Single-Source Shortest Paths and Strong Connectivity in Dynamic Planar Graphs

Panagiotis Charalampopoulos\textsuperscript{1,2}  Adam Karczmarz\textsuperscript{2}

\textsuperscript{1}King’s College London, UK
\textsuperscript{2}University of Warsaw, Poland

ESA 2020
Dynamic Shortest Paths

Problem

Maintain a data structure over a graph $G$ that supports:
- updates to the graph (e.g. edge insertions/deletions),
- shortest paths/distance queries.

Unless explicitly stated otherwise, we consider directed weighted graphs.
Problem: Dynamic Distance Oracle

Maintain a data structure over a graph $G$ that supports:

- updates to the graph (e.g. edge insertions/deletions),
- shortest paths/distance queries.
Problem: Dynamic Distance Oracle

Maintain a data structure over a graph $G$ that supports:
- updates to the graph (e.g. edge insertions/deletions),
- shortest paths/distance queries.

All-pairs: The source and target are specified by the query.
Problem: Dynamic Distance Oracle

Maintain a data structure over a graph $G$ that supports:
- updates to the graph (e.g. edge insertions/deletions),
- shortest paths/distance queries.

**All-pairs**: The source and target are specified by the query.

**Single-source**: The source is fixed and the target is specified by the query.
Dynamic Shortest Paths

Problem: Dynamic Distance Oracle

Maintain a data structure over a graph $G$ that supports:
- updates to the graph (e.g. edge insertions/deletions),
- shortest paths/distance queries.

**All-pairs**: The source and target are specified by the query.

**Single-source**: The source is fixed and the target is specified by the query.

Unless explicitly stated otherwise, we consider directed weighted graphs.
Related Work I: General Graphs

All-pairs
[Johnson; J. ACM 1977, Demetrescu-Italiano; J. ACM 2004]
The all-pairs distance matrix can be computed in \(\tilde{O}(nm) = \tilde{O}(n^3)\) time and maintained in \(\tilde{O}(n^2)\) amortized time.

Single-source
[Roditty-Zwick; Algorithmica 2011]
\(O(n^3 - \epsilon)\) initialization, \(O(n^2 - \epsilon)\) amortized update, and \(O(n^{1-\epsilon})\) query times impossible conditional on the APSP conjecture.

For sparse graphs (i.e. \(m = O(n)\)), nothing better than recomputing from scratch is known for either of the variants.

Workarounds:
settle for approximate answers, study more structured graph classes.
### All-pairs [Johnson; J. ACM 1977, Demetrescu-Italiano; J. ACM 2004]

The all-pairs distance matrix can be computed in $\tilde{O}(nm) = \tilde{O}(n^3)$ time and maintained in $\tilde{O}(n^2)$ amortized time.
All-pairs [Johnson; J. ACM 1977, Demetrescu-Italiano; J. ACM 2004]

The all-pairs distance matrix can be computed in $\tilde{O}(nm) = \tilde{O}(n^3)$ time and maintained in $\tilde{O}(n^2)$ amortized time.

Single-source [Roditty-Zwick; Algorithmica 2011]

...
Related Work I: General Graphs

**All-pairs [Johnson; J. ACM 1977, Demetrescu-Italiano; J. ACM 2004]**

The all-pairs distance matrix can be computed in $\tilde{O}(nm) = \tilde{O}(n^3)$ time and maintained in $\tilde{O}(n^2)$ amortized time.

**Single-source [Roditty-Zwick; Algorithmica 2011]**

$O(n^3-\varepsilon)$ initialization, $O(n^2-\varepsilon)$ amortized update, and $O(n^{1-\varepsilon})$ query times impossible conditional on the APSP conjecture.
All-pairs [Johnson; J. ACM 1977, Demetrescu-Italiano; J. ACM 2004]

The all-pairs distance matrix can be computed in $\tilde{O}(nm) = \tilde{O}(n^3)$ time and maintained in $\tilde{O}(n^2)$ amortized time.

Single-source [Roditty-Zwick; Algorithmica 2011]

$O(n^{3-\epsilon})$ initialization, $O(n^{2-\epsilon})$ amortized update, and $O(n^{1-\epsilon})$ query times impossible conditional on the APSP conjecture.

For sparse graphs (i.e. $m = O(n)$), nothing better than recomputing from scratch is known for either of the variants.
Related Work I: General Graphs

**All-pairs** [Johnson; J. ACM 1977, Demetrescu-Italiano; J. ACM 2004]

The all-pairs distance matrix can be computed in $\tilde{O}(nm) = \tilde{O}(n^3)$ time and maintained in $\tilde{O}(n^2)$ amortized time.

**Single-source** [Roditty-Zwick; Algorithmica 2011]

$O(n^{3-\epsilon})$ initialization, $O(n^{2-\epsilon})$ amortized update, and $O(n^{1-\epsilon})$ query times impossible conditional on the APSP conjecture.

