Exponentially Faster Shortest Paths in the Congested Clique

Michal Dory (Technion), Merav Parter (Weizmann Institute)

This project has received funding from the European Union's Horizon 2020
Research and Innovation Programme under grant agreement no. 755839

This Talk

$poly(\log \log n)$ round algorithms for approximate shortest paths in the Congested Clique

This Talk

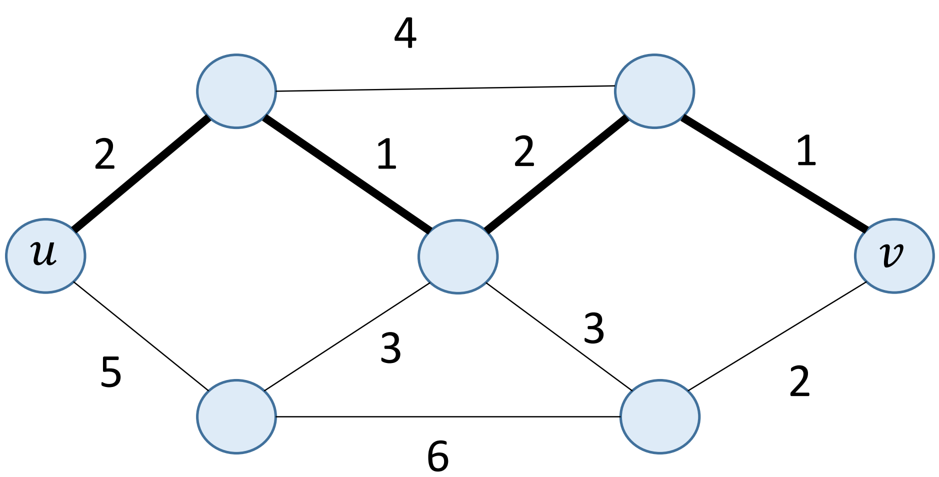
$poly(\log\log n)$ round algorithms for approximate shortest paths in the Congested Clique

- Background & our results
- Techniques:
  - Near-additive emulators
  - Distance sensitive toolkit

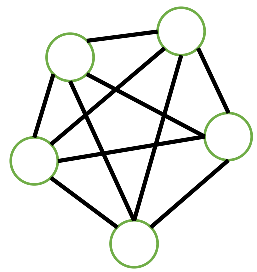
Background & Our Results

Distance Computation

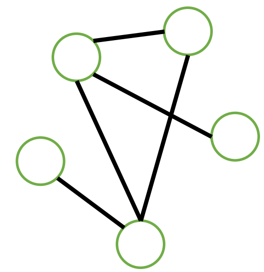
- All-pairs shortest paths (APSP)
- Single-source shortest paths (SSSP)
- Multi-source shortest paths (MSSP)



The Congested Clique Model



Communication Network



Input Graph

- n vertices
- Synchronous rounds, $\Theta(\log n)$-bit messages
- All-to-All communication
- Input and output are local

Previous Work

- Polynomial time algorithms for exact APSP based on matrix multiplication [Censor-Hillel et al. 15, Le Gall 16]

Previous Work

- Polynomial time algorithms for exact APSP based on matrix multiplication [Censor-Hillel et al. 15, Le Gall 16]

Round Complexity	Variant
$\tilde{O}(n^{1/3})$	weighted directed
$O(n^{0.158})$	unweighted undirected

Previous Work

- Polynomial time algorithms for exact APSP based on matrix multiplication [Censor-Hillel et al. 15, Le Gall 16]

- Poly-logarithmic algorithms for approximate shortest paths [Becker et al. 17, Censor-Hillel et al. 19]

Previous Work: Approximations

Round Complexity	Problem	Reference
$O(\epsilon^{-3}\text{polylog}\, n)$	$(1+\epsilon)$-**SSSP**	[Becker, Karrenbauer, Krinninger, Lenzen '17]
$O(\log^2 n/\epsilon)$	$(1+\epsilon)$-**MSSP** with $O(\sqrt{n})$ sources	[Censor-Hillel, D, Korhonen, Leitersdorf, '19]
$O(\log^2 n/\epsilon)$	$(3+\epsilon)$-**APSP**	
$O(\log^2 n/\epsilon)$	$(2+\epsilon)$-**unweighted APSP**	

- All results are for **weighted undirected** graphs, unless specified otherwise

Previous Work: Approximations

Round Complexity	Problem	Reference
$O(\epsilon^{-3}\text{polylog}\, n)$	$(1+\epsilon)$-**SSSP**	[Becker, Karrenbauer, Krinninger, Lenzen '17]
$O(\log^2 n/\epsilon)$	$(1+\epsilon)$-**MSSP** with $O(\sqrt{n})$ sources	[Censor-Hillel, D, Korhonen, Leitersdorf, '19]
$O(\log^2 n/\epsilon)$	$(3+\epsilon)$-**APSP**	
$O(\log^2 n/\epsilon)$	$(2+\epsilon)$-**unweighted APSP**	

Can we get faster algorithms?

Distances in the Clique

Can we get faster algorithms?

Distances in the Clique

Can we get faster algorithms?

- In the matrix multiplication algorithms, in iteration i we deal with paths of 2^i edges

Distances in the Clique

Can we get faster algorithms?

