Fast Approximate Shortest Paths in the Congested Clique

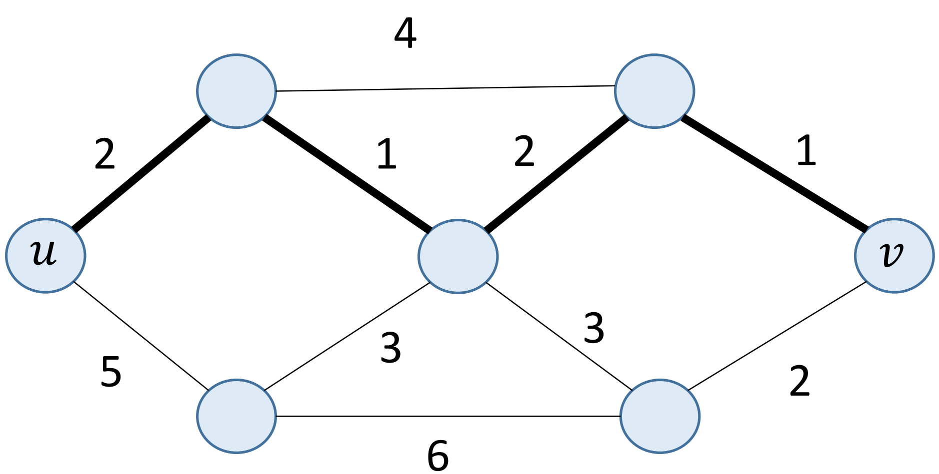
Michal Dory, Technion

Joint work with: Keren Censor-Hillel (Technion), Janne Korhonen (IST Austria), Dean Leitersdorf (Technion)

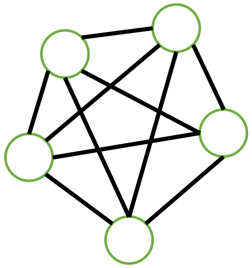
This project has received funding from the European Union's Horizon 2020
Research and Innovation Programme under grant agreement no. 755839

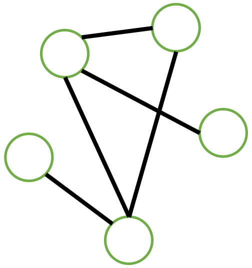
Distance Computation

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



The Congested Clique model





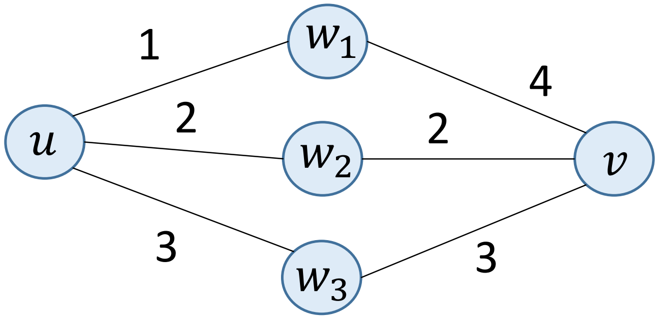
Communication Network **Input Graph**

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

Computing Distances using Matrix Multiplication

- A – weighted adjacency matrix
- Distance product:

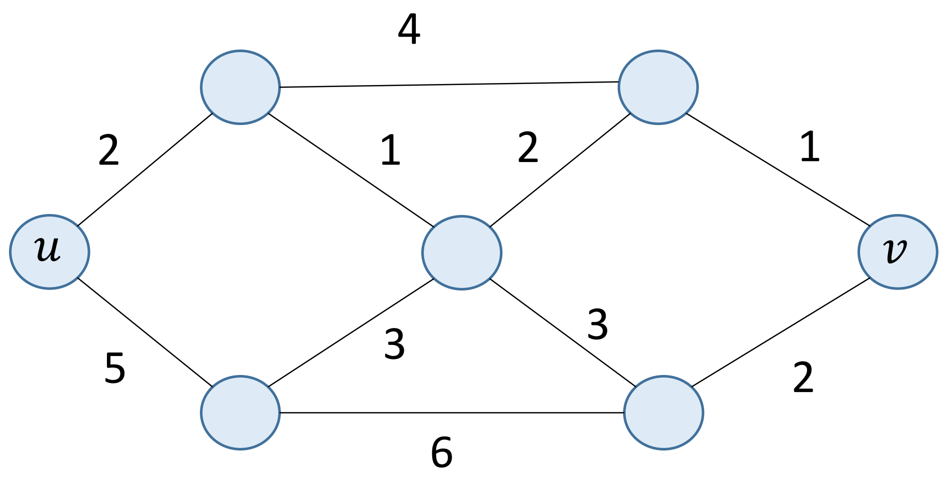
$$A^2[u,v] = \min_{w} A(u,w) + A(w,v)$$



- This is the minimum weight path between u and v of at most 2 edges

Computing Distances using Matrix Multiplication

- Similarly, $A^i[u, v]$ = minimum weight path between u and v of at most i edges (hops).
- <u>Our goal:</u> compute A^n



Computing Distances using Matrix Multiplication

- Our goal: compute A^n
- Requires $O(\log n)$ matrix multiplications:

$$A \rightarrow A^2 \rightarrow A^4 \rightarrow \ldots \rightarrow A^n$$

- How fast can we multiply matrices?

