Simple label-correcting algorithms for partially dynamic approximate shortest paths in directed graphs

Adam Karczmarz, Jakub Łącki

Mara Grilnberger

University of Salzburg
Department of Computer Science

Setting

- maintaining (approximate) shortest paths in weighted, directed graph G where weights are non-negative
- partially dynamic setting
- incremental setting:
  - edge can be inserted
  - weight of an edge can decrease
- decremental Setting:
  - edge deletions
  - weight of an edge can increase

Related Work and Motivation

- many existing solutions for different settings
- main focus: APSP in decremental setting

Related Work and Motivation

- many existing solutions for different settings
- main focus: APSP in decremental setting
- best deterministic algorithm (dense graphs):
  using King's decremental transitive closure algorithm:
  $\rightarrow$ graphs G^{2^i} contain edge uv: path u to v in G with $\leq 2^i$ hops $\rightarrow$
  h-SSSP algorithm (Bernstein) to maintain approximate distances
- $\mathrm{O}(n^3 \log^3 n \log(nW)/\epsilon + \Delta)$ total update time
  $\mathrm{O}(n^2 \log n \log(nW))$ space

Related Work and Motivation

- many existing solutions for different settings
- main focus: APSP in decremental setting
- best deterministic algorithm (dense graphs):
  using King's decremental transitive closure algorithm:
  $\rightarrow$ graphs G^{2^i} contain edge uv: path u to v in G with $\leq 2^i$ hops $\rightarrow$
  h-SSSP algorithm (Bernstein) to maintain approximate distances
- $\mathrm{O}(n^3 \log^3 n \log(nW)/\epsilon + \Delta)$ total update time
  $\mathrm{O}(n^2 \log n \log(nW))$ space
- this paper: $\mathrm{O}(n^3 \log n \log(nW)/\epsilon + \Delta)$
  additional space: $\mathrm{O}(n^2)$

Motivation

- shortest path algorithms maintain distance estimates $\mathrm{d} : V \to \mathbb{R}$ and relaxing edges/vertices

Motivation

- shortest path algorithms maintain distance estimates $\mathrm{d} : V \to \mathbb{R}$ and relaxing edges/vertices

edge relaxation

A weighted edge uv is called relaxed, if $\mathrm{d}(v) \leq \mathrm{d}(u) + \mathrm{w}(uv)$ where $\mathrm{w}(uv)$ is the weight of edge uv, and tense otherwise.

Motivation

- shortest path algorithms maintain distance estimates $\mathrm{d} : V \to \mathbb{R}$ and relaxing edges/vertices

edge relaxation

A weighted edge uv is called relaxed, if $\mathrm{d}(v) \leq \mathrm{d}(u) + \mathrm{w}(uv)$ where $\mathrm{w}(uv)$ is the weight of edge uv, and tense otherwise.

- relaxing a tense edge: set $\mathrm{d}(v) = \mathrm{d}(u) + \mathrm{w}(uv)$
- also works in incremental setting

Motivation

- shortest path algorithms maintain distance estimates $\mathrm{d} : V \to \mathbb{R}$ and relaxing edges/vertices

edge relaxation

A weighted edge uv is called relaxed, if $\mathrm{d}(v) \leq \mathrm{d}(u) + \mathrm{w}(uv)$ where $\mathrm{w}(uv)$ is the weight of edge uv, and tense otherwise.

- relaxing a tense edge: set $\mathrm{d}(v) = \mathrm{d}(u) + \mathrm{w}(uv)$
- also works in incremental setting
- decremental setting:

vertex relaxation

A vertex v is called relaxed, if $\mathrm{d}(v) < \min_{uv \in E(G)}\{\mathrm{d}(u) + \mathrm{w}(uv)\}$ and we set $\mathrm{d}(v) := \min_{uv \in E(G)}\{\mathrm{d}(u) + \mathrm{w}(uv)\}$