- In the matrix multiplication algorithms, in iteration i we deal with paths of 2^i edges
- We need $\log n$ iterations

Our Results

Unweighted undirected graphs:

$poly(\log\log n)$	• $(1+\epsilon)$-**MSSP** with $O(n^{1/2})$ sources • $(2+\epsilon)$-**APSP**

Our Results

Unweighted undirected graphs:

$poly(\log\log n)$	• $(1+\epsilon)$-**MSSP** with $O(n^{1/2})$ sources • $(2+\epsilon)$-**APSP**

Previous work:

$poly(\log n)$	• $(1+\epsilon)$-**MSSP** with $O(n^{1/2})$ sources • $(2+\epsilon)$-**APSP**

Our Results

Unweighted undirected graphs:

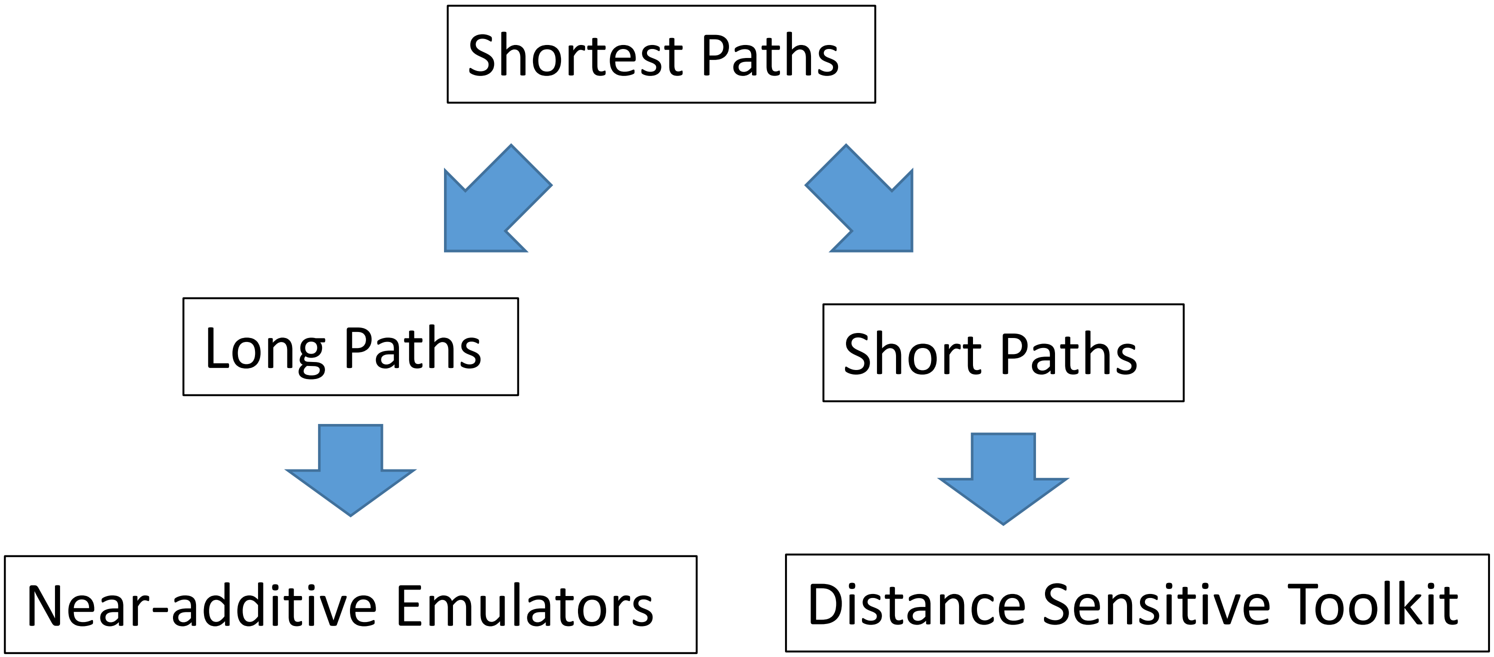
$poly(\log\log n)$	• $(1+\epsilon)$-**MSSP** with $O(n^{1/2})$ sources • $(2+\epsilon)$-**APSP**

The APSP approximation is near-tight:

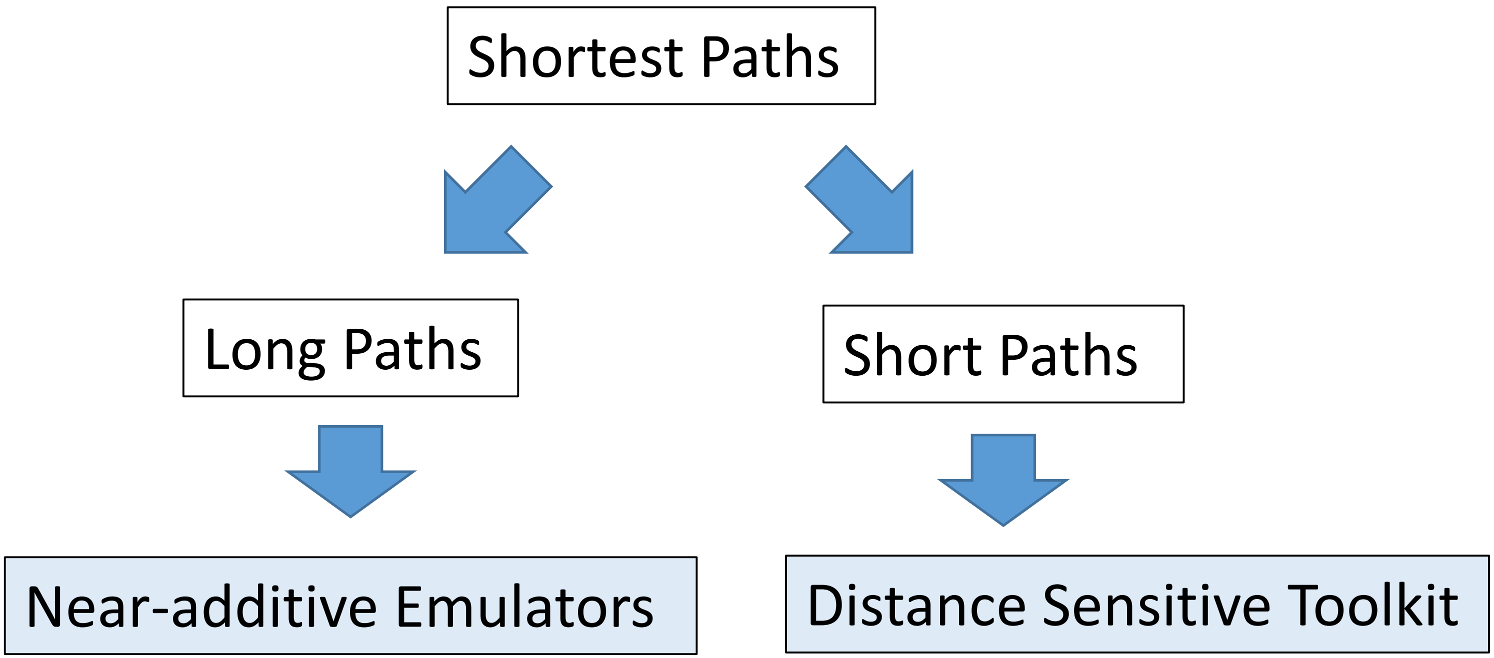
$(2-\epsilon)$-APSP implies Matrix Multiplication	[Dor, Halperin, Zwick '00 Korhonen, Suomela '18]

