we need is the k-nearest neighborhood of each vertex.
    - $d(u, v)$ for any u and $v \in k$-nearest of u.

New skeleton graph

- **Centers:** Sample $\widetilde{\Theta}\left(\frac{n}{k}\right)$ vertices.

$c(v)$ is the center closest to vertex v.

- **Skeleton graph:**
  - for any vertex u,
    - for any $t \in k$-nearest of u,
      - for any v adjacent to t,
        - add an edge $\{c(u), c(v)\}$ of weight $d(c(u), u) + d(u, t) + d(t, v) + d\big(v, c(v)\big)$.

Intuition: We do two steps of exploration.

$u \longrightarrow t \longrightarrow v$

$t \in k$-nearest of u v is adjacent to t

New skeleton graph

- **Centers:** Sample $\widetilde{\Theta}\left(\frac{n}{k}\right)$ vertices.

$c(v)$ is the center closest to vertex v.

- **Skeleton graph:**
  - for any vertex u,
    - for any $t \in k$-nearest of u,
      - for any v adjacent to t,
        - add an edge $\{c(u), c(v)\}$ of weight $d(c(u), u) + d(u, t) + d(t, v) + d\big(v, c(v)\big)$.

$O(1)$ rounds in congested clique, given that each vertex u knows $d(u, v)$ for all $v \in k$-nearest of u.

by sparse matrix multiplication

New skeleton graph

- **Centers:** Sample $\widetilde{\Theta}\left(\frac{n}{k}\right)$ vertices.

$c(v)$ is the center closest to vertex v.

- **Skeleton graph:**
  - (…)
- How to approximate $d(u, v)$?
  - Case: ($u \in k$-nearest of v) or ($v \in k$-nearest of u).
    - We already know $d(u, v)$.
  - Remaining case.
    - $d\big(u, c(u)\big) + d\big(c(u), c(v)\big) + d(c(v), v)$ is an $O(1)$-approximation.

Distance in the skeleton graph

High-level plan

- Successive improvement in the approximation ratio:

Starting point: $O(1)$ rounds — Chechik and Zhang (PODC 2022)

- $O(\log n) \longrightarrow O\left(\sqrt{\log n}\right) \longrightarrow O\left(\log^{1/4} n\right) \longrightarrow O\left(\log^{1/8} n\right) \longrightarrow \cdots \longrightarrow O(1)$

High-level plan

- Successive improvement in the approximation ratio:

Starting point: $O(1)$ rounds — Chechik and Zhang (PODC 2022)

- $O(\log n) \longrightarrow O\left(\sqrt{\log n}\right) \longrightarrow O\left(\log^{1/4} n\right) \longrightarrow O\left(\log^{1/8} n\right) \longrightarrow \cdots \longrightarrow O(1)$

Our plan: $O(1)$ rounds, $O(1)$ rounds, $O(1)$ rounds, $O(1)$ rounds, $O(1)$ rounds

$O(\log \log \log n)$ rounds

High-level plan

- Successive improvement in the approximation ratio:

We focus on this step.

- $O(\log n) \longrightarrow O\left(\sqrt{\log n}\right) \longrightarrow O\left(\log^{1/4} n\right) \longrightarrow O\left(\log^{1/8} n\right) \longrightarrow \cdots \longrightarrow O(1)$

  $O(1)$ rounds

Reduction

- **Goal:** $O(\log n) \longrightarrow O\left(\sqrt{\log n}\right)$ in $O(1)$ rounds.

Each vertex computes its $2^{\Theta\left(\sqrt{\log n}\right)}$-nearest neighborhood.

▽

A skeleton graph with $\frac{n}{2^{\Theta\left(\sqrt{\log n}\right)}}$ vertices can be computed in $O(1)$ rounds.

Reduction

- **Goal:** $O(\log n) \longrightarrow O\left(\sqrt{\log n}\right)$ in $O(1)$ rounds.

Each vertex computes its $2^{\Theta\left(\sqrt{\log n}\right)}$-nearest neighborhood.

▽

A skeleton graph with $\frac{n}{2^{\Theta\left(\sqrt{\log n}\right)}}$ vertices can be computed in $O(1)$ rounds.

▽

An $O\left(\sqrt{\log n}\right)$-spanner of the skeleton graph can be computed in $O(1)$ rounds. ← The spanner has $O(n)$ edges.

Chechik and Zhang (PODC 2022)

Reduction

- **Goal:** $O(\log n) \longrightarrow O\left(\sqrt{\log n}\right)$ in $O(1)$ rounds.