Computing Distances using Matrix Multiplication

$O\left(n^{1-2/\omega}\right) = O(n^{0.158})$	Ring	[Censor-Hillel, Kaski, Korhonen, Lenzen, Paz, Suomela '15]
$O\left(n^{1/3}\right)$	Semiring	
	Rectangular, Multiple instances, more	[Le Gall '16]

Computing Distances using Matrix Multiplication

<table>
<tr><td>$O(n^{0.158})$</td><td>• Exact unweighted undirected APSP
• $\left(1+o(1)\right)$-approximation for weighted directed APSP</td><td rowspan="2">[Censor-Hillel, Kaski, Korhonen, Lenzen, Paz, Suomela ‘15]</td></tr>
<tr><td>$\tilde{O}\left(n^{1/3}\right)$</td><td>Exact weighted directed APSP</td></tr>
<tr><td>$O(n^{0.2096})$</td><td>Exact APSP in directed graphs with constant weights</td><td>[Le Gall ‘16]</td></tr>
</table>

Computing Distances using Matrix Multiplication

<table>
<tr><td>$O(n^{0.158})$</td><td>• Exact unweighted undirected APSP
• $\big(1 + o(1)\big)$-approximation for weighted directed APSP</td><td rowspan="2">[Censor-Hillel, Kaski, Korhonen, Lenzen, Paz, Suomela '15]</td></tr>
<tr><td>$\tilde{O}\left(n^{1/3}\right)$</td><td>Exact weighted directed APSP</td></tr>
<tr><td>$O(n^{0.2096})$</td><td>Exact APSP in directed graphs with constant weights</td><td>[Le Gall '16]</td></tr>
</table>

All complexities are polynomial!

What about approximations?

- We can compute a **spanner**: a sparse subgraph that approximates the distances.

$\tilde{O}(n^{1/k})$	$(2k-1)$-approximation for **weighted undirected** APSP

What about approximations?

- We can compute a **spanner**: a sparse subgraph that approximates the distances.

$\tilde{O}(n^{1/k})$	$(2k-1)$-approximation for **weighted undirected** APSP

Still polynomial for any constant k!

Computing Distances in the Congested Clique

Can we get constant approximation for APSP in sub-polynomial time?

Computing Distances in the Congested Clique

Can we get constant approximation for APSP in sub-polynomial time?

- <u>For SSSP:</u>

$O(\epsilon^{-3}\text{polylog}\, n)$-round $(1+\epsilon)$-approximation [Becker, Karrenbauer, Krinninger, Lenzen ‘17]

Computing Distances in the Congested Clique

Can we get constant approximation for APSP in sub-polynomial time?

- <u>For SSSP:</u>

$O(\epsilon^{-3} \text{polylog}\, n)$-round $(1+\epsilon)$-approximation [Becker, Karrenbauer, Krinninger, Lenzen ‘17]

Only for a single source!

Our Results: APSP

$O(\log^2 n/\epsilon)$	• $(2+\epsilon)$-approximation for **unweighted undirected** APSP • $(3+\epsilon)$-approximation for **weighted undirected** APSP

First polylog constant-factor approximation!

$(2-\epsilon)$-APSP implies MM	[Dor, Halperin, Zwick ‘00 Korhonen, Suomela ‘18]

Our Results: MSSP and more

$O(\log^2 n/\epsilon)$	$(1+\epsilon)$-approximation weighted undirected **MSSP** with $O(n^{1/2})$ sources
$O(\log^2 n/\epsilon)$	Near $(3/2)$-approximation for **diameter**
$\tilde{O}(n^{1/6})$	Exact weighted undirected **SSSP**

Previous results:

$\tilde{O}(n^{1/3})$	Exact weighted **SSSP**	[Censor-Hillel, Kaski, Korhonen, Lenzen, Paz, Suomela '15]
$O(\epsilon^{-3}\text{polylog}\, n)$	$(1+\epsilon)$-**SSSP**	[Becker, Karrenbauer, Krinninger, Lenzen '17]

Our Techniques

- We can multiply *sparse* matrices faster:

$O\left(1 + \frac{(\rho_S \rho_T)^{1/3}}{n^{1/3}}\right)$	Semiring, Sparse	[Censor-Hillel, Leitersdorf, Turner '18]

- ρ_A= density of A, the average number of non-zero entries on a row
- Example: $O(1)$ rounds for $O(n^{3/2})$ edges.

Our Techniques

- We can multiply *sparse* matrices faster.
- How can we use this?

  - Even if A is sparse, A^2 can be dense.

  - We want to compute distances in *general* graphs.

Our Techniques

- We can multiply *sparse* matrices faster.
- How can we use this?

  - Even if A is sparse, A^2 can be dense.

  - We want to compute distances in *general* graphs.

Many *building blocks* for distance computation are actually based on computations in *sparse* graphs

Building blocks for distance computation

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

Building blocks for distance computation

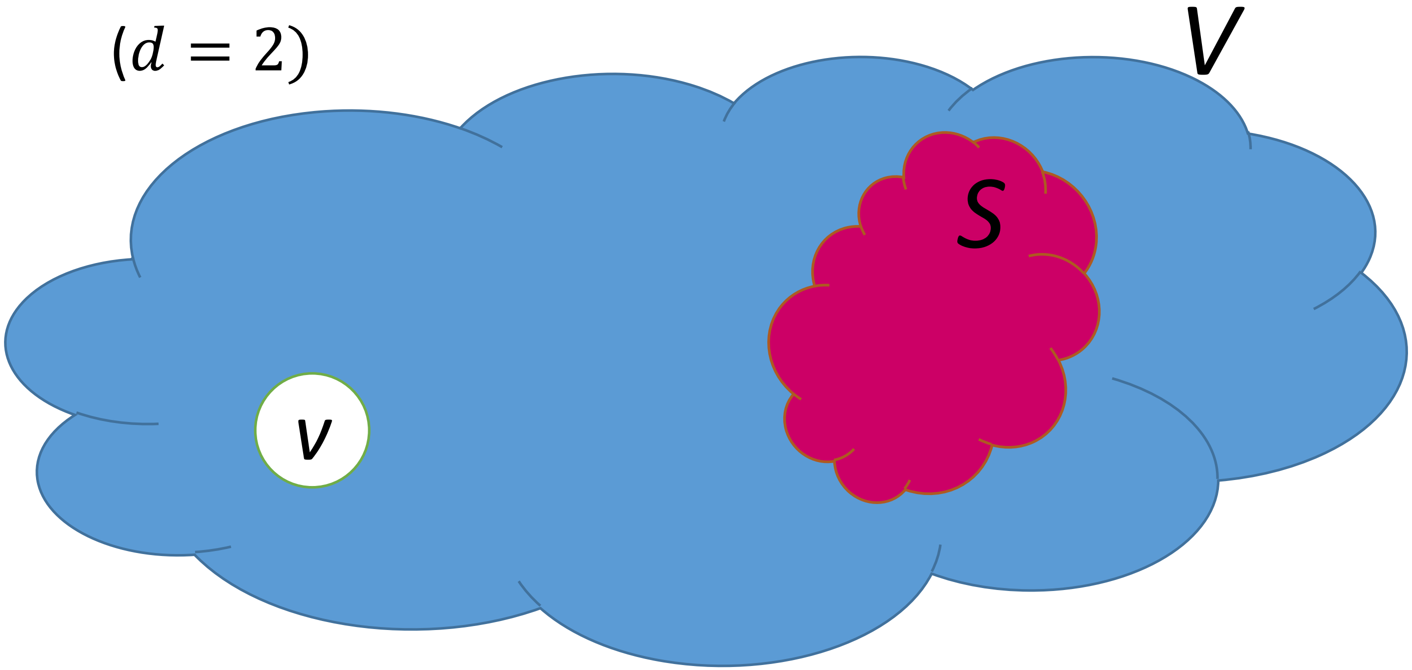
- k-nearest: for each vertex, compute distances to k nearest vertices
- (S, d, k)-source detection: for each vertex, distances to k nearest sources in S, up to hop d

