Universität Konstanz

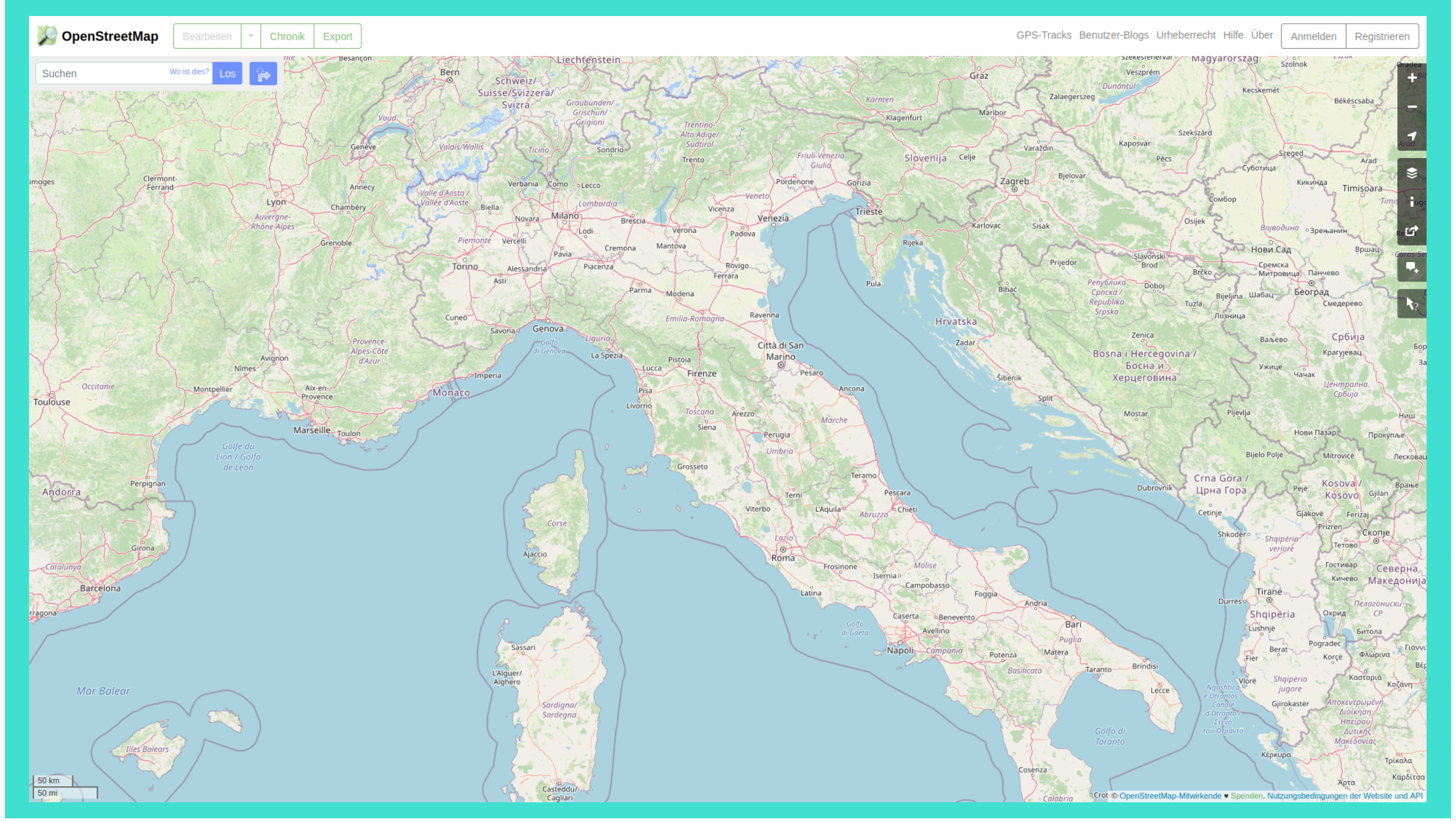
Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Local-Fréchet Polyline Simplification Has Subcubic Complexity in Two Dimensions

