Warm-Starting **All-Pairs Shortest Paths** with Predictions

Adam Polak
Jonas Schmidt

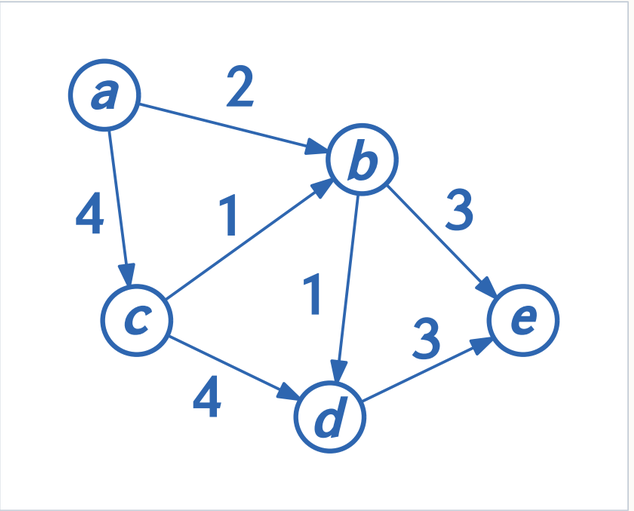
Bocconi University

New Frontiers in Learning-Augmented Algorithms · August 24, 2026

All-Pairs Shortest Paths (APSP)

All-Pairs Shortest Paths (APSP)

Input: A **directed**, **weighted** graph $G = (V, E, w)$

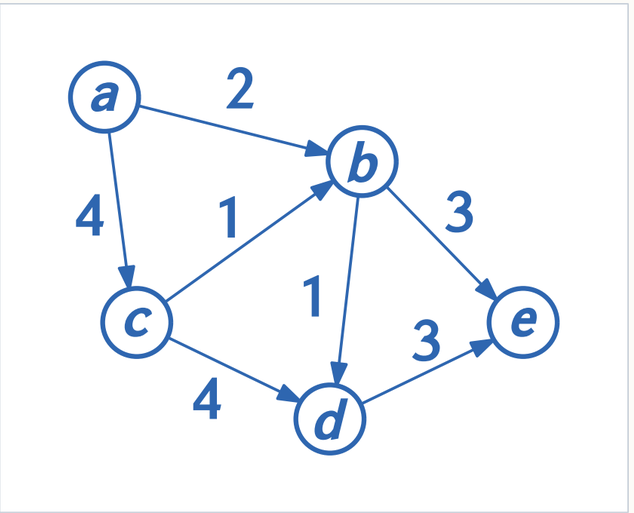


All-Pairs Shortest Paths (APSP)

Input: A **directed**, **weighted** graph $G = (V, E, w)$

Output: The distance matrix D of G

$$\forall u, v \in V : D[u, v] = \mathbf{dist}_G(u, v)$$



D	a	b	c	d	e
a	0	1	4	3	5
b	∞	0	∞	1	3
c	∞	1	0	2	4
d	∞	∞	∞	0	3
e	∞	∞	∞	∞	0

All-Pairs Shortest Paths (APSP)

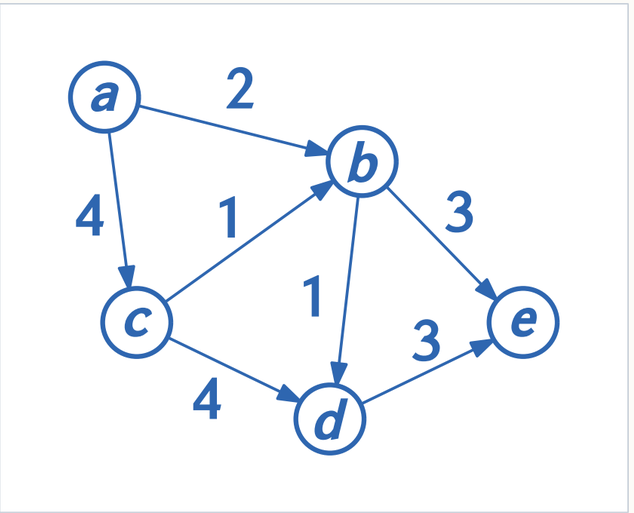
Input: A **directed**, **weighted** graph $G = (V, E, w)$

Output: The distance matrix D of G

$$\forall u, v \in V : D[u, v] = \mathbf{dist}_G(u, v)$$

Complexity:

- Floyd–Warshall algorithm: $O(n^3)$ [Floyd '62] [Warshall '62]
- $n \times$ SSSP: $\tilde{O}(nm)$



D	a	b	c	d	e
a	0	1	4	3	5
b	∞	0	∞	1	3
c	∞	1	0	2	4
d	∞	∞	∞	0	3
e	∞	∞	∞	∞	0

All-Pairs Shortest Paths (APSP)

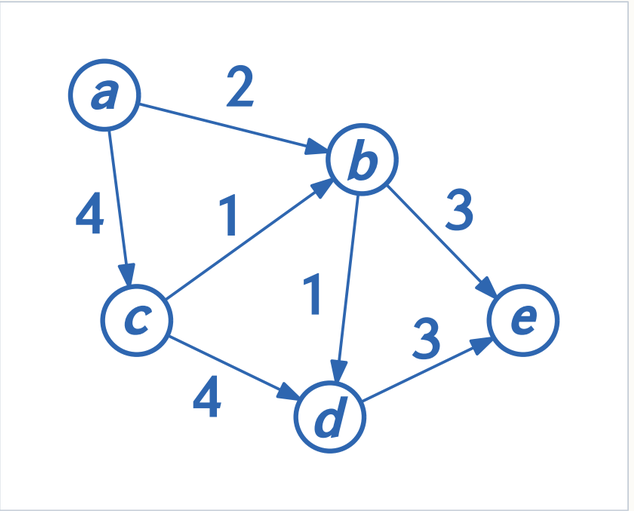
Input: A **directed**, **weighted** graph $G = (V, E, w)$

Output: The distance matrix D of G

$$\forall u, v \in V : D[u, v] = \mathbf{dist}_G(u, v)$$

Complexity:

- Floyd–Warshall algorithm: $O(n^3)$ [Floyd '62] [Warshall '62]
- $n \times$ SSSP: $\tilde{O}(nm)$
- State-of-the-art for dense graphs: $\frac{n^3}{2^{\Omega(\sqrt{\log n})}}$ [Williams 2014]



D	a	b	c	d	e
a	0	1	4	3	5
b	∞	0	∞	1	3
c	∞	1	0	2	4
d	∞	∞	∞	0	3
e	∞	∞	∞	∞	0

All-Pairs Shortest Paths (APSP)

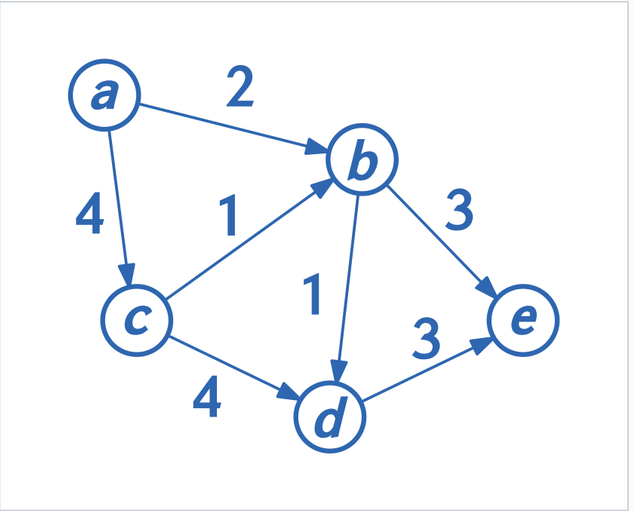
Input: A **directed**, **weighted** graph $G = (V, E, w)$

Output: The distance matrix D of G

$$\forall u, v \in V : D[u, v] = \mathbf{dist}_G(u, v)$$

Complexity:

- Floyd–Warshall algorithm: $O(n^3)$ [Floyd '62] [Warshall '62]
- $n \times$ SSSP: $\tilde{O}(nm)$
- State-of-the-art for dense graphs: $\frac{n^3}{2^{\Omega(\sqrt{\log n})}}$ [Williams 2014]
- **APSP Hypothesis:** no algorithm in time $O(n^{2.9})$



D	a	b	c	d	e
a	0	1	4	3	5
b	∞	0	∞	1	3
c	∞	1	0	2	4
d	∞	∞	∞	0	3
e	∞	∞	∞	∞	0

All-Pairs Shortest Paths (APSP)

Input: A **directed**, **weighted** graph $G = (V, E, w)$

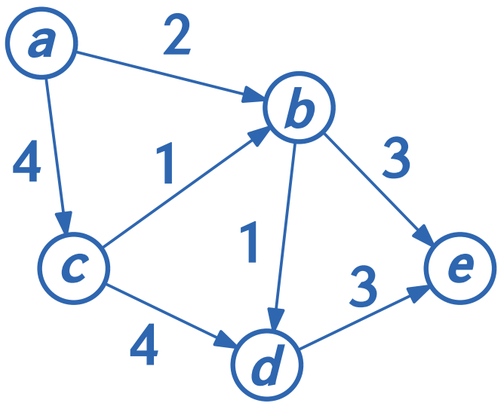
Output: The distance matrix D of G

$$\forall u, v \in V : D[u, v] = \mathbf{dist}_G(u, v)$$

Complexity:

- Floyd–Warshall algorithm: $O(n^3)$ [Floyd '62] [Warshall '62]
- $n \times$ SSSP: $\tilde{O}(nm)$
- State-of-the-art for dense graphs: $\frac{n^3}{2^{\Omega(\sqrt{\log n})}}$ [Williams 2014]
- **APSP Hypothesis:** no algorithm in time $O(n^{2.9})$

Can we **use predictions** to go below the APSP Hypothesis?



D	a	b	c	d	e
a	0	1	4	3	5
b	∞	0	∞	1	3
c	∞	1	0	2	4
d	∞	∞	∞	0	3
e	∞	∞	∞	∞	0

Intermezzo: Fine-Grained Complexity

Intermezzo: Fine-Grained Complexity

- 3 main problems with hypotheses saying they **cannot** be solved polynomially faster than a simple algorithm

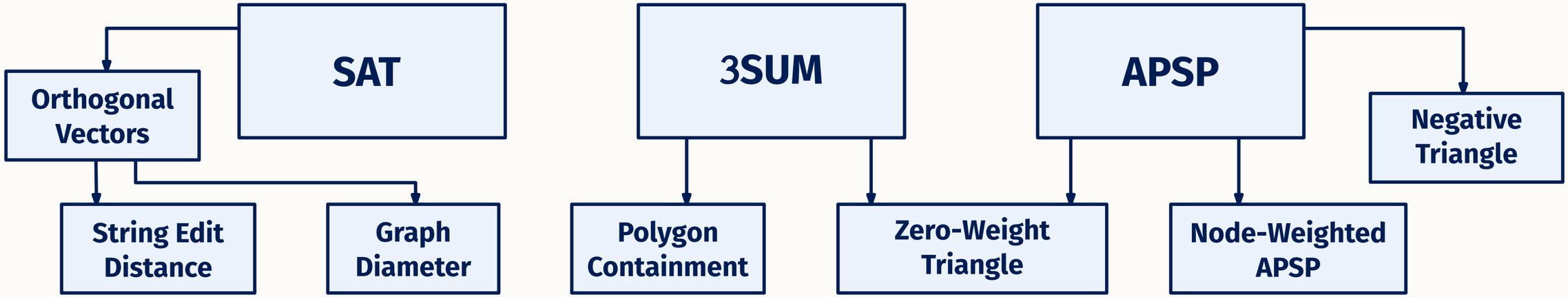
SAT

3SUM

APSP

Intermezzo: Fine-Grained Complexity

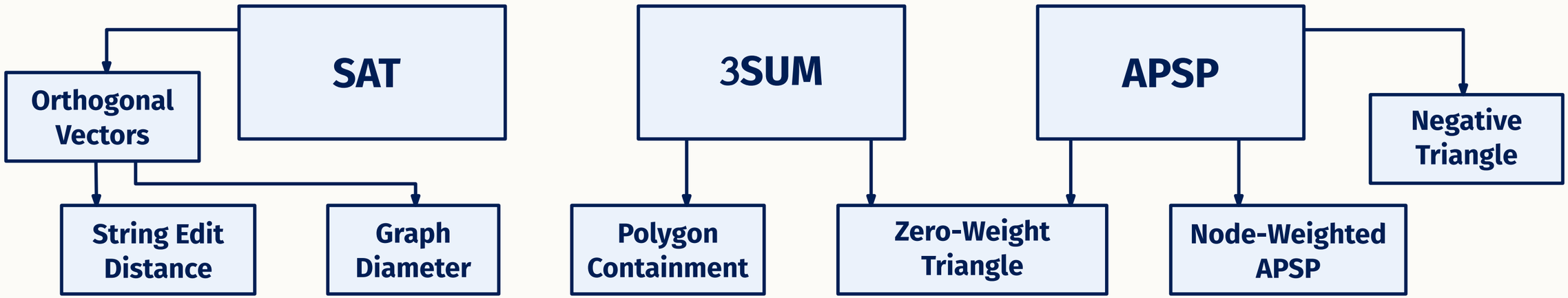
- 3 main problems with hypotheses saying they **cannot** be solved polynomially faster than a simple algorithm



- Reductions give **lower bounds** for many other problems

Intermezzo: Fine-Grained Complexity

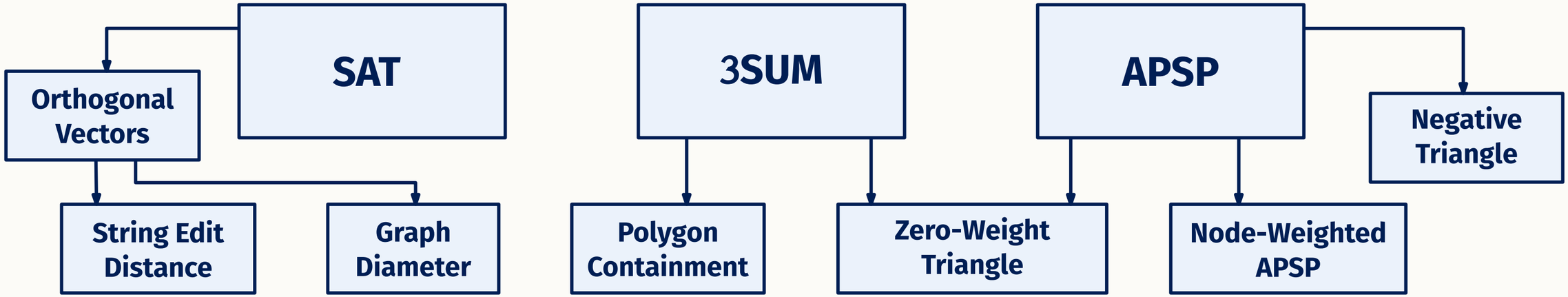
- 3 main problems with hypotheses saying they **cannot** be solved polynomially faster than a simple algorithm



- Reductions give **lower bounds** for many other problems
- Guides algorithmic research to work on key problems

Intermezzo: Fine-Grained Complexity

- 3 main problems with hypotheses saying they **cannot** be solved polynomially faster than a simple algorithm



- Reductions give **lower bounds** for many other problems
- Guides algorithmic research to work on key problems

Can we **use predictions** to circumvent these hypotheses?

