

Highway and VC Dimensions: from Practice to Theory and Back

Ittai Abraham Daniel Delling Amos Fiat
Andrew V. Goldberg¹ Renato F. Werneck

Microsoft Research – Silicon Valley

<http://research.microsoft.com/~goldberg/>

Scientific Method

- “Science is divided into natural and unnatural.”
Vladimir Steklov, *mathematician*

Scientific Method

- “Science is divided into natural and unnatural.”
Vladimir Steklov, *mathematician*
- “No Sir, into social and anti-social.”
Sergei Platonov, *historian*

Scientific Method

- “Science is divided into natural and unnatural.”
Vladimir Steklov, *mathematician*
- “No Sir, into social and anti-social.”
Sergei Platonov, *historian*

Scientific Method: Basis for Natural Sciences

- Make an observation.
- Form a hypothesis or theory.
- Make a prediction.
- Verify by experiment.

Scientific Method

- “Science is divided into natural and unnatural.”
Vladimir Steklov, *mathematician*
- “No Sir, into social and anti-social.”
Sergei Platonov, *historian*

Scientific Method: Basis for Natural Sciences

- Make an observation.
- Form a hypothesis or theory.
- Make a prediction.
- Verify by experiment.

Mathematics of algorithms vs. algorithm engineering vs.
science of algorithmics: algorithm research via the scientific method.

Outline

- 1 Motivation
- 2 Contraction Hierarchies
- 3 Transit Node Algorithm
- 4 VC-dimension and Shortest Paths
- 5 Highway Dimension and Shortest Path Covers (SPCs)
- 6 Computing SPCs
- 7 Labeling Algorithm

Shortest Paths: Recent Developments

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

Dijkstra's algorithm: ≈ 5 s

Shortest Paths: Recent Developments

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

Dijkstra's algorithm: ≈ 5 s

Recent work

- Arc flags [Lauther 04, Köhler et al. 06, Bauer & Delling 08].
- 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).

Shortest Paths: Recent Developments

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

Dijkstra's algorithm: ≈ 5 s

Recent work

- Arc flags [Lauther 04, Köhler et al. 06, Bauer & Delling 08].
- 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: $\ll 1$ ms, ≈ 0.1 s on a mobile device.
Only a few hundred or less intersections searched.

Definitions and Model

Input

- Graph $G = (V, E)$ with **unique** shortest paths.
- $|V| = n$, $|E| = m$.
- For this talk, assume G is undirected.
- Weight function ℓ (length, transit time, fuel consumption, ...).
- **Static problem, G and ℓ incorporate all modeling information.**

Definitions and Model

Input

- Graph $G = (V, E)$ with **unique** shortest paths.
- $|V| = n$, $|E| = m$.
- For this talk, assume G is undirected.
- 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 $G = (V, E)$ with **unique** shortest paths.
- $|V| = n$, $|E| = m$.
- For this talk, assume G is undirected.
- 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.**

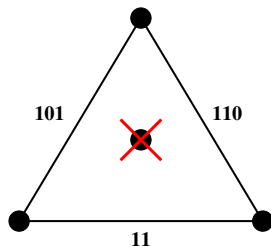
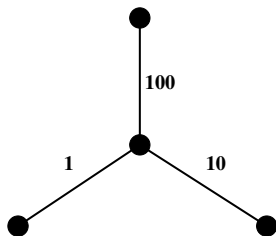
Algorithms with preprocessing

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

Shortcut Operation

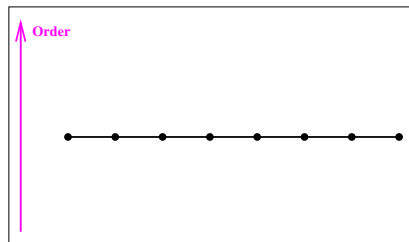
[Sanders & Schultes 05]

The key operation for Contraction Hierarchies algorithm



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

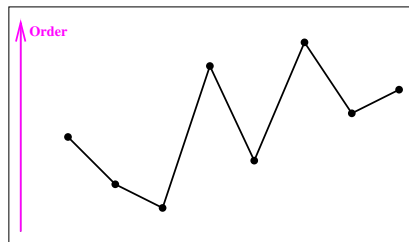
Contraction Hierarchies



[Geisberger et al. 08]

Preprocessing orders vertices, order corresponds to importance. Both **forward** and **reverse** searches consider only “up” (more local to more global) edges. **Effective pruning**.

