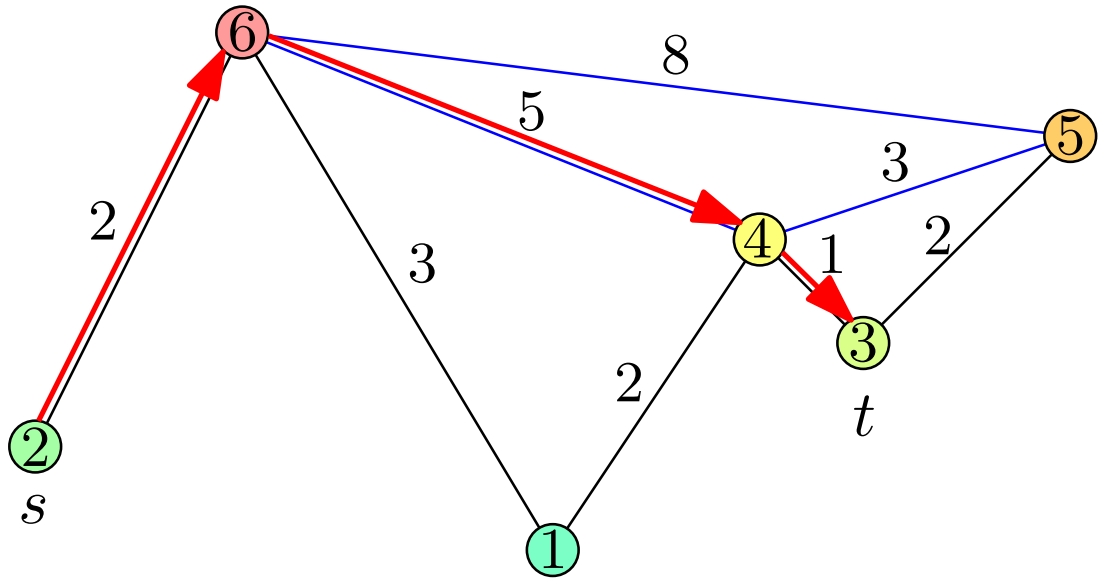
Karlsruhe Institute of Technology

Heuristic Contraction Hierarchies with Approximation Guarantee

Robert Geisberger – *geisberger@kit.edu*
Dennis Schieferdecker – *schieferdecker@kit.edu*

Institute for Theoretical Informatics, Algorithms II



Motivation

My domain: Route planning in road networks.

- Input: Graph $G = (V, E)$ with edge weight function $c : E \to \mathbb{R}_+$.
- Task: Compute the shortest path distance $d(s, t)$.

Previous work:

- Perform a preprocessing step.
- Several algorithms with different tradeoffs of preprocessing time, preprocessing space, and query speedup.

algorithm	space [B/n]	preproc. [min]	speedup
Dijkstra	0	0	1
contraction hierarchies	-4	32	41 041
transit nodes + edge flags	320	229	3 327 372

Motivation

Arising Question:

- Can we transfer these results to other graph classes?

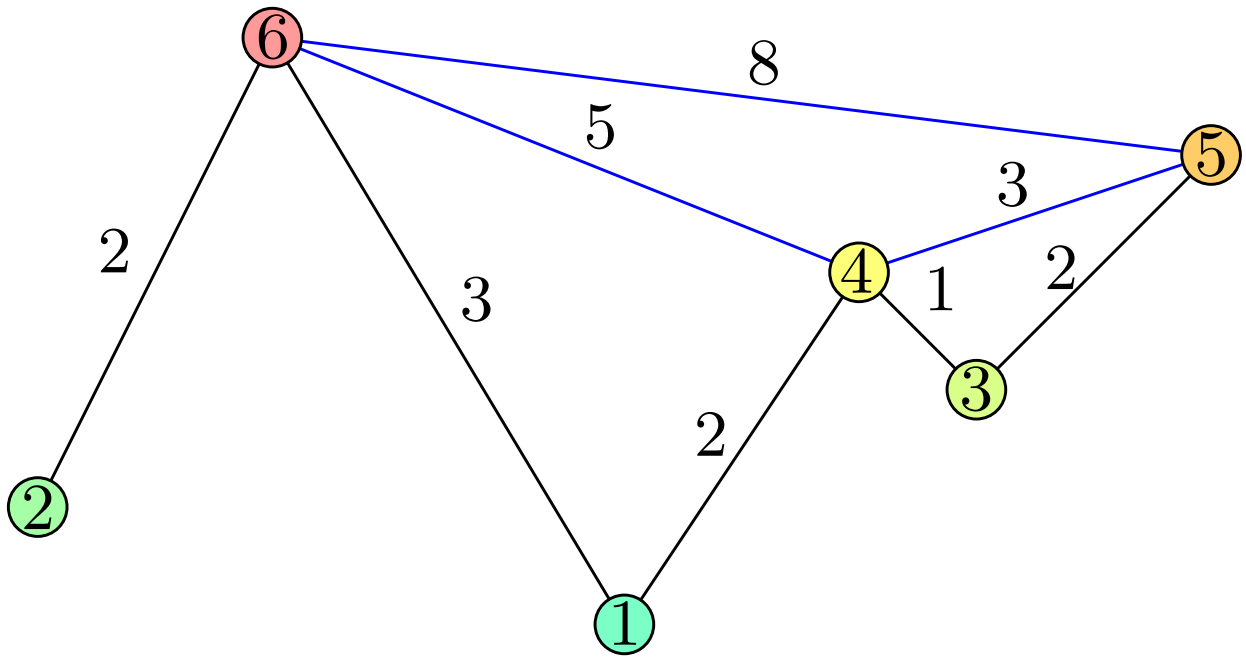
Problems:

- We just observe good performance.
- We miss hard theory explaining the good performance.
- Some of our algorithms have in the worst case a runtime worse than Dijkstra on arbitrary graphs.

Our contribution:

- We applied our algorithms on other promising graph classes, observed their performance, and looked for potential optimizations.
- The result is an approximate version of contraction hierarchies that improves the performance on certain graph classes.

Contraction Hierarchies (CH)



Related Work

Contraction Hierarchies (CH) [WEA 2008]

- CH is based on the concept of node contraction: removing a node and adding shortcuts to preserve shortest paths distances.

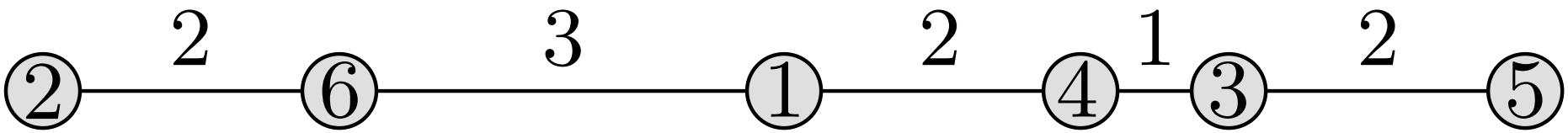
in more detail:

- Order nodes by "importance", $V = \{1, 2, \ldots, n\}$.
- Contract nodes in this order, node v is contracted by

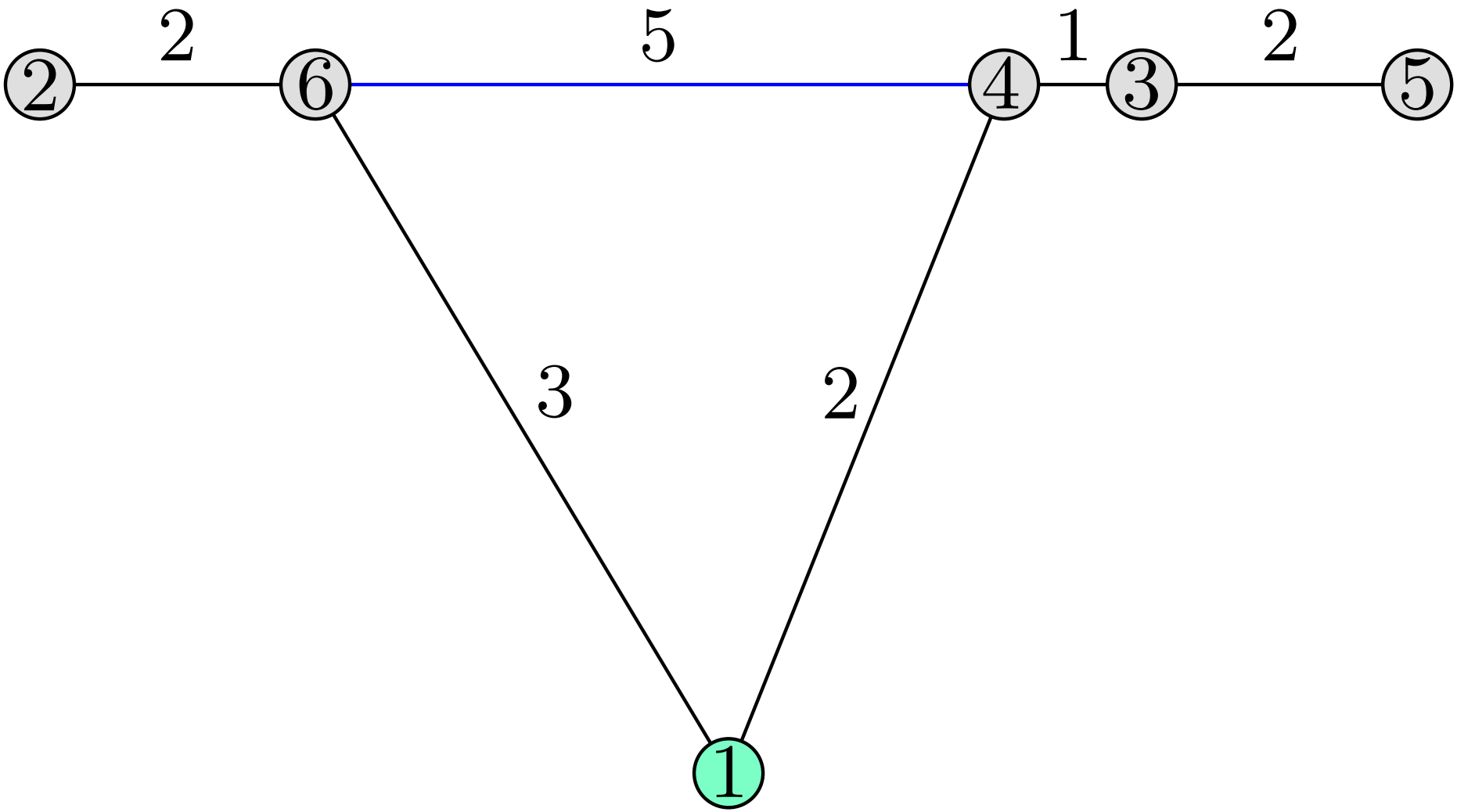
  foreach *pair* (u, v) *and* (v, w) *of edges* **do**
  - **if** $\langle u, v, w \rangle$ *is a unique shortest path* **then**
    - add shortcut (u, w) with weight $c(\langle u, v, w \rangle)$

