Recent Results on Dynamic Distance Computation

Sebastian Forster, né Krinninger

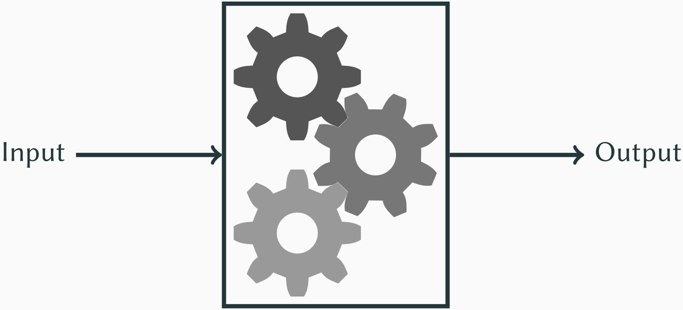
University of Salzburg

@ETH Zurich (June 26, 2023)

Joint works with Jan van den Brand, Michal Dory, Gramoz Goranci, Yasamin Nazari, Maximilian Probst Gutenberg, Antonis Skarlatos, and Tijn de Vos

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreements No 853109 and 947702). Funded by ONR BRC grant N00014-18-1-2562 and by the Simons Institute for the Theory of Computing through a Simons-Berkeley Postdoctoral Fellowship. Supported by the Austrian Science Fund (FWF): P 32863-N. The research leading to these results has received funding from the Swiss National Science Foundation (no. 200021-184735 and no. 200021-204787).

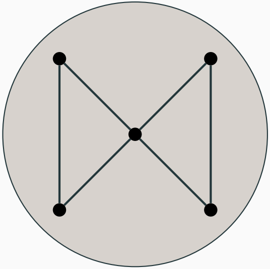
Static Approach



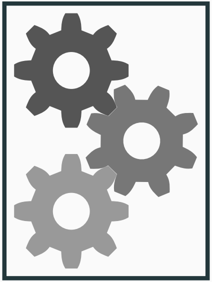
Dynamic Environments

Dynamic Distance Maintenance

Input graph G



Algorithm

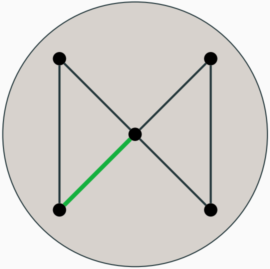


Distance Matrix

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

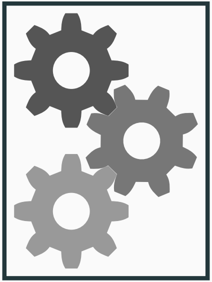
Dynamic Distance Maintenance

Input graph G



Adversary inserts and deletes edges

Algorithm

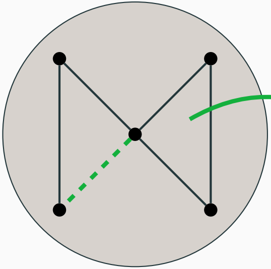


Distance Matrix

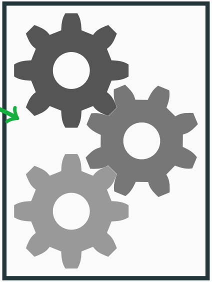
$$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 2 & 2 \\ 1 & 2 & 2 & 0 & 1 \\ 1 & 2 & 2 & 1 & 0 \end{pmatrix}$$