Building blocks for distance computation

- k-nearest: for each vertex, compute distances to k nearest vertices
- (S, d, k)-source detection: for each vertex, distances to k nearest sources in S, up to hop d

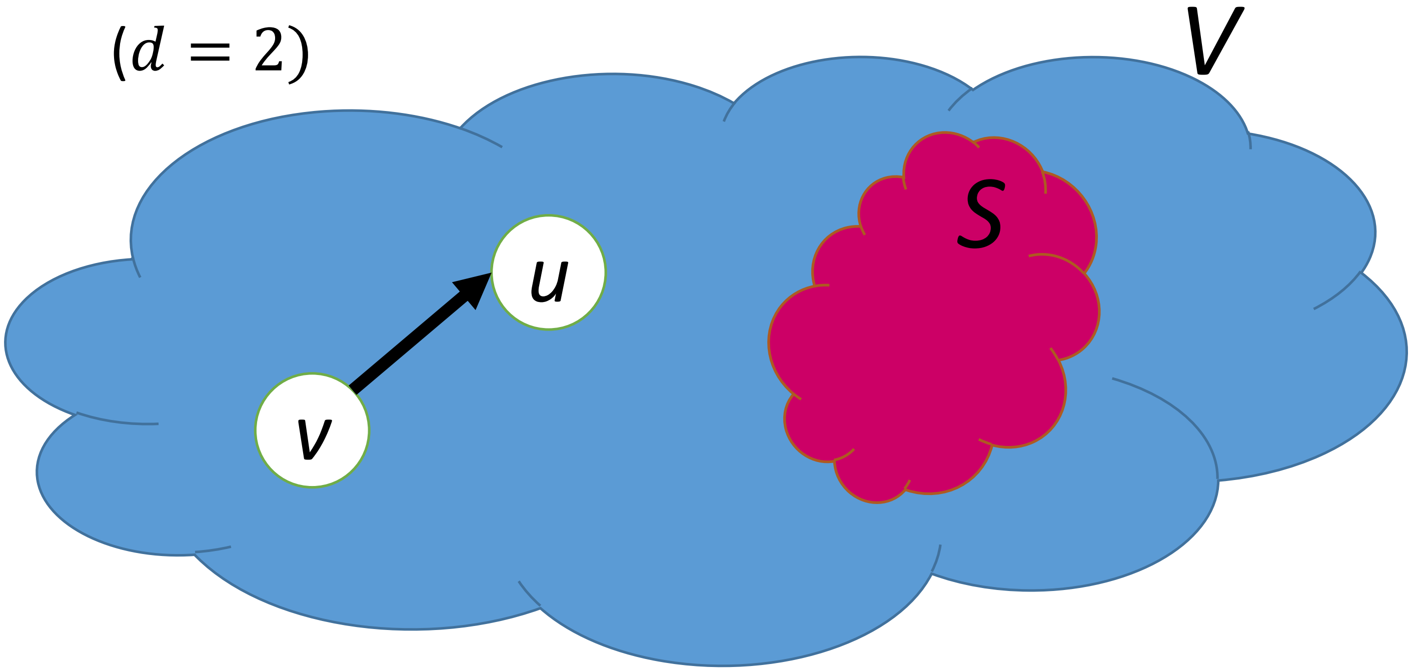
Building blocks for distance computation

- (S, d, k)-source detection: for each vertex, compute distances to k nearest sources in S, up to hop d



