Highway Dimension and Provably Efficient Shortest Path Algorithms

Andrew V. Goldberg

Microsoft Research – Silicon Valley
http://research.microsoft.com/∼goldberg/

(Joint work with Ittai Abraham, Amos Fiat, and Renato Werneck)

Motivation: Computing Driving Directions

Outline

Theory vs. Practice

[Anonymous]

- Theory is when you know something, but it doesn't work.

Theory vs. Practice

[Anonymous]

- Theory is when you know something, but it doesn't work.
- Practice is when something works, but you don't know why.

Theory vs. 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 vs. 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.

Bridging the theory–practice gap:

Theory vs. 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.

Bridging the theory–practice gap:

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

MIND THE GAP

Theory vs. 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.

Bridging the theory–practice gap:

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

Natural Science
Something works $\Rightarrow$ explain why.

Theory vs. 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.

Bridging the theory–practice gap:

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

Natural Science
Something works $\Rightarrow$ explain why.

This talk
How and why modern routing algorithms work.

Recent Developments

Continent-sized road networks have 10s of millions intersections.

Dijkstra's algorithm: ≈ 5 s

Recent Developments

Continent-sized road networks have 10s of millions intersections.

Dijkstra's algorithm: $\approx$ 5 s

Recent work

- Arc flags [Lauther 04, Köhler et al. 06].
- A^* with landmarks [Goldberg & Harrelson 05].
- Reach [Gutman 04, Goldberg et al. 06].
- Highway hierarchies [Sanders & Schultes 05].
- Contraction hierarchies [Geisberger et al. 08].
- Transit nodes [Bast et al. 06].
- DIMACS Shortest Paths Implementation Challenge (2005–2006).

Recent Developments

Continent-sized road networks have 10s of millions intersections.

Dijkstra's algorithm: $\approx$ 5 s

Recent work

- Arc flags [Lauther 04, Köhler et al. 06].
- A^* with landmarks [Goldberg & Harrelson 05].
- Reach [Gutman 04, Goldberg et al. 06].
- Highway hierarchies [Sanders & Schultes 05].
- Contraction hierarchies [Geisberger et al. 08].
- Transit nodes [Bast et al. 06].
- DIMACS Shortest Paths Implementation Challenge (2005–2006).

Greatly improved performance: < 1 ms, ≈ 0.1 s on a mobile device.
Only a few hundred intersections searched.

Definitions and Model

Input

- Graph $G = (V, E)$ (intersections, road segments), $|V| = n$, $|E| = m$.
- Weight function ℓ (length, transit time, fuel consumption, ...).
- Static problem, G and ℓ incorporate all modeling information.

Definitions and Model

Input

- Graph $G = (V, E)$ (intersections, road segments), $|V| = n$, $|E| = m$.
- Weight function ℓ (length, transit time, fuel consumption, ...).
- Static problem, G and ℓ incorporate all modeling information.

Query (multiple times for the same input network)

- Given origin s and destination t, find optimal path from s to t.
- Exact algorithms help modeling and debugging.

Definitions and Model

Input

- Graph $G = (V, E)$ (intersections, road segments), $|V| = n$, $|E| = m$.
- Weight function ℓ (length, transit time, fuel consumption, ...).
- Static problem, G and ℓ incorporate all modeling information.

Query (multiple times for the same input network)

- Given origin s and destination t, find optimal path from s to t.
- Exact algorithms help modeling and debugging.

Algorithms with preprocessing

- Two phases: practical preprocessing and real-time queries.
- Preprocessing output not much bigger than the input.
- Preprocessing may use more resources than queries.

Dijkstra's Algorithm

[Dijkstra 1959], [Dantzig 1963].

Dijkstra's Algorithm

- Examine vertices in the order of their distance from s.
- Stop when t is reached.

Dijkstra's Algorithm

[Dijkstra 1959], [Dantzig 1963].

