Highway Dimension and Provably Efficient Shortest Paths Algorithms

Andrew V. Goldberg
Microsoft Research – Silicon Valley
www.research.microsoft.com/∼goldberg/

Joint with Ittai Abraham, Amos Fiat, and Renato Werneck

Theory & Practice

[Anonymous]

- Theory is when you know something, but it doesn't work.
- Practice is when something works, but you don't know why.
- Programmers combine theory and practice: Nothing works and they don't know why.

Theory & Practice

[Anonymous]

- Theory is when you know something, but it doesn't work.
- Practice is when something works, but you don't know why.
- Programmers combine theory and practice: Nothing works and they don't know why.

Algorithm Engineering:
Know something $\Rightarrow$ make it work.

Reverse Engineering:
Something works $\Rightarrow$ explain why (this talk).

Theory & Practice

[Anonymous]

- Theory is when you know something, but it doesn't work.
- Practice is when something works, but you don't know why.
- Programmers combine theory and practice: Nothing works and they don't know why.

Algorithm Engineering:
Know something $\Rightarrow$ make it work.

Reverse Engineering:
Something works $\Rightarrow$ explain why (this talk).

Efficient driving directions: How and why modern routing algorithms work.

Motivation

Computing driving directions.

Recent shortest path algorithms (exact):

- Arc flags [Lauther 04, Köhler et al. 06].
- A^* with landmarks [Goldberg & Harrelson 05].
- Reach [Gutman 04, Goldberg et al. 06].
- Highway [Sanders & Schultes 05] and contraction [Geisberger 08] hierarchies.
- Transit nodes [Bast et al. 06].
- Combinations (pruning and directing algorithms).

Good performance on continent-sized road networks:
Tens of millions of vertices, a few hundred vertices scanned, under a millisecond on a server, about 0.1 second on a mobile device. Used in practice.

Result Summary

Elegant and practical algorithms, but...

Until now: No analysis for non-trivial network classes and no understanding of what makes the algorithms work.

Our results:

- We define highway dimension (HD).
- Give provably efficient implementations of several recent algorithms under the small HD assumption.
- Analysis highlights algorithm similarities.
- Give a generative model of small HD networks (road network formation).

Compare to the small world model [Milgram 67, Kleinberg 99].

Outline

- Dijkstra's algorithm review.
- Shortcuts and CH algorithm.
- Reach and RE algorithm.
- Transit nodes and TN algorithm.
- Highway dimension and sparse covers.
- Analysis of RE and CH.
- Generative model.
- Concluding remarks.

Definitions

- Undirected graph $G = (V, E)$, $|V| = n$, $|E| = m$.
- $\ell : E \to N$, minimum length 1, diameter D.
- Maximum degree Δ. Small for road networks, assume constant for this talk.
- Ball $B_{v,r}$ induced by vertices within distance r from v.
- For a path P, $|P|$ denotes the length of P.
- Query: find the shortest path from s to t.

Algorithms with preprocessing:

- Two phases, preprocessing and query.
- Preprocessing output not much bigger than the input.
- Preprocessing may use more resources than queries.
- Practical preprocessing, fast queries.

Three Intuitive Methods

- Reach pruning (RE): Local intersections far from origin/destination can be ignored.
- Transit nodes (TN): All long shortest paths to/from a region pass through a small number of nodes.
- Highway/contraction hierarchies (CH): Shortest path goes from local roads to local highways to global highways to local highways to local roads.

These ideas can be mathematically formalized and lead to provably correct algorithms which work very well on road networks.

HD intuition based on TN; all preprocessing is based on CH; query analysis in this talk is for RE.

Scanning Method

- For each vertex v maintain its distance label $d_s(v)$ and status $S(v) \in \{\texttt{unreached}, \texttt{labeled}, \texttt{scanned}\}$.
- Unreached vertices have $d_s(v) = \infty$.
- If $d_s(v)$ decreases, v becomes labeled.
- To scan a labeled vertex v, for each arc (v, w), if $d_s(w) > d_s(v) + \ell(v, w)$ set $d_s(w) = d_s(v) + \ell(v, w)$.
- Initially all vertices are unreached.
- Start by decreasing $d_s(s)$ to 0.
- While there are labeled vertices, pick one and scan it.
- Different selection rules lead to different algorithms.

Dijkstra's Algorithm

[Dijkstra 1959], [Dantzig 1963].

- At each step scan a labeled vertex with the minimum label.
- Stop when t is selected for scanning.

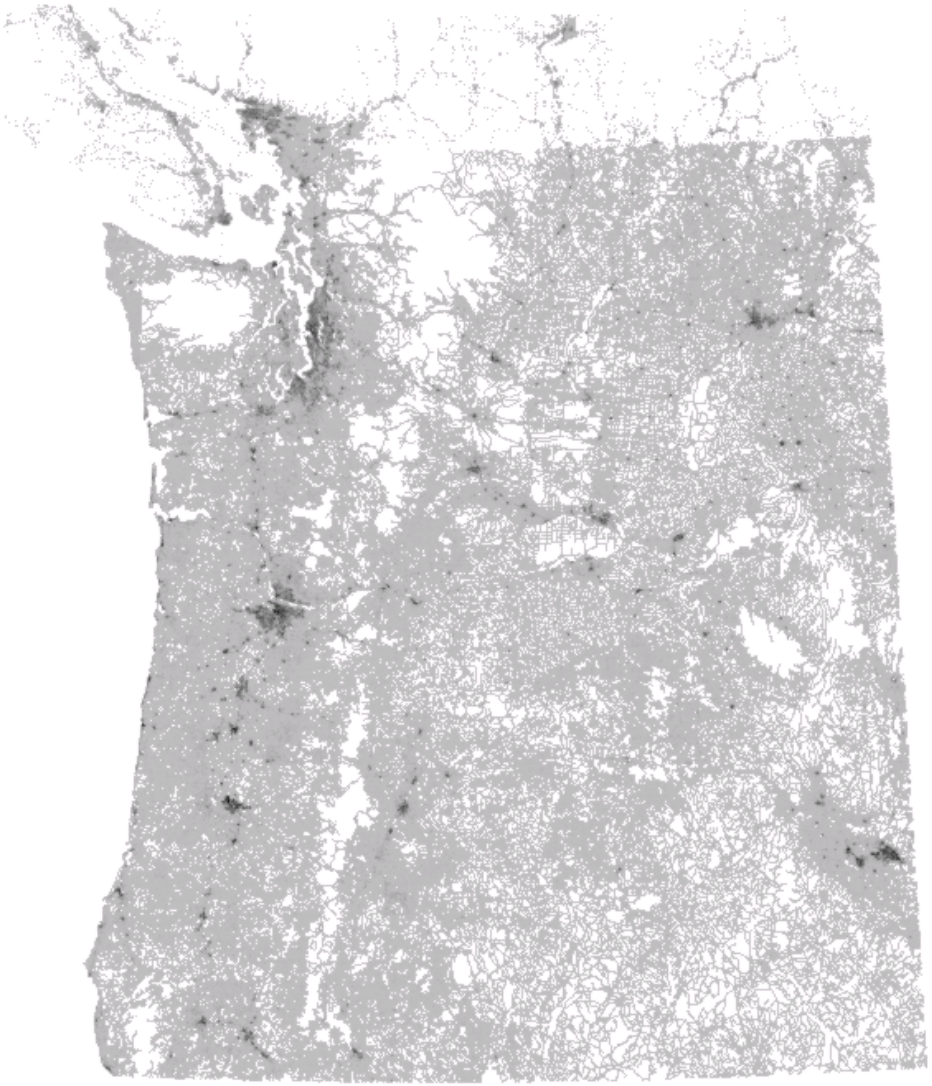
Work almost linear in the visited subgraph size.