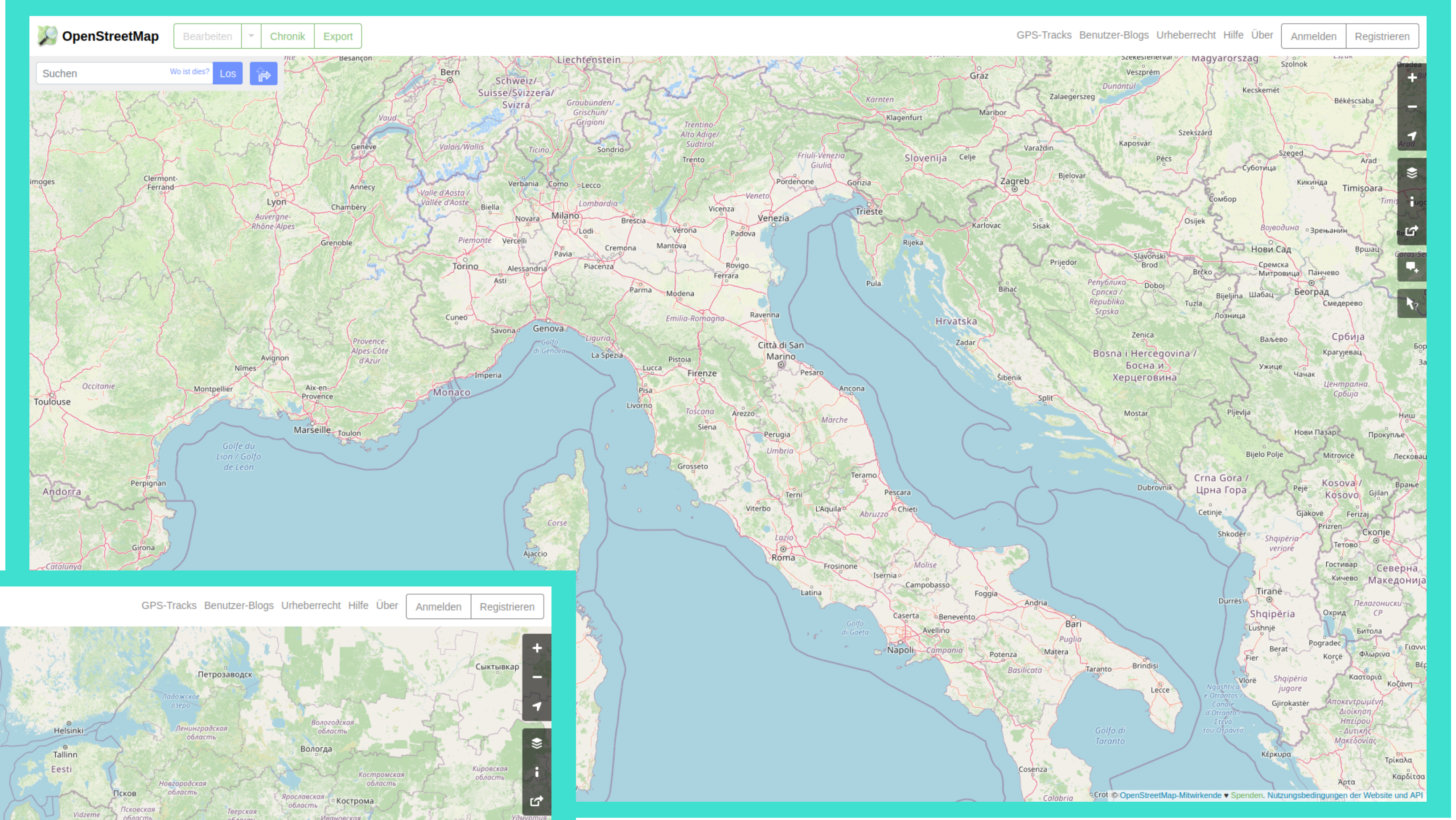
EuroCG 2022

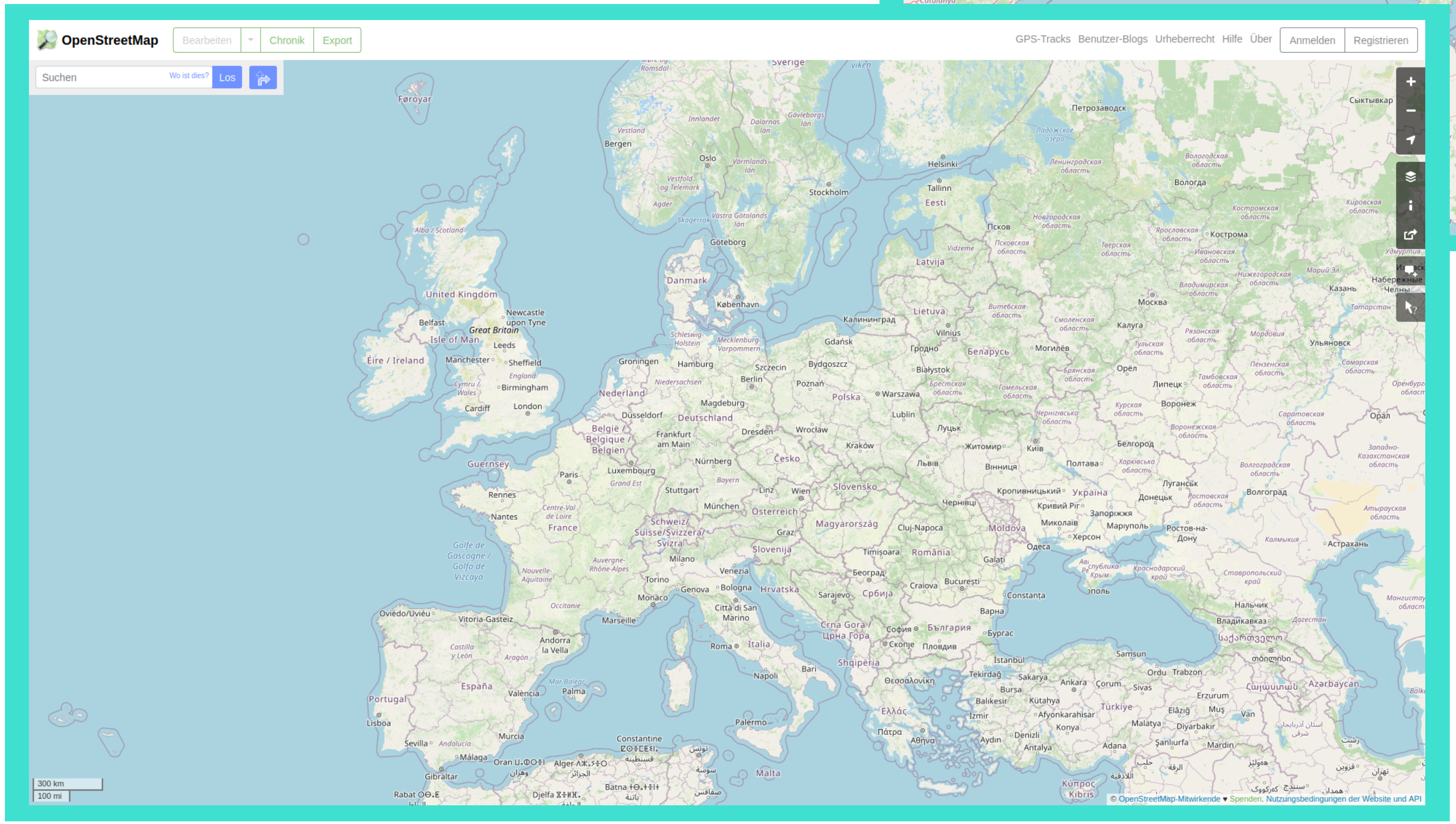
