Constant Approximation of Min-Distances in Near-Linear Time

Shiri Chechik               Tianyi Zhang

TEL AVIV UNIVERSITY
Distance parameters in graphs

\[ G = (V, E, \omega) \] be a weighted directed graph

**Distance parameters:**
- **Diameter**, **radius**, **eccentricity**

- **Eccentricity** of any vertex = Maximum distance to other vertices
- **Radius** = Minimum eccentricity among all vertices
- **Diameter** = Maximum eccentricity among all vertices
Distance parameters in graphs

$G = (V, E, \omega)$ be a weighted directed graph

Distance parameters:
- **diameter**, **radius**, **eccentricity**

- **Eccentricity** of any vertex
  - $= \text{Maximum distance to other vertices}$

- **Radius** = Minimum eccentricity among all vertices

- **Diameter** = Maximum eccentricity among all vertices
Distance parameters in graphs

$G = (V, E, \omega)$ be a weighted directed graph

Distance parameters:
- diameter, radius, eccentricity

- **Eccentricity** of any vertex  
  = Maximum distance to other vertices

- **Radius** = Minimum eccentricity among all vertices

- **Diameter** = Maximum eccentricity among all vertices
Distance parameters in graphs

$G = (V, E, \omega)$ be a weighted directed graph

Distance parameters:
  - **diameter**, **radius**, **eccentricity**

- **Eccentricity** of any vertex
  = Maximum distance to other vertices

- **Radius** = Minimum eccentricity among all vertices

- **Diameter** = Maximum eccentricity among all vertices
Variants of distances in digraphs

• In directed graphs, \( \text{dist}(u, v) \neq \text{dist}(v, u) \) not symmetric

• **Round-trip distance** = \( \text{dist}(u, v) + \text{dist}(v, u) \)

• **Max-distance** = \( \max \{ \text{dist}(u, v), \text{dist}(v, u) \} \)

• **Min-distance** = \( \min \{ \text{dist}(u, v), \text{dist}(v, u) \} \)

Exactly computing distance parameters (\( \text{ecc}, \text{diam}, \text{rad} \)) requires \( m^{2-c} \) time under SETH and HSC

Faster algorithms need to allow approximations
Variants of distances in digraphs

- **Rndtrip-dist & max-dist** satisfy **triangle inequalities**

- **Min-dist** violates **triangle inequalities**

![Diagram](image)

**round-trip distance**

\[ \text{rnd}(u, v) + \text{rnd}(v, w) \geq \text{rnd}(u, w) \]

**min-distance**

\[ \text{min-dist}(u, v) + \text{min-dist}(v, w) \]

\[ < \text{min-dist}(u, w) = \infty \]
Variants of distances in digraphs

- Rndtrip-dist & max-dist satisfy **triangle inequalities**
- Min-dist violates **triangle inequalities**

\[ \text{Ecc}(v) \text{ is } 2\text{-approx of round-trip diameter} \]

\[ \text{Ecc}(v) \text{ is } \infty\text{-approx of min-diameter} \]
## A short history

<table>
<thead>
<tr>
<th>reference</th>
<th>which param?</th>
<th>approx</th>
<th>runtime</th>
<th>input type</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVW, 2016</td>
<td>min-diameter</td>
<td>2</td>
<td>m</td>
<td>acyclic</td>
</tr>
<tr>
<td>AVW, 2016</td>
<td>min-radius</td>
<td>3</td>
<td>$mn^{1/2}$</td>
<td>acyclic</td>
</tr>
<tr>
<td>DK, 2021</td>
<td>min-radius</td>
<td>k</td>
<td>$mn^{1/k}$</td>
<td>acyclic</td>
</tr>
<tr>
<td>DK, 2021</td>
<td>min-eccentricities</td>
<td>$k + 0.001$</td>
<td>$mn^{1/k}$</td>
<td>acyclic</td>
</tr>
<tr>
<td>open</td>
<td>all param</td>
<td>$O(1)$</td>
<td>m</td>
<td>acyclic</td>
</tr>
</tbody>
</table>
# A short history

