Route Planning

- Tabulation
- Dijkstra
- Bidirectional
- A*
- Landmarks
- Reach
- ArcFlags
- Transit Nodes
- Contraction Hierarchies
- Hub-based labelling

[ADGW11] Ittai Abraham, Daniel Delling, Andrew V. Goldberg, Renato Fonseca F. Werneck.
A Hub-Based Labeling Algorithm for Shortest Paths in Road Networks.
Proc. 10th International Symposium on Experimental Algorithms (SEA), LNCS 6630, 2011, 230-241.

[BFMSS07] Holger Bast, Stefan Funke, Domagoj Matijevic, Peter Sanders, and Dominik Schultes.
In Transit to Constant Time Shortest-Path Queries in Road Networks.
Proceedings of the Ninth Workshop on Algorithm Engineering and Experiments (ALENEX), 2007.

[GSSD08] Robert Geisberger, Peter Sanders, Dominik Schultes, and Daniel Delling.
Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks.
Proc. 7th International Workshop on Experimental Algorithms (WEA), LNCS 5038, 2008, 319-333.

Route Planning

Input: Directed weighted graph G

Query(s,t) – find shortest route in G from s to t

Lot of algorithm engineering work for road networks

Example: US Tigerline, 58 M edges & 24 M vertices

No preprocessing	**With preprocessing**
Fast query time	Query Time ↔ Space trade-off
Variations of Dijksta’s algorithm	Trivial: Distance table O(1) time & O(n^2) space
	Practice: Try to exploit graph properties

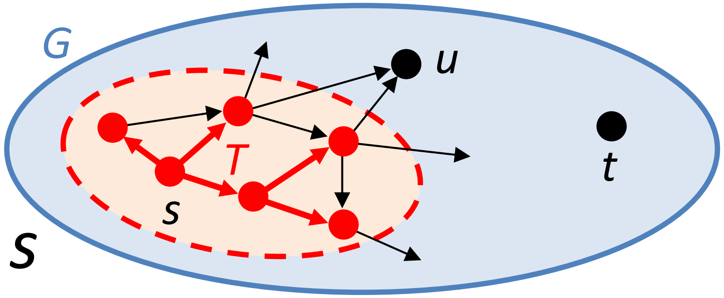
Route Planning – no preprocessing

(non-negative edge weights)