Sabine Storandt and Johannes Zink

Motivation

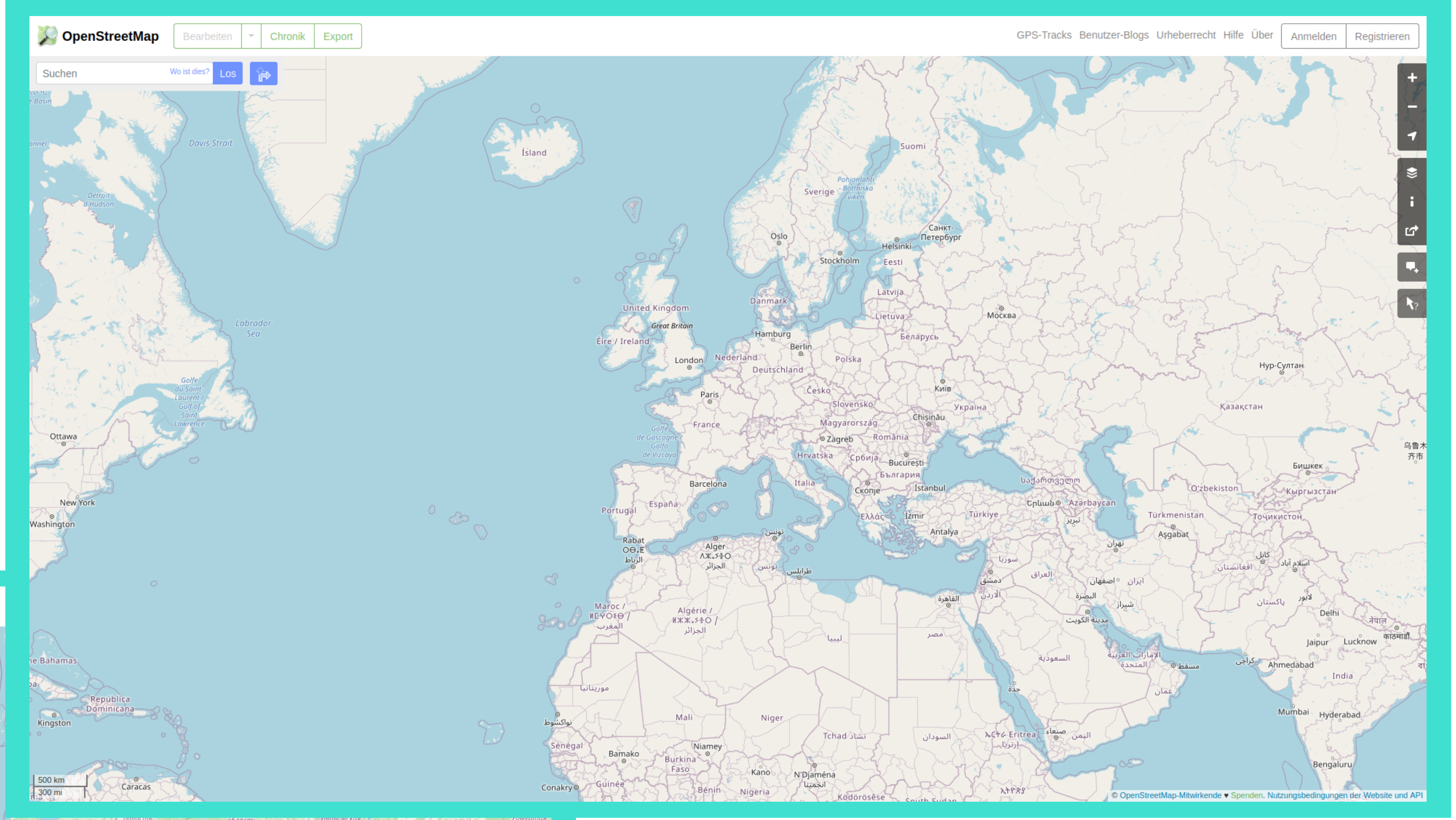

OpenStreetMap
Bearbeiten
Chronik
Export
GPS-Tracks
Benutzer-Blogs
Urheberrecht
Hilfe
Über
Anmelden
Registrieren
Suchen
Wo ist dies?
Los
50 km
50 mi

