Karlsruhe Institute of Technology

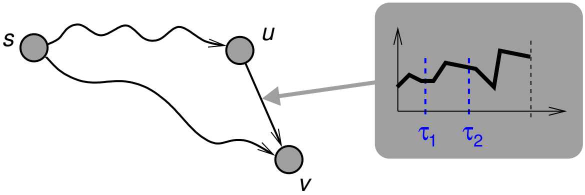
Contraction of Timetable Networks with Realistic Transfers

Robert Geisberger – *geisberger@kit.edu*

Institute for Theoretical Computer Science, Algorithmics II

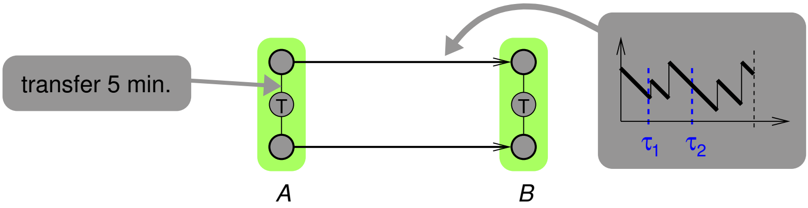
Motivation

- Route planning in time-dependent road networks is fast.
- Speed-up techniques gain three orders of magnitude over time-dependent Dijkstra (earliest arrival queries).
  - Contraction Hierarchies [ALENEX'09, SEA'10]
  - SHARC [ESA'08, SEA'10]
- Road network is modelled as graph with time-dependent edge weights that map arrival time → travel time.



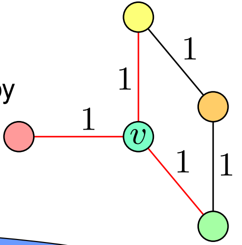
Motivation

- Route planning in public transportation networks is slow.
- Speed-up techniques gain one order of magnitude.
  - SHARC [ESA'08, ATMOS'09]
- Network is still modelled as graph with time-dependent edge weights that map arrival time → travel time.
- Parallel edges necessary to model realistic transfers (minimum transfer buffer for each station).



Motivation

Node contraction, a very successful technique for route network performs worse when parallel edges are involved:



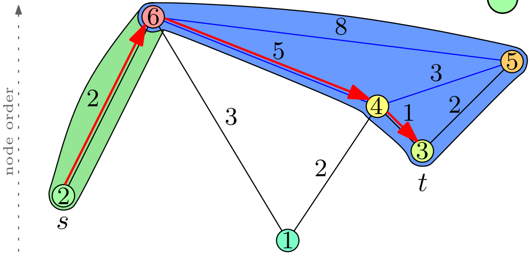
- Order nodes by ‘importance’, $V = \{1, 2, \ldots, n\}$.
- Contract nodes in this order, node v is contracted by

  foreach *pair* (u, v) *and* (v, w) *of edges* **do**

  if $\langle u, v, w \rangle$ *is a unique shortest path* **then**

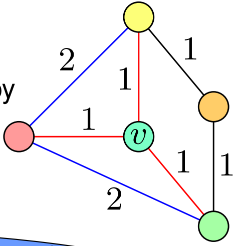
  add shortcut (u, w) with weight $w(\langle u, v, w \rangle)$

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



Motivation

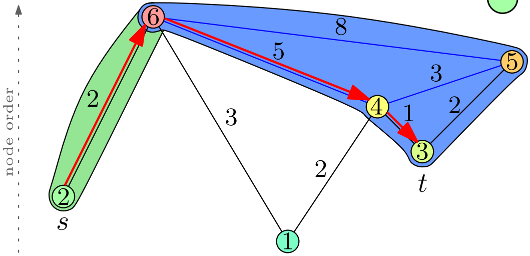
Node contraction, a very successful technique for route network performs worse when parallel edges are involved:



- 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 $w(\langle u, v, w \rangle)$

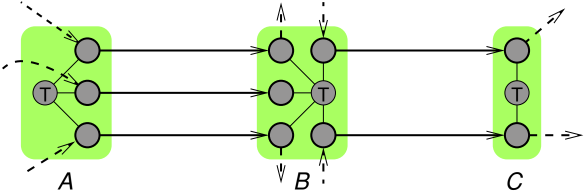
- Query relaxes only edges to more "important" nodes
  $\Rightarrow$ valid due to shortcuts.



Motivation

Node contraction, a very successful technique for route network performs worse when parallel edges are involved:

- Shortcuts can multiply: a incoming parallel edges and b outgoing parallel edges may result in $a \cdot b$ parallel shortcuts.