Approximate APSP - Idea

- each pair of vertices: maintain distance estimate $\mathrm{d}(u, v)$
- distance estimates: $(1 + \epsilon)$ approximations of real distance
- relaxation operation:
  - compute $\mathrm{t}(u, v)$: estimated length of shortest path from u to v
  - set distance estimate to $\mathrm{t}(u, v)$

Approximate APSP - Idea

- each pair of vertices: maintain distance estimate $\mathrm{d}(u, v)$
- distance estimates: $(1+\epsilon)$ approximations of real distance
- relaxation operation:
  - compute $\mathrm{t}(u, v)$: estimated length of shortest path from u to v
  - set distance estimate to $\mathrm{t}(u, v)$
- when distance estimate increases
  - check all possibly affected distance estimates $\mathrm{d}(w, z)$
  - increase them if $\mathrm{d}(w, z) < \mathrm{t}(w, z)$

Relaxation Operation

- $M_{u,v} = \{\mathrm{d}(u,z) + \mathrm{d}(z,v) : z \in V \setminus \{u,v\}\}$

$$\mathrm{t}(u,v) := \mathrm{r}_{1+\epsilon}(\min(M_{u,v}, \mathrm{w}(uv)))$$

where:

$$\mathrm{r}_{1+\epsilon}(x) = (1+\epsilon)^{\lceil \log_{1+\epsilon} x \rceil}$$

we round the value $x > 0$ up to nearest $(1+\epsilon)^i$ for $i \in \mathbb{N}_0$

Relaxation Operation

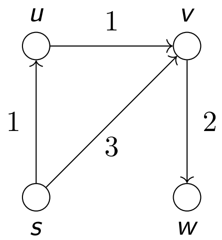
- $M_{u,v} = \{\mathrm{d}(u,z) + \mathrm{d}(z,v) : z \in V \setminus \{u,v\}\}$

$$\mathrm{t}(u,v) := \mathrm{r}_{1+\epsilon}(\min(M_{u,v}, \mathrm{w}(uv)))$$

where:

$$\mathrm{r}_{1+\epsilon}(x) = (1+\epsilon)^{\lceil \log_{1+\epsilon} x \rceil}$$

we round the value $x > 0$ up to nearest $(1+\epsilon)^i$ for $i \in \mathbb{N}_0$



Relaxation Operation

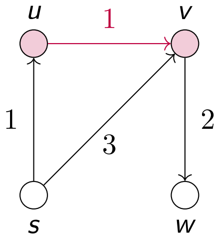
- $M_{u,v} = \{\mathrm{d}(u,z) + \mathrm{d}(z,v) : z \in V \setminus \{u,v\}\}$

$$\mathrm{t}(u,v) := \mathrm{r}_{1+\epsilon}(\min(M_{u,v}, \mathrm{w}(uv)))$$

where:

$$\mathrm{r}_{1+\epsilon}(x) = (1+\epsilon)^{\lceil \log_{1+\epsilon} x \rceil}$$

we round the value $x > 0$ up to nearest $(1+\epsilon)^i$ for $i \in \mathbb{N}_0$



$$\mathrm{t}(u,v) = \mathrm{r}_{1+\epsilon}(1) = 1$$

Relaxation Operation

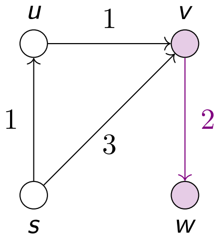
- $M_{u,v} = \{\mathrm{d}(u, z) + \mathrm{d}(z, v) : z \in V \setminus \{u, v\}\}$

$$\mathrm{t}(u, v) := \mathrm{r}_{1+\epsilon}(\min(M_{u,v}, \mathrm{w}(uv)))$$

where:

$$\mathrm{r}_{1+\epsilon}(x) = (1+\epsilon)^{\lceil \log_{1+\epsilon} x \rceil}$$

we round the value $x > 0$ up to nearest $(1+\epsilon)^i$ for $i \in \mathbb{N}_0$