Each vertex computes its $2^{\Theta\left(\sqrt{\log n}\right)}$-nearest neighborhood.

▽

A skeleton graph with $\frac{n}{2^{\Theta\left(\sqrt{\log n}\right)}}$ vertices can be computed in $O(1)$ rounds.

▽

An $O\left(\sqrt{\log n}\right)$-spanner of the skeleton graph can be computed in $O(1)$ rounds. ← The spanner has $O(n)$ edges.

▽

$O\left(\sqrt{\log n}\right)$-approximation of APSP in $O(1)$ rounds. ← Broadcasting the spanner to the entire network.

Reduction

- **Goal:** $O(\log n) \longrightarrow O\left(\sqrt{\log n}\right)$ in $O(1)$ rounds.

Each vertex computes its $2^{\Theta\left(\sqrt{\log n}\right)}$-nearest neighborhood. ← **Just need to do this** in $O(1)$ rounds.

▽

A skeleton graph with $\frac{n}{2^{\Theta\left(\sqrt{\log n}\right)}}$ vertices can be computed in $O(1)$ rounds.

▽

An $O\left(\sqrt{\log n}\right)$-spanner of the skeleton graph can be computed in $O(1)$ rounds.

▽

$O\left(\sqrt{\log n}\right)$-approximation of APSP in $O(1)$ rounds.

Hopset construction

- **Goal:** Each vertex computes its $2^{\Theta(\sqrt{\log n})}$-nearest neighborhood.

- **Tool:** k-nearest β-hopset.
  - Addition of new edges to the graph so that the k-nearest neighborhood of each vertex is reachable within (a shortest path of) β hops.
    - The new edges should not change any shortest-path distance.

Hopset construction

- **Goal:** Each vertex computes its $2^{\Theta(\sqrt{\log n})}$-nearest neighborhood.

- **Tool:** k-nearest β-hopset.
  - Addition of new edges to the graph so that the k-nearest neighborhood of each vertex is reachable within (a shortest path of) β hops.

Claim: A $\sqrt{n}$-nearest $O(\alpha \log d)$-hopset can be constructed in $O(1)$ rounds.
- α is the approximation ratio of the given APSP approximation.
- d is the weighted diameter.

Hopset construction

- **Goal:** Each vertex computes its $2^{\Theta(\sqrt{\log n})}$-nearest neighborhood.

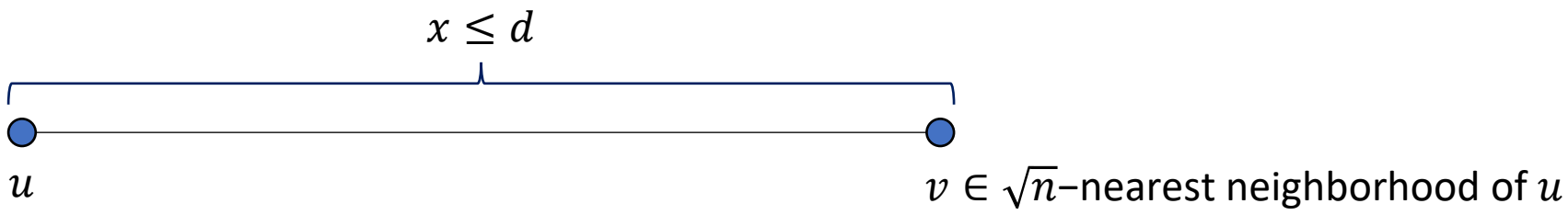
- **Tool:** k-nearest β-hopset.
  - Addition of new edges to the graph so that the k-nearest neighborhood of each vertex is reachable within (a shortest path of) β hops.

Claim: A $\sqrt{n}$-nearest $O(\alpha \log d) \subseteq O(\log^2 n)$-hopset can be constructed in $O(1)$ rounds.
- $\alpha \in O(\log n)$ is the approximation ratio of the given APSP approximation.
- $d \in n^{O(1)}$ is the weighted diameter.

Proof of the claim

Claim: A $\sqrt{n}$-nearest $O(\alpha \log d)$-hopset can be constructed in $O(1)$ rounds.