Building blocks for distance computation

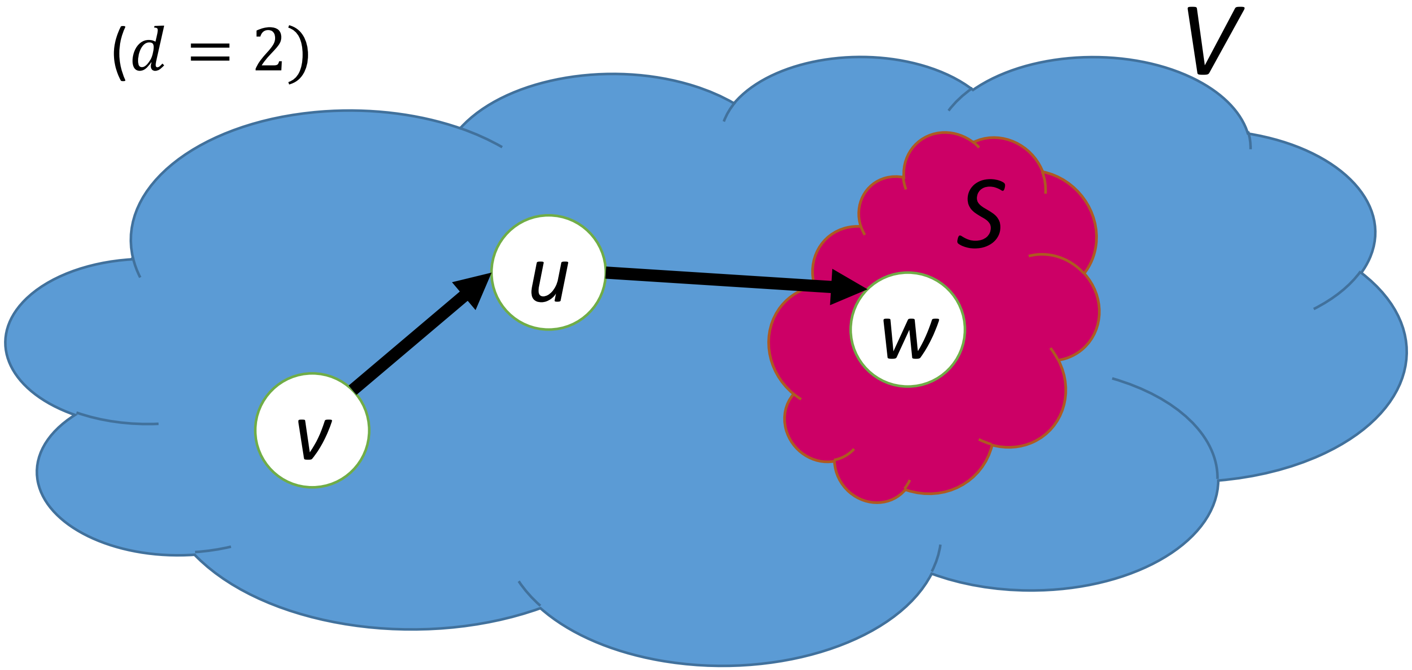
- (S, d, k)-source detection: for each vertex, compute distances to k nearest sources in S, up to hop d



Building blocks for distance computation

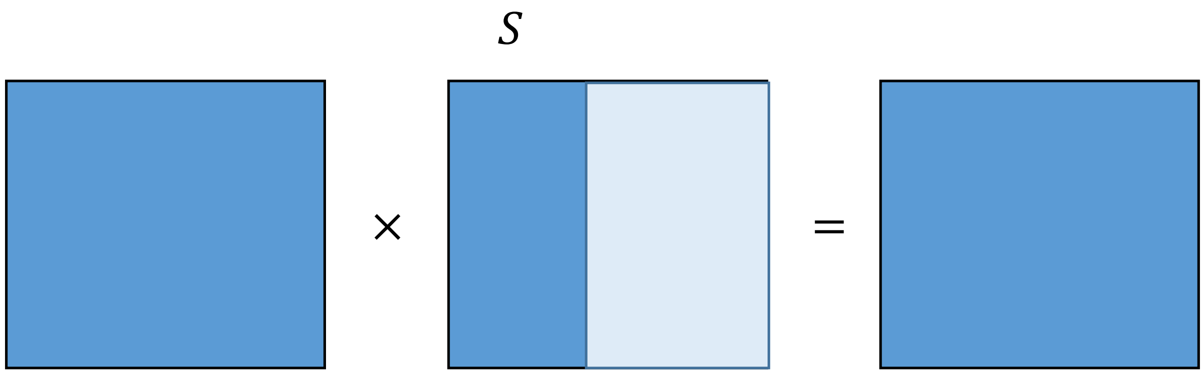
- (S, d, k)-source detection: for each vertex, compute distances to k nearest sources in S, up to hop d

$(d = 2)$



Building blocks for distance computation

- (S, d, k)-source detection: for each vertex, compute distances to k nearest sources in S, up to hop d

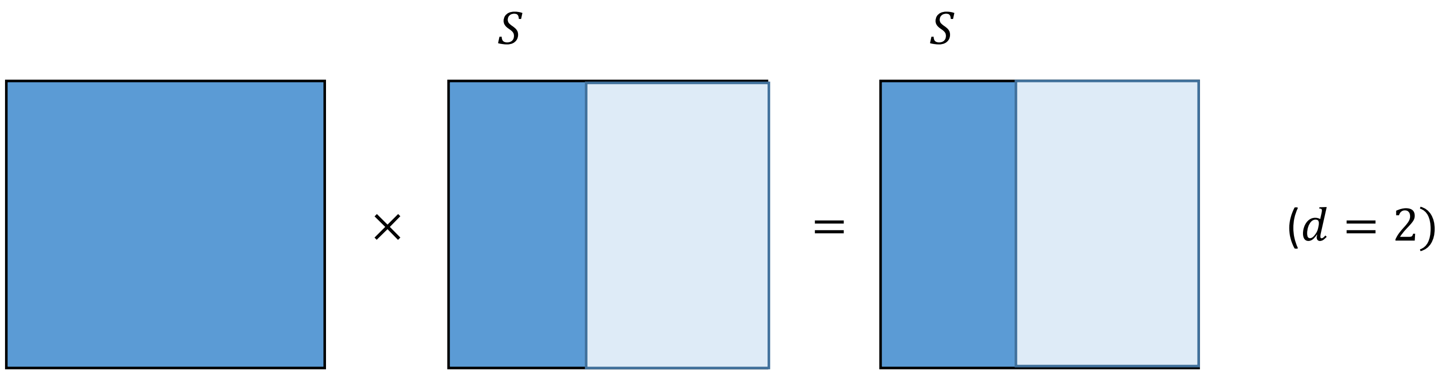


$(d = 2)$

Multiplication of sparse matrix by dense matrix: previous MM algorithm is *still polynomial*

Building blocks for distance computation

- (S, d, k)-source detection: for each vertex, compute distances to k nearest sources in S, up to hop d



Output matrix is also sparse!