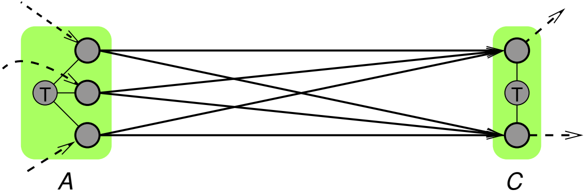


Faculty for Computer Science
Institute for Theoretical Computer Science, Algorithmics II

Motivation

Node contraction, a very successful technique for route network performs worse when parallel edges are involved:

- Shortcuts can multiply: a incoming parallel edges and b outgoing parallel edges may result in $a \cdot b$ parallel shortcuts.

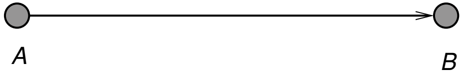


Station graph model

First contribution

- 1:1 mapping between nodes and stations.
- No parallel edges.
- Each edge stores a set of connections, no FIFO-property required.
- Store additional train information to respect transfer buffers.

departure train	departure time	arrival time	arrival train
1	09:30	10:15	1
2	09:45	10:15	2
2	10:45	11:10	3



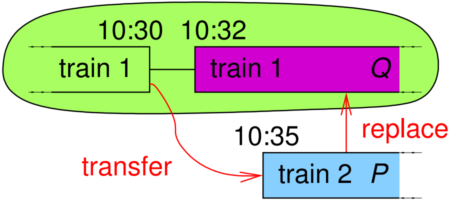
Another station model was independently developed by Berger et al. [ATMOS'09], but requires parallel edges and the FIFO-property.

Station graph model

Dominant connections

We say that a connection P dominates a connection Q if we can replace Q by P, i.e.

- P does not depart before Q and does not arrive after Q.
- When their departure trains differ, there has to be enough time to transfer from the train of Q to the train of P.
- The same has to hold for arrival trains.



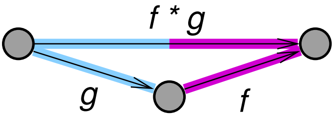
Station graph model

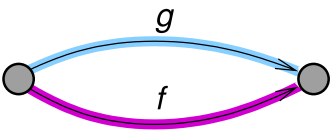
Operations

- Store only dominant set of connections with each edge.
- A search computes dominant sets of connections.

Required operations:

- Link the connections of two incident edges.
- Build the minimum of two sets of connections between the same station pair.



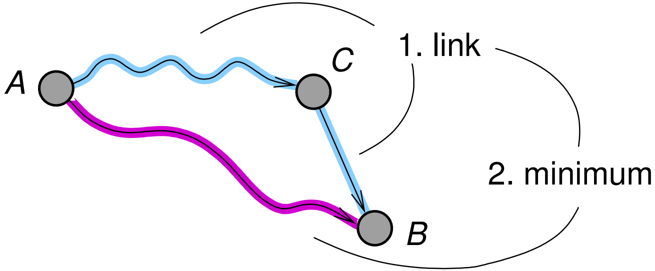


Both operations are ‘almost’ linear in the number of connections.

Station graph model

Profile query

- Compute a dominant set of all connections between a pair of stations (A, B).
- Dijkstra-like label correcting algorithm based on new link and minimum operation. Priority queue key: minimum duration.

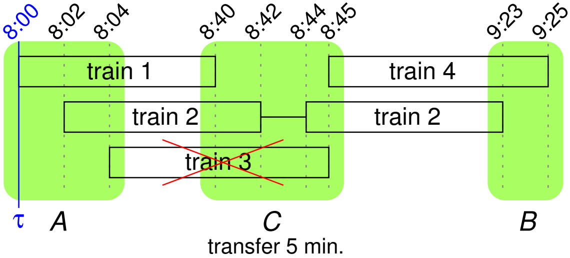


- Combine link and minimum operation to avoid some copying.

Station graph model

Time query

- Compute the fastest connection between a pair of stations (A, B) not departing earlier than time τ.
- Problem: Subpath-optimality not given when we only look at time.
- We need to compute for each train the earliest arrival time, and only drop connections that arrive 'transfer buffer' or more minutes later.



Timetable Contraction Hierarchies

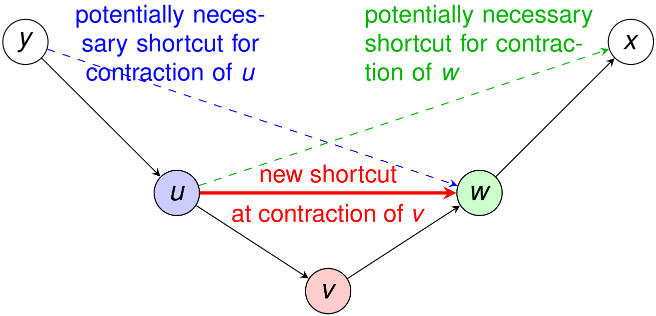
Second contribution

- Most difficult part is preprocessing. Highlevel:
  1. Assign each node a priority on how attractive it is to contract it.
  2. Contract the most attractive node.
  3. Update the priorities of the neighbors of the contracted node.
  4. Repeat from Step 2 until all nodes are contracted.
- Step 3 is the most time-consuming, as it performs a simulated contraction for each neighbor to compute the number of necessary shortcuts (used for node priority).
- Problem: For road networks, min-max-search helps to speedup contraction. But maximum on timetable networks is mostly too high, e.g. when there is no service during the night.