For sparse graphs (i.e. $m = O(n)$), nothing better than recomputing from scratch is known for either of the variants.

**Workarounds:**
Related Work I: General Graphs

**All-pairs** [Johnson; J. ACM 1977, Demetrescu-Italiano; J. ACM 2004]

The all-pairs distance matrix can be computed in $\tilde{O}(nm) = \tilde{O}(n^3)$ time and maintained in $\tilde{O}(n^2)$ amortized time.

**Single-source** [Roditty-Zwick; Algorithmica 2011]

$O(n^3-\epsilon)$ initialization, $O(n^2-\epsilon)$ amortized update, and $O(n^{1-\epsilon})$ query times impossible conditional on the APSP conjecture.

For sparse graphs (i.e. $m = O(n)$), nothing better than recomputing from scratch is known for either of the variants.

**Workarounds:** settle for approximate answers
Related Work I: General Graphs

**All-pairs [Johnson; J. ACM 1977, Demetrescu-Italiano; J. ACM 2004]**

The all-pairs distance matrix can be computed in $\tilde{\mathcal{O}}(nm) = \tilde{\mathcal{O}}(n^3)$ time and maintained in $\tilde{\mathcal{O}}(n^2)$ amortized time.

**Single-source [Roditty-Zwick; Algorithmica 2011]**

$\mathcal{O}(n^{3-\epsilon})$ initialization, $\mathcal{O}(n^{2-\epsilon})$ amortized update, and $\mathcal{O}(n^{1-\epsilon})$ query times impossible conditional on the APSP conjecture.

For sparse graphs (i.e. $m = \mathcal{O}(n)$), nothing better than recomputing from scratch is known for either of the variants.

**Workarounds:** settle for approximate answers, study more structured graph classes.
Related Work II: Planar Graphs

All-pairs Exact: $\tilde{O}(n^{2/3})$ update and query time.

[Abboud-Dahlgaard; FOCS 2016] $(1+\epsilon)$-approx: $\tilde{O}(n^{1/2})$ update and query time for undirected graphs.

[Abraham et al.; STOC 2012] $(1+\epsilon)$-approx: $\tilde{O}(n^{1/2})$ update and $O(1)$ query time for decremental graphs.

Single-source Exact: $\tilde{O}(n^{2/3})$ update time and $\tilde{O}(n^{1/3})$ query time. (Using [Klein; SODA 2005].)

Lower bound: $t_u \times t_q$ cannot be $O(n^{1-\epsilon})$ conditional on the APSP conjecture.

Related Work II: Planar Graphs

All-pairs

- **Exact:** $\tilde{O}\left(\frac{n^2}{3}\right)$ update and query time.\\[Fakcharoenphol-Rao; JCSS 2006, Klein; SODA 2005\\]
- **Lower bound:** $t_u \times t_q$ cannot be $O\left(\frac{n^{1-\epsilon}}{3}\right)$ conditional on the APSP conjecture.\\[Abboud-Dahlgaard; FOCS 2016\\]

- **(1+ε)-approx:** $\tilde{O}\left(\frac{n^{1/2}}{2}\right)$ update and query time for undirected graphs.\\[Abraham et al.; STOC 2012\\]

- **Single-source**
  - **Exact:** $\tilde{O}\left(\frac{n^2}{3}\right)$ update time and $\tilde{O}\left(\frac{n^{1/3}}{}\right)$ query time. (Using [Klein; SODA 2005].)
  - **(1+ε)-approx:** $\tilde{O}\left(\frac{n^{1/2}}{2}\right)$ update time and $O\left(\frac{1}{\epsilon}\right)$ query time for decremental graphs.\\[Karczmarz; SODA 2018\\]
All-pairs

- **Exact**: $\tilde{O}(n^{2/3})$ update and query time.

  [Fakcharoenphol-Rao; JCSS 2006, Klein; SODA 2005]
# Related Work II: Planar Graphs

## All-pairs

- **Exact**: $\tilde{O}(n^{2/3})$ update and query time.
  - [Fakcharoenphol-Rao; JCSS 2006, Klein; SODA 2005]

- **Lower bound**: $t_u \times t_q$ cannot be $\mathcal{O}(n^{1-\epsilon})$ conditional on the APSP conjecture.
  - [Abboud-Dahlgaard; FOCS 2016]
All-pairs

- **Exact:** $\tilde{O}(n^{2/3})$ update and query time.
  [Fakcharoenphol-Rao; JCSS 2006, Klein; SODA 2005]
- **Lower bound:** $t_u \times t_q$ cannot be $O(n^{1-\epsilon})$ conditional on the APSP conjecture. [Abboud-Dahlgaard; FOCS 2016]
- **$(1 + \epsilon)$-approx:** $\tilde{O}(n^{1/2})$ update and query time for undirected graphs. [Abraham et al.; STOC 2012]
### All-pairs