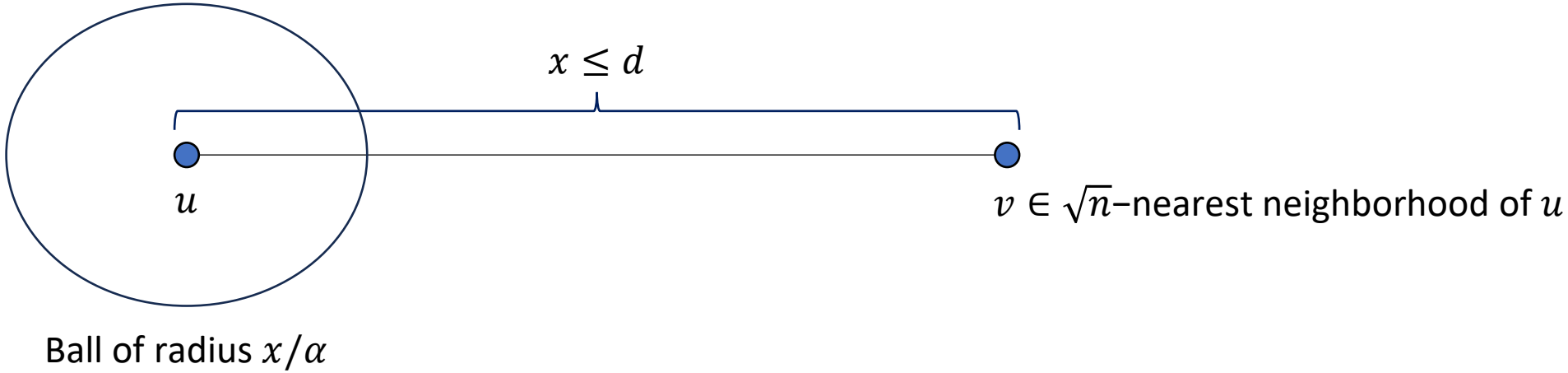
- α is the approximation ratio of the given APSP approximation.
- d is the weighted diameter.



Proof of the claim

Claim: A $\sqrt{n}$-nearest $O(\alpha \log d)$-hopset can be constructed in $O(1)$ rounds.

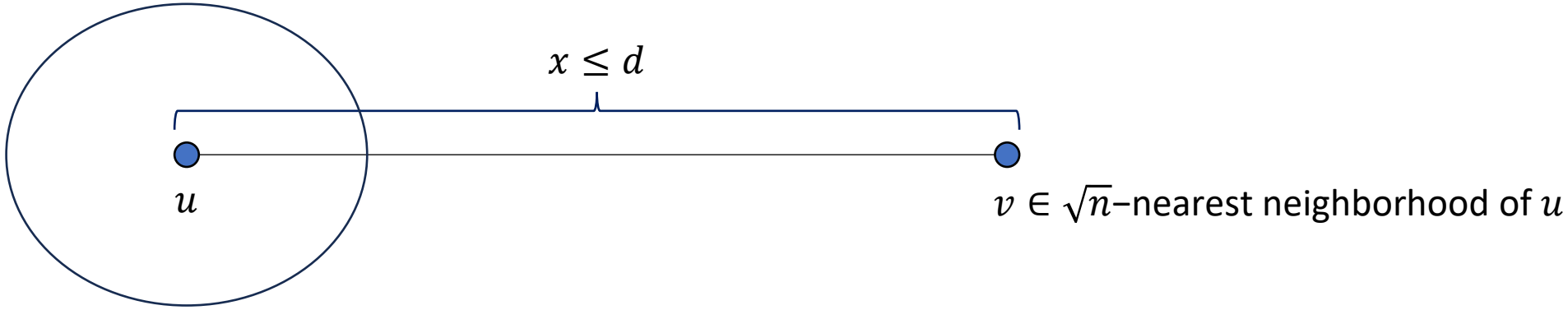
- α is the approximation ratio of the given APSP approximation.
- d is the weighted diameter.



Proof of the claim

Claim: A $\sqrt{n}$-nearest $O(\alpha \log d)$-hopset can be constructed in $O(1)$ rounds.

- α is the approximation ratio of the given APSP approximation.
- d is the weighted diameter.