Building blocks for distance computation

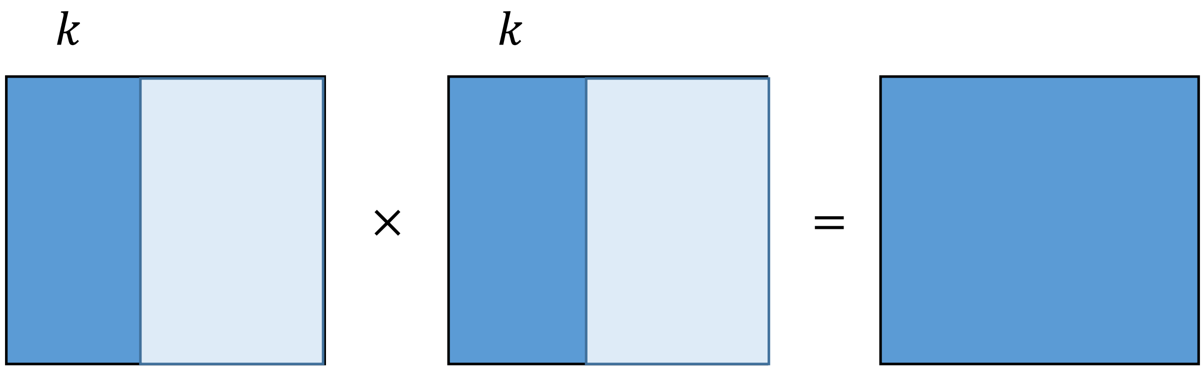
- k-nearest: for each vertex, compute distances to k nearest vertices
- (S, d, k)-source detection: for each vertex, distances to k nearest sources in S, up to hop d

Building blocks for distance computation

- k-nearest: for each vertex, compute distances to k nearest vertices
- (S, d, k)-source detection: for each vertex, distances to k nearest sources in S, up to hop d

Building blocks for distance computation

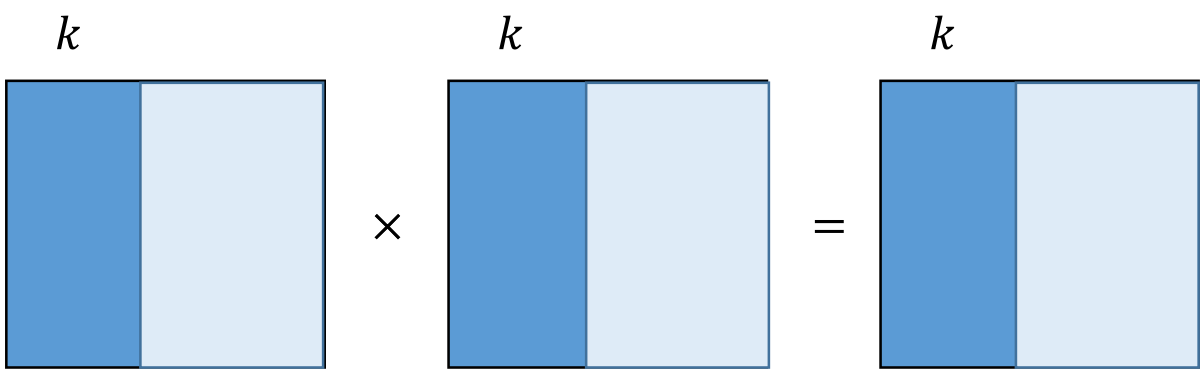
- k-nearest: for each vertex, compute distances to k nearest vertices



It's enough to look only at the k closest vertices to each vertex

Building blocks for distance computation

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



It's enough to look only at the k closest vertices to each vertex: also in the *output*

We don't know the identity of the k closest vertices before the computation

New Matrix Multiplication algorithm

- Previous algorithm:

$O\left(1+\frac{(\rho_S\rho_T)^{1/3}}{n^{1/3}}\right)$	Semiring, Sparse	[Censor-Hillel, Leitersdorf, Turner '18]

- Our algorithm:

$O\left(1+\frac{(\rho_S\rho_T\rho_P)^{1/3}}{n^{2/3}}\right)$	Semiring, Sparse	[Censor-Hillel, Dory, Korhonen, Leitersdorf, '19]

$S \times T = P$

New Matrix Multiplication algorithm

- Our algorithm:

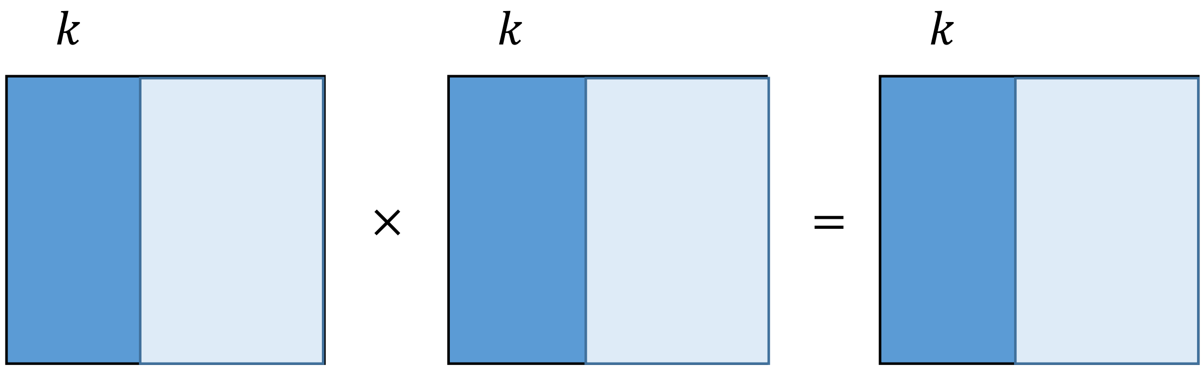
$O\left(1+\frac{(\rho_S \rho_T \rho_P)^{1/3}}{n^{2/3}}\right)$	Semiring, Sparse	[Censor-Hillel, Dory, Korhonen, Leitersdorf, '19]