- **Exact**: $\tilde{O}(n^{2/3})$ update and query time. 
  [Fakcharoenphol-Rao; JCSS 2006, Klein; SODA 2005]
- **Lower bound**: $t_u \times t_q$ cannot be $O(n^{1-\epsilon})$ conditional on the APSP conjecture. [Abboud-Dahlgaard; FOCS 2016]
- **(1 + $\epsilon$)-approx**: $\tilde{O}(n^{1/2})$ update and query time for undirected graphs. [Abraham et al.; STOC 2012]

### Single-source

- **Exact**: $\tilde{O}(n^{2/3})$ update and query time. 
  *(Using [Klein; SODA 2005].)*
- **(1 + $\epsilon$)-approx**: $\tilde{O}(n^{1/2})$ update time and $O(1)$ query time for decremental graphs. [Karczmarz; SODA 2018]

---

P. Charalampopoulos and A. Karczmarz

Single-Source Shortest Paths and Strong Connectivity in Dynamic Planar Graphs
## Related Work II: Planar Graphs

### All-pairs

- **Exact:** $\tilde{O}(n^{2/3})$ update and query time.  
  [Fakcharoenphol-Rao; JCSS 2006, Klein; SODA 2005]

- **Lower bound:** $t_u \times t_q$ cannot be $O(n^{1-\epsilon})$ conditional on the APSP conjecture. [Abboud-Dahlgaard; FOCS 2016]

- **$(1 + \epsilon)$-approx:** $\tilde{O}(n^{1/2})$ update and query time for undirected graphs. [Abraham et al.; STOC 2012]

### Single-source

- **Exact:** $\tilde{O}(n^{2/3})$ update time and $\tilde{O}(n^{1/3})$ query time.  
  (Using [Klein; SODA 2005].)
## Related Work II: Planar Graphs

### All-pairs

- **Exact**: $\tilde{O}(n^{2/3})$ update and query time.
  - [Fakcharoenphol-Rao; JCSS 2006, Klein; SODA 2005]
- **Lower bound**: $t_u \times t_q$ cannot be $\mathcal{O}(n^{1-\epsilon})$ conditional on the APSP conjecture. [Abboud-Dahlgaard; FOCS 2016]
- **$(1 + \epsilon)$-approx**: $\tilde{O}(n^{1/2})$ update and query time for undirected graphs. [Abraham et al.; STOC 2012]

### Single-source

- **Exact**: $\tilde{O}(n^{2/3})$ update time and $\tilde{O}(n^{1/3})$ query time.
  (Using [Klein; SODA 2005].)
- **$(1 + \epsilon)$-approx**: $\tilde{O}(n^{1/2})$ update time and $\mathcal{O}(1)$ query time for decremental graphs. [Karczmarz; SODA 2018]
Our Results

Theorem

We can maintain an $n$-vertex planar graph $G$ under:
- edge insertions,
- edge deletions, and
- changes of the sources in $\tilde{O}(n^{4/5})$ worst-case time per update so that $dist_G(s, v)$ for any $v \in V(G)$ can be computed in $O(\log^2 n)$ time.

The first fully dynamic single-source shortest paths algorithm for planar graphs breaking through the $\tilde{O}(n)$ update-query time product barrier, even in the approximate setting.

Our approach, combined with a few more ingredients, also yields a fully dynamic strong connectivity data structure with the same complexities.
Our Results

Theorem

We can maintain an \( n \)-vertex planar graph \( G \) under:

- edge insertions,
- edge deletions, and
- changes of the source \( s \)

in \( \tilde{O}(n^{4/5}) \) worst-case time per update so that \( \text{dist}_G(s,v) \) for any \( v \in V(G) \) can be computed in \( O(\log^2 n) \) time.
Our Results

Theorem

We can maintain an \( n \)-vertex planar graph \( G \) under:

- edge insertions,
- edge deletions, and
- changes of the source \( s \)

in \( \tilde{O}(n^{4/5}) \) worst-case time per update so that \( \text{dist}_G(s, v) \) for any \( v \in V(G) \) can be computed in \( O(\log^2 n) \) time.

The first fully dynamic single-source shortest paths algorithm for planar graphs breaking through the \( \tilde{O}(n) \) update-query time product barrier, even in the approximate setting.
Our Results

Theorem

We can maintain an $n$-vertex planar graph $G$ under:
- edge insertions,
- edge deletions, and
- changes of the source $s$

in $\tilde{O}(n^{4/5})$ worst-case time per update so that $\text{dist}_{G}(s, v)$ for any $v \in V(G)$ can be computed in $O(\log^2 n)$ time.

The first fully dynamic single-source shortest paths algorithm for planar graphs breaking through the $\tilde{O}(n)$ update-query time product barrier, even in the approximate setting.

Our approach, combined with a few more ingredients, also yields a fully dynamic strong connectivity data structure with the same complexities.