Ball of radius x/α

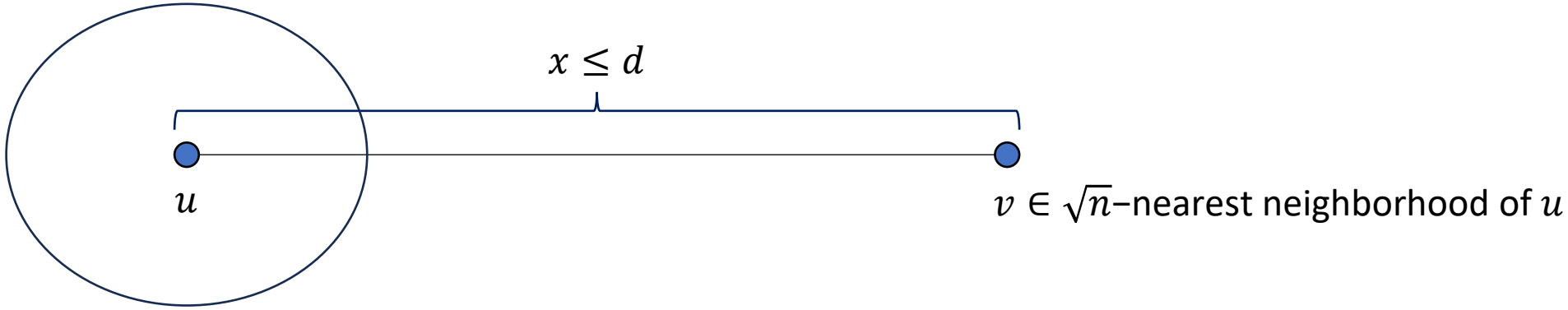
All vertices in this ball are within the $\sqrt{n}$ vertices closest to u according to the given α-approximate APSP.

The number of edges over these $\sqrt{n}$ vertices is $O(n)$, so they can be learned in $O(1)$ rounds, and this covers the entire ball.

Proof of the claim

Claim: A $\sqrt{n}$-nearest $O(\alpha \log d)$-hopset can be constructed in $O(1)$ rounds.

- α is the approximation ratio of the given APSP approximation.
- d is the weighted diameter.



Ball of radius x/α

△

All vertices in this ball are within the $\sqrt{n}$ vertices closest to u according to the given α-approximate APSP.

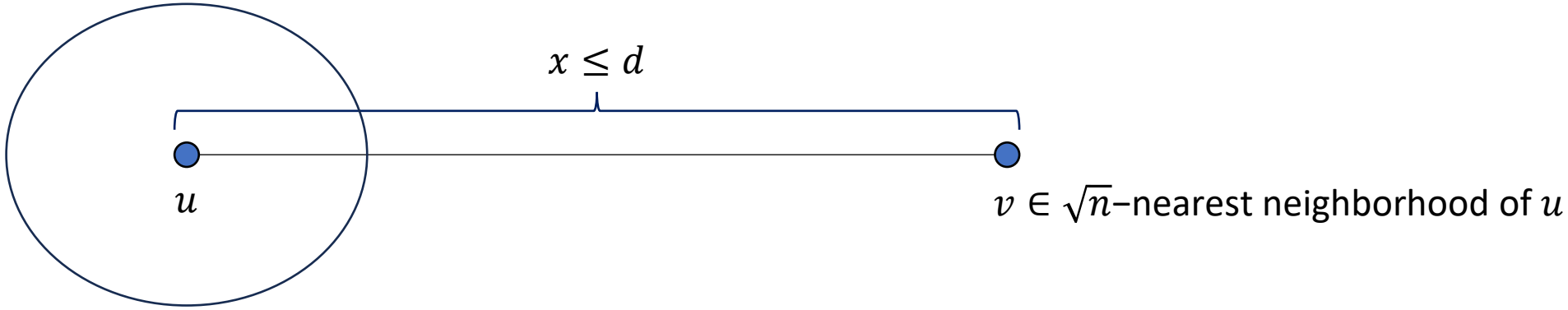
◁ u can learn this ball in $O(1)$ rounds.

In each step, the remaining distance to v is reduced by a factor of $1 - 1/\alpha$, so $O(\alpha \log d)$ steps suffice.

Proof of the claim

Claim: A $\sqrt{n}$-nearest $O(\alpha \log d)$-hopset can be constructed in $O(1)$ rounds.

- α is the approximation ratio of the given APSP approximation.
- d is the weighted diameter.



Ball of radius x/α

△

All vertices in this ball are within the $\sqrt{n}$ vertices closest to u according to the given α-approximate APSP.

◁

u can learn this ball in $O(1)$ rounds.

▽

v is reached within $O(\alpha \log d)$ steps.