<table>
<thead>
<tr>
<th>reference</th>
<th>which param?</th>
<th>approx</th>
<th>runtime</th>
<th>input type</th>
</tr>
</thead>
<tbody>
<tr>
<td>DWV+, 2019</td>
<td>min-diameter</td>
<td>4k-5</td>
<td>$mn^{1/k}$</td>
<td>general</td>
</tr>
<tr>
<td>DWV+, 2019</td>
<td>min-radius</td>
<td>3</td>
<td>$mn^{1/2}$</td>
<td>general</td>
</tr>
<tr>
<td>DWV+, 2019</td>
<td>min-eccentricities</td>
<td>5.001</td>
<td>$mn^{1/2}$</td>
<td>general</td>
</tr>
<tr>
<td>open</td>
<td>any param</td>
<td>O(1)</td>
<td>m</td>
<td>general</td>
</tr>
</tbody>
</table>
# Our results

<table>
<thead>
<tr>
<th>reference</th>
<th>which param?</th>
<th>approx</th>
<th>runtime</th>
<th>input type</th>
</tr>
</thead>
<tbody>
<tr>
<td>open</td>
<td>all/any param</td>
<td>O(1)</td>
<td>m</td>
<td>acyclic / general</td>
</tr>
<tr>
<td>new</td>
<td>min-diameter</td>
<td>4k-5 vs 4</td>
<td>$mn^{1/k}$ vs m</td>
<td>general</td>
</tr>
<tr>
<td>new</td>
<td>min-radius</td>
<td>3 vs 4</td>
<td>$mn^{1/2}$ vs m</td>
<td>general</td>
</tr>
<tr>
<td>new</td>
<td>min-eccentricities</td>
<td>5.001</td>
<td>$mn^{1/2}$ vs m</td>
<td>general</td>
</tr>
<tr>
<td>new</td>
<td>min-radius</td>
<td>k vs 3</td>
<td>$mn^{1/k}$ vs m</td>
<td>acyclic</td>
</tr>
<tr>
<td>new</td>
<td>min-eccentricities</td>
<td>k+0.01 vs 3.01</td>
<td>$mn^{1/k}$ vs m</td>
<td>acyclic</td>
</tr>
</tbody>
</table>
## Our results

<table>
<thead>
<tr>
<th>Reference</th>
<th>Which Param?</th>
<th>Approx</th>
<th>Runtime</th>
<th>Input Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>open</td>
<td>all/any param</td>
<td>O(1)</td>
<td>m</td>
<td>acyclic / general</td>
</tr>
<tr>
<td>new</td>
<td>min-diameter</td>
<td>4k-5 vs 4</td>
<td>mn^{1/k} vs m</td>
<td>general</td>
</tr>
<tr>
<td>new</td>
<td>min-radius</td>
<td>3 vs 4</td>
<td>mn^{1/2} vs m</td>
<td>general</td>
</tr>
<tr>
<td>new</td>
<td>min-eccentricities</td>
<td>5.001</td>
<td>mn^{1/2} vs m</td>
<td>general</td>
</tr>
<tr>
<td>new</td>
<td>min-radius</td>
<td>k vs 3</td>
<td>mn^{1/k} vs m</td>
<td>acyclic</td>
</tr>
<tr>
<td>new</td>
<td>min-eccentricities</td>
<td>k+0.01 vs 3.01</td>
<td>mn^{1/k} vs m</td>
<td>acyclic</td>
</tr>
</tbody>
</table>
Min-diameter in general digraphs
**Goal:** decide min-diam $< D$ or $> D/4$

1. Pick a random vertex $t$

2. Compute SSSP in to and from $t$

3. Define two sets:
   
   $U_+ = \{u \mid \text{dist}(t, u) < D/4\}$
   
   $U_- = \{u \mid \text{dist}(u, t) < D/4\}$

4. Recurse on $G[U_+]$ and $G[U_-]$
**Goal:** decide min-diam $< D$ or $> D/4$

1. **Pick a random vertex** $t$

2. Compute SSSP in to and from $t$

3. Define two sets:
   
   $U_+ = \{ u \mid \text{dist}(t, u) < D/4 \}$
   
   $U_- = \{ u \mid \text{dist}(u, t) < D/4 \}$

4. Recurse on $G[U_+]$ and $G[U_-]$
**Goal:** decide min-diam $<D$ or $>D/4$

1. Pick a random vertex $t$

2. **Compute SSSP in** to and **from** $t$

3. Define two sets:
   \[ U_+ = \{ u \mid \text{dist}(t, u) < D/4 \} \]
   \[ U_- = \{ u \mid \text{dist}(u, t) < D/4 \} \]

