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

Adam Polak
Jonas Schmidt

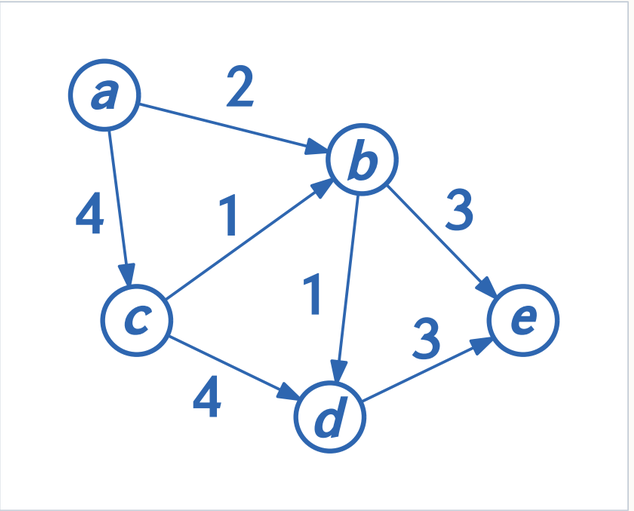
Bocconi University

ESA 2026

All-Pairs Shortest Paths (APSP)

All-Pairs Shortest Paths (APSP)

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

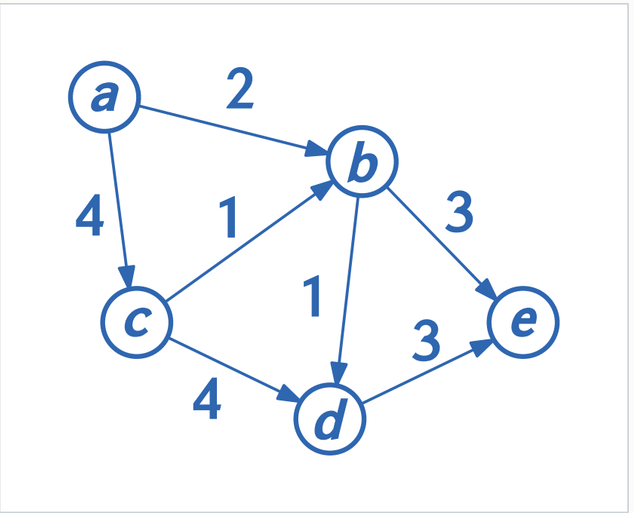


All-Pairs Shortest Paths (APSP)

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)

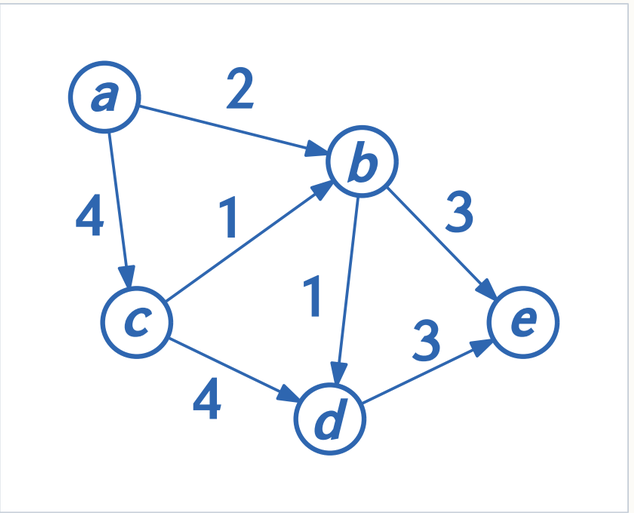
Input: A **directed**, **weighted** graph $G = (V, E, w)$

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)

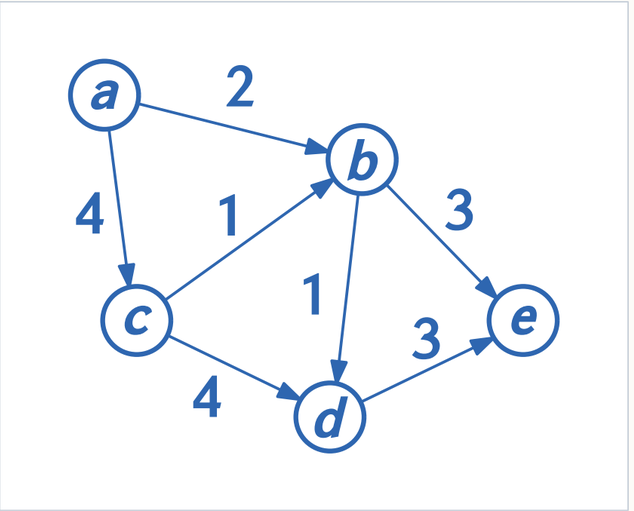
Input: A **directed**, **weighted** graph $G = (V, E, w)$

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 '14]



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)

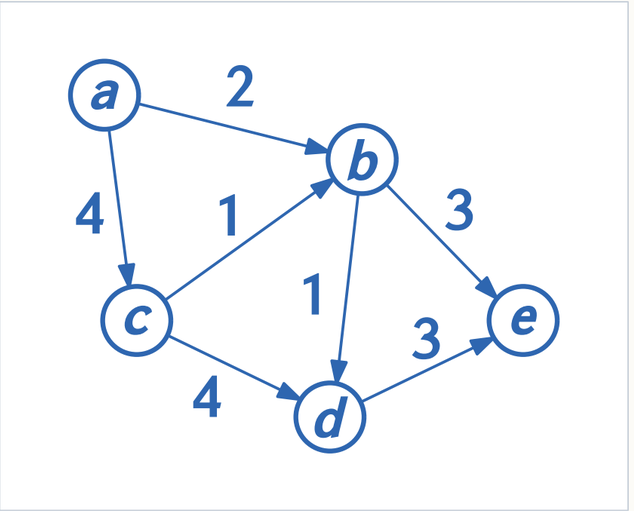
Input: A **directed**, **weighted** graph $G = (V, E, w)$

Output: The distance matrix D of G

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

Complexity:

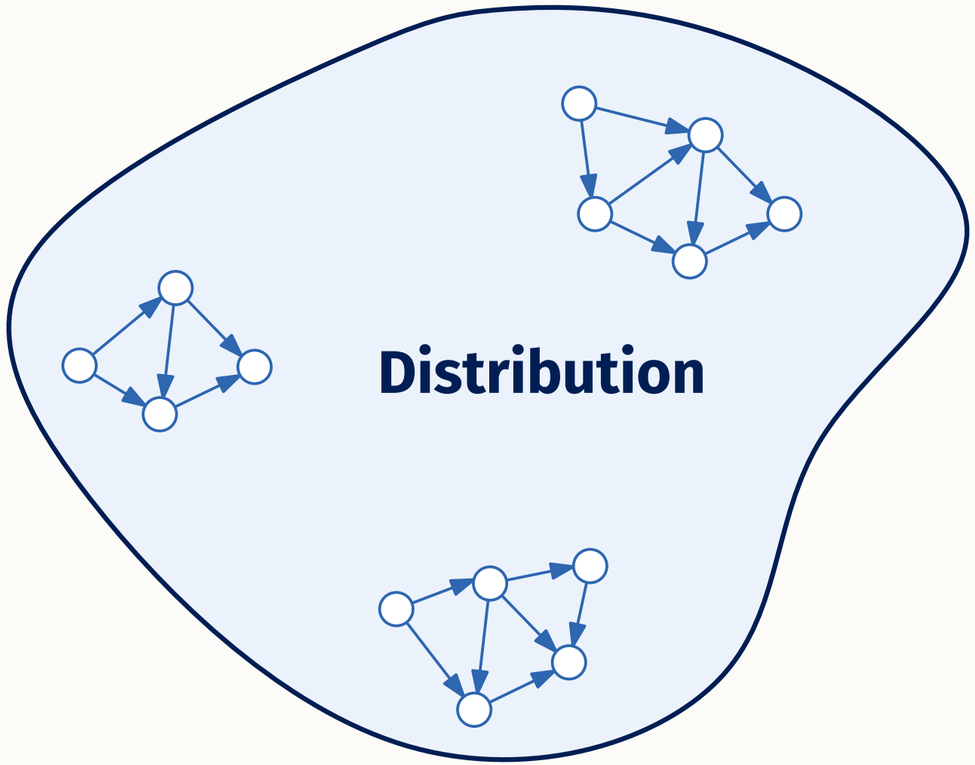
- Floyd–Warshall algorithm: $O(n^3)$ [Floyd '62] [Warshall '62]
- $n \times$ SSSP: $\tilde{O}(nm)$
- State-of-the-art for dense graphs: $\frac{n^3}{2^{\Omega(\sqrt{\log n})}}$ [Williams '14]
- **APSP Hypothesis:** no algorithm in time $O(n^{2.99})$