Reverse Algorithm: Run algorithm from t in the graph with all arcs reversed, stop when s is selected for scanning.

Bidirectional Algorithm

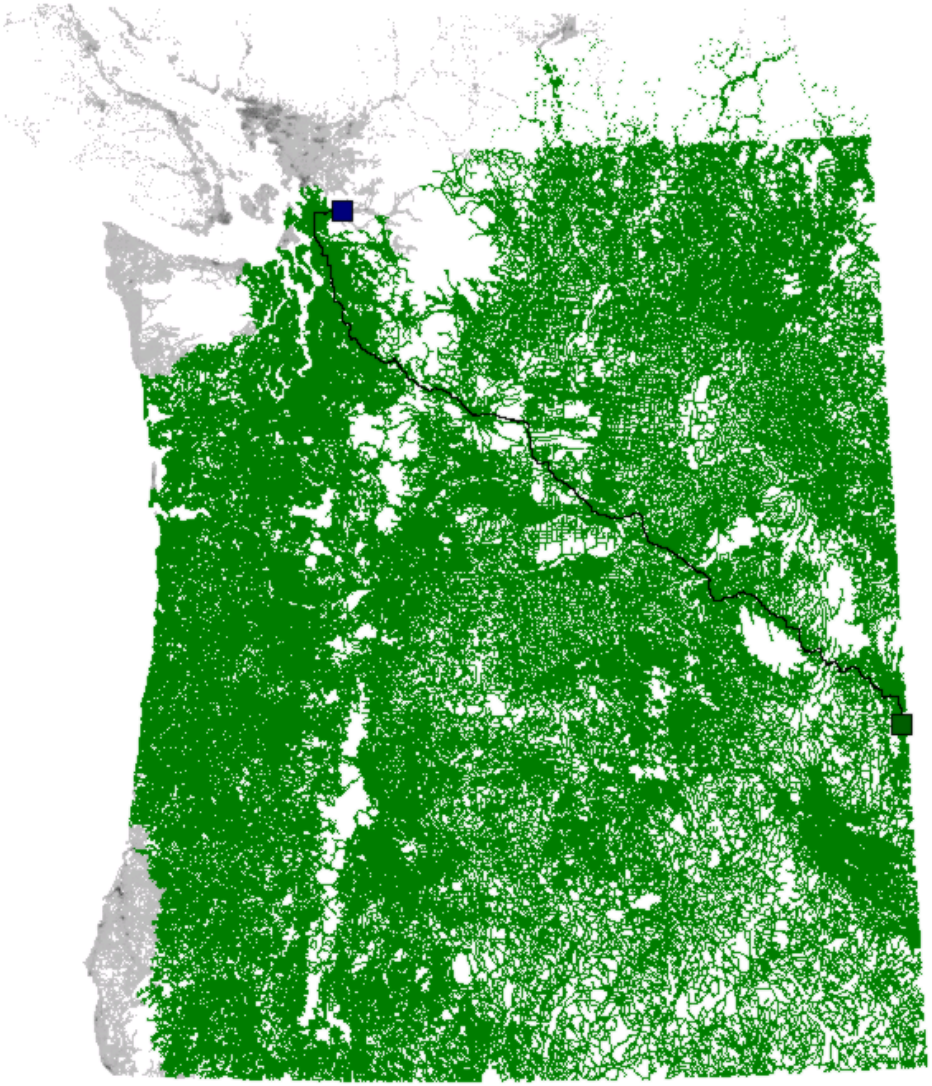
- Run forward Dijkstra from s and backward from t.
- Stopping criteria (careful!).

Example Graph



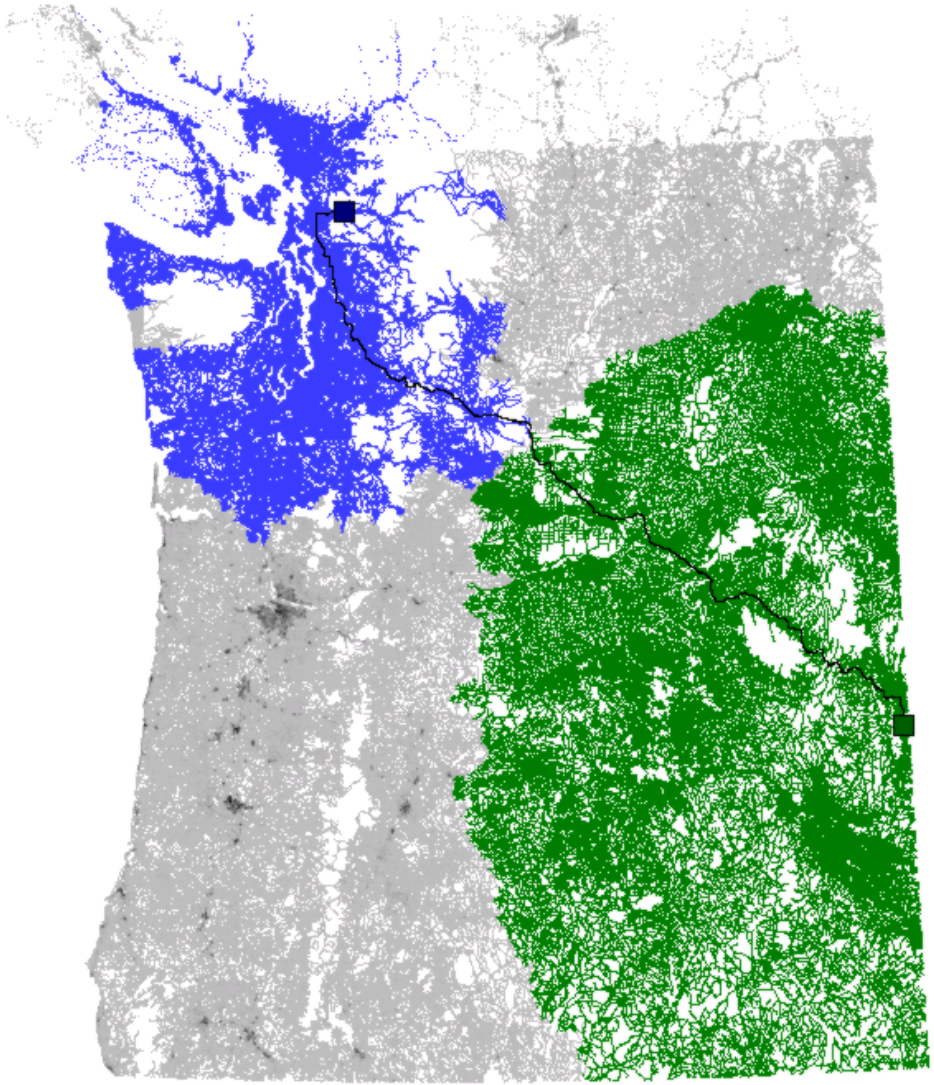
1.6M vertices, 3.8M arcs, travel time metric.