4. Recurse on $G[U_+]$ and $G[U_-]$
**Goal:** decide min-diam $<D$ or $>D/4$

1. Pick a random vertex $t$
2. Compute SSSP in to and from $t$
3. Define two sets:
   
   $U_+ = \{u \mid \text{dist}(t, u) < D/4\}$
   
   $U_- = \{u \mid \text{dist}(u, t) < D/4\}$

4. Recurse on $G[U_+]$ and $G[U_-]$
First attempt

**Goal:** decide min-diam \(<D\) or \(>D/4\)

1. Pick a random vertex \(t\)
2. Compute SSSP in to and from \(t\)
3. Define two sets:
   - \(U_+ = \{u \mid \text{dist}(t, u) < D/4\}\)
   - \(U_- = \{u \mid \text{dist}(u, t) < D/4\}\)
   - If \(U_+ \cup U_- \neq V\) then **claim min-diam \(>D/4\)**
4. Recurse on \(G[U_+]\) and \(G[U_-]\)
First attempt

**Goal:** decide min-diam $<D$ or $>D/4$

1. Pick a random vertex $t$

2. Compute SSSP in to and from $t$

3. **Define two sets:**

   - $U_+ = \{ v \mid \text{dist}(t,v) < D/4 \}$
   - $U_- = \{ v \mid \text{dist}(v,t) < D/4 \}$

   **Endpoints of min-diam should belong to the same side**

4. Recurse on $G[U_+]$ and $G[U_-]$

The min-distance between $U_+$ and $U_-$ is at most $D/2$
First attempt

**Goal:** decide min-diam $< D$ or $> D/4$

1. Pick a random vertex $t$

2. Compute SSSP in to and from $t$

3. Define two sets:
   
   - $U_+ = \{ u \mid \text{dist}(t, u) < D/4 \}$
   - $U_- = \{ u \mid \text{dist}(u, t) < D/4 \}$

4. Recurse on $G[U_+]$ and $G[U_-]$
**Goal:** decide min-diam \(<D \text{ or } >D/4\)

1. Pick a random vertex \(t\)

2. Compute SSSP in \(t\) and from \(t\)

3. Define two sets:
   \[ U_+ = \{ u \mid \text{dist}(t, u) < D/4 \} \]
   \[ U_- = \{ u \mid \text{dist}(u, t) < D/4 \} \]

4. **Recurse on** \(G[U_+]\) and \(G[U_-]\)
First attempt

Some issues:

- **Soundness**
  A large min-distance within \( G[U_+] \) does not imply a large min-distance in \( G[V] \)

- **Runtime**
  \( G[U_+] \) and \( G[U_-] \) are usually intersecting, runtime can be high

Recursions on \( G[U_+] \) and \( G[U_-] \)
Some issues:

- **Soundness**
  A large min-distance within $G[U_+]$ does not imply a large min-distance in $G[V]$

- **Runtime**
  $G[U_+]$ and $G[U_-]$ are usually intersecting, runtime can be high

Recursions on $G[U_+]$ and $G[U_-]$
First attempt

Some issues:

- **Soundness**
  A large min-distance within $G[U_+]$ does not imply a large min-distance in $G[V]$

- **Runtime**
  $G[U_+]$ and $G[U_-]$ are usually intersecting, runtime can be high
Soundness issue

- A large min-distance within $G[U_+]$ does not imply a large min-distance in $G[V]$
Soundness issue

- A large min-distance within $G[U_+]$ does not imply a large min-distance in $G[V]$

- Observation
  Distance from $t$ to $z$ is $< D/4 + D/4$

A shortcut through $G[U_-]$ of length $< D/4$
Soundness issue

- A large min-distance within $G[U_+]$ does not imply a large min-distance in $G[V]$

- **Observation**
  Distance from $t$ to $z$ is $< D/4 + D/4$

- **Idea**
  Add more vertices in the recursion
  $S_+ = \{ u \mid \text{dist}(t, u) < D/2 \}$
  This preserves pairwise distances for
  $U_+ = \{ u \mid \text{dist}(t, u) < D/4 \}$