Techniques

Our Techniques



Our Techniques



How to Get Faster Algorithms?

- $poly(\log n)$ looks like a natural barrier

How to Get Faster Algorithms?

- $poly(\log n)$ looks like a natural barrier
- But maybe we can improve the approximation?

APSP Approximation

- $(2+\epsilon)$-approximation for APSP:

$$\delta(u,v) \leq (2+\epsilon)d(u,v)$$

APSP Approximation

- $(2 + \epsilon)$-approximation for APSP:

$$\delta(u, v) \le (2 + \epsilon) d(u, v)$$

- $(1 + \epsilon, \beta)$-approximation:

$$\delta(u, v) \le (1 + \epsilon) d(u, v) + \beta$$

APSP Approximation

- $(2+\epsilon)$-approximation for APSP:

$$\delta(u,v) \leq (2+\epsilon)d(u,v)$$

- $(1+\epsilon,\beta)$-approximation:

$$\delta(u,v) \leq (1+\epsilon)d(u,v)+\beta$$

Much better for long distances!

Near-Additive Emulator

A sparse graph H such that for all u, v:

$$d(u, v) \leq d_H(u, v) \leq (1 + \epsilon)d(u, v) + \beta$$

Near-Additive Emulator

A sparse graph H such that for all u, v:

$$d(u,v) \leq d_H(u,v) \leq (1+\epsilon)d(u,v) + \beta$$

Build near-additive emulator with $O(n \log\log n)$ edges

Collect it by all vertices in $O(\log\log n)$ rounds

Near-Additive Emulator

A sparse graph H such that for all u, v:

$$d(u,v) \leq d_H(u,v) \leq (1+\epsilon)d(u,v) + \beta$$

Build near-additive emulator with $O(n \log\log n)$ edges

→

Collect it by all vertices in $O(\log\log n)$ rounds

$(1+\epsilon, \beta)$-APSP, for $\beta = O\left(\frac{\log\log n}{\epsilon}\right)^{\log\log n}$

Shortest Paths via Emulators

Near-additive emulators

$\downarrow$

$(1+\epsilon,\beta)$-APSP, $\beta = O\left(\frac{\log\log n}{\epsilon}\right)^{\log\log n}$

$\downarrow$

If $d(u,v) = \Omega(\beta/\epsilon)$: $(1+\Theta(\epsilon))$-approximation

Shortest Paths via Emulators

Near-additive emulators

$(1+\epsilon,\beta)$-APSP, $\beta = O\left(\frac{\log\log n}{\epsilon}\right)^{\log\log n}$

If $d(u,v) = \Omega(\beta/\epsilon)$: $(1+\Theta(\epsilon))$-approximation

Left with short paths of length $t = O(\beta/\epsilon)$

Shortest Paths via Emulators

Left with short paths of length $t = O(\beta/\epsilon)$

Requires $poly(\log t) = poly(\log\log n)$ time!

Shortest Paths via Emulators

Shortest Paths

Long Paths: $\Omega(\beta/\epsilon)$

Near-additive Emulators: $(1+\Theta(\epsilon))$-approximation

Short Paths: $O(\beta/\epsilon)$

Distance Sensitive Toolkit: $poly\left(\log\frac{\beta}{\epsilon}\right)$ time

Shortest Paths via Emulators

Shortest Paths

Long Paths: $\Omega(\beta/\epsilon)$

Short Paths: $O(\beta/\epsilon)$

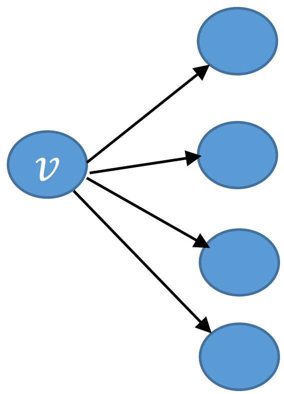
Near-additive Emulators: $(1+\Theta(\epsilon))$-approximation