Using the hopset

- Given a k-nearest β-hopset, how do we learn the k-nearest neighborhood?
  - $\beta \in O(\log^2 n)$
  - $k \in 2^{\Theta(\sqrt{\log n})}$

Using the hopset

- Given a k-nearest β-hopset, how do we learn the k-nearest neighborhood?
  - $\beta \in O(\log^2 n)$
  - $k \in 2^{\Theta(\sqrt{\log n})}$

Sparse matrix multiplication only: $O(\beta)$ rounds.

not good enough

Using the hopset

- Given a k-nearest β-hopset, how do we learn the k-nearest neighborhood?
  - $\beta \in O(\log^2 n)$
  - $k \in 2^{\Theta(\sqrt{\log n})}$

Sparse matrix multiplication only: $O(\beta)$ rounds. not good enough

Sparse matrix multiplication + repeated squaring: $O(\log \beta)$ rounds. not good enough

Using the hopset

- Given a k-nearest β-hopset, how do we learn the k-nearest neighborhood?
  - $\beta \in O(\log^2 n)$
  - $k \in 2^{\Theta(\sqrt{\log n})}$

Sparse matrix multiplication only: $O(\beta)$ rounds. not good enough

Sparse matrix multiplication + repeated squaring: $O(\log \beta)$ rounds. not good enough

Claim: This is doable in $O(1)$ rounds, if $\exists \epsilon \in \Theta(1)$ such that $k^{\beta^\epsilon} \leq \sqrt{n}$. ✔

Proof of the claim

- Given a k-nearest β-hopset, how do we learn the k-nearest neighborhood?

Claim: This is doable in $O(1)$ rounds, if $\exists \epsilon \in \Theta(1)$ such that $k^{\beta^{\epsilon}} \leq \sqrt{n}$.

△

- We focus on the easier case of $k^{\beta} \leq \sqrt{n}$.
  - $\epsilon = 1$
- Extension to the general case can be achieved by repeating for $O(1)$ rounds.

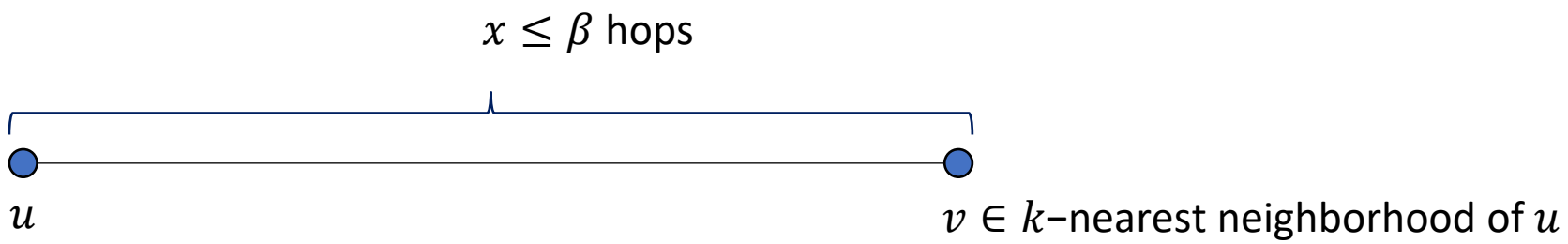
△

This is a **key observation**!

Proof of the claim

- Given a k-nearest β-hopset, how do we learn the k-nearest neighborhood?

Claim: This is doable in $O(1)$ rounds, under the condition $k^{\beta} \leq \sqrt{n}$.



- Since we focus on the computation of the k-nearest neighborhood, the number of relevant edges is at most kn.

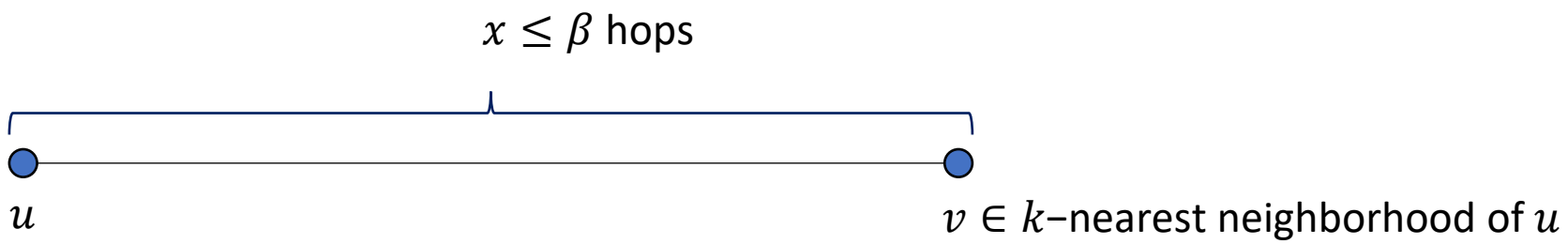
△

For each vertex, keep the k smallest-weight incident edges.