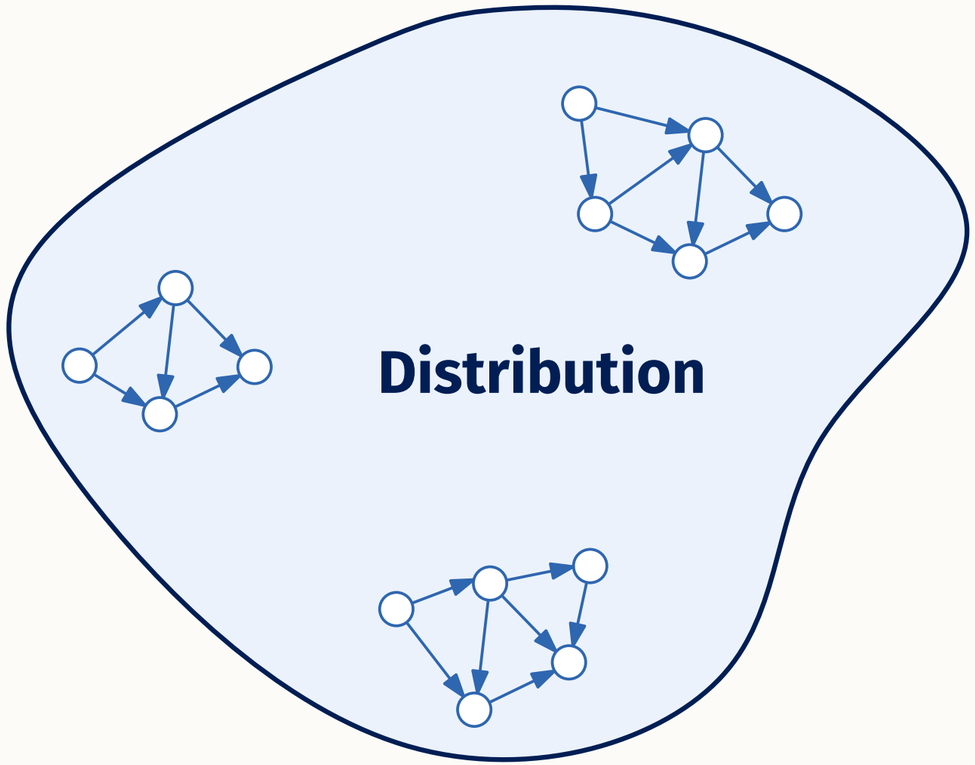
D	a	b	c	d	e
a	0	1	4	3	5
b	∞	0	∞	1	3
c	∞	1	0	2	4
d	∞	∞	∞	0	3
e	∞	∞	∞	∞	0

Warm-Starting APSP

Warm-Starting APSP

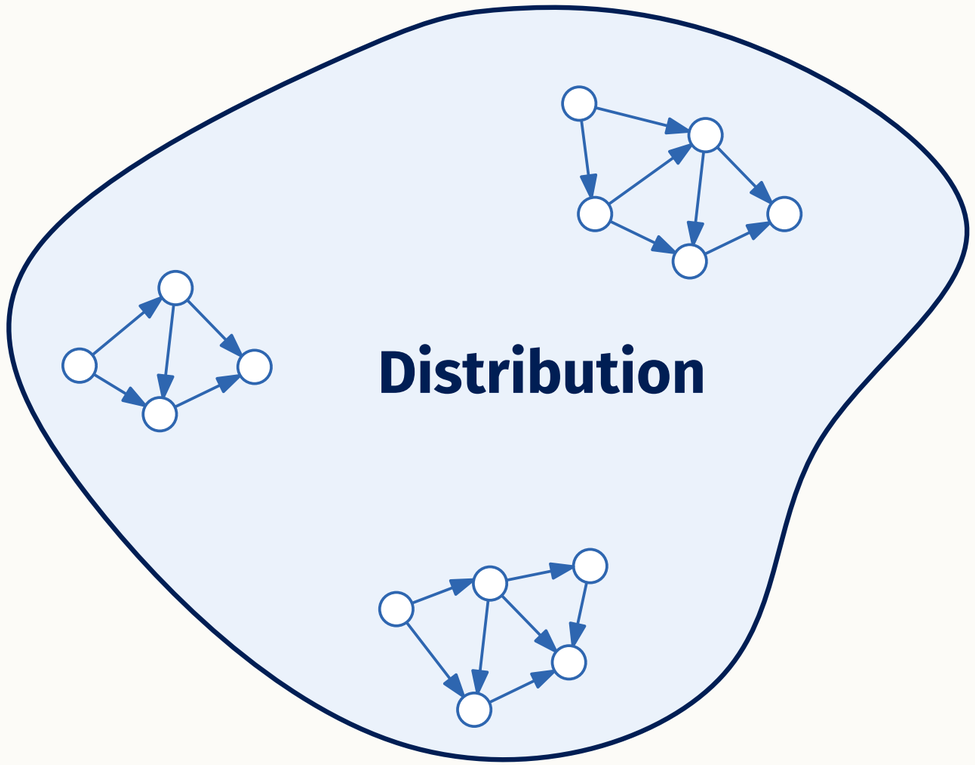


Warm-Starting APSP



Q: Can we use **common structure** to solve instances faster (than n^3)?

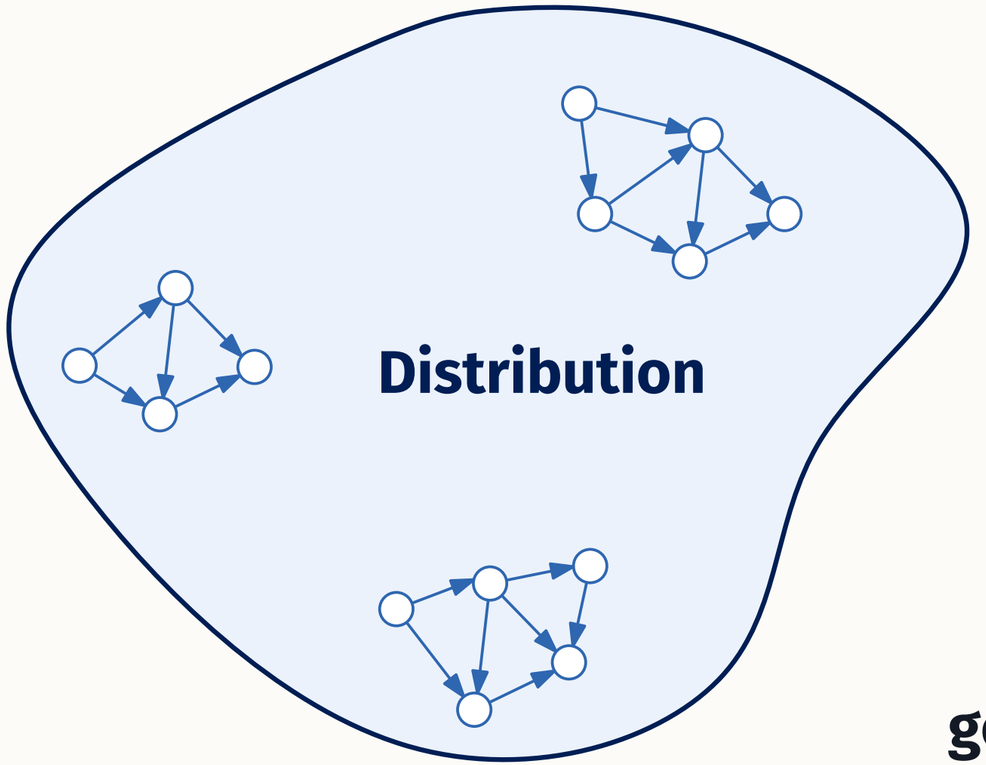
Warm-Starting APSP



Q: Can we use **common structure** to solve instances faster (than n^3)?

Algs with Predictions: Learn some **prediction** to speed up the algorithm!

Warm-Starting APSP



Q: Can we use **common structure** to solve instances faster (than n^3)?

Algs with Predictions: Learn some **prediction** to speed up the algorithm!

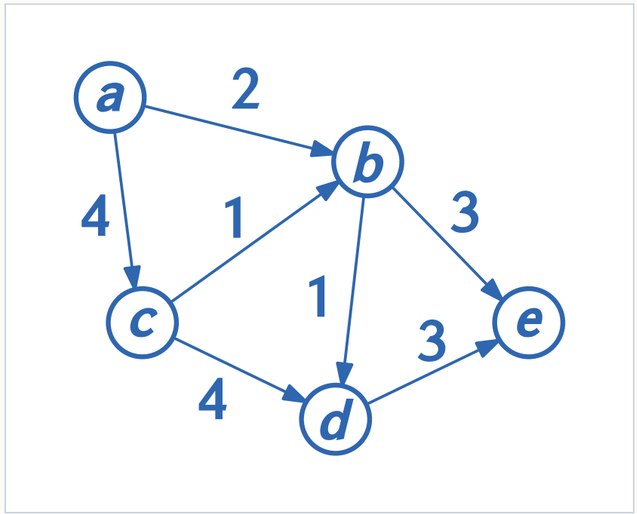
good prediction: provable fast alg

smooth tradeoff

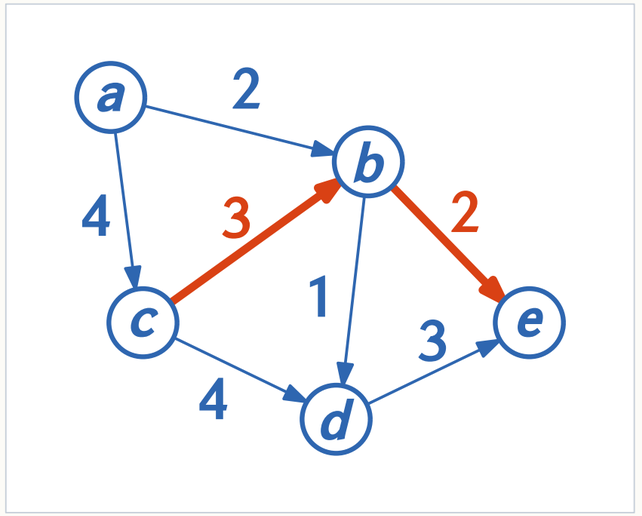
bad prediction: worst-case alg

What kind of structure?