Dijkstra's Algorithm

- Examine vertices in the order of their distance from s.
- Stop when t is reached.

Reverse Algorithm

- Run algorithm from t in the graph with all arcs reversed.
- Stop when s is reached.

Dijkstra's Algorithm

[Dijkstra 1959], [Dantzig 1963].

Dijkstra's Algorithm

- Examine vertices in the order of their distance from s.
- Stop when t is reached.

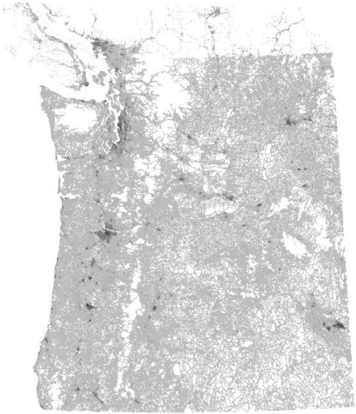
Reverse Algorithm

- Run algorithm from t in the graph with all arcs reversed.
- Stop when s is reached.

Bidirectional Algorithm

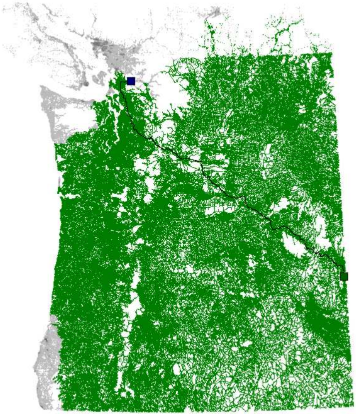
- Run forward Dijkstra from s and backward from t.
- Stop when the searches meet.

Example Graph



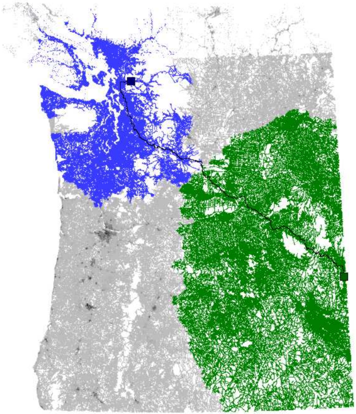
1.6M vertices, 3.8M arcs, travel time metric.

Dijkstra's Algorithm



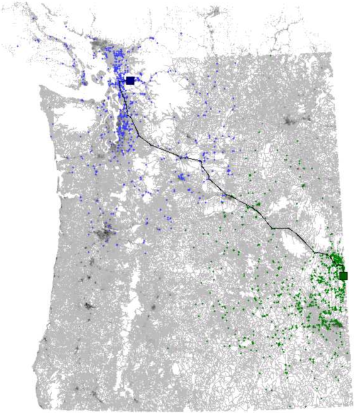
Searched area

Bidirectional Algorithm



forward search/ **reverse search**

Reach Algorithm



Pruning leads to amazing speedup.

Three Recent Algorithms

Algorithm intuition

- Reach pruning (RE): Local intersections far from origin/destination can be ignored.

Three Recent Algorithms

Algorithm intuition

- Reach pruning (RE): Local intersections far from origin/destination can be ignored.
- Highway/contraction hierarchies (CH): Shortest path goes from local roads to local highways to global highways to local highways to local roads.

Three Recent Algorithms

Algorithm intuition

- Reach pruning (RE): Local intersections far from origin/destination can be ignored.
- Highway/contraction hierarchies (CH): Shortest path goes from local roads to local highways to global highways to local highways to local roads.
- Transit nodes (TN): For any region, a small number of “toll booths” covers all sufficiently long optimal in/out paths.