P. Charalampopoulos and A. Karczmarz
Single-Source Shortest Paths and Strong Connectivity in Dynamic Planar Graphs
Miller [JCSS’86]

There always exists a Jordan curve separator of size $O(\sqrt{n})$ such that there are at most $\frac{2}{3}n$ vertices on its inside/outside.
There always exists a Jordan curve separator of size $O(\sqrt{n})$ such that there are at most $\frac{2n}{3}$ vertices on its inside/outside.
For \( r \in [1, n] \), a decomposition of the graph into:

- \( \mathcal{O}(n/r) \) pieces;
- each piece has \( \mathcal{O}(r) \) vertices;
- each piece has \( \mathcal{O}(\sqrt{r}) \) boundary vertices (vertices incident to edges in other pieces).
For $r \in [1, n]$, a decomposition of the graph into:

- $O(n/r)$ pieces;
- each piece has $O(r)$ vertices;
- each piece has $O(\sqrt{r})$ boundary vertices (vertices incident to edges in other pieces).

We denote the boundary of a piece $P$ by $\partial P$ and assume that all such vertices lie on a single face of $P$. 

[Klein & Subramanian; WADS 1993]
For $r \in [1, n]$, a decomposition of the graph into:

- $\mathcal{O}(n/r)$ pieces;
- each piece has $\mathcal{O}(r)$ vertices;
- each piece has $\mathcal{O}(\sqrt{r})$ boundary vertices (vertices incident to edges in other pieces).

We denote the boundary of a piece $P$ by $\partial P$ and assume that all such vertices lie on a single face of $P$.

An $r$-division can be maintained in $\mathcal{O}(r)$ worst-case time per update with $\mathcal{O}(1)$ pieces changing. [Klein & Subramanian; WADS 1993]
In nearly-linear time (in the size of the graph), we can construct a data structure that can report in logarithmic time the distance between any vertex \( u \) on the infinite face and any other vertex \( v \) of the graph.
In nearly-linear time (in the size of the graph), we can construct a data structure that can report in logarithmic time the distance between any vertex \( u \) on the infinite face and any other vertex \( v \) of the graph.
In *nearly-linear* time (in the size of the graph), we can construct a data structure that can report in *logarithmic* time the distance between any vertex \( u \) on the infinite face and any other vertex \( v \) of the graph.
Dense Distance Graph (DDG)

The distance matrix capturing pairwise distances between vertices of a set $\partial H$ of vertices lying on a single face of a plane graph $H$ can be computed in $\tilde{O}(|H| + |\partial H|^2)$ time using MSSP.
We can compute single-source shortest paths in a collection of DDGs with $N$ vertices in total (with multiplicities) in $\tilde{O}(N)$ time.
We can compute single-source shortest paths in a collection of DDGs with $N$ vertices in total (with multiplicities) in $\tilde{O}(N)$ time. $\tilde{O}(n/\sqrt{r})$ time for the DDGs of the pieces of an $r$-division.
Warm-up: $\tilde{O}(n^{2/3})$ update time, $\tilde{O}(n^{1/3})$ query time

Update:
Maintain an $r$-division. $O(r)$
Maintain an MSSP data structure and the DDG for each piece. $\tilde{O}(r)$
Run FR-Dijkstra from $s$ in the union of DDGs. $\tilde{O}(n^{2/3})$

Query:
Retrieve a piece $P$ containing $v$. $O(1)$
Compute $\min \{ \text{dist}_G(s, u) + \text{dist}_P(u, v) : u \in \partial P \}$. $O(\sqrt{r} \log r)$
Warm-up: $\tilde{O}(n^{2/3})$ update time, $\tilde{O}(n^{1/3})$ query time

Update:

---

P. Charalampopoulos and A. Karczmarz

Single-Source Shortest Paths and Strong Connectivity in Dynamic Planar Graphs
Warm-up: $\tilde{O}(n^{2/3})$ update time, $\tilde{O}(n^{1/3})$ query time

Update:
- Maintain an $r$-division. $O(r)$
Warm-up: $\tilde{O}(n^{2/3})$ update time, $\tilde{O}(n^{1/3})$ query time

**Update:**

- Maintain an $r$-division. $O(r)$
- Maintain an MSSP data structure and the DDG for each piece. $\tilde{O}(r)$
Update:

- Maintain an \( r \)-division. \( \mathcal{O}(r) \)
- Maintain an MSSP data structure and the DDG for each piece. \( \tilde{\mathcal{O}}(r) \)
- Run FR-Dijkstra from \( s \) in the union of DDGs. \( \tilde{\mathcal{O}}(n/\sqrt{r}) \)
Warm-up: $\tilde{O}(n^{2/3})$ update time, $\tilde{O}(n^{1/3})$ query time

Update:
- Maintain an $r$-division. $O(r)$
- Maintain an MSSP data structure and the DDG for each piece. $\tilde{O}(r)$
- Run FR-Dijkstra from $s$ in the union of DDGs. $\tilde{O}(n/\sqrt{r})$