Dijkstra

Build shortest path tree T

Visit vertices in increasing distance to s

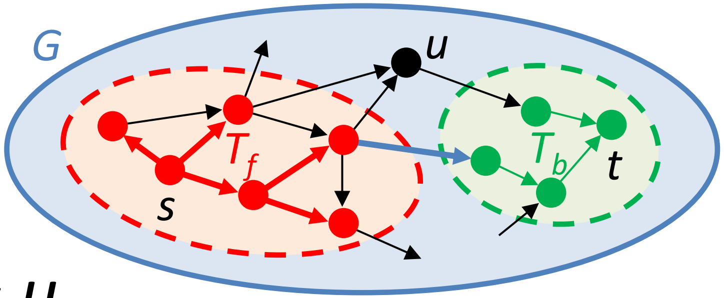


Bidirectional Dijkstra

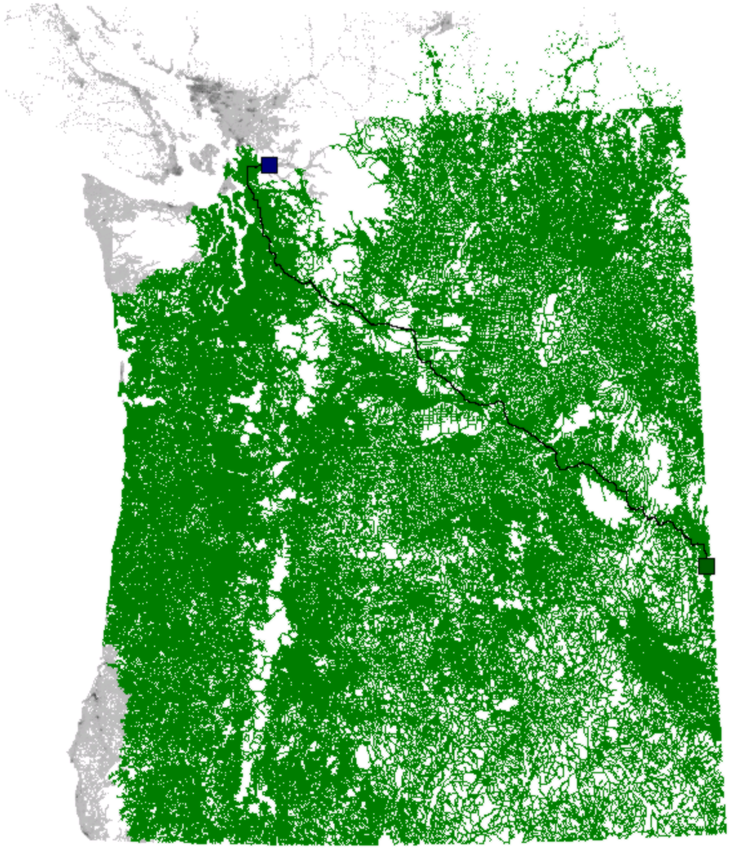
Grow s.p. tree T_f from s and T_b to t

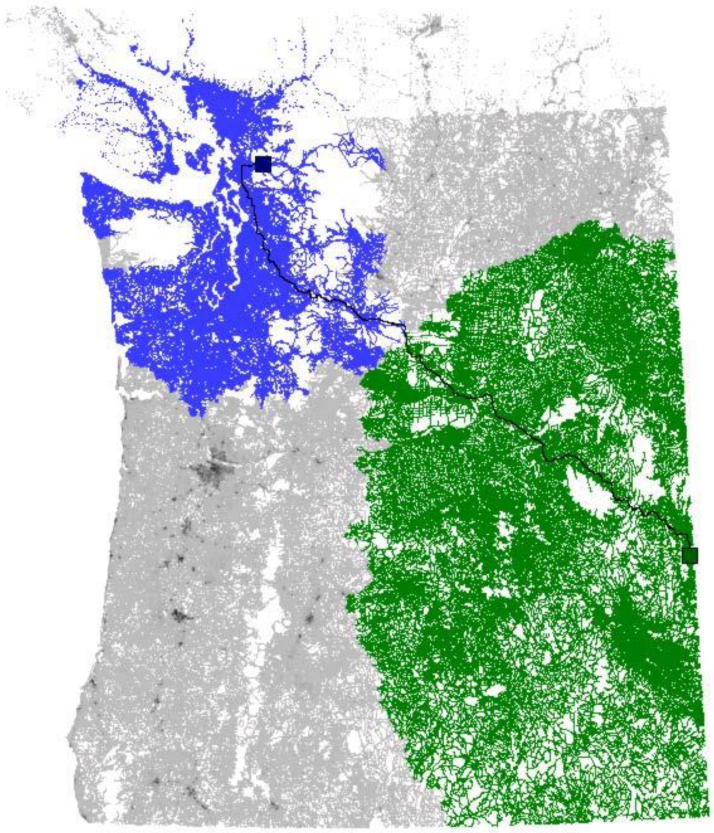
Maintain best so far $s \rightarrow t$ distance μ

Termination condition: $\text{next}_f + \text{next}_b \geq \mu$



Dijkstra vs Bidirectional Dijkstra





http://www.cs.princeton.edu/courses/archive/spr06/cos423/Handouts/EPP%20shortest%20path%20algorithms.pdf

A* ≡ Goal directed

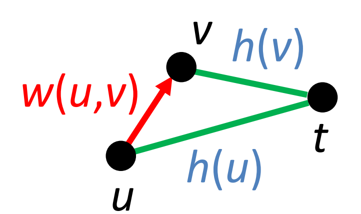
Input: Weighted graph G with non-negative edges
Query(s,t): Shortest route queries

Idea Let $h(v)$ be "heights" & define $w'(u,v) = w(u,v) + h(v) - h(u)$

Fact $w'(s \to v_1 \cdots v_k \to t) = w(s \to v_1 \cdots v_k \to t) + h(t) - h(s)$
$\Rightarrow$ G and G' have <u>identical shortest paths</u>

Fact If $w' \geq 0 \Rightarrow$ we can use Dijkstra's algorithm
If $w' \geq 0$ and $h(t)=0 \Rightarrow h(v)$ **lower bound** on distance $v \to t$

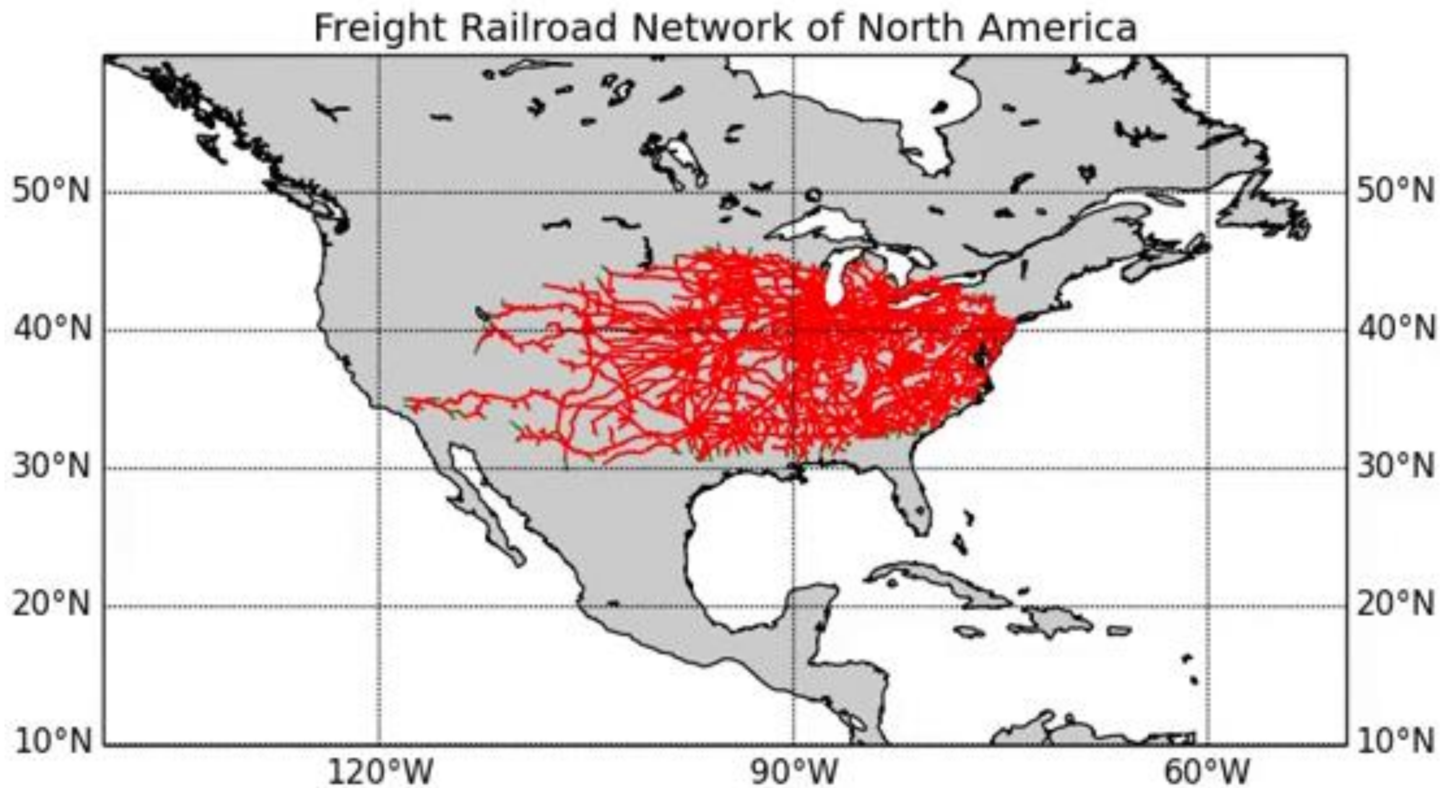
Ex. 1 Planar graphs with L_2 distance, let $h(v) = |t-v|_2$
$\Rightarrow$ triangle inequality ensures w' non-negative