Dijkstra's Algorithm



Searched area

Bidirectional Algorithm

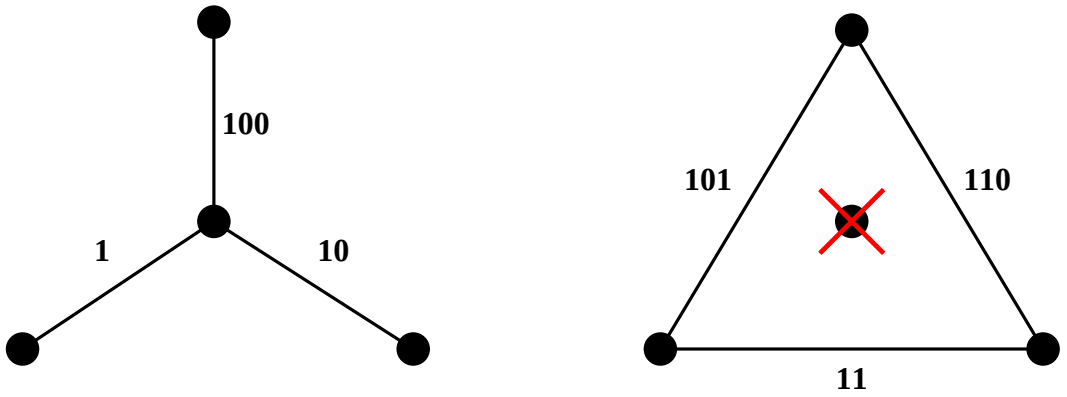


forward search/ reverse search

Contraction Hierarchies

[Geisberget et al. 08] Based on preprocessing that orders vertices and adds shortcuts. Fast and simple!

Shortcuts:



A shortcut arc can be omitted if redundant (alternative path exists).

CH Preprocessing

(1) Heuristically order vertices.

(2) Shortcut vertices in that order.

(3) To the original graph, add all shortcuts introduced in step 2.

For road networks, a good heuristic ordering is fast and increases the number of arcs by a small constant factor.

Intuition: Higher-order vertices are more important ("global"). Query is bidirectional, each search goes from lower to higher order vertices.

CH Query

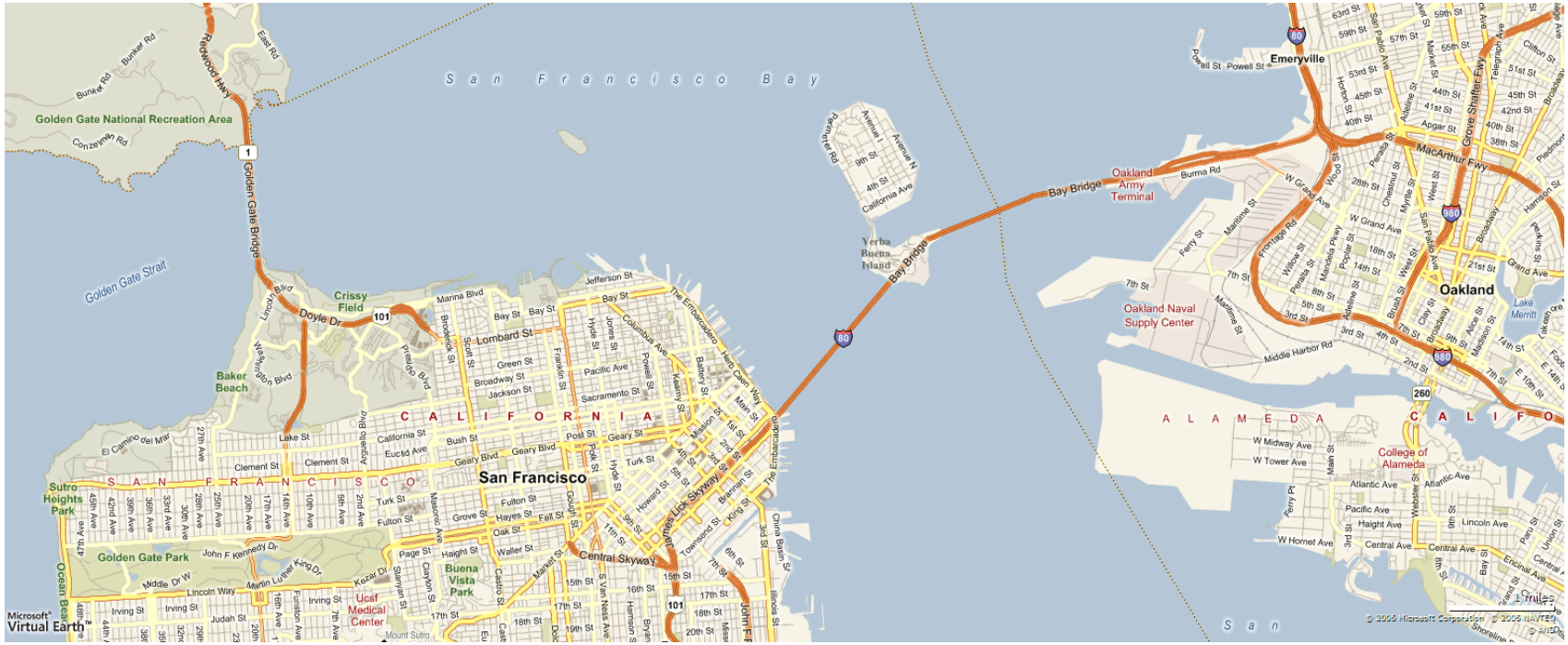
Let $(V, E \cup E^+)$ be the graph augmented by shortcuts, recall that the vertices are ordered. Let G_f be the directed graph induced by
$(v, w) : (v, w) \in E \cup E^+,\ v < w$
and G_r − by
$(w, v) : (v, w) \in E \cup E^+,\ v > w$.