Distance Sensitive Toolkit: $poly\left(\log\frac{\beta}{\epsilon}\right)$ time

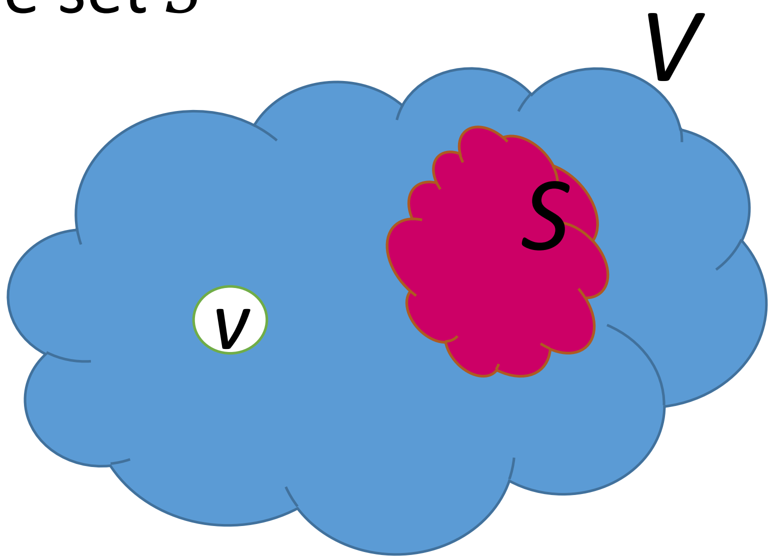
Distance Tools

[Censor-Hillel, D, Korhonen, Leitersdorf, '19]

- k-nearest: for each vertex, compute distances to k nearest vertices



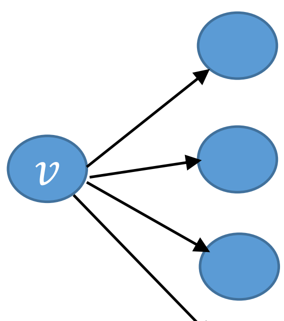
- MSSP: for each vertex, compute $(1+\epsilon)$-approximate distances to the set S



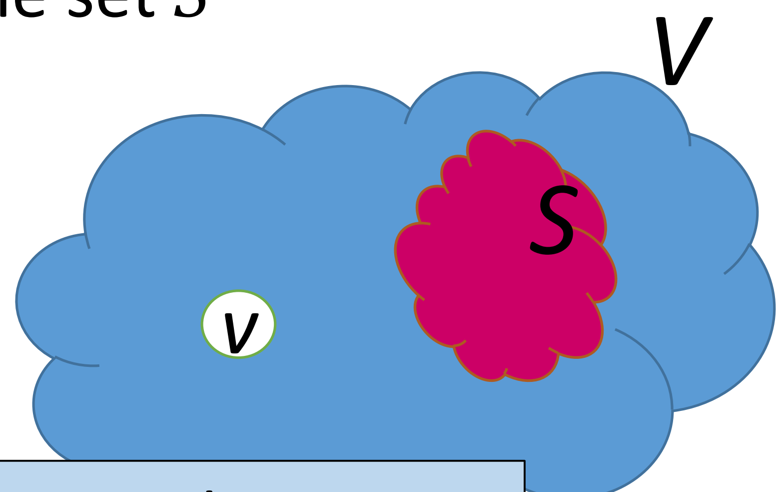
Distance Tools

[Censor-Hillel, D, Korhonen, Leitersdorf, '19]

- k-nearest: for each vertex, compute distances to k nearest vertices



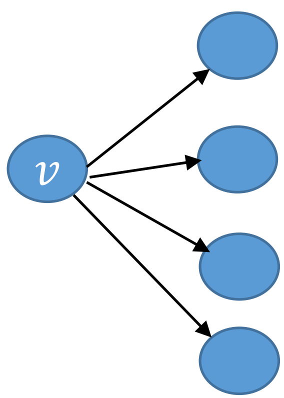
- MSSP: for each vertex, compute $(1+\epsilon)$-approximate distances to the set S



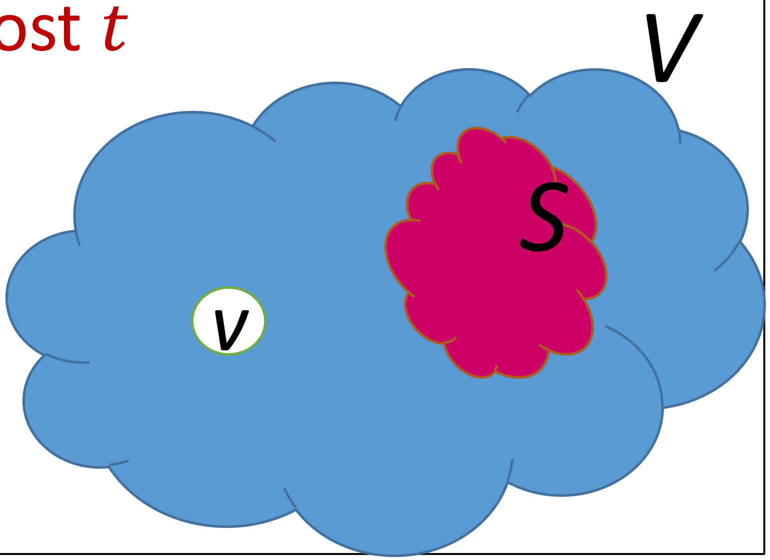
Can be implemented in $poly(\log n)$ time

Distance Sensitive Tools

- (k, t)-nearest: for each vertex, compute distances to k nearest vertices of distance at most t