Ex. 2 $h(v) = d_G(v,t) \quad \Rightarrow \quad w'(s,t) = 0$
$\Rightarrow$ Dijkstra's algorithm would only explore the shortest path

Note Bidirectional A* ≡ Bidirectional Dijkstra and A* combined

A*



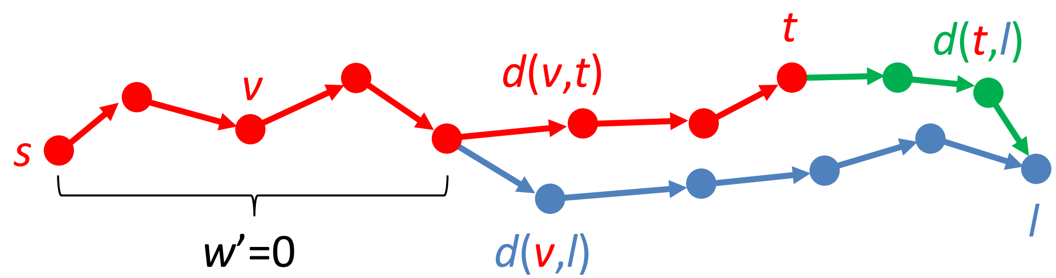
http://en.wikipedia.org/wiki/A*_search_algorithm

Landmarks

Select a *small* number of vertices L (Landmarks)

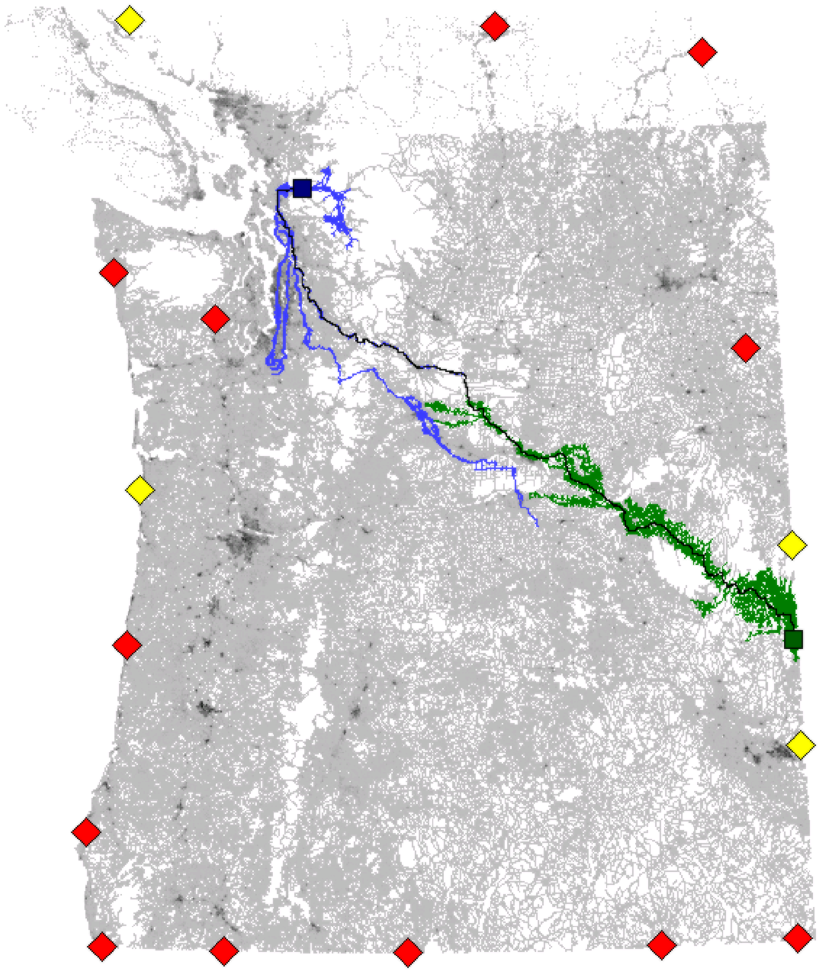
For all nodes v store distance vector $d(v,l)$ to all landmarks $l \in L$