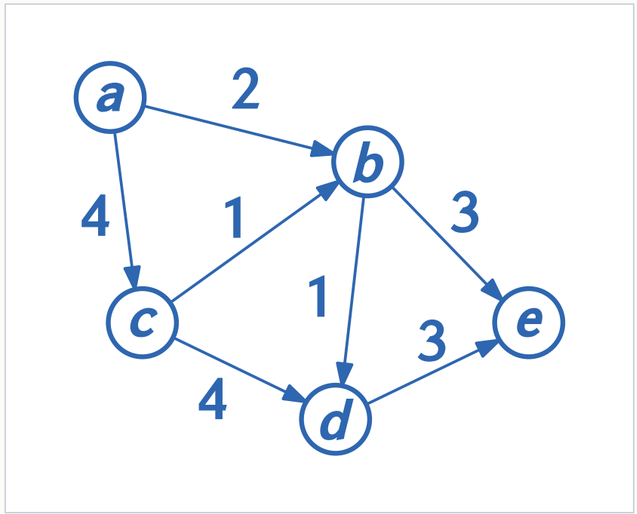
What kind of structure?



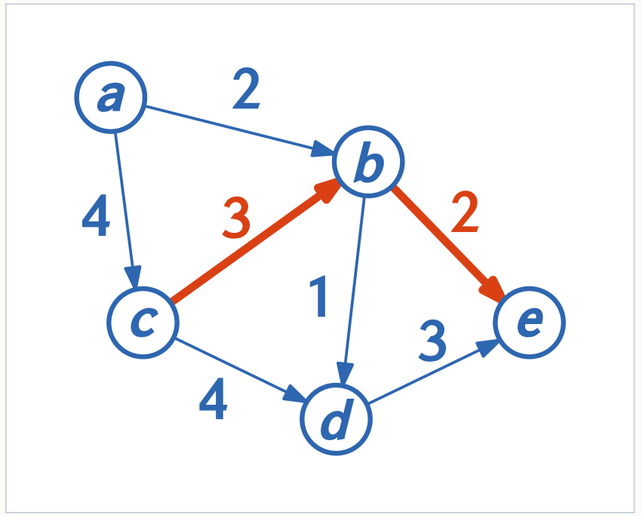
structure = **few input changes??**



What kind of structure?



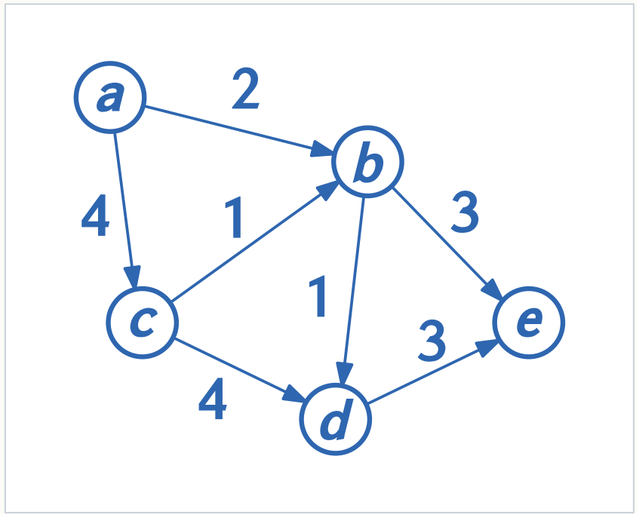
structure = **few input changes??**



Use **dynamic algorithms** instead!

- Subcubic algorithm exists for less than $\sqrt{n}$ vertices changed [Mao '24]

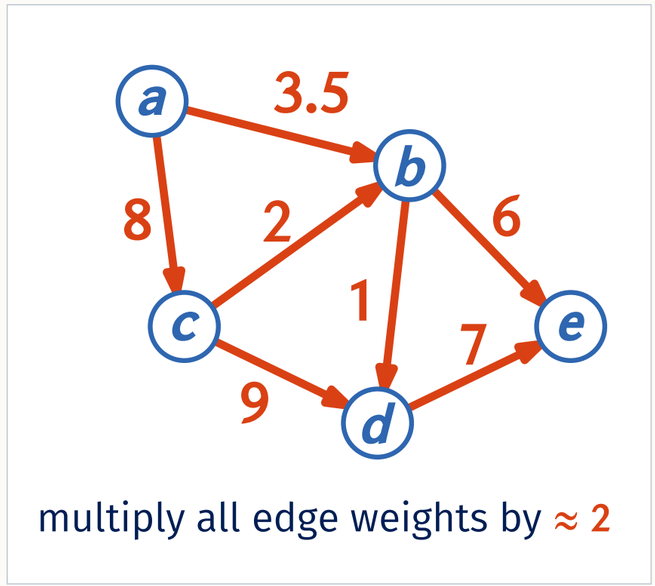
What kind of structure?



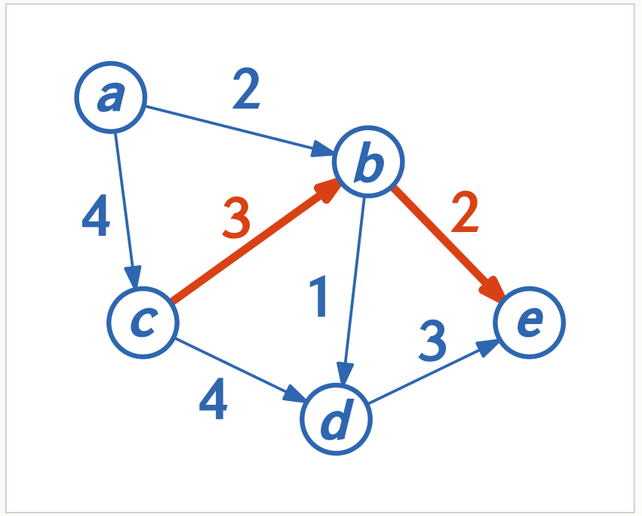
structure = **few input changes??**

different numbers, similar **path structure**

e.g., road network during rush hours



multiply all edge weights by ≈ 2



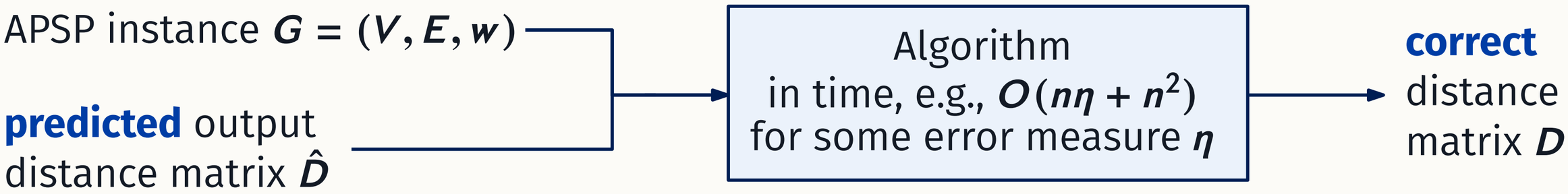
Use **dynamic algorithms** instead!

- Subcubic algorithm exists for less than $\sqrt{n}$ vertices changed [Mao '24]

APSP with Predictions: Approach 1

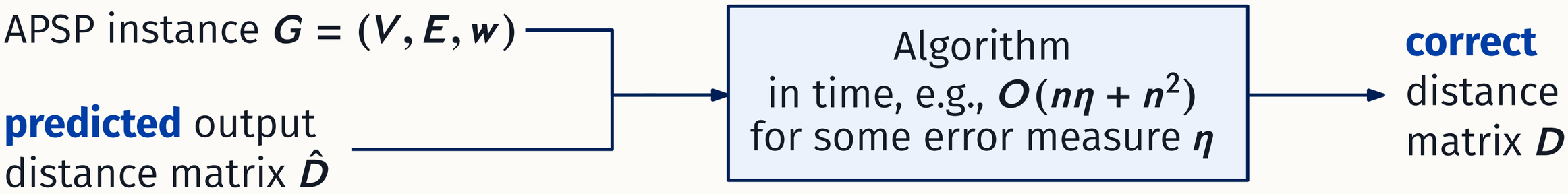
APSP with Predictions: Approach 1

What we would like to have:



APSP with Predictions: Approach 1

What we would like to have:



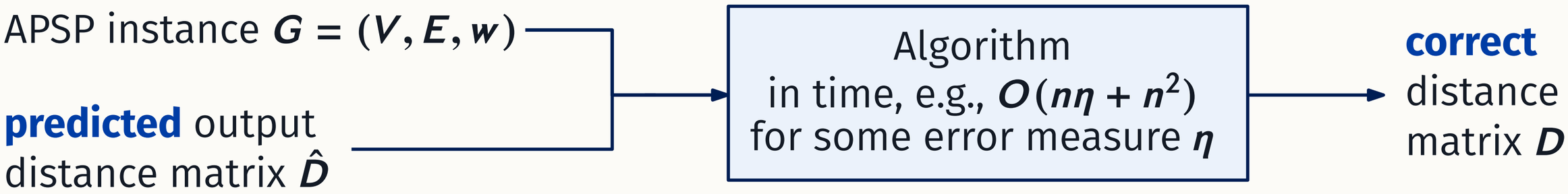
Issue:

Theorem [Vassilevska Williams, Williams '10]

If there is a $O(n^{2.99})$-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:

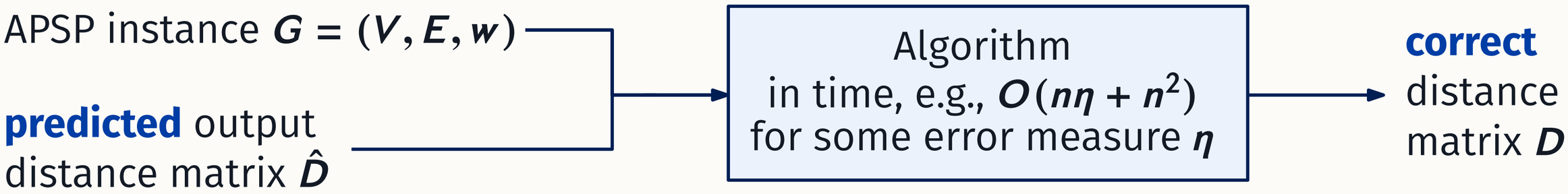
- Verifying $\hat{D}$ is as hard as computing D

Theorem [Vassilevska Williams, Williams '10]

If there is a $O(n^{2.99})$-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:

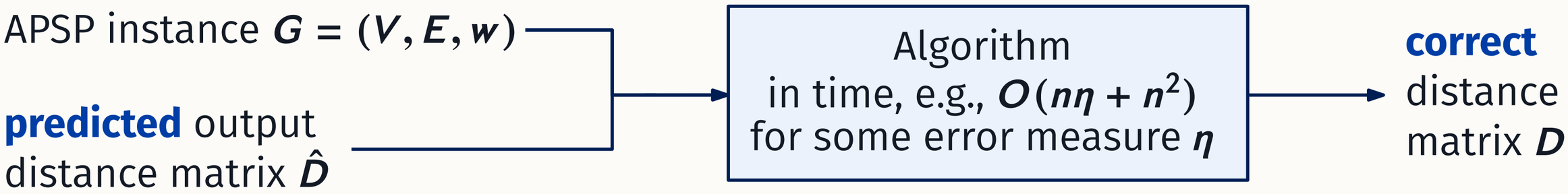
- Verifying $\hat{D}$ is as hard as computing D
- Approach **cannot work** with **any reasonable η**!