Given a query, run a modified bidirectional Dijkstra's algorithm, with the forward search in G_f and the reverse search in G_r. (Both searches go "up".)

Upon termination $\text{dist}(s, t) = \min_v(d(s, v) + d(v, t))$.

Experiments show that for road networks queries look at a very small subgraph, even in the worst case.

Reach Intuition



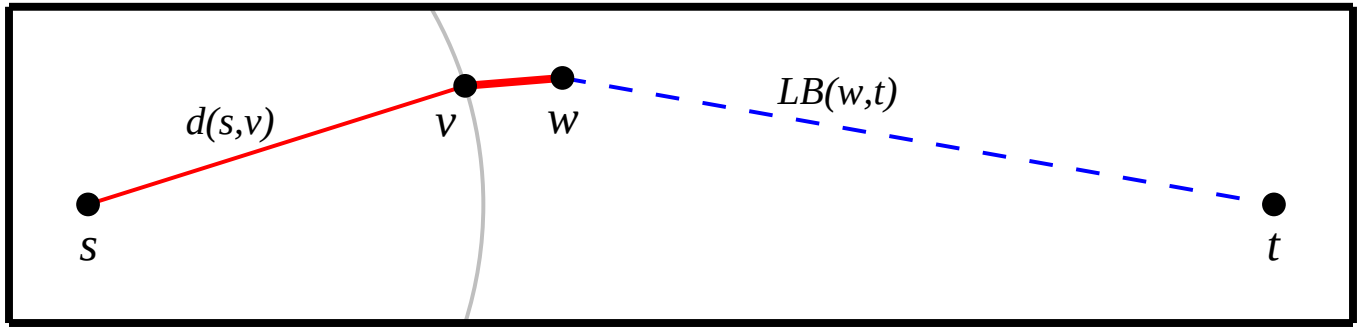
Identify local intersections and prune them when searching far from s and t.

Reach Definition

[Gutman 04]

- Consider a vertex v that splits a path P into P_1 and P_2. $r_P(v) = \min(\ell(P_1), \ell(P_2))$.
- $r(v) = \max_P(r_P(v))$ over all shortest paths P through v.

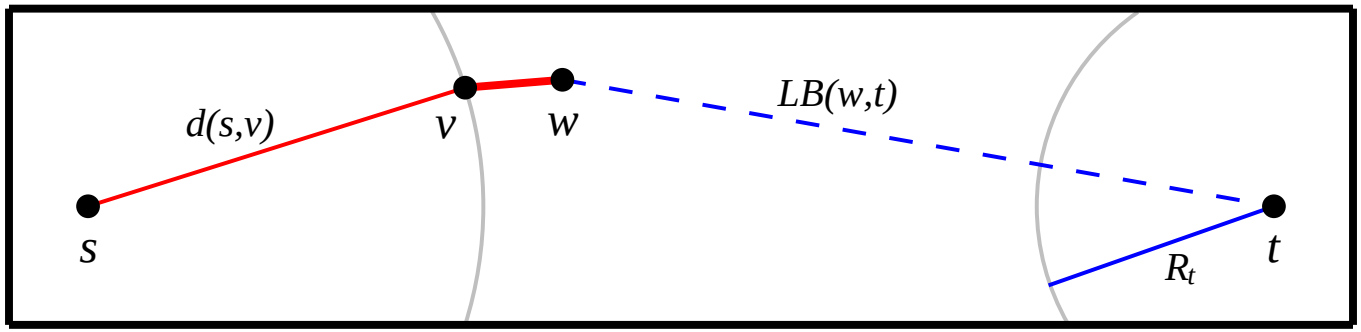
Using reaches to prune Dijkstra:



If $r(w) < \min(d(v) + \ell(v,w), LB(w,t))$ then prune w.

Lower Bounds for Nothing

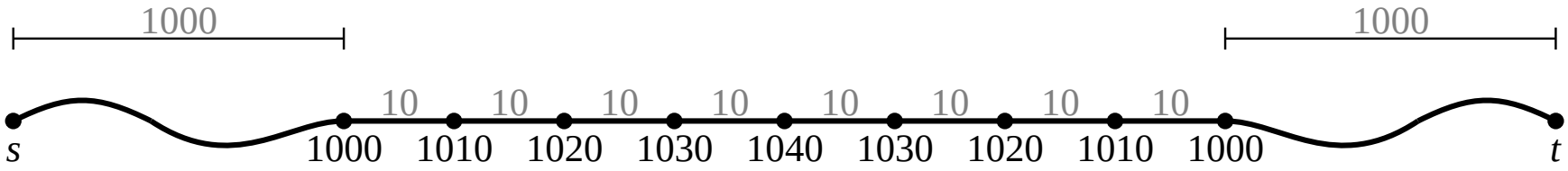
Bidirectional search gives implicit bounds (R_t below).



Reach-based query algorithm is Dijkstra's algorithm with pruning based on reaches. Given a lower-bound subroutine, a small change to Dijkstra's algorithm.

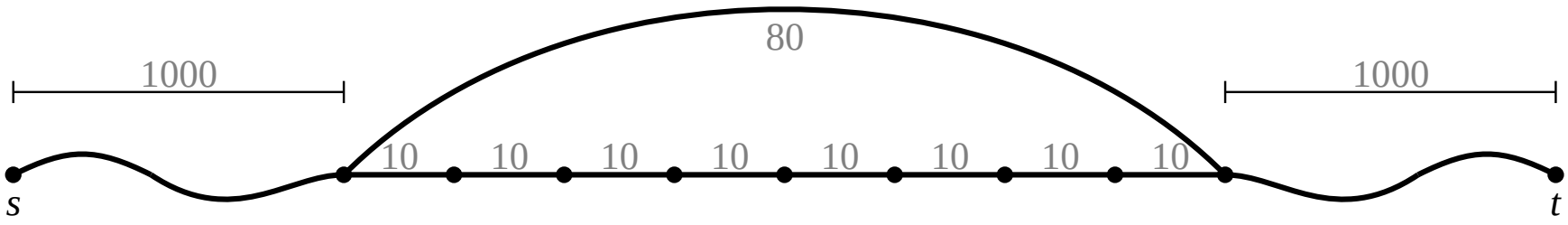
Shortcuts

- Consider the graph below.
- Many vertices have large reach.



