Fast Deterministic Fully Dynamic Distance Approximation

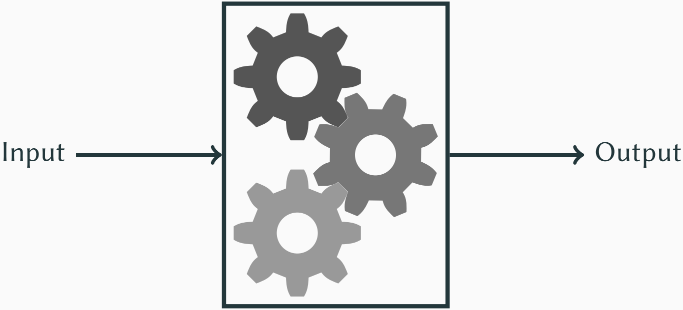
Sebastian Forster

Meeting on Algorithmic Challenges of Big Data 2022

University of Salzburg

Joint work with: Jan van den Brand   Yasamin Nazari

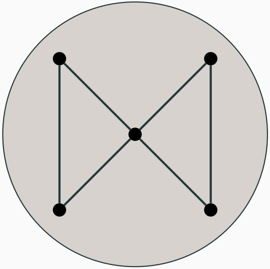
Static Approach



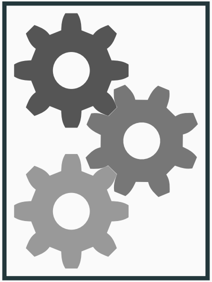
Dynamic Environments

Dynamic Distance Maintenance

Input graph G



Algorithm

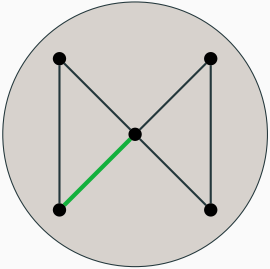


Distance Matrix

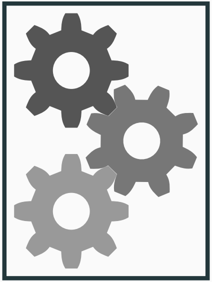
$$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 2 & 2 \\ 1 & 2 & 2 & 0 & 1 \\ 1 & 2 & 2 & 1 & 0 \end{pmatrix}$$

Dynamic Distance Maintenance

Input graph G



Algorithm



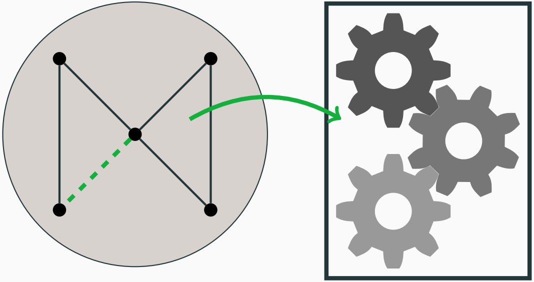
Distance Matrix

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 2 & 2 \\ 1 & 2 & 2 & 0 & 1 \\ 1 & 2 & 2 & 1 & 0 \end{pmatrix}$$

Adversary inserts
and deletes edges

Dynamic Distance Maintenance

Input graph G — Algorithm — Distance Matrix

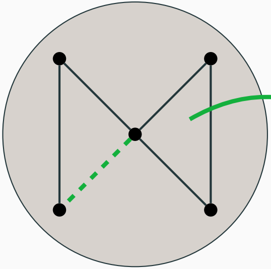


$$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 2 & 2 \\ 1 & 2 & 2 & 0 & 1 \\ 1 & 2 & 2 & 1 & 0 \end{pmatrix}$$

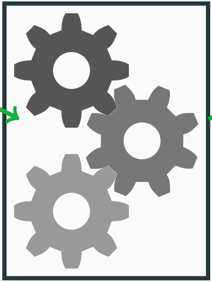
Adversary inserts
and deletes edges

Dynamic Distance Maintenance

Input graph G



Algorithm



Distance Matrix

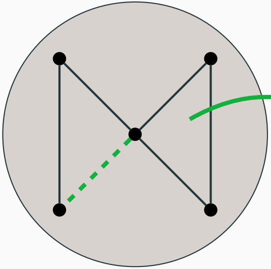
$$\begin{pmatrix} 0 & \mathbf{2} & 1 & 1 & 1 \\ \mathbf{2} & 0 & 1 & \mathbf{3} & \mathbf{3} \\ 1 & 1 & 0 & 2 & 2 \\ \mathbf{3} & 2 & 2 & 0 & 1 \\ \mathbf{3} & 2 & 2 & 1 & 0 \end{pmatrix}$$