$\mathrm{t}(u, v) = \mathrm{r}_{1+\epsilon}(1) = 1$

$\mathrm{t}(v, w) = \mathrm{r}_{1+\epsilon}(2) = (1+\epsilon)^j$

Relaxation Operation

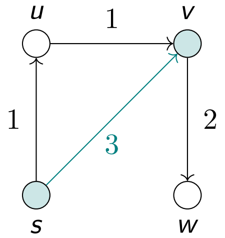
- $M_{u,v} = \{\mathrm{d}(u,z) + \mathrm{d}(z,v) : z \in V \setminus \{u,v\}\}$

$$\mathrm{t}(u,v) := \mathrm{r}_{1+\epsilon}(\min(M_{u,v}, \mathrm{w}(uv)))$$

where:

$$\mathrm{r}_{1+\epsilon}(x) = (1+\epsilon)^{\lceil \log_{1+\epsilon} x \rceil}$$

we round the value $x > 0$ up to nearest $(1+\epsilon)^i$ for $i \in \mathbb{N}_0$



$\mathrm{t}(u,v) = \mathrm{r}_{1+\epsilon}(1) = 1$

$\mathrm{w}(sv) = 3$

Relaxation Operation

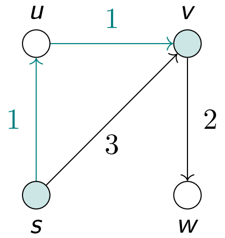
- $M_{u,v} = \{\mathrm{d}(u,z) + \mathrm{d}(z,v) : z \in V \setminus \{u,v\}\}$

$$\mathrm{t}(u,v) := \mathrm{r}_{1+\epsilon}(\min(M_{u,v}, \mathrm{w}(uv)))$$

where:

$$\mathrm{r}_{1+\epsilon}(x) = (1+\epsilon)^{\lceil \log_{1+\epsilon} x \rceil}$$

we round the value $x > 0$ up to nearest $(1+\epsilon)^i$ for $i \in \mathbb{N}_0$



$\mathrm{t}(u,v) = \mathrm{r}_{1+\epsilon}(1) = 1$

$\mathrm{w}(sv) = 3$

$M_{s,v} = \{(1+1), \infty\}$

Relaxation Operation

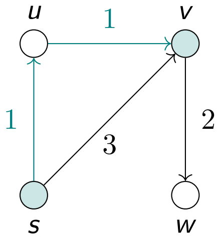
- $M_{u,v} = \{\mathrm{d}(u,z) + \mathrm{d}(z,v) : z \in V \setminus \{u,v\}\}$

$$\mathrm{t}(u,v) := \mathrm{r}_{1+\epsilon}(\min(M_{u,v}, \mathrm{w}(uv)))$$

where:

$$\mathrm{r}_{1+\epsilon}(x) = (1+\epsilon)^{\lceil \log_{1+\epsilon} x \rceil}$$

we round the value $x > 0$ up to nearest $(1+\epsilon)^i$ for $i \in \mathbb{N}_0$



$\mathrm{t}(u,v) = \mathrm{r}_{1+\epsilon}(1) = 1$

$\mathrm{w}(sv) = 3$

$M_{s,v} = \{(1+1), \infty\}$

$\mathrm{t}(s,v) = \mathrm{r}_{1+\epsilon}(2)$

Relaxation Operation

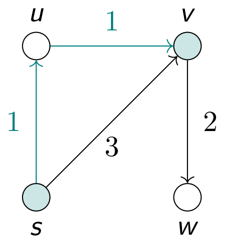
- $M_{u,v} = \{\mathrm{d}(u,z) + \mathrm{d}(z,v) : z \in V \setminus \{u,v\}\}$

$$\mathrm{t}(u,v) := \mathrm{r}_{1+\epsilon}(\min(M_{u,v}, \mathrm{w}(uv)))$$

where:

$$\mathrm{r}_{1+\epsilon}(x) = (1+\epsilon)^{\lceil \log_{1+\epsilon} x \rceil}$$

we round the value $x > 0$ up to nearest $(1+\epsilon)^i$ for $i \in \mathbb{N}_0$



$$\mathrm{t}(u,v) = \mathrm{r}_{1+\epsilon}(1) = 1$$
$$\mathrm{w}(sv) = 3$$
$$M_{s,v} = \{(1+1), \infty\}$$
$$\mathrm{t}(s,v) = \mathrm{r}_{1+\epsilon}(2) = (1+\epsilon)^j$$