Proof of the claim

- Given a k-nearest β-hopset, how do we learn the k-nearest neighborhood?

Claim: This is doable in $O(1)$ rounds, under the condition $k^{\beta} \leq \sqrt{n}$.

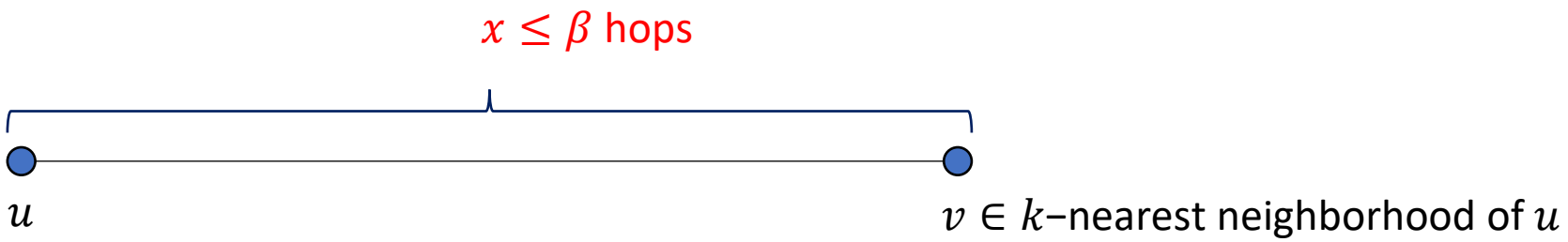


- Since we focus on the computation of the k-nearest neighborhood, the number of relevant edges is at most kn.
- Send each edge to each machine with probability $1/k$.
  - This ensures that each machine receives $O(n)$ edges with high probability.

Proof of the claim

- Given a k-nearest β-hopset, how do we learn the k-nearest neighborhood?

Claim: This is doable in $O(1)$ rounds, under the condition $k^{\beta} \leq \sqrt{n}$.



- Since we focus on the computation of the k-nearest neighborhood, the number of relevant edges is at most kn.
- Send each edge to each machine with probability $1/k$.
  - This ensures that each machine receives $O(n)$ edges with high probability.
- The probability that a machine learns the entire u-v path is at least $\frac{1}{k^x} \geq \frac{1}{k^{\beta}} \geq \frac{1}{\sqrt{n}}$.
  - With high probability, some machine learns this path.

Comments

- This is a known trick that is behind essentially all nontrivial constant-round congested clique algorithms.
  - It has been rediscovered several times.
- The procedure can be derandomized.

Approximation ratio improvement

- **Summary:** $\alpha \in O(\log n) \longrightarrow \alpha \in O\left(\sqrt{\log n}\right)$ in $O(1)$ rounds.
  - Are we done yet?

An issue

- **Summary:** $\alpha \in O(\log n) \longrightarrow \alpha \in O\left(\sqrt{\log n}\right)$ in $O(1)$ rounds.
  - Are we done yet?

- **Recall:** Our hopset allows each vertex to reach its $\sqrt{n}$-nearest neighbors in $O(\alpha \log d)$ hops.

The **approximation ratio α** improves as our algorithm progresses.

The **weighted diameter $d \in n^{O(1)}$** remains unchanged. ◁ This is an **issue**.

Weighted diameter reduction

- **Summary:** $\alpha \in O(\log n) \longrightarrow \alpha \in O\left(\sqrt{\log n}\right)$ in $O(1)$ rounds.
  - Are we done yet?

- **Recall:** Our hopset allows each vertex to reach its $\sqrt{n}$-nearest neighbors in $O(\alpha \log d)$ hops.

The **approximation ratio α** improves as our algorithm progresses.

The **weighted diameter $d \in n^{O(1)}$** remains unchanged. ◁ This is an **issue**.

△

Solution: A scaling technique allowing us to assume $d \in \log^{O(1)} n$. ◁ This is enough to get the round complexity $O(\log \log \log n)$.

Scaling technique

- **Goal:** Reduce the <u>weighted diameter</u> to $\log^{O(1)} n$.
- **Proof idea:**
  - Our hopset already reduces the <u>hop-diameter</u> to $O(\log^2 n)$.
    - We are not too far from our goal!