Dynamic Distance Maintenance

Input graph G



Algorithm



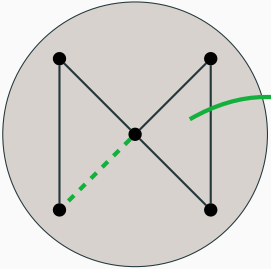
Distance Matrix

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

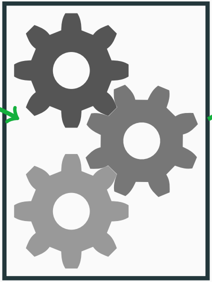
Adversary inserts
and deletes edges

Dynamic Distance Maintenance

Input graph G



Algorithm



Distance Matrix

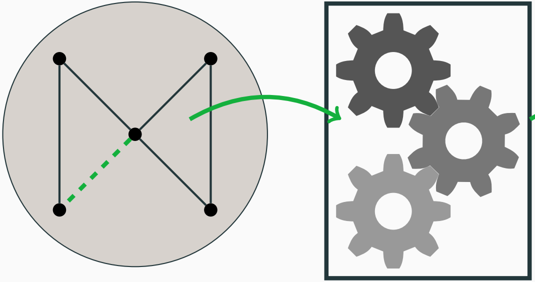
$$\begin{pmatrix} 0 & \mathbf{2} & 1 & 1 & 1 \\ \mathbf{2} & 0 & 1 & \mathbf{3} & \mathbf{3} \\ 1 & 1 & 0 & 2 & 2 \\ \mathbf{3} & 2 & 2 & 0 & 1 \\ \mathbf{3} & 2 & 2 & 1 & 0 \end{pmatrix}$$

Adversary inserts and deletes edges

Algorithm updates distance matrix

Dynamic Distance Maintenance

Input graph G — Algorithm — Distance Matrix



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

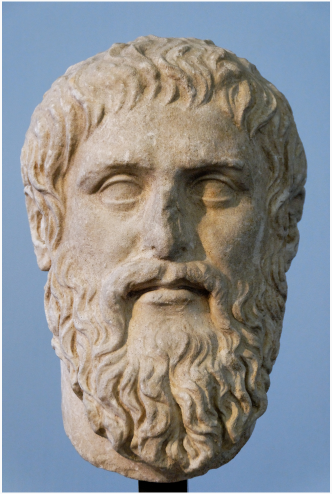
Adversary inserts and deletes edges

Algorithm updates distance matrix

State of the Art

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

1. Exact
2. $1 + \epsilon$
3. Small constant (2 to < 3)
4. Thorup-Zwick (3 to $O(\log n)$)
5. Above Logarithmic



Regime 1: Exact

Fully dynamic:

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

Regime 1: Exact

Fully dynamic:

- Amortized update time $\tilde{O}(n^2)$ [Demetrescu, Italiano '03]
- Worst-case update time $\tilde{O}(n^{2+2/3})$ [Abraham, Chechik, **K** '16]

Regime 1: Exact

Fully dynamic:

- Amortized update time $\tilde{O}(n^2)$ [Demetrescu, Italiano '03]
- Worst-case update time $\tilde{O}(n^{2+2/3})$ [Abraham, Chechik, **K** '16]
  Recent claim of $\tilde{O}(n^{2.5})$ [Mao arXiv '23]

Regime 1: Exact

Fully dynamic:

- Amortized update time $\tilde{O}(n^2)$ [Demetrescu, Italiano ’03]
- Worst-case update time $\tilde{O}(n^{2+2/3})$ [Abraham, Chechik, **K** ’16]
  Recent claim of $\tilde{O}(n^{2.5})$ [Mao arXiv ’23]
- Deterministic worst-case update time:
  - $\tilde{O}(n^{2+41/61})$ [Chechik, Zhang ’23]
  - $\tilde{O}(n^{2.6})$ unweighted [Probst Gutenberg, Wulff-Nilsen ’20]
- Trade-off: worst-case update time/query time $O(n^{1.724})$
  [Sankowsi ’05; v.d. Brand, Nanongkai, Saranurak ’19]

Regime 1: Exact

Fully dynamic:

- Amortized update time $\tilde{O}(n^2)$ [Demetrescu, Italiano '03]
- Worst-case update time $\tilde{O}(n^{2+2/3})$ [Abraham, Chechik, **K** '16]
  Recent claim of $\tilde{O}(n^{2.5})$ [Mao arXiv '23]
- Deterministic worst-case update time:
  - $\tilde{O}(n^{2+41/61})$ [Chechik, Zhang '23]
  - $\tilde{O}(n^{2.6})$ unweighted [Probst Gutenberg, Wulff-Nilsen '20]
- Trade-off: worst-case update time/query time $O(n^{1.724})$ [Sankowsi '05; v.d. Brand, Nanongkai, Saranurak '19]