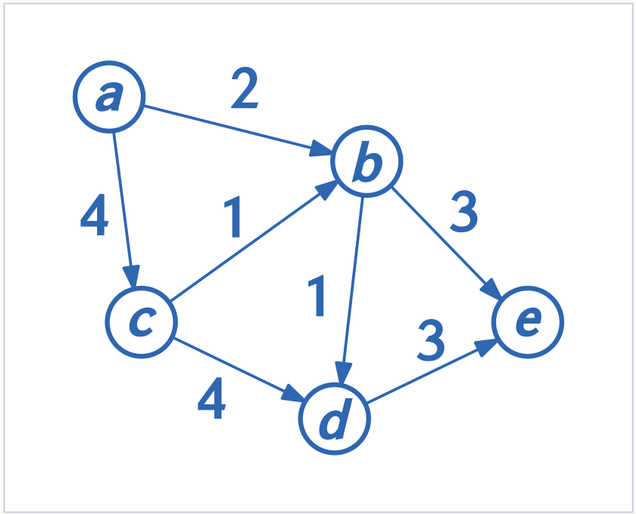
Warm-Starting APSP

Warm-Starting APSP

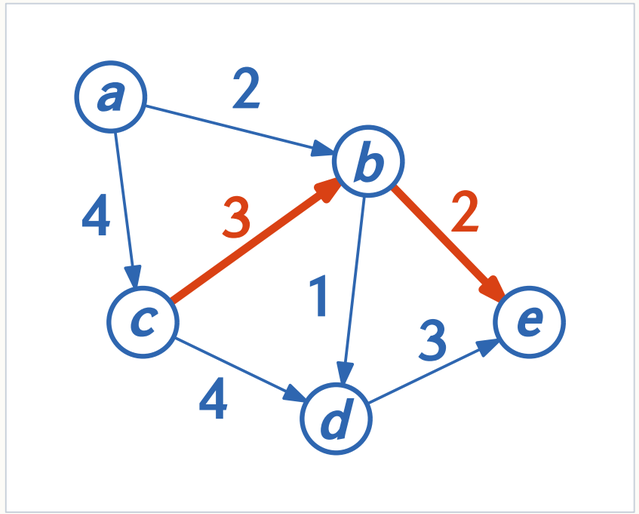
- **Motivation:** Solve APSP faster on different instances with **similar structure**

Warm-Starting APSP

- **Motivation:** Solve APSP faster on different instances with **similar structure**

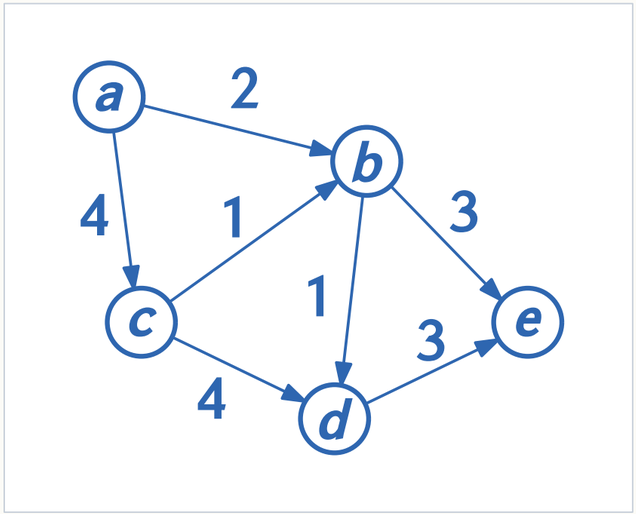


structure = **few input changes??**

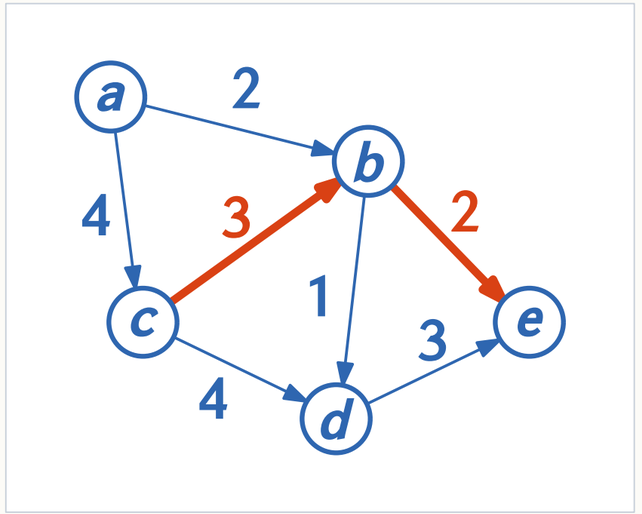


Warm-Starting APSP

- **Motivation:** Solve APSP faster on different instances with **similar structure**



structure = **few input changes??**

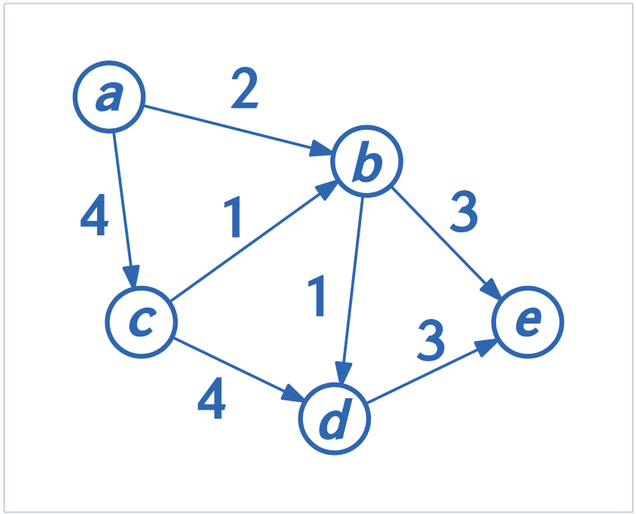


Use **dynamic algorithms** instead!

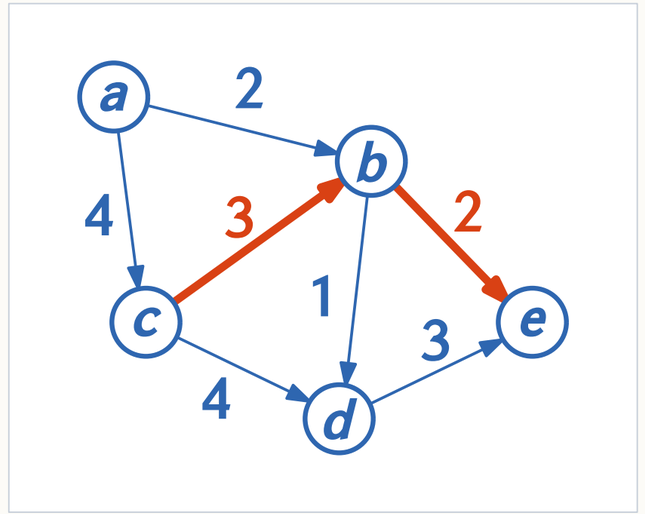
- Subcubic algorithm exists for less than $\sqrt{n}$ vertices changed [Mao 2024]

Warm-Starting APSP

- **Motivation:** Solve APSP faster on different instances with **similar structure**



structure = **few input changes??**

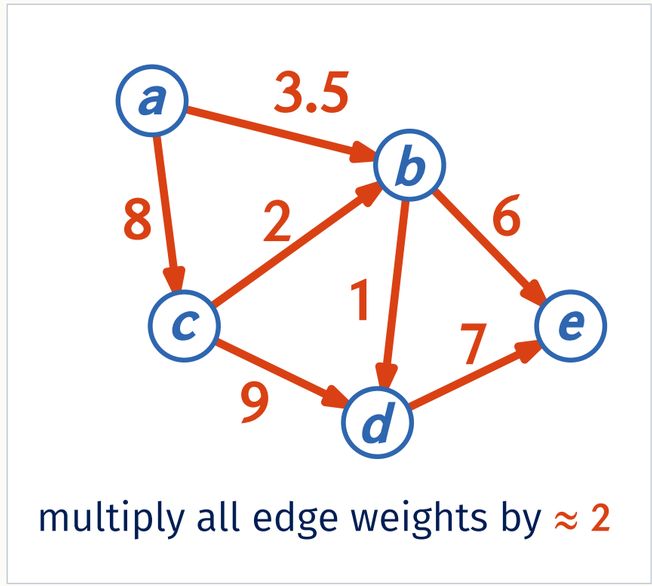


Use **dynamic algorithms** instead!

- Subcubic algorithm exists for less than $\sqrt{n}$ vertices changed [Mao 2024]

different numbers, similar **path structure**

e.g., road network during rush hours

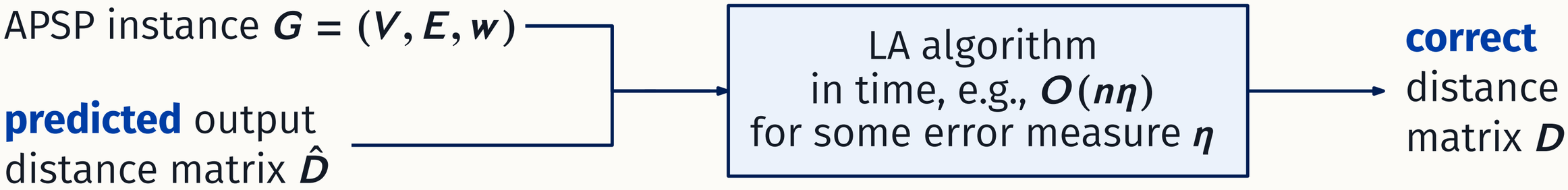


multiply all edge weights by ≈ 2

APSP with Predictions: Approach 1

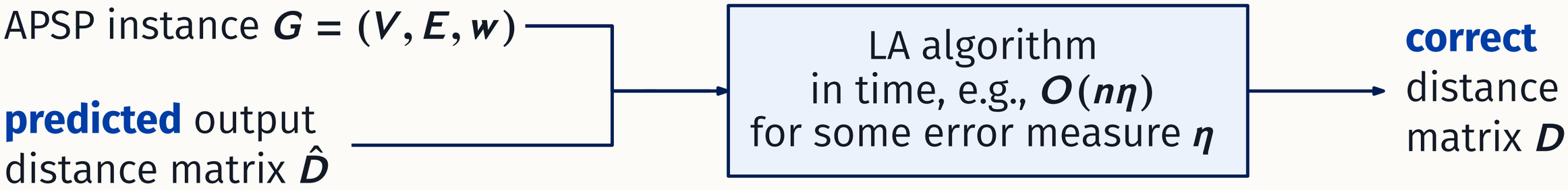
APSP with Predictions: Approach 1

What we would like to have:



APSP with Predictions: Approach 1

What we would like to have:



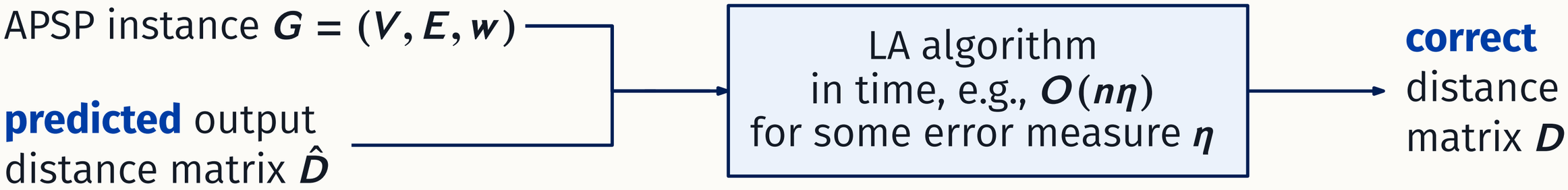
Issue:

Theorem [Vassilevska Williams–Williams 2010]

If there is a $O(n^{2.9})$-time algorithm that **verifies** if a given matrix $\hat{D}$ is the correct distance matrix, the APSP hypothesis is false.

APSP with Predictions: Approach 1

What we would like to have:



Issue:

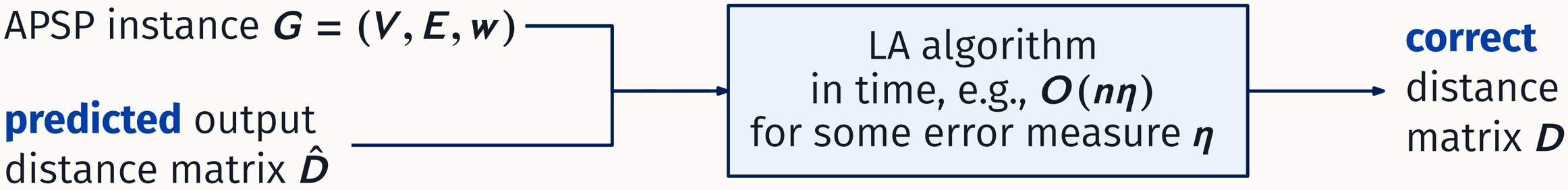
- Verifying $\hat{D}$ is as hard as computing D

Theorem [Vassilevska Williams–Williams 2010]

If there is a $O(n^{2.9})$-time algorithm that **verifies** if a given matrix $\hat{D}$ is the correct distance matrix, the APSP hypothesis is false.