Theorem [Vassilevska Williams, Williams '10]

If there is a $O(n^{2.99})$-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
- Approach **cannot work** with **any reasonable η**!

Theorem [Vassilevska Williams, Williams '10]

If there is a $O(n^{2.99})$-time algorithm that **verifies** if a given matrix $\hat{D}$ is the correct distance matrix, the APSP hypothesis is false.

We need a **stronger prediction**!

APSP → **APSP Verification**

(Co-)nondeterministic APSP

(Co-)nondeterministic APSP

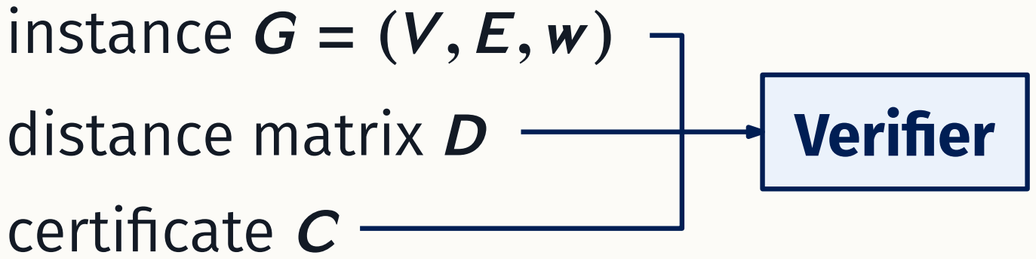
- Does there even exist a prediction with a subcubic algorithm if $\eta = 0$ ($\hat{D} = D$)?

(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

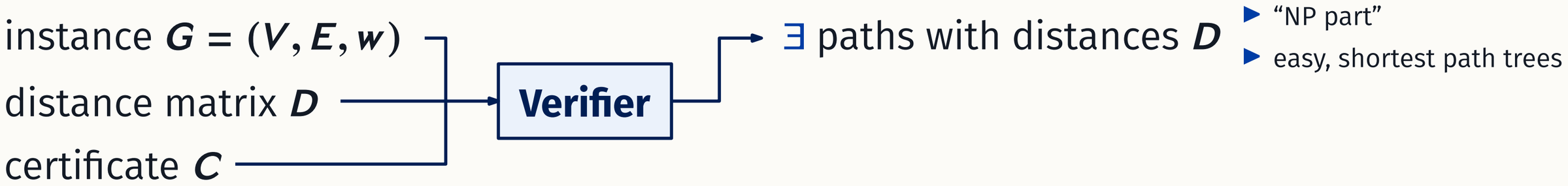
(Co-)nondeterministic APSP

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



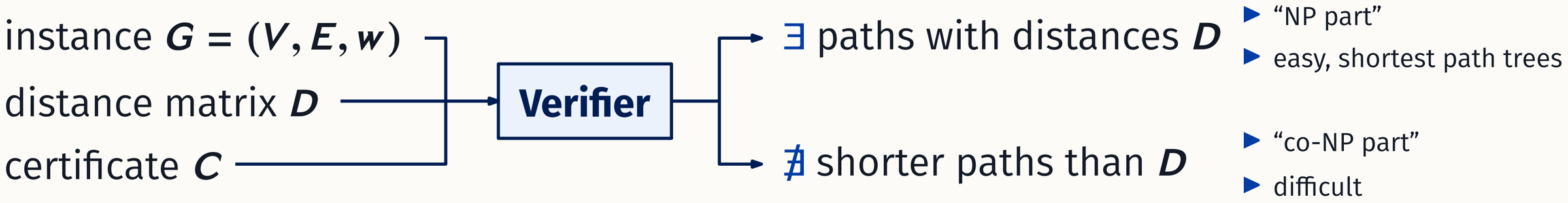
(Co-)nondeterministic APSP

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



(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

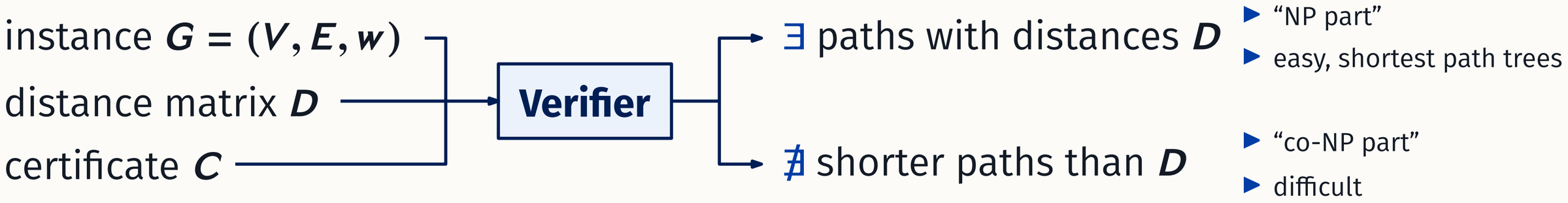


- "NP part"
- easy, shortest path trees

- "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

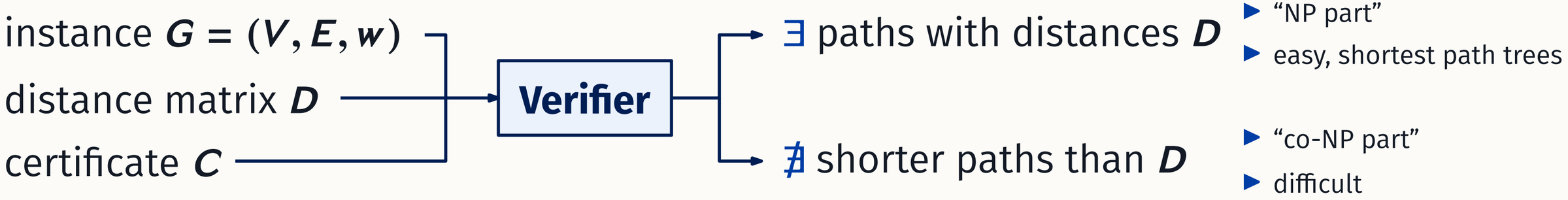
- "co-NP part"
- difficult

2 co-nondeterministic algorithms known:

- Hashing-based: time $O(n^{(\omega+3)/2})$ [Carmosino, Gao, Impagliazzo, Mihajlin, Paturi, Schneider '17]
- More structural / real inputs: time $O(n^{(\omega+6)/3})$ [Chan, Vassilevska Williams, Xu '23]

(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

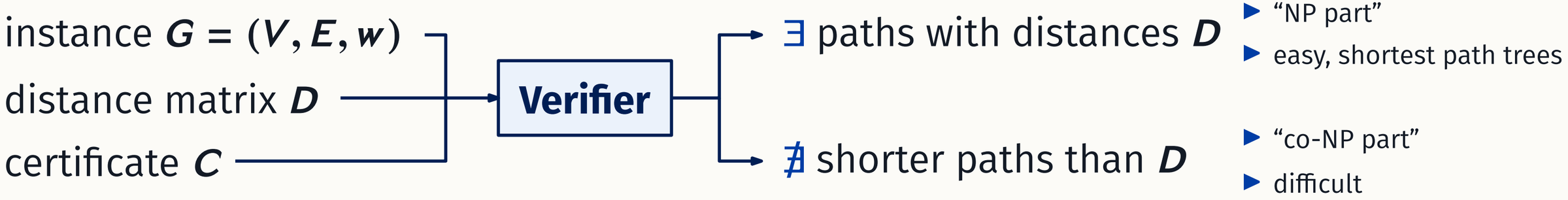
2 co-nondeterministic algorithms known:

$2 \leq \omega < 2.372$ 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 '17]
- More structural / real inputs: time $O(n^{2.66})$ [Chan, Vassilevska Williams, Xu '23]