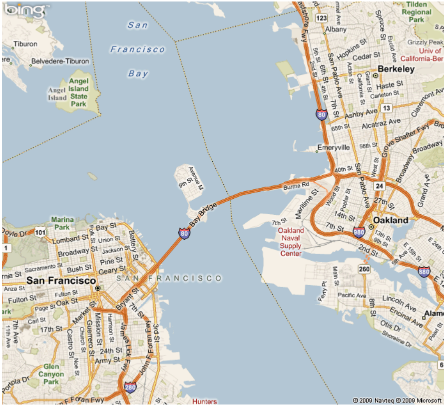
Three Recent Algorithms

Algorithm intuition

- Reach pruning (RE): Local intersections far from origin/destination can be ignored.
- Highway/contraction hierarchies (CH): Shortest path goes from local roads to local highways to global highways to local highways to local roads.
- Transit nodes (TN): For any region, a small number of “toll booths” covers all sufficiently long optimal in/out paths.

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

Reach Fundamentals

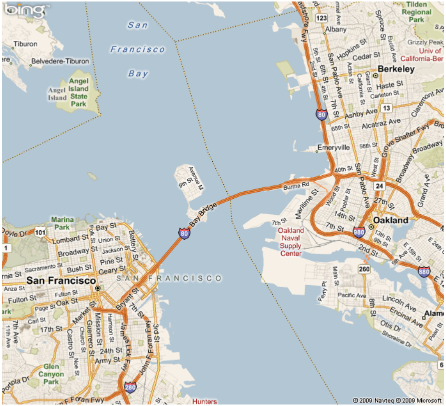


[Gutman 04; Goldberg et al. 06]

Preprocessing computes intersection locality.

Query uses locality to prune search.

Reach Fundamentals



[Gutman 04; Goldberg et al. 06]

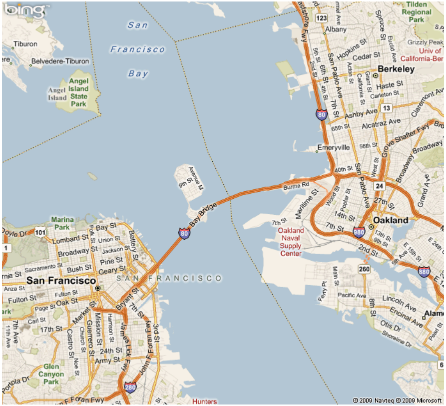
Preprocessing computes intersection locality.

Query uses locality to prune search.

Definition of reach

- Consider a vertex v that splits a path P into P_1 and P_2. $r_P(v) = \min(\ell(P_1), \ell(P_2))$.

Reach Fundamentals



[Gutman 04; Goldberg et al. 06]

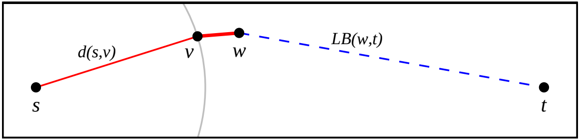
Preprocessing computes intersection locality.

Query uses locality to prune search.

Definition of reach

- 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.

Pruning Search Using Reach

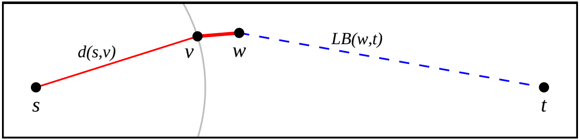


Pruning

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

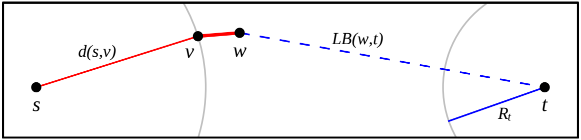
Pruning Search Using Reach

Pruning



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).

RE Algorithm

RE Query

- Bidirectional Dijkstra's algorithm with pruning based on reaches.
- A small change to Dijkstra's algorithm.

RE Algorithm

RE Query

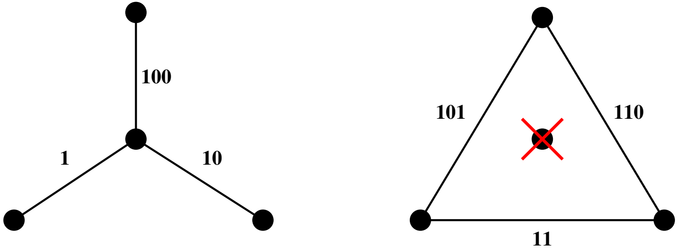
- Bidirectional Dijkstra's algorithm with pruning based on reaches.
- A small change to Dijkstra's algorithm.