Contraction Hierarchies



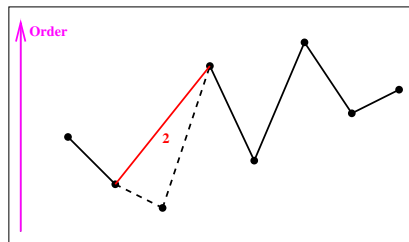
[Geisberger et al. 08]

Preprocessing orders vertices, order corresponds to importance. Both **forward** and **reverse** searches consider only “up” (more local to more global) edges. **Effective pruning**.

Preprocessing algorithm

- 1 Heuristically order vertices.

Contraction Hierarchies



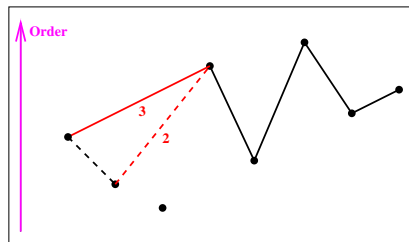
[Geisberger et al. 08]

Preprocessing orders vertices, order corresponds to importance. 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



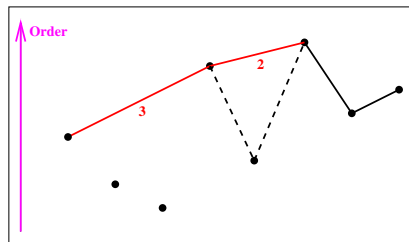
[Geisberger et al. 08]

Preprocessing orders vertices, order corresponds to importance. 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



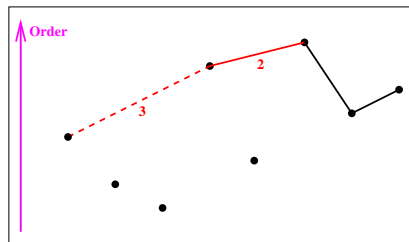
[Geisberger et al. 08]

Preprocessing orders vertices, order corresponds to importance. 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



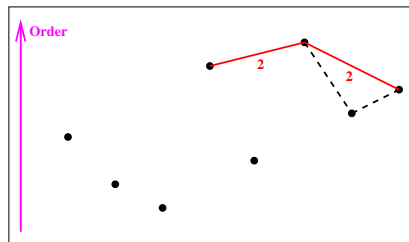
[Geisberger et al. 08]

Preprocessing orders vertices, order corresponds to importance. 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



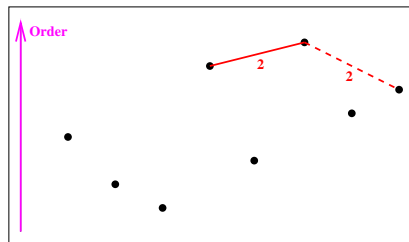
[Geisberger et al. 08]

Preprocessing orders vertices, order corresponds to importance. 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



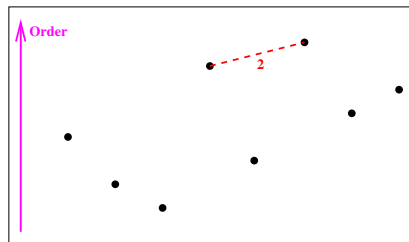
[Geisberger et al. 08]

Preprocessing orders vertices, order corresponds to importance. 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



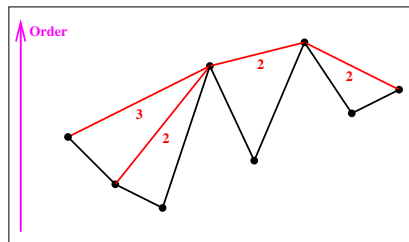
[Geisberger et al. 08]

Preprocessing orders vertices, order corresponds to importance. 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



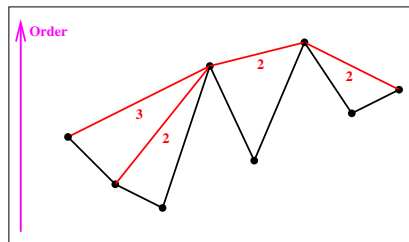
[Geisberger et al. 08]

Preprocessing orders vertices, order corresponds to importance. 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 importance. Both **forward** and **reverse** searches consider only “up” (more local to more global) edges. **Effective pruning**.

