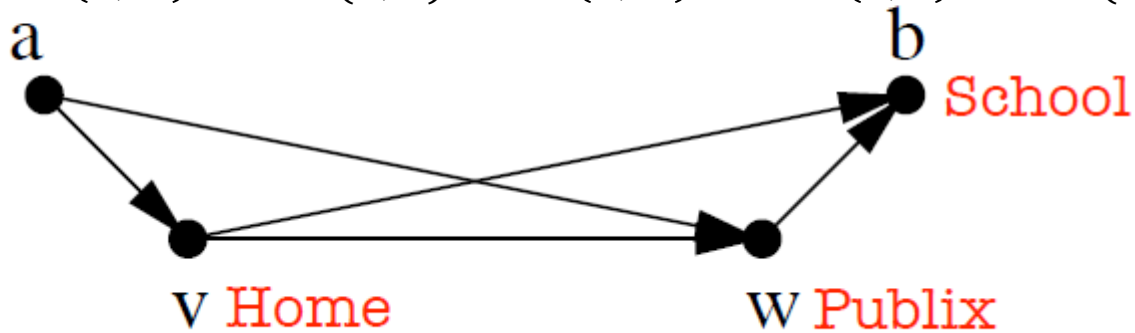


Applications of Search

landmarks

triangle inequality

$$\geq \text{dist}(a, w) - \text{dist}(a, v) \quad \text{dist}(v, w) \geq \text{dist}(v, b) - \text{dist}(w, b)$$

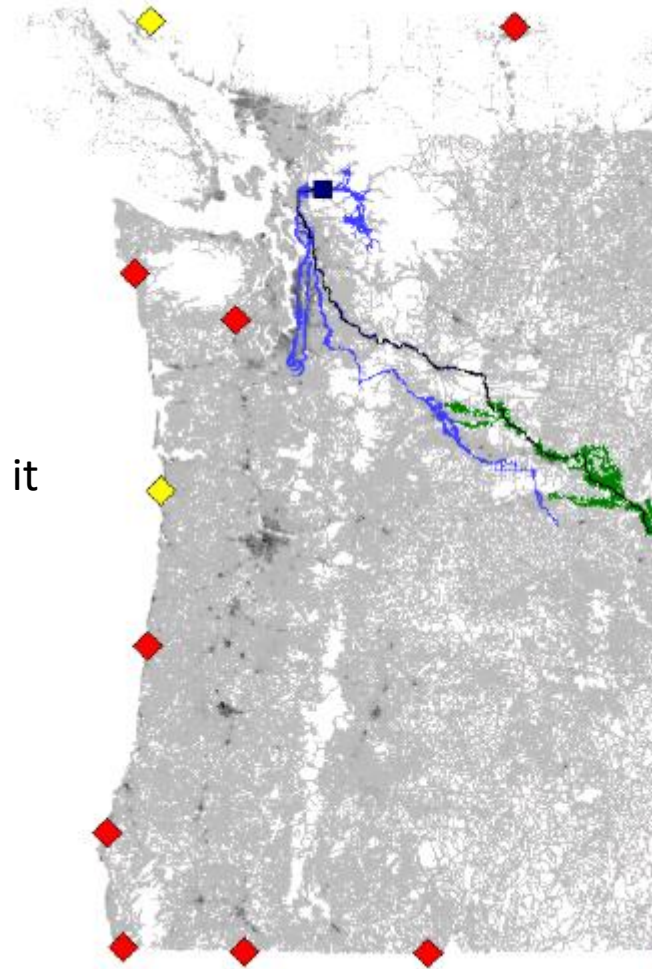


function that serves as a heuristic! This may be Euclidean Distance but it

computation

lower bounds (depending on landmark) than simple Euclidean distance

depending on the landmark, heuristic cost can be 0!!



Images from Goldberg (2011), R

reach

Given a vertex v , calculate all the shortest paths through it.

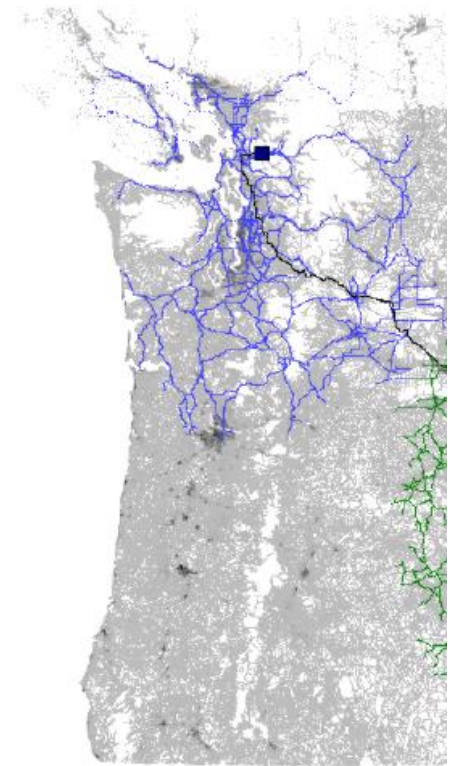
- For a shortest path P , the reach of the vertex:

$$r_P(v) = \min(\text{cost}(s, v), \text{cost}(v, t))$$

- Total reach across all paths: $r(v) = \max_p(r_P)$

Prune v when $r(v) < \min(\text{dist}(s, v), \text{cost}(v, t))$

Forces you to prioritize highways



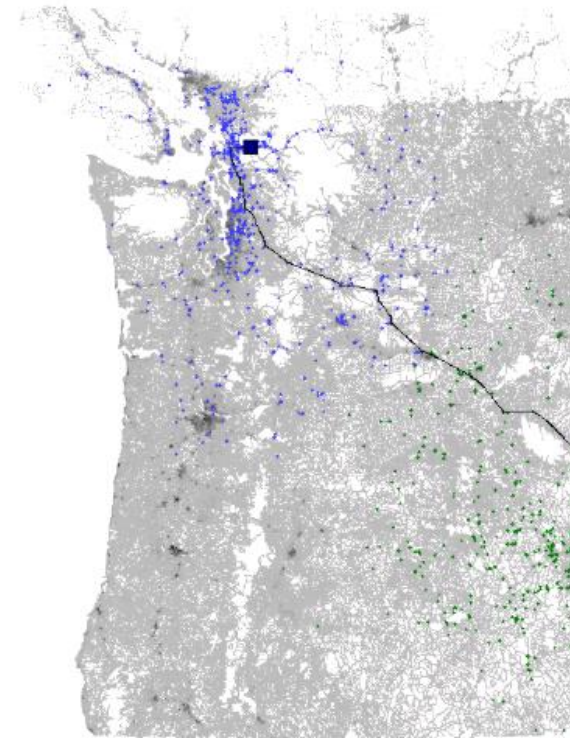
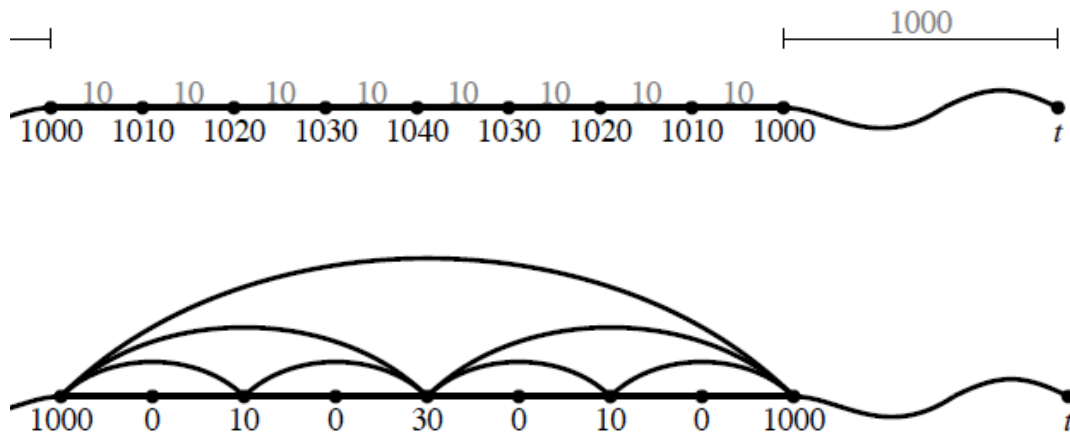
Images from Goldberg (2011), R

shortcuts

add in vertices to reduce the number of shortest paths through a large number of nodes

- Reduces the Reach of intermediate nodes

can prune more paths as a result



Images from Goldberg (2011), R

Impact of Landmark, Reach and Shortcuts

West (1.6M vertices), random queries, 16 landmarks.

Method	preprocessing		query		
	minutes	MB	avgscan	maxscan	ms
Standard Dijkstra	—	28	518 723	1 197 607	340.74

North America (30M vertices), random queries, 16 landmarks.

Method	preprocessing minutes	MB	query		
			avgscan	maxscan	ms
Standard Dijkstra	—	28	518 723	1 197 607	340.74
+Landmark	4	1 100	10 255 356	27 166 866	76
+Short	17	1 100	250 381	3 584 377	3
+Short+ALT	21	1 100	14 684	24 618	3
+Short+Reach	—	—	—	—	—
+Short+Reach+ALT	—	—	—	—	—

Johnson (2011), Reach for A*

ore Resources

test version of slides explaining Landmarks, Reach, and Shortcuts

<http://www.columbia.edu/~cs2035/courses/ieor6614.S16/goldberg.pdf>)

iper Introducing Landmarks

<http://www.cs.princeton.edu/courses/archive/spr06/cos423/Handouts/GH05.pdf>)

iper Introducing Reach (<http://www.siam.org/meetings/alnex04/abstracts/rgutman1.p>

iper Comparing all of these algorithms (<https://www.microsoft.com/en-research/wp-content/uploads/2004/07/tr-2004-24.pdf>)

Google Maps – Transfer Patterns

Add time dependent nodes to your graph and precompute lowest cost routes for disjoint subsets (http://ad-publications.informatik.uni-freiburg.de/ESA_transferpatterns_BCEGHRV_2010.pdf)

Encode time domain data in frequency space and modify Dijkstra to work in that sparse representation (http://ad-publications.informatik.uni-freiburg.de/SIGSPATIAL_frequency_BS_2014.pdf)

Cluster nodes in subgraphs to minimize precomputation costs (http://ad-publications.informatik.uni-freiburg.de/ALENEX_scalable_tp_BHS_2016.pdf)