Remarks

- $O(nm)$ perprocessing impractical on large graphs.
- Fast heuristic preprocessing computes reach upper bounds.
- Shortcuts speed up both preprocessing and query.
- CH algorithm shows that shortcuts are crucial.

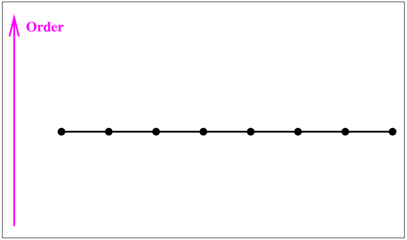
Shortcut Operation

[Sanders & Schultes 05]
The key operation for Contraction Hierarchies algorithm

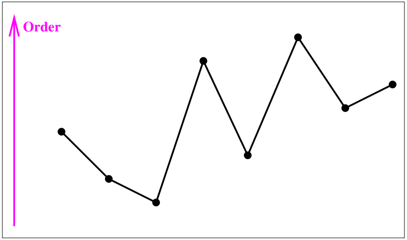


A shortcut arc can be omitted if redundant (alternative path exists).

Contraction Hierarchies



[Geisberger et al. 08]
Preprocessing orders vertices, order corresponds to locality. Both forward and reverse searches consider only “up” (more local to more global) edges. Effective pruning.

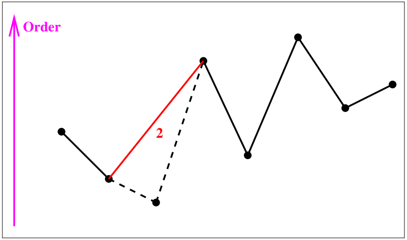


[Geisberger et al. 08]
Preprocessing orders vertices, order corresponds to locality. Both forward and reverse searches consider only “up” (more local to more global) edges. Effective pruning.

Preprocessing Algorithm

1. Heuristically order vertices.

Contraction Hierarchies

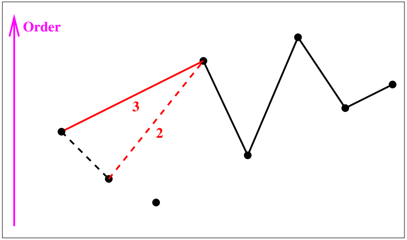


[Geisberger et al. 08]
Preprocessing orders vertices, order corresponds to locality. Both forward and reverse searches consider only “up” (more local to more global) edges. Effective pruning.

Preprocessing Algorithm

1. Heuristically order vertices.
2. Shortcut vertices in that order.

Contraction Hierarchies

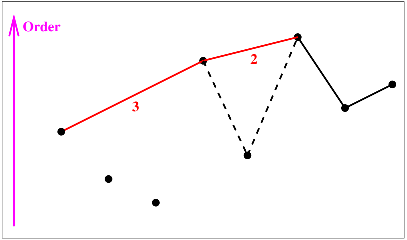


[Geisberger et al. 08]
Preprocessing orders vertices, order corresponds to locality. Both forward and reverse searches consider only “up” (more local to more global) edges. Effective pruning.

Preprocessing Algorithm

1. Heuristically order vertices.
2. Shortcut vertices in that order.

Contraction Hierarchies

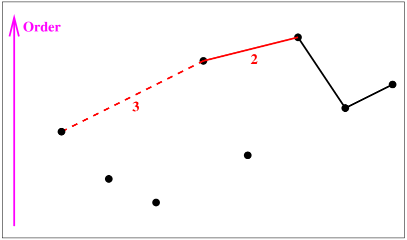


[Geisberger et al. 08]
Preprocessing orders vertices, order corresponds to locality. Both forward and reverse searches consider only “up” (more local to more global) edges. Effective pruning.