Approximate APSP - Update

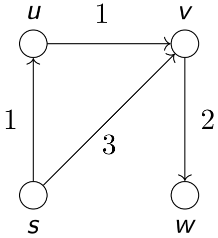
Update(u, v):

- Calculate $\mathrm{t}(u, v)$
- If distance estimate $\mathrm{d}(u, v) \neq \mathrm{t}(u, v)$: update it
  - For every $y \in V \setminus \{u, v\}$:
    Update(y, v) and Update(u, y)

Approximate APSP - Update

Update(u, v):

- Calculate $\mathrm{t}(u, v)$
- If distance estimate $\mathrm{d}(u, v) \neq \mathrm{t}(u, v)$: update it
  - For every $y \in V \setminus \{u, v\}$:
    Update(y, v) and Update(u, y)

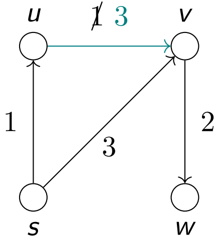


	s	u	v	w
s	0	1	$\mathrm{r}(2)$	$\mathrm{r}(2\,\mathrm{r}(2))$
u	∞	0	1	$\mathrm{r}(\mathrm{r}(2)+1)$
v	∞	∞	0	$\mathrm{r}(2)$
w	∞	∞	∞	0

Approximate APSP - Update

Update(u, v):

- Calculate $\mathrm{t}(u, v)$
- If distance estimate $\mathrm{d}(u, v) \neq \mathrm{t}(u, v)$: update it
  - For every $y \in V \setminus \{u, v\}$:
    Update(y, v) and Update(u, y)

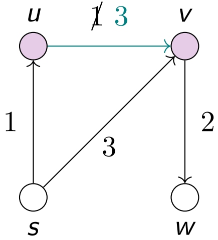


	s	u	v	w
s	0	1	$\mathrm{r}(2)$	$\mathrm{r}(2\,\mathrm{r}(2))$
u	∞	0	1	$\mathrm{r}(\mathrm{r}(2)+1)$
v	∞	∞	0	$\mathrm{r}(2)$
w	∞	∞	∞	0

Approximate APSP - Update

Update(u, v):

- Calculate $\mathrm{t}(u, v)$
- If distance estimate $\mathrm{d}(u, v) \neq \mathrm{t}(u, v)$: update it
  - For every $y \in V \setminus \{u, v\}$:
    Update(y, v) and Update(u, y)

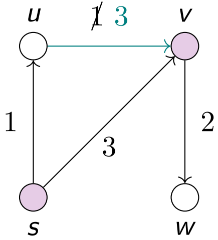


	s	u	v	w
s	0	1	$\mathrm{r}(2)$	$\mathrm{r}(2\,\mathrm{r}(2))$
u	∞	0	1	$\mathrm{r}(\mathrm{r}(2) + 1)$
v	∞	∞	0	$\mathrm{r}(2)$
w	∞	∞	∞	0

Approximate APSP - Update

Update(u, v):

- Calculate $\mathrm{t}(u, v)$
- If distance estimate $\mathrm{d}(u, v) \neq \mathrm{t}(u, v)$: update it
  - For every $y \in V \setminus \{u, v\}$:
    Update(y, v) and Update(u, y)