Idea In A* algorithm fix one landmark $l \in L$, and use $h(v) = d(v,l)$ (valid by triangle inequality)



Practice: Use more than one landmark to find lower bounds on $d(v,t)$
Dynamicly increase landmark set during search
Bidirectional A*

Bidirectional A* with Landmarks

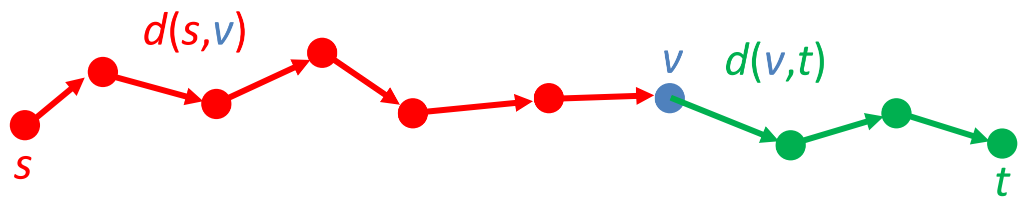


http://www.cs.princeton.edu/courses/archive/spr06/cos423/Handouts/EPP%20shortest%20path%20algorithms.pdf

Reach

For all nodes v store

$$\text{Reach}(v) = \max_{(s,t)\,:\,v \text{ on shortest path } s \to t} \min\{\, d(s,v),\, d(v,t) \,\}$$



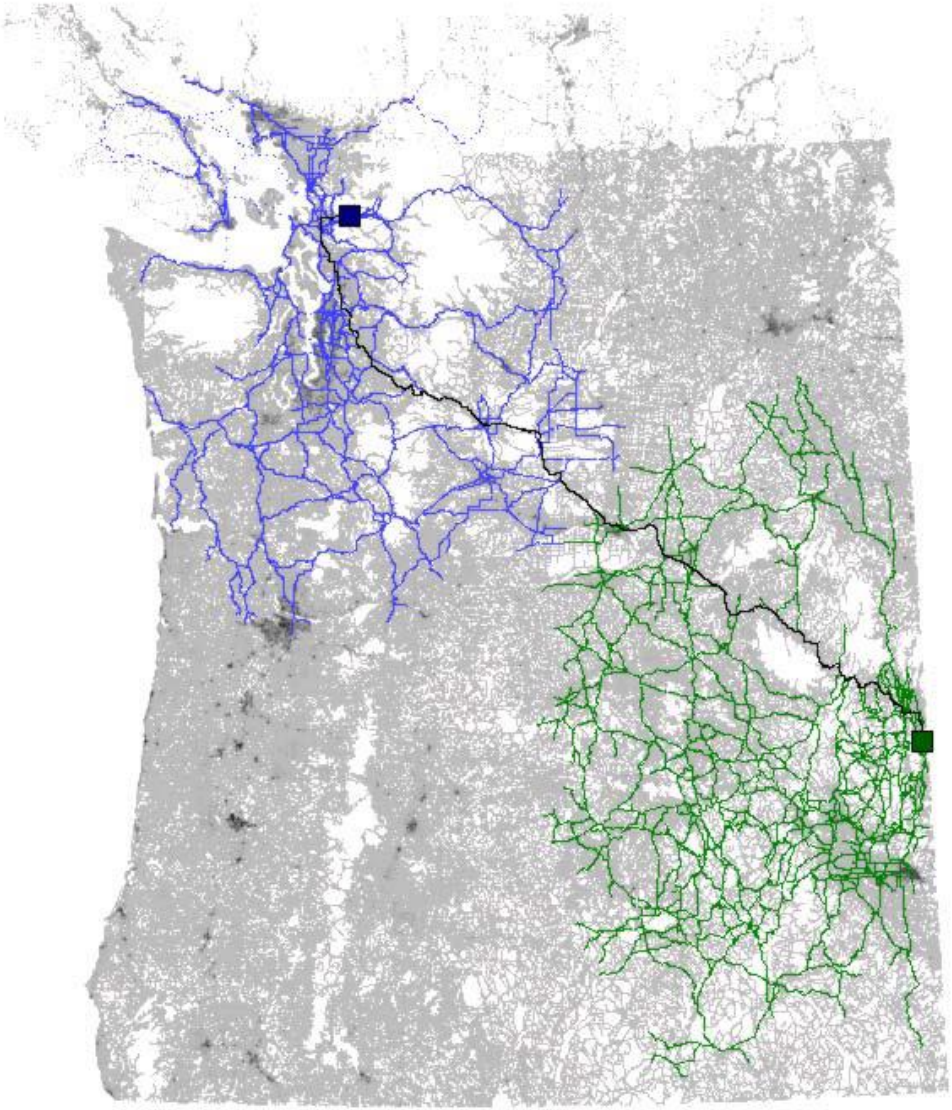
Idea Reach(v) defines ball around v.
If both s and t outside ball, v is not on shortest path