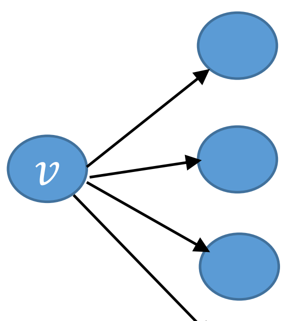


- Bounded MSSP: for each vertex, compute $(1+\epsilon)$-approximate distances to vertices in S of distance at most t

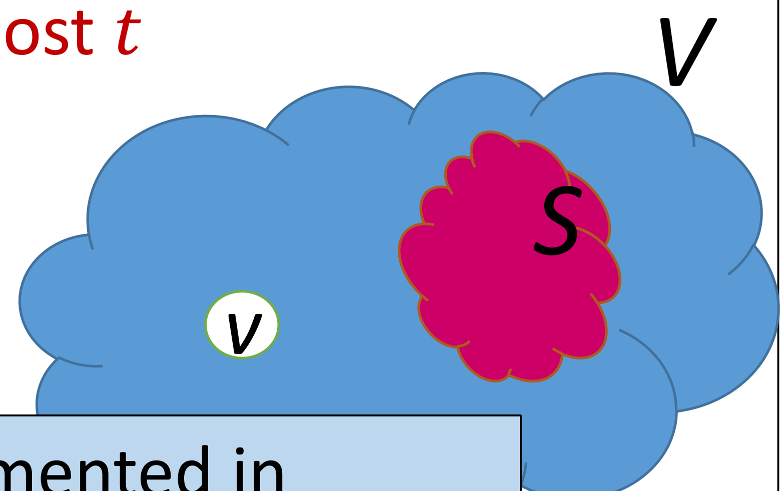


Distance Sensitive Tools

- (k, t)-nearest: for each vertex, compute distances to k nearest vertices of distance at most t



- Bounded MSSP: for each vertex, compute $(1 + \epsilon)$-approximate distances to vertices in S of distance at most t



Can be implemented in $poly(\log t) = poly(\log \log n)$ time

Shortest Paths via Emulators

Shortest Paths

Long Paths: $\Omega(\beta/\epsilon)$

Short Paths: $O(\beta/\epsilon)$

Near-additive Emulators: $(1+\Theta(\epsilon))$-approximation

Distance Sensitive Toolkit: $poly\left(\log\frac{\beta}{\epsilon}\right)$ time

Shortest Paths via Emulators

Shortest Paths

Long Paths: $\Omega(\beta/\epsilon)$

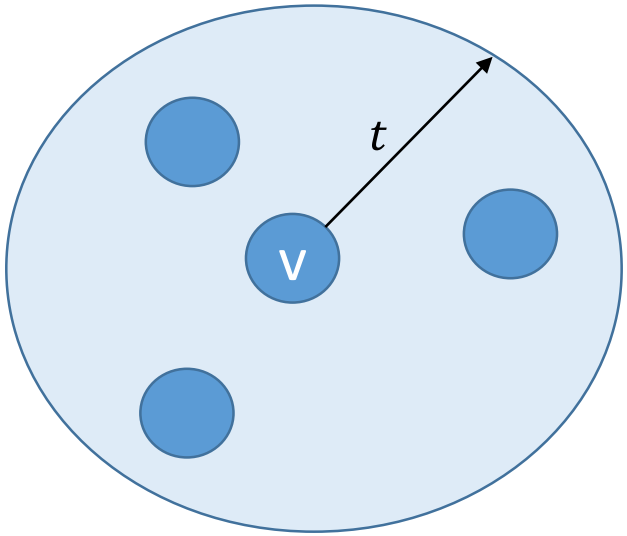
Short Paths: $O(\beta/\epsilon)$

Near-additive Emulators: $(1+\Theta(\epsilon))$-approximation

Distance Sensitive Toolkit: $poly\left(\log\frac{\beta}{\epsilon}\right)$ time

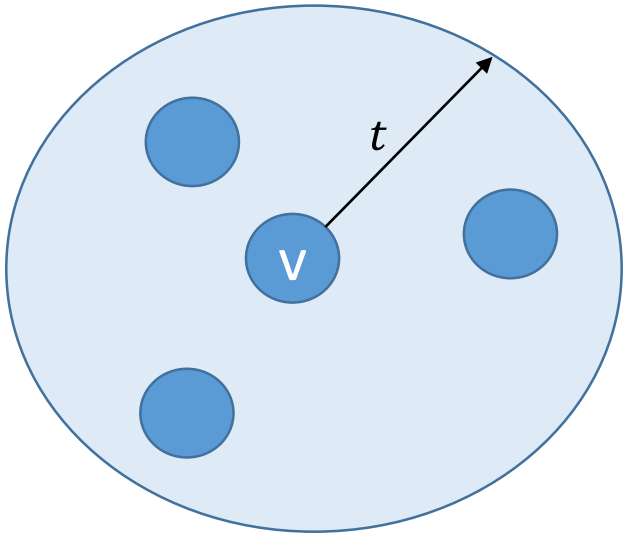
Near-Additive Emulators

- Each vertex inspects its $t = O(\beta/\epsilon)$-radius ball, and adds to the emulator edges to some of these vertices



Near-Additive Emulators

- Each vertex inspects its $t = O(\beta/\epsilon)$-radius ball, and adds to the emulator edges to some of these vertices
- Can be implemented in $poly(\log t) = poly(\log \log n)$ time using the distance sensitive toolkit



Near-Additive Emulators

Inspired by [Elkin-Neiman, 2018] and [Thorup-Zwick, 2006]