Preprocessing Algorithm

1. Heuristically order vertices.
2. Shortcut vertices in that order.

Contraction Hierarchies

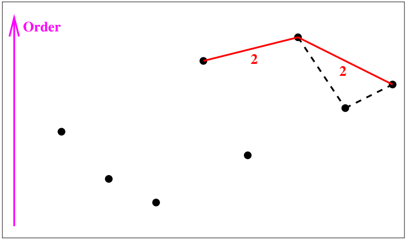


[Geisberger et al. 08]
Preprocessing orders vertices, order corresponds to locality. Both forward and reverse searches consider only “up” (more local to more global) edges. Effective pruning.

Preprocessing Algorithm

1. Heuristically order vertices.
2. Shortcut vertices in that order.

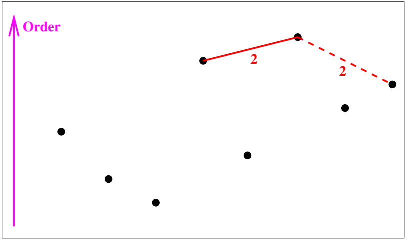
Contraction Hierarchies



[Geisberger et al. 08]
Preprocessing orders vertices, order corresponds to locality. Both forward and reverse searches consider only “up” (more local to more global) edges. Effective pruning.

Preprocessing Algorithm

1. Heuristically order vertices.
2. Shortcut vertices in that order.

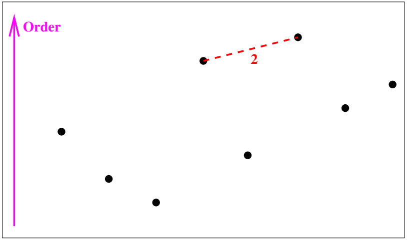


[Geisberger et al. 08]
Preprocessing orders vertices, order corresponds to locality. Both forward and reverse searches consider only "up" (more local to more global) edges. Effective pruning.

Preprocessing Algorithm

1. Heuristically order vertices.
2. Shortcut vertices in that order.

Contraction Hierarchies

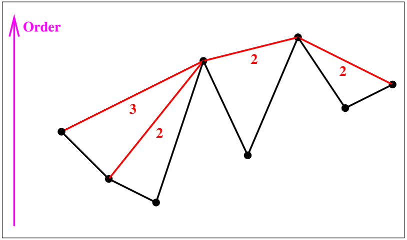


[Geisberger et al. 08]
Preprocessing orders vertices, order corresponds to locality. Both forward and reverse searches consider only "up" (more local to more global) edges. Effective pruning.

Preprocessing Algorithm

1. Heuristically order vertices.
2. Shortcut vertices in that order.

Contraction Hierarchies

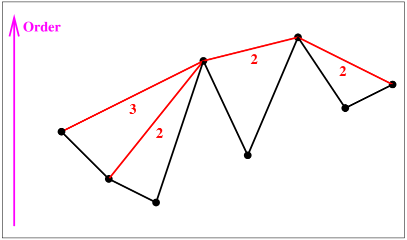


[Geisberger et al. 08]
Preprocessing orders vertices, order corresponds to locality. Both forward and reverse searches consider only “up” (more local to more global) edges. Effective pruning.

Preprocessing Algorithm

1. Heuristically order vertices.
2. Shortcut vertices in that order.
3. To the original graph, add all shortcuts introduced in step 2.

Contraction Hierarchies



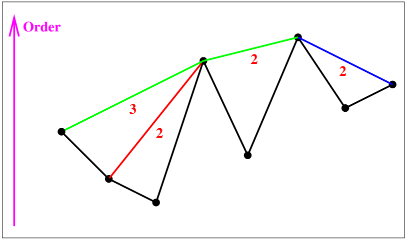
[Geisberger et al. 08]
Preprocessing orders vertices, order corresponds to locality. Both forward and reverse searches consider only "up" (more local to more global) edges. Effective pruning.