(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

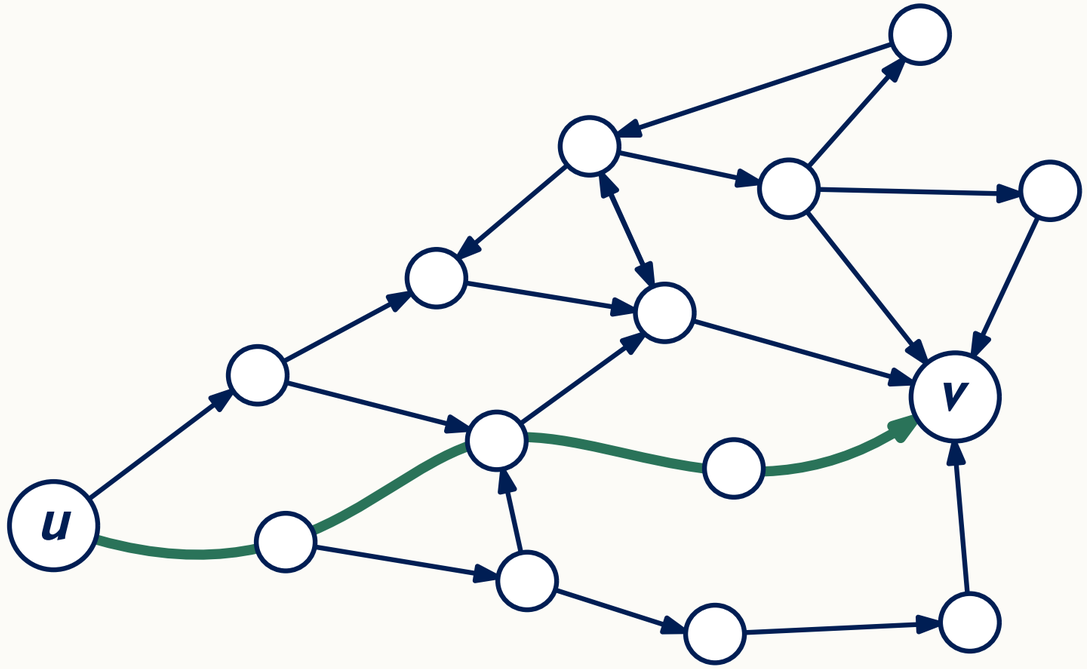
- "co-NP part"
- difficult

2 co-nondeterministic algorithms known:

$2 \leq \omega < 2.372$ 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 '17]
- More structural / real inputs: time $O(n^{2.66})$ [Chan, Vassilevska Williams, Xu '23]

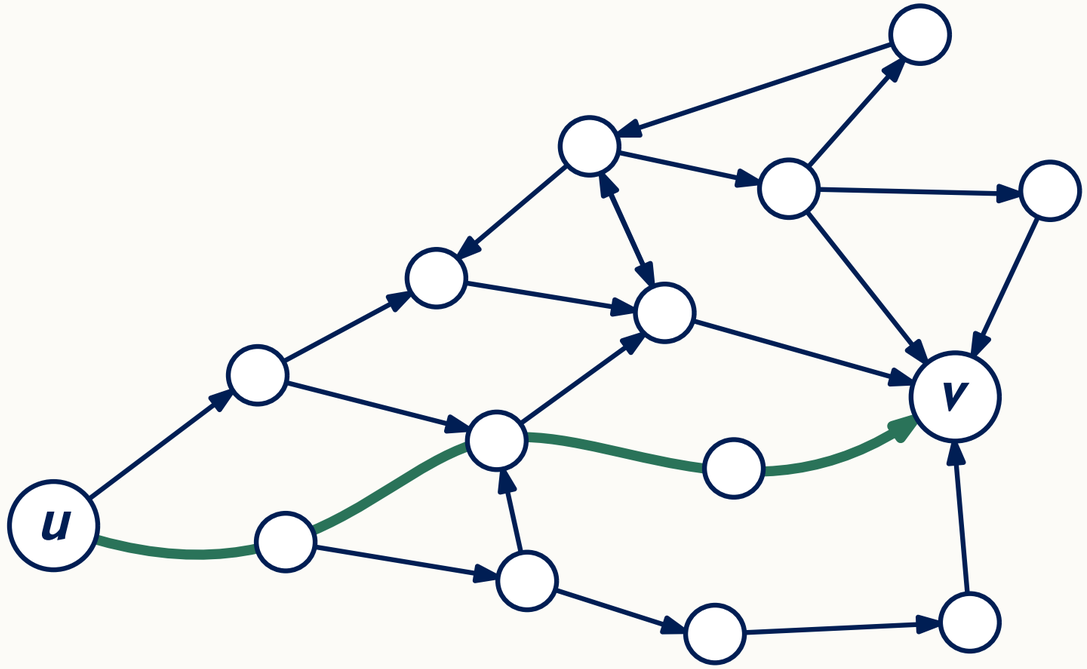
Our Result (Informal)



Our Result (Informal)

Certificate / Prediction C: (informal)

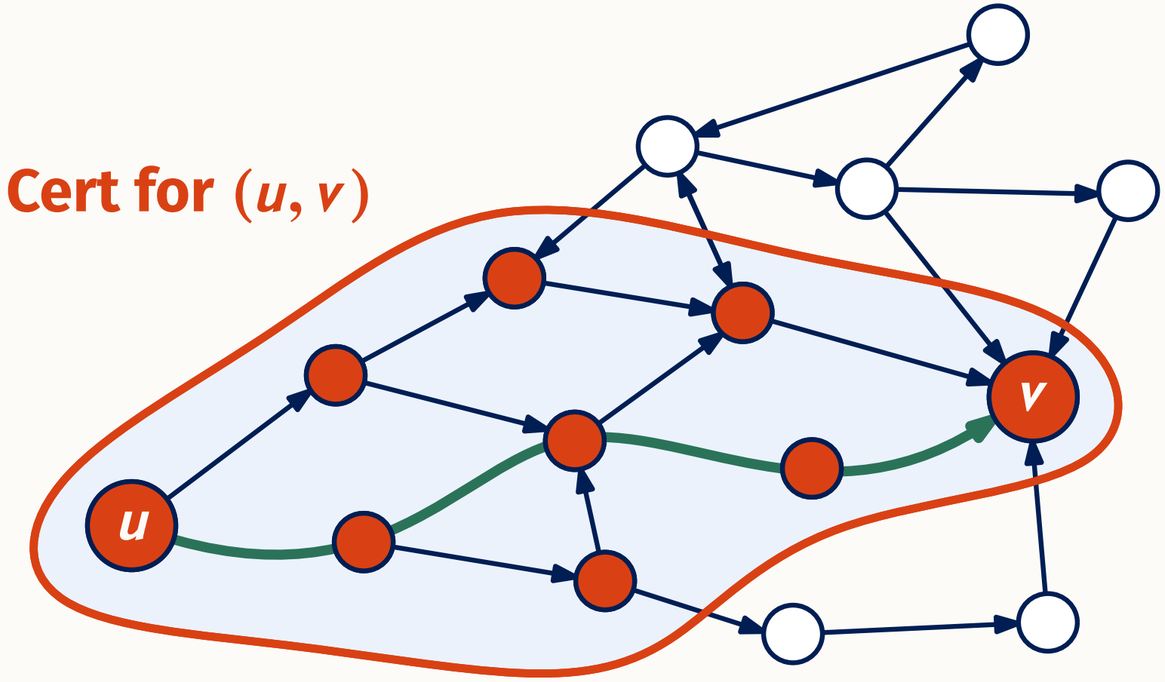
For every vertex pair (u, v), vertices **causing the shortest detour** on a u-v-path.



Our Result (Informal)

Certificate / Prediction C: (informal)

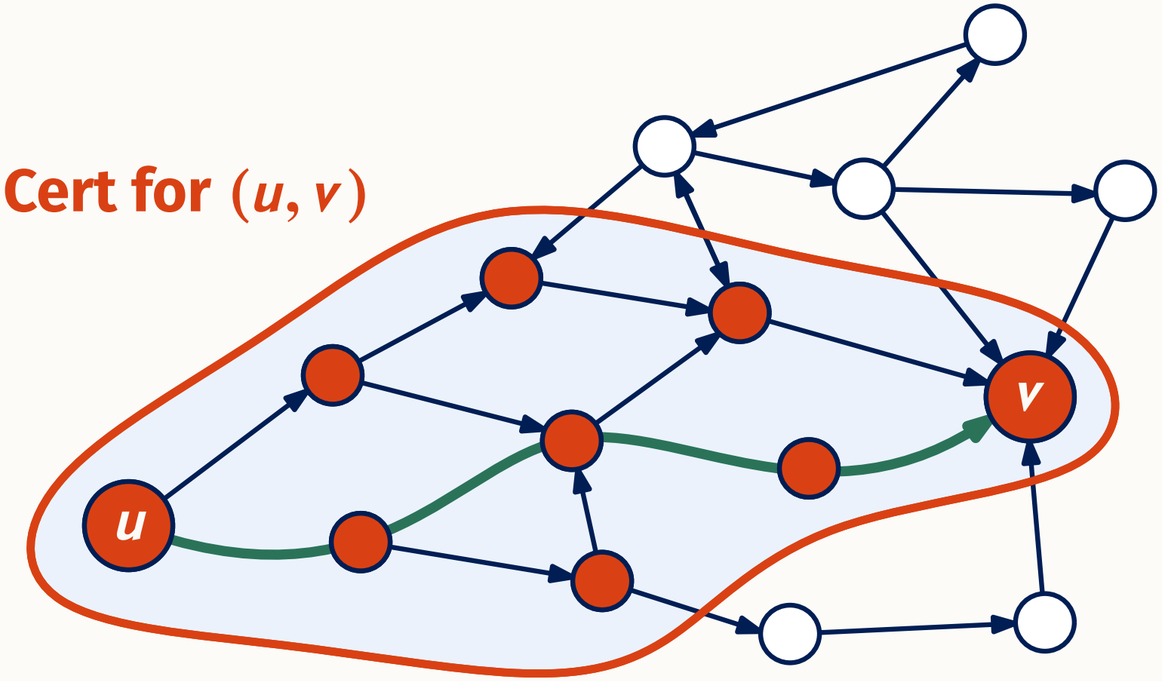
For every vertex pair (u, v), vertices **causing the shortest detour** on a u-v-path.