Query Prune edges (u,v) in Dijkstra, when relaxing (u,v) and

$$\text{Reach}(v) < \min\{\, d(s,u)+w(u,v) \,,\, \text{LowerBound}(v,t) \,\}$$

Practice: Approximate Reach for fast preprocessing

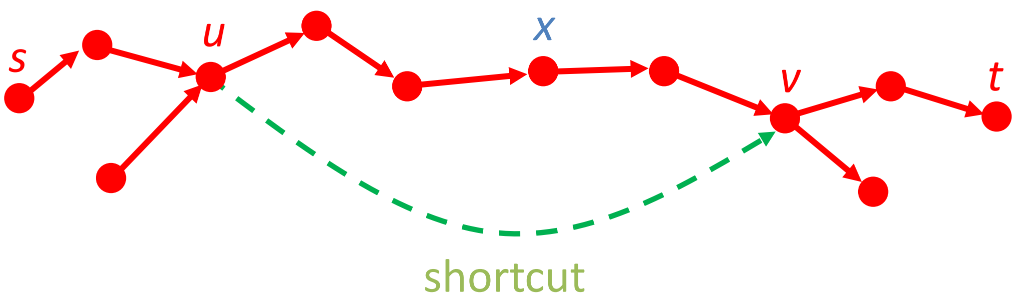
Reach(v)



http://www.cs.princeton.edu/courses/archive/spr06/cos423/Handouts/EPP%20shortest%20path%20algorithms.pdf

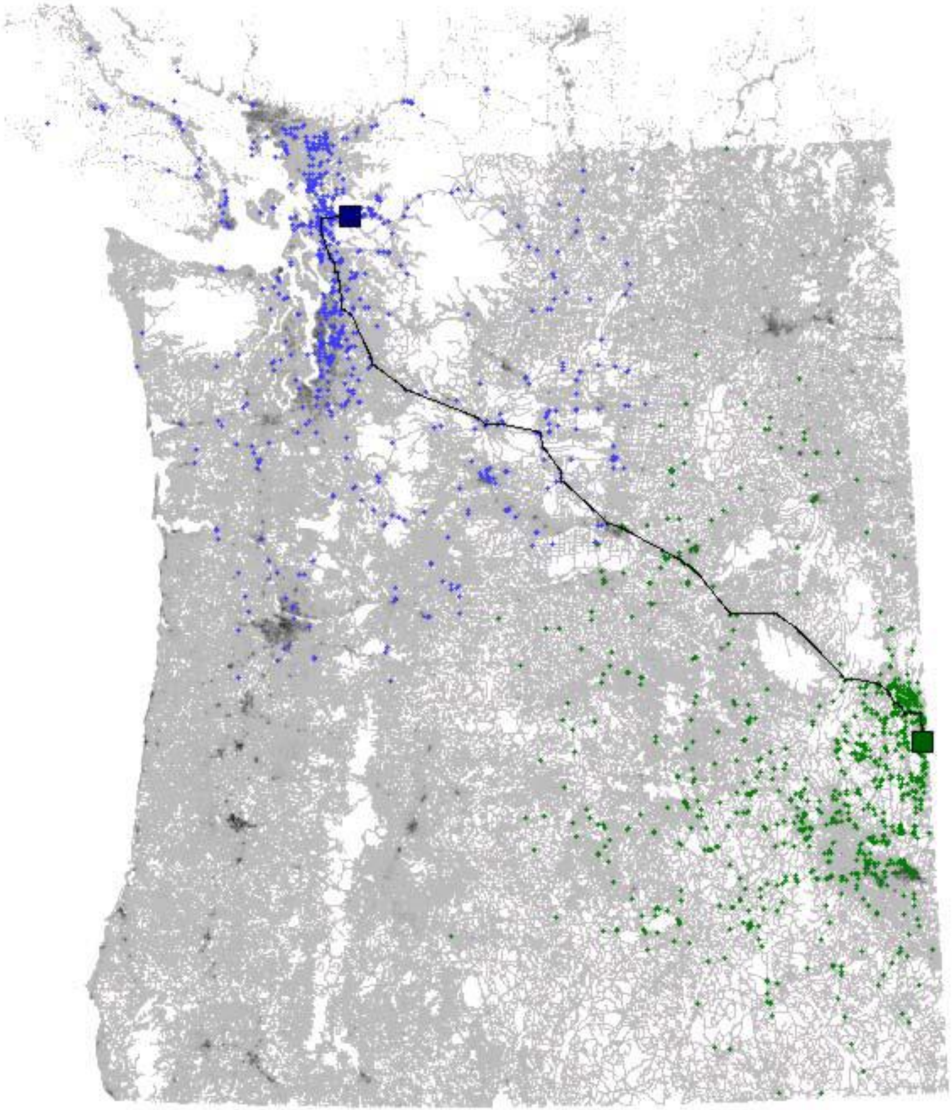
Shortcuts

A directed path $u \rightarrow v$ can be shortcut by a new edge (u,v)