- Query relaxes only edges to more "important" nodes.
  $\Rightarrow$ valid due to shortcuts

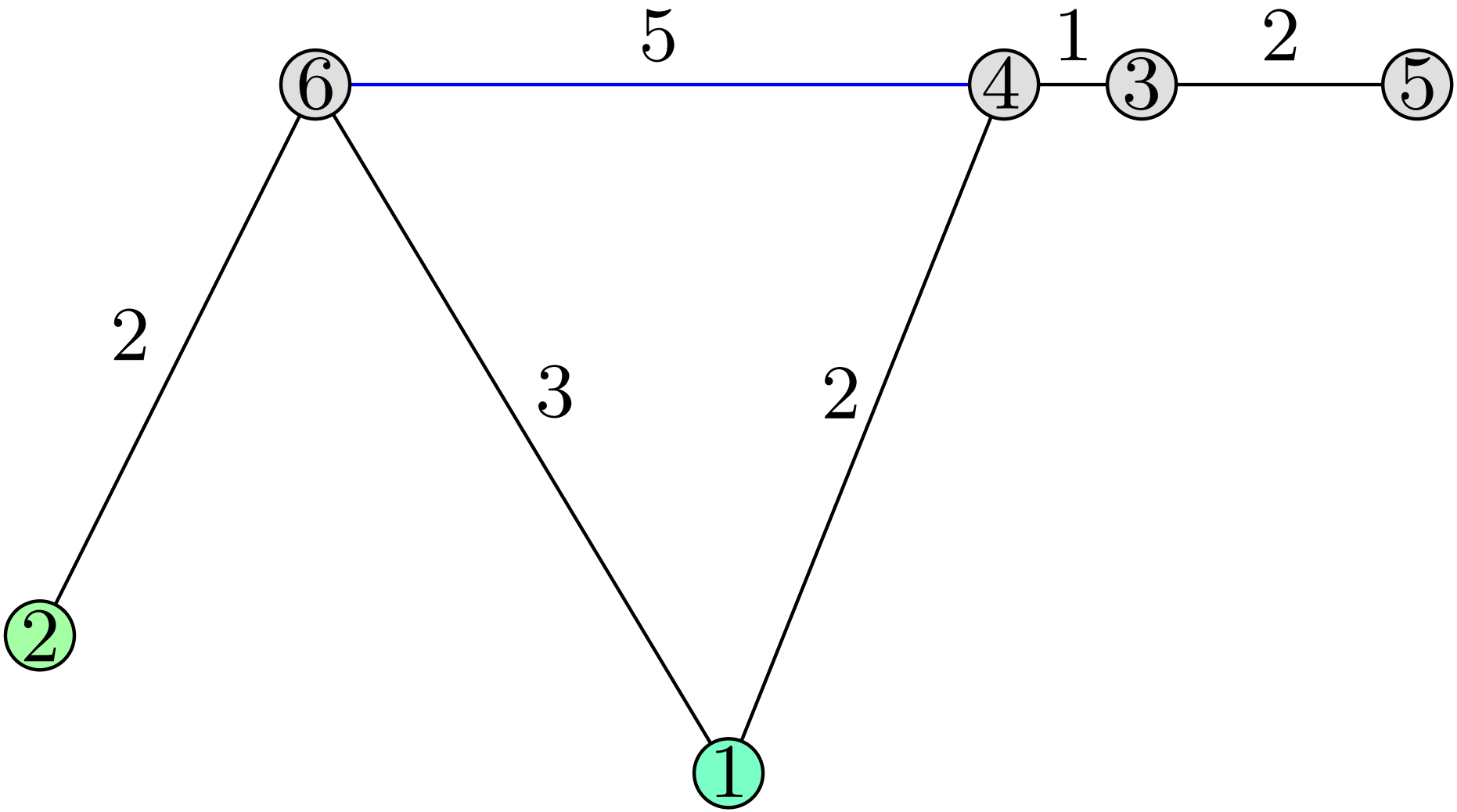
Example: Construction



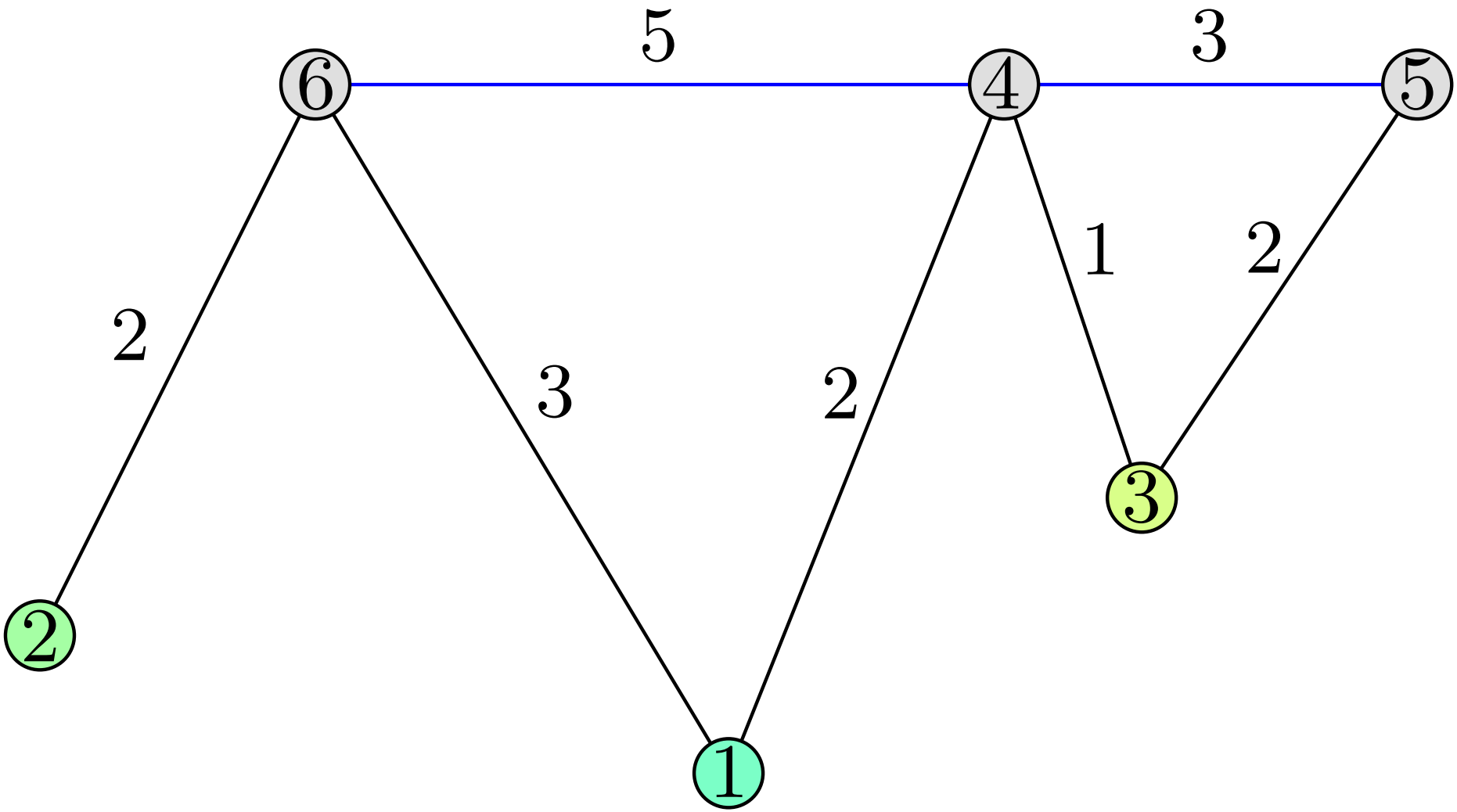
Example: Construction



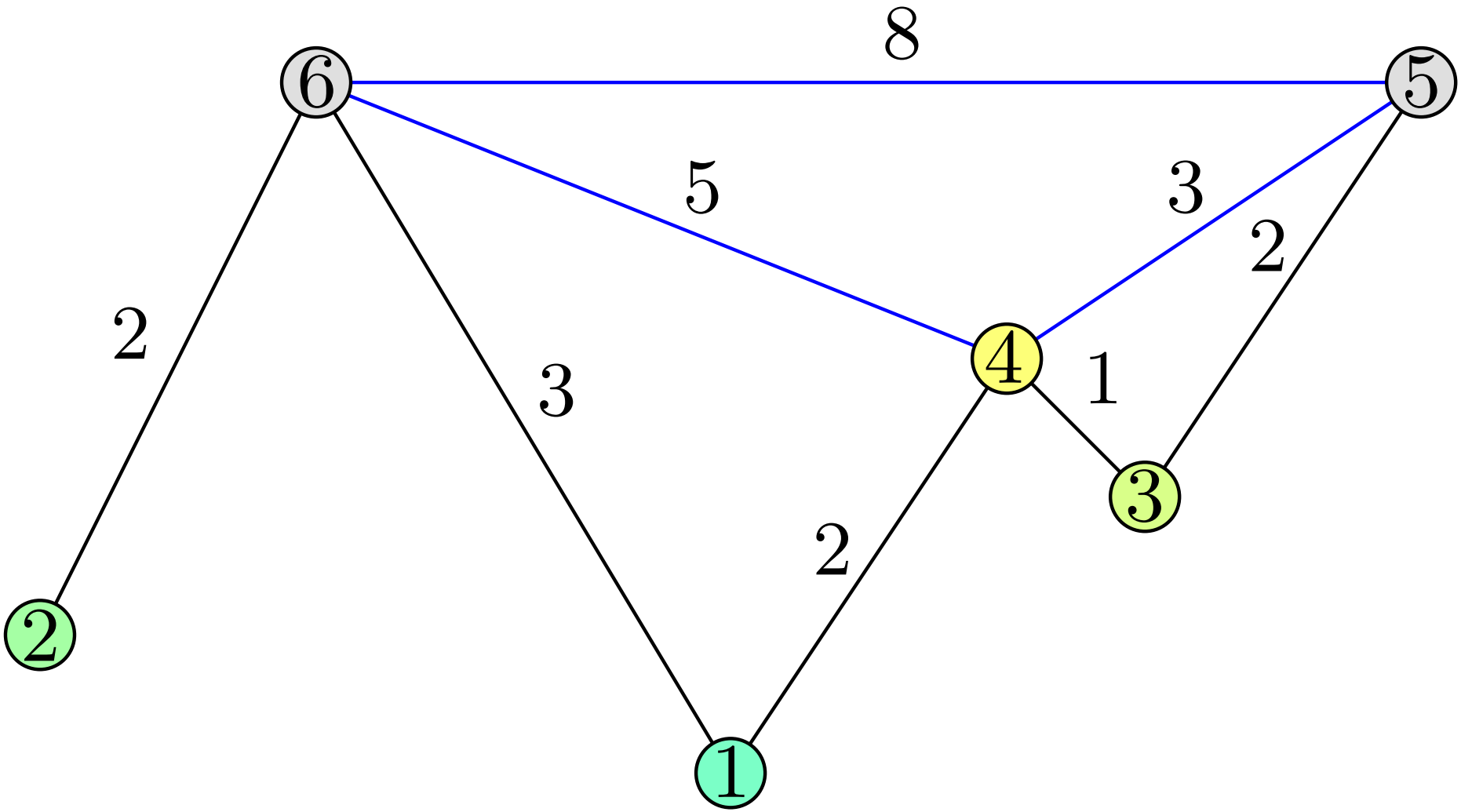
Example: Construction



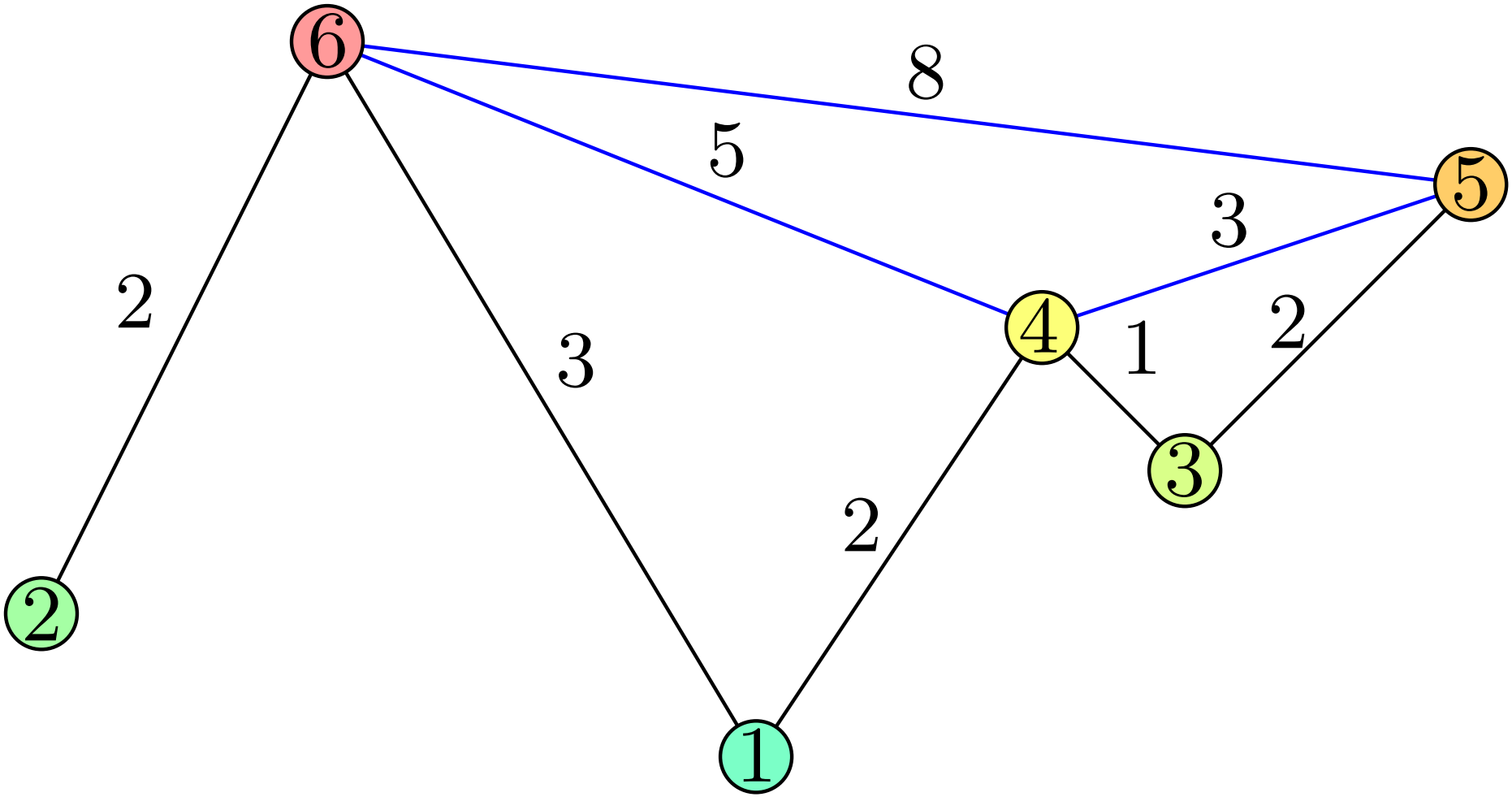
Example: Construction



Example: Construction



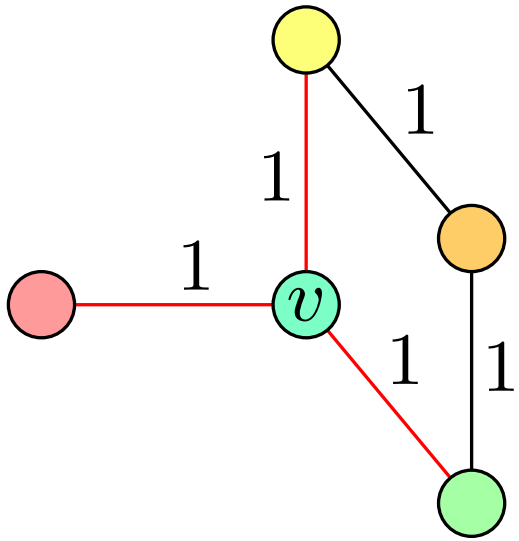
Example: Construction



Construction

To identify necessary shortcuts,