APSP with Predictions: Approach 1

What we would like to have:



Issue:

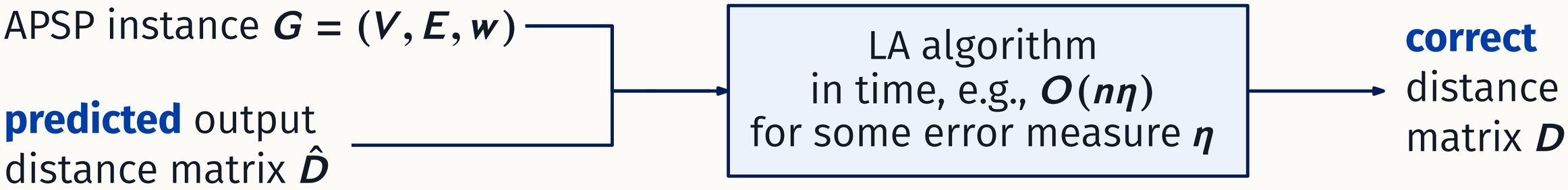
- Verifying $\hat{D}$ is as hard as computing D
- Under the APSP Hypothesis, this prediction **cannot work** with **any reasonable** η!

Theorem [Vassilevska Williams–Williams 2010]

If there is a $O(n^{2.9})$-time algorithm that **verifies** if a given matrix $\hat{D}$ is the correct distance matrix, the APSP hypothesis is false.

APSP with Predictions: Approach 1

What we would like to have:



Issue:

- Verifying $\hat{D}$ is as hard as computing D
- Under the APSP Hypothesis, this prediction **cannot work** with **any reasonable** η!

We need a **stronger prediction**!

Theorem [Vassilevska Williams–Williams 2010]

If there is a $O(n^{2.9})$-time algorithm that **verifies** if a given matrix $\hat{D}$ is the correct distance matrix, the APSP hypothesis is false.

APSP → **APSP Verification**

(Co-)nondeterministic APSP

(Co-)nondeterministic APSP

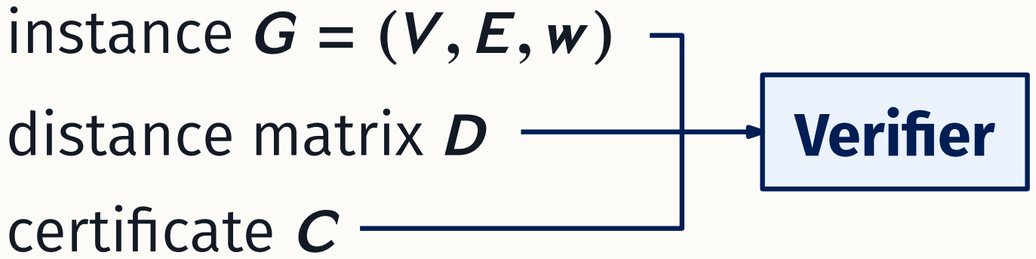
- Does there even exist a prediction with a subcubic algorithm if $\eta = 0$ ($\hat{D} = D$)?

(Co-)nondeterministic APSP

- Does there even exist a prediction with a subcubic algorithm if $\eta = 0$ ($\hat{D} = D$)?
- Yes! Co-nondeterministic algorithms for APSP

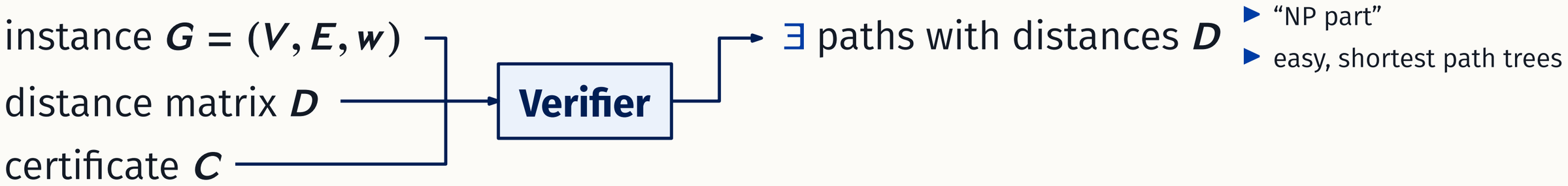
(Co-)nondeterministic APSP

- Does there even exist a prediction with a subcubic algorithm if $\eta = 0$ ($\hat{D} = D$)?
- Yes! Co-nondeterministic algorithms for APSP



(Co-)nondeterministic APSP

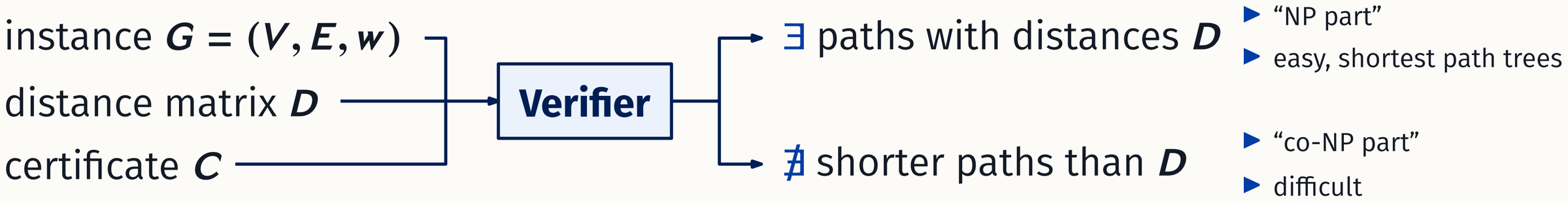
- Does there even exist a prediction with a subcubic algorithm if $\eta = 0$ ($\hat{D} = D$)?
- Yes! Co-nondeterministic algorithms for APSP



- "NP part"
- easy, shortest path trees

(Co-)nondeterministic APSP

- Does there even exist a prediction with a subcubic algorithm if $\eta = 0$ ($\hat{D} = D$)?
- Yes! Co-nondeterministic algorithms for APSP

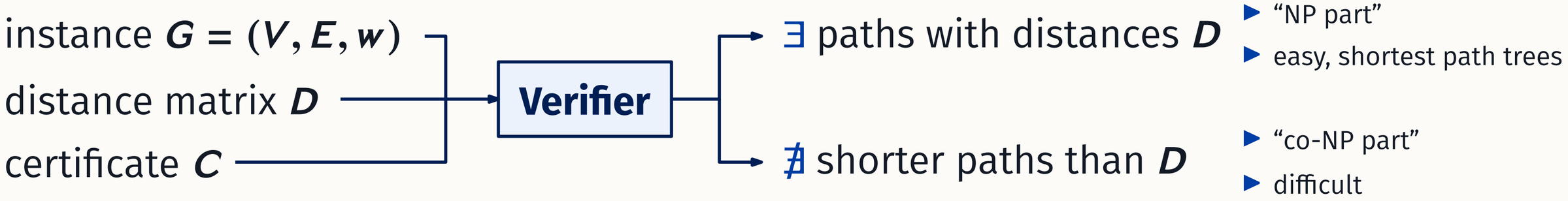


- "NP part"
- easy, shortest path trees

- "co-NP part"
- difficult

(Co-)nondeterministic APSP

- Does there even exist a prediction with a subcubic algorithm if $\eta = 0$ ($\hat{D} = D$)?
- Yes! Co-nondeterministic algorithms for APSP



- "NP part"
- easy, shortest path trees

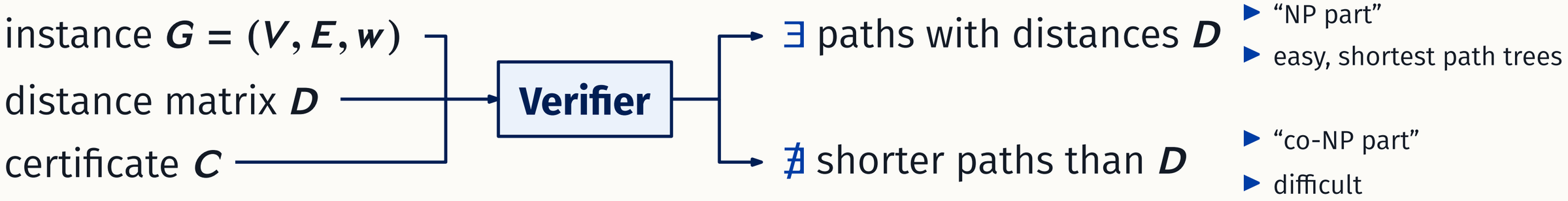
- "co-NP part"
- difficult

2 co-nondeterministic algorithms known:

- Hashing-based: time $O(n^{(\omega+3)/2})$ [Carmosino–Gao–Impagliazzo–Mihajlin–Paturi–Schneider 2017]
- More structural / real inputs: time $O(n^{(\omega+6)/3})$ [Chan–Vassilevska Williams–Xu 2023]

(Co-)nondeterministic APSP

- Does there even exist a prediction with a subcubic algorithm if $\eta = 0$ ($\hat{D} = D$)?
- Yes! Co-nondeterministic algorithms for APSP



- "NP part"
- easy, shortest path trees

- "co-NP part"
- difficult

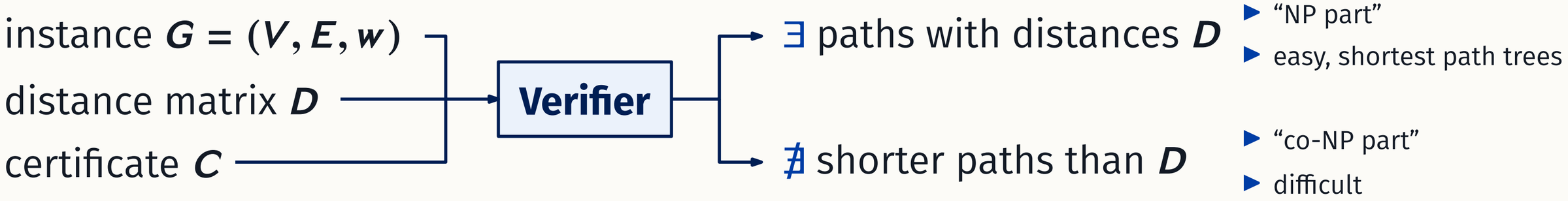
2 co-nondeterministic algorithms known:

$2 \le \omega < 2.372$ is matrix multiplication exponent.
Assume $\omega = 2$. MM in time $O(n^\omega)$

- Hashing-based: time $O(n^{2.5})$ [Carmosino–Gao–Impagliazzo–Mihajlin–Paturi–Schneider 2017]
- More structural / real inputs: time $O(n^{2.66})$ [Chan–Vassilevska Williams–Xu 2023]

(Co-)nondeterministic APSP

- Does there even exist a prediction with a subcubic algorithm if $\eta = 0$ ($\hat{D} = D$)?
- Yes! Co-nondeterministic algorithms for APSP



- "NP part"
- easy, shortest path trees

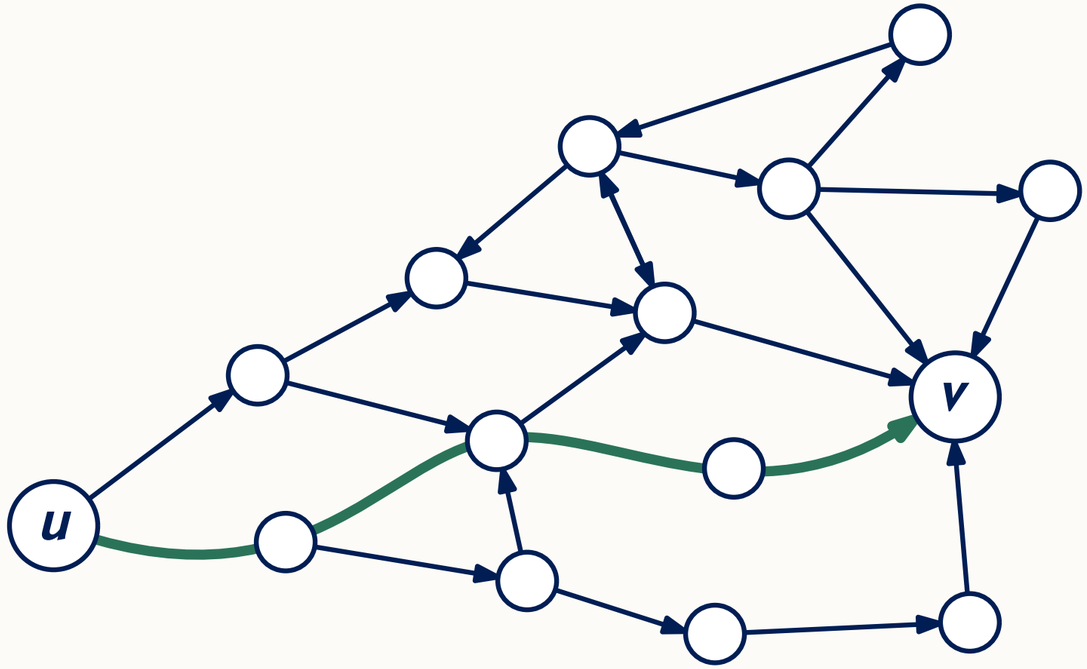
- "co-NP part"
- difficult

2 co-nondeterministic algorithms known:

$2 \le \omega < 2.372$ is matrix multiplication exponent.
Assume $\omega = 2$. MM in time $O(n^\omega)$

- Hashing-based: time $O(n^{2.5})$ [Carmosino–Gao–Impagliazzo–Mihajlin–Paturi–Schneider 2017]
- More structural / real inputs: time $O(n^{2.66})$ [Chan–Vassilevska Williams–Xu 2023]