Decremental:

- Total update time $\tilde{O}(n^3)$ in unweighted graphs [Demetrescu, Italiano '01; Baswana, Hariharan, Sen '02]
- Deterministic version: [Evald, Fredslund-Hansen, Probst Gutenberg, Wulff-Nilsen '21]

Regime 2: $1 + \epsilon$

Decremental: Pretty well explored

- Total upate time $\tilde{O}(mn)$ [Bernstein '13]
- Deterministic total update time $O(mn^{1+o(1)})$ in undirected graphs [Bernstein, Probst Gutenberg, Saranurak]

Regime 2: $1 + \epsilon$

Decremental: Pretty well explored

- Total upate time $\tilde{O}(mn)$ [Bernstein '13]
- Deterministic total update time $O(mn^{1+o(1)})$ in undirected graphs [Bernstein, Probst Gutenberg, Saranurak]

Fully dynamic: [v.d. Brand, Nanongkai '19]

- Update time $\tilde{O}(n^{2.045})$
- Update time $\tilde{O}(n^2)$ in unweighted, undirected graphs
- Trade-offs:
  - Update time $n^{1.863}$, query time $O(n^{0.666})$
  - Update time $n^{1.823}$, query time $O(n^{0.45})$ in unweighted, undirected graphs

Our Results

[v.d. Brand, **F**, Nazari '22]

- Deterministic $1 + \epsilon$ in unweighted, undirected graphs
  - k sources: worst-case update time $O(n^{1.529} + kn^{1+o(1)})$
  - single pair: worst-case update time $O(n^{1.407})$

Our Results

[v.d. Brand, **F**, Nazari '22]

- Deterministic $1+\epsilon$ in unweighted, undirected graphs
  - k sources: worst-case update time $O(n^{1.529} + kn^{1+o(1)})$
  - single pair: worst-case update time $O(n^{1.407})$
- Trade-offs:
  - $1+\epsilon$ with worst-case upate time $O(n^{1.788})$ and query time $O(n^{0.45})$ in unweighted, undirected graphs
  - **Exact** with worst-case upate time/query time $O(n^{1.704})$ in unweighted, directed graphs

Our Results

[v.d. Brand, **F**, Nazari '22]

- Deterministic $1+\epsilon$ in unweighted, undirected graphs
  - k sources: worst-case update time $O(n^{1.529}+kn^{1+o(1)})$
  - single pair: worst-case update time $O(n^{1.407})$
- Trade-offs:
  - $1+\epsilon$ with worst-case upate time $O(n^{1.788})$ and query time $O(n^{0.45})$ in unweighted, undirected graphs
  - **Exact** with worst-case upate time/query time $O(n^{1.704})$ in unweighted, directed graphs

Remarks:

- Randomized $O(n^{1.529}+kn^{1+o(1)})$ by [Bergamaschi et al. '21]
- $n^{1.529}$ and $n^{1.407}$ match CLBs [v.d. Brand, Nanongkai, Saranurak '19]

Technique

Two Worlds

- **Combinatorial** emulator with multiplicative stretch $1 + \epsilon$ and additive stretch 4
- **Algebraic** data structure for maintaining small distances

Two Worlds

- **Combinatorial** emulator with multiplicative stretch $1+\epsilon$ and additive stretch 4
- **Algebraic** data structure for maintaining small distances

Three ideas:

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

Technique

Two Worlds

- **Combinatorial** emulator with multiplicative stretch $1 + \epsilon$ and additive stretch 4
- **Algebraic** data structure for maintaining small distances

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

Technique

Two Worlds

- **Combinatorial** emulator with multiplicative stretch $1 + \epsilon$ and additive stretch 4
- **Algebraic** data structure for maintaining small distances

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

Regime 3: Small constant (2 to < 3)

From now on: all-pairs distances, undirected graphs, amortized update times

This regime is not very well explored!