- Depends also on the sparsity of the *output* matrix
- Even if we don't know the structure of the output matrix, we can *sparsify* the output matrix *on-the-fly*, keeping only ρ_P smallest entries for each row

Application: Distance Tools

k-nearest:

$O\left(\left(\frac{k}{n^{2/3}} + \log n\right)\log k\right)$ rounds ⇨ $O(\log^2 n)$ for $k = n^{2/3}$

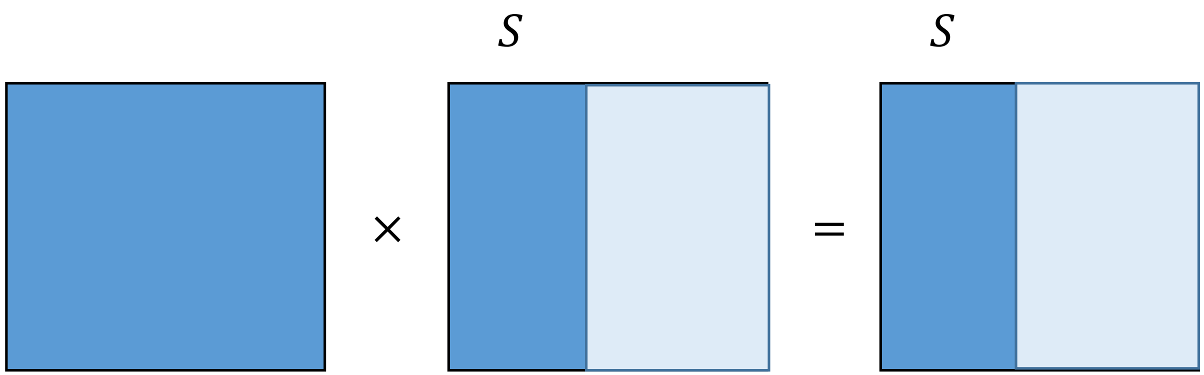


Application: Distance Tools

(S, d, k)-source detection:

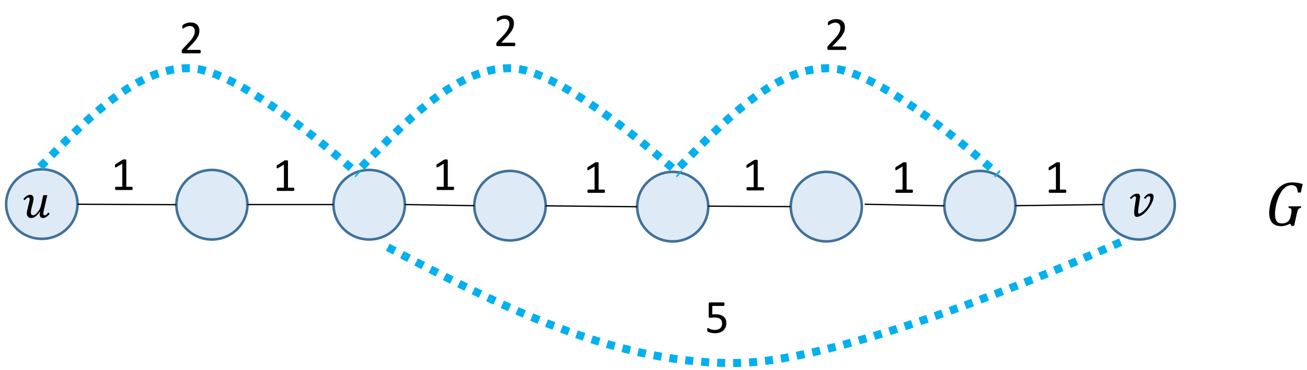
$O\left(\left(\frac{m^{1/3}|S|^{2/3}}{n}+1\right)d\right)$ rounds ($m =$ number of edges)

- To exploit the sparsity we need $\boldsymbol{d = n}$ multiplications - too expensive!



Solution: Hopsets

(β, ϵ)-hopset H:
A graph $H = (V, E')$, such that the β-hop distances in $G \cup H$ give $(1 + \epsilon)$-approximation for the distances in G



Enough to look at β-hop distances!

Solution: Hopsets

(β, ϵ)-hopset H:
A graph $H = (V, E')$, such that the β-hop distances in $G \cup H$ give $(1 + \epsilon)$-approximation for the distances in G

Our goal: to have small β and small running time t

What is known?

- We can get $\beta = t = O\left(\frac{\log \log n}{\epsilon}\right)^{\log \log n}$
  [Elkin, Neiman '17]

Solution: Hopsets

(β, ϵ)-hopset H:
A graph $H = (V, E')$, such that the β-hop distances in $G \cup H$ give $(1 + \epsilon)$-approximation for the distances in G

Our goal: to have small β and small running time t

What is known?

- We can get $\beta = t = O\left(\frac{\log \log n}{\epsilon}\right)^{\log \log n}$
  [Elkin, Neiman '17]

Can we get a poly-logarithmic complexity?

Solution: Hopsets

(β, ϵ)-hopset H:
A graph $H = (V, E')$, such that the β-hop distances in $G \cup H$ give $(1+\epsilon)$-approximation for the distances in G

Our goal: to have small β and small running time t

What is known?