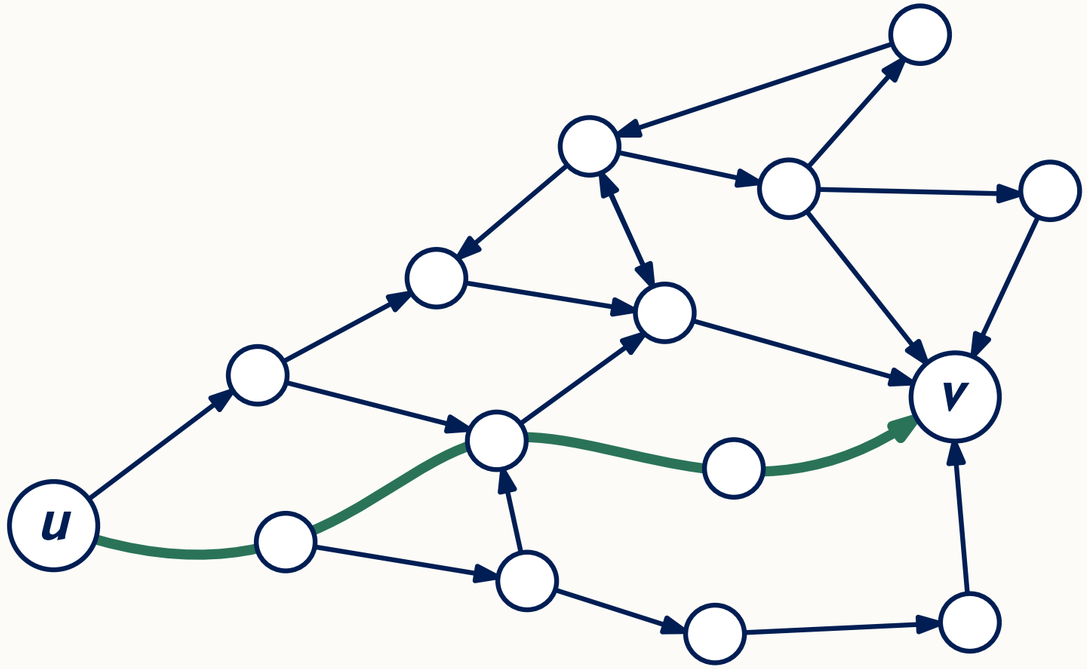
Our Result (Informal)



Our Result (Informal)

Certificate / Prediction C: (informal)

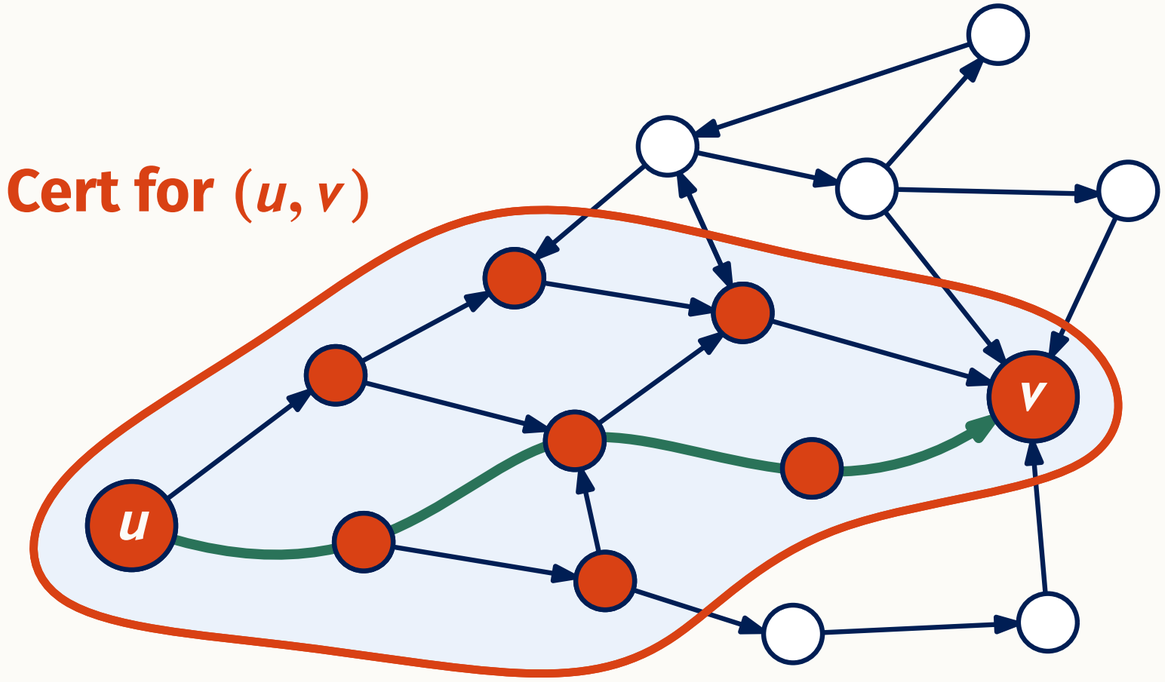
For every vertex pair (u, v), vertices **causing the shortest detour** on a u-v-path.



Our Result (Informal)

Certificate / Prediction C: (informal)

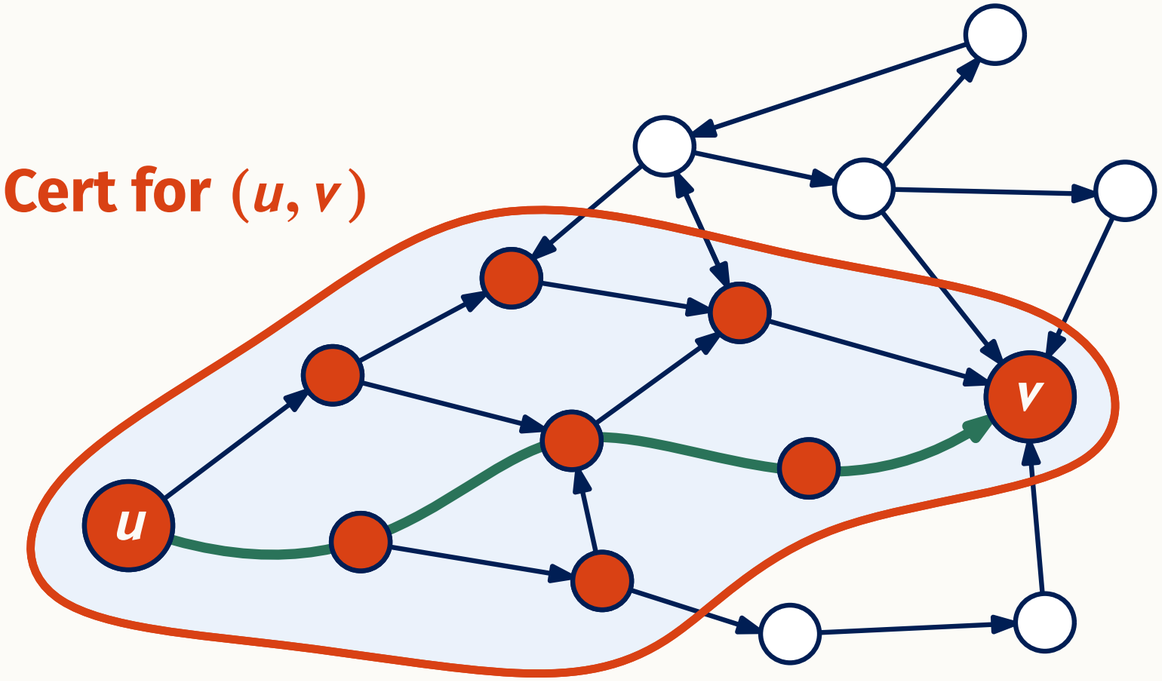
For every vertex pair (u, v), vertices **causing the shortest detour** on a u-v-path.



Our Result (Informal)

Certificate / Prediction C: (informal)

For every vertex pair (u, v), vertices **causing the shortest detour** on a u-v-path.



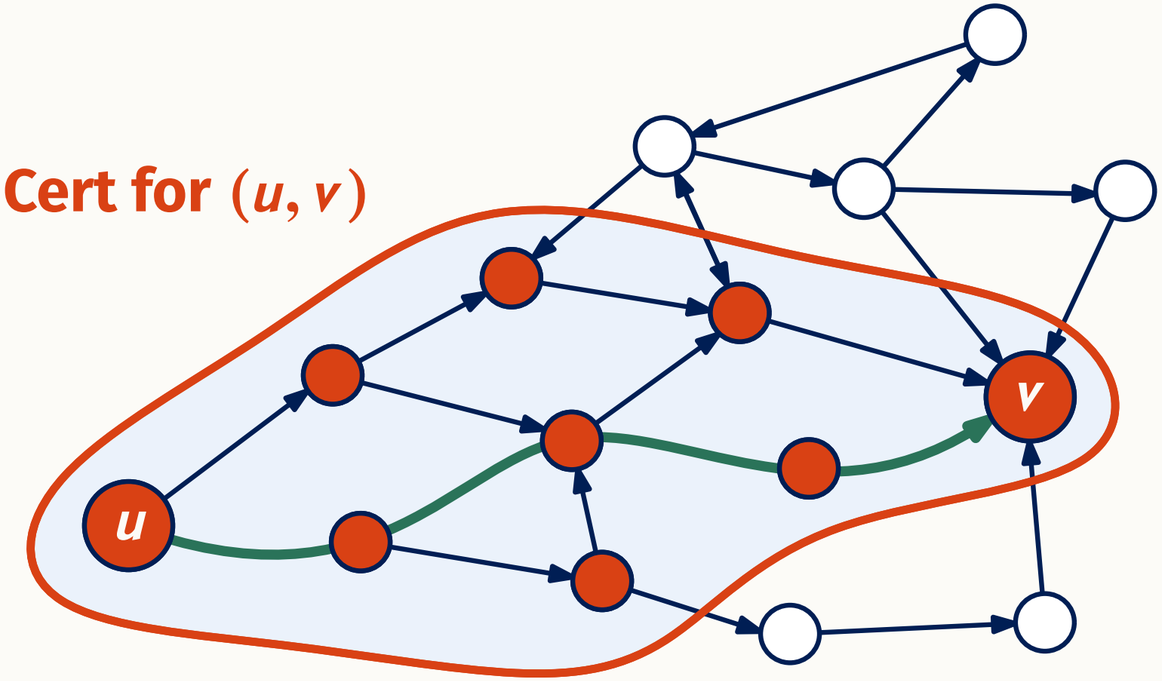
Error Measure η: (informal)

The number of (u, v) where a **very short detour is missing** or **cannot deduce dist(u, v)** from C.

Our Result (Informal)

Certificate / Prediction C: (informal)

For every vertex pair (u, v), vertices **causing the shortest detour** on a u-v-path.



Error Measure η: (informal)

The number of (u, v) where a **very short detour is missing** or **cannot deduce dist(u, v)** from C.

Theorem

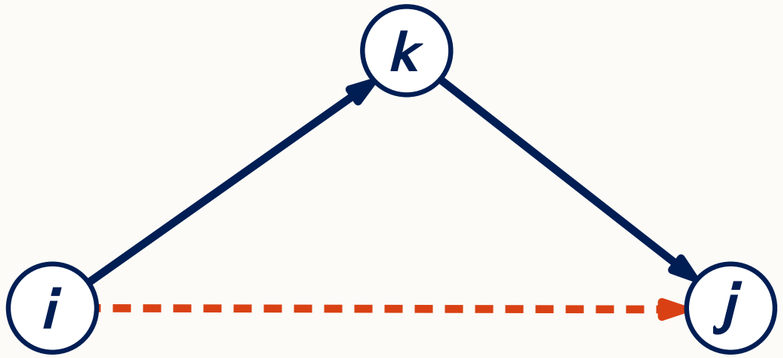
Given G and C, we can compute D in time $\tilde{O}(n\eta + n^{2.75})$.

The Verifier

Reminder: Relaxations

Reminder: Relaxations

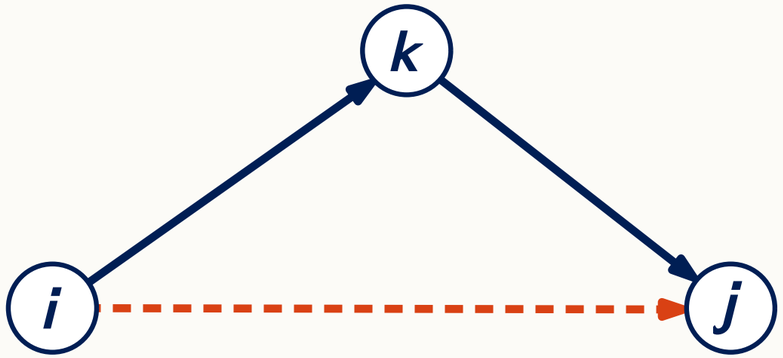
Relaxation of (i, j) with k:



$$D[i,j] = \min\{ D[i,j],\ D[i,k] + D[k,j] \}$$

Reminder: Relaxations

Relaxation of (i, j) with k:

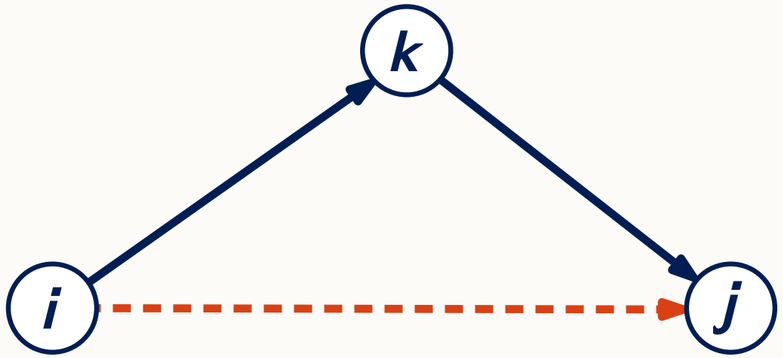


$$D[i,j] = \min\{ D[i,j],\ D[i,k] + D[k,j] \}$$

- n^3 possible relaxations

Reminder: Relaxations

Relaxation of (i, j) with k:



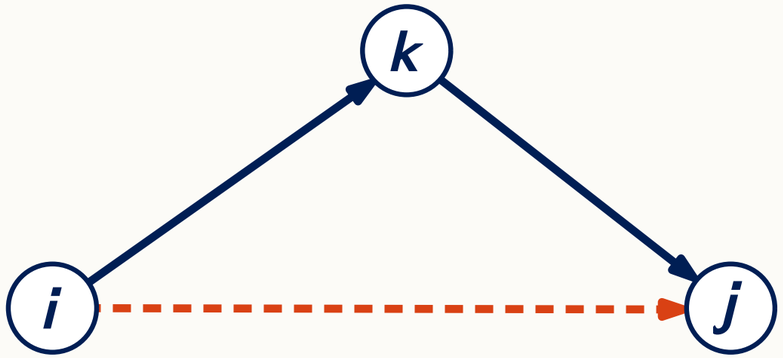
$$D[i,j] = \min\{ D[i,j],\ D[i,k] + D[k,j] \}$$

- n^3 possible relaxations
- D contains correct distances $\iff \nexists$ decreasing relaxation

(Assuming D contains only path lengths $\leq$ direct edge weight)

Reminder: Relaxations

Relaxation of (i, j) with k:



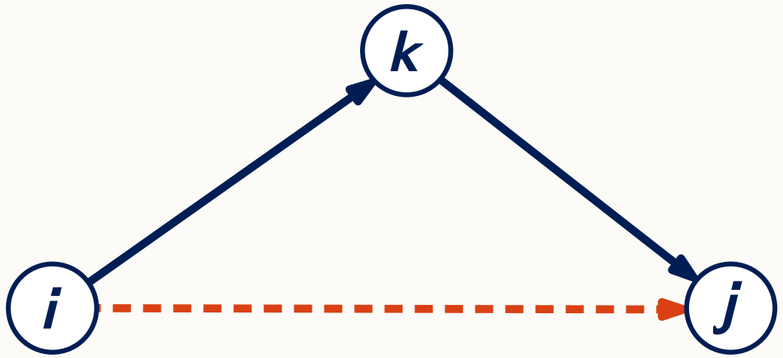
$$D[i,j] = \min\{ D[i,j],\ D[i,k] + D[k,j] \}$$

- n^3 possible relaxations
- D contains correct distances $\iff \nexists$ decreasing relaxation
- A = adjacency matrix (with edge weights)

(Assuming D contains only path lengths $\leq$ direct edge weight)

Reminder: Relaxations

Relaxation of (i, j) with k:



$$D[i,j] = \min\{ D[i,j],\ D[i,k] + D[k,j] \}$$

- n^3 possible relaxations
- D contains correct distances $\iff \nexists$ decreasing relaxation
- A = adjacency matrix (with edge weights)

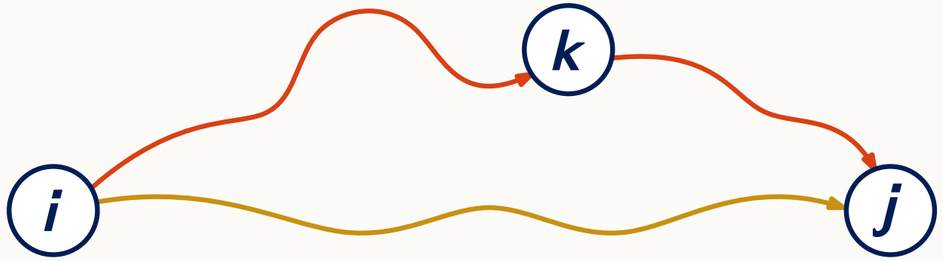
(Assuming D contains only path lengths $\leq$ direct edge weight)

Theorem (Floyd–Warshall) [Floyd '62] [Warshall '62]

$\exists$ permutation of all n^3 relaxations s.t., after performing them once to A, it is the correct distance matrix.

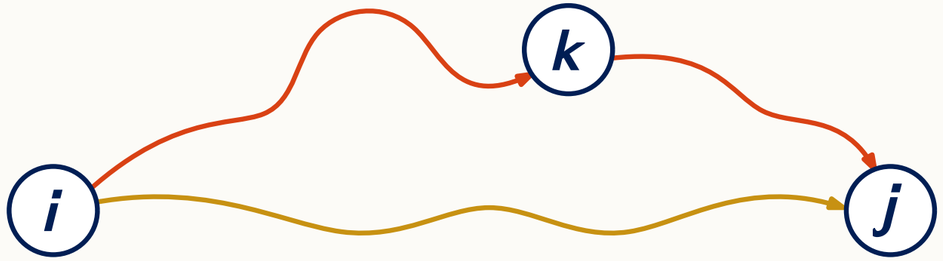
Sketch: APSP Verifier [CVWX]

Sketch: APSP Verifier [CVWX]



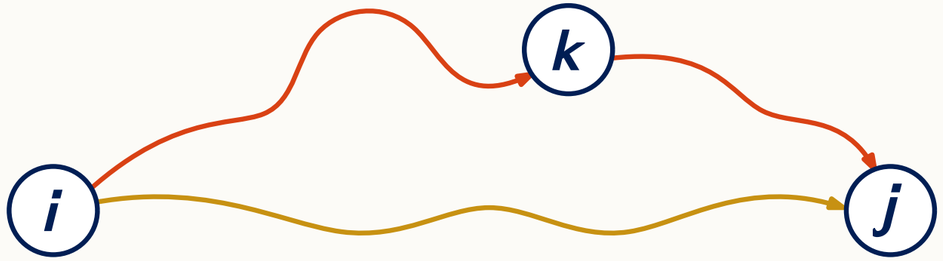
- Define the **detour of i and j via k** as $\textbf{detour}_{i,j}(k) := D[i,k] + D[k,j] - D[i,j]$

Sketch: APSP Verifier [CVWX]



- Define the **detour of i and j via k** as $\textbf{detour}_{i,j}(k) := D[i,k] + D[k,j] - D[i,j]$
- **Intuition:** Small detour $\Longrightarrow$ k close to shortest path, good relaxation

Sketch: APSP Verifier [CVWX]

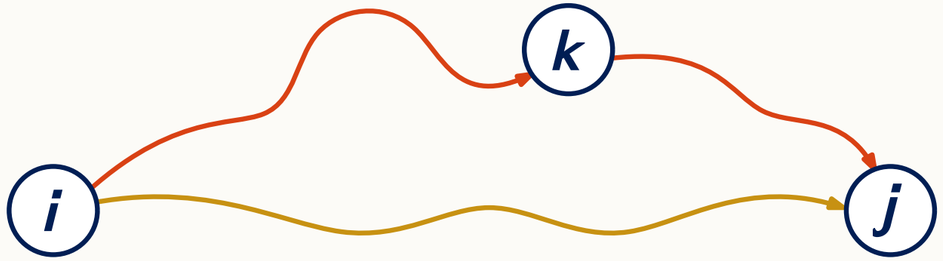


- Define the **detour of i and j via k** as $\textbf{detour}_{i,j}(k) := D[i,k] + D[k,j] - D[i,j]$
- **Intuition:** Small detour $\Longrightarrow$ k close to shortest path, good relaxation

Certificate:

A matrix C where $C[i,j] \subseteq V$ is the set of the $n^{0.75}$ best **detour vertices**.

Sketch: APSP Verifier [CVWX]



- Define the **detour of i and j via k** as $\textbf{detour}_{i,j}(k) := D[i,k] + D[k,j] - D[i,j]$
- **Intuition:** Small detour $\Longrightarrow$ k close to shortest path, good relaxation

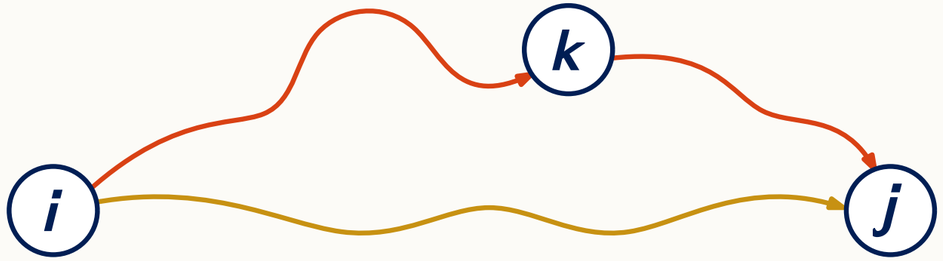
Certificate:

A matrix C where $C[i,j] \subseteq V$ is the set of the $n^{0.75}$ best **detour vertices**.

Claim:

Can verify distance matrix D in time $\tilde{O}(n^{2.75})$, given correct certificate C.

Sketch: APSP Verifier [CVWX]



- Define the **detour of i and j via k** as $\mathbf{detour}_{i,j}(k) := D[i,k] + D[k,j] - D[i,j]$
- **Intuition:** Small detour $\Longrightarrow$ k close to shortest path, good relaxation

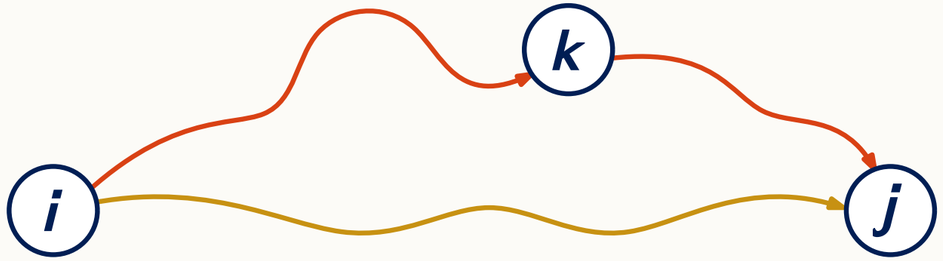
Certificate:

A matrix C where $C[i,j] \subseteq V$ is the set of the $n^{0.75}$ best **detour vertices**.

Claim:

Can verify distance matrix D in time $\tilde{O}(n^{2.75})$, given correct certificate C.

Sketch: APSP Verifier [CVWX]



- Define the **detour of i and j via k** as $\mathbf{detour}_{i,j}(k) := D[i,k] + D[k,j] - D[i,j]$
- **Intuition:** Small detour $\Longrightarrow$ k close to shortest path, good relaxation

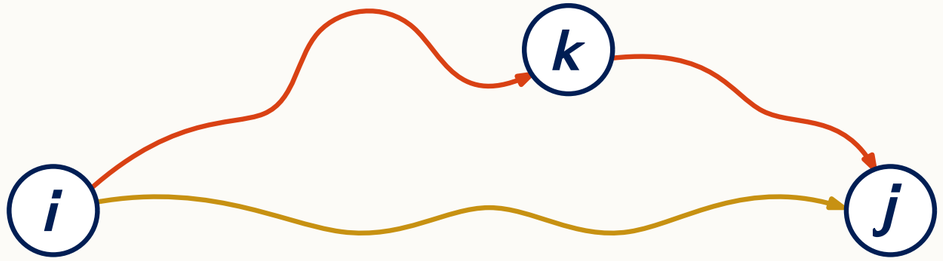
Certificate:

A matrix C where $C[i,j] \subseteq V$ is the set of the $n^{0.75}$ best **detour vertices**.

Claim:

Can verify distance matrix D in time $\tilde{O}(n^{2.75})$, given correct certificate C.

Sketch: APSP Verifier [CVWX]



- Define the **detour of i and j via k** as $\textbf{detour}_{i,j}(k) := D[i,k] + D[k,j] - D[i,j]$
- **Intuition:** Small detour $\Longrightarrow$ k close to shortest path, good relaxation

Certificate:

A matrix C where $C[i,j] \subseteq V$ is the set of the $n^{0.75}$ best **detour vertices**.

Claim:

Can verify distance matrix D in time $\tilde{O}(n^{2.75})$, given correct certificate C.

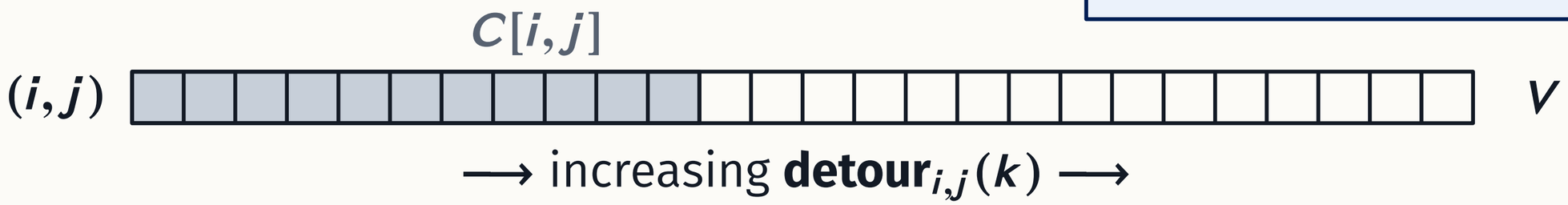
- **Step 1:** Verify that there is no decreasing relaxation in $C[i,j]$

Sketch: APSP Verifier [CVWX]

$\textbf{detour}_{i,j}(k) := D[i,k] + D[k,j] - D[i,j]$

Certificate:

A matrix C where $C[i,j] \subseteq V$ is the set of the $n^{0.75}$ best **detour vertices**.

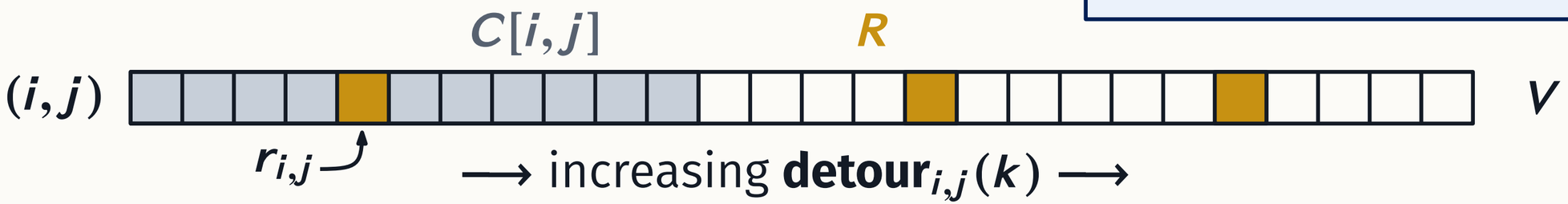


$\textbf{detour}_{i,j}(k) := D[i,k] + D[k,j] - D[i,j]$

Sketch: APSP Verifier [CVWX]

Certificate:

A matrix C where $C[i,j] \subseteq V$ is the set of the $n^{0.75}$ best **detour vertices**.