Regime 3: Small constant (2 to < 3)

From now on: all-pairs distances, undirected graphs, amortized update times

This regime is not very well explored!

Prior work:

- Fully dynamic: $2 + \epsilon$ with update time $m^{1+o(1)}$ [Bernstein '09]
- Decremental: 2 with total update time $n^{2.5}$ [Henzinger, **K**, Nanongkai '13]

Our Result

[Dory, **F**, Nazari, de Vos arXiv '22]

- Decremental $2+\epsilon$-approximation in weighted graphs
- Total update time $\tilde{O}(m^{1/2}n^{3/2+o(1)})$ if $m \leq n^{5/3}$
- Total update time $\tilde{O}(mn^{2/3})$ if $m \geq n^{5/3}$
- Constant query time



Technique

Main ideas:

- Setup based on bunches and clusters (similar to TZ distance oracle)
- Simplify static algorithm at cost of higher running time
- Exploit monotonicity

Technique

Main ideas:

- Setup based on bunches and clusters (similar to TZ distance oracle)
- Simplify static algorithm at cost of higher running time
- Exploit monotonicity
- Exploit monotonicity

Technique

Main ideas:

- Setup based on bunches and clusters (similar to TZ distance oracle)
- Simplify static algorithm at cost of higher running time
- Exploit monotonicity
- Exploit monotonicity
- Powerful $(1 + \epsilon)$-approximate SSSP primitive [Henzinger, **K**, Nanongkai '14]

Technique

Main ideas:

- Setup based on bunches and clusters (similar to TZ distance oracle)
- Simplify static algorithm at cost of higher running time
- Exploit monotonicity
- Exploit monotonicity
- Powerful $(1+\epsilon)$-approximate SSSP primitive [Henzinger, **K**, Nanongkai '14]

My impression

Static algorithms are still too "messy". A candidate for SOSA?

Regime 4: Thorup-Zwick style (3 to $O(\log n)$)

Partially dynamic:

- Decremental with stretch $(2k-1)(1+\epsilon)$, update time $\tilde{O}((m+n^{1+o(1)})n^{1/k})$, and query time $\min(O(\log\log(nW), k)$ [Chechik '18; Łącki, Nazari '22]
- Incremental with stretch $(2k-1)^t$ and worst case update/query time $\tilde{O}(m^{1/(t+1)}n^{t/k})$ [Chen, Goranci, Monika Henzinger, Peng, Saranurak '20]

Regime 4: Thorup-Zwick style (3 to $O(\log n)$)

Partially dynamic:

- Decremental with stretch $(2k-1)(1+\epsilon)$, update time $\tilde{O}((m + n^{1+o(1)})n^{1/k})$, and query time $\min(O(\log\log(nW), k)$ [Chechik '18; Łącki, Nazari '22]
- Incremental with stretch $(2k-1)^t$ and worst case update/query time $\tilde{O}(m^{1/(t+1)}n^{t/k})$ [Chen, Goranci, Monika Henzinger, Peng, Saranurak '20]

Fully dynamic:

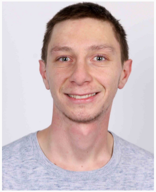
- Stretch $2^{O(\rho k)}$, update time $\tilde{O}(m^{1/2}n^{1/k})$, and query time $O(k^2\rho^2)$ ($\rho = 1 + \lceil \log n^{1-1/k} / \log(m/n^{1-1/k})\rceil$) [Abraham, Chechik, Talwar '14]
- Stretch $\tilde{O}(\log n)$, update/query time $O(m^{2/3+o(1)})$ [Chen, Goranci, Monika Henzinger, Peng, Saranurak '20]

Our Result

[**F**, Goranci, Nazari, Skarlatos '23]

Fully dynamic distance oracle, for any constant $0 < \rho < 1$:

- Stretch $(256/\rho^2)^{4/\rho}$, update time n^ρ, query time $n^{\rho/8}$ or
- Stretch $O(\log\log n)$, update time n^ρ, query time $n^{o(1)}$