Our Result (Informal)

Certificate / Prediction C: (informal)

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



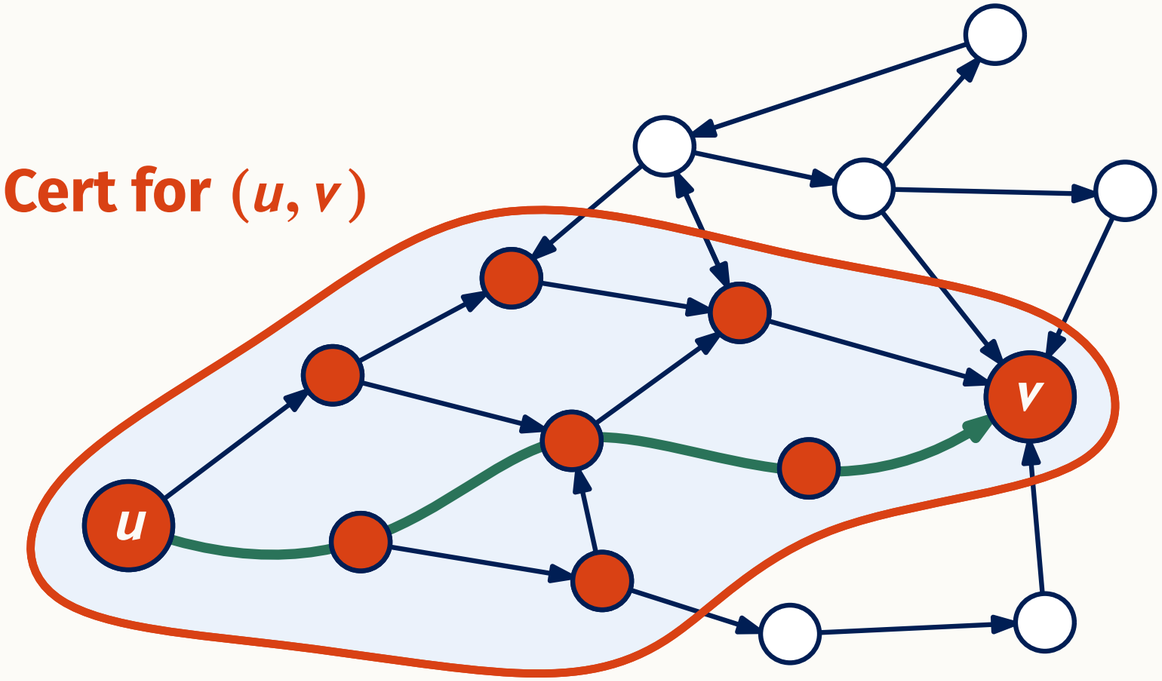
Error Measure η: (informal)

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 $\tilde{O}(n\eta + n^{2.75})$.

Turning the Verifier Into an Algorithm with Predictions

Turning the Verifier Into an Algorithm with Predictions

- Use the certificate C as our prediction

Certificate:

A matrix C where $C[i, j] \subseteq V$ contains the p best **detour vertices** for (i, j).

Turning the Verifier Into an Algorithm with Predictions

- Use the certificate C as our prediction

Prediction:

A matrix C where $C[i, j] \subseteq V$ **should contain** the p best **detour vertices** for (i, j).

Turning the Verifier Into an Algorithm with Predictions

- Use the certificate C as our prediction

3 steps:

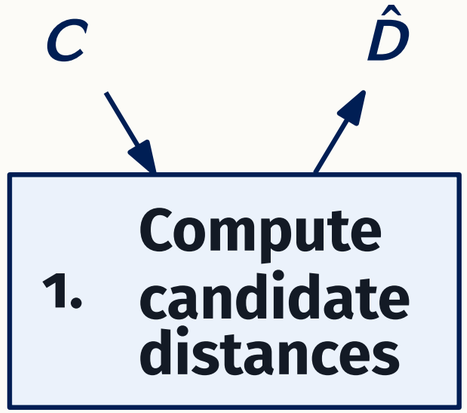
Prediction:

A matrix C where $C[i, j] \subseteq V$ **should contain** the p best **detour vertices** for (i, j).

Turning the Verifier Into an Algorithm with Predictions

- Use the certificate C as our prediction

3 steps:



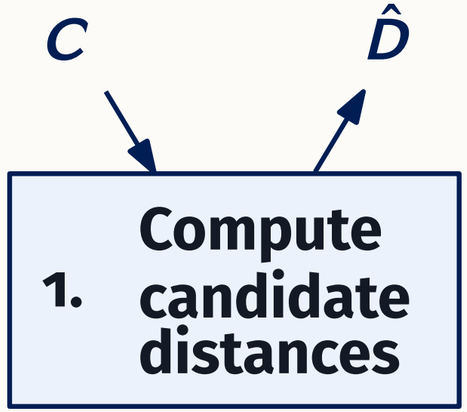
Prediction:

A matrix C where $C[i, j] \subseteq V$ **should contain** the p best **detour vertices** for (i, j).

Turning the Verifier Into an Algorithm with Predictions

- Use the certificate C as our prediction

3 steps:



Prediction:

A matrix C where $C[i,j] \subseteq V$ **should contain** the p best **detour vertices** for (i,j).

Lemma (informal)

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

"perform **all and only** relaxations in C" ← in a **good order**!

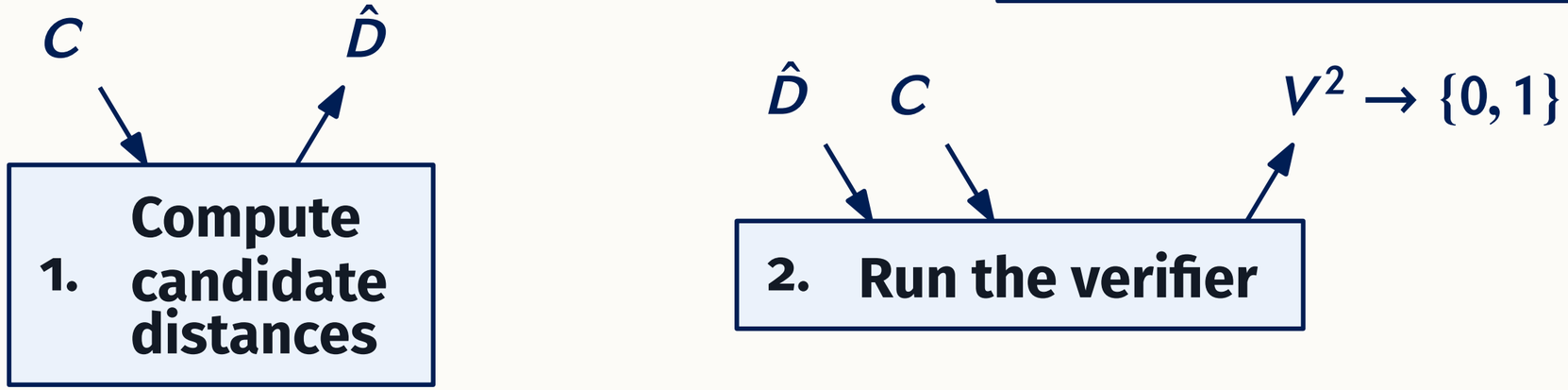
Turning the Verifier Into an Algorithm with Predictions

- Use the certificate C as our prediction

3 steps:

Prediction:

A matrix C where $C[i,j] \subseteq V$ **should contain** the p best **detour vertices** for (i,j).



Lemma (informal)

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

"perform **all and only** relaxations in C" ← in a **good order**!

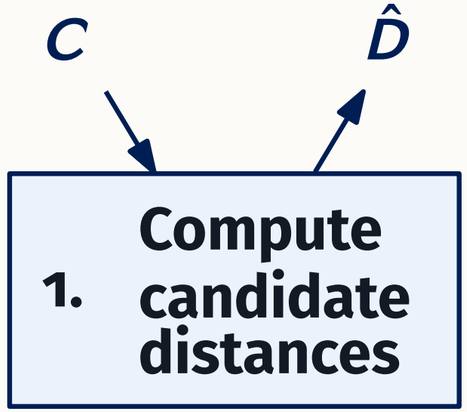
Turning the Verifier Into an Algorithm with Predictions

- Use the certificate C as our prediction

3 steps:

Prediction:

A matrix C where $C[i,j] \subseteq V$ **should contain** the p best **detour vertices** for (i,j).



verifies each entry $\hat{D}[i,j]$ **locally**:

$$\hat{D}[i,j] \stackrel{?}{=} \min_{k \in V} \hat{D}[i,k] + \hat{D}[k,j]$$

Lemma (informal)

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

"perform **all and only** relaxations in C" ← in a **good order**!

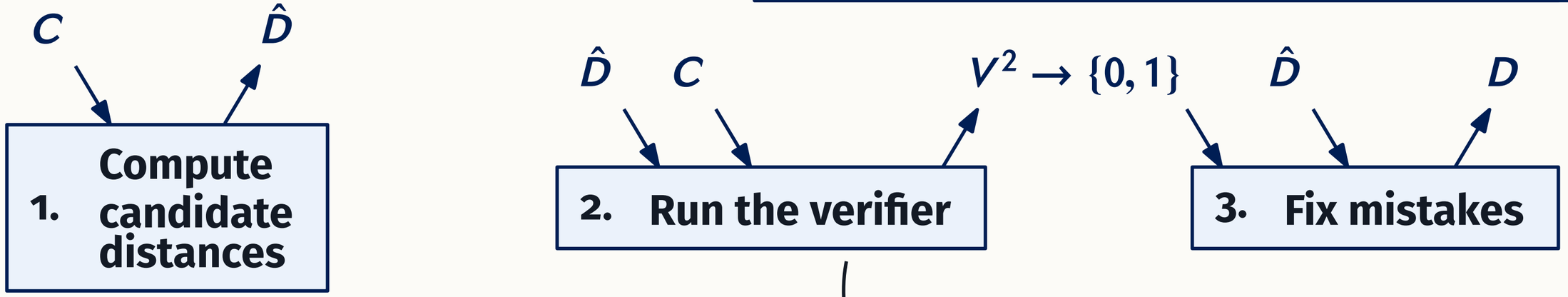
Turning the Verifier Into an Algorithm with Predictions

- Use the certificate C as our prediction

3 steps:

Prediction:

A matrix C where $C[i,j] \subseteq V$ **should contain** the p best **detour vertices** for (i,j).