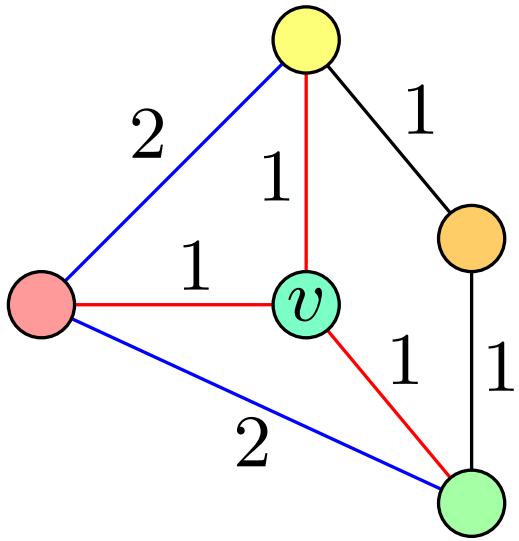
- perform local witness search from node u with incoming edge (u, v),
- ignore node v at search,
- and add shortcut (u, w) iff no witness is found or $c(\text{witness}) > c(u, v) + c(v, w)$.



Construction

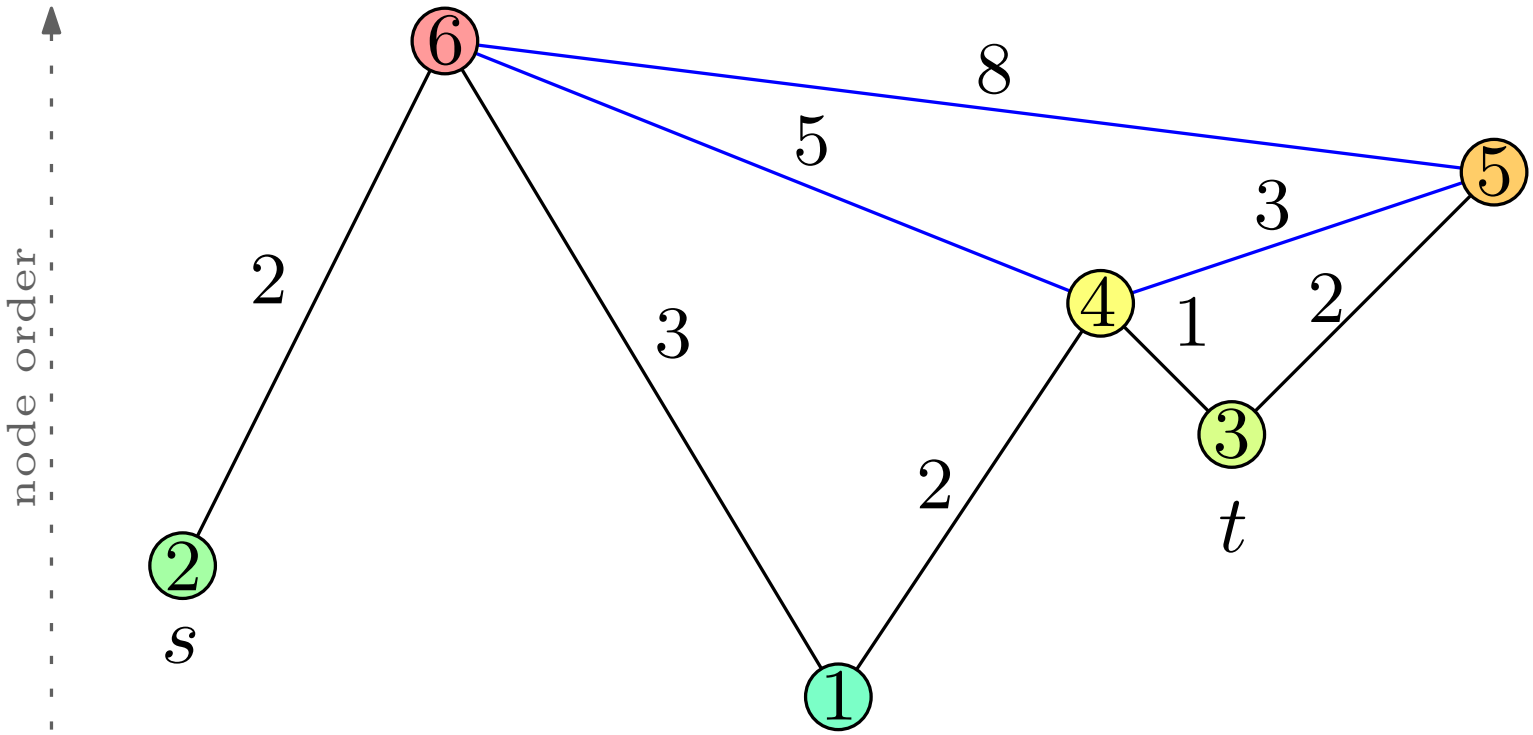
To identify necessary shortcuts,

- perform local witness search from node u with incoming edge (u, v),
- ignore node v at search,
- and add shortcut (u, w) iff no witness is found or $c(\text{witness}) > c(u, v) + c(v, w)$.