Adversary inserts and deletes edges

Algorithm updates distance matrix

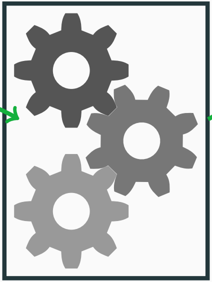
Dynamic Distance Maintenance

Input graph G



Adversary inserts and deletes edges

Algorithm



Distance Matrix

$$\begin{pmatrix} 0 & \mathbf{2} & 1 & 1 & 1 \\ \mathbf{2} & 0 & 1 & \mathbf{3} & \mathbf{3} \\ 1 & 1 & 0 & 2 & 2 \\ \mathbf{3} & 2 & 2 & 0 & 1 \\ \mathbf{3} & 2 & 2 & 1 & 0 \end{pmatrix}$$

Algorithm updates distance matrix

State of the Art

Amortized update time $\tilde{O}(n^2)$ [Demetrescu, Italiano '03]

Subquadratic Update Time: State of the Art

- **Update-query time trade-offs:**
  - exact: [Sankowski ’05] [v.d. Brand, Nanongkai, Saranurak ’18]
  - $(1+\epsilon)$-approximation: [v.d. Brand, Nanongkai ’18]

Subquadratic Update Time: State of the Art

- **Update-query time trade-offs:**
  - exact: [Sankowski '05] [v.d. Brand, Nanongkai, Saranurak '18]
  - $(1+\epsilon)$-approximation: [v.d. Brand, Nanongkai '18]
- **Partial information (single source, single pair):**
  - exact: [Sankowski '05]
  - $(1+\epsilon)$-approximation: [v.d. Brand, Nanongkai '18] [Bergamaschi, Henzinger, Gutenberg, Vassilevska Williams, Wein '21]

Subquadratic Update Time: State of the Art

- **Update-query time trade-offs:**
  - exact: [Sankowski ’05] [v.d. Brand, Nanongkai, Saranurak ’18]
  - $(1+\epsilon)$-approximation: [v.d. Brand, Nanongkai ’18]
- **Partial information (single source, single pair):**
  - exact: [Sankowski ’05]
  - $(1+\epsilon)$-approximation: [v.d. Brand, Nanongkai ’18] [Bergamaschi, Henzinger, Gutenberg, Vassilevska Williams, Wein ’21]
- **Large multiplicative stretch:**
  - Dynamic spanners: [Ausiello, Franciosa, Italiano ’05] [Elkin ’07] [Baswana, Khurana, Sarkar ’12], [Bodwin, K ’16] [Bernstein, F, Henzinger ’19] [Bernstein, v.d. Brand, Gutenberg, Nanongkai, Saranurak, Sidford, Sun ’22]
  - Dynamic distance oracles: [Abraham, Chechik, Talwar ’14] [F, Goranci, Henzinger ’21]

“Gold standard”:

- Fully dynamic
- Worst-case update time
- Deterministic
- Meet an update-time barrier

Towards Dynamic Algorithms without Caveats

“Gold standard”:

- Fully dynamic
- Worst-case update time
- Deterministic
- Meet an update-time barrier

List of problems with such algorithms is small

Towards Dynamic Algorithms without Caveats

“Gold standard”:

- Fully dynamic
- Worst-case update time
- Deterministic
- Meet an update-time barrier

List of problems with such algorithms is small

Contribution

We add to this list: $(1+\epsilon)$-approximate distance approximation in unweighted, undirected graphs [van den Brand, **F**, Nazari arXiv ’21]

Our Results

Distance approximation in unweighted, undirected graphs:

Approx.	Type	Update Time
$1+\epsilon$	single pair	$O(n^{1.407})$
$1+\epsilon$	single source	$O(n^{1.529})$
$1+\epsilon$	k sources	$O(n^{1.529}+kn^{1+o(1)})$
$1+\epsilon$	all pairs	$O(n^{2+o(1)})$

Our Results

Distance approximation in unweighted, undirected graphs:

Approx.	Type	Update Time
$1+\epsilon$	single pair	$O(n^{1.407})$
$1+\epsilon$	single source	$O(n^{1.529})$
$1+\epsilon$	k sources	$O(n^{1.529} + kn^{1+o(1)})$
$1+\epsilon$	all pairs	$O(n^{2+o(1)})$

- Improvement from randomized to deterministic
  (and smaller update time in case of single pair)

Our Results

Distance approximation in unweighted, undirected graphs:

Approx.	Type	Update Time
$1+\epsilon$	single pair	$O(n^{1.407})$
$1+\epsilon$	single source	$O(n^{1.529})$
$1+\epsilon$	k sources	$O(n^{1.529} + kn^{1+o(1)})$
$1+\epsilon$	all pairs	$O(n^{2+o(1)})$

- Improvement from randomized to deterministic
  (and smaller update time in case of single pair)
- Update times match (conditional) lower bounds [van den Brand, Nanongkai, Saranurak '19]