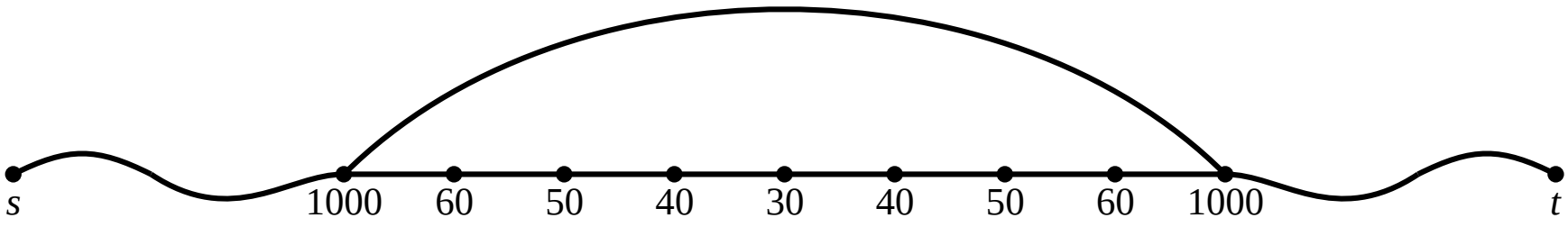
Shortcuts

- Consider the graph below.
- Many vertices have large reach.
- Add a shortcut arc, break ties by the number of hops.



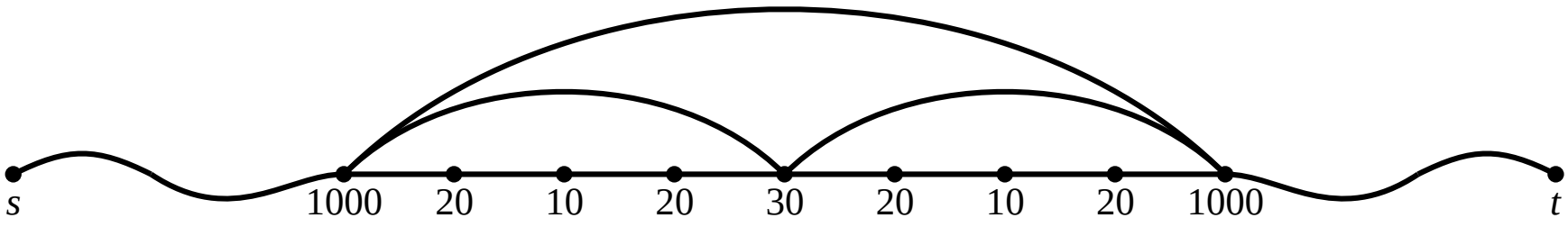
Shortcuts

- Consider the graph below.
- Many vertices have large reach.
- Add a shortcut arc, break ties by the number of hops.
- Reaches decrease.



Shortcuts

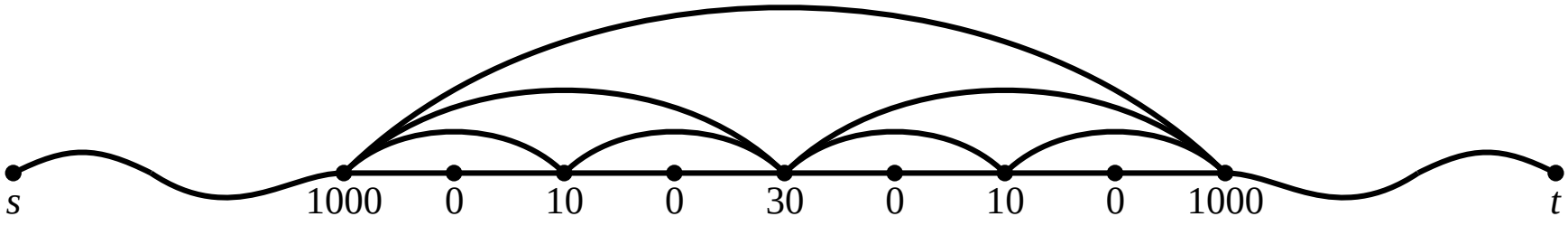
- Consider the graph below.
- Many vertices have large reach.
- Add a shortcut arc, break ties by the number of hops.
- Reaches decrease.
- Repeat.