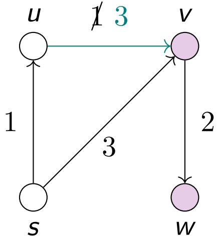


	s	u	v	w
s	0	1	$\mathrm{r}(2)$	$\mathrm{r}(2\,\mathrm{r}(2))$
u	∞	0	$\mathrm{r}(3)$	$\mathrm{r}(\mathrm{r}(2)+1)$
v	∞	∞	0	$\mathrm{r}(2)$
w	∞	∞	∞	0

Approximate APSP - Update

Update(u, v):

- Calculate $\mathrm{t}(u, v)$
- If distance estimate $\mathrm{d}(u, v) \neq \mathrm{t}(u, v)$: update it
    - For every $y \in V \setminus \{u, v\}$:
      Update(y, v) and Update(u, y)

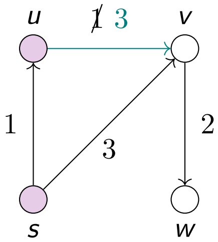


	s	u	v	w
s	0	1	$\mathrm{r}(3)$	$\mathrm{r}(2\,\mathrm{r}(2))$
u	∞	0	$\mathrm{r}(3)$	$\mathrm{r}(\mathrm{r}(2)+1)$
v	∞	∞	0	$\mathrm{r}(2)$
w	∞	∞	∞	0

Approximate APSP - Update

Update(u, v):

- Calculate $\mathrm{t}(u, v)$
- If distance estimate $\mathrm{d}(u, v) \neq \mathrm{t}(u, v)$: update it
  - For every $y \in V \setminus \{u, v\}$:
    Update(y, v) and Update(u, y)

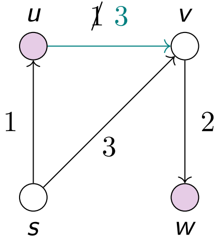


	s	u	v	w
s	0	1	$\mathrm{r}(3)$	$\mathrm{r}(2\,\mathrm{r}(2))$
u	∞	0	$\mathrm{r}(3)$	$\mathrm{r}(\mathrm{r}(2)+1)$
v	∞	∞	0	$\mathrm{r}(2)$
w	∞	∞	∞	0

Approximate APSP - Update

Update(u, v):

- Calculate $\mathrm{t}(u, v)$
- If distance estimate $\mathrm{d}(u, v) \neq \mathrm{t}(u, v)$: update it
  - For every $y \in V \setminus \{u, v\}$:
    Update(y, v) and Update(u, y)

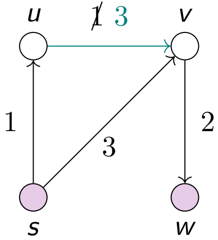


	s	u	v	w
s	0	1	$\mathrm{r}(3)$	$\mathrm{r}(2\,\mathrm{r}(2))$
u	∞	0	$\mathrm{r}(3)$	$\mathrm{r}(\mathrm{r}(2) + \mathrm{r}(3))$
v	∞	∞	0	$\mathrm{r}(2)$
w	∞	∞	∞	0

Approximate APSP - Update

Update(u, v):

- Calculate $\mathrm{t}(u, v)$
- If distance estimate $\mathrm{d}(u, v) \neq \mathrm{t}(u, v)$: update it
  - For every $y \in V \setminus \{u, v\}$:
    Update(y, v) and Update(u, y)



	s	u	v	w
s	0	1	$\mathrm{r}(3)$	$\mathrm{r}(\mathrm{r}(2) + \mathrm{r}(3))$
u	∞	0	$\mathrm{r}(3)$	$\mathrm{r}(\mathrm{r}(2) + \mathrm{r}(3))$
v	∞	∞	0	$\mathrm{r}(2)$
w	∞	∞	∞	0