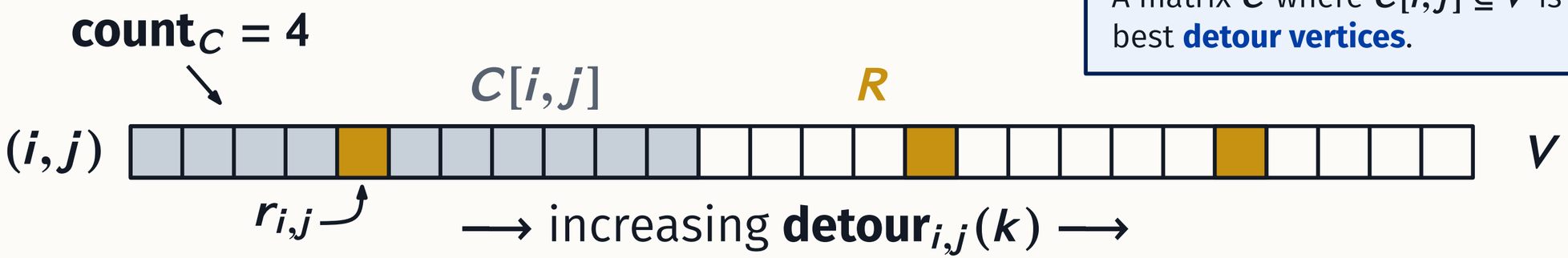
- Let $R \subseteq V$ be a **random hitting set for all** $C[i,j]$ with $r_{i,j} \in R \cap C[i,j]$

$\mathbf{detour}_{i,j}(k) := D[i,k] + D[k,j] - D[i,j]$

Sketch: APSP Verifier [CVWX]

Certificate:

A matrix C where $C[i,j] \subseteq V$ is the set of the $n^{0.75}$ best **detour vertices**.



- Let $R \subseteq V$ be a **random hitting set for all** $C[i,j]$ with $r_{i,j} \in R \cap C[i,j]$
- For every (i,j), compute:

$$\mathbf{count}_C = \#\{k \in C[i,j] : \mathbf{detour}_{i,j}(k) < \mathbf{detour}_{i,j}(r_{i,j})\}$$

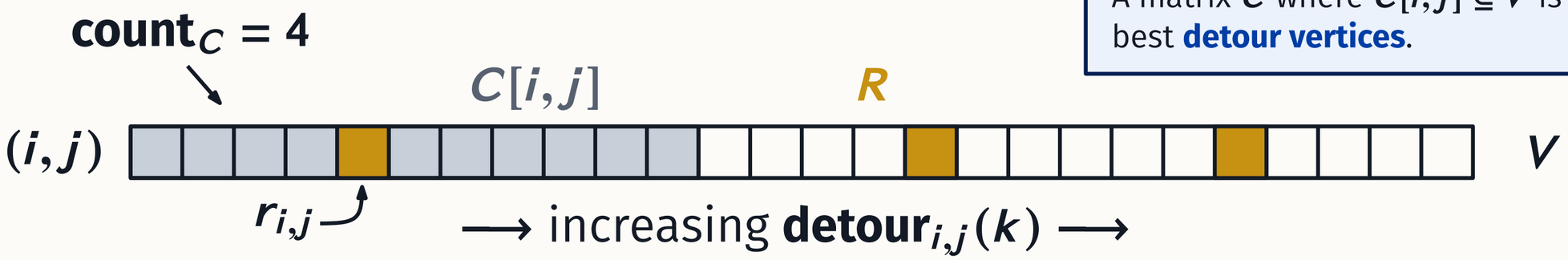
$$\mathbf{count}_V = \#\{k \in V : \mathbf{detour}_{i,j}(k) < \mathbf{detour}_{i,j}(r_{i,j})\}$$

$\mathbf{detour}_{i,j}(k) := D[i,k] + D[k,j] - D[i,j]$

Sketch: APSP Verifier [CVWX]

Certificate:

A matrix C where $C[i,j] \subseteq V$ is the set of the $n^{0.75}$ best **detour vertices**.



- Let $R \subseteq V$ be a **random hitting set for all** $C[i,j]$ with $r_{i,j} \in R \cap C[i,j]$
- For every (i,j), compute:

$$\mathbf{count}_C = \#\{k \in C[i,j] : \mathbf{detour}_{i,j}(k) < \mathbf{detour}_{i,j}(r_{i,j})\}$$
$$\mathbf{count}_V = \#\{k \in V : \mathbf{detour}_{i,j}(k) < \mathbf{detour}_{i,j}(r_{i,j})\}$$

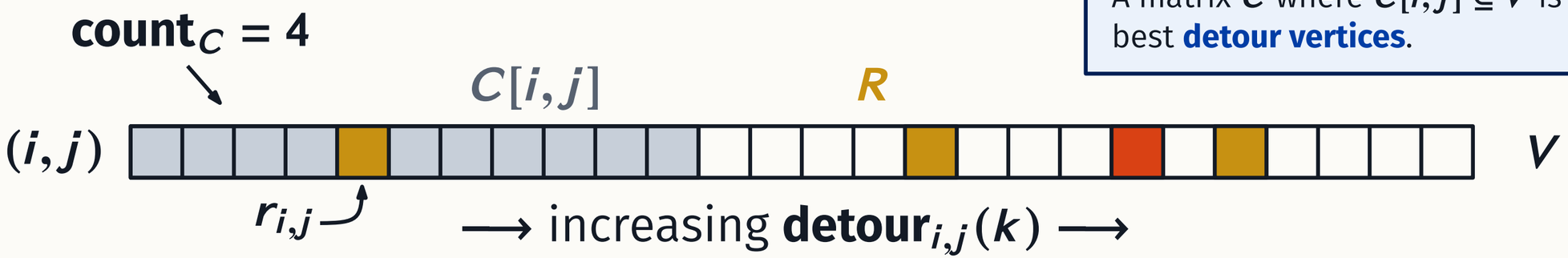
- If $\mathbf{count}_C = \mathbf{count}_V$ for all (i,j), then no entry can be relaxed $\Longrightarrow$ D correct

$\mathbf{detour}_{i,j}(k) := D[i,k] + D[k,j] - D[i,j]$

Sketch: APSP Verifier [CVWX]

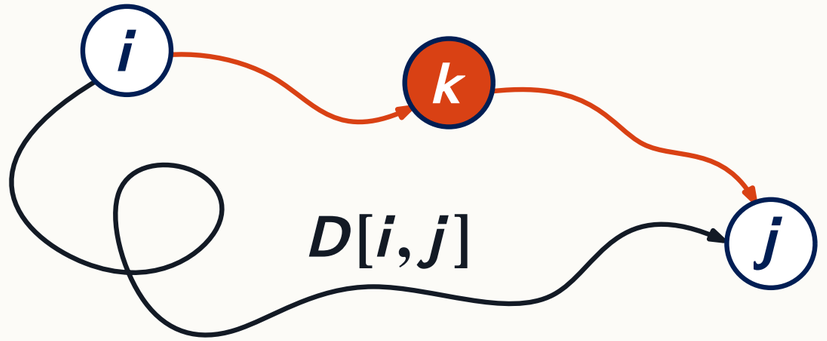
Certificate:

A matrix C where $C[i,j] \subseteq V$ is the set of the $n^{0.75}$ best **detour vertices**.



- Let $R \subseteq V$ be a **random hitting set for all** $C[i,j]$ with $r_{i,j} \in R \cap C[i,j]$
- For every (i,j), compute:

$$\mathbf{count}_C = \#\{k \in C[i,j] : \mathbf{detour}_{i,j}(k) < \mathbf{detour}_{i,j}(r_{i,j})\}$$
$$\mathbf{count}_V = \#\{k \in V : \mathbf{detour}_{i,j}(k) < \mathbf{detour}_{i,j}(r_{i,j})\}$$



- If $\mathbf{count}_C = \mathbf{count}_V$ for all (i,j), then no entry can be relaxed $\Longrightarrow$ D correct
- If $\mathbf{count}_C < \mathbf{count}_V$ for some (i,j), there might be a **decreasing relaxation** somewhere

Sketch: APSP Verifier [CVWX]

$$\textbf{detour}_{i,j}(k) := D[i,k] + D[k,j] - D[i,j]$$

Certificate:

A matrix C where $C[i,j] \subseteq V$ is the set of the $n^{0.75}$ best **detour vertices**.

R = hitting set of size $\tilde{O}(n^{0.25})$

$$\textbf{count}_C = \#\{k \in C[i,j] : \textbf{detour}_{i,j}(k) < \textbf{detour}_{i,j}(r_{i,j})\}$$

$$\textbf{count}_V = \#\{k \in V : \textbf{detour}_{i,j}(k) < \textbf{detour}_{i,j}(r_{i,j})\}$$

Sketch: APSP Verifier [CVWX]

- Computing $\mathbf{count}_C$ is easy in time $|C| = n^{2.75}$

$\mathbf{detour}_{i,j}(k) := D[i,k] + D[k,j] - D[i,j]$

Certificate:

A matrix C where $C[i,j] \subseteq V$ is the set of the $n^{0.75}$ best **detour vertices**.

R = hitting set of size $\tilde{O}(n^{0.25})$

$\mathbf{count}_C = \#\{k \in C[i,j] : \mathbf{detour}_{i,j}(k) < \mathbf{detour}_{i,j}(r_{i,j})\}$

$\mathbf{count}_V = \#\{k \in V : \mathbf{detour}_{i,j}(k) < \mathbf{detour}_{i,j}(r_{i,j})\}$

Sketch: APSP Verifier [CVWX]

- Computing $\mathbf{count}_C$ is easy in time $|C| = n^{2.75}$
- How to compute $\mathbf{count}_V$?

$\mathbf{detour}_{i,j}(k) := D[i,k] + D[k,j] - D[i,j]$

Certificate:

A matrix C where $C[i,j] \subseteq V$ is the set of the $n^{0.75}$ best **detour vertices**.

R = hitting set of size $\tilde{O}(n^{0.25})$

$\mathbf{count}_C = \#\{k \in C[i,j] : \mathbf{detour}_{i,j}(k) < \mathbf{detour}_{i,j}(r_{i,j})\}$

$\mathbf{count}_V = \#\{k \in V : \mathbf{detour}_{i,j}(k) < \mathbf{detour}_{i,j}(r_{i,j})\}$

Sketch: APSP Verifier [CVWX]

- Computing $\mathbf{count}_C$ is easy in time $|C| = n^{2.75}$
- How to compute $\mathbf{count}_V$?
- For every $r \in R$ compute $\mathbf{count}_V$ **for all** (i, j) **with** $r_{i,j} = r$:

$$\mathbf{detour}_{i,j}(k) := D[i,k] + D[k,j] - D[i,j]$$

Certificate:

A matrix C where $C[i,j] \subseteq V$ is the set of the $n^{0.75}$ best **detour vertices**.

R = hitting set of size $\tilde{O}(n^{0.25})$

$$\mathbf{count}_C = \#\{k \in C[i,j] : \mathbf{detour}_{i,j}(k) < \mathbf{detour}_{i,j}(r_{i,j})\}$$

$$\mathbf{count}_V = \#\{k \in V : \mathbf{detour}_{i,j}(k) < \mathbf{detour}_{i,j}(r_{i,j})\}$$

Sketch: APSP Verifier [CVWX]

$\mathbf{detour}_{i,j}(k) := D[i,k] + D[k,j] - D[i,j]$

Certificate:

A matrix C where $C[i,j] \subseteq V$ is the set of the $n^{0.75}$ best **detour vertices**.

R = hitting set of size $\tilde{O}(n^{0.25})$

$\mathbf{count}_C = \#\{k \in C[i,j] : \mathbf{detour}_{i,j}(k) < \mathbf{detour}_{i,j}(r_{i,j})\}$

$\mathbf{count}_V = \#\{k \in V : \mathbf{detour}_{i,j}(k) < \mathbf{detour}_{i,j}(r_{i,j})\}$

- Computing $\mathbf{count}_C$ is easy in time $|C| = n^{2.75}$
- How to compute $\mathbf{count}_V$?
- For every $r \in R$ compute $\mathbf{count}_V$ **for all** (i,j) **with** $r_{i,j} = r$:

$$\mathbf{count}_V = \#\{k \in V : \mathbf{detour}_{i,j}(k) < \mathbf{detour}_{i,j}(r_{i,j})\}$$

Sketch: APSP Verifier [CVWX]

$\mathbf{detour}_{i,j}(k) := D[i,k] + D[k,j] - D[i,j]$

Certificate:

A matrix C where $C[i,j] \subseteq V$ is the set of the $n^{0.75}$ best **detour vertices**.

R = hitting set of size $\tilde{O}(n^{0.25})$

$\mathbf{count}_C = \#\{k \in C[i,j] : \mathbf{detour}_{i,j}(k) < \mathbf{detour}_{i,j}(r_{i,j})\}$

$\mathbf{count}_V = \#\{k \in V : \mathbf{detour}_{i,j}(k) < \mathbf{detour}_{i,j}(r_{i,j})\}$

- Computing $\mathbf{count}_C$ is easy in time $|C| = n^{2.75}$
- How to compute $\mathbf{count}_V$?
- For every $r \in R$ compute $\mathbf{count}_V$ **for all (i,j) with $r_{i,j} = r$**:

$$\begin{aligned}\mathbf{count}_V &= \#\{k \in V : \mathbf{detour}_{i,j}(k) < \mathbf{detour}_{i,j}(r_{i,j})\} \\ &= \#\{k \in V : D[i,k] + D[k,j] < D[i,r_{i,j}] + D[r_{i,j},j]\}\end{aligned}$$

Sketch: APSP Verifier [CVWX]

$\mathbf{detour}_{i,j}(k) := D[i,k] + D[k,j] - D[i,j]$

Certificate:

A matrix C where $C[i,j] \subseteq V$ is the set of the $n^{0.75}$ best **detour vertices**.