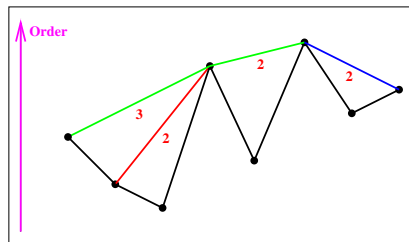
Preprocessing algorithm

- 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.

Contraction Hierarchies



[Geisberger et al. 08]

Preprocessing orders vertices, order corresponds to importance. Both **forward** and **reverse** searches consider only “up” (more local to more global) edges. **Effective pruning**.

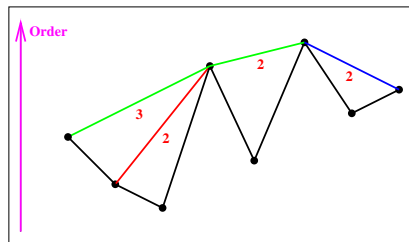
Preprocessing algorithm

- 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.

Contraction Hierarchies



[Geisberger et al. 08]

Preprocessing orders vertices, order corresponds to importance. 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.

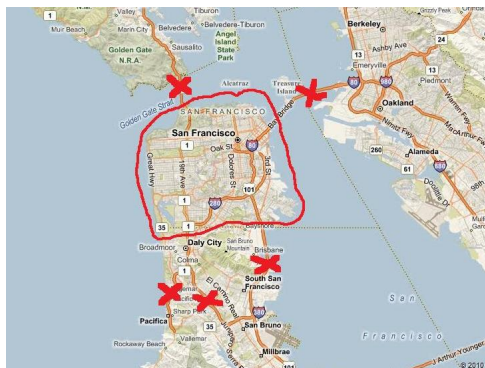
Query algorithm

- Run a modified bidirectional Dijkstra’s algorithm.
- The searches only consider “up” edges.

Correctness: Highest-rank SP vertex gets correct distance label.

Transit Node (TN) Algorithm

[Bast et al. 06]



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

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

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 ($\approx 10\,000$).

Empirical observation: small number of access/transit nodes.

TN Preprocessing

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 ($\approx 10\,000$).

Empirical observation: small number of access/transit nodes.

Preprocessing algorithm

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

Long-range query algorithm

- The shortest path has the form
 $s - \text{access}(s) - \text{access}(t) - t$

Long-range query algorithm

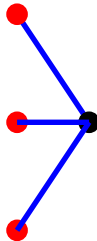
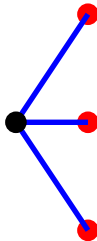
- The shortest path has the form
 $s - \text{access}(s) - \text{access}(t) - t$



TN Query

Long-range query algorithm

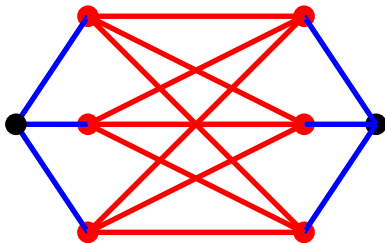
- The shortest path has the form
 $s - \text{access}(s) - \text{access}(t) - t$



TN Query

Long-range query algorithm

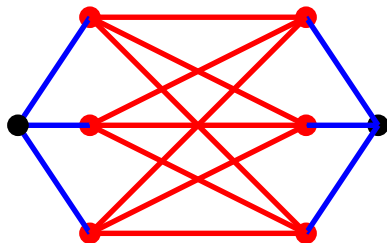
- The shortest path has the form
 $s - \text{access}(s) - \text{access}(t) - t$



TN Query

Long-range query algorithm

- The shortest path has the form $s - \text{access}(s) - \text{access}(t) - t$
- Table look-up for the $(\text{access}(s), \text{access}(t))$ node pairs.



Remarks

- Very fast: 10×10 table look-ups per long-range query.
- Local queries: another method (e.g., CH) 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 et al. 10]

- Define **highway dimension (HD)**.
- Good time bounds for transit nodes, highway hierarchies, and reach algorithms assuming HD is small.
- Analysis highlights algorithm similarities.
- Generative model of small HD networks (road network formation).