Query:
Warm-up: $\tilde{O}(n^{2/3})$ update time, $\tilde{O}(n^{1/3})$ query time

Update:
- Maintain an $r$-division. $O(r)$
- Maintain an MSSP data structure and the DDG for each piece. $\tilde{O}(r)$
- Run FR-Dijkstra from $s$ in the union of DDGs. $\tilde{O}(n/\sqrt{r})$

Query:
- Retrieve a piece $P$ containing $v$. $O(1)$
Warm-up: $\tilde{O}(n^{2/3})$ update time, $\tilde{O}(n^{1/3})$ query time

Update:
- Maintain an $r$-division. $O(r)$
- Maintain an MSSP data structure and the DDG for each piece. $\tilde{O}(r)$
- Run FR-Dijkstra from $s$ in the union of DDGs. $\tilde{O}(n/\sqrt{r})$

Query:
- Retrieve a piece $P$ containing $v$. $O(1)$
- Compute $\min\{\text{dist}_G(s, u) + \text{dist}_P(u, v) : u \in \partial P\}$. $O(\sqrt{r} \log r)$
Warm-up: $\tilde{O}(n^{2/3})$ update time, $\tilde{O}(n^{1/3})$ query time

Update:
- Maintain an $r$-division. $O(r)$
- Maintain an MSSP data structure and the DDG for each piece. $\tilde{O}(r)$
- Run FR-Dijkstra from $s$ in the union of DDGs. $\tilde{O}(n/\sqrt{r})$

Query:
- Retrieve a piece $P$ containing $v$. $O(1)$
- Compute $\min\{\text{dist}_G(s, u) + \text{dist}_P(u, v) : u \in \partial P\}$. $O(\sqrt{r} \log r)$
Warm-up: $\tilde{O}(n^{2/3})$ update time, $\tilde{O}(n^{1/3})$ query time

**Update:**
- Maintain an $r$-division. $O(r)$
- Maintain an MSSP data structure and the DDG for each piece. $\tilde{O}(r)$
- Run FR-Dijkstra from $s$ in the union of DDGs. $\tilde{O}(n/\sqrt{r})$

**Query:**
- Retrieve a piece $P$ containing $v$. $O(1)$
- Compute $\min\{\text{dist}_G(s, u) + \text{dist}_P(u, v) : u \in \partial P\}$. $\tilde{O}(\sqrt{r}\log r)$
Warm-up: $\tilde{O}(n^{2/3})$ update time, $\tilde{O}(n^{1/3})$ query time

**Update:**
- Maintain an $r$-division. $\mathcal{O}(r)$
- Maintain an MSSP data structure and the DDG for each piece. $\tilde{O}(r)$
- Run FR-Dijkstra from $s$ in the union of DDGs. $\tilde{O}(n/\sqrt{r})$

**Query:**
- Retrieve a piece $P$ containing $v$. $\mathcal{O}(1)$
- Compute $\min\{\text{dist}_G(s, u) + \text{dist}_P(u, v) : u \in \partial P\}$. $\mathcal{O}(\sqrt{r} \log r)$
Update:
- Maintain an $r$-division. $\mathcal{O}(r)$
- Maintain an MSSP data structure and the DDG for each piece. $\tilde{\mathcal{O}}(r)$
- Run FR-Dijkstra from $s$ in the union of DDGs. $\tilde{\mathcal{O}}(n/\sqrt{r})$

Query:
- Retrieve a piece $P$ containing $v$. $\mathcal{O}(1)$
- Compute $\min\{\text{dist}_G(s, u) + \text{dist}_P(u, v) : u \in \partial P\}$. $\mathcal{O}(\sqrt{r} \log r)$

Balance for update time: $r = n^{2/3}$. 

P. Charalampopoulos and A. Karczmarz  
Single-Source Shortest Paths and Strong Connectivity in Dynamic Planar Graphs
Instead of trying all possible $\mathcal{O}(\sqrt{r})$ candidate boundary vertices, we want to compute the last boundary vertex $u$ visited by the shortest path in $\tilde{\mathcal{O}}(1)$ time.
The $s \rightarrow \partial P$ distances define a set of additive weights for $\partial P$. 

![Diagram](image-url)
The $s \rightarrow \partial P$ distances define a set of additive weights for $\partial P$.

Point Location via Voronoi Diagrams

Given a set of additive weights for $\partial P$, there exists an $\tilde{O}(\sqrt{r})$-sized data structure that given access to an MSSP data structure for $P$ with sources $\partial P$ answers point location queries in $O(\log^2 n)$ time.

[Gawrychowski et al.; SODA’18]
Point Location

- The $s \rightarrow \partial P$ distances define a set of additive weights for $\partial P$.

Point Location via Voronoi Diagrams

Given a set of additive weights for $\partial P$, there exists an $\tilde{O}(\sqrt{r})$-sized data structure that given access to an MSSP data structure for $P$ with sources $\partial P$ answers point location queries in $O(\log^2 n)$ time. It can be constructed in $\tilde{O}(r^{3/4})$ time.