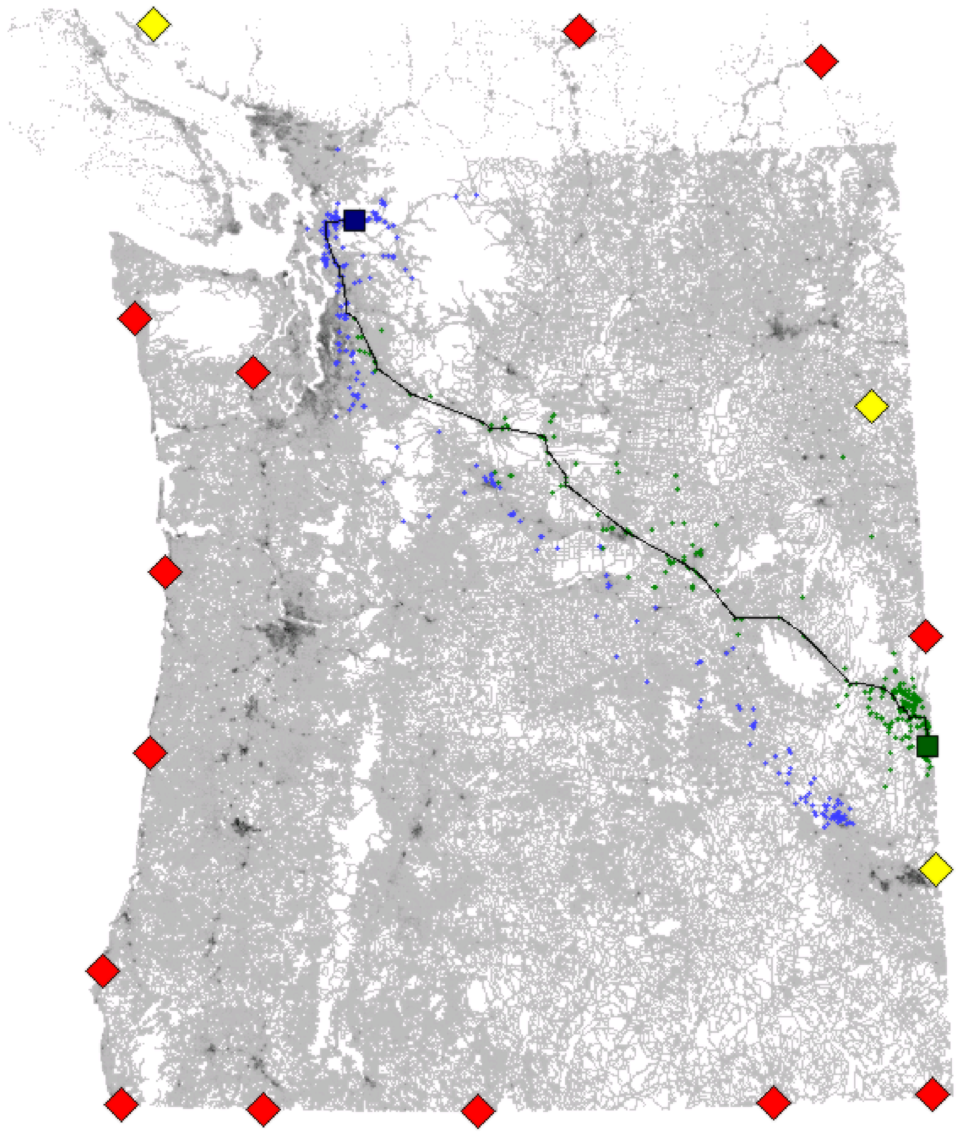
Idea: Shortcuts reduce Reach(x) of vertices x along the shortcut path ($s \rightarrow t$ distances are unchanged)

Reach(v) + Shortcuts



http://www.cs.princeton.edu/courses/archive/spr06/cos423/Handouts/EPP%20shortest%20path%20algorithms.pdf

Reach(v) + Shortcuts + Landmarks



http://www.cs.princeton.edu/courses/archive/spr06/cos423/Handouts/EPP%20shortest%20path%20algorithms.pdf

Experiments – Northwest US

	PREPROCESSING		QUERY		
METHOD	minutes	MB	avgscan	maxscan	ms
Bidirectional Dijkstra	—	28	518 723	1 197 607	340.74
Landmarks	4	132	16 276	150 389	12.05
Reaches	1100	34	53 888	106 288	30.61
Reaches+Shortcuts	17	100	2 804	5 877	2.39
Reaches+Shortcuts+Landmarks	21	204	367	1 513	0.73

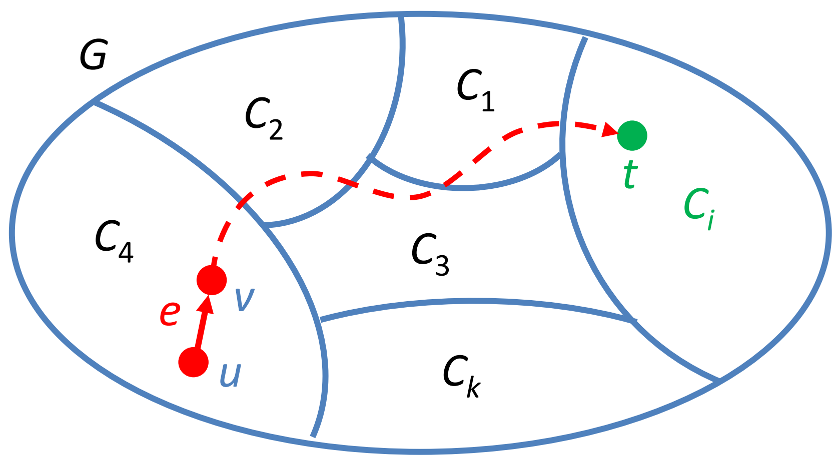
http://www.cs.princeton.edu/courses/archive/spr06/cos423/Handouts/EPP%20shortest%20path%20algorithms.pdf

Arc Flags

Partition vertices into k **components** $C_1,\ldots,C_k$.

For all edges $e = (u,v)$ store a bitvector $\mathrm{Af}_e[1..k]$, where

$\mathrm{Af}_e[i] = \text{true} \iff$ Exist shortest path $u \rightarrow t$ where e is first edge and $t \in C_i$

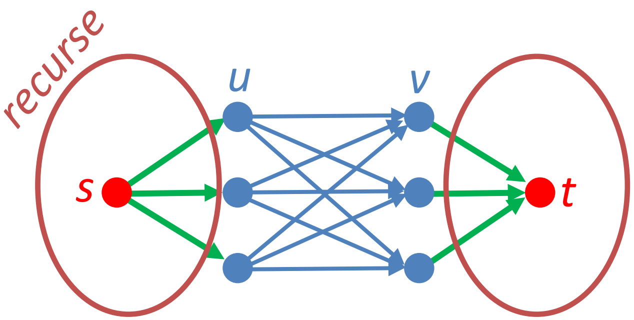


Queries Prune edges e where $\mathrm{Af}_e[i]$ = false and $t \in C_i$

Preprocessing Expensive !