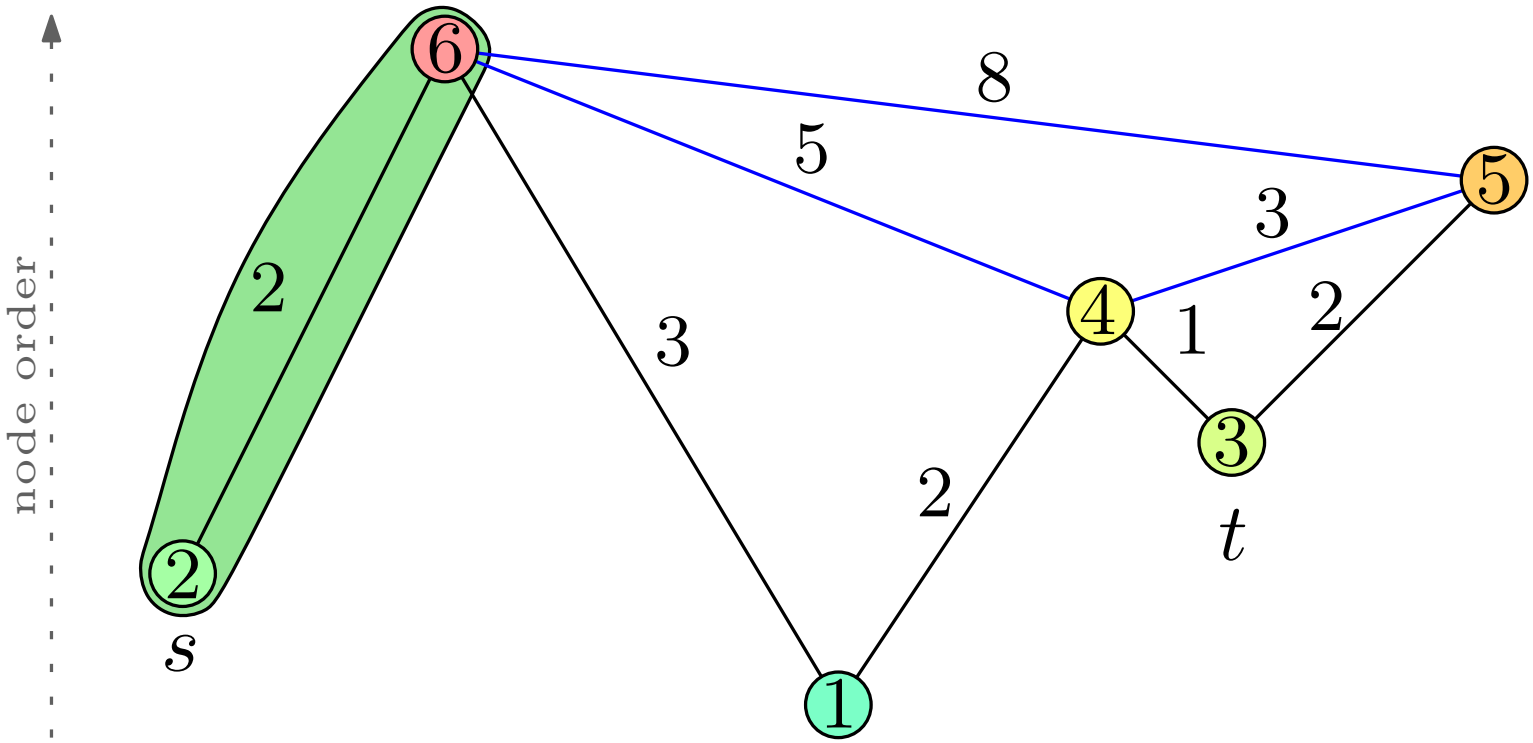
Query

- The query is a modified bidirectional Dijkstra algorithm.
- upward graph $G^{\uparrow} := (V, E^{\uparrow})$ with $E^{\uparrow} := \{(u, v) \in E \mid u < v\}$
  downward graph $G^{\downarrow} := (V, E^{\downarrow})$ with $E^{\downarrow} := \{(u, v) \in E \mid u > v\}$
- We perform a forward search in $G^{\uparrow}$ and a backward search in $G^{\downarrow}$.



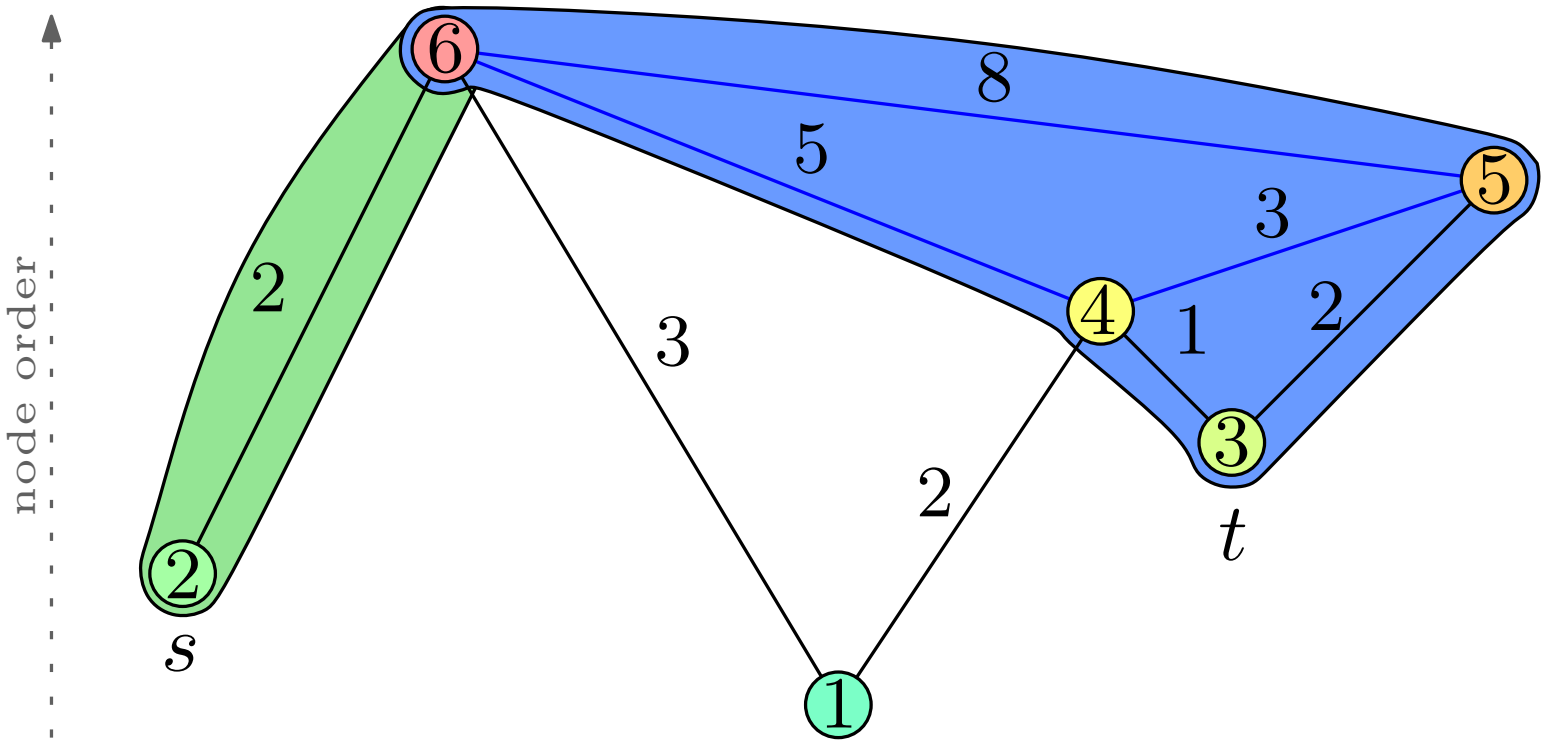
Query

- The query is a modified bidirectional Dijkstra algorithm.
- upward graph $G^{\uparrow} := (V, E^{\uparrow})$ with $E^{\uparrow} := \{(u, v) \in E \mid u < v\}$
  downward graph $G^{\downarrow} := (V, E^{\downarrow})$ with $E^{\downarrow} := \{(u, v) \in E \mid u > v\}$
- We perform a forward search in $G^{\uparrow}$ and a backward search in $G^{\downarrow}$.



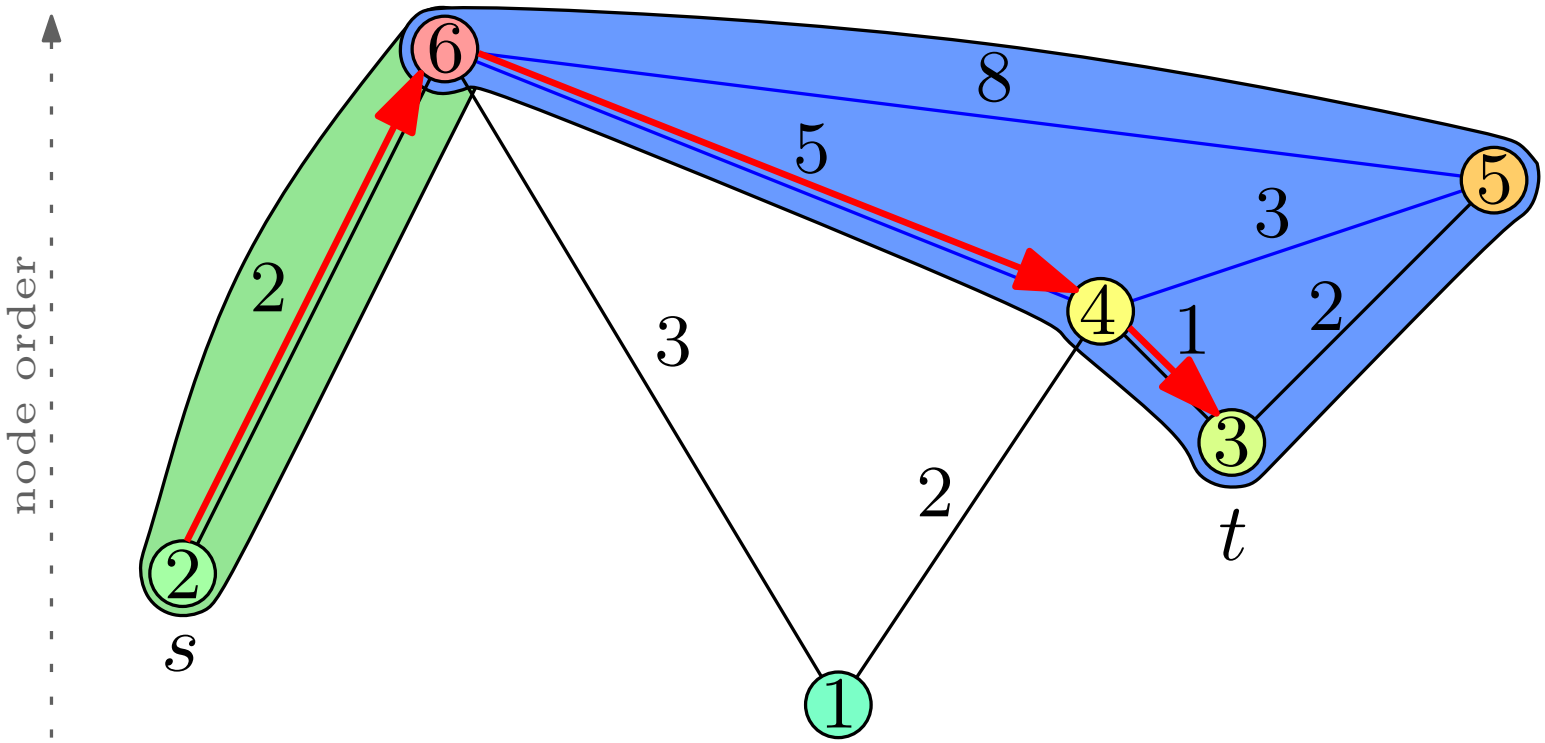
Query

- The query is a modified bidirectional Dijkstra algorithm.
- upward graph $G^{\uparrow} := (V, E^{\uparrow})$ with $E^{\uparrow} := \{(u, v) \in E \mid u < v\}$
  downward graph $G^{\downarrow} := (V, E^{\downarrow})$ with $E^{\downarrow} := \{(u, v) \in E \mid u > v\}$
- We perform a forward search in $G^{\uparrow}$ and a backward search in $G^{\downarrow}$.