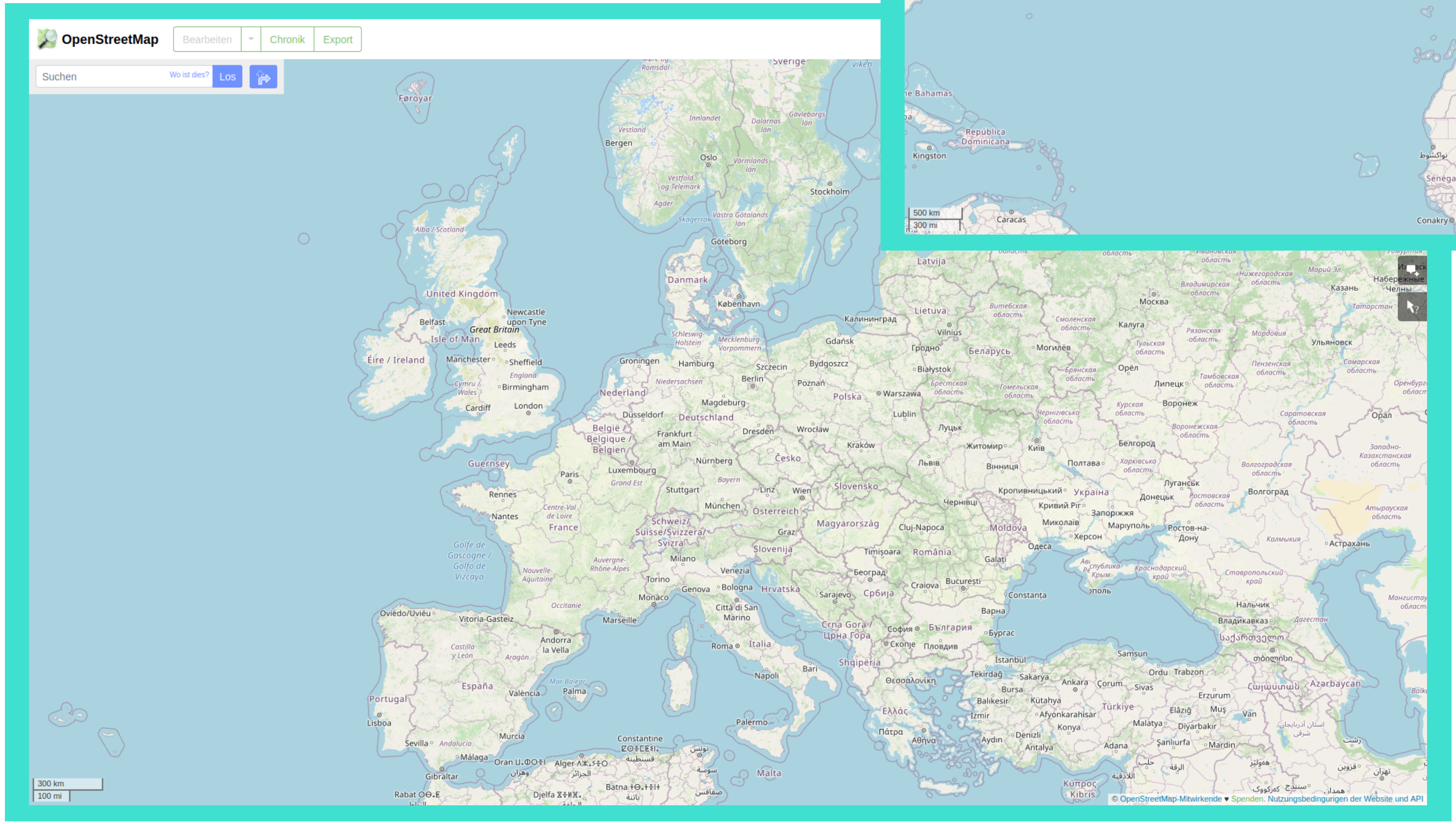
Motivation


OpenStreetMap
Bearbeiten
Chronik
Export
GPS-Tracks
Benutzer-Blogs
Urheberrecht
Hilfe
Über
Anmelden
Registrieren
Suchen
Wo ist dies?
Los
© OpenStreetMap-Mitwirkende ♥ Spenden. Nutzungsbedingungen der Website und API