- We can get $\beta = t = O\left(\frac{\log\log n}{\epsilon}\right)^{\log\log n}$ [Elkin, Neiman '17]

Yes! we can get: $\beta = O\left(\frac{\log n}{\epsilon}\right)$, $t = O\left(\frac{\log^2 n}{\epsilon}\right)$

Solution: Hopsets

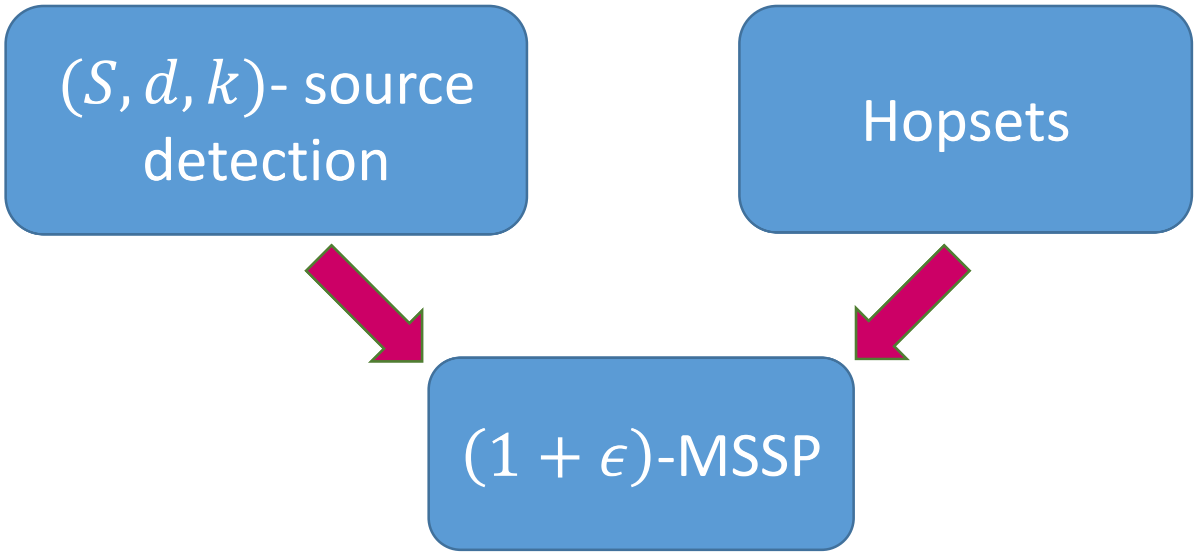
(β, ϵ)-hopset H:
A graph $H = (V, E')$, such that the β-hop distances in $G \cup H$ give $(1+\epsilon)$-approximation for the distances in G

Our goal: to have small β and small running time t

- We can get: $\beta = O\left(\frac{\log n}{\epsilon}\right)$, $t = O\left(\frac{\log^2 n}{\epsilon}\right)$

Idea: using our *distance tools* we can implement efficiently the hopset construction of [Elkin, Neiman '17] [Huang, Pettie '19] [Thorup, Zwick '06]

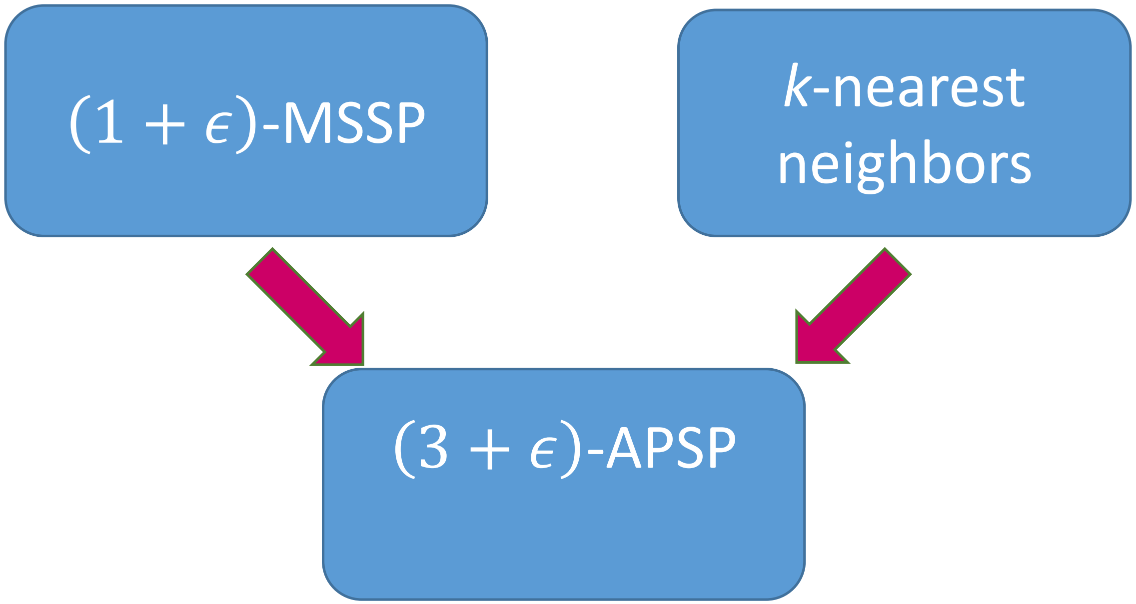
Applications: MSSP



Complexity: $O\left(\left(\frac{|S|^{2/3}}{n^{1/3}} + \log n\right)\frac{\log n}{\epsilon}\right)$