Query

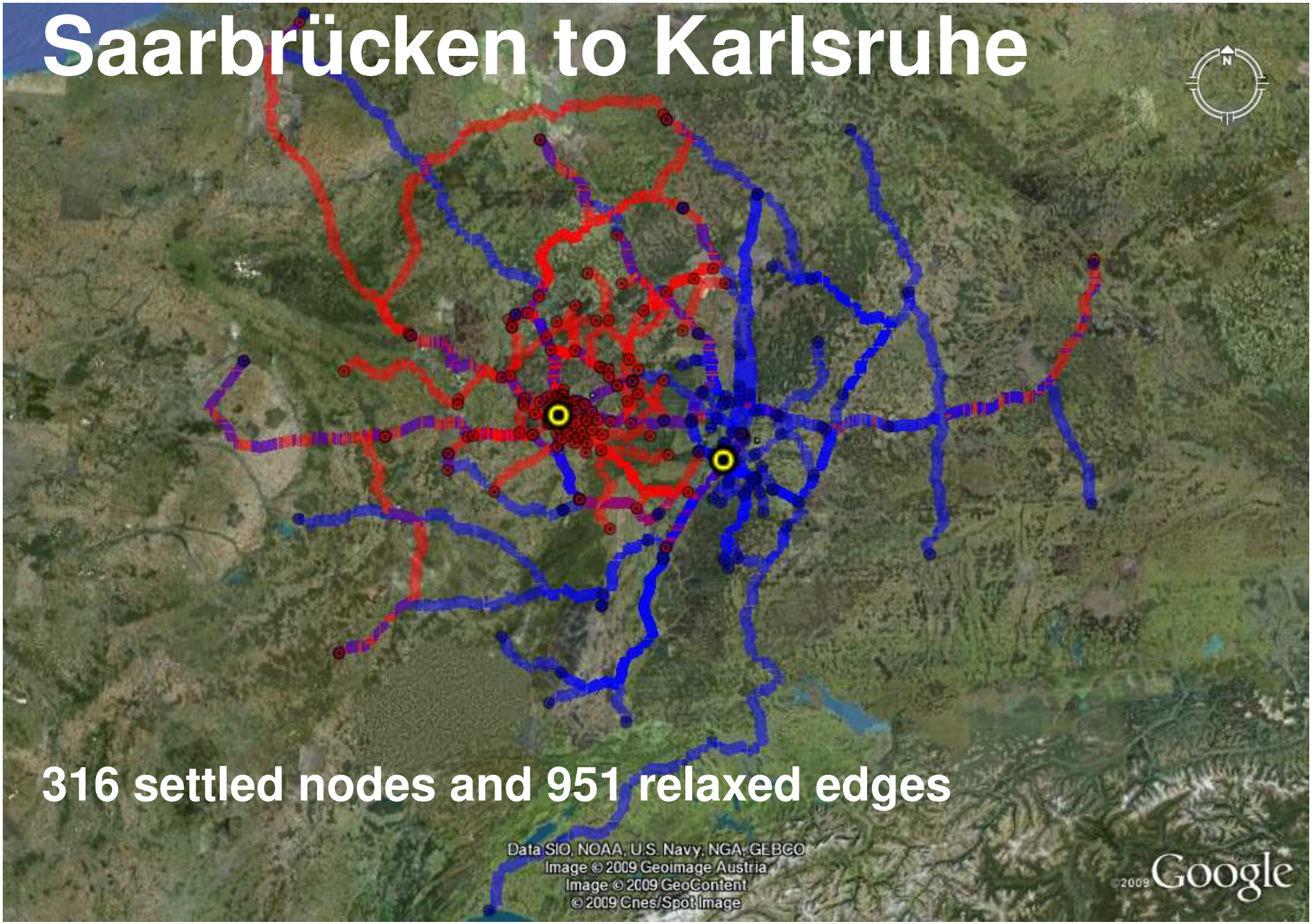
- The query is a modified bidirectional Dijkstra algorithm.
- upward graph $G^{\uparrow} := (V, E^{\uparrow})$ with $E^{\uparrow} := \{(u, v) \in E \mid u < v\}$
  downward graph $G^{\downarrow} := (V, E^{\downarrow})$ with $E^{\downarrow} := \{(u, v) \in E \mid u > v\}$
- We perform a forward search in $G^{\uparrow}$ and a backward search in $G^{\downarrow}$.



Saarbrücken to Karlsruhe

299 edges compressed to 13 shortcuts.

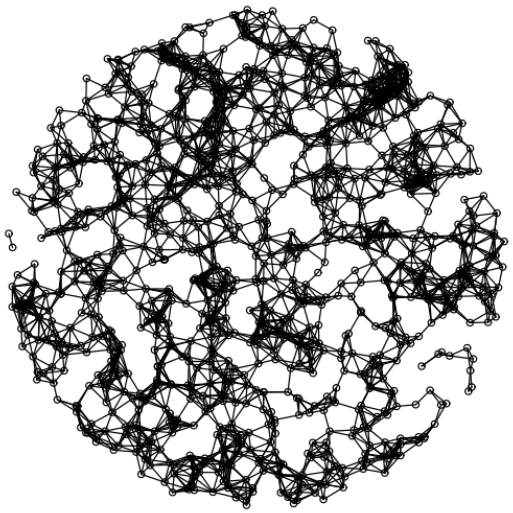
Saarbrücken to Karlsruhe



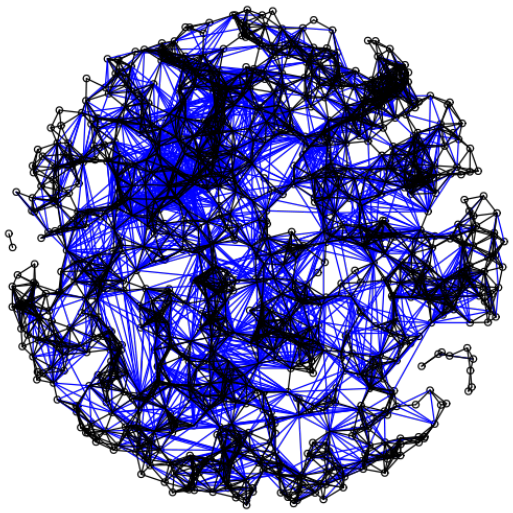
Our Contribution

Approximate Contraction Hierachies (apxCH)

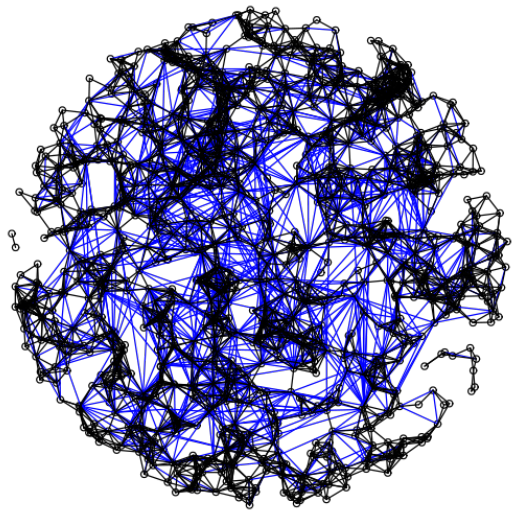
- Path found by query algorithm has a maximum stretch of $(1 + \varepsilon)$.
- Improves performance on certain instances.