Further Results

Randomized Algorithms

- *Exact* single-pair distance: $O(n^{1.704})$
  (Improves upon $O(n^{1.724})$ [Sankowski '05] [v.d. Brand, Nanongkai, Saranurak '19])

Further Results

Randomized Algorithms

- *Exact* single-pair distance: $O(n^{1.704})$
  (Improves upon $O(n^{1.724})$ [Sankowski '05] [v.d. Brand, Nanongkai, Saranurak '19])
- "Nearly" $(\frac{3}{2} + \epsilon)$-approximation of diameter: $O(n^{1.596})$
  (Improves upon $O(n^{1.779})$ [v.d. Brand, Nanongkai '19]

Further Results

Randomized Algorithms

- *Exact* single-pair distance: $O(n^{1.704})$
  (Improves upon $O(n^{1.724})$ [Sankowski '05] [v.d. Brand, Nanongkai, Saranurak '19])
- "Nearly" $(\frac{3}{2} + \epsilon)$-approximation of diameter: $O(n^{1.596})$
  (Improves upon $O(n^{1.779})$ [v.d. Brand, Nanongkai '19]
- Update/query trade-off for $(1 + \epsilon)$-approximate distance:
  $O(n^{1.788})$ update time / $O(n^{0.45})$ query time
  (Improves upon $O(n^{1.862})$ / $O(n^{0.45})$ [v.d. Brand, Nanongkai '19]

Further Results

Randomized Algorithms

- *Exact* single-pair distance: $O(n^{1.704})$
  (Improves upon $O(n^{1.724})$ [Sankowski '05] [v.d. Brand, Nanongkai, Saranurak '19])
- "Nearly" $(\frac{3}{2} + \epsilon)$-approximation of diameter: $O(n^{1.596})$
  (Improves upon $O(n^{1.779})$ [v.d. Brand, Nanongkai '19]
- Update/query trade-off for $(1 + \epsilon)$-approximate distance:
  $O(n^{1.788})$ update time / $O(n^{0.45})$ query time
  (Improves upon $O(n^{1.862})$ / $O(n^{0.45})$ [v.d. Brand, Nanongkai '19]

Warm Up

Randomized fully dynamic $(1 + \epsilon)$-approximate single-source distances with worst-case update time $O(n^{1.529})$.

Our Approach

Idea

Maintain sparsifier and recompute from scratch on sparsifier

Our Approach

Idea

Maintain sparsifier and recompute from scratch on sparsifier

- Maintain hitting set for neighbors of nodes of degree $\geq \sqrt{n}$
- Maintain $\Theta(1/\epsilon)$-bounded distances to all nodes from hitting set nodes and source node s

Our Approach

Idea

Maintain sparsifier and recompute from scratch on sparsifier

- Maintain hitting set for neighbors of nodes of degree $\geq \sqrt{n}$
- Maintain $\Theta(1/\epsilon)$-bounded distances to all nodes from hitting set nodes and source node s
- Additionally, after each update:
  - Obtain $\Theta(1/\epsilon)$-bounded distances $\hat{d}_G(\cdot, \cdot)$
  - Compute $(1+\epsilon, 2)$-emulator H of size $\tilde{O}(n^{1.5})$

Our Approach

Idea

Maintain sparsifier and recompute from scratch on sparsifier

- Maintain hitting set for neighbors of nodes of degree $\geq \sqrt{n}$
- Maintain $\Theta(1/\epsilon)$-bounded distances to all nodes from hitting set nodes and source node s
- Additionally, after each update:
  - Obtain $\Theta(1/\epsilon)$-bounded distances $\hat{d}_G(\cdot,\cdot)$
  - Compute $(1+\epsilon, 2)$-emulator H of size $\tilde{O}(n^{1.5})$
  - Compute (exact) single-source distances on H
  - Return $\min(\hat{d}_G(s,v), d_H(s,v))$ for every node v

Our Approach

Idea

Maintain sparsifier and recompute from scratch on sparsifier

- Maintain hitting set for neighbors of nodes of degree $\geq \sqrt{n}$
- Maintain $\Theta(1/\epsilon)$-bounded distances to all nodes from hitting set nodes and source node s
- Additionally, after each update:
  - Obtain $\Theta(1/\epsilon)$-bounded distances $\hat{d}_G(\cdot, \cdot)$
  - Compute $(1+\epsilon, 2)$-emulator H of size $\tilde{O}(n^{1.5})$
  - Compute (exact) single-source distances on H
  - Return $\min(\hat{d}_G(s, v), d_H(s, v))$ for every node v

Related Work

Randomized algorithm for maintaining $(1+\epsilon, n^{o(1)})$-spanner of size $n^{1+o(1)}$ with update time $O(n^{1.529})$ [Bergamaschi et al. '21]

Hitting Set

Hitting Set

We maintain a set of nodes $S \subseteq V$ of size $\tilde{O}(\sqrt{n})$ such that every heavy node of degree $> d := \sqrt{n}$ has at least one node of S in its neighborhood.