R = hitting set of size $\tilde{O}(n^{0.25})$

$\mathbf{count}_C = \#\{k \in C[i,j] : \mathbf{detour}_{i,j}(k) < \mathbf{detour}_{i,j}(r_{i,j})\}$

$\mathbf{count}_V = \#\{k \in V : \mathbf{detour}_{i,j}(k) < \mathbf{detour}_{i,j}(r_{i,j})\}$

- Computing $\mathbf{count}_C$ is easy in time $|C| = n^{2.75}$
- How to compute $\mathbf{count}_V$?
- For every $r \in R$ compute $\mathbf{count}_V$ **for all (i,j) with $r_{i,j} = r$**:

$$\begin{aligned}\mathbf{count}_V &= \#\{k \in V : \mathbf{detour}_{i,j}(k) < \mathbf{detour}_{i,j}(r_{i,j})\} \\ &= \#\{k \in V : D[i,k] + D[k,j] < D[i,r_{i,j}] + D[r_{i,j},j]\} \\ &= \#\{k \in V : D[i,k] - D[i,r_{i,j}] < D[r_{i,j},j] - D[k,j]\}\end{aligned}$$

Sketch: APSP Verifier [CVWX]

$\mathbf{detour}_{i,j}(k) := D[i,k] + D[k,j] - D[i,j]$

Certificate:

A matrix C where $C[i,j] \subseteq V$ is the set of the $n^{0.75}$ best **detour vertices**.

R = hitting set of size $\tilde{O}(n^{0.25})$

$\mathbf{count}_C = \#\{k \in C[i,j] : \mathbf{detour}_{i,j}(k) < \mathbf{detour}_{i,j}(r_{i,j})\}$

$\mathbf{count}_V = \#\{k \in V : \mathbf{detour}_{i,j}(k) < \mathbf{detour}_{i,j}(r_{i,j})\}$

- Computing $\mathbf{count}_C$ is easy in time $|C| = n^{2.75}$
- How to compute $\mathbf{count}_V$?
- For every $r \in R$ compute $\mathbf{count}_V$ **for all** (i,j) **with** $r_{i,j} = r$:

$$\begin{aligned}
\mathbf{count}_V &= \#\{k \in V : \mathbf{detour}_{i,j}(k) < \mathbf{detour}_{i,j}(r_{i,j})\} \\
&= \#\{k \in V : D[i,k] + D[k,j] < D[i,r_{i,j}] + D[r_{i,j},j]\} \\
&= \#\{k \in V : \underbrace{D[i,k] - D[i,r_{i,j}]}_{A[i,k]} < \underbrace{D[r_{i,j},j] - D[k,j]}_{B[k,j]}\}
\end{aligned}$$

$\mathbf{detour}_{i,j}(k) := D[i,k] + D[k,j] - D[i,j]$

Sketch: APSP Verifier [CVWX]

Certificate:

A matrix C where $C[i,j] \subseteq V$ is the set of the $n^{0.75}$ best **detour vertices**.

R = hitting set of size $\tilde{O}(n^{0.25})$

$\mathbf{count}_C = \#\{k \in C[i,j] : \mathbf{detour}_{i,j}(k) < \mathbf{detour}_{i,j}(r_{i,j})\}$

$\mathbf{count}_V = \#\{k \in V : \mathbf{detour}_{i,j}(k) < \mathbf{detour}_{i,j}(r_{i,j})\}$

- Computing $\mathbf{count}_C$ is easy in time $|C| = n^{2.75}$
- How to compute $\mathbf{count}_V$?
- For every $r \in R$ compute $\mathbf{count}_V$ **for all (i,j) with $r_{i,j} = r$**:

$$\begin{aligned}
\mathbf{count}_V &= \#\{k \in V : \mathbf{detour}_{i,j}(k) < \mathbf{detour}_{i,j}(r_{i,j})\} \\
&= \#\{k \in V : D[i,k] + D[k,j] < D[i,r_{i,j}] + D[r_{i,j},j]\} \\
&= \#\{k \in V : \underbrace{D[i,k] - D[i,r_{i,j}]}_{A[i,k]} < \underbrace{D[r_{i,j},j] - D[k,j]}_{B[k,j]}\}
\end{aligned}$$

"Fredman's trick"

Dominance product

- After removing addition, can compute in time $\tilde{O}(n^{(\omega+3)/2}) = \tilde{O}(n^{2.5})$ [Matoušek 1991]

Sketch: APSP Verifier [CVWX]

$\mathbf{detour}_{i,j}(k) := D[i,k] + D[k,j] - D[i,j]$

Certificate:

A matrix C where $C[i,j] \subseteq V$ is the set of the $n^{0.75}$ best **detour vertices**.

R = hitting set of size $\tilde{O}(n^{0.25})$

$\mathbf{count}_C = \#\{k \in C[i,j] : \mathbf{detour}_{i,j}(k) < \mathbf{detour}_{i,j}(r_{i,j})\}$

$\mathbf{count}_V = \#\{k \in V : \mathbf{detour}_{i,j}(k) < \mathbf{detour}_{i,j}(r_{i,j})\}$

- Computing $\mathbf{count}_C$ is easy in time $|C| = n^{2.75}$
- How to compute $\mathbf{count}_V$?
- For every $r \in R$ compute $\mathbf{count}_V$ **for all (i,j) with $r_{i,j} = r$**:

$$\begin{aligned}
\mathbf{count}_V &= \#\{k \in V : \mathbf{detour}_{i,j}(k) < \mathbf{detour}_{i,j}(r_{i,j})\} \\
&= \#\{k \in V : D[i,k] + D[k,j] < D[i,r_{i,j}] + D[r_{i,j},j]\} \\
&= \#\{k \in V : \underbrace{D[i,k] - D[i,r_{i,j}]}_{A[i,k]} < \underbrace{D[r_{i,j},j] - D[k,j]}_{B[k,j]}\}
\end{aligned}$$

"Fredman's trick"

Dominance product

- After removing addition, can compute in time $\tilde{O}(n^{(\omega+3)/2}) = \tilde{O}(n^{2.5})$ [Matoušek 1991]
- Over all $r \in R$ it takes $|R| \cdot \tilde{O}(n^{2.5}) = \tilde{O}(n^{2.75})$

Turning the Verifier Into a Learning-Augmented Algorithm

Turning the Verifier Into an ALP

Turning the Verifier Into an ALP

- Use the certificate C as our prediction

Prediction:

A matrix C where $C[i, j] \subseteq V$ **should contain** the $n^{0.75}$ best **detour vertices**.

Turning the Verifier Into an ALP

- Use the certificate C as our prediction

3 steps:

Prediction:

A matrix C where $C[i, j] \subseteq V$ **should contain** the $n^{0.75}$ best **detour vertices**.

Turning the Verifier Into an ALP

- Use the certificate C as our prediction

3 steps:

1. **Compute candidate distances**

Prediction:

A matrix C where $C[i, j] \subseteq V$ **should contain** the $n^{0.75}$ best **detour vertices**.

Turning the Verifier Into an ALP

- Use the certificate C as our prediction

Prediction:

A matrix C where $C[i, j] \subseteq V$ **should contain** the $n^{0.75}$ best **detour vertices**.

3 steps:

1. **Compute candidate distances**

Lemma (informal)

In time $\tilde{O}(|C|)$, we can compute "good candidate distances" $\hat{D}$ from C.

in a **good order**!

"perform **all and only** relaxations in C"

Turning the Verifier Into an ALP

- Use the certificate C as our prediction

Prediction:

A matrix C where $C[i, j] \subseteq V$ **should contain** the $n^{0.75}$ best **detour vertices**.

3 steps:

1. **Compute candidate distances**

Lemma (informal)

In time $\tilde{O}(|C|)$, we can compute "good candidate distances" $\hat{D}$ from C.

in a **good order**!

"perform **all and only** relaxations in C"

2. **Run the verifier**

Turning the Verifier Into an ALP

Prediction:

A matrix C where $C[i, j] \subseteq V$ **should contain** the $n^{0.75}$ best **detour vertices**.

- Use the certificate C as our prediction

3 steps:

1. **Compute candidate distances**

Lemma (informal)

In time $\tilde{O}(|C|)$, we can compute "good candidate distances" $\hat{D}$ from C.

in a **good order**!

"perform **all and only** relaxations in C"

2. **Run the verifier**

3. **Fix mistakes**

Turning the Verifier Into an ALP

Prediction:

A matrix C where $C[i,j] \subseteq V$ **should contain** the $n^{0.75}$ best **detour vertices**.

- Use the certificate C as our prediction

3 steps:

1. **Compute candidate distances**

Lemma (informal)

In time $\tilde{O}(|C|)$, we can compute "good candidate distances" $\hat{D}$ from C.

in a **good order**!

"perform **all and only** relaxations in C"

2. **Run the verifier** ← just seen

3. **Fix mistakes** ← **now**

Error Measure

Theorem

Given G and C, we can compute D in time $\tilde{O}(n\eta + n^{2.75})$.

Prediction:

A matrix C where $C[i, j] \subseteq V$ **should contain** the $n^{0.75}$ best **detour vertices**.

Error Measure

Theorem

Given G and C, we can compute D in time $\tilde{O}(n\eta + n^{2.75})$.

Prediction:

A matrix C where $C[i,j] \subseteq V$ **should contain** the $n^{0.75}$ best **detour vertices**.

Error measure:

$$\eta \quad := \quad \#\{\ (i,j)\ :\ \text{verifier failed}\ \} \quad + \quad \#\{\ (i,j)\ :\ \hat{D}[i,j] \text{ incorrect}\ \}$$

Error Measure

Theorem

Given G and C, we can compute D in time $\tilde{O}(n\eta + n^{2.75})$.

Prediction:

A matrix C where $C[i,j] \subseteq V$ **should contain** the $n^{0.75}$ best **detour vertices**.

Error measure:

$$\eta \quad := \quad \#\{\,(i,j) : \text{verifier failed}\,\} \quad + \quad \#\{\,(i,j) : \hat{D}[i,j] \text{ incorrect}\,\}$$

Verifier detects these

Might have incorrect $\hat{D}$

If not all best detours are in $C[i,j]$

Verifier might not detect

Error Measure

Theorem

Given G and C, we can compute D in time $\tilde{O}(n\eta + n^{2.75})$.

Prediction:

A matrix C where $C[i,j] \subseteq V$ **should contain** the $n^{0.75}$ best **detour vertices**.

Error measure:

$$\eta := \#\{\,(i,j) : \text{verifier failed}\,\} + \#\{\,(i,j) : \hat{D}[i,j] \text{ incorrect}\,\}$$

Verifier detects these

Might have incorrect $\hat{D}$

If not all best detours are in $C[i,j]$

Verifier might not detect

Want to perform **all $n \cdot \eta$ relaxations** only once!

Error Measure

Theorem

Given G and C, we can compute D in time $\tilde{O}(n\eta + n^{2.75})$.

Prediction:

A matrix C where $C[i,j] \subseteq V$ **should contain** the $n^{0.75}$ best **detour vertices**.

Error measure:

$$\eta := \#\{ (i,j) : \text{verifier failed} \} + \#\{ (i,j) : \hat{D}[i,j] \text{ incorrect} \}$$

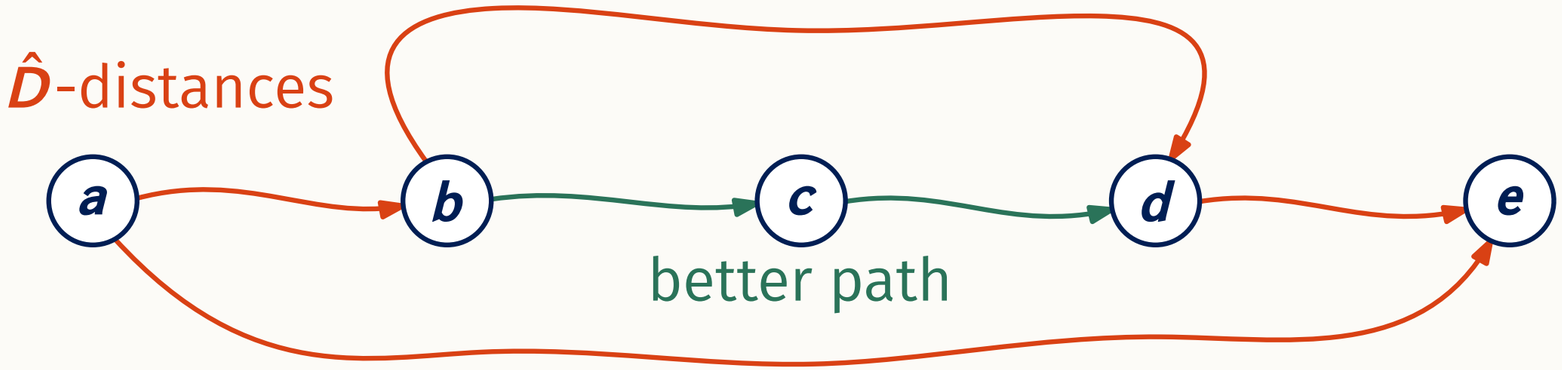
Verifier detects these

Might have incorrect $\hat{D}$

If not all best detours are in $C[i,j]$

Verifier might not detect

Want to perform **all $n \cdot \eta$ relaxations** only once!



Error Measure

Theorem

Given G and C, we can compute D in time $\tilde{O}(n\eta + n^{2.75})$.

Prediction:

A matrix C where $C[i,j] \subseteq V$ **should contain** the $n^{0.75}$ best **detour vertices**.