Preprocessing Algorithm

1. Heuristically order vertices.
2. Shortcut vertices in that order.
3. To the original graph, add all shortcuts introduced in step 2.

Query algorithm

- Run a modified bidirectional Dijkstra's algorithm.



[Geisberger et al. 08]
Preprocessing orders vertices, order corresponds to locality. Both forward and reverse searches consider only "up" (more local to more global) edges. Effective pruning.

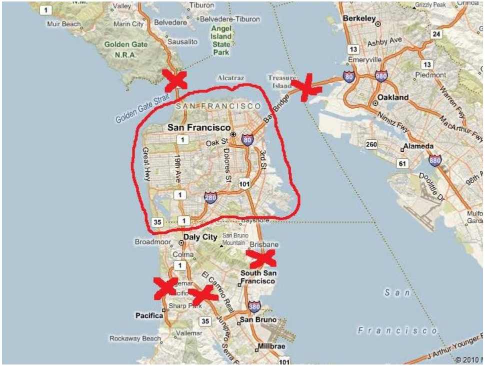
Preprocessing Algorithm

1. Heuristically order vertices.
2. Shortcut vertices in that order.
3. To the original graph, add all shortcuts introduced in step 2.

Query algorithm

- Run a modified bidirectional Dijkstra's algorithm.
- The searches only consider "up" edges.

Transit Node Intuition



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

TN Preprocessing

[Bast et al. 06]

Basic concepts

- Divide a map into regions (a few thousand).
- For each region, optimal paths to far away places pass through one of a small number of access nodes (≈ 10 on the average).
- The union of access nodes is the set of transit nodes ($\approx 10\,000$).

TN Preprocessing

[Bast et al. 06]

Basic concepts

- Divide a map into regions (a few thousand).
- For each region, optimal paths to far away places pass through one of a small number of access nodes (≈ 10 on the average).
- The union of access nodes is the set of transit nodes ($\approx 10\,000$).

Preprocessing Algorithm

- Find access nodes for every region.
- Connect each vertex to its access nodes.
- Compute all pairs of shortest paths between transit nodes.

TN Query

Long-range query algorithm

- The shortest path has the form s – access(s) – access(t) – t

TN Query

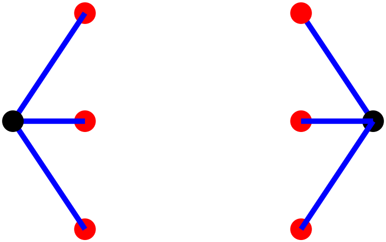
Long-range query algorithm

- The shortest path has the form s – access(s) – access(t) – t

TN Query

Long-range query algorithm

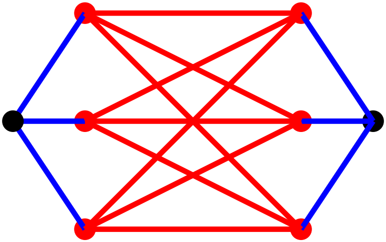
- The shortest path has the form s – access(s) – access(t) – t



TN Query

Long-range query algorithm

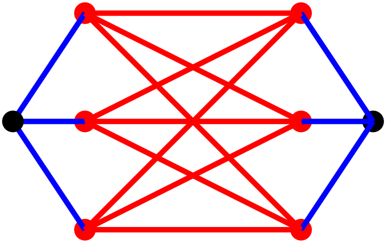
- The shortest path has the form s – access(s) – access(t) – t



TN Query

Long-range query algorithm

- The shortest path has the form s – access(s) – access(t) – t
- Table look-up for the (access(s), access(t)) node pairs.



Remarks

- Very fast: 10×10 table look-ups per long-range query.
- Local queries: another method or hierarchical approach.