Recall: Our hopset allows each vertex to reach its $\sqrt{n}$-nearest neighbors in $O(\alpha \log d) \subseteq O(\log^2 n)$ hops.

Scaling technique

- **Goal:** Reduce the weighted diameter to $\log^{O(1)} n$.
- **Proof idea:**
  - Our hopset already reduces the hop-diameter to $O(\log^2 n)$.
    - We are not too far from our goal!

Note: We do not care about the distances to the vertices not within the $\sqrt{n}$-nearest neighbors – e.g., we may just add edges with infinite weight to connect all pairs of vertices.



Recall: Our hopset allows each vertex to reach its $\sqrt{n}$-nearest neighbors in $O(\alpha \log d) \subseteq O(\log^2 n)$ hops.

Scaling technique

- **Goal:** Reduce the weighted diameter to $\log^{O(1)} n$.
- **Proof idea:**
  - Our hopset already reduces the hop-diameter to $O(\log^2 n)$.
    - We are not too far from our goal!
  - Consider a shortest path.
    - Suppose the *average edge weight* in the path is x.
    - Rounding up *all* edge weights to the nearest integer multiple of $\tilde{x} \in \Theta(x)$.

Scaling technique

- **Goal:** Reduce the weighted diameter to $\log^{O(1)} n$.
- **Proof idea:**
  - Our hopset already reduces the hop-diameter to $O(\log^2 n)$.
    - We are not too far from our goal!
  - Consider a shortest path.
    - Suppose the *average edge weight* in the path is x.
    - Rounding up *all* edge weights to the nearest integer multiple of $\tilde{x} \in \Theta(x)$.

This distorts the distance by a factor of $O(1)$, which is fine.

Hop-distance and the weighted distance are now within a factor of $O(1)$. ◁ This meets our goal, at least for this path.

All we need to do is to create $O(\log n)$ graphs to cover all possible choices of $\tilde{x} = 1, 2, 4, 8, \ldots$

Scaling technique

- **Goal:** Reduce the weighted diameter to $\log^{O(1)} n$.
- **Proof idea:**
  - Our hopset already reduces the hop-diameter to $O(\log^2 n)$.
    - We are not too far from our goal!
  - Consider a shortest path.
    - Suppose the *average edge weight* in the path is x.
    - Rounding up *all* edge weights to the nearest integer multiple of $\tilde{x} \in \Theta(x)$.

This distorts the distance by a factor of $O(1)$, which is fine.

Hop-distance and the weighted distance are now within a factor of $O(1)$.

◁ This meets our goal, at least for this path.

All we need to do is to create $O(\log n)$ graphs to cover all possible choices of $\tilde{x} = 1, 2, 4, 8, \ldots$

Scaling technique

Issue: Need to process all $O(\log n)$ graphs **in parallel**.

- **Goal:** Reduce the weighted diameter to $\log^{O(1)} n$.
- **Proof idea:**
  - Our hopset already reduces the hop-diameter to $O(\log^2 n)$.
    - We are not too far from our goal!
  - Consider a shortest path.
    - Suppose the *average edge weight* in the path is x.
    - Rounding up *all* edge weights to the nearest integer multiple of $\tilde{x} \in \Theta(x)$.

This distorts the distance by a factor of $O(1)$, which is fine.

Hop-distance and the weighted distance are now within a factor of $O(1)$.

This meets our goal, at least for this path.

Size reduction

- **Goal:** Reduce the number of vertices by a factor of $\Omega(\log n)$.

$2^{\Theta(\sqrt{\log n})}$-nearest neighborhood can be computed in $O(1)$ rounds.

▽

A skeleton graph with $\frac{n}{2^{\Theta(\sqrt{\log n})}}$ vertices can be computed in $O(1)$ rounds.

▽

$O(1)$-approximate APSP is reduced to $O(1)$-approximate APSP in the skeleton graph.

Conclusions

- **Main result:** $O(\log\log\log n)$-round $O(1)$-approximation APSP.
  - Exponential improvement in the unweighted case.
  - Doubly exponential improvement in the weighted case.
  - Smooth tradeoff between approximation ratio and round complexity.
    - This is achieved by terminating the algorithm early.

Conclusions

- **Main result:** $O(\log\log\log n)$-round $O(1)$-approximation APSP.
  - Exponential improvement in the unweighted case.
  - Doubly exponential improvement in the weighted case.
  - Smooth tradeoff between approximation ratio and round complexity.
    - This is achieved by terminating the algorithm early.