We construct:

$$\left(1+\epsilon, O\left(\frac{r}{\epsilon}\right)^{r-1}\right)\text{-emulator with } O\left(rn^{1+1/2^r}\right) \text{ edges}$$

- Choosing $r = \log\log n$ gives:

$$(1+\epsilon, \beta)\text{-emulator with } O(n \log\log n) \text{ edges,}$$

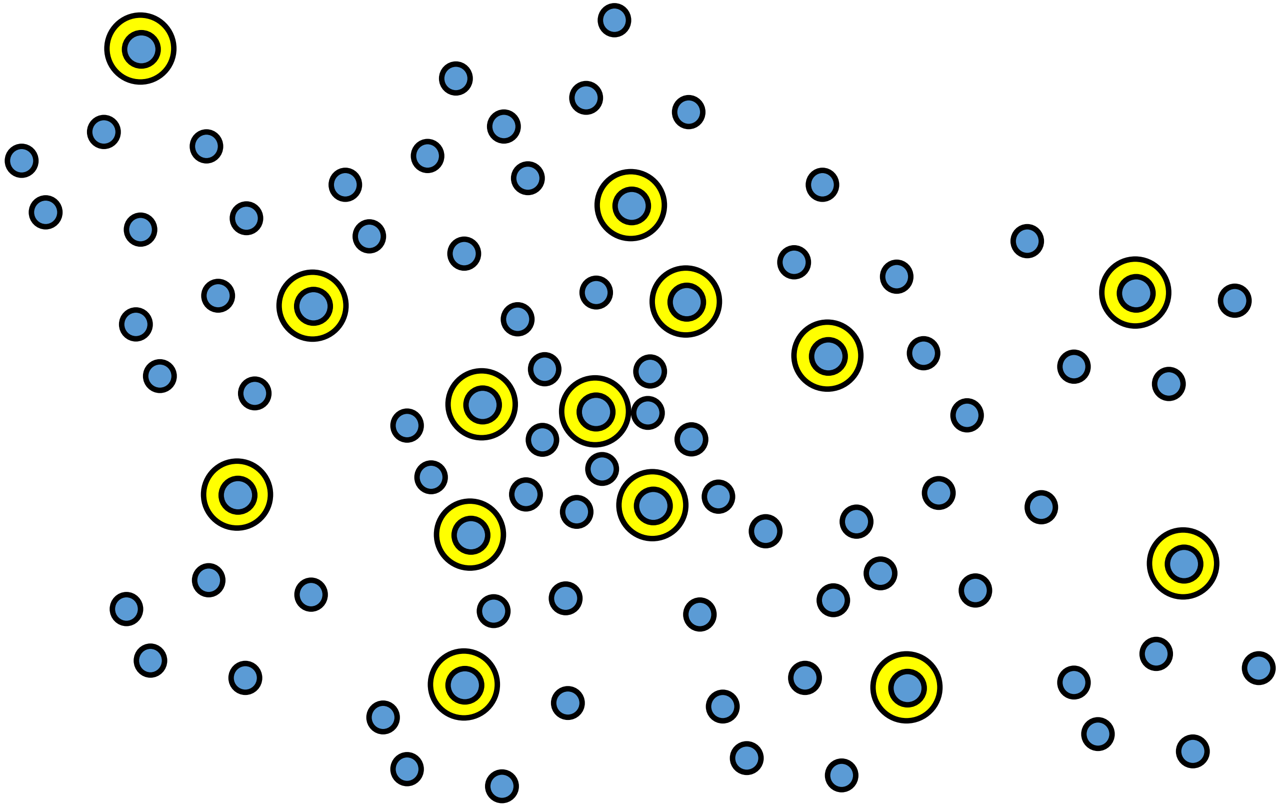
where $\beta = O\left(\frac{\log\log n}{\epsilon}\right)^{\log\log n}$

Near-Additive Emulators

- Define sampled subsets $V = S_0 \supseteq S_1 \supseteq \cdots \supseteq S_r \supseteq S_{r+1} = \emptyset$
- $S_i \leftarrow Sample(S_{i-1}, p_i)$