VC-Dimension [Vapnik & Chervonenkis 71]

- **Base set** X , collection of subsets \mathcal{R} , **set system** (X, \mathcal{R}) .
- For $Y \subseteq X$, $Y|_{\mathcal{R}} = (Y, \{R \cap Y | R \in \mathcal{R}\})$.
- \mathcal{R} **shatters** Y if $Y|_{\mathcal{R}} = 2^Y$.
- (X, \mathcal{R}) has **VC-dimension** d if d is the smallest integer such that no $d + 1$ subset of X can be shattered.

VC-Dimension [Vapnik & Chervonenkis 71]

- **Base set** X , collection of subsets \mathcal{R} , **set system** (X, \mathcal{R}) .
- For $Y \subseteq X$, $Y|_{\mathcal{R}} = (Y, \{R \cap Y | R \in \mathcal{R}\})$.
- \mathcal{R} **shatters** Y if $Y|_{\mathcal{R}} = 2^Y$.
- (X, \mathcal{R}) has **VC-dimension** d if d is the smallest integer such that no $d + 1$ subset of X can be shattered.

Example (points and half-planes)

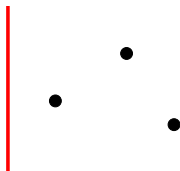
X is a plane, \mathcal{R} is the set of all half-planes, VC-dimension is 3.

VC-Dimension [Vapnik & Chervonenkis 71]

- **Base set** X , collection of subsets \mathcal{R} , **set system** (X, \mathcal{R}) .
- For $Y \subseteq X$, $Y|_{\mathcal{R}} = (Y, \{R \cap Y | R \in \mathcal{R}\})$.
- \mathcal{R} **shatters** Y if $Y|_{\mathcal{R}} = 2^Y$.
- (X, \mathcal{R}) has **VC-dimension** d if d is the smallest integer such that no $d + 1$ subset of X can be shattered.

Example (points and half-planes)

X is a plane, \mathcal{R} is the set of all half-planes, VC-dimension is 3.

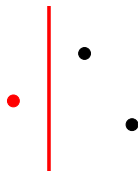


VC-Dimension [Vapnik & Chervonenkis 71]

- **Base set** X , collection of subsets \mathcal{R} , **set system** (X, \mathcal{R}) .
- For $Y \subseteq X$, $Y|_{\mathcal{R}} = (Y, \{R \cap Y | R \in \mathcal{R}\})$.
- \mathcal{R} **shatters** Y if $Y|_{\mathcal{R}} = 2^Y$.
- (X, \mathcal{R}) has **VC-dimension** d if d is the smallest integer such that no $d + 1$ subset of X can be shattered.

Example (points and half-planes)

X is a plane, \mathcal{R} is the set of all half-planes, VC-dimension is 3.

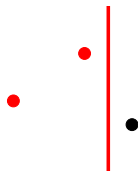


VC-Dimension [Vapnik & Chervonenkis 71]

- **Base set** X , collection of subsets \mathcal{R} , **set system** (X, \mathcal{R}) .
- For $Y \subseteq X$, $Y|_{\mathcal{R}} = (Y, \{R \cap Y | R \in \mathcal{R}\})$.
- \mathcal{R} **shatters** Y if $Y|_{\mathcal{R}} = 2^Y$.
- (X, \mathcal{R}) has **VC-dimension** d if d is the smallest integer such that no $d + 1$ subset of X can be shattered.

Example (points and half-planes)

X is a plane, \mathcal{R} is the set of all half-planes, VC-dimension is 3.

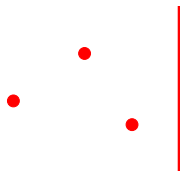


VC-Dimension [Vapnik & Chervonenkis 71]

- **Base set** X , collection of subsets \mathcal{R} , **set system** (X, \mathcal{R}) .
- For $Y \subseteq X$, $Y|_{\mathcal{R}} = (Y, \{R \cap Y | R \in \mathcal{R}\})$.
- \mathcal{R} **shatters** Y if $Y|_{\mathcal{R}} = 2^Y$.
- (X, \mathcal{R}) has **VC-dimension** d if d is the smallest integer such that no $d + 1$ subset of X can be shattered.