Transit Node Routing

Idea All shortest paths $s \rightarrow t$,
where s and t are far away,
must cross **few** possible **transit nodes**



1. Identify few transit nodes in graph $\sim\sqrt{n}$
2. Compute All-Pair-Shortest-Path matrix for transit nodes
3. For each vertex s find very few transit node distances (US ~10)

Query(s,t) far away queries

For all (u,v), transit nodes u and v for s and t respectively,
find $d(s,t)=d(s,u)+d(u,v)+d(v,t)$ using table lookup

Locality filter = table over when to switch to other algorithm

Practice: Combine recursively with Highway Hierarchies

Transit Node Routing

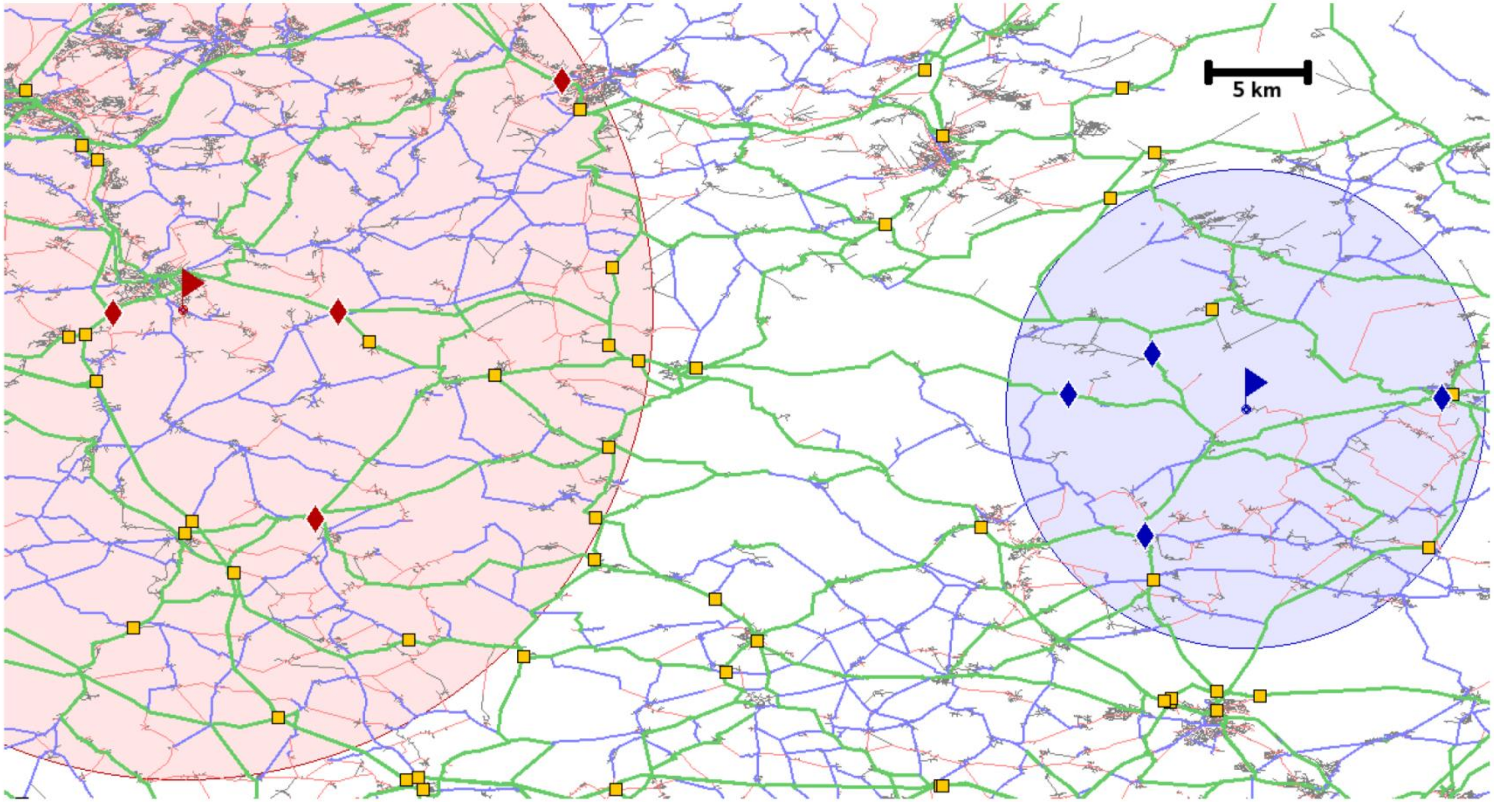


Figure 1: Finding the optimal travel time between two points (flags) somewhere between Saarbrücken and Karlsruhe amounts to retrieving the 2×4 *access nodes* (diamonds), performing 16 table lookups between all pairs of access nodes, and checking that the two disks defining the *locality filter* do not overlap. *Transit nodes* that are not relevant for the depicted query are drawn as small squares.