Recurse on a larger graph
Two different vertex sets:

- \( S_+ = \{ u \mid \text{dist}(t, u) < D/2 \} \)
- \( U_+ = \{ u \mid \text{dist}(t, u) < D/4 \} \)

- \( G[S_+] \) only preserves pairwise distances among \( U_+ \), not the entire \( S_+ \)

- In general, the recursive algorithm needs to take two parameters \((S, U)\) such that \( U \subseteq S \subseteq V \)
Soundness issue

- Two different vertex sets:
  \[ S_+ = \{ u \mid \text{dist}(t, u) < D/2 \} \]
  \[ U_+ = \{ u \mid \text{dist}(t, u) < D/4 \} \]

- \( G[S_+] \) only preserves pairwise distances among \( U_+ \), not the entire \( S_+ \)

- In general, the recursive algorithm needs to take two parameters \((S, U)\) such that \( U \subseteq S \subseteq V \)
Soundness issue

- Two different vertex sets:
  \[ S_+ = \{ u \mid \text{dist}(t, u) < D/2 \} \]
  \[ U_+ = \{ u \mid \text{dist}(t, u) < D/4 \} \]

- \( G[S_+] \) only preserves pairwise distances among \( U_+ \), not the entire \( S_+ \)

- In general, the recursive algorithm needs to take two parameters \((S, U)\) such that \( U \subseteq S \subseteq V \)
Runtime issue

- $S_+$ and $S_-$ are usually intersecting, so the runtime can be high

\[ S_+ = \{ u \mid \text{dist}(t, u) < D/2 \} \]

\[ S_- = \{ u \mid \text{dist}(u, t) < D/2 \} \]
Runtime issue

- $S_+$ and $S_-$ are usually intersecting, so the runtime can be high
  
  $S_+ = \{u \mid \text{dist}(t, u) < D/2\}$
  
  $S_- = \{u \mid \text{dist}(u, t) < D/2\}$

- **Enforce disjointness** by recursing on instances $(S_+ \setminus S_-, U_+) \& (S_- \setminus S_+, U_-)$
Runtime issue

- $S_+$ and $S_-$ are usually intersecting, so the runtime can be high
  $S_+ = \{ u \mid \text{dist}(t, u) < D/2 \}$
  $S_- = \{ u \mid \text{dist}(u, t) < D/2 \}$

- **Enforce disjointness** by recursing on instances $(S_+ \setminus S_-, U_+)$ & $(S_- \setminus S_+, U_-)$

- How about soundness again?
Runtime issue

- **Enforce disjointness** by recursing on instances \((S_+ \setminus S_-, U_+) \& (S_- \setminus S_+, U_-)\)

- How about soundness again? 
  \(x, y\) such that \(\text{dist}_{G[S_\pm S_-]}(x, y) = \infty\)
Runtime issue

- **Enforce disjointness** by recursing on instances \((S_+ \setminus S_-, U_+)\) & \((S_- \setminus S_+, U_-)\)

- How about soundness again? \(x, y\) such that \(\text{dist}^{\min}_{G[S_+ \setminus S_-]}(x, y) = \infty\) but \(\text{dist}_{G[S_+]}(x, y) < D/4\)

A shortcut through \(G[S_+ \cap S_-]\) of length < \(D/4\)
Runtime issue

- **Enforce disjointness** by recursing on instances $(S_+ \setminus S_-, U_+) \& (S_- \setminus S_+, U_-)$

- How about soundness again? $x, y$ such that $\text{dist}^{\text{min}}_{G[S_+ \setminus S_-]}(x, y) = \infty$
  but $\text{dist}_{G[S_+]}(x, y) < D/4$

- $\text{dist}(x, t) \leq \text{dist}(x, z) + \text{dist}(z, t) < 3D/4$
Runtime issue

- **Enforce disjointness** by recursing on instances \((S_+ \setminus S_-, U_+)\) & \((S_- \setminus S_+, U_-)\)

- How about soundness again?
  - \(x, y\) such that \(\text{dist}^{\min}_{G[S_+ \setminus S_-]}(x, y) = \infty\)
  - but \(\text{dist}_{G[S_+]}(x, y) < D/4\)