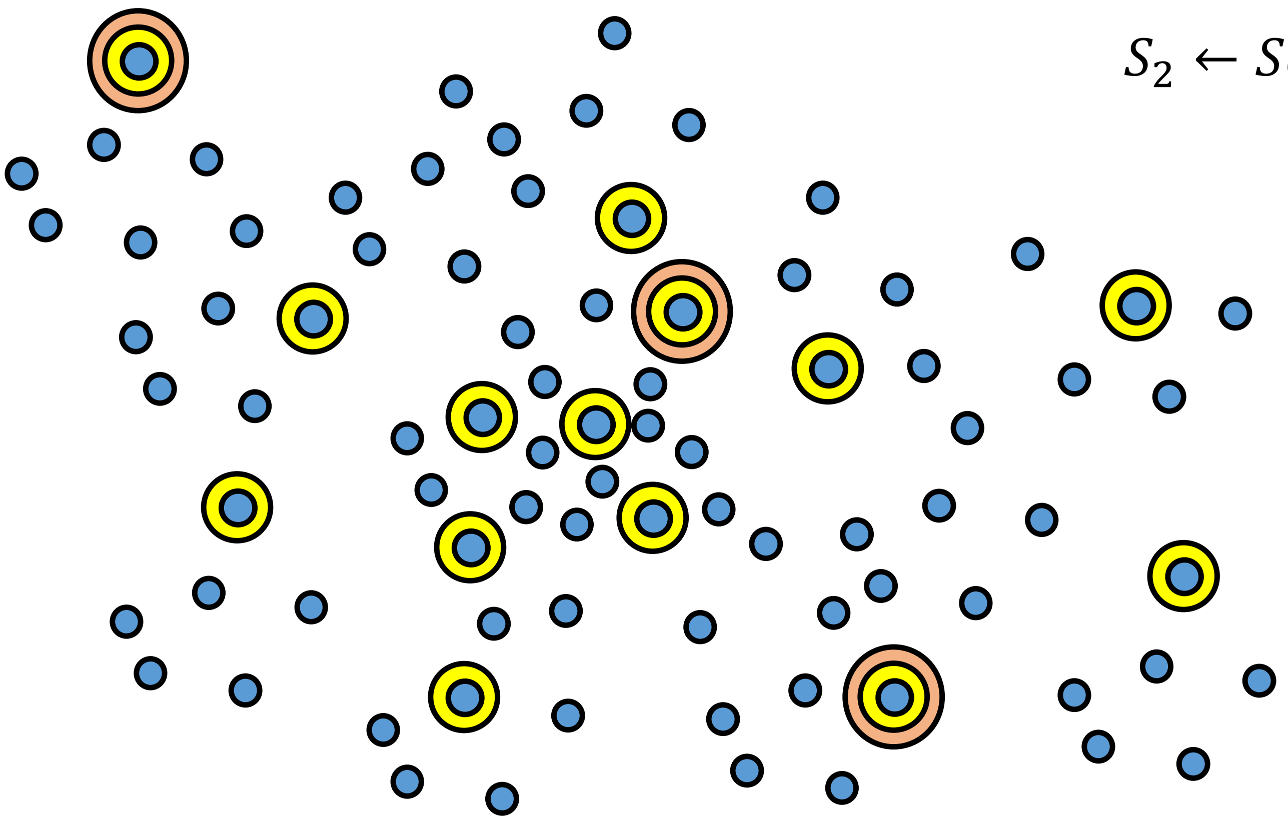
The choice of p_i determines the size of the emulator.

$$S_1 \leftarrow Sample(V, p_1)$$



$$S_1 \leftarrow Sample(V, p_1)$$

$$S_2 \leftarrow Sample(S_1, p_2)$$

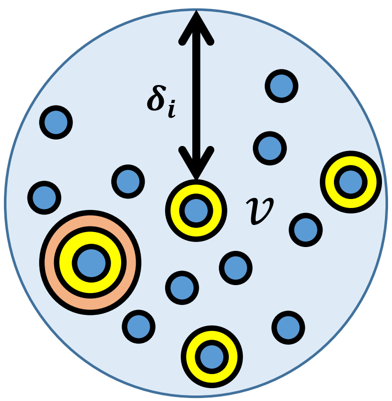


Near-Additive Emulators

- Define sampled subsets $V = S_0 \supseteq S_1 \supseteq \cdots \supseteq S_r \supseteq S_{r+1} = \emptyset$
- $S_i \leftarrow Sample(S_{i-1}, p_i)$

A vertex in $\boldsymbol{v} \in \boldsymbol{S_i}$ looks at the ball of radius $\boldsymbol{\delta_i} = \Theta(1/\epsilon^i)$

Is there a vertex in $\boldsymbol{B(v, \delta_i)} \cap \boldsymbol{S_{i+1}}$?



Near-Additive Emulators

A vertex in $\boldsymbol{v} \in \boldsymbol{S_i}$ looks at the ball of radius $\boldsymbol{\delta_i} = \Theta(1/\epsilon^i)$

Is there a vertex in $\boldsymbol{B(v, \delta_i) \cap S_{i+1}}$?

Yes: add an edge to such vertex

No: add edges to all vertices in $B(v, \delta_i) \cap S_i$

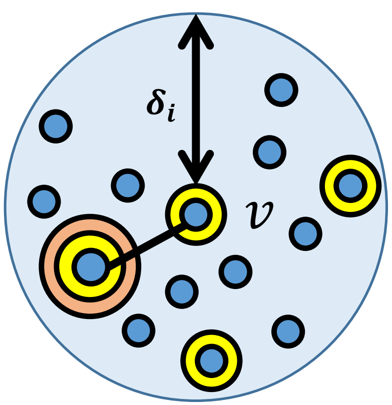
Near-Additive Emulators

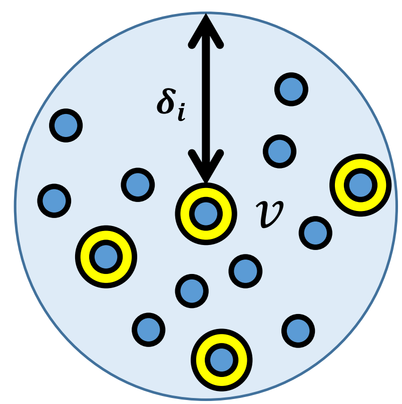
A vertex in $\boldsymbol{v} \in \boldsymbol{S_i}$ looks at the ball of radius $\boldsymbol{\delta_i} = \Theta(1/\epsilon^i)$

Is there a vertex in $\boldsymbol{B(v, \delta_i) \cap S_{i+1}}$?