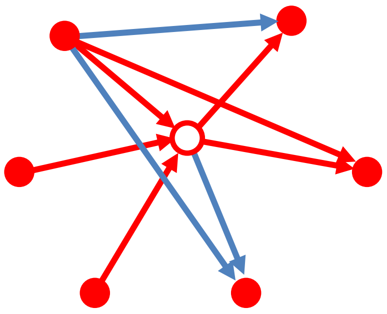
Holger Bast, Stefan Funke, Domagoj Matijevic, Peter Sanders, and Dominik Schultes. In Transit to Constant Time Shortest-Path Queries in Road Networks. Proceedings of the Ninth Workshop on Algorithm Engineering and Experiments (ALENEX), 2007.

Highway Hierachies

- Each nodes findes *H* closest nodes (Neighborhood)
- **Highway edge** (u,v) $\Leftrightarrow$ exist some shortest path $s \rightarrow$t containing (u,v), where $s \notin H$ and t$\notin H$
- **Contract & Recurse** $\Rightarrow$ Hierarchy
- Queries
  - Heuristics similar to Reach
  - Bidrectional Dijkstra (skipping lower level edges)

Contraction Hierarchies

- Order nodes $v_1,\ldots,v_n$ in increasing order of importance
- Repeatedly **contract** unimportant nodes by adding **shortcuts** required by shortest paths

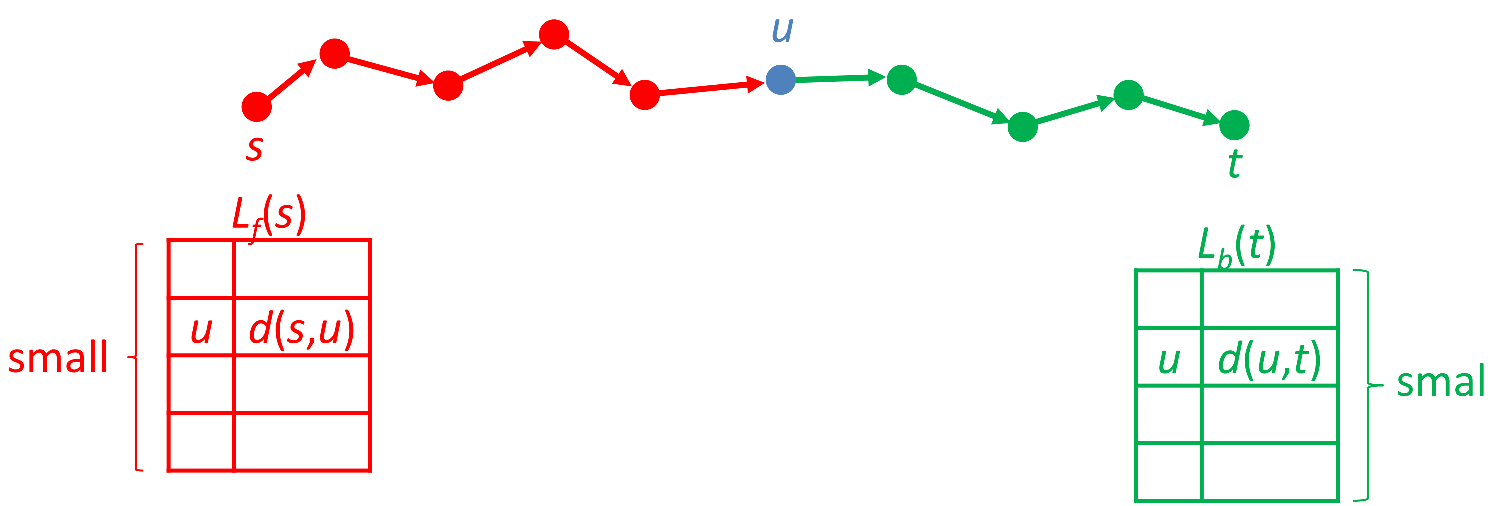


- Many heuristics in construction phase
- **Query**: Bidirectional – only go to more important nodes

Hub Labelling

For all nodes v store two lists $L_f(v)$ and $L_b(v)$, such that for all (s,t) pairs, the shortest path $s \rightarrow t$ contains a node u, where $u \in L_f(s) \cap L_b(t)$

Trivially exist; hard part is to limit space usage



Hub Labelling comparison

		Europe			USA		
	method	preprocessing time [h:m]	space [GB]	query [ns]	preprocessing time [h:m]	space [GB]	query [ns]
Contraction hierarchies	CH [5]	0:13	0.4	93 995	0:14	0.5	67 885
	CHASE [5]	0:52	0.6	9 034	1:59	0.7	9 922
Arc flags	HPML [9]	≈12:00	3.0	9 817	≈12:00	5.1	10 078
Transit node	TNR [5]	0:58	3.7	1 775	0:47	5.4	1 566
	TNR+AF [5]	2:00	5.7	992	1:22	6.3	888
Hub labelling	HL prefix	2:31 + 0:45	5.7	527	2:17 + 0:40	6.4	542
	HL local	2:31 + 0:08	20.1	572	2:17 + 0:07	22.7	627
	HL global	2:31 + 0:14	21.3	276	2:17 + 0:18	25.4	266
	Table Lookup	> 11:03	1 208 358.7	56	> 22:44	2 293 902.1	56

Ittai Abraham, Daniel Delling, Andrew V. Goldberg, Renato Fonseca F. Werneck. A Hub-Based Labeling Algorithm for Shortest Paths in Road Networks. Symposium on Experimental Algorithms (SEA), Lecture Notes in Computer Science, Volume 6630, 2011, pp 230-241.