[Charalampopoulos et al.; SODA’18, Charalampopoulos et al.; STOC 2019]
The Voronoi cell of each site consists of all vertices closer to it with respect to the additive distances.
A point location query returns the Voronoi cell containing a queried vertex $v$. 
Because all sites are adjacent to one face, the diagram can be described by a tree on $\mathcal{O}(|\partial P|) = \mathcal{O}(\sqrt{r})$ vertices.
\( \tilde{O}(n^{4/5}) \) update time, \( O(\log^2 n) \) query time

**Update:**
- Maintain an \( r \)-division. \( O(r) \) time.
- Maintain an MSSP data structure and the DDG for each piece. \( \tilde{O}(r) \)
- Run FR-Dijkstra from \( s \) in the union of DDGs. \( \tilde{O}(n/\sqrt{r}) \)

**Query:**
- Retrieve a piece \( P \) containing \( v \). \( O(1) \) time.
- Compute \( \min\{\text{dist}_G(s, u) + \text{dist}_P(u, v) : u \in \partial P\} \).
  \( O(\sqrt{r} \log r) \)
\( \tilde{O}(n^{4/5}) \) update time, \( \mathcal{O}(\log^2 n) \) query time

**Update:**
- Maintain an \( r \)-division. \( \mathcal{O}(r) \) time.
- Maintain an MSSP data structure and the DDG for each piece. \( \tilde{O}(r) \)
- Run FR-Dijkstra from \( s \) in the union of DDGs. \( \tilde{O}(n/\sqrt{r}) \)
- **Construct a point location data structure for each piece \( P \) with \( s \to \partial P \) distances as additive weights.** \( \tilde{O}(n/r \cdot r^{3/4}) = \tilde{O}(n/r^{1/4}) \)

**Query:**
- Retrieve a piece \( P \) containing \( v \). \( \mathcal{O}(1) \) time.
- Compute \( \min \{ \text{dist}_G(s, u) + \text{dist}_P(u, v) : u \in \partial P \} \). \( \mathcal{O}(\sqrt{r} \log r) \)
\(\tilde{O}(n^{4/5})\) update time, \(O(\log^2 n)\) query time

**Update:**
- Maintain an \(r\)-division. \(O(r)\) time.
- Maintain an MSSP data structure and the DDG for each piece. \(\tilde{O}(r)\)
- Run FR-Dijkstra from \(s\) in the union of DDGs. \(\tilde{O}(n/\sqrt{r})\)
- *Construct a point location data structure for each piece \(P\) with \(s \rightarrow \partial P\) distances as additive weights.*
  \(\tilde{O}(n/r \cdot r^{3/4}) = \tilde{O}(n/r^{1/4})\)

**Query:**
- Retrieve a piece \(P\) containing \(v\). \(O(1)\) time.
- Compute \(\min\{\text{dist}_G(s, u) + \text{dist}_P(u, v) : u \in \partial P\}\).
  \(\tilde{O}(\sqrt{r} \log r)\)
- *Perform point location.* \(O(\log^2 r)\)
\( \tilde{O}(n^{4/5}) \) update time, \( O(\log^2 n) \) query time

**Update:**
- Maintain an \( r \)-division. \( O(r) \) time.
- Maintain an MSSP data structure and the DDG for each piece. \( \tilde{O}(r) \)
- Run FR-Dijkstra from \( s \) in the union of DDGs. \( \tilde{O}(n/\sqrt{r}) \)
- **Construct a point location data structure for each piece \( P \) with \( s \to \partial P \) distances as additive weights.**
  \( \tilde{O}(n/r \cdot r^{3/4}) = \tilde{O}(n/r^{1/4}) \)

**Query:**
- Retrieve a piece \( P \) containing \( v \). \( O(1) \) time.
- Compute \( \min \{ \text{dist}_G(s, u) + \text{dist}_P(u, v) : u \in \partial P \} \).
  \( O(\sqrt{r} \log r) \)
- **Perform point location.** \( O(\log^2 r) \)

**Balance:** \( r = n^{4/5} \).
Strong Connectivity

Maintain $G$ so that an identifier of the strongly connected component of each vertex can be returned.

General Graphs
Both update and query time $O(n^{1-\epsilon})$ is not possible conditional on SETH.

[Abboud-Vassilevska Williams; FOCS 2014]

Plane Graphs
The dynamic plane transitive closure data structure of [Diks-Sankowski; ESA 2007] yields $\tilde{O}(n^{1/2})$ update and query time.
Task

Maintain $G$ so that an identifier of the strongly connected component of each vertex can be returned.
Task

Maintain $G$ so that an identifier of the strongly connected component of each vertex can be returned.