Yes: add an edge to such vertex

No: add edges to all vertices in $B(v, \delta_i) \cap S_i$



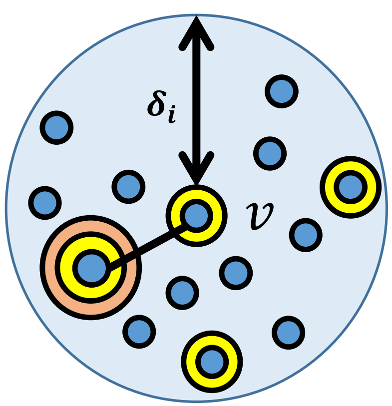


Near-Additive Emulators

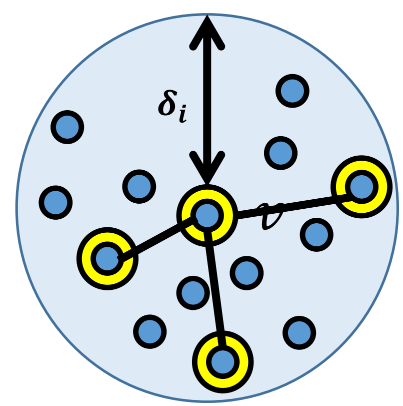
A vertex in $\boldsymbol{v} \in \boldsymbol{S_i}$ looks at the ball of radius $\boldsymbol{\delta_i} = \Theta(1/\epsilon^i)$

Is there a vertex in $\boldsymbol{B(v, \delta_i) \cap S_{i+1}}$?

Yes: add an edge to such vertex

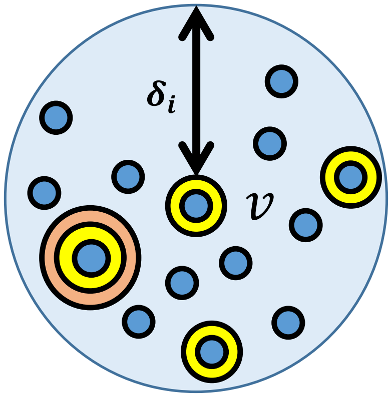


No: add edges to all vertices in $B(v, \delta_i) \cap S_i$



Near-Additive Emulators

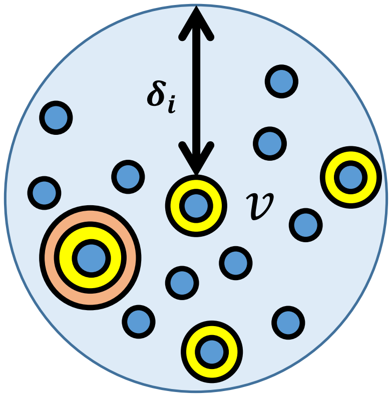
- Vertices inspect balls of radius $\boldsymbol{\delta_i}$
- Using the distance sensitive toolkit can be done in $poly(\log \delta_i)$ rounds



Stretch Analysis

- i-clustered vertex: there is a vertex in S_i close-by

Lemma: if all vertices in the shortest $u - v$ path are at most i-clustered, $d_H(u,v) \leq (1 + \Theta(\epsilon i))d(u,v) + \Theta\left(\frac{1}{\epsilon^{i-1}}\right)$



Stretch Analysis

- i-clustered vertex: there is a vertex in S_i close-by

Lemma: if all vertices in the shortest $u - v$ path are at most i-clustered, $d_H(u,v) \leq (1 + \Theta(\epsilon i))d(u,v) + \Theta\left(\frac{1}{\epsilon^{i-1}}\right)$

- Leads to $\left(1 + \Theta(\epsilon r), \Theta\left(\frac{1}{\epsilon^{r-1}}\right)\right)$ stretch

Stretch Analysis

- i-clustered vertex: there is a vertex in S_i close-by

Lemma: if all vertices in the shortest $u - v$ path are at most i-clustered, $d_H(u, v) \le (1 + \Theta(\epsilon i))d(u, v) + \Theta\left(\frac{1}{\epsilon^{i-1}}\right)$

- Leads to $\left(1 + \Theta(\epsilon r), \Theta\left(\frac{1}{\epsilon^{r-1}}\right)\right)$ stretch
- After rescaling: $\left(1 + \epsilon, O\left(\frac{r}{\epsilon}\right)^{r-1}\right)$

Stretch Analysis

- i-clustered vertex: there is a vertex in S_i close-by

Lemma: if all vertices in the shortest $u - v$ path are at most i-clustered, $d_H(u,v) \leq (1 + \Theta(\epsilon i))d(u,v) + \Theta\left(\frac{1}{\epsilon^{i-1}}\right)$

- Leads to $\left(1 + \Theta(\epsilon r), \Theta\left(\frac{1}{\epsilon^{r-1}}\right)\right)$ stretch
- After rescaling: $\left(1 + \epsilon, O\left(\frac{\log\log n}{\epsilon}\right)^{\log\log n}\right)$

Shortest Paths via Emulators

Shortest Paths

Long Paths: $\Omega(\beta/\epsilon)$

Short Paths: $O(\beta/\epsilon)$

Near-additive Emulators:
$(1+\Theta(\epsilon))$-approximation

Distance Sensitive Toolkit:
$poly\left(\log\frac{\beta}{\epsilon}\right)$ time

Conclusion

Near-additive Emulators → Long Paths

Distance Sensitive Toolkit → Short Paths

$poly(\log\log n)$	$(1+\epsilon)$-**MSSP** with $O(n^{1/2})$ sources In **unweighted** graphs
$poly(\log\log n)$	$(2+\epsilon)$-**APSP** in **unweighted** graphs

Summary

$poly(\log\log n)$ round algorithms for approximate shortest paths in the Congested Clique

Unweighted graphs:

$poly(\log\log n)$	• $(1+\epsilon)$-**MSSP** with $O(n^{1/2})$ sources • $(2+\epsilon)$-**APSP** • $(1+\epsilon,\beta)$-**APSP,** $\beta = O\left(\frac{\log\log n}{\epsilon}\right)^{\log\log n}$

Open Questions

- Faster algorithms
- Weighted APSP
- Directed/exact shortest paths