Hitting Set

Hitting Set

We maintain a set of nodes $S \subseteq V$ of size $\tilde{O}(\sqrt{n})$ such that every heavy node of degree $> d := \sqrt{n}$ has at least one node of S in its neighborhood.

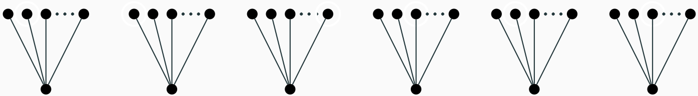
Randomized approach: Initially, sample a set of size $\tilde{\Theta}(\sqrt{n})$ uniformly at random [Ullman, Yannakakis '90]

Hitting Set

Hitting Set

We maintain a set of nodes $S \subseteq V$ of size $\tilde{O}(\sqrt{n})$ such that every heavy node of degree $> d := \sqrt{n}$ has at least one node of S in its neighborhood.

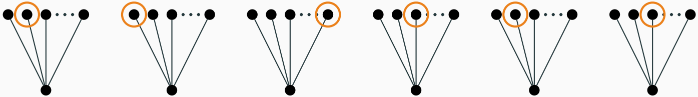
Randomized approach: Initially, sample a set of size $\tilde{\Theta}(\sqrt{n})$ uniformly at random [Ullman, Yannakakis '90]



Hitting Set

We maintain a set of nodes $S \subseteq V$ of size $\tilde{O}(\sqrt{n})$ such that every heavy node of degree $> d := \sqrt{n}$ has at least one node of S in its neighborhood.

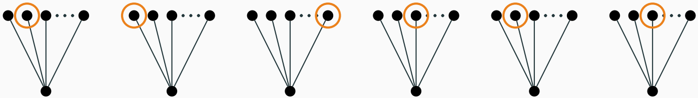
Randomized approach: Initially, sample a set of size $\tilde{\Theta}(\sqrt{n})$ uniformly at random [Ullman, Yannakakis '90]



Hitting Set

We maintain a set of nodes $S \subseteq V$ of size $\tilde{O}(\sqrt{n})$ such that every heavy node of degree $> d := \sqrt{n}$ has at least one node of S in its neighborhood.

Randomized approach: Initially, sample a set of size $\tilde{\Theta}(\sqrt{n})$ uniformly at random [Ullman, Yannakakis ’90]



Adversarial model

Only works against an oblivious adversary

Emulator Construction

Definition

A $(1+\epsilon, \beta)$-**emulator** of $G = (V, E)$ is a graph $H = (V, E')$ such that

$$dist_G(u, v) \leq dist_H(u, v) \leq (1+\epsilon) \cdot dist_G(u, v) + \beta$$

for all pairs of nodes $u, v \in V$.

Emulator Construction

Definition

A $(1+\epsilon,\beta)$-**emulator** of $G=(V,E)$ is a graph $H=(V,E')$ such that

$$dist_G(u,v) \leq dist_H(u,v) \leq (1+\epsilon)\cdot dist_G(u,v)+\beta$$

for all pairs of nodes $u,v \in V$.

Emulator H has two types of edges:

- For every light node of degree $\leq \sqrt{n}$: edges to all neighbors
- For every node in hitting set: (weighted) edges to all nodes in distance $\leq \lceil 6/\epsilon \rceil$

similar to [Henzinger, **K**, Nanongkai '13; Dor, Halperin, Zwick '97]

Emulator Construction

Definition

A $(1+\epsilon, \beta)$-**emulator** of $G = (V, E)$ is a graph $H = (V, E')$ such that

$$dist_G(u, v) \leq dist_H(u, v) \leq (1+\epsilon) \cdot dist_G(u, v) + \beta$$

for all pairs of nodes $u, v \in V$.

Emulator H has two types of edges:

- For every light node of degree $\leq \sqrt{n}$: edges to all neighbors
- For every node in hitting set: (weighted) edges to all nodes in distance $\leq \lceil 6/\epsilon \rceil$

similar to [Henzinger, **K**, Nanongkai '13; Dor, Halperin, Zwick '97]

Lemma

H is a $(1+\frac{\epsilon}{2}, 2)$-emulator of size $\tilde{O}(n^{1.5})$

Emulator Construction

Definition

A $(1+\epsilon, \beta)$-**emulator** of $G=(V,E)$ is a graph $H=(V,E')$ such that

$$dist_G(u,v) \leq dist_H(u,v) \leq (1+\epsilon) \cdot dist_G(u,v) + \beta$$

for all pairs of nodes $u, v \in V$.