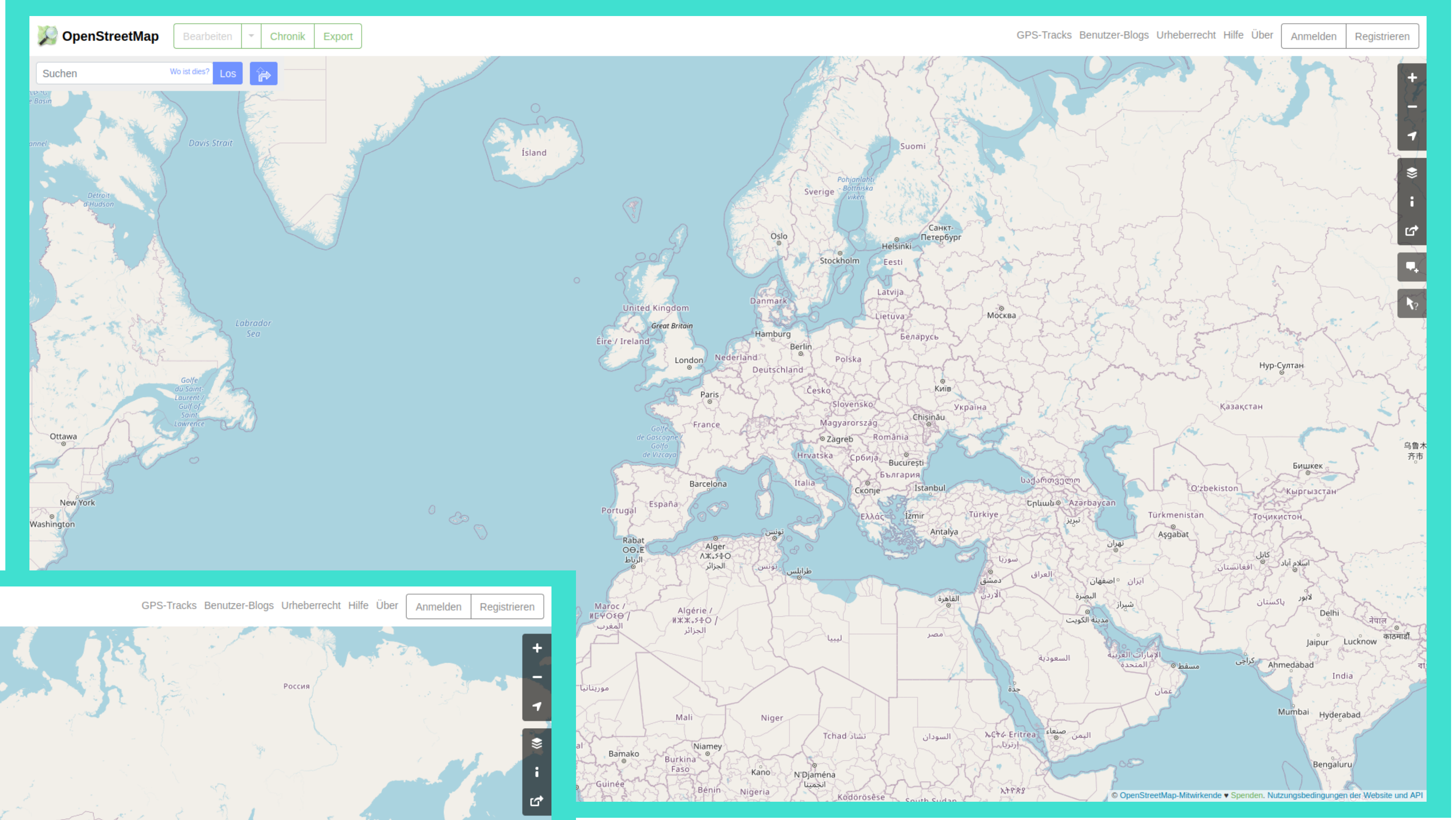

OpenStreetMap
Bearbeiten
Chronik
Export
GPS-Tracks
Benutzer-Blogs
Urheberrecht
Hilfe
Über
Anmelden
Registrieren
Suchen
Wo ist dies?
Los
300 km
100 mi
© OpenStreetMap-Mitwirkende ♥ Spenden. Nutzungsbedingungen der Website und API