Example (points and half-planes)

X is a plane, \mathcal{R} is the set of all half-planes, VC-dimension is 3.

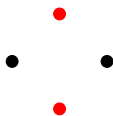


VC-Dimension [Vapnik & Chervonenkis 71]

- **Base set** X , collection of subsets \mathcal{R} , **set system** (X, \mathcal{R}) .
- For $Y \subseteq X$, $Y|_{\mathcal{R}} = (Y, \{R \cap Y | R \in \mathcal{R}\})$.
- \mathcal{R} **shatters** Y if $Y|_{\mathcal{R}} = 2^Y$.
- (X, \mathcal{R}) has **VC-dimension** d if d is the smallest integer such that no $d + 1$ subset of X can be shattered.

Example (points and half-planes)

X is a plane, \mathcal{R} is the set of all half-planes, VC-dimension is 3.



USP Systems

Definition

Unique Shortest Path (USP) Systems

- G is a graph with unique shortest paths, $X = V$.
- \mathcal{R} contains (some) sets of vertices on shortest paths.

USP Systems

Definition

Unique Shortest Path (USP) Systems

- G is a graph with unique shortest paths, $X = V$.
- \mathcal{R} contains (some) sets of vertices on shortest paths.

Strange question: VC-dimension of a USP system?

USP Systems

Definition

Unique Shortest Path (USP) Systems

- G is a graph with unique shortest paths, $X = V$.
- \mathcal{R} contains (some) sets of vertices on shortest paths.

Strange question: VC-dimension of a USP system?

Theorem

VC-dimension of a USP system is at most two.

USP Systems

Definition

Unique Shortest Path (USP) Systems

- G is a graph with unique shortest paths, $X = V$.
- \mathcal{R} contains (some) sets of vertices on shortest paths.

Strange question: VC-dimension of a USP system?

Theorem

VC-dimension of a USP system is at most two.

Proof:

- Three points not on a shortest path cannot be shattered.
- Three points on a (unique) shortest path cannot be shattered.



More Definitions

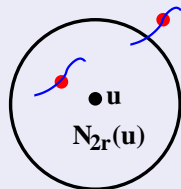
- Constant maximum degree.
- $\ell(P)$ denotes the length of P .
- Network diameter D .
- P_q : all SP $P : \ell(P) \leq q$.
- P_q^r : all SP $P : r < \ell(P) \leq q$.
- **r -neighborhood** $N_r(v) = \bigcup(P \in P_r : v \in P)$ (almost a ball).
- **r -neighborhood system** $S_r(v) = \{P \in P_{2r}^r : P \cap N_r(v) \neq \emptyset\}$.
- **Hitting set** for (X, \mathcal{R}) : $H \subseteq X : \forall R \in \mathcal{R}, H \cap R \neq \emptyset$.

Highway Dimension Definition

Highway dimension (HD) h

Locally, a small set of vertices hits all long SPs.

$\forall r \in \mathbb{R}, \forall u \in V, \exists$ a hitting set H for $(V, S_{2r}(u))$
such that $|H| \leq h$.

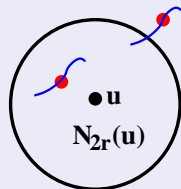


Highway Dimension Definition

Highway dimension (HD) h

Locally, a small set of vertices hits all long SPs.

$\forall r \in \mathbb{R}, \forall u \in V, \exists$ a hitting set H for $(V, S_{2r}(u))$
such that $|H| \leq h$.

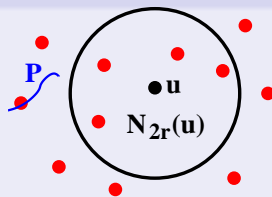


(r, k) Shortest path cover $((r, k)$ -SPC):

All SPs in P_{2r}^r can be hit by a sparse set.

A set C such that

- 1 C is a hitting set for P_{2r}^r .
- 2 $\forall u \in V, |C \cap N_{2r}(u)| \leq k$.



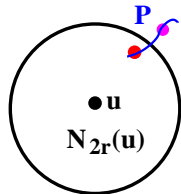
HD vs. SPC

Theorem

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

Proof:

- Show S^* , the smallest set hitting for P_{2r}^r , is an (r, h) -SPC.
- Let $S'(v) = S^* \cap N_{2r}(v)$.
- Suppose $|S'(v)| > h$.
- $S'(v)$ hits only paths in $S_{2r}(v)$.
- Let H be the hitting set of S_{2r} such that $|H| \leq h$.
- Replacing $S'(v)$ by H gives a smaller set S^* .