Emulator H has two types of edges:

- For every light node of degree $\leq \sqrt{n}$: edges to all neighbors
- For every node in hitting set: (weighted) edges to all nodes in distance $\leq \lceil 6/\epsilon \rceil$

similar to [Henzinger, **K**, Nanongkai '13; Dor, Halperin, Zwick '97]

Lemma

H is a $(1+\frac{\epsilon}{2}, 2)$-emulator of size $\tilde{O}(n^{1.5})$

$\rightarrow$ single-source distance on H in time $\tilde{O}(n^{1.5})$

Approximation Guarantee

Subdivide any shortest path into segments of length $\lceil 6/\epsilon \rceil$ (with potentially one segment of smaller length)

Approximation Guarantee

Subdivide any shortest path into segments of length $\lceil 6/\epsilon \rceil$ (with potentially one segment of smaller length)

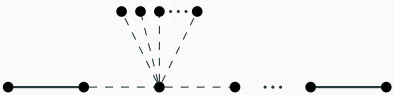
- Case 1: Segment contains no high-degree node

Approximation Guarantee

Subdivide any shortest path into segments of length $\lceil 6/\epsilon \rceil$ (with potentially one segment of smaller length)

- Case 1: Segment contains no high-degree node

- Case 2: Segment contains high-degree node

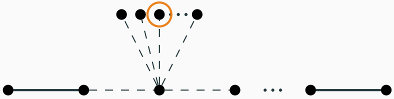


Approximation Guarantee

Subdivide any shortest path into segments of length $\lceil 6/\epsilon \rceil$ (with potentially one segment of smaller length)

- Case 1: Segment contains no high-degree node

- Case 2: Segment contains high-degree node

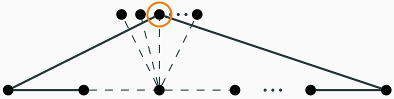


Approximation Guarantee

Subdivide any shortest path into segments of length $\lceil 6/\epsilon \rceil$ (with potentially one segment of smaller length)

- Case 1: Segment contains no high-degree node

- Case 2: Segment contains high-degree node



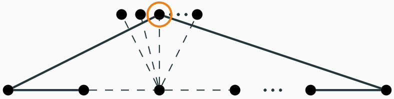
$\rightarrow$ Detour of additive surplus 2

Approximation Guarantee

Subdivide any shortest path into segments of length $\lceil 6/\epsilon \rceil$ (with potentially one segment of smaller length)

- Case 1: Segment contains no high-degree node

- Case 2: Segment contains high-degree node



$\rightarrow$ Detour of additive surplus 2

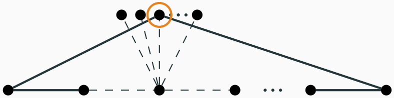
  - If segment has length $\lceil 6/\epsilon \rceil$, then multiplicative error of $\leq \frac{\lceil 6/\epsilon \rceil + 2}{\lceil 6/\epsilon \rceil} \leq \frac{6/\epsilon + 3}{6/\epsilon} = 1 + \frac{\epsilon}{2}$

Approximation Guarantee

Subdivide any shortest path into segments of length $\lceil 6/\epsilon \rceil$ (with potentially one segment of smaller length)

- Case 1: Segment contains no high-degree node

- Case 2: Segment contains high-degree node



$\rightarrow$ Detour of additive surplus 2

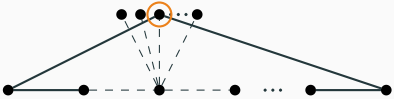
- If segment has length $\lceil 6/\epsilon \rceil$, then multiplicative error of $\leq \frac{\lceil 6/\epsilon \rceil + 2}{\lceil 6/\epsilon \rceil} \leq \frac{6/\epsilon + 3}{6/\epsilon} = 1 + \frac{\epsilon}{2}$
- If segment has length $< \lceil 6/\epsilon \rceil$, then additive error of 2

Approximation Guarantee

Subdivide any shortest path into segments of length $\lceil 6/\epsilon \rceil$ (with potentially one segment of smaller length)

- Case 1: Segment contains no high-degree node

- Case 2: Segment contains high-degree node



$\rightarrow$ Detour of additive surplus 2

- If segment has length $\lceil 6/\epsilon \rceil$, then multiplicative error of $\leq \frac{\lceil 6/\epsilon \rceil + 2}{\lceil 6/\epsilon \rceil} \leq \frac{6/\epsilon + 3}{6/\epsilon} = 1 + \frac{\epsilon}{2}$
- If segment has length $< \lceil 6/\epsilon \rceil$, then additive error of 2

Overall: multiplicative error of $1 + \frac{\epsilon}{2}$, additive error of 2

Algebraic Data Structure

Theorem ([Sankowski '05])

Given any $0 \leq \mu \leq 1$ and any sets $A, B \subseteq V$, there is a randomized data structure for maintaining the $A \times B$ distances up to $\leq h$ with update time $\tilde{O}((n^{\omega(1,\mu,1)-\mu} + n^{1+\mu} + |A| \cdot |B|) \cdot h)$.