Theoretical Results

Practice

- Intuitive and practical algorithms, but...
- Why do they work well on road networks?
- What is a road network (formally)?

Theoretical Results

Practice

- Intuitive and practical algorithms, but...
- Why do they work well on road networks?
- What is a road network (formally)?

Theory [Abraham, Fiat, Goldberg & Werneck 10]

- Define highway dimension (HD).
- Good time bounds for the three algorithms assuming HD is small.
- Analysis highlights algorithm similarities.
- Generative model of small HD networks (road network formation).

In the spirit of the small world model [Milgram 67, Kleinberg 99].

Definitions and Remarks

Definitions and assumptions

- Constant maximum degree.
- $B_{v,r}$ denotes the set of vertices within distance r from v.
- $|P|$ denotes the length of P.
- h denotes highway dimension.
- k denotes either h or $O(h \log n)$ (exponential or poly-time preprocessing).
- Network diameter D.

Definitions and Remarks

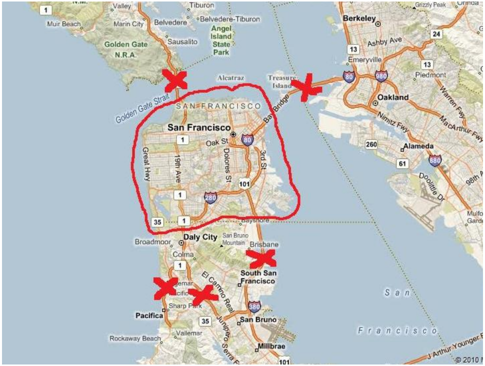
Definitions and assumptions

- Constant maximum degree.
- $B_{v,r}$ denotes the set of vertices within distance r from v.
- $|P|$ denotes the length of P.
- h denotes highway dimension.
- k denotes either h or $O(h \log n)$ (exponential or poly-time preprocessing).
- Network diameter D.

Remarks

- HD definition motivated by Transit Nodes.
- Preprocessing based on Contraction Hierarchies ideas.
- Analysis based on Reach ideas.

Highway Dimension Motivation



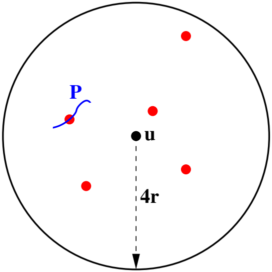
For a region, there is a small set of nodes such that all sufficiently long shortest paths out of the region pass a node in the set.

Highway Dimension Definition

Locally, a small set covers all long SPs.

Highway Dimension Definition

Locally, a small set covers all long SPs.



Highway dimension (HD) h

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

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

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

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

Highway vs. Doubling Dimension

Definition

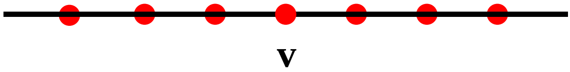
A metric space has a doubling dimension α if every ball of radius r can be covered by 2^α balls of radius $r/2$.

Highway vs. Doubling Dimension

Definition

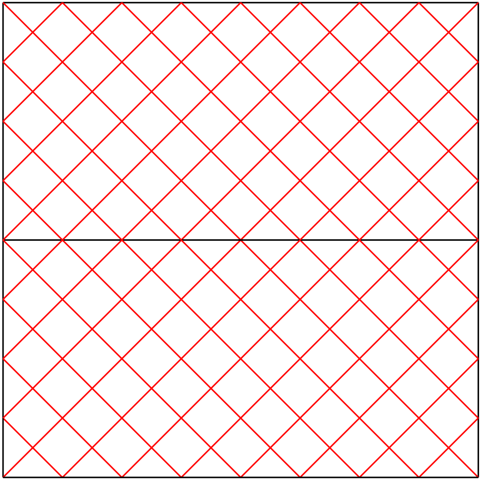
A metric space has a doubling dimension α if every ball of radius r can be covered by 2^α balls of radius $r/2$.

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.