(a) input



(b) CH

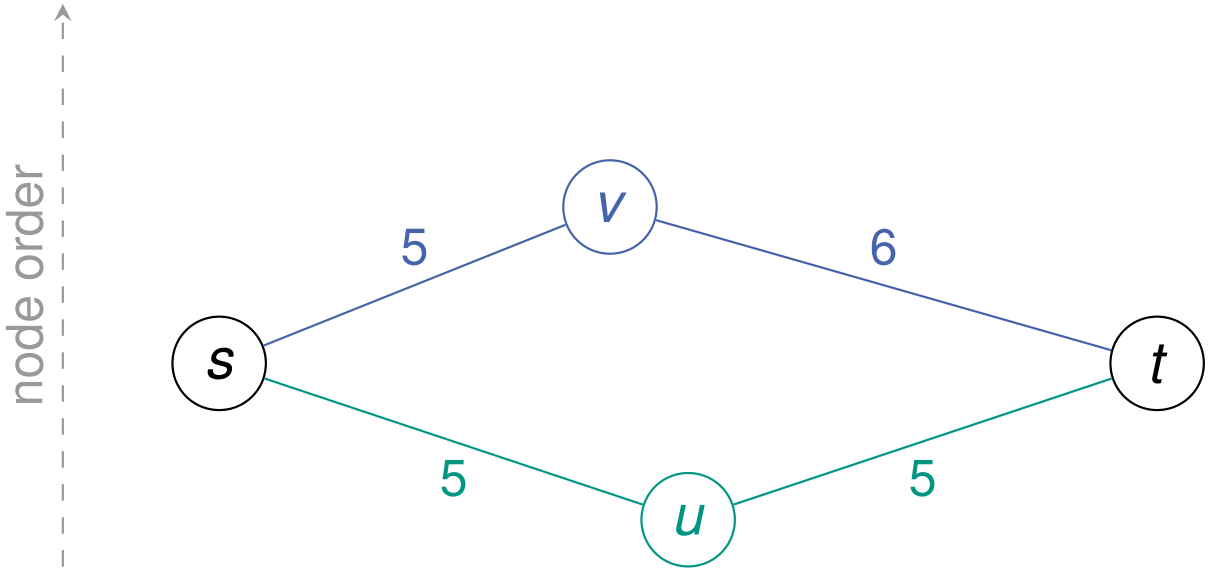


(c) apxCH-10%

Approximate Contraction Hierarchies

Basic idea

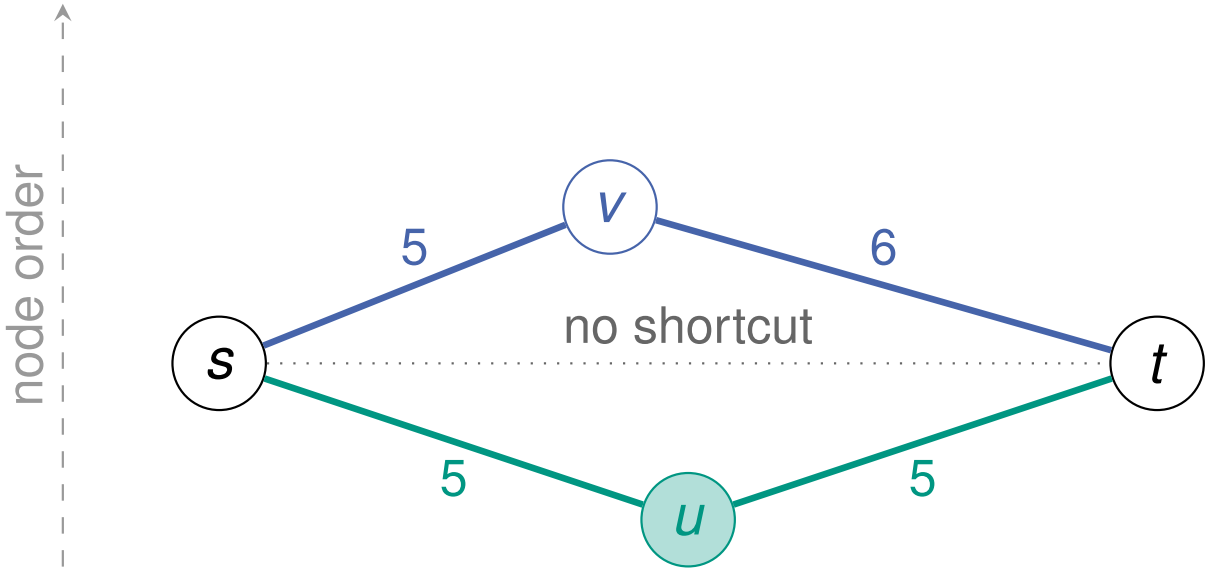
- Avoid a shortcut, even when a witness is $(1 + \varepsilon)$ times longer.



Approximate Contraction Hierarchies

Basic idea

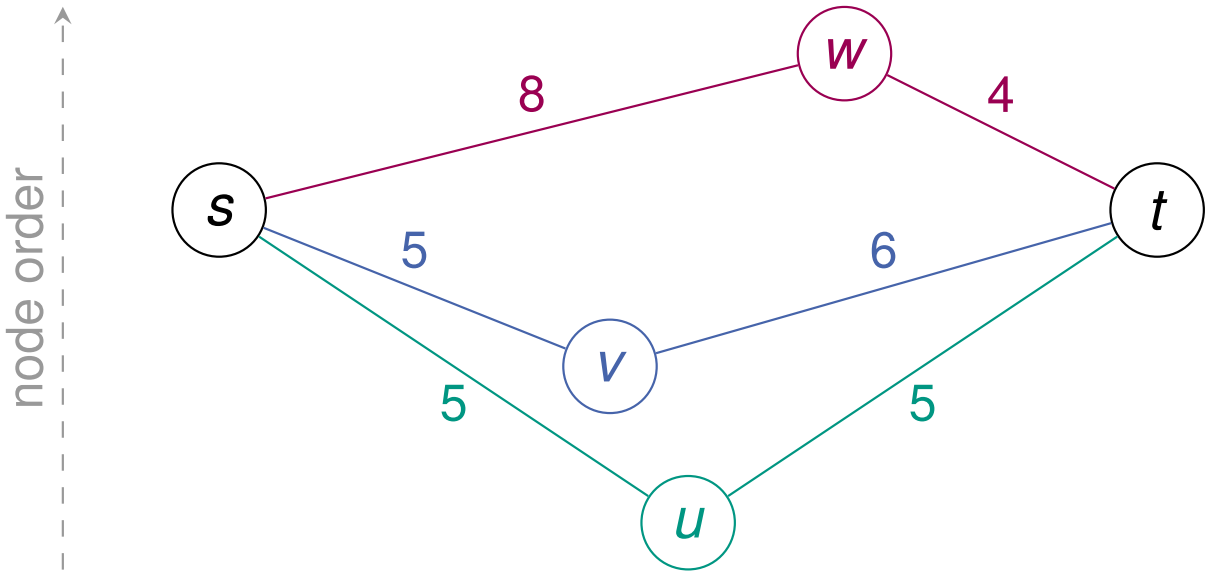
- Avoid a shortcut, even when a witness is $(1 + \varepsilon)$ times longer.



Approximate Contraction Hierarchies

Basic idea

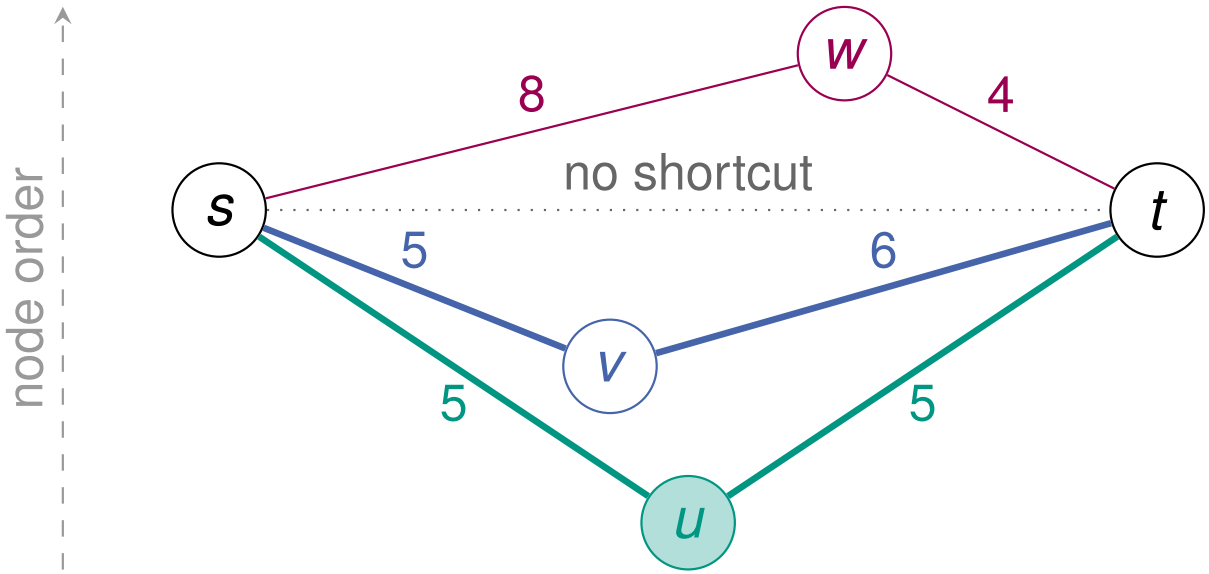
- Avoid a shortcut, even when a witness is $(1 + \varepsilon)$ times longer.
- But: straightforward implementation can cause errors to stack.



Approximate Contraction Hierarchies

Basic idea

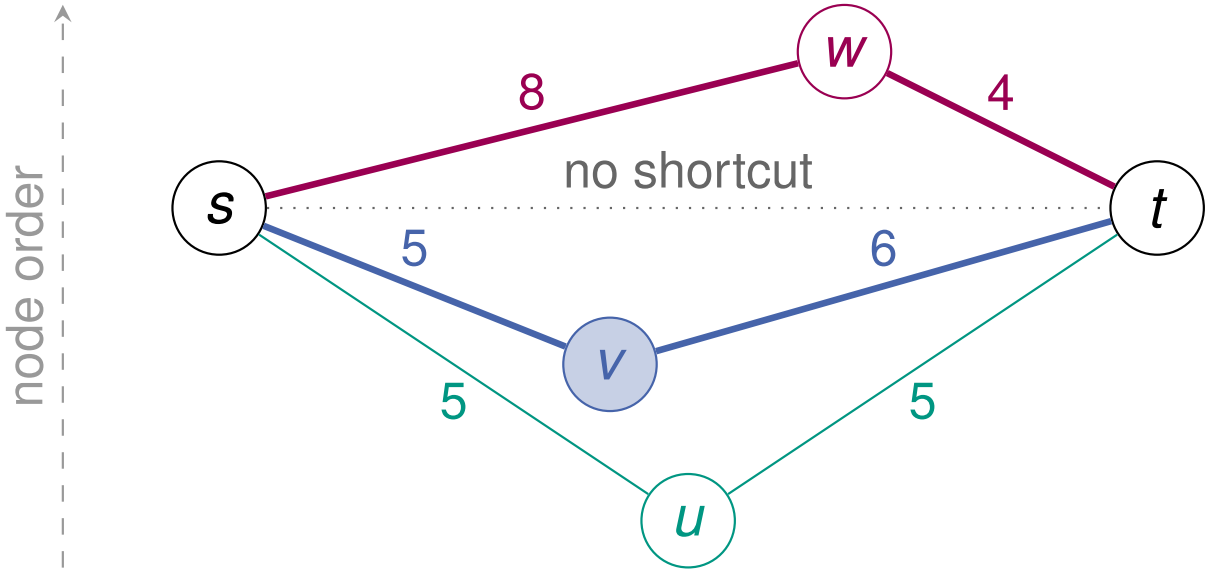
- Avoid a shortcut, even when a witness is $(1+\varepsilon)$ times longer.
- But: straightforward implementation can cause errors to stack.



