Faster shortest path algorithms, part 2

Jesper Nederlof

12 October 2007

Outline

The problem

- A very large road network is given.
- Weight on edges could be distance, but also could be traveling time for example.
- We have to answer single pair shortest path queries very fast.
- Preprocessing is allowed, only linear storage available.
- We assume shortest paths are unique.

Usual approach

- Most commercial systems use some bidirectional search.
- Search from the source and target within a certain radius (for example 30 km), considering all roads
- Continue the search within a larger radius (for example 100 km), considering only national roads and highways.
- Continue the search, only considering highways.
- This is fast but not exact.
- The method which we will discuss does look like this, but uses a careful definition of different levels of roads.
- Because of this, this method will give an optimal route.

The idea

- We want to use some kind of local search, in which we only look to the H closest nodes (not equal to the source), where H is an tuning parameter.
- After this local search we want to switch over into a *highway network* that is much smaller.

Some notation...

- The distance between nodes x and v, $d(x,v)$, is the length of the shortest x-v-path.
- The neighborhood of a node x (or the set of nearest neighbors), $\mathcal{N}_H(x)$, is *the* set of H+1 nodes v with the smallest $d(x,v)$ (we assume there's only one).
- So after $H+1$ removals of nodes from Q in Dijkstra's algorithm from a node s, all nodes $\mathcal{N}_H(x)$ are outside Q (because all nodes v are removed from Q in order of ascending $d(s,v)$).
- Because H will be fixed during runtime, most of the times we will forget it and simply use $\mathcal{N}(\cdot)$

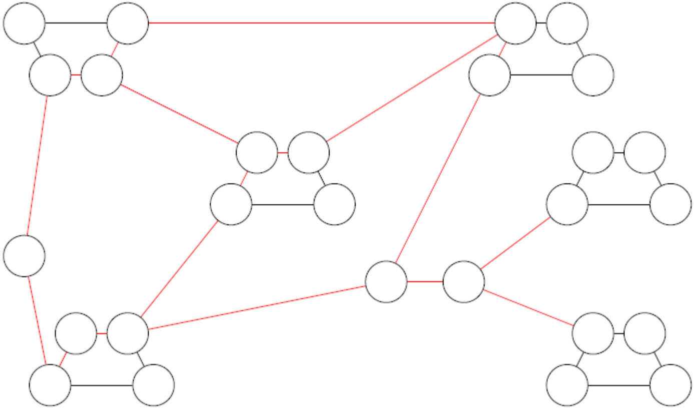
Formalization

Highway network

For a given parameter H (the neighborhood size), $G_1 = (V_1, E_1)$ is the highway network of a graph $G = (V, E)$, with:

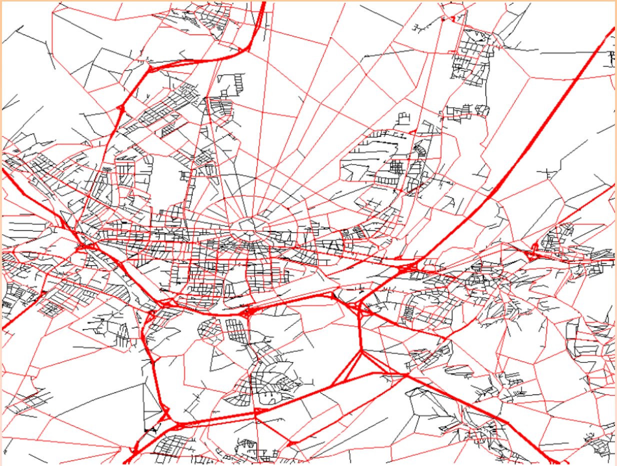
- E_1 is the set of edges $(u, v) \in E$ that appear in a shortest path $\langle s, \ldots, u, v, \ldots, t \rangle$ such that $v \notin \mathcal{N}_H(s)$ and u $\notin \mathcal{N}_H(t)$.
- V_1 is the maximal subset of V such that G_1 contains no isolated vertices.

An example



A simple example of a highway network. The highway edges are highlighted. The weight of an edge is the length of the line segment that represents the edge in this figure. The neighbourhood size H is 3.

Another example



The highway network of Europe, clipped by a bounding box around Karlsruhe. The highway edges are highlighted.