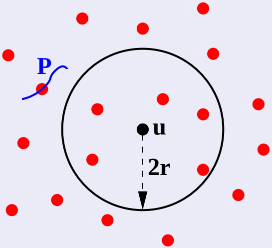
Shortest Path Covers

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

(r, k) Shortest path cover ((r, k)-SPC):

A set C such that

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



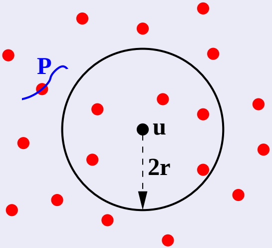
Shortest Path Covers

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

(r,k) Shortest path cover ((r,k)-SPC):

A set C such that

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



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

HD vs. SPC

Theorem

If G has highway dimension h, then $\forall\ r\ \exists$ 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.

HD vs. SPC

Theorem

If G has highway dimension h, then $\forall\ r\ \exists$ 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?

HD vs. SPC

Theorem

If G has highway dimension h, then $\forall\ r\ \exists$ an (r, h)-SPC.

Proof idea: Show that S^*, the smallest set that covers all shortest paths $P:\ r < |P| \le 2r$, 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^*.

HD vs. SPC

Theorem

If G has highway dimension h, then $\forall\ r\ \exists$ 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

Preprocessing algorithm

- Let $S_0 = V$. For $1 \le i \le \log D$ build $(2^i, k)$-SPC covers S_i.
- Let $L_i = S_i - \bigcup_{i+1}^{\log D} S_j$ (vertex partitioning into 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^+.

Generic Preprocessing

Preprocessing algorithm

- Let $S_0 = V$. For $1 \le i \le \log D$ build $(2^i, k)$-SPC covers S_i.
- Let $L_i = S_i - \bigcup_{i+1}^{\log D} S_j$ (vertex partitioning into 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^+.

Running time

Preprocessing is exponential ($k = h$) or polynomial ($k = O(h \log n)$).

Preprocessing Space

Lemma

For $v \in L_i$, $j \geq i$, 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.

Preprocessing Space

Lemma

For $v \in L_i$, $j \geq i$, 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 $O(k \log D)$ and $|E \cup E^+| = O(nk \log D)$

Preprocessing Space

Lemma

For $v \in L_i$, $j \geq i$, 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 $O(k \log D)$ and $|E \cup E^+| = O(nk \log D)$

Things are better than the worst case in practice.

RE Preprocessing

Remarks

- Reach bounds are in the graph with shortcuts.
- Break ties based on hop count (prefer shortcuts).

Lemma

If $v \in L_i$ *then* $\text{reach}(v) \leq 2 \cdot 2^i$.

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.

Query Time Bounds

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.

Query Time Bounds

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.

Remarks

- 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.

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, dist), doubling dim. $\log \alpha$, diameter D.
- Speedup parameter $\delta \in (0, 1)$.
- Edge $\{v, w\}$ has length $\text{dist}(v, w)$ and transit time $\tau(v, w) = \text{dist}^{1-\delta}(v, w)$. (Long roads are fast.)
- On-line network formation, adversarial 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 $\alpha^{O(1)}$.

Final Remarks

Summary

- Intuitive and practical routing algorithms.
- Efficient implementations.
- Used in practice.
- Theoretical understanding and justification.
- Further research, e.g., improved bounds or algorithms for other problems assuming small HD (TSP, vehicle routing, etc.).
- Static problem solved, dynamic – active research area.
  - Real time traffic.
  - Historical data.
  - Combination (prediction).

Thank You!

SPA (Shortest Path Algorithms) project page

http://research.microsoft.com/en-us/projects/SPA/

Questions?

1. Introduction and Motivation
2. Definitions and Model
3. Classical Algorithms
4. Recent Algorithms
   - Reach
   - Contraction Hierarchies
   - Transit Nodes
5. Theoretical Results
   - Highway Dimension
   - Theoretical Bounds
6. Final Remarks