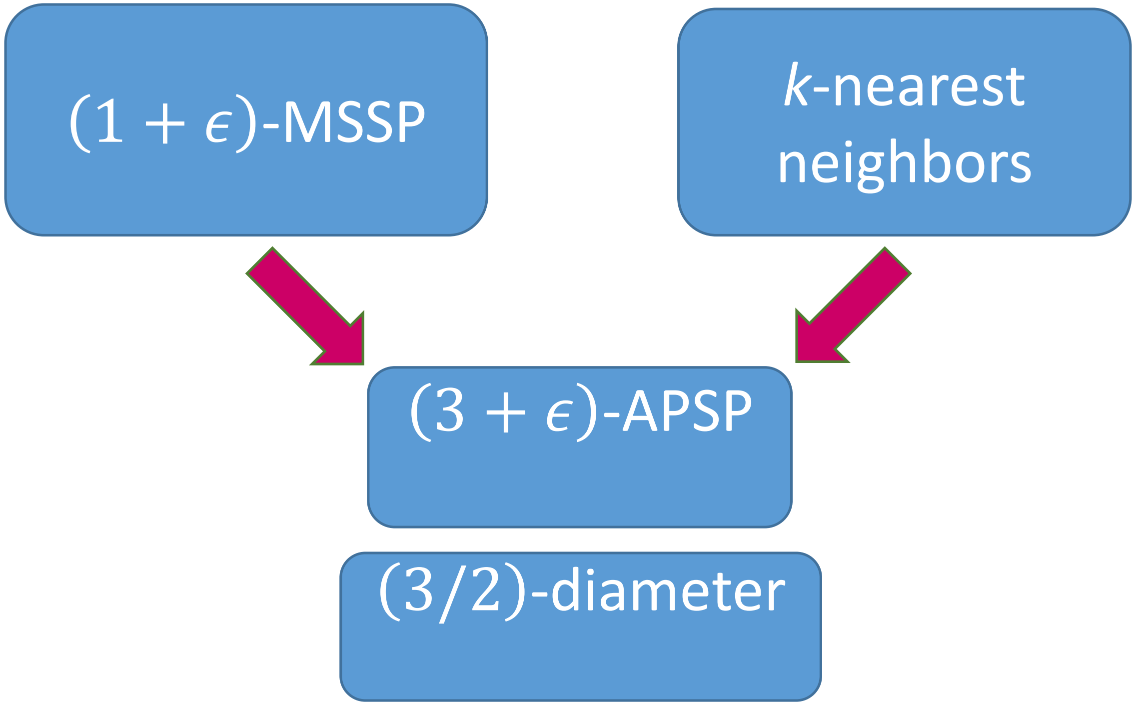
➡ **poly-logarithmic** for $|S| = O(\sqrt{n})$

Applications: weighted APSP



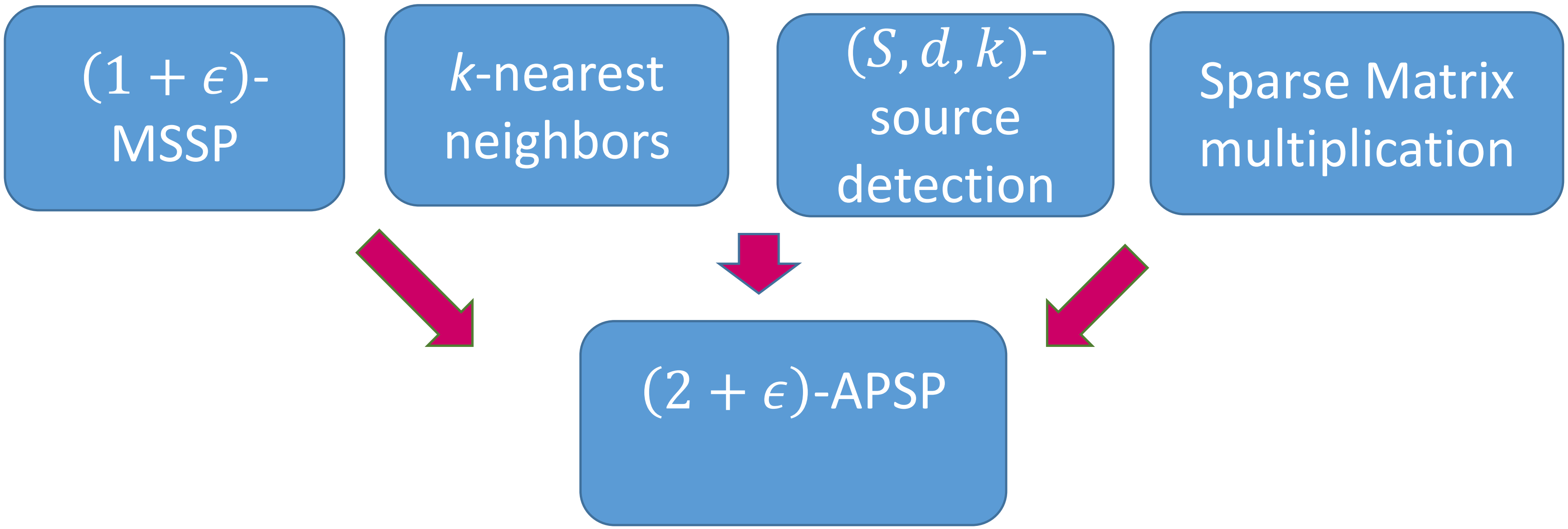
Complexity: $O\left(\frac{\log^2 n}{\epsilon}\right)$

Applications: APSP, diameter



Complexity: $O\left(\frac{\log^2 n}{\epsilon}\right)$

Applications: **unweighted** APSP



<u>Complexity:</u> $O\left(\frac{\log^2 n}{\epsilon}\right)$

Conclusion

- We show a fast algorithm for matrix multiplication that depends on the *sparsity* and is *output*-sensitive.
- Allows to build efficient *distance tools.*
- Together with *hopsets*: polylog algorithms for MSSP, APSP.

Summary

$O(\log^2 n/\epsilon)$	• $(2+\epsilon)$-approximation for **unweighted undirected** APSP • $(3+\epsilon)$-approximation for **weighted undirected** APSP
$O(\log^2 n/\epsilon)$	$(1+\epsilon)$-approximation for **weighted** undirected **MSSP** with $O(n^{1/2})$ sources
$O(\log^2 n/\epsilon)$	Near $(3/2)$-approximation for **diameter**
$\tilde{O}(n^{1/6})$	Exact weighted undirected **SSSP**

Open Questions

- Additional applications for distance tools
- Can we get a $(2+\epsilon)$-approximation for weighted APSP?
- Can we get sub-polynomial algorithm for exact SSSP? Or directed SSSP?