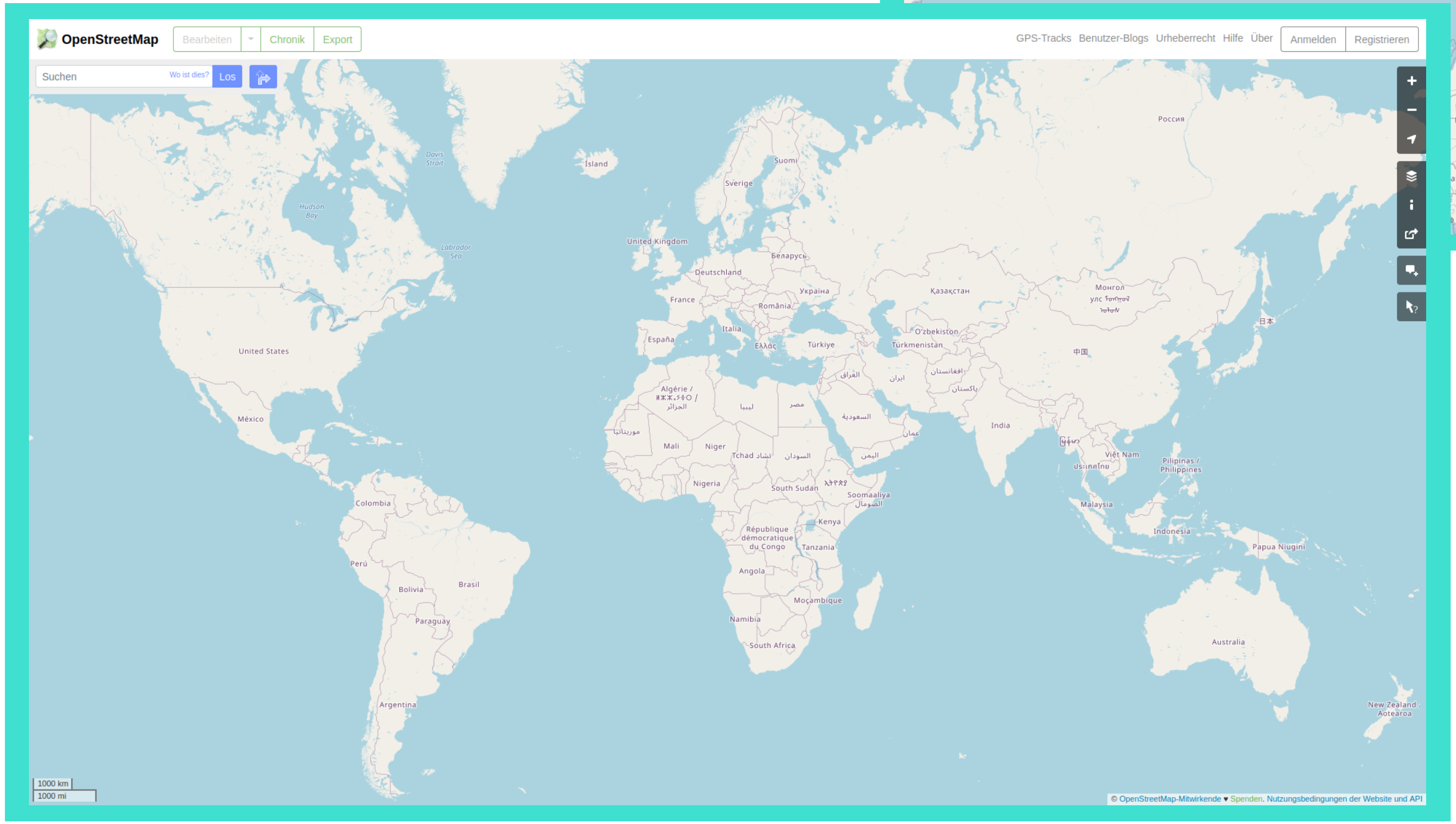
Motivation


OpenStreetMap
Bearbeiten
Chronik
Export
GPS-Tracks
Benutzer-Blogs
Urheberrecht
Hilfe
Über
Anmelden
Registrieren
Suchen
Wo ist dies?
Los
500 km
300 mi
© OpenStreetMap-Mitwirkende ♥ Spenden. Nutzungsbedingungen der Website und API


OpenStreetMap
Bearbeiten
Chronik
Export
Suchen
Wo ist dies?
Los
300 km
100 mi
© OpenStreetMap-Mitwirkende ♥ Spenden. Nutzungsbedingungen der Website und API

Motivation


OpenStreetMap
Bearbeiten
Chronik
Export
GPS-Tracks
Benutzer-Blogs
Urheberrecht
Hilfe
Über
Anmelden
Registrieren
Suchen
Wo ist dies?
Los


OpenStreetMap
Bearbeiten
Chronik
Export
GPS-Tracks
Benutzer-Blogs
Urheberrecht
Hilfe
Über
Anmelden
Registrieren
Suchen
Wo ist dies?
Los
1000 km
1000 mi