Algebraic Data Structure

Theorem ([Sankowski '05])

Given any $0 \leq \mu \leq 1$ and any sets $A, B \subseteq V$, there is a randomized data structure for maintaining the $A \times B$ distances up to $\leq h$ with update time $\tilde{O}((n^{\omega(1,\mu,1)-\mu} + n^{1+\mu} + |A| \cdot |B|) \cdot h)$.

- $O(n^{\omega(1,\mu,1)})$ denotes time needed for multiplying an $n \times n^{\mu}$ matrix with an $n^{\mu} \times n$ matrix

Algebraic Data Structure

Theorem ([Sankowski '05])

Given any $0 \leq \mu \leq 1$ and any sets $A, B \subseteq V$, there is a randomized data structure for maintaining the $A \times B$ distances up to $\leq h$ with update time $\tilde{O}((n^{\omega(1,\mu,1)-\mu} + n^{1+\mu} + |A| \cdot |B|) \cdot h)$.

- $O(n^{\omega(1,\mu,1)})$ denotes time needed for multiplying an $n \times n^{\mu}$ matrix with an $n^{\mu} \times n$ matrix
- With $\mu = 0.528\ldots$, update time is $\tilde{O}((n^{1.529} + |A| \cdot |B|) \cdot h)$

Algebraic Data Structure

Theorem ([Sankowski '05])

Given any $0 \leq \mu \leq 1$ and any sets $A, B \subseteq V$, there is a randomized data structure for maintaining the $A \times B$ distances up to $\leq h$ with update time $\tilde{O}((n^{\omega(1,\mu,1)-\mu} + n^{1+\mu} + |A| \cdot |B|) \cdot h)$.

- $O(n^{\omega(1,\mu,1)})$ denotes time needed for multiplying an $n \times n^{\mu}$ matrix with an $n^{\mu} \times n$ matrix
- With $\mu = 0.528\ldots$, update time is $\tilde{O}((n^{1.529} + |A| \cdot |B|) \cdot h)$
- With $A = S \cup \{s\}$, $B = V$ (where $|S| = \tilde{O}(\sqrt{n})$), and $h = O(1/\epsilon)$: update time $O(n^{1.529}/\epsilon)$

Algebraic Data Structure

Theorem ([Sankowski '05])

Given any $0 \leq \mu \leq 1$ and any sets $A, B \subseteq V$, there is a randomized data structure for maintaining the $A \times B$ distances up to $\leq h$ with update time $\tilde{O}((n^{\omega(1,\mu,1)-\mu} + n^{1+\mu} + |A| \cdot |B|) \cdot h)$.

- $O(n^{\omega(1,\mu,1)})$ denotes time needed for multiplying an $n \times n^{\mu}$ matrix with an $n^{\mu} \times n$ matrix
- With $\mu = 0.528\ldots$, update time is $\tilde{O}((n^{1.529} + |A| \cdot |B|) \cdot h)$
- With $A = S \cup \{s\}$, $B = V$ (where $|S| = \tilde{O}(\sqrt{n})$), and $h = O(1/\epsilon)$: update time $O(n^{1.529}/\epsilon)$

Approximation Guarantee:

- If $d_G(s, v) \leq \lceil 6/\epsilon \rceil$: distance from algebraic data structure

Algebraic Data Structure

Theorem ([Sankowski '05])

Given any $0 \leq \mu \leq 1$ and any sets $A, B \subseteq V$, there is a randomized data structure for maintaining the $A \times B$ distances up to $\leq h$ with update time $\tilde{O}((n^{\omega(1,\mu,1)-\mu} + n^{1+\mu} + |A| \cdot |B|) \cdot h)$.

- $O(n^{\omega(1,\mu,1)})$ denotes time needed for multiplying an $n \times n^{\mu}$ matrix with an $n^{\mu} \times n$ matrix
- With $\mu = 0.528\ldots$, update time is $\tilde{O}((n^{1.529} + |A| \cdot |B|) \cdot h)$
- With $A = S \cup \{s\}$, $B = V$ (where $|S| = \tilde{O}(\sqrt{n})$), and $h = O(1/\epsilon)$: update time $O(n^{1.529}/\epsilon)$

Approximation Guarantee:

- If $d_G(s, v) \leq \lceil 6/\epsilon \rceil$: distance from algebraic data structure
- If $d_G(s, v) > \lceil 6/\epsilon \rceil$, then approximation from H becomes $(1 + \frac{\epsilon}{2})d_G(s, v) + 2 \leq (1 + \frac{\epsilon}{2})d_G(s, v) + \frac{\epsilon}{3}d_G(s, v) \leq (1 + \epsilon)d_G(s, v)$