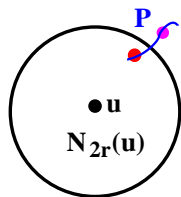
HD vs. SPC

Theorem

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

Proof:

- Show S^* , the smallest set hitting for P_{2r}^r , is an (r, h) -SPC.
- Let $S'(v) = S^* \cap N_{2r}(v)$.
- Suppose $|S'(v)| > h$.
- $S'(v)$ hits only paths in $S_{2r}(v)$.
- Let H be the hitting set of S_{2r} such that $|H| \leq h$.
- Replacing $S'(v)$ by H gives a smaller set S^* .



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

Computing Approximate SPCs

- Greedy gives an $O(\log n)$ approximation [Abraham et al. 10].
- Approximation independent of n ?

Computing Approximate SPCs

- Greedy gives an $O(\log n)$ approximation [Abraham et al. 10].
- Approximation independent of n ?

Theorem

Suppose we have a poly-time, $(c \log h)$ -approximation algorithm for hitting set. If G has HD h , then for any r we can construct, in polynomial time, an $(r, O(h \log h))$ -SPC.

Proof: Similar to the previous proof. Maintain a hitting set C . If for some v , $|C \cap N_{2r}| > ch \log h$, compute a hitting set for (V, S_{2r}) of size at most $ch \log h$ and get a smaller hitting set C .

Hitting Sets in Low VC-dimension

[Clarkson 88, Brönnimann & Goodrich 95, Even et al. 95]

Theorem

For a set system with optimal hitting set of size h and VC-dimension d , an $O(hd \log(hd))$ hitting set is poly-time computable.

Corollary

For an HD h graph, we can efficiently compute an $O(h \log h)$ -size hitting set for $(V, S_{2r}(v))$.

Hitting Sets in Low VC-dimension

[Clarkson 88, Brönnimann & Goodrich 95, Even et al. 95]

Theorem

For a set system with optimal hitting set of size h and VC-dimension d , an $O(hd \log(hd))$ hitting set is poly-time computable.

Corollary

For an HD h graph, we can efficiently compute an $O(h \log h)$ -size hitting set for $(V, S_{2r}(v))$.

Combining with [Abraham et al. 10] we get:

Theorem

With poly-time preprocessing, CH queries take $O((h \log h \log D)^2)$ time.

Hitting Set Approximation Algorithm

Clarkson's Algorithm Outline

- 1 Start with all vertices having weight one.
- 2 Pick a random weighted set C of size $(ch \log h)$.
- 3 If C is a hitting set, halt.
- 4 Find an SP $P \in S_{2r}$ that is not covered, double vertex weights on P .
- 5 Goto 2.

Hitting Set Approximation Algorithm

Clarkson's Algorithm Outline

- 1 Start with all vertices having weight one.
- 2 Pick a random weighted set C of size $(ch \log h)$.
- 3 If C is a hitting set, halt.
- 4 Find an SP $P \in S_{2r}$ that is not covered, double vertex weights on P .
- 5 Goto 2.

Remarks

- Not the first algorithm for SPC one would think of.
- Currently, our best SPC algorithm uses these ideas.
- [Even et al. 95] use LP relaxation.

Labeling Algorithm

[Gavoille et al. 04, Thorup & Zwick 05]

Generic preprocessing

- $\forall v \in V$ precompute a **label** $L(v) \subseteq V$.
- $\forall w \in L(v)$ precompute distances $\text{dist}(v, w)$

Label property: $\forall s, t$, a shortest s - t path intersects $L(s) \cap L(t)$.

Labeling Algorithm

[Gavoille et al. 04, Thorup & Zwick 05]

Generic preprocessing

- $\forall v \in V$ precompute a **label** $L(v) \subseteq V$.
- $\forall w \in L(v)$ precompute distances $\text{dist}(v, w)$

Label property: $\forall s, t$, a shortest s - t path intersects $L(s) \cap L(t)$.