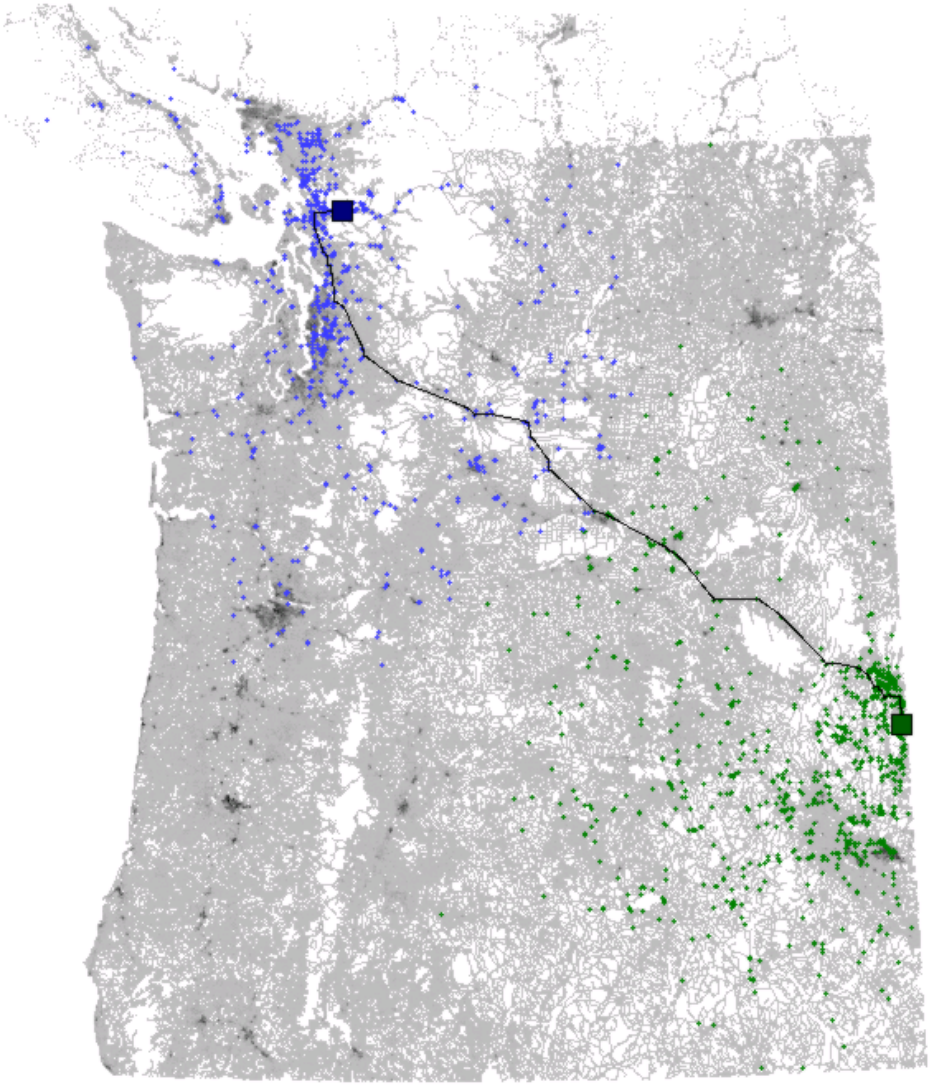
Reach Algorithm

- Consider the graph below.
- Many vertices have large reach.
- Add a shortcut arc, break ties by the number of hops.
- Reaches decrease.
- Repeat.
- A small number of shortcuts can greatly decrease many reaches.

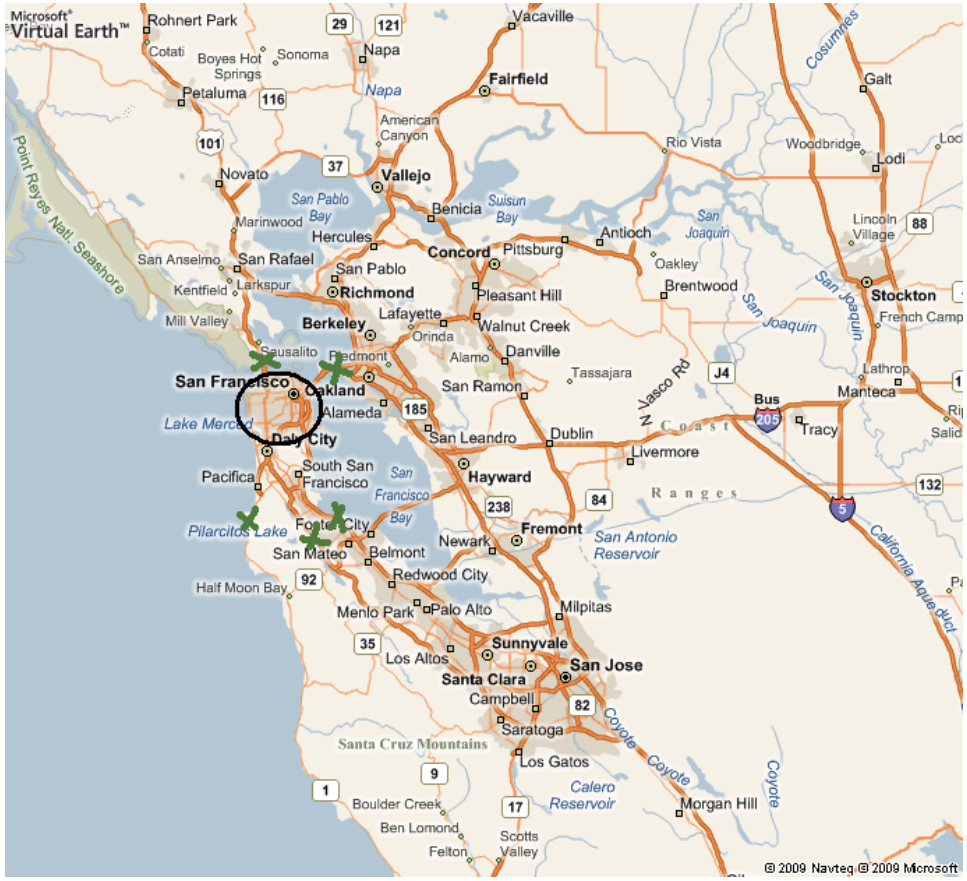
RE algorithm: reaches with shortcuts, bidirectional Dijkstra, balance the searches by the search radius.



Reach Algorithm



TN Algorithm: Intuition



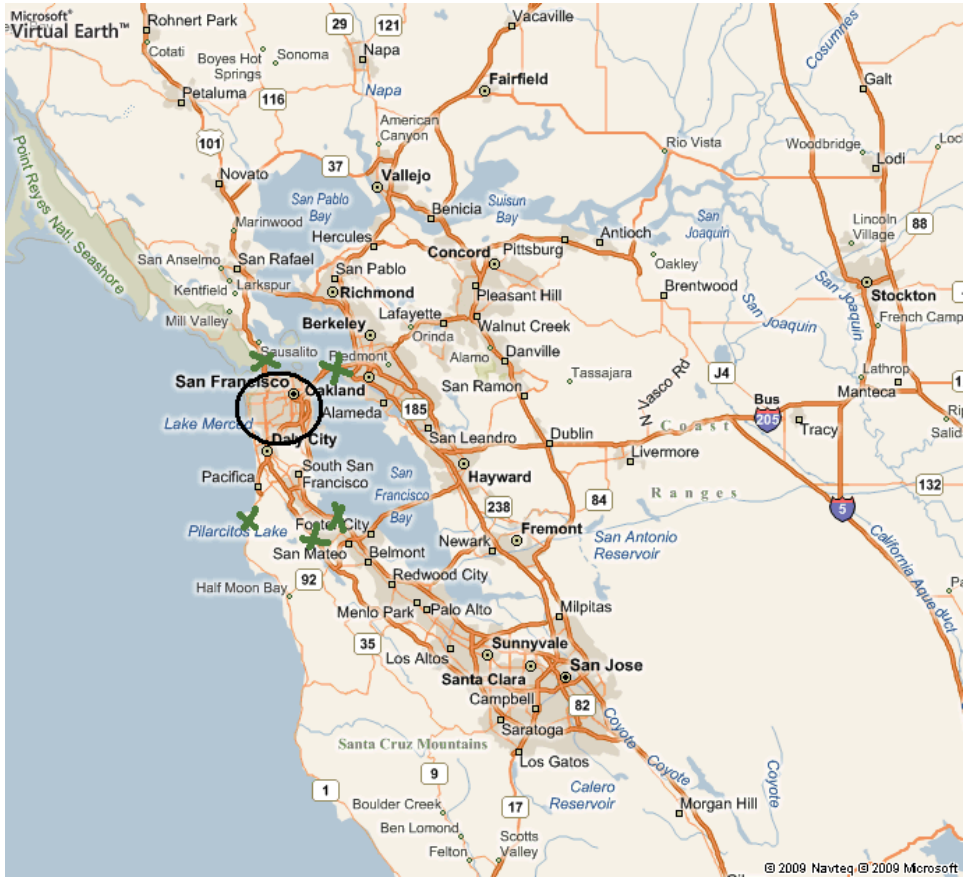
For a region, there is a small set of nodes such that all sufficiently long shortest paths out of the region pass a vertex in the set.