The creation of a highway network

For each $s_0 \in V$, do the following:

- Phase 1: Construct a partial shortest path tree with s_0 as source.
    - This tree will be such that all leaves are “sufficiently” far away.
- Phase 2: Select the highway edges from this tree and add them to the resulting edge set.

Phase 1: Partial shortest path tree

s_1 is the second node one the shortest path from s_0 to u.

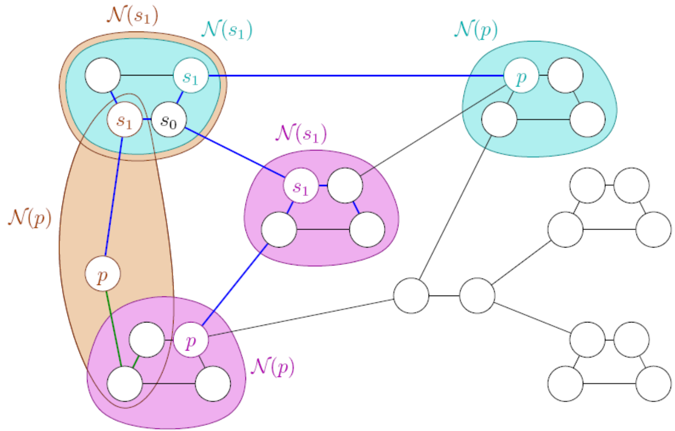
$\textsc{Dijkstra}((V, E), w, s)$

for each $v \in V$ **do** $d(v) \leftarrow \infty$
make s active
$d(s) \leftarrow 0$
initialize priority queue Q with $v \in V$ and priorities $d(v)$
while there are any active nodes left in priority queue Q
    $u \leftarrow$ remove node with smallest $d(u)$ in Q
    if $|\mathcal{N}(s_1) \cap \mathcal{N}(u)| \leq 1$ **then** make u passive
    for each $(u, v) \in E$
        $\textsc{Relax2}(u, v, w)$

Relaxation step

RELAX2(u, v, w)

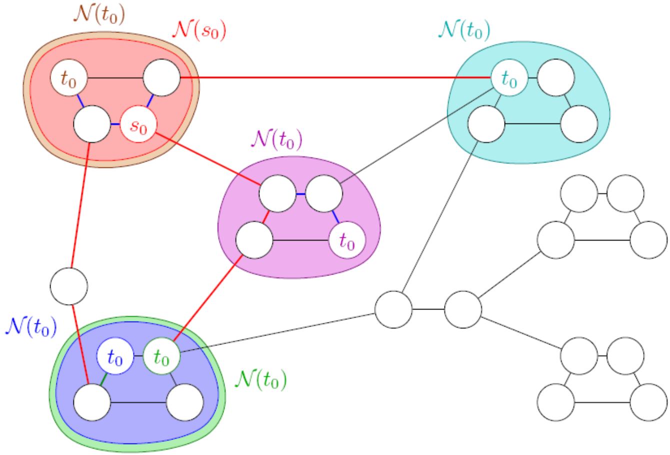
```
if d(v) > d(u) + w(u, v)
then d(v) ← d(u) + w(u, v)
     p(v) ← u
     if u is passive
        make v passive
     else
        make v active
```



H is 3. An SSSP search is performed from s_0. The abort criterion applies three times: the involved nodes s_1 and p and the corresponding neighbourhoods are marked in cyan, magenta, and brown, respectively. In the brown case, the intersection of the concerned neighbourhoods contains exactly one element; in the other two cases, the intersections are empty. All edges that belong to s_0's partial shortest path tree are coloured: edges that leave active nodes are blue, edges that leave passive nodes are green.

Second phase: Select highway edges

- Add all edges (u, v) to the result that lie on a path $\langle s_0, \ldots, u, v, \ldots, t_0 \rangle$ in the partial shortest tree B where t_0 is a leaf in B, $v \notin \mathcal{N}(s_0)$ and $u \notin \mathcal{N}(t_0)$.
- This can be done in $\mathcal{O}(|B|)$ time.



An example of Phase 2 of the construction. s_0's partial shortest path tree has five leaves t_0, which are marked in different colours. The edges that are added to E_1 are highlighted.