Lemma (informal)

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

verifies each entry $\hat{D}[i,j]$ **locally**:

$$\hat{D}[i,j] \stackrel{?}{=} \min_{k \in V} \hat{D}[i,k] + \hat{D}[k,j]$$

"perform **all and only** relaxations in C" ← in a **good order**!

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

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:

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

Verifier detects these

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**.

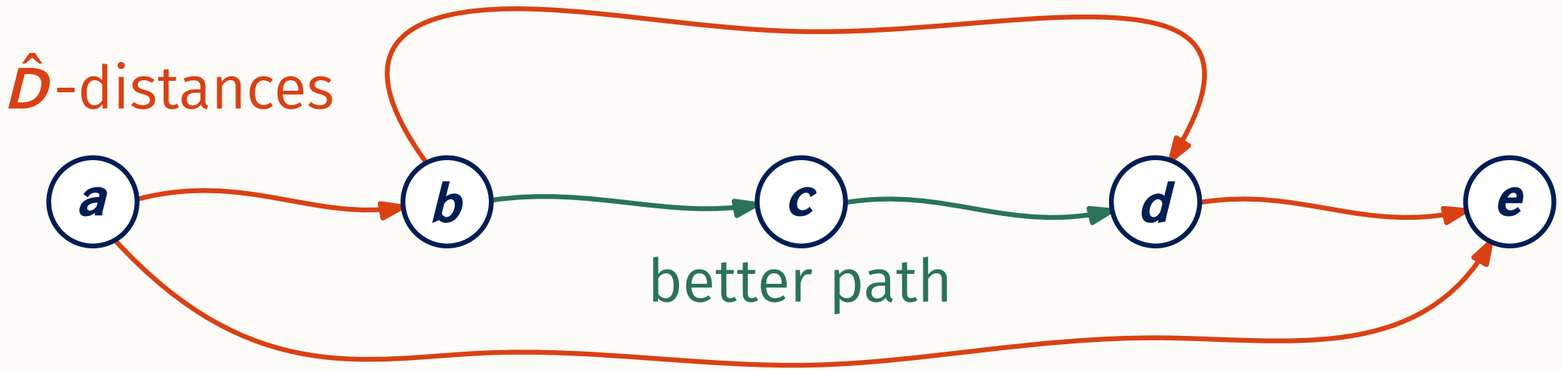
Error measure:

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

Verifier detects these

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:

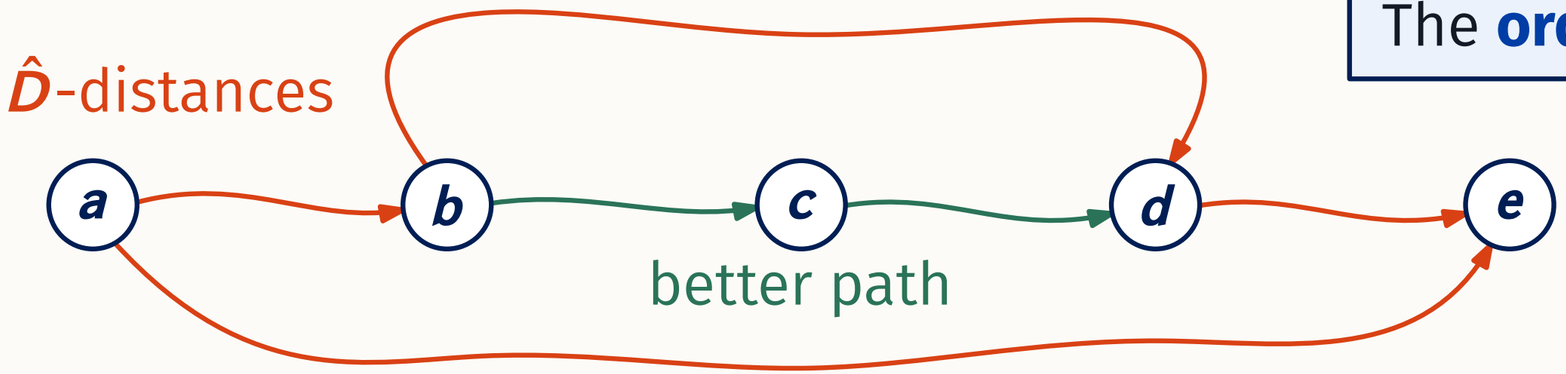
$$\eta := \#\{\,(i,j) : \text{verifier failed}\,\} + \#\{\,(i,j) : \hat{D}[i,j] \text{ incorrect}\,\}$$

Verifier detects these

Verifier might not detect

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

The **order of relaxations** matters!



Fixing: Algorithm Idea

Error measure:

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

Fixing: Algorithm Idea

Error measure:

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

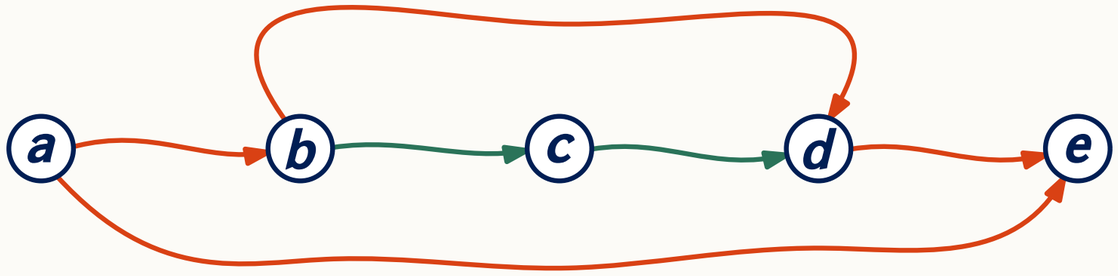
- Use **Johnson's trick** to assume w.l.o.g. that **weights are non-negative**

Fixing: Algorithm Idea

Error measure:

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

- Use **Johnson's trick** to assume w.l.o.g. that **weights are non-negative**
- Cosider pairs in **increasing order** of $\hat{D}[i,j]$



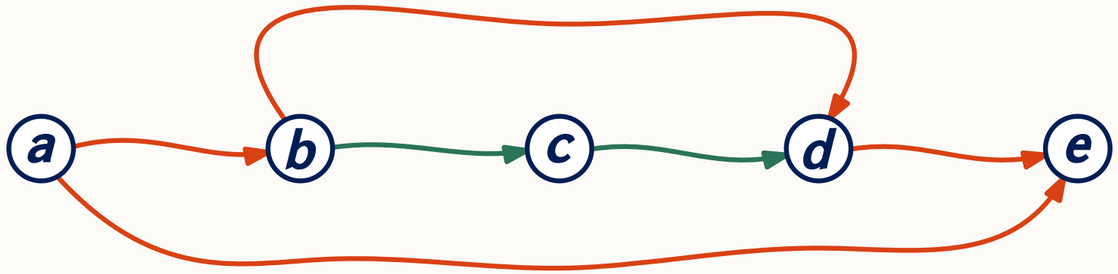
Fixing: Algorithm Idea

Error measure:

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

- Use **Johnson's trick** to assume w.l.o.g. that **weights are non-negative**
- Cosider pairs in **increasing order** of $\hat{D}[i,j]$

Subpaths of shortest paths will be considered first!



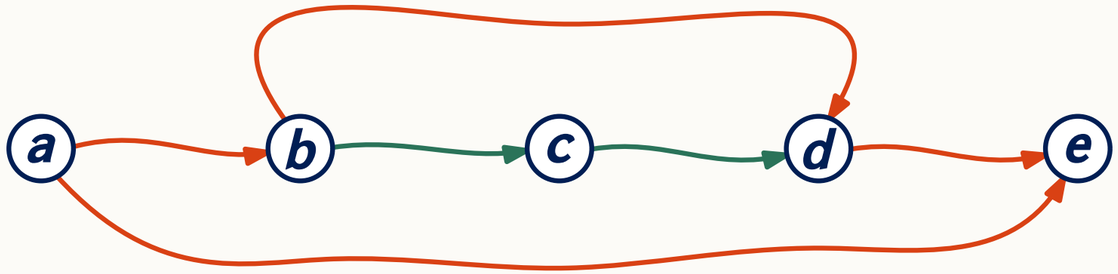
Fixing: Algorithm Idea

Error measure:

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

- Use **Johnson's trick** to assume w.l.o.g. that **weights are non-negative**
- Cosider pairs in **increasing order** of $\hat{D}[i,j]$

Subpaths of shortest paths will be considered first!