Towards Deterministic Algorithm

Three ideas:

1. Randomization not necessary in algebraic data structure for very small distances

Towards Deterministic Algorithm

Three ideas:

1. Randomization not necessary in algebraic data structure for very small distances
2. Hitting set for neighborhoods can be maintained with a lazy approach giving low recourse

Towards Deterministic Algorithm

Three ideas:

1. Randomization not necessary in algebraic data structure for very small distances
2. Hitting set for neighborhoods can be maintained with a lazy approach giving low recourse
3. Algebraic data structure can be extended to slowly changing set of nodes

Dynamic Hitting Set

Our approach:

- Static recomputation: Time $O(nd)$
  Greedy algorithm: $O(\log n)$-approximation

Dynamic Hitting Set

Our approach:

- Static recomputation: Time $O(nd)$
  Greedy algorithm: $O(\log n)$-approximation
- Hitting set needs to be fixed after each update

Dynamic Hitting Set

Our approach:

- Static recomputation: Time $O(nd)$
  Greedy algorithm: $O(\log n)$-approximation
- Hitting set needs to be fixed after each update
  - Each update affects at most two neighborhoods!
  - Hitting set grows by ≤ 2 nodes with each update
    $\rightarrow$ size $\tilde{O}(n/d + k)$ after k updates; can set $k = n/d$

Dynamic Hitting Set

Our approach:

- Static recomputation: Time $O(nd)$
  Greedy algorithm: $O(\log n)$-approximation
- Hitting set needs to be fixed after each update
  - Each update affects at most two neighborhoods!
  - Hitting set grows by ≤ 2 nodes with each update
    $\rightarrow$ size $\tilde{O}(n/d + k)$ after k updates; can set $k = n/d$
- Simple amortized algorithm: Update time $\frac{O(nd)}{k} = O(d^2) = \tilde{O}(n)$
- Can make worst-case with standard technique

Dynamic Hitting Set

Our approach:

- Static recomputation: Time $O(nd)$
  Greedy algorithm: $O(\log n)$-approximation
- Hitting set needs to be fixed after each update
  - Each update affects at most two neighborhoods!
  - Hitting set grows by ≤ 2 nodes with each update
    $\rightarrow$ size $\tilde{O}(n/d + k)$ after k updates; can set $k = n/d$
- Simple amortized algorithm: Update time $\frac{O(nd)}{k} = O(d^2) = \tilde{O}(n)$
- Can make worst-case with standard technique

Dynamic Set Cover:

- Well studied problem [Gupta, Krishnaswamy, Panigrahi '17] [Abboud, Addanki, Grandoni, Panigrahi, Saha '19] [Bhattacharya, Henzinger, Nanongkai '19] [Bhattacharya, Henzinger, Nanongkai, Wu '21]
- Off-the shelf algorithms not applicable in our setting

Deterministic Path Counting

Interpretation of algebraic data structures:

- Maintain $M_{i,j}[k]$: number of paths from i to j of length exactly k

Deterministic Path Counting

Interpretation of algebraic data structures:

- Maintain $M_{i,j}[k]$: number of paths from i to j of length exactly k
- Entries might be as large as $\Theta(n^k)$
  $\rightarrow$ Field operation takes time $O(k \log n^k) = O(k^2 \log n)$
- Significant overhead!

Deterministic Path Counting

Interpretation of algebraic data structures:

- Maintain $M_{i,j}[k]$: number of paths from i to j of length exactly k
- Entries might be as large as $\Theta(n^k)$
  $\rightarrow$ Field operation takes time $O(k \log n^k) = O(k^2 \log n)$
- Significant overhead!

Randomized approach:

- Actually interested in smallest k for which $A_{i,j}[k] \neq 0$
- Less time per operation with computation modulo random prime, Schwartz-Zippel lemma

Deterministic Path Counting

Interpretation of algebraic data structures:

- Maintain $M_{i,j}[k]$: number of paths from i to j of length exactly k
- Entries might be as large as $\Theta(n^k)$
  $\rightarrow$ Field operation takes time $O(k \log n^k) = O(k^2 \log n)$
- Significant overhead!

Randomized approach:

- Actually interested in smallest k for which $A_{i,j}[k] \neq 0$
- Less time per operation with computation modulo random prime, Schwartz-Zippel lemma

Observation: For $k = O(1/\epsilon)$ we can live with overhead of $O(k^2 \log n) = \tilde{O}(1/\epsilon^2)$

Novel Algebraic Bounded-Distance Data Structure

Theorem

Given any $0 \leq \nu \leq \mu \leq 1$ *and any sets* $A, B \subseteq V$ *s.t.* $|A|, |B| \leq n^{\mu}$, *there is a randomized data structure for maintaining the* $A \times B$ *distances up to* $\leq h$ *under edge updates and* **set updates** *with update time* $\tilde{O}((n^{\omega(1,1,\mu)-\mu} + n^{\omega(1,\mu,\nu)-\nu} + n^{\mu+\nu} + |A| \cdot |B|) \cdot h^2)$.