Correctness

Construction methods constructs highway graph

An edge $(u, v) \in E$ is added to E_1 by the construction algorithm $\Leftrightarrow$ it belongs to a shortest path $\langle s, \ldots, u, v, \ldots, t \rangle$ and $v \notin \mathcal{N}(s)$ and $u \notin \mathcal{N}(t)$

- $\Rightarrow$: Each edge from the root (s_0) to a leaf (t_0) is a shortest path. Rest follows from specification of phase 2.
- $\Leftarrow$: The edge (u, v) will be added to the result when Phase 1 and 2 are executed from s_0 where $v \notin \mathcal{N}(s_0)$ and $d(s_0, v)$ is minimal.
  - B is the partial tree created in the iteration from s_0.
  - If t is a leaf in B, phase 2 will add (u, v).
  - If t is an internal node in B, the shortest path can be extended to a leaf t_0 with $u \notin \mathcal{N}(t)$.
  - If t is not in B, it gets more complicated. We omit this case.

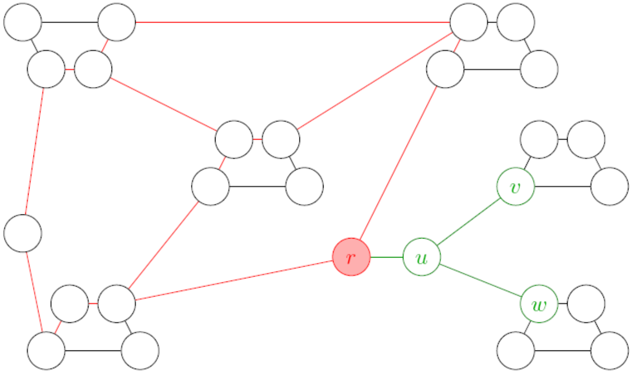
The 2-core of a graph

2-Core

The 2-Core of a graph is the maximal vertex induced subgraph with minimum degree 2.

- Can be found in $\mathcal{O}(m)$
- Each graph consists of its 2-core and attached trees.
- Attached trees are trees whose roots belong to the 2-core, but all other nodes don't.

The 2-core of a graph



The 2-core of the highway network and an attached tree whose root r belongs to the 2-core. Note that u does not belong to the 2-core although it has degree 3 in the highway network.

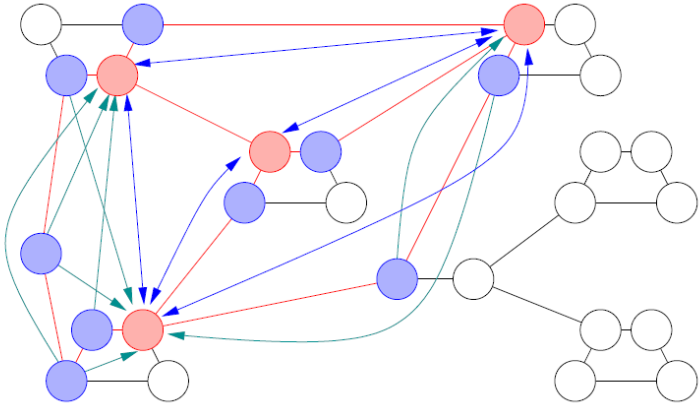
Contracted highway network

Line

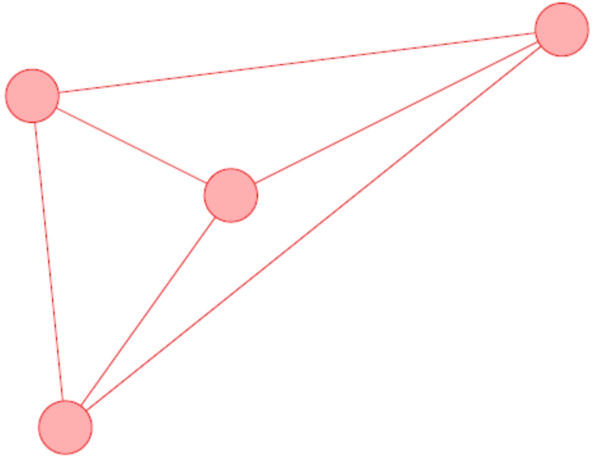
A line is a path $\langle u_0, u_1 \ldots, u_k \rangle$ where the inner nodes $u_1 \ldots, u_{k-1}$ have degree two.