Query algorithm

Find and return $\min_{u \in L(s) \cap L(t)} (\text{dist}(s, u) + \text{dist}(u, t))$.

Labeling Algorithm

[Gavoille et al. 04, Thorup & Zwick 05]

Generic preprocessing

- $\forall v \in V$ precompute a **label** $L(v) \subseteq V$.
- $\forall w \in L(v)$ precompute distances $\text{dist}(v, w)$

Label property: $\forall s, t$, a shortest s - t path intersects $L(s) \cap L(t)$.

Query algorithm

Find and return $\min_{u \in L(s) \cap L(t)} (\text{dist}(s, u) + \text{dist}(u, t))$.

Remarks

- Very simple query, efficient if the labels are small.

Labeling for Low HD

Low HD preprocessing

- For $i = 0, 1, \dots, \log D$ efficiently compute $(2^i, O(h \log h))$ -SPC C_i .
- $\forall v \in V$ set $L(v) = \bigcup_i (C_i \cap N_{2 \cdot 2^i}(v))$.

Lemma: $\forall v \in V, |L(v)| = O(h \log h \log D)$.

Correctness

- Let P be the shortest s - t path and $2^i < \ell(P) \leq 2^{i+1}$.
- $P \subseteq N_{2 \cdot 2^i}(s)$ and $P \subseteq N_{2 \cdot 2^i}(t)$.
- $\exists u \in P \cap C_i$, for such u we have $u \in L(s) \cap L(t)$.

Theorem

Labeling algorithm preprocessing is poly-time and queries take $O(h \log h \log D)$ time.

Practical Label Computation

- Theory predicts that small labels exist.
- Theoretically good algorithm too slow for large road networks.

Practical Label Computation

- Theory predicts that small labels exist.
- Theoretically good algorithm too slow for large road networks.

CH labels

- Vertices visited during a CH search from v form a valid label.
- ≈ 500 vertices visited on Western Europe.
- Labels too big but a good starting point.

Practical Label Computation

- Theory predicts that small labels exist.
- Theoretically good algorithm too slow for large road networks.

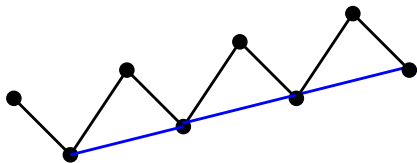
CH labels

- Vertices visited during a CH search from v form a valid label.
- ≈ 500 vertices visited on Western Europe.
- Labels too big but a good starting point.

Correctness: Both $L(s)$ and $L(t)$ contain the vertex u on the shortest s - t path with the highest CH rank.

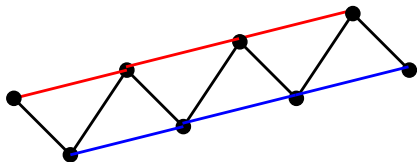
The common vertex u has **correct label**.