- \(\text{dist}(x, t) \leq \text{dist}(x, z) + \text{dist}(z, t) < 3D/4\)
  - \(\text{dist}(x, t) + \text{dist}(t, y) < D\)

- Recurse on \((S_+ \setminus S_-, C_+)\)
  - \(C_+ = U_+ \cap \{x \mid \text{dist}(x, t) \geq 3D/4\}\)

Already **rule out** that \(x, y\) could be endpoints of diameter by **single-source distances from** \(t\)
The recursive algorithm

On input pair \((S, C)\)

1. Pick a random vertex \(t \in C\)
2. Compute SSSP in to and from \(t\)
3. Define sets:
   \[S_+ = \{x \mid \text{dist}(t, x) < D/2\}\]
   \[C_+ = C \cap \{x \mid \text{dist}(t, x) < D/4, \text{dist}(x, t) \geq 3D/4\}\]
4. Recurse on \((S_+ \setminus S_-, C_+ \setminus S_-)\)
   and symmetrically \((S_- \setminus S_+, C_- \setminus S_+)\)
The recursive algorithm

On input pair \((S, C)\)

1. **Pick a random vertex** \(t \in C\)

2. Compute SSSP in to and from \(t\)

3. Define sets:
   \[ S_+ = \{ x \mid \text{dist}(t, x) < D/2 \} \]
   \[ C_+ = C \cap \{ x \mid \text{dist}(t, x) < D/4, \text{dist}(x, t) \geq 3D/4 \} \]

4. Recurse on \((S_+ \setminus S_-, C_+ \setminus S_-)\)
   and symmetrically \((S_- \setminus S_+, C_- \setminus S_+)\)
The recursive algorithm

On input pair \((S, C)\)

1. Pick a random vertex \(t \in C\)

2. **Compute SSSP in to and from** \(t\)

3. Define sets:
   \[S_+ = \{x \mid \text{dist}(t, x) < D/2\}\]
   \[C_+ = C \cap \{x \mid \text{dist}(t, x) < D/4, \text{dist}(x, t) \geq 3D/4\}\]

4. Recurse on \((S_+ \setminus S_-, C_+ \setminus S_-)\)
   and symmetrically \((S_- \setminus S_+, C_- \setminus S_+)\)
The recursive algorithm

On input pair \((S, C)\)

1. Pick a random vertex \(t \in C\)

2. Compute SSSP in to and from \(t\)

3. Define sets:
   \[S_+ = \{x \mid \text{dist}(t, x) < D/2\}\]
   \[C_+ = C \cap \{x \mid \text{dist}(t, x) < D/4, \text{dist}(x, t) \geq 3D/4\}\]

4. Recurse on \((S_+ \setminus S_-, C_+ \setminus S_-)\)
   and symmetrically \((S_- \setminus S_+, C_- \setminus S_+)\)
The recursive algorithm

On input pair \((S, C)\)

1. Pick a random vertex \(t \in C\)

2. Compute SSSP in to and from \(t\)

3. Define sets:
   \[
   S_+ = \{x \mid \text{dist}(t, x) < D/2\} \\
   C_+ = C \cap \{x \mid \text{dist}(t, x) < D/4, \text{dist}(x, t) \geq 3D/4\}
   \]

4. Recurse on \((S_+ \setminus S_-, C_+ \setminus C_-)\) and symmetrically \((S_- \setminus S_+, C_- \setminus C_+)\)
\(O(\log n)\)-approx min-radius in general digraphs
Goal: decide if the min-radius is $< O(R \log n)$ or $> R$
Goal: decide if the min-radius is $< O(R \log n)$ or $> R$
Goal: decide if the min-radius is $< O(R \log n)$ or $> R$

1. Define four sets $C_{+, -}, S_{+, -}$
   
   $C_+ = \{ u \mid \text{dist}(t, u) < R \}$
   
   $S_+ = \{ u \mid \text{dist}(t, u) < 2R \}$
**Goal:** decide if the min-radius is $< O(R \log n)$ or $> R$

1. Define four sets $C_{+,-}, S_{+,-}$
   - $C_{+} = \{ u \mid \text{dist}(t, u) < R \}$
   - $S_{+} = \{ u \mid \text{dist}(t, u) < 2R \}$