Transit Node (TN) Algorithm

[Bast et. al 06]

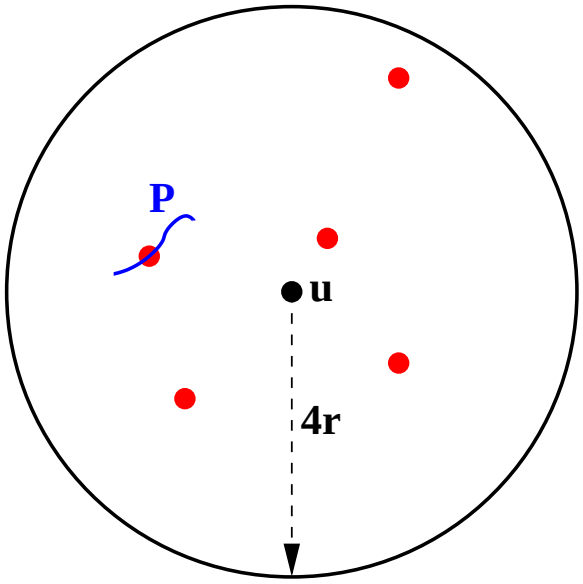
- Divide a map into regions.
- For each region, paths to far away places pass through a small number of access points.
- The union of all access nodes is the set of transit nodes.
- **Preprocessing:** find all access points, connect each vertex to its access nodes, compute all pairs of shortest paths between transit nodes.
- **Query:** Look for a "local" shortest path or a shortest path of the form origin – transit node – transit node – destination.
- Various ways to speed up local queries.
- 10 access nodes per region on the average.
- Leads to the fastest algorithms.

Highway Dimension: Intuition



TN algorithm works because for a region, there is a small set of nodes such that all sufficiently long shortest paths out of the region pass a vertex in the set.

Highway Dimension



Highway dimension (HD) h:

$\forall \quad r \in \Re, \forall u \in V, \exists S \subseteq B_{u,4r}, |S| \leq h,$ such that

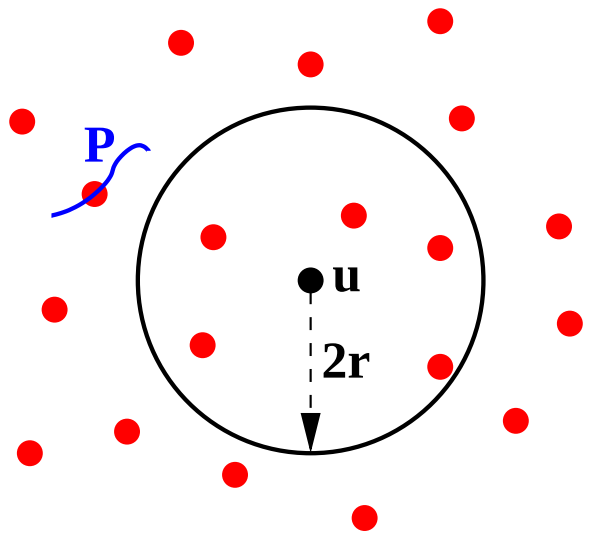
$\forall \quad v, w \in B_{u,4r},$

if P is a SP: $|P(v,w)| > r$ and $P(v,w) \subseteq B_{u,4r}$,

then $P(v,w) \cap S \neq \emptyset$.

Locally, a small set covers all long SPs.

Shortest Path Cover



(r, k) **Shortest path cover ((r, k)-SPC):** a set C such that

$$\forall \quad \text{SP } P : \; r < |P| \leq 2r \Rightarrow$$
$$P \cap C \neq \emptyset \quad \text{and}$$
$$\forall u \in V, \; |C \cap B_{u,2r}| \leq k$$

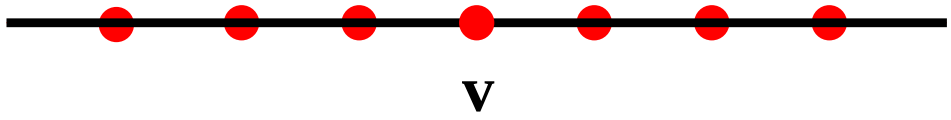
All SPs in a range can be covered by a sparse set.

Can use constants different from 4 and 2, but the constants are related.