Approximate APSP

- Eventually no distance estimate left to update
- invariant: $\mathrm{d}(u, v) \leq \mathrm{t}(u, v)$ at all times and $d(u, v) = \mathrm{t}(u, v)$ after Update procedure stops
- weights only increase or edges deleted: $\mathrm{t}(u, v)$ can only become larger or stay the same
- when $\mathrm{d}(u, v)$ is not (yet) reset: $\mathrm{d}(u, v) \leq \mathrm{t}(u, v)$
  Update(u,v) sets $\mathrm{d}(u, v)$ to $\mathrm{t}(u, v)$
- path from y to v contains path $u \rightarrow v$, $\mathrm{d}(y, v)$ is also updated and set to $\mathrm{t}(y, v)$
  Similar for a path that begins with $u \rightarrow v$

Approximation

Repeated use of $\mathrm{r}_{1+\epsilon}$: not a $(1+\epsilon)$-approximation
Specifically:

Lemma 1

Let G be a non-negatively weighted directed graph.
If $\mathrm{d} : V \times V \to \mathbb{R} \cup \{\infty\}$ satisfies the following:

1. $\mathrm{d}(v, v) = 0$ for all $v \in V$
2. $0 \leq \mathrm{d}(u, v) = \mathrm{t}(u, v)$ for all $u, v \in V$ such that $u \neq v$

Then for any $u, v \in V$ and any integer $h \geq 0$, we have
$\delta_G(u, v) \leq \mathrm{d}(u, v) \leq (1+\epsilon)^{\lceil \log_2 h \rceil + 1} \delta_G^h(u, v)$

where $\delta_G^h(u, v)$ is the length of the shortest path from u to v with at most h edges

Approximation - Proof

For: $\delta_G(u, v) \leq \mathrm{d}(u, v)$
$\mathrm{d}(u, v) = \mathrm{t}(u, v)$ and $\mathrm{t}(u, v)$ cannot underestimate the actual distance

For $\mathrm{d}(u, v) < \infty$

- $\mathrm{d}(u, v) = \mathrm{r}_{1+\epsilon}(\mathrm{w}(uv)) \rightarrow$ edge uv is in G
- $\mathrm{d}(u, v) = \mathrm{r}_{1+\epsilon}(\mathrm{d}(u, w) + \mathrm{d}(w, v))$ for some w

  $\rightarrow$ path P_1 from u to w, P_2 from w to v
  $\rightarrow$ eventually break down into edges
  $\rightarrow$ rounding only makes the values larger

Approximation - Proof

For: $\delta_G(u, v) \leq \mathrm{d}(u, v)$
$\mathrm{d}(u, v) = \mathrm{t}(u, v)$ and $\mathrm{t}(u, v)$ cannot underestimate the actual distance

Show that: $\mathrm{d}(u, v) \leq (1 + \epsilon)^{\lceil \log_2 h \rceil + 1} \delta_G^h(u, v)$ by induction on h
Assume: $\delta_G^h(u, v) \leq \infty$

h = 1:

Edge uv is in G and therefore $\delta_G^h(u, v) \leq w(uv)$

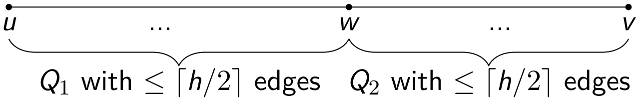
By definition of $\mathrm{t}(u, v)$ and (2) we have that:

$$d(u, v) \leq \mathrm{r}_{1+\epsilon}(w(uv)) \leq (1 + \epsilon) w(uv)$$
$$= (1 + \epsilon)^{\lceil \log_2 h \rceil + 1} \delta_G^h(u, v)$$

Approximation - Proof

Show that: $\mathrm{d}(u, v) \leq (1+\epsilon)^{\lceil \log_2 h \rceil + 1} \delta_G^h(u, v)$ by induction on h
Assume: $\delta_G^h(u, v) \leq \infty$

$h \geq 2$:
Path Q with h edges:



By IH we get:

$$
\begin{aligned}
d(u, w) &\leq (1+\epsilon)^{\lceil \log_2 \lceil h/2 \rceil \rceil + 1} \delta_G^{\lceil h/2 \rceil}(u, v) \\
&\leq (1+\epsilon)^{\lceil \log_2 \lceil h/2 \rceil \rceil + 1} \operatorname{length}(Q_1)
\end{aligned}
$$