General Graphs

Both update and query time $\mathcal{O}(n^{1-\epsilon})$ is not possible conditional on SETH. [Abboud-Vassilevska Williams; FOCS 2014]
Strong Connectivity

Task

Maintain $G$ so that an identifier of the strongly connected component of each vertex can be returned.

General Graphs

Both update and query time $O(n^{1-\epsilon})$ is not possible conditional on SETH. [Abboud-Vassilevska Williams; FOCS 2014]

Plane Graphs

The dynamic plane transitive closure data structure of [Diks-Sankowski; ESA 2007] yields $\tilde{O}(n^{1/2})$ update and query time.
Theorem [Subramanian; ESA 1993]

Let $P$ be a piece of an $r$-division.
Theorem [Subramanian; ESA 1993]

Let $P$ be a piece of an $r$-division. There exists a directed graph $X_P$, where $\partial P \subseteq V(X_P)$, of size $O(\sqrt{r \log r})$ satisfying the following property:

for any $u, v \in \partial P$, $u$ can reach $v$ in $P$ $\iff$ $u$ can reach $v$ in $X_P$. The graph $X_P$ can be computed in $O(r \log r)$ time.
Theorem [Subramanian; ESA 1993]

Let $P$ be a piece of an $r$-division. There exists a directed graph $X_P$, where $\partial P \subseteq V(X_P)$, of size $O(\sqrt{r} \log r)$ satisfying the following property:

for any $u, v \in \partial P$, $u$ can reach $v$ in $P \iff u$ can reach $v$ in $X_P$. 

Theorem [Subramanian; ESA 1993]

Let \( P \) be a piece of an \( r \)-division. There exists a directed graph \( X_P \), where \( \partial P \subseteq V(X_P) \), of size \( O(\sqrt{r \log r}) \) satisfying the following property:
for any \( u, v \in \partial P \), \( u \) can reach \( v \) in \( P \) \( \iff \) \( u \) can reach \( v \) in \( X_P \).

The graph \( X_P \) can be computed in \( O(r \log r) \) time.
Strong Connectivity

Maintain an \( r \)-division.

Maintain a reachability certificate for each piece of the \( r \)-division.

Run a textbook SCC algorithm on graph \( X = \bigcup X_P \) (i.e. the union of reachability certificates).

Sort the SCCs topologically.

\[ \tilde{O}(n/\sqrt{r}) \]
Maintain an \( r \)-division. \( \mathcal{O}(r) \)
Maintain an $r$-division. $\mathcal{O}(r)$

Maintain a reachability certificate for each piece of the $r$-division. $\tilde{\mathcal{O}}(r)$
- Maintain an $r$-division. $O(r)$

- Maintain a reachability certificate for each piece of the $r$-division. $\tilde{O}(r)$

- Run a textbook SCC algorithm on graph $X = \bigcup X_P$ (i.e. the union of reachability certificates). $\tilde{O}(n/\sqrt{r})$
Strong Connectivity

- Maintain an $r$-division. $\mathcal{O}(r)$

- Maintain a reachability certificate for each piece of the $r$-division. $\tilde{\mathcal{O}}(r)$

- Run a textbook SCC algorithm on graph $X = \bigcup X_P$ (i.e. the union of reachability certificates). $\tilde{\mathcal{O}}(n/\sqrt{r})$

- Sort the SCCs topologically. $\tilde{\mathcal{O}}(n/\sqrt{r})$
Observation

Let $b_1, \ldots, b_k$ be some vertices of $G$ lying in distinct strongly connected components.
Observation
Let $b_1, \ldots, b_k$ be some vertices of $G$ lying in distinct strongly connected components.
A vertex $v$ is strongly connected to some $b_j$ if and only if
- $b_j$ is in the topologically earliest SCC reachable from $v$, and
- $b_j$ is in the topologically latest SCC that can reach $v$. 
Observation

Let $b_1, \ldots, b_k$ be some vertices of $G$ lying in distinct strongly connected components. A vertex $v$ is strongly connected to some $b_j$ if and only if

1. $b_j$ is in the topologically earliest SCC reachable from $v$, and
2. $b_j$ is in the topologically latest SCC that can reach $v$. 

Diagram:

- $b_1$
- $b_2$
- $b_3$
- $b_4$
- $b_5$
- $b_6$
- $v$
Observation

Let $b_1, \ldots, b_k$ be some vertices of $G$ lying in distinct strongly connected components.

A vertex $v$ is strongly connected to some $b_j$ if and only if
- $b_j$ is in the topologically earliest SCC reachable from $v$, and
- $b_j$ is in the topologically latest SCC that can reach $v$. 