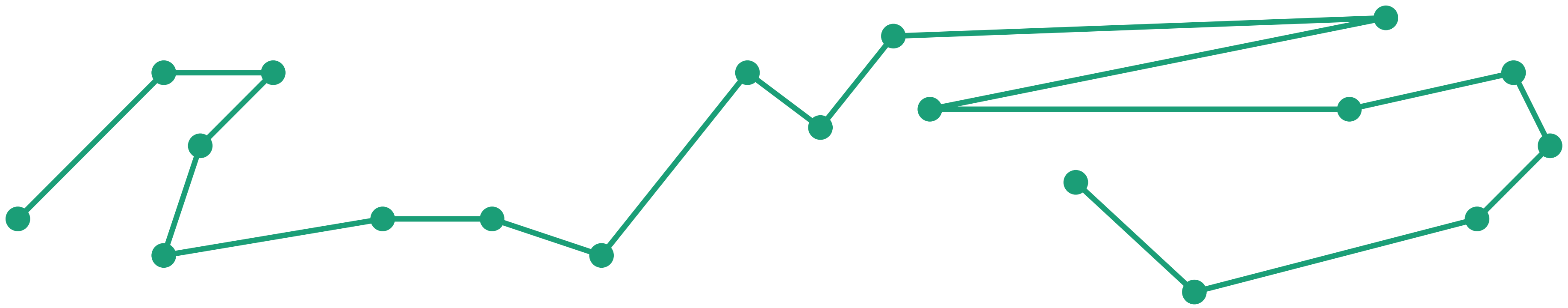
Definition: Polyline Simplification

Definition: Polyline Simplification

Given: ■ polyline P as a sequence of n points in the plane

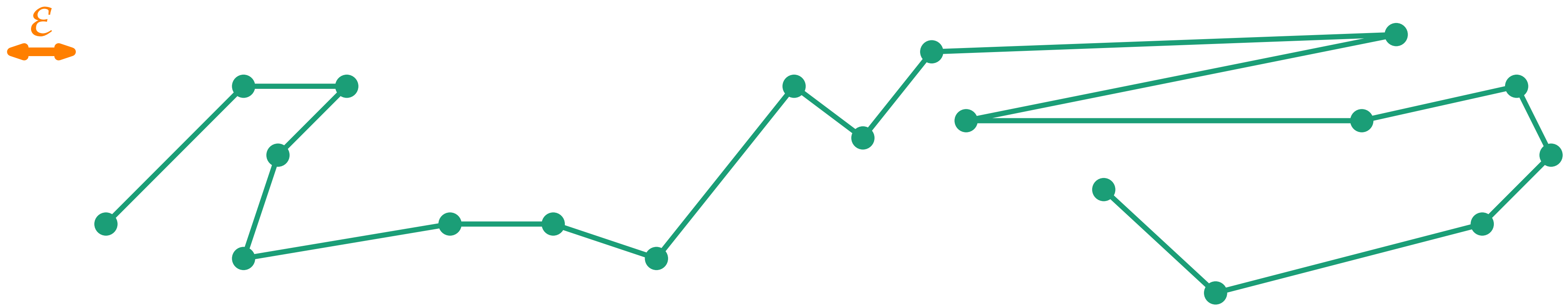
Definition: Polyline Simplification

Given: ■ polyline P as a sequence of n points in the plane



Definition: Polyline Simplification

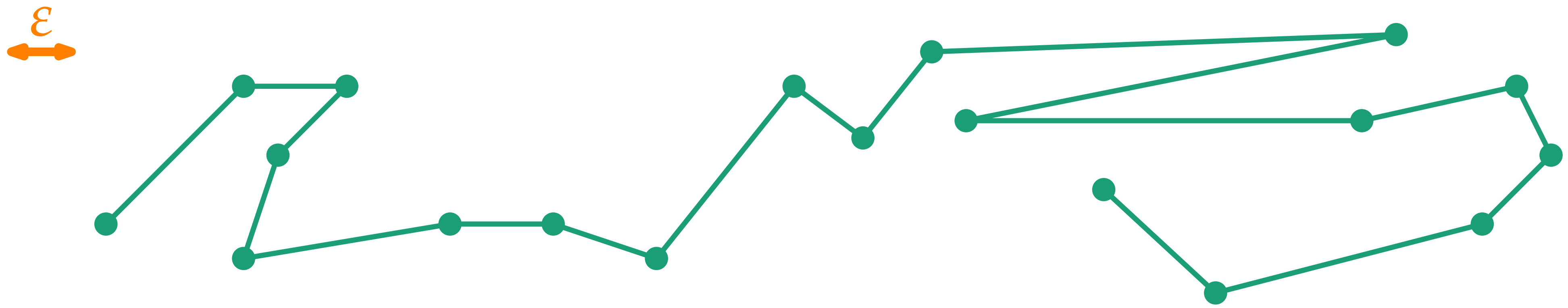
Given:
- polyline P as a sequence of n points in the plane
- distance threshold ε



Definition: Polyline Simplification

Given:
- polyline P as a sequence