Error measure:

$$\eta := \#\{\,(i,j) : \text{verifier failed}\,\} + \#\{\,(i,j) : \hat{D}[i,j] \text{ incorrect}\,\}$$

Verifier detects these

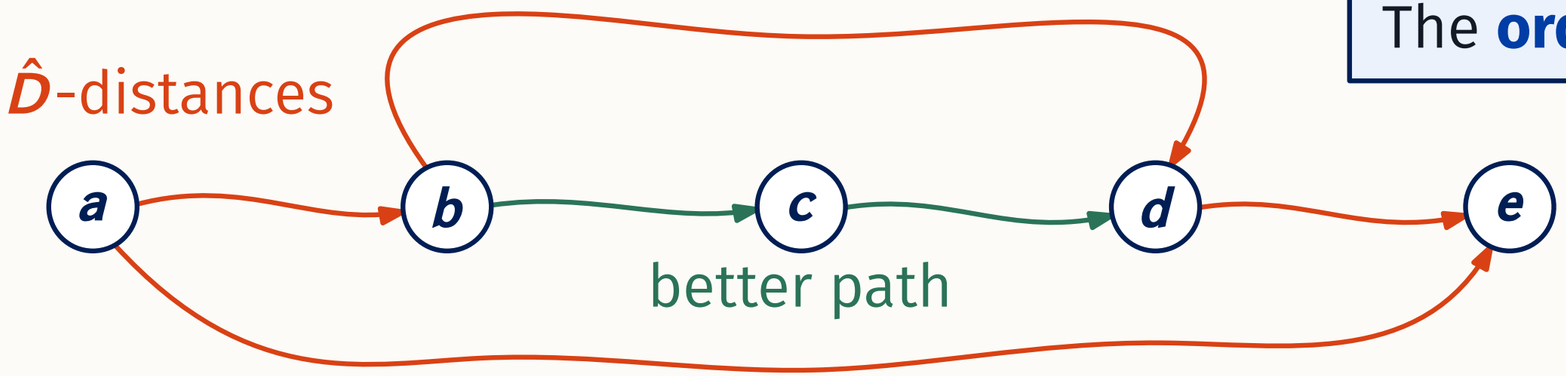
Might have incorrect $\hat{D}$

If not all best detours are in $C[i,j]$

Verifier might not detect

Want to perform **all $n \cdot \eta$ relaxations** only once!

The **order of relaxations** matters!



Fixing: Full Algorithm

Error measure:

$$\eta \quad := \quad \#\{\ (i,j)\ :\ \text{verifier failed}\ \} \quad + \quad \#\{\ (i,j)\ :\ \hat{D}[i,j] \text{ incorrect}\ \}$$

Fixing: Full Algorithm

Error measure:

$$\eta \quad := \quad \#\{\,(i,j) : \text{verifier failed}\,\} \quad + \quad \#\{\,(i,j) : \hat{D}[i,j] \text{ incorrect}\,\}$$

- Assume non-negative edge weights

Fixing: Full Algorithm

Error measure:

$$\eta \quad := \quad \#\{\,(i,j) : \text{verifier failed}\,\} \quad + \quad \#\{\,(i,j) : \hat{D}[i,j] \text{ incorrect}\,\}$$

- Assume non-negative edge weights
- Keep a set of **marked pairs**, initially the set of unverified pairs

Fixing: Full Algorithm

Error measure:

$$\eta \quad := \quad \#\{\,(i,j) : \text{verifier failed}\,\} \quad + \quad \#\{\,(i,j) : \hat{D}[i,j] \text{ incorrect}\,\}$$

- Assume non-negative edge weights
- Keep a set of **marked pairs**, initially the set of unverified pairs
- Consider pairs (i,j) in **increasing order** of $\hat{D}[i,j]$:

Fixing: Full Algorithm

Error measure:

$$\eta \quad := \quad \#\{\,(i,j) : \text{verifier failed}\,\} \quad + \quad \#\{\,(i,j) : \hat{D}[i,j] \text{ incorrect}\,\}$$

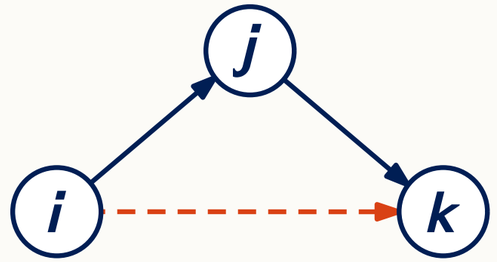
- Assume non-negative edge weights
- Keep a set of **marked pairs**, initially the set of unverified pairs
- Consider pairs (i,j) in **increasing order** of $\hat{D}[i,j]$:
  - If (i,j) is marked:
    - Relax **every pair** with $\hat{D}[i,j]$
    - If changed, mark

Fixing: Full Algorithm

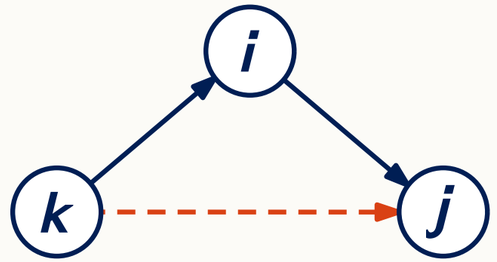
Error measure:

η := $\#\{\ (i,j)\ :$ verifier failed $\}$ + $\#\{\ (i,j)\ :\ \hat{D}[i,j]$ incorrect $\}$

- Assume non-negative edge weights
- Keep a set of **marked pairs**, initially the set of unverified pairs
- Consider pairs (i,j) in **increasing order** of $\hat{D}[i,j]$:
  - If (i,j) is marked:
    - Relax **every pair** with $\hat{D}[i,j]$
    - If changed, mark



$$D[i,k] = \min\{\ D[i,k],\ D[i,j] + D[j,k]\ \}$$



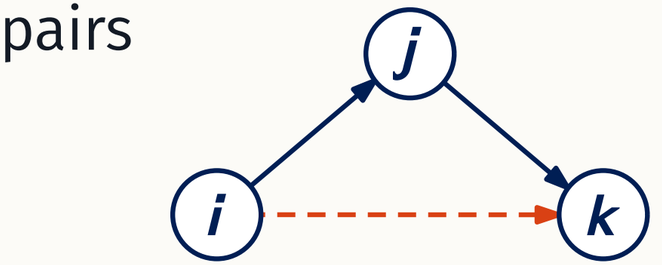
$$D[k,j] = \min\{\ D[k,j],\ D[k,i] + D[i,j]\ \}$$

Fixing: Full Algorithm

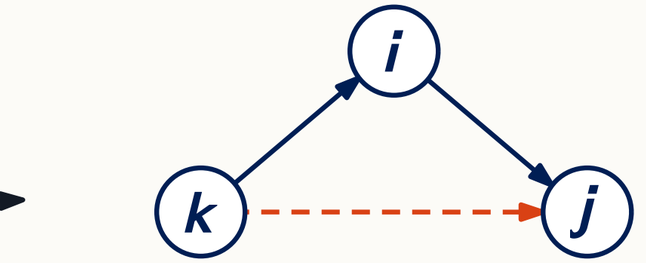
Error measure:

η := #{ (i,j) : verifier failed } + #{ (i,j) : $\hat{D}[i,j]$ incorrect }

- Assume non-negative edge weights
- Keep a set of **marked pairs**, initially the set of unverified pairs
- Consider pairs (i,j) in **increasing order** of $\hat{D}[i,j]$:
  - If (i,j) is marked:
    - Relax **every pair** with $\hat{D}[i,j]$
    - If changed, mark
  - If (i,j) is **not** marked:
    - Relax **every marked pair** with $\hat{D}[i,j]$



$D[i,k] = \min\{ D[i,k],\ D[i,j] + D[j,k] \}$



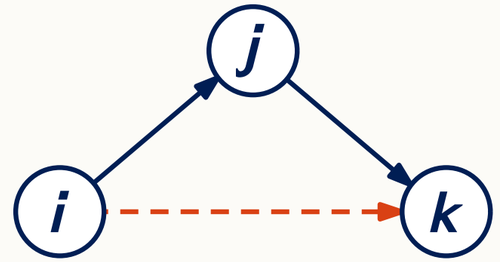
$D[k,j] = \min\{ D[k,j],\ D[k,i] + D[i,j] \}$

Fixing: Full Algorithm

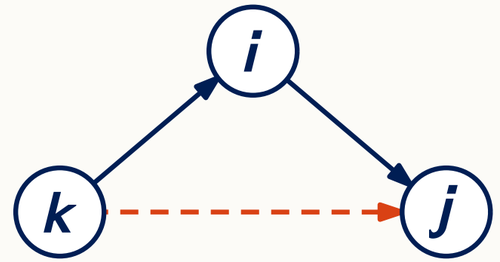
Error measure:

η := #{ (i,j) : verifier failed } + #{ (i,j) : $\hat{D}[i,j]$ incorrect }

- Assume non-negative edge weights
- Keep a set of **marked pairs**, initially the set of unverified pairs
- Consider pairs (i,j) in **increasing order** of $\hat{D}[i,j]$:
  - If (i,j) is marked:
    - Relax **every pair** with $\hat{D}[i,j]$
    - If changed, mark
  - If (i,j) is **not** marked:
    - Relax **every marked pair** with $\hat{D}[i,j]$



$D[i,k] = \min\{ D[i,k],\ D[i,j] + D[j,k] \}$



$D[k,j] = \min\{ D[k,j],\ D[k,i] + D[i,j] \}$

Total # relaxations:

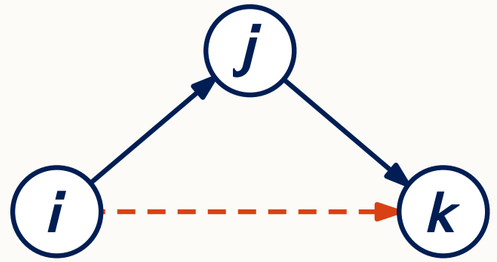
#{ total marked pairs } $\cdot\, O(n)$ = $O(\eta \cdot n)$

Fixing: Full Algorithm

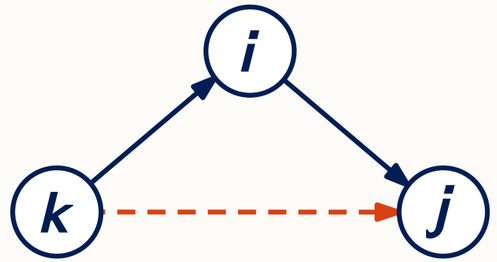
Error measure:

η := #{ (i,j) : verifier failed } + #{ (i,j) : $\hat{D}[i,j]$ incorrect }

- Assume non-negative edge weights
- Keep a set of **marked pairs**, initially the set of unverified pairs
- Consider pairs (i,j) in **increasing order** of $\hat{D}[i,j]$:
  - If (i,j) is marked:
    - Relax **every pair** with $\hat{D}[i,j]$
    - If changed, mark
  - If (i,j) is **not** marked:
    - Relax **every marked pair** with $\hat{D}[i,j]$



$D[i,k] = \min\{ D[i,k],\ D[i,j] + D[j,k] \}$



$D[k,j] = \min\{ D[k,j],\ D[k,i] + D[i,j] \}$

Total # relaxations:

#{ total marked pairs } $\cdot\, O(n) = O(\eta \cdot n)$

Correctness follows from order.

Conclusion

Conclusion

Error measure:

$\eta := \#\{\ (i,j)\ :\ \text{verifier failed}\ \} + \#\{\ (i,j)\ :\ \hat{D}[i,j] \text{ incorrect}\ \}$

Prediction:

A matrix C where $C[i,j] \subseteq V$ **should contain** the $n^{0.75}$ best **detour vertices**.

Conclusion

Error measure:

$\eta := \#\{ (i,j) : \text{verifier failed} \} + \#\{ (i,j) : \hat{D}[i,j] \text{ incorrect} \}$

Prediction:

A matrix C where $C[i,j] \subseteq V$ **should contain** the $n^{0.75}$ best **detour vertices**.

Theorem

Given G and C, we can compute D in time $\tilde{O}(n\eta + n^{2.75})$.

1. **Compute candidate distances**

2. **Run the verifier**

3. **Fix mistakes**

Conclusion

Error measure:

$\eta := \#\{ (i,j) : \text{verifier failed} \} + \#\{ (i,j) : \hat{D}[i,j] \text{ incorrect} \}$

Prediction:

A matrix C where $C[i,j] \subseteq V$ **should contain** the $n^{0.75}$ best **detour vertices**.

Theorem

Given G and C, we can compute D in time $\tilde{O}(n\eta + n^{2.75})$.

1. **Compute candidate distances**

2. **Run the verifier**

3. **Fix mistakes**

Open Questions

Error measure:

$$\eta \quad := \quad \#\{\,(i,j) : \text{verifier failed}\,\} \quad + \quad \#\{\,(i,j) : \hat{D}[i,j] \text{ incorrect}\,\}$$

Open Questions

Error measure:

$$\eta \quad := \quad \#\{\,(i,j) : \text{verifier failed}\,\} \quad + \quad \#\{\,(i,j) : \hat{D}[i,j] \text{ incorrect}\,\}$$

- **Remove $\#\{(i,j) : \hat{D}[i,j]$ incorrect$\}$ from η?**
  - Other term seems necessary to make verifier work...

Open Questions

Error measure:

$$\eta \quad := \quad \#\{\,(i,j) : \text{verifier failed}\,\} \quad + \quad \#\{\,(i,j) : \hat{D}[i,j] \text{ incorrect}\,\}$$

- **Remove $\#\{(i,j) : \hat{D}[i,j]$ incorrect$\}$ from η?**
  - Other term seems necessary to make verifier work...
- **Learn the certificate?**
  - Exact ERM is NP-hard, approximation possible?

Open Questions

Error measure:

$$\eta \quad := \quad \#\{\,(i,j) : \text{verifier failed}\,\} \quad + \quad \#\{\,(i,j) : \hat{D}[i,j] \text{ incorrect}\,\}$$

- **Remove $\#\{(i,j) : \hat{D}[i,j]$ incorrect$\}$ from η?**
  - Other term seems necessary to make verifier work...
- **Learn the certificate?**
  - Exact ERM is NP-hard, approximation possible?
- **Circumvent more lower bounds with predictions?**

Open Questions

Error measure:

$$\eta \quad := \quad \#\{\,(i,j) : \text{verifier failed}\,\} \quad + \quad \#\{\,(i,j) : \hat{D}[i,j] \text{ incorrect}\,\}$$

- **Remove $\#\{(i,j) : \hat{D}[i,j]$ incorrect$\}$ from η?**
  - Other term seems necessary to make verifier work...
- **Learn the certificate?**
  - Exact ERM is NP-hard, approximation possible?
- **Circumvent more lower bounds with predictions?**

Thank you!