Approximation - Proof

Since $h \geq 2$:

$$\begin{aligned} \mathrm{d}(u,w) + \mathrm{d}(w,v) &\leq (1+\epsilon)^{\lceil \log_2 h/2 \rceil + 1} (\mathrm{length}(Q_1) + \mathrm{length}(Q_2)) \\ &\leq (1+\epsilon)^{\lceil \log_2 h \rceil} \mathrm{length}(Q) \end{aligned}$$

Also:

$$\begin{aligned} d(u,v) &\leq \mathrm{r}_{1+\epsilon}(\mathrm{d}(u,w) + \mathrm{d}(w,v)) \\ &\leq (1+\epsilon)(\mathrm{d}(u,w) + \mathrm{d}(w,v)) \\ &\leq (1+\epsilon)^{\lceil \log_2 h \rceil + 1} \mathrm{length}(Q) \end{aligned}$$

Also holds for shortest path with at most h edges between u and v.

$$d(u,v) \leq (1+\epsilon)^{\lceil \log_2 h \rceil + 1} \delta_G^h(u,v)$$

$(1+\epsilon)$ - Approximation

Approximation depends on the number of hops $h \leq n$ we allow.

To get $(1+\epsilon)$-approximation:

Let $\epsilon' = \frac{\epsilon}{2\lceil(\log_2 n)\rceil}$

$$\left(1 + \frac{\epsilon}{2\left\lceil(\log_2 n)\right\rceil}\right)^{\lceil\log_2 n\rceil+1} \leq e^{\epsilon/2}$$

and since $\epsilon \in (0,1)$

$$\leq 1 + \epsilon$$

Computing Minima

Recomputing $\mathrm{t}(u, v)$ every time a distance between two vertices might have changed $\rightarrow$ Not very efficient!
Instead: store an approximation $\mathrm{t}'(u, v)$ along with an index $\beta(u, v)$

- order vertices $w_1, ..., w_n$ in some way
- remember first index i for which:

$$\mathrm{r}_{1+\epsilon}(\mathrm{d}(u, w_i) + \mathrm{d}(w_i, v)) = \mathrm{t}'(u, v)$$

- reevaluating $\mathrm{t}'(u, v)$:
  - if $\mathrm{r}_{1+\epsilon}(\mathrm{w}(uv)) = \mathrm{t}'(u, v) \rightarrow \mathrm{t}'(u, v)$ stays the same
  - look for alternative path that lets us keep distance $\mathrm{t}'(u, v)$: Only need to look at indices $j \geq \beta(u, v)$
  - if estimated length of shortest path actually changed: recompute $\mathrm{t}'(u, v)$

Total Update Time

How often can $\mathrm{d}(u, v)$ change?

- for a $\mathrm{d}(u, v) < \infty$: $\mathrm{d}(u, v) \leq \mathrm{t}(u, v) < (1 + \epsilon')^{\lceil \log_2 n \rceil + 2} nW$
- $\mathrm{d}(u, v)$ can only increase
- $\mathrm{d}(u, v)$ always non-negative integral power of $(1 + \epsilon')$

Total Update Time

How often can $\mathrm{d}(u, v)$ change?

- for a $\mathrm{d}(u, v) < \infty$: $\mathrm{d}(u, v) \leq \mathrm{t}(u, v) < (1 + \epsilon')^{\lceil \log_2 n \rceil + 2} nW$
- $\mathrm{d}(u, v)$ can only increase
- $\mathrm{d}(u, v)$ always non-negative integral power of $(1 + \epsilon')$
- changes:

$$\mathrm{O}\left(\log_{1+\epsilon'} (1 + \epsilon')^{\log_2 n} nW\right) = \mathrm{O}(\log(nW)/\epsilon')$$

Total Update Time

How often can $\mathrm{d}(u, v)$ change?