- Do relaxations as **lazily** as possible

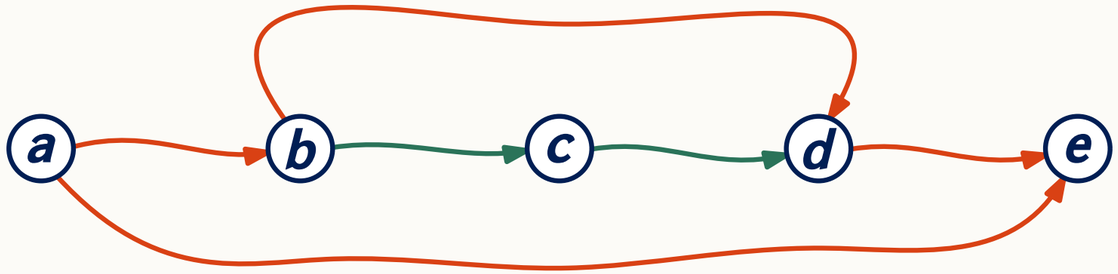
Fixing: Algorithm Idea

Error measure:

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

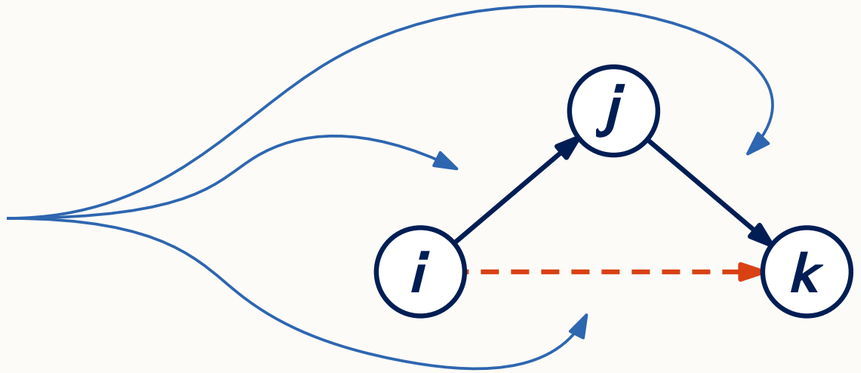
- Use **Johnson's trick** to assume w.l.o.g. that **weights are non-negative**
- Cosider pairs in **increasing order** of $\hat{D}[i,j]$

Subpaths of shortest paths will be considered first!



- Do relaxations as **lazily** as possible

only if we know that one of these is counted in η



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

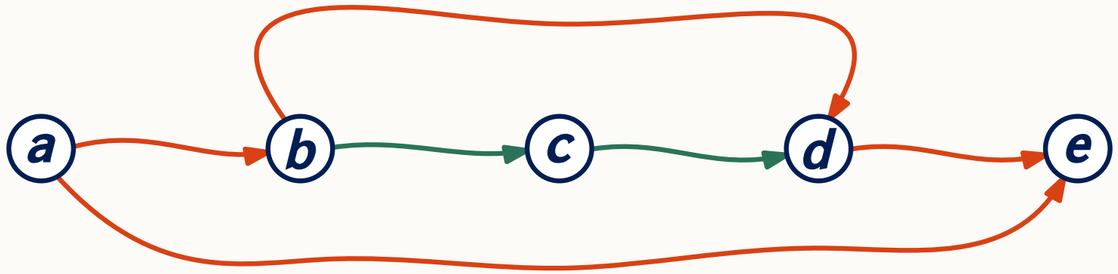
Fixing: Algorithm Idea

Error measure:

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

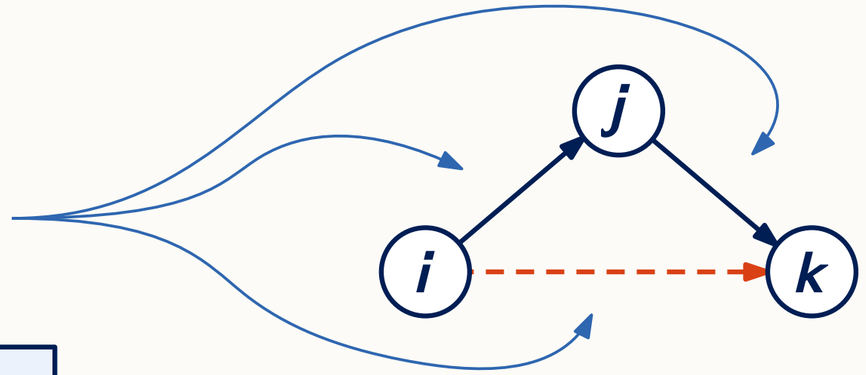
- Use **Johnson's trick** to assume w.l.o.g. that **weights are non-negative**
- Cosider pairs in **increasing order** of $\hat{D}[i,j]$

Subpaths of shortest paths will be considered first!



- Do relaxations as **lazily** as possible

only if we know that one of these is counted in η



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

Total # relaxations:

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

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!

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 w.l.o.g. 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 w.l.o.g. 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 w.l.o.g. 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:

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

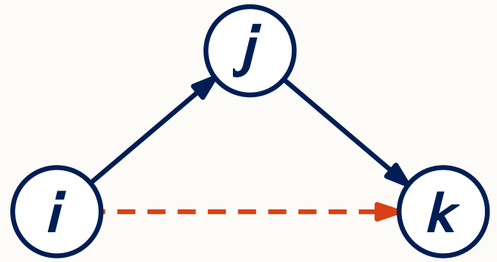
- Assume w.l.o.g. non-negative edge weights
- Keep a set of **marked pairs**, initially the set of unverified pairs
- Consider pairs (i,j) in **increasing order** of $\hat{D}[i,j]$:
  - If (i,j) is marked:
    - Relax **every pair** with $\hat{D}[i,j]$
    - If changed, mark

Fixing: Full Algorithm

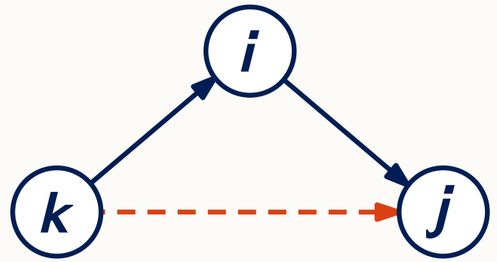
Error measure:

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

- Assume w.l.o.g. 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



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



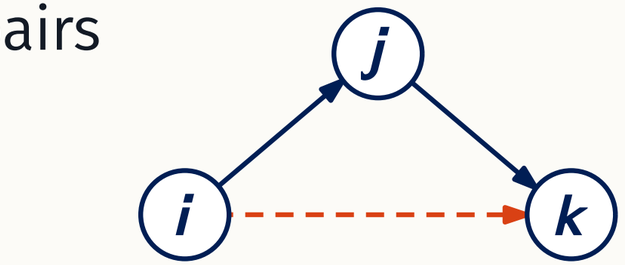
$D[k,j] = \min\{\ D[k,j],\ D[k,i] + D[i,j]\ \}$

Fixing: Full Algorithm

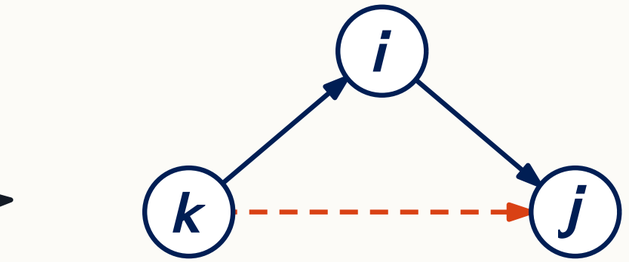
Error measure:

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

- Assume w.l.o.g. 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 $\hat{D}[i,j]$



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



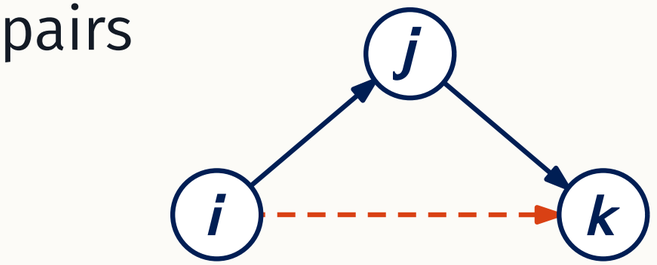
$$D[k,j] = \min\{ D[k,j],\ D[k,i] + D[i,j] \}$$

Fixing: Full Algorithm

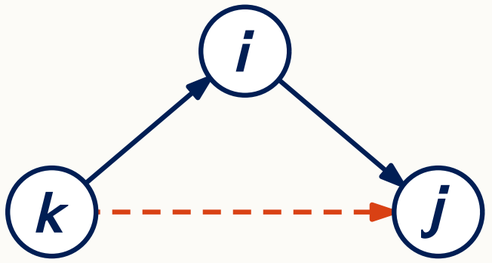
Error measure:

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

- Assume w.l.o.g. 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:

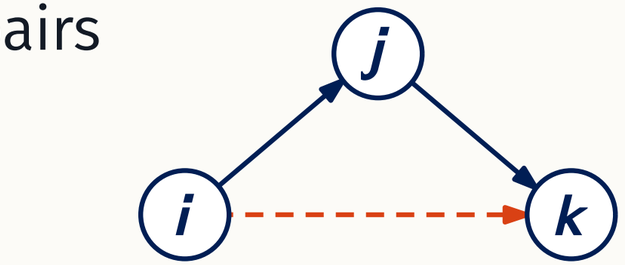
$\#\{\ \text{total marked pairs}\ \} \cdot O(n) \quad = \quad O(\eta \cdot n)$

Fixing: Full Algorithm

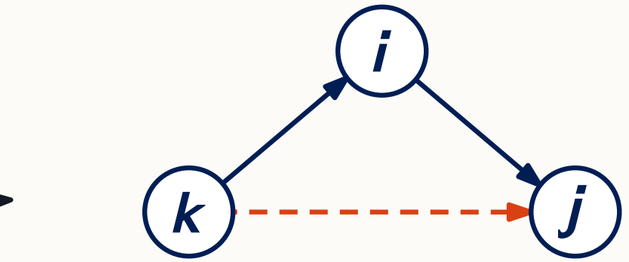
Error measure:

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

- Assume w.l.o.g. 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:

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

Correctness follows from order.