Approximate Contraction Hierarchies

Basic idea

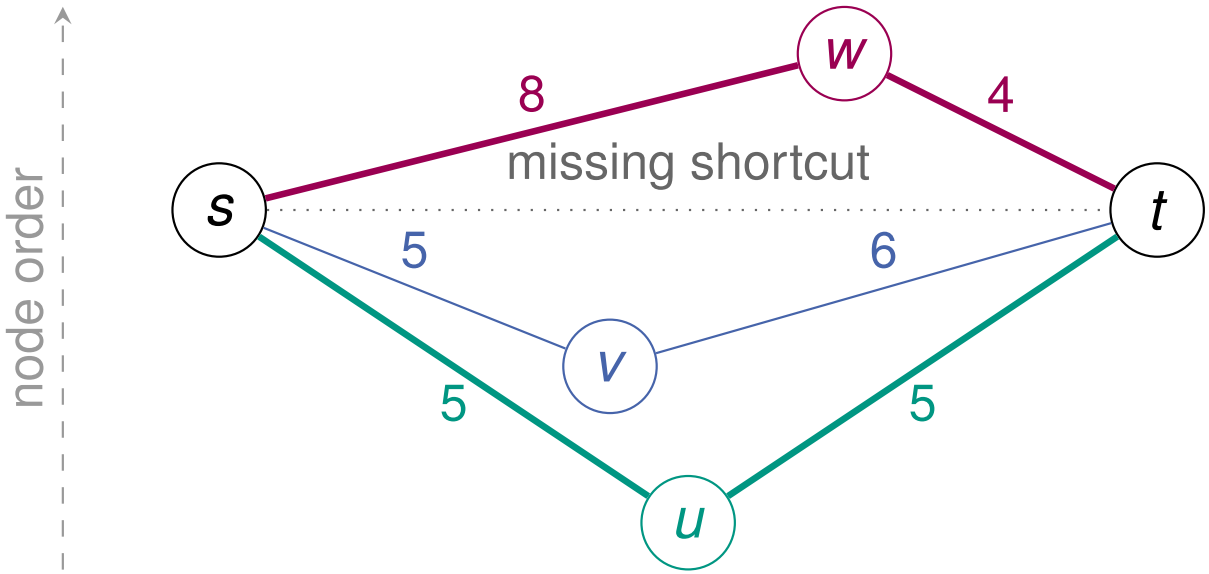
- Avoid a shortcut, even when a witness is $(1 + \varepsilon)$ times longer.
- But: straightforward implementation can cause errors to stack.



Approximate Contraction Hierarchies

Basic idea

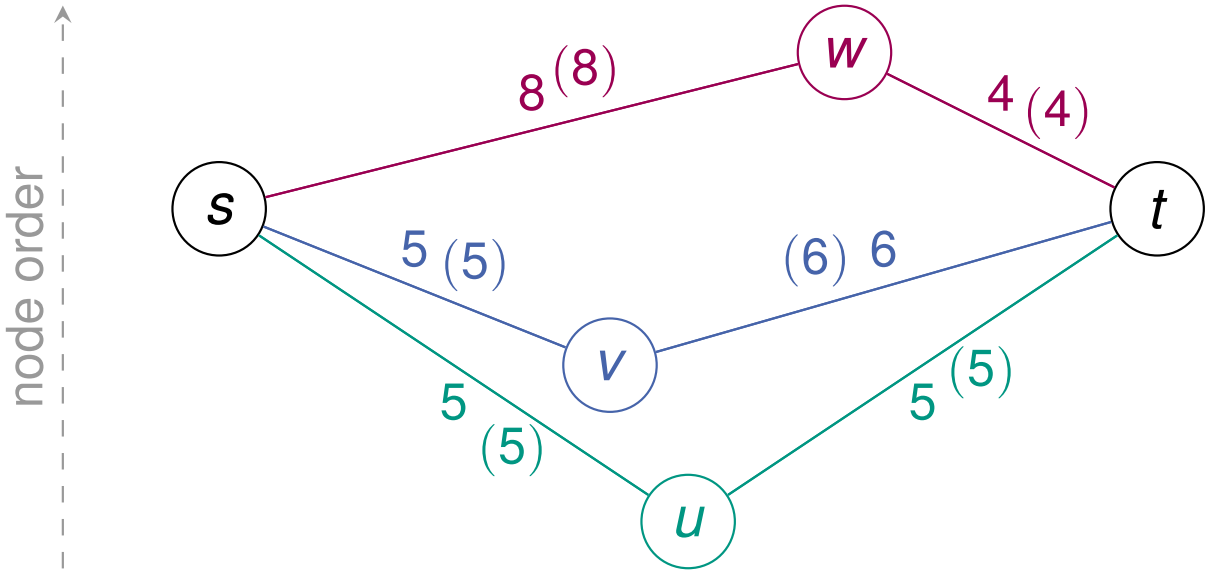
- Avoid a shortcut, even when a witness is $(1 + \varepsilon)$ times longer.
- But: straightforward implementation can cause errors to stack.



Approximate Contraction Hierarchies

Details

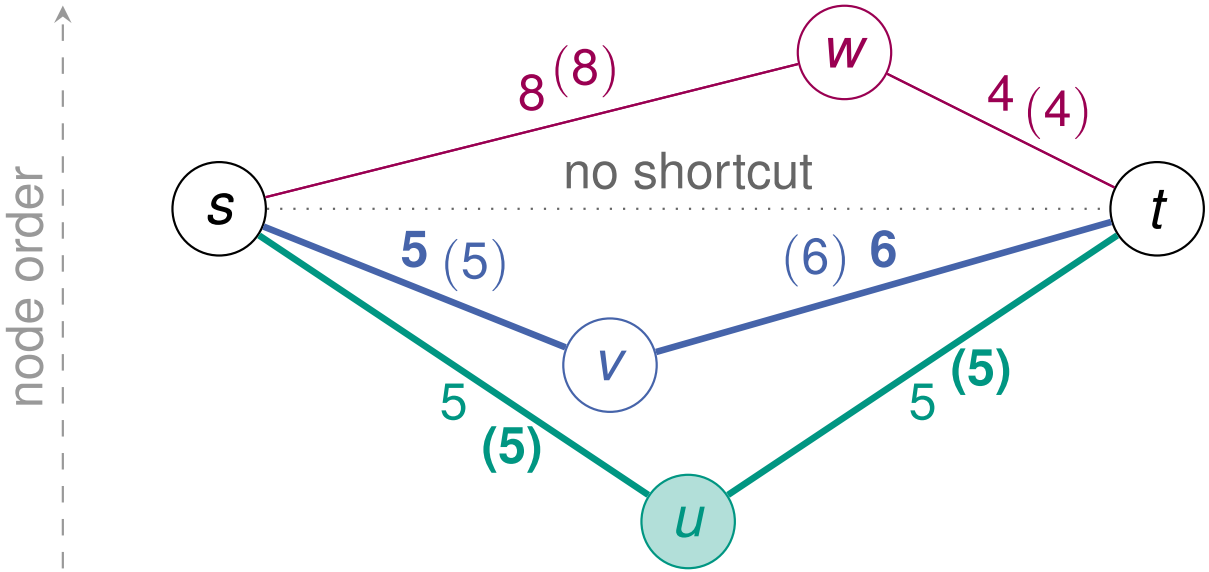
- Store a second edge weight $\tilde{c}$ with each edge (initially $\tilde{c} := c$).
- Add shortcut iff $c(\text{witness}) > (1+\varepsilon)\tilde{c}(\text{shortcut})$.
- Update $\tilde{c}(x,y) := \min\left\{\tilde{c}(x,y), \tilde{c}(\text{shortcut})\frac{c(x,y)}{c(\text{witness})}\right\}$ for all edges (x,y) on the witness path that prevented the shortcut.
  $\Rightarrow \tilde{c}(\text{witness}) \leq \tilde{c}(\text{shortcut})$



Approximate Contraction Hierarchies

Details

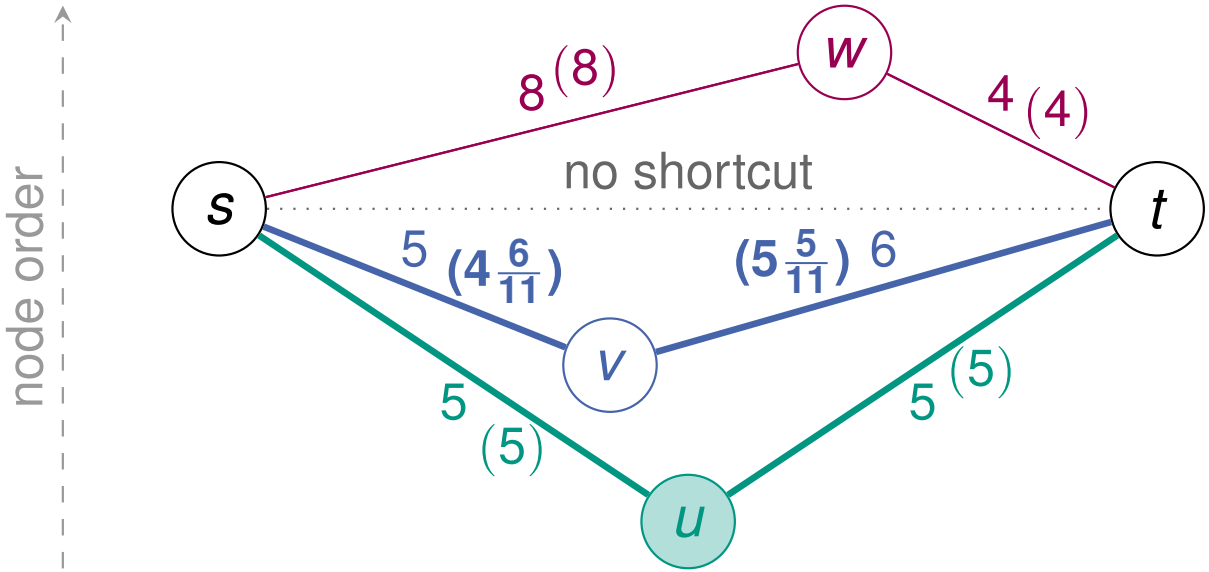
- Store a second edge weight $\tilde{c}$ with each edge (initially $\tilde{c} := c$).
- Add shortcut iff $c(\text{witness}) > (1+\varepsilon)\tilde{c}(\text{shortcut})$.
- Update $\tilde{c}(x,y) := \min\left\{\tilde{c}(x,y), \tilde{c}(\text{shortcut})\frac{c(x,y)}{c(\text{witness})}\right\}$ for all edges (x,y) on the witness path that prevented the shortcut.
  $\Rightarrow \tilde{c}(\text{witness}) \leq \tilde{c}(\text{shortcut})$