Novel Algebraic Bounded-Distance Data Structure

Theorem

Given any $0 \leq \nu \leq \mu \leq 1$ *and any sets* $A, B \subseteq V$ *s.t.* $|A|, |B| \leq n^{\mu}$*, there is a randomized data structure for maintaining the* $A \times B$ *distances up to* $\leq h$ *under edge updates and* **set updates** *with update time* $\tilde{O}((n^{\omega(1,1,\mu)-\mu} + n^{\omega(1,\mu,\nu)-\nu} + n^{\mu+\nu} + |A| \cdot |B|) \cdot h^2)$*.*

Two regimes:

- $\tilde{O}((n^{1.407} + |A| \cdot |B|) \cdot h^2)$ for $|A|, |B| \leq n^{0.55}$
- $\tilde{O}((n^{1.529} + |A| \cdot |B|) \cdot h^2)$

Novel Algebraic Bounded-Distance Data Structure

Theorem

Given any $0 \leq \nu \leq \mu \leq 1$ and any sets $A, B \subseteq V$ s.t. $|A|, |B| \leq n^{\mu}$, there is a randomized data structure for maintaining the $A \times B$ distances up to $\leq h$ under edge updates and **set updates** *with update time $\tilde{O}((n^{\omega(1,1,\mu)-\mu} + n^{\omega(1,\mu,\nu)-\nu} + n^{\mu+\nu} + |A| \cdot |B|) \cdot h^2)$.*

Two regimes:

- $\tilde{O}((n^{1.407} + |A| \cdot |B|) \cdot h^2)$ for $|A|, |B| \leq n^{0.55}$
- $\tilde{O}((n^{1.529} + |A| \cdot |B|) \cdot h^2)$

Idea:

- (Vanilla) algebraic approach based on periodic recomputations

Novel Algebraic Bounded-Distance Data Structure

Theorem

Given any $0 \le \nu \le \mu \le 1$ and any sets $A, B \subseteq V$ s.t. $|A|, |B| \le n^{\mu}$, there is a randomized data structure for maintaining the $A \times B$ distances up to $\le h$ under edge updates and **set updates** *with update time $\tilde{O}((n^{\omega(1,1,\mu)-\mu} + n^{\omega(1,\mu,\nu)-\nu} + n^{\mu+\nu} + |A| \cdot |B|) \cdot h^2)$.*

Two regimes:

- $\tilde{O}((n^{1.407} + |A| \cdot |B|) \cdot h^2)$ for $|A|, |B| \le n^{0.55}$
- $\tilde{O}((n^{1.529} + |A| \cdot |B|) \cdot h^2)$

Idea:

- (Vanilla) algebraic approach based on periodic recomputations
  → Extension to set/row updates somewhat natural
  → Essential case: Sets A and B fixed in advance

Novel Algebraic Bounded-Distance Data Structure

Theorem

Given any $0 \le \nu \le \mu \le 1$ and any sets $A, B \subseteq V$ s.t. $|A|, |B| \le n^{\mu}$, there is a randomized data structure for maintaining the $A \times B$ distances up to $\le h$ under edge updates and **set updates** *with update time $\tilde{O}((n^{\omega(1,1,\mu)-\mu} + n^{\omega(1,\mu,\nu)-\nu} + n^{\mu+\nu} + |A| \cdot |B|) \cdot h^2)$.*

Two regimes:

- $\tilde{O}((n^{1.407} + |A| \cdot |B|) \cdot h^2)$ for $|A|, |B| \le n^{0.55}$
- $\tilde{O}((n^{1.529} + |A| \cdot |B|) \cdot h^2)$

Idea:

- (Vanilla) algebraic approach based on periodic recomputations
  → Extension to set/row updates somewhat natural
  → Essential case: Sets A and B fixed in advance
- We extend approach of [v.d. Brand, Nanongkai, Saranurak '19] to optimize for case of large query set

Challenges

- “Path-reporting” for algebraic approaches [Bergamaschi, Henzinger, Gutenberg, Vassilevska Williams, Wein ’21] [Karczmarz, Mukherjee, Sankowski ’22]
- Extend emulator-based approximation approach to weighted graphs

Challenges

- “Path-reporting” for algebraic approaches [Bergamaschi, Henzinger, Gutenberg, Vassilevska Williams, Wein ’21] [Karczmarz, Mukherjee, Sankowski ’22]
- Extend emulator-based approximation approach to weighted graphs
- More dynamic algorithms without caveats