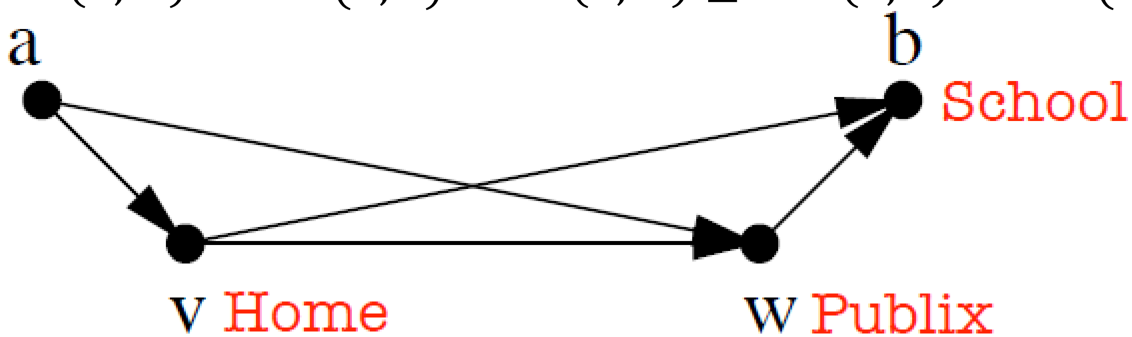
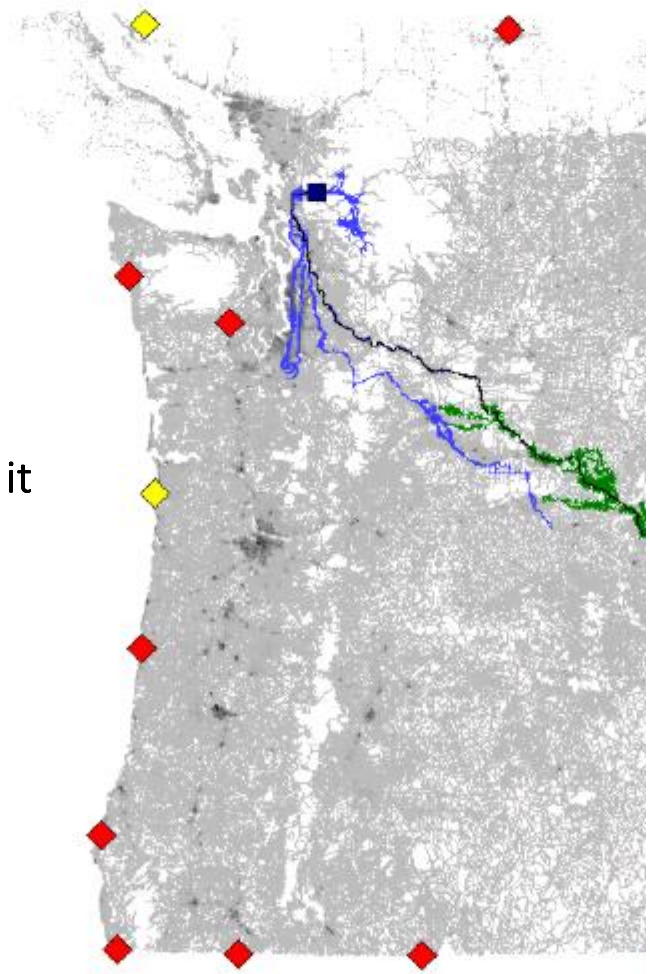
Applications of Search

ndmarks

triangle inequality

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





ction that serves as a <u>heuristic</u>! This may be Euclidean Distance but it

computation

er bounds (depending on landmark) than simple Euclidean

ance

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

Images from Goldberg (2011), R

:ach

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

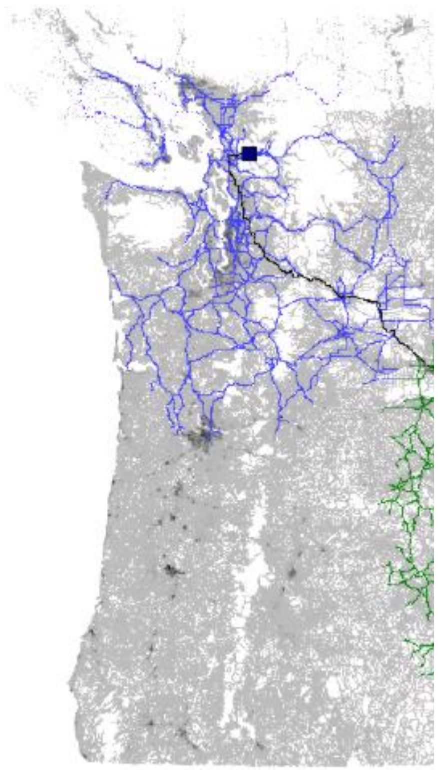
- For a shortest path P, the reach of the vertex:

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

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

:une v when $r(v) < \min(dist(s,v), cost(v,t))$

ɔrces you to prioritize highways

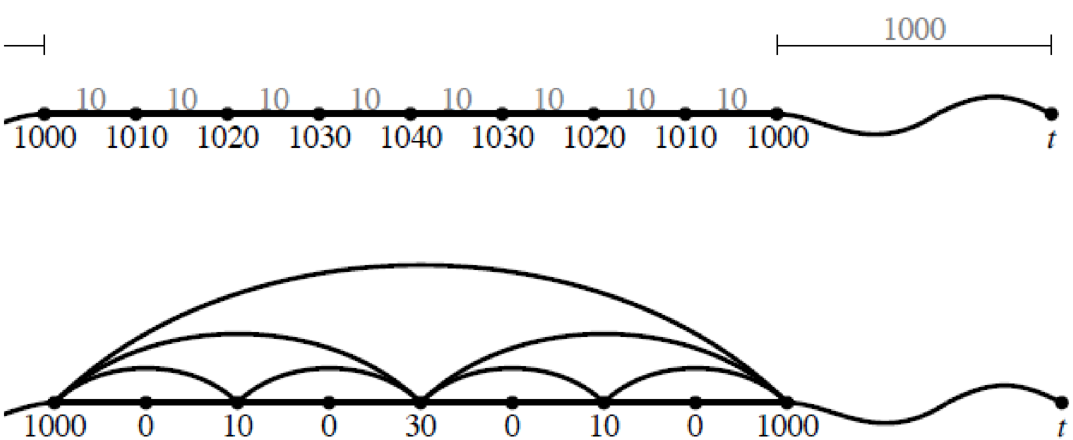


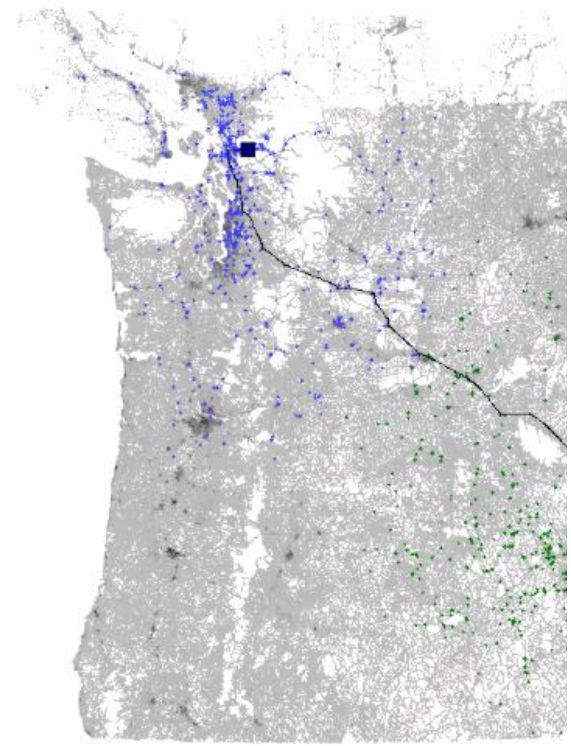
Images from Goldberg (2011), R

ortcuts

- ıdd in vertices to reduce the number of shortest paths through a l
  ⁱ nodes
  - Reduces the Reach of intermediate nodes

- ın prune more paths as a result





Images from Goldberg (2011), R

ıpact of Landmark, Reach and Shortcuts

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

ɔd	preprocessing minutes	MB	query avgscan	maxscan	ms
ctional Dijkstra	—	28	518 723	1 197 607	340.74
	4				
	1 100				
+Short	17				
+Short+ALT	21				

North America (30M vertices), random queries, 16 landm

method	preprocessing hours	GB	query avgscan	maxscan	
Bidirectional Dijkstra	—	0.5	10 255 356	27 166 866	7 6
ALT	1.6	2.3	250 381	3 584 377	3
Reach	impractical				
Reach+Short	11.3	1.8	14 684	24 618	
Reach+Short+ALT	12.9	3.6	1 595	7 450	

ɭberg (2011), Reach for A*

ore Resources

test version of slides explaining Landmarks, Reach, and Shortcuts
:tp://www.columbia.edu/~cs2035/courses/ieor6614.S16/goldberg.pdf)

ıper Introducing Landmarks
:tp://www.cs.princeton.edu/courses/archive/spr06/cos423/Handouts/GH05.pdf)

ıper Introducing Reach (http://www.siam.org/meetings/alenex04/abstacts/rgutman1.p

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

›ogle Maps – Transfer Patterns

ıdd time dependent nodes to your graph and precompute lowest ›st routes for disjoint subsets (http://ad-publications.informatik.uni-·iburg.de/ESA_transferpatterns_BCEGHRV_2010.pdf)

ıcode time domain data in frequency space and modify Dijkstra t‹ ork in that sparse representation (http://ad-publications.informatik.uni-·iburg.de/SIGSPATIAL_frequency_BS_2014.pdf)

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