Approximate Contraction Hierarchies

Details

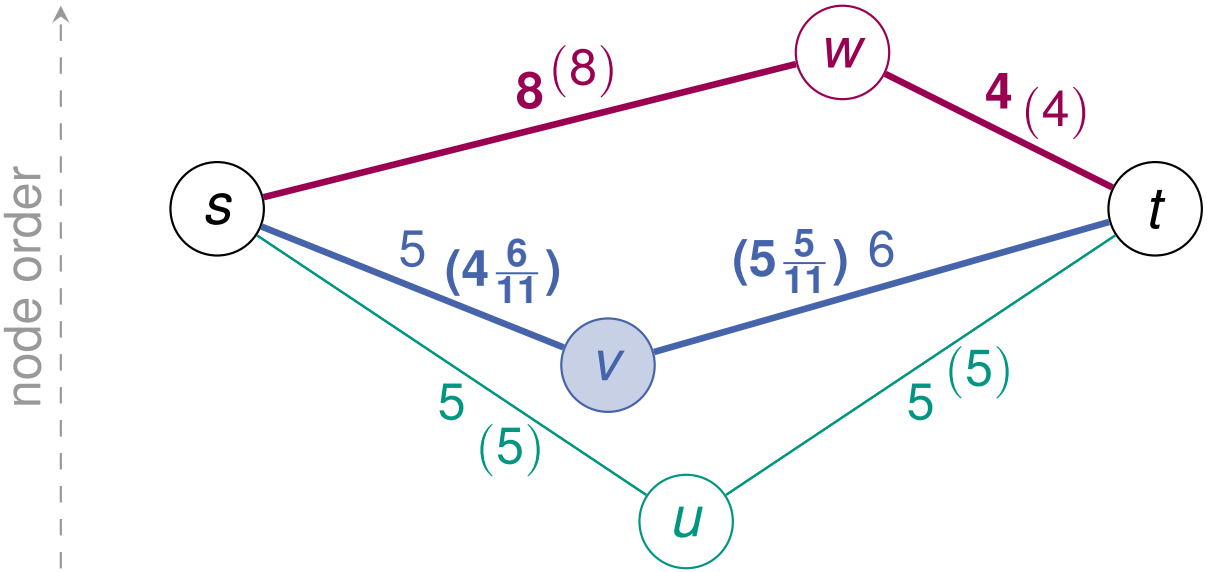
- Store a second edge weight $\tilde{c}$ with each edge (initially $\tilde{c} := c$).
- Add shortcut iff $c(\text{witness}) > (1+\varepsilon)\tilde{c}(\text{shortcut})$.
- Update $\tilde{c}(x,y) := \min\left\{\tilde{c}(x,y), \tilde{c}(\text{shortcut})\frac{c(x,y)}{c(\text{witness})}\right\}$ for all edges (x,y) on the witness path that prevented the shortcut.
  $\Rightarrow \tilde{c}(\text{witness}) \leq \tilde{c}(\text{shortcut})$



Approximate Contraction Hierarchies

Details

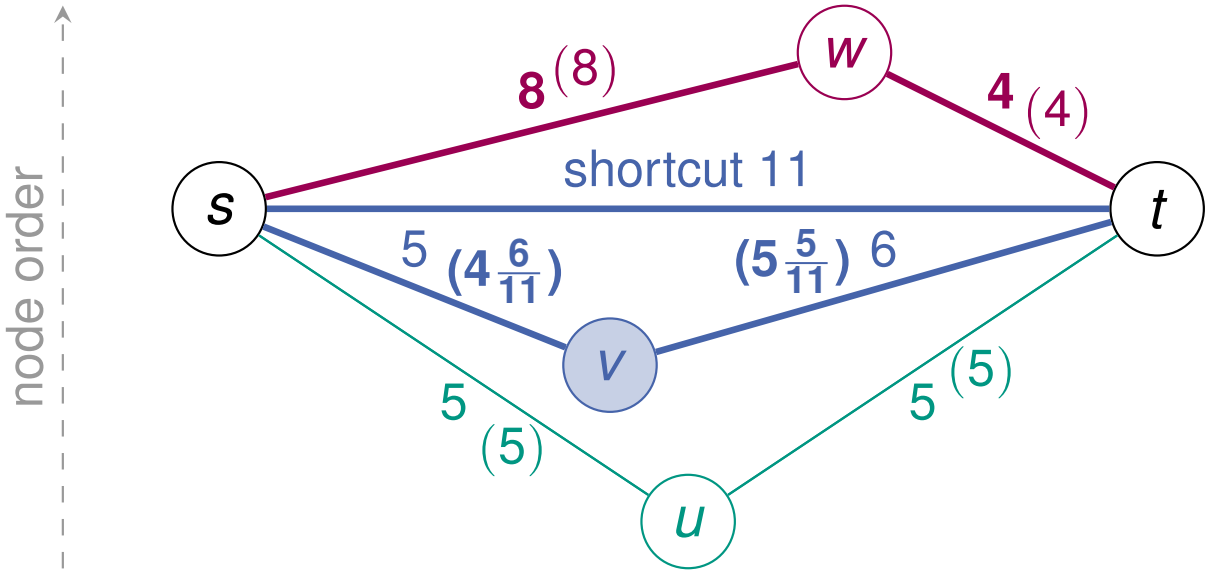
- Store a second edge weight $\tilde{c}$ with each edge (initially $\tilde{c} := c$).
- Add shortcut iff $c(\text{witness}) > (1+\varepsilon)\tilde{c}(\text{shortcut})$.
- Update $\tilde{c}(x,y) := \min\left\{\tilde{c}(x,y), \tilde{c}(\text{shortcut})\frac{c(x,y)}{c(\text{witness})}\right\}$ for all edges (x,y) on the witness path that prevented the shortcut.
  $\Rightarrow \tilde{c}(\text{witness}) \leq \tilde{c}(\text{shortcut})$



Approximate Contraction Hierarchies

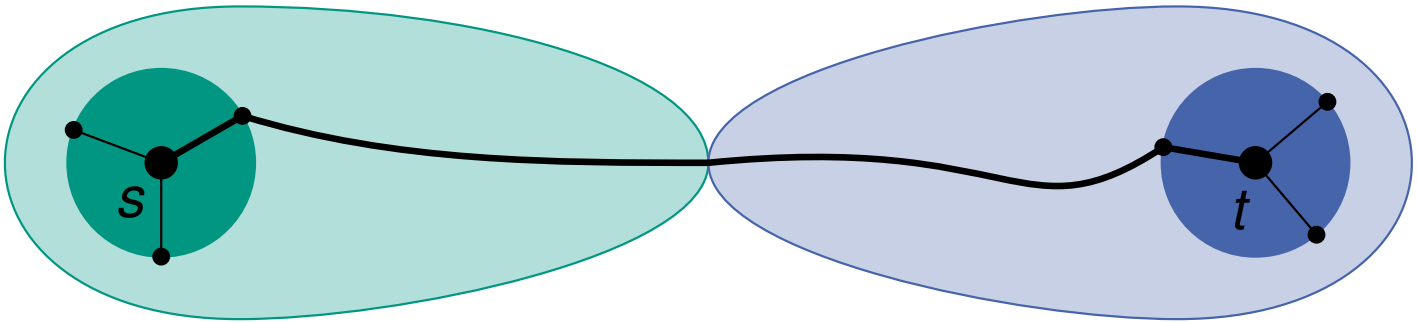
Details

- Store a second edge weight $\tilde{c}$ with each edge (initially $\tilde{c} := c$).
- Add shortcut iff $c(\text{witness}) > (1+\varepsilon)\tilde{c}(\text{shortcut})$.
- Update $\tilde{c}(x,y) := \min\left\{\tilde{c}(x,y), \tilde{c}(\text{shortcut})\frac{c(x,y)}{c(\text{witness})}\right\}$ for all edges (x,y) on the witness path that prevented the shortcut.
  $\Rightarrow \tilde{c}(\text{witness}) \leq \tilde{c}(\text{shortcut})$



Combination with Goal-directed Techniques

- Techniques: A*, ALT, Arc flags.
- Goal-direction only on the core of 5% highest ordered nodes.
- Previous algorithms [Bauer et al. 2008] work almost directly with approximate node contraction.
- A* and ALT can be weighted with $(1 + \varepsilon)$ [Pohl 1970].



Experimental Results

sensor network, 1 000 000 nodes, average degree 10

	preproc.		query		
	[s]	[B/n]	#settled	[ms]	error
bidir. Dijkstra	0	0	326 597	127.1	-
bidir. ALT-a64	194	512	3 173	2.8	-
unidir. A*	0	16	57 385	36.6	-
unidir. WA*-10%	0	16	1 234	1.0	1.25%
unidir. WA*-21%	0	16	724	0.7	2.87%
CH	1 887	0	2 969	4.0	-
apxCH-1%	993	-4	2 742	2.7	0.16%
apxCH-10%	474	-18	2 584	1.9	2.17%
apxCHALT-10%	489	7	215	0.3	2.16%
apxCHALT-10% W-10%	489	7	102	0.2	3.56%

Experimental Results

sensor network, 1 000 000 nodes, average degree 10

	preproc.		query		
	[s]	[B/n]	#settled	[ms]	error
bidir. Dijkstra	0	0	326 597	127.1	-
bidir. ALT-a64	194	512	3 173	2.8	-
unidir. A*	0	16	57 385	36.6	-
unidir. WA*-10%	0	16	1 234	1.0	1.25%
unidir. WA*-21%	0	16	724	0.7	2.87%
CH	1 887	0	2 969	4.0	-
apxCH-1%	993	-4	2 742	2.7	0.16%
apxCH-10%	474	-18	2 584	1.9	2.17%
apxCHALT-10%	489	7	215	0.3	2.16%
apxCHALT-10% W-10%	489	7	102	0.2	3.56%

Experimental Results

sensor network, 1 000 000 nodes, average degree 10

	preproc.		query		
	[s]	[B/n]	#settled	[ms]	error
bidir. Dijkstra	0	0	326 597	127.1	-
bidir. ALT-a64	194	512	3 173	2.8	-
unidir. A*	0	16	57 385	36.6	-
unidir. WA*-10%	0	16	1 234	1.0	1.25%
unidir. WA*-21%	0	16	724	0.7	2.87%
CH	1 887	0	2 969	4.0	-
apxCH-1%	993	-4	2 742	2.7	0.16%
apxCH-10%	474	-18	2 584	1.9	2.17%
apxCHALT-10%	489	7	215	0.3	2.16%
apxCHALT-10% W-10%	489	7	102	0.2	3.56%

Experimental Results

road network of Western Europe, 18 million nodes, 42.2 million edges

		preproc.		query		
		[s]	[B/n]	#settled	[ms]	error
TRAVEL TIME	CH	1 050	-1	430	0.206	-
	apxCH-10%	1 099	-2	430	0.199	0.40%
	CHASE	13 421	7	42	0.028	-
	apxCHASE-10%	11 977	5	42	0.026	0.40%
DISTANCE	CH	1 258	0	1 333	1.198	-
	apxCH-10%	950	0	1 248	0.873	1.32%
	CHASE	84 759	15	59	0.058	-
	apxCHASE-10%	33 147	10	62	0.048	1.32%

Experimental Results

road network of Western Europe, 18 million nodes, 42.2 million edges

		preproc.		query		
		[s]	[B/n]	#settled	[ms]	error
TRAVEL TIME	CH	1 050	-1	430	0.206	-
	apxCH-10%	1 099	-2	430	0.199	0.40%
	CHASE	13 421	7	42	0.028	-
	apxCHASE-10%	11 977	5	42	0.026	0.40%
DISTANCE	CH	1 258	0	1 333	1.198	-
	apxCH-10%	950	0	1 248	0.873	1.32%
	CHASE	84 759	15	59	0.058	-
	apxCHASE-10%	33 147	10	62	0.048	1.32%

Conclusion

- Approximation extends the application range of contraction hierarchies beyond road networks.
- Further speed-up achieved with goal-direction.

Karlsruhe Institute of Technology

Thanks for your attention.

Any question?