- **Techniques:**
  - New framework to successively improve the approximation ratio.
  - Hopsets and skeleton graphs based on the notion of k-nearest neighborhood.

Subsequent work

- The algorithm can be made **deterministic**.

Kyungjin Cho, Michal Dory, Yannic Maus, and Tijn de Vos.
Deterministic Distance Approximation in MPC via Improved Hitting Sets.
SPAA 2026

Replacing random sampling and randomized spanner construction with deterministic hitting sets and spanners.

Subsequent work

- Our approximation ratio is $7^4 + \epsilon = 2401 + \epsilon$.
- This can be improved to $2 + \epsilon$ for unweighted graphs.

Shiri Chechik.
From APSP to MSSP and Back Again: Improved Shortest Paths Algorithms in the Congested Clique Model.
STOC 2027 submission #78.

A constant-round algorithm to go from $O(1)$-approximation to $(2 + \epsilon)$-approximation.

Subsequent work

- The proof ideas extend to the massively parallel computation (MPC) model.

$O(1)$-approximation



The distance oracle problem:
Preprocess the graph so that any distance query can be done in $O(1)$ rounds.

Subsequent work

- The proof ideas extend to the massively parallel computation (MPC) model.

Yi-Jun Chang and Shaun Quek.
Improved Massively Parallel Shortest Path Computation.
FSTTCS 2026

$O(1)$-approximation



The distance oracle problem:
Preprocess the graph so that any distance query can be done in $O(1)$ rounds.

Heterogeneous regime: $O(\log^2 \log n)$
Near-linear regime: $O(\log \log n)$

Weighted graphs

Subsequent work

$O(1)$-approximation

- The proof ideas extend to the massively parallel computation (MPC) model.

The distance oracle problem:
Preprocess the graph so that any distance query can be done in $O(1)$ rounds.

Yi-Jun Chang and Shaun Quek.
Improved Massively Parallel Shortest Path Computation.
FSTTCS 2026

Heterogeneous regime: $O(\log^2 \log n)$
Near-linear regime: $O(\log \log n)$

Weighted graphs

Heterogeneous regime: $O(\log^{2+\epsilon} \log n)$
Near-linear regime: $O(\log^{2+\epsilon} \log n)$

Unweighted graphs

Michal Dory and Shaked Matar.
Massively Parallel Algorithms for Approximate Shortest Paths.
SPAA 2024

Previous work

- The *essential idea* of our new skeleton graph appeared before our work:

Alexandr Andoni, Clifford Stein, and Peilin Zhong.
Parallel Approximate Undirected Shortest Paths Via Low Hop Emulators.
STOC 2020

$(1+\epsilon)$-approximate s-t shortest path in
$\tilde{O}(m)$ work and polylog n depth.

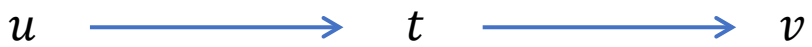
Previous work

- The *essential idea* of our new skeleton graph appeared before our work:

Alexandr Andoni, Clifford Stein, and Peilin Zhong.
Parallel Approximate Undirected Shortest Paths Via Low Hop Emulators.
STOC 2020

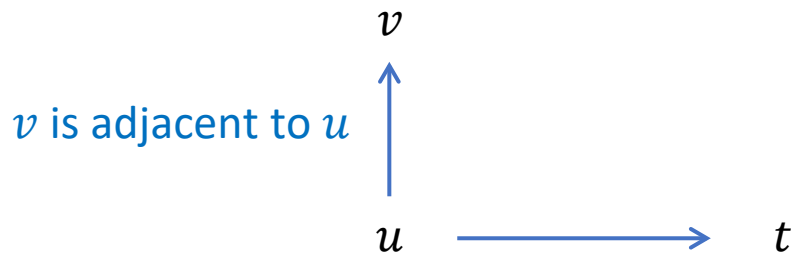
$(1+\epsilon)$-approximate s-t shortest path in
$\tilde{O}(m)$ work and polylog n depth.

Our construction: two steps of exploration.



$t \in k$-nearest of u v is adjacent to t

Their construction: two steps are done independently.



$t \in k$-nearest of u

Future directions

- **Main open question:**
  - Can you improve the $O(\log\log\log n)$ upper bound?

- **Interesting research direction to try:**
  - Can you find further applications of our techniques?

Future directions

- **Main open question:**
  - Can you improve the $O(\log\log\log n)$ upper bound?

- **Interesting research direction to try:**
  - Can you find further applications of our techniques?

Thank you!