Smaller Labels



length 1
length 3

Smaller Labels

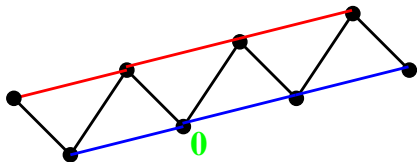


length 1

length 3

length 2

Smaller Labels

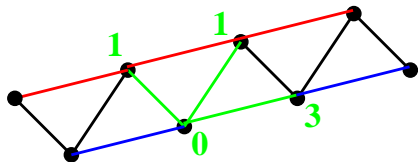


length 1

length 3

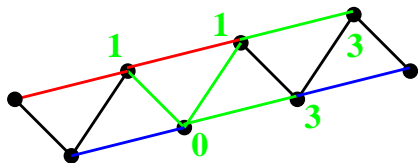
length 2

Smaller Labels



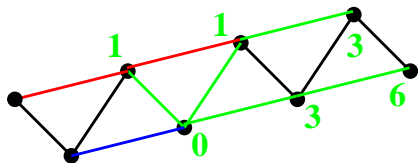
length 1
length 3
length 2

Smaller Labels



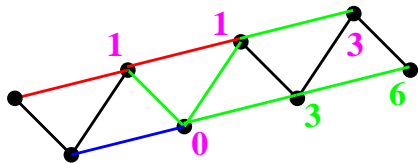
length 1
length 3
length 2

Smaller Labels



length 1
length 3
length 2

Smaller Labels



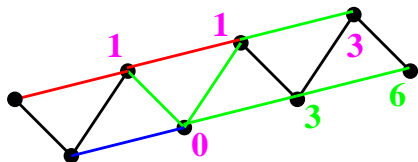
length 1

length 3

length 2

correct / incorrect d values

Smaller Labels



length 1

length 3

length 2

correct / incorrect d values

Improvements

- Prune vertices with incorrect d values.
- Use SPCs to improve CH ordering (theory helps again!).
- Western Europe label size reduces to 80's, becomes practical.
- Labels are compressible (very similar for nearby vertices).
- Techniques to speed up preprocessing.
- Techniques to speed up queries.

Experimental Results

CH	CH&AF	TN&CH	TN&CH&AF	Labeling
93 995	9 034	1 775	992	276

*Time in nanoseconds on a 3.33 GHz Xeon X5680 (or scaled)
Western Europe, 18M vertices and 42M arcs, time metric*

- **Good theoretical bounds** currently known for CH and labeling.
- **Arc Flags (AF)** [Lauther 06] is a directional way to prune Dijkstra.
- TN&CH&AF is the fastest previous algorithm [Bauer et al. 08] for random queries.

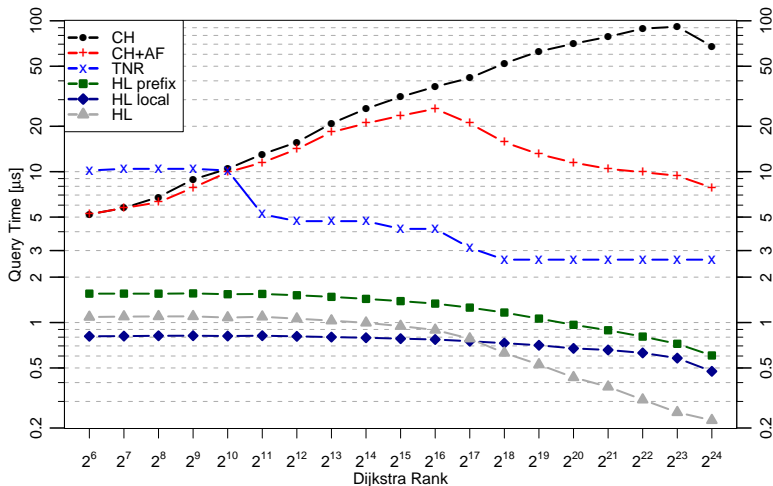
Experimental Results

CH	CH&AF	TN&CH	TN&CH&AF	Labeling
93 995	9 034	1 775	992	276

*Time in nanoseconds on a 3.33 GHz Xeon X5680 (or scaled)
Western Europe, 18M vertices and 42M arcs, time metric*

- Good theoretical bounds currently known for CH and labeling.
- Arc Flags (AF) [Lauther 06] is a directional way to prune Dijkstra.
- TN&CH&AF is the fastest previous algorithm [Bauer et al. 08] for random queries.
- Careful implementation needed (56 ns. cache line fetch).
- Labels can be compressed; space/time tradeoff.
- Labeling is practical.

Local Queries



Even higher speedup.

Concluding Remarks

Research Directions

- Complexity of computing SPCs in small HD networks (NP-hard?).
- Practical algorithms for SPCs for large road networks.
- Other applications of HD and SPCs.
- Dynamic and time-dependent shortest path algorithms.

Concluding Remarks

Research Directions

- Complexity of computing SPCs in small HD networks (NP-hard?).
- Practical algorithms for SPCs for large road networks.
- Other applications of HD and SPCs.
- Dynamic and time-dependent shortest path algorithms.

Science of Algorithmics

- Experimental observations: several algorithms are very fast, small number of transit nodes.
- Theory: highway dimension and sublinear query bounds.
- Prediction: the labeling algorithm is fast.
- Verification: practical implementation and experiments.

Thank You!

SPA (Shortest Path Algorithms) project page

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

Questions?

- 1 Motivation
- 2 Contraction Hierarchies
- 3 Transit Node Algorithm
- 4 VC-dimension and Shortest Paths
- 5 Highway Dimension and Shortest Path Covers (SPCs)
- 6 Computing SPCs
- 7 Labeling Algorithm