Impossible
Observation

Let $b_1, \ldots, b_k$ be some vertices of $G$ lying in distinct strongly connected components. A vertex $v$ is strongly connected to some $b_j$ if and only if

- $b_j$ is in the topologically earliest SCC reachable from $v$, and
- $b_j$ is in the topologically latest SCC that can reach $v$. 
Observation

Let $b_1, \ldots, b_k$ be some vertices of $G$ lying in distinct strongly connected components. A vertex $v$ is strongly connected to some $b_j$ if and only if:

- $b_j$ is in the **topologically earliest** SCC reachable from $v$, and
- $b_j$ is in the **topologically latest** SCC that can reach $v$. 

\[ b_1 \rightarrow b_2 \rightarrow b_3 \rightarrow b_4 \rightarrow b_5 \rightarrow b_6 \rightarrow v \]
Observation

Let $b_1, \ldots, b_k$ be some vertices of $G$ lying in distinct strongly connected components. A vertex $v$ is strongly connected to some $b_j$ if and only if

- $b_j$ is in the **topologically earliest SCC reachable from $v$**, and
- $b_j$ is in the **topologically latest SCC that can reach $v$**.

```
  b_1   b_2   b_3   b_4   b_5   b_6
```

Impossible
Observation
Let $b_1, \ldots, b_k$ be some vertices of $G$ lying in distinct strongly connected components.
A vertex $v$ is strongly connected to some $b_j$ if and only if
- $b_j$ is in the topologically earliest SCC reachable from $v$, and
- $b_j$ is in the topologically latest SCC that can reach $v$.

Construct per-piece point location data structures with additive weights stemming from the topological order of the SCCs of $X$. $\tilde{O}(n/r^{1/4})$
**Observation**

Let $b_1, \ldots, b_k$ be some vertices of $G$ lying in distinct strongly connected components.

A vertex $v$ is strongly connected to some $b_j$ if and only if

- $b_j$ is in the **topologically earliest SCC** reachable from $v$, and
- $b_j$ is in the **topologically latest SCC** that can reach $v$.

- Construct per-piece point location data structures with additive weights stemming from the topological order of the SCCs of $X$. $\tilde{O}(n/r^{1/4})$

- If the point location queries are inconclusive, $v$ is in an SCC fully contained in $P$. 
Observation

Let \( b_1, \ldots, b_k \) be some vertices of \( G \) lying in distinct strongly connected components.
A vertex \( v \) is strongly connected to some \( b_j \) if and only if
- \( b_j \) is in the topologically earliest SCC reachable from \( v \), and
- \( b_j \) is in the topologically latest SCC that can reach \( v \).

Construct per-piece point location data structures with additive weights stemming from the topological order of the SCCs of \( X \). \( \tilde{O}(n/r^{1/4}) \)

If the point location queries are inconclusive, \( v \) is in an SCC fully contained in \( P \).
(Maintain per-piece SCC identifiers. \( \tilde{O}(r) \))
Final Remarks

Our results: $O\left(\frac{n^4}{5}\right)$ update time and $O\left(\log^2 n\right)$ query time for:

- single-source shortest paths,
- strong connectivity in fully dynamic weighted directed planar graphs.

The initialization requires $O\left(n \log^2 n\right)$ time and the space occupied is $O\left(n \log n\right)$.

We also show a dynamic closest facility data structure. This is the first work that utilizes additively weighted Voronoi diagrams machinery in dynamic graph algorithms.

Can more problems benefit?
Our results: $\tilde{O}(n^{4/5})$ update time and $O(\log^2 n)$ query time for:

- single-source shortest paths, and
- strong connectivity

in fully dynamic weighted directed planar graphs.
Our results: $\tilde{O}(n^{4/5})$ update time and $O(\log^2 n)$ query time for:
- single-source shortest paths, and
- strong connectivity

in fully dynamic weighted directed planar graphs.

The initialization requires $O(n \log^2 n)$ time and the space occupied is $O(n \log n)$. 
Our results: $\tilde{O}(n^{4/5})$ update time and $O(\log^2 n)$ query time for:

- single-source shortest paths, and
- strong connectivity

in fully dynamic weighted directed planar graphs.

- The initialization requires $O(n \log^2 n)$ time and the space occupied is $O(n \log n)$.
- We also show a dynamic closest facility data structure.
Our results: $\tilde{O}(n^{4/5})$ update time and $O(\log^2 n)$ query time for:
- single-source shortest paths, and
- strong connectivity
in fully dynamic weighted directed planar graphs.

The initialization requires $O(n \log^2 n)$ time and the space occupied is $O(n \log n)$.

We also show a dynamic closest facility data structure.

This is the first work that utilizes additively weighted Voronoi diagrams machinery in dynamic graph algorithms.
Our results: $\tilde{O}(n^{4/5})$ update time and $O(\log^2 n)$ query time for:
- single-source shortest paths, and
- strong connectivity

in fully dynamic weighted directed planar graphs.

- The initialization requires $O(n \log^2 n)$ time and the space occupied is $O(n \log n)$.
- We also show a dynamic closest facility data structure.

This is the first work that utilizes additively weighted Voronoi diagrams machinery in dynamic graph algorithms.

Can more problems benefit?
Thank you for your attention!

Questions?