Timetable Contraction Hierarchies

Preprocessing

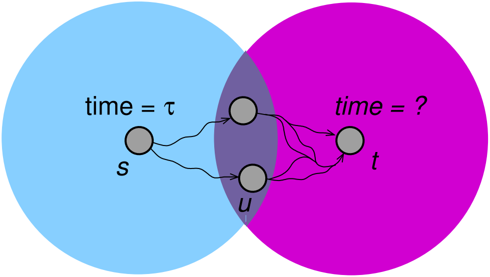
- Store for each remaining node the set of necessary shortcuts.
  - Feasible, as there are much less nodes as in road networks.
- After contraction of a node, we need to update theses sets.
  - Only the endpoints of added shortcuts are affected ($\subseteq$ neighbors).
  - At most one forward profile search and one backward profile search from each neighbor are necessary.



Timetable Contraction Hierarchies

Query

- Profile query is performed bidirectional from source and target.
- Time query does not know the arrival time. Two-phase approach:
  1. Backward BFS from the target using downward edges.
  2. Forward query using upward edges, and then used downward edges.



- Additional optimizations important for road networks (min-max-search, stall-on-demand) bring no advantage.

Experiments

Setup

Environment: Intel Xeon X5550 at 2.67 GHz

Networks:

- long distance connections of Europe
- local traffic in Berlin/Brandenburg in Germany

	time-dependent		station based		factor	
network	nodes	edges	nodes	edges	nodes	edges
long	550 975	1 488 978	30 517	88 091	18.1	16.9
local	228 874	599 406	12 069	33 473	16.9	17.9

Experiments

Station graph model

	query type	model	#delete mins	speed up	time [ms]	speed up
long	time	time-dep.	259 506	-	54.3	-
		station	14 504	17.9	9.4	5.8
	profile	time-dep.	1 949 940	-	1 994	-
		station	48 216	40.4	242	8.2
local	time	time-dep.	112 683	-	20.9	-
		station	5 969	18.9	4.0	5.2
	profile	time-dep.	1 167 630	-	1 263	-
		station	33 592	34.8	215	5.9

Experiments

Timetable Contraction Hierarchies

	PREPROC. time [s]	PREPROC. edge inc.	type	QUERY #del. mins	QUERY speed up	QUERY time [ms]	QUERY speed up
long	619	86%	time	183	79	0.2	43.5
			profile	251	192	3.4	71.4
local	685	128%	time	186	32	0.4	9.2
			profile	426	79	24.2	8.9

Experiments

Timetable Contraction Hierarchies

Comparison with time-dependent SHARC [ESA'08] on network long.

	PREPROC.		QUERY				
	time [s]	edge inc.	type	#del. mins	speed up	time [ms]	speed up
eco SHARC	2 268	74%	time	32 575	8	7.4	7.2
			profile	181 782	11	415.0	5.4
gen SHARC	18 522	74%	time	8 771	30	2.0	26.6
			profile	55 306	35	114.7	19.5
CH	619	86%	time	183	79	0.2	43.5
			profile	251	192	3.4	71.4

We scaled timings of SHARC based on plain Dijkstra timings.

Experiments

Timetable Contraction Hierarchies

Comparison with time-dependent SHARC [ESA'08] on network long.

	PREPROC.		QUERY				
	time [s]	edge inc.	type	#del. mins	speed up	time [ms]	speed up
eco SHARC	2 268	74%	time	32 575	8	7.4	7.2
			profile	181 782	11	415.0	5.4
gen SHARC	18 522	74%	time	8 771	30	2.0	26.6
			profile	55 306	35	114.7	19.5
CH	619	86%	time	183	1 418	0.2	251.4
			profile	251	7 769	3.4	586.5

We scaled timings of SHARC based on plain Dijkstra timings.

Conclusion

- Station graph model is superior to time-dependent model for the given scenario.
- Node contraction works, as hierarchy is better visible due to 1:1 mapping between nodes and stations.
- Timetable Contraction Hierarchies have preprocessing time of a few minutes with query times of half a millisecond.

Open work:

- Support for multi-criteria scenarios, that e.g. respect number of transfers.
- Combination with goal-directed techniques.

Thank You

Thank you for your attention.

Questions

Questions?