2. If $S_{+} \cup S_{-} = V$
   
   **t is a center** with small radius

An empty area
Goal: decide if the min-radius is $< O(R \log n)$ or $> R$

1. Define four sets $C_{+,-}, S_{+,-}$
   - $C_+ = \{ u \mid \text{dist}(t, u) < R \}$
   - $S_+ = \{ u \mid \text{dist}(t, u) < 2R \}$

2. Define $W = V \setminus (S_+ \cup S_-) \neq \emptyset$
**Goal:** decide if the min-radius is $< O(R \log n)$ or $> R$

1. Define four sets $C_{+,-}, S_{+,-}$
   - $C_+ = \{ u \mid \text{dist}(t, u) < R \}$
   - $S_+ = \{ u \mid \text{dist}(t, u) < 2R \}$

2. Define $W = V \setminus (S_+ \cup S_-) \neq \emptyset$

3. **Contract** $S_-$ into a single node, and **recurse** on the contracted graph (symmetrically for $S_+$)
**Goal:** decide if the min-radius is 

\[ < O(R \log n) \] or \( > R \)

1. Define four sets \( C_{+,-}, S_{+,-} \)
   - \( C_+ = \{ u \mid \text{dist}(t,u) < R \} \)
   - \( S_+ = \{ u \mid \text{dist}(t,u) < 2R \} \)

2. Define \( W = V \setminus (S_+ \cup S_-) \neq \emptyset \)

3. **Contract** \( S_- \) into a single node, and **recurse** on the contracted graph (symmetrically for \( S_+ \))
Runtime issue: \( W = V \setminus (S_+ \cup S_-) \)

appears in both recursion branches
**Runtime issue:** $W = V \setminus (S_+ \cup S_-)$

appears in both recursion branches

- A single vertex could appear **everywhere** on the recursion tree
Runtime issue: $W = V \setminus (S_+ \cup S_-)$ appears in both recursion branches

- A single vertex could appear everywhere on the recursion tree
**Runtime issue:** $W = V \setminus (S_+ \cup S_-)$ appears in **both** recursion branches

- A single vertex could appear **everywhere** on the recursion tree
Contraction & Recurse

Runtime issue: $W = V \setminus (S_+ \cup S_-)$ appears in both recursion branches

• A single vertex could appear everywhere on the recursion tree
Contraction & Recurse

**Runtime issue:** $W = V \setminus (S_+ \cup S_-)$ appears in both recursion branches

- A single vertex could appear **everywhere** on the recursion tree
**Runtime issue:** \( W = V \setminus (S_+ \cup S_-) \) appears in **both** recursion branches

- A single vertex could appear **everywhere** on the recursion tree

- **Pruning while recursing:**
  Every vertex appears in at most two branches on the recursion tree

(Details omitted in this presentation)
Brief sketch of 4-approx of min-radius
**Goal:** decide if the min-radius is $<4R$ or $>R$

- Cannot use contractions
Achieving 4-approx

**Goal:** decide if the min-radius is $<4R$ or $>R$

- Cannot use contractions

- Partition $S = S_+ \cup S_-
  
  Recurse on $(S_+, C_+), (S_-, C_-)$
Achieving 4-approx

**Goal:** decide if the min-radius is $<4R$ or $>R$

- Cannot use contractions

- Partition $S = S_+ \cup S_-$
  - Recurse on $(S_+, C_+)$, $(S_-, C_-)$

- **Impossible** to preserve pairwise distances in subgraphs

**Diagram:**

- Shortest paths crossing the border
**Achieving 4-approx**

**Goal:** decide if the min-radius is $<4R$ or $>R$

- Cannot use contractions
- Partition $S = S_+ \cup S_-$
  - Recurse on $(S_+, C_+), (S_-, C_-)$
- **Impossible** to preserve pairwise distances in subgraphs
- **Solution:** Only preserve some pairs of distances, i.e. $C \times (C \cup T)$
Further questions

• Better approximation ratio in near-linear time?

Min-diam/radius in general digraphs:

3 in $\tilde{O}(mn^{1/2})$ vs 4 in $\tilde{O}(m)$

Thanks for listening