Main Idea

Very general reduction

- **Decremental** algorithm maintaining hub set $H(v)$ for every node v such that for any pair s, t:

$$\delta(s,t) = \min_{v \in H(s) \cap H(t)} \delta_H(s,v) + \delta_H(s,t)$$

- can be extended to **fully dynamic** distance oracle

Thorup-Zwick distance oracle as hub labeling scheme

- Stretch $2k-1$
- Hub set of size $O(n^{1/k})$
- *Additional* query mechanism for query time $O(k)$ instead of $O(n^{1/k})$

Independent result [Chuzhoy, Zhang '23]: stretch $(\log\log n)^{2^{1/\rho^3}}$, update time $\tilde{O}(n^{O(\rho)})$, and query time $\tilde{O}(2^{O(1/\rho)})$ for any $\frac{2}{(\log n)^{1/200}} < \rho < \frac{1}{400}$

Regime 5: Above Logarithmic

Motivation:

- Stretch goes into running time of another algorithm
- Main interest: Very fast update time, deterministic

Regime 5: Above Logarithmic

Motivation:

- Stretch goes into running time of another algorithm
- Main interest: Very fast update time, deterministic

Decremental: [Chuzhoy '21; Bernstein, Probst Gutenberg, Saranurak '21]

- Polylogarithmic stretch, total update time $m^{1+\rho}$ for any constant ρ
- Stretch $n^{o(1)}$, total update time $m^{1+o(1)}$

Regime 5: Above Logarithmic

Motivation:

- Stretch goes into running time of another algorithm
- Main interest: Very fast update time, deterministic

Decremental: [Chuzhoy '21; Bernstein, Probst Gutenberg, Saranurak '21]

- Polylogarithmic stretch, total update time $m^{1+\rho}$ for any constant ρ
- Stretch $n^{o(1)}$, total update time $m^{1+o(1)}$

Fully dynamic:

- Stretch $O(\log n)^{3k-2}$, update time time $m^{1/k+o(1)} \cdot O(\log n)^{4k-2}$, and query time $O(k(\log n)^2)$ [Forster, Goranci, Henzinger '21]

Our Result

[**F**, Nazari, Probst Gutenberg '23]

- **Incremental**
- Polylogarithmic stretch
- Total update time $\tilde{O}(m)$
- Query time $O(\log \log n)$

Technique

Notable:

- No Even-Shiloach tree
- No expander decomposition
- No Thorup-Zwick based construction

Technique

Notable:

- No Even-Shiloach tree
- No expander decomposition
- No Thorup-Zwick based construction

Overall setup:

- Hierarchy of $k = \Theta(\log \log n)$ sparsifiers: $G = H_1, H_2, \ldots, H_k$ following [Andoni, Stein, Zhong '20]
- H_i is an α-approximation of H_{i-1} for some constant α
  $\alpha^k = \operatorname{polylog} n$

Technique

Notable:

- No Even-Shiloach tree
- No expander decomposition
- No Thorup-Zwick based construction

Overall setup:

- Hierarchy of $k = \Theta(\log \log n)$ sparsifiers: $G = H_1, H_2, \ldots, H_k$ following [Andoni, Stein, Zhong '20]
- H_i is an α-approximation of H_{i-1} for some constant α
  $\alpha^k = \operatorname{polylog} n$

Challenge:

- Naively, number of edges added to H_{i+1}: $O(\ldots + |E(H_i)| \log n)$
- For $k = \Theta(\log \log n)$ levels: $m \cdot O(\log n)^{\log \log n}$ insertions (at best)

Open Problems

I would like to see the following:

1. Conditional lower bound of $n^{2.5}$ for worst-case update time in exact APSP
2. Stretch $(2k-1)$, (worst case) update time $O(n^{1/k})$, query time $O(k)$
3. Deterministic decremental with polylogarithmic stretch, polylogarithmic query time and update time $\tilde{O}(m)$