- for a $\mathrm{d}(u, v) < \infty$: $\mathrm{d}(u, v) \leq \mathrm{t}(u, v) < (1 + \epsilon')^{\lceil \log_2 n \rceil + 2} nW$
- $\mathrm{d}(u, v)$ can only increase
- $\mathrm{d}(u, v)$ always non-negative integral power of $(1 + \epsilon')$
- changes:

$$\mathrm{O}\left(\log_{1+\epsilon'} (1 + \epsilon')^{\log_2 n} nW\right) = \mathrm{O}(\log(nW)/\epsilon')$$

- whenever $\mathrm{d}(u, v)$ changes: $< 2n$ recursive calls to Update

Total Update Time

How often can $\mathrm{d}(u, v)$ change?

- for a $\mathrm{d}(u, v) < \infty$: $\mathrm{d}(u, v) \leq \mathrm{t}(u, v) < (1 + \epsilon')^{\lceil \log_2 n \rceil + 2} nW$
- $\mathrm{d}(u, v)$ can only increase
- $\mathrm{d}(u, v)$ always non-negative integral power of $(1 + \epsilon')$
- changes:

$$\mathrm{O}\left(\log_{1+\epsilon'} (1 + \epsilon')^{\log_2 n} nW\right) = \mathrm{O}(\log(nW)/\epsilon')$$

- whenever $\mathrm{d}(u, v)$ changes: $< 2n$ recursive calls to Update
- total update time:

$$\mathrm{O}(n^3 \log(nW)/\epsilon' + \Delta)$$

Total Update Time

How often can $\mathrm{d}(u, v)$ change?

- for a $\mathrm{d}(u, v) < \infty$: $\mathrm{d}(u, v) \leq \mathrm{t}(u, v) < (1 + \epsilon')^{\lceil \log_2 n \rceil + 2} nW$
- $\mathrm{d}(u, v)$ can only increase
- $\mathrm{d}(u, v)$ always non-negative integral power of $(1 + \epsilon')$
- changes:

$$\mathrm{O}\left(\log_{1+\epsilon'} (1 + \epsilon')^{\log_2 n} nW\right) = \mathrm{O}(\log(nW)/\epsilon')$$

- whenever $\mathrm{d}(u, v)$ changes: $< 2n$ recursive calls to Update
- total update time:

$$\mathrm{O}(n^3 \log(nW)/\epsilon' + \Delta) = \mathrm{O}(n^3 \log n \log(nW)/\epsilon + \Delta)$$

Computing Minima

In each call to Update(u, v) compute $\mathrm{t}(u, v)$
3 cases:

- relevant path not affected by changes (no change to $\mathrm{t}'(u, v)$ and $\beta(u, v)$)
  $\rightarrow \mathrm{O}(1)$

Computing Minima

In each call to Update(u, v) compute $\mathrm{t}(u, v)$
3 cases:

- relevant path not affected by changes (no change to $\mathrm{t}'(u, v)$ and $\beta(u, v)$)
  $\rightarrow \mathrm{O}(1)$
- alternative path can be found (only $\beta(u, v)$ increases)
  $\rightarrow$ at most n times before $\mathrm{t}'(u, v)$ also changes
- $\mathrm{t}'(u, v)$ is updated:

Computing Minima

In each call to Update(u, v) compute $\mathrm{t}(u, v)$
3 cases:

- relevant path not affected by changes (no change to $\mathrm{t}'(u, v)$ and $\beta(u, v)$)
  $\rightarrow \mathrm{O}(1)$
- alternative path can be found (only $\beta(u, v)$ increases)
  $\rightarrow$ at most n times before $\mathrm{t}'(u, v)$ also changes
- $\mathrm{t}'(u, v)$ is updated:
  $\rightarrow \mathrm{O}(\log(nW)/\epsilon')$ times
- total cost to compute $\mathrm{t}(u, v)$:

$$\mathrm{O}(n \log(nW)/\epsilon')$$

Thank you for your attention!