Contracted highway network

From the highway network G_1 of G_0' the contracted highway network G_1' of G_0' is obtained by taking the 2-core of G_1, and, then, replacing all lines $\langle u_0, u_1 \ldots, u_k \rangle$ with an edge (u_0, u_k).



The 2-core of the highway network from containing five lines. Both endpoints of a line are connected by an undirected shortcut. There is a directed shortcut from each inner node of a line to both endpoints.



The *contracted highway network* obtained from the highway network

Highway hierarchy

Highway hierarchy

The highway hierarchy $\mathcal{G}$ consists of $G_0, G_1, \ldots G_L$, which can be constructed by iteratively taking the contract highway graph G'_{i+1} from G'_i.

- Denote with v_i the copy of v in G_i.
- On top of the edges from all graphs $G_0, G_1, \ldots G_L$ directed edges (v_i, v_{i+1}) are added with weight 0.
- For all inner nodes of a line segment, we add shortcuts to its corresponding outer nodes.

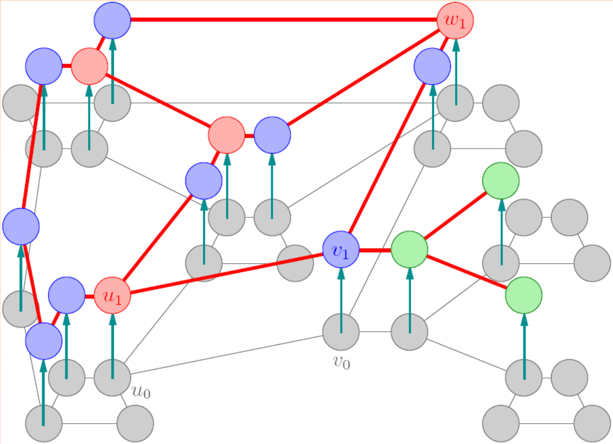


Figure 4.1: A highway hierarchy $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of the graph given in Fig. 2.4 consisting of two levels: $G_0 = (V_0, E_0)$ ('level 0') and $G_1 = (V_1, E_1)$ ('level 1'). As in the previous examples, the neighbourhood size H is 3. Nodes and horizontal edges in level 0 are plotted in grey. There are directed vertical edges from level 0 to level 1. Horizontal edges in level 1 are red. The nodes in level 1 are coloured by type: tree nodes, line nodes, and nodes that belong to the core $G_1' = (V_1', E_1')$. $u_0 \in V_0$ and $u_1 \in V_1$ are copies of the node $u \in V$. $(u_0, v_0) \in E_0$ and $(u_1, v_1) \in E_1$ are copies of the edge $(u, v) \in E$.

Querying Highway hierarchy

- Bidirectional search
- With bidirectional search we only know the shortest path when forward and backward steps meet, if all nodes which are not considered yet have a larger distance than those who are.
- Doesn't work right away, some additional work has to be done.
- The intuition behind the 2-core and line-replacement reductions is:
  - Attached trees and inner nodes of line will only be visited in a shortest path near the end or the beginning.
  - If at the beginning, this part of the path will be found by the forward search in a more detailed graph.
  - If at the end, this part of the path will be found by the backward search in a more detailed graph.

Results

		USA	Europe	Germany
input	#nodes	24 278 285	18 029 721	4 345 567
	#edges	29 106 596	22 217 686	5 446 916
	#degree 2 nodes	7 316 573	2 375 778	604 540
	#road categories	4	13	13
parameters	average speeds [km/h]	40–100	10–130	10–130
	H	225	125	100
construction	CPU time [h]	4.3	2.7	0.5
	#levels	7	11	11
query	CPU time [ms]	7.04	7.38	5.30
	#settled nodes	3 912	4 065	3 286
	main memory usage [MB]	2 443	1 850	466 (346)

Conclusions

- Appropriate definition of a highway resulted in an optimal algorithm.
- Iteratively simplifying the road network resulted in fast queries.
- Bidirectional search is really useful.
- If time left: Demo!
- Questions?

References

- 'Fast and Exact Shortest Path Queries Using Highway Hierarchies', Dominik Schultes, MSc. Thesis , July 2005, http://algo2.iti.uka.de/schultes/hwy/