Highway vs. Doubling Dimension

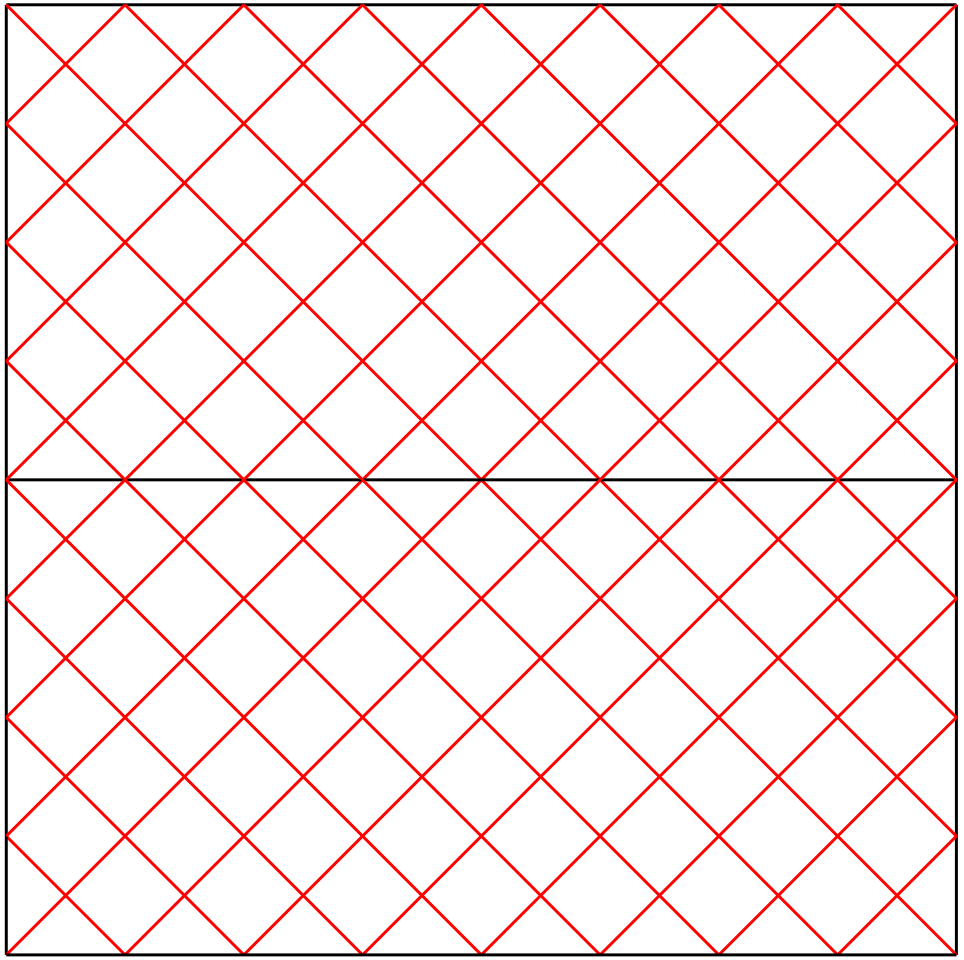
A metric space has a doubling dimension α if every ball or radius r can be covered by 2^{α} balls of radius $r/2$.

Line: Small HD



A line has HD 7 and doubling dimension 1.

Grid: High HD



A grid has HD $\Theta(\sqrt{n})$ and the doubling dimension 2.

HD vs. SPC

Theorem. If G has highway dimension h, then for any r there exists an (r, h)-SPC.
Proof idea: Show that S^*, the smallest set that covers all shortest paths $P : r < |P| \leq 2r$, is an (r, h)-SPC.

Finding S^* is NP-hard. Efficient construction?

Theorem. If G has highway dimension h, then for any r we can construct, in polynomial time, an $(r, O(h \log n))$-SPC.
Proof idea: Use the greedy set-cover algorithm to get an $O(\log n)$ approximation of S^*.

Proofs depend on the choice of constants in the definitions.

Generic Preprocessing

Let k denote h or $O(h \log n)$ (for poly-time preprocessing).

- Let $S_0 = V$. For $1 \leq i \leq \log D$ build $(2^i, k)$-SPC covers S_i.
- Let $L_i = S_i - \bigcup_{i+1}^{\log D} S_j$ (vertex layers).
- Order vertices so that L_i comes before L_{i+1}; ordering inside layers is arbitrary.
- Do shortcutting in this order to get E^+.

Lemma. Let $v \in L_i$ and $j \geq i$. Then the number of $(v, w) \in E^+$ with $w \in L_j$ is at most k.

Proof. (v, w) corresponds to P with internal vertices less than v, w. Thus $w \in B_{v,2 \cdot 2^i}$. The SPC definition implies the lemma.

Theorem. In $(V, E \cup E^+)$ vertex degrees are bounded by $\Delta + k \log D$ and $|E \cup E^+| = O(m + nk \log D)$.

Additional RE Preprocessing

Reach bounds are in the graph with shortcuts.

Lemma: If $v \in L_i$ then reach$(v) \leq 2 \cdot 2^i$.
Proof: Suppose the reach is greater. Then there is a shortest path P that v divides into P_1 and P_2 with $|P_1|, |P_2| > 2 \cdot 2^i$. Both P_1 and P_2 contain vertices in L_j with $j > i$, so there is a shortcut from P_1 to P_2. But then P is not a shortest path.

Additional work is linear.

RE Query Bound

Theorem: RE query takes $O((k \log D)^2)$ time.

Proof: Consider a forward search from s. In $B_{s,2 \cdot 2^i}$, the search scans only vertices of L_i in $B_{s,2 \cdot 2^i}$. Thus $O(k \log D)$ scans.

- Shortest path can be extracted in time linear in the number of its arcs.
- Similar analysis for CH yields the same bound.
- Also develop a faster version of TN: $O(k \log D)$ query.

Network Formation

Natural networks with constant highway dimension?

Attempt to model road networks:

- Build on the Earth surface (low doubling dimension).
- Build in decentralized and incremental manner.
- Highways are faster than local roads.

Capture some, but not all, real-life properties.

Setting:

- Metric space (M, d), doubling dimension $\log \alpha$, diameter D.
- Speedup parameter $\delta \in (0, 1)$.
- Edge $\{v, w\}$ has length $d(v, w)$ and transit time $\tau(v, w) = d^{1-\delta}(v, w)$. (Long roads are fast.)
- On-line network formation, adversarial (or random) vertex placement.

Network Formation (cont.)

In the spirit of dynamic spanners [Gottlieb & Roditty 08].

- Adversary adds vertices, we connect them.
- Intuition: connect a new village to all nearby villages and to the closest town.
- Formally: maintain covers C_i for $0 \leq i \leq \log D$.
  $C_0 = V$, $C_{i+1} \subseteq C_i$, vertices in C_i are at least 2^i apart.
- When adding a new vertex v, add v to $C_0, \ldots, C_i$ for appropriate i. (The first vertex added to all C's.)
- For $0 \leq j \leq i$, connect v to $C_j \cap B_{v,6 \cdot 2^j}$.
- If $i < \log D$, connect v to the closest element of C_{i+1}.

Theorem: The network has highway dimension of $\alpha^{O(1)}$.

Concluding Remarks

- Highway dimension explains why intuitive algorithms work well on a non-trivial graph class.
- Unified view of several different-looking algorithms.
- Extensions for directed graphs.
- Additional assumptions, such as small separators, may lead to better bounds.
- For real road networks, small HD condition may not hold for some r and some v.
- Does low HD make other problems simpler?
- Computing HD dimension of road networks (work in progress with Daniel Delling): USA/Europe highway dimension is $\approx 1,000$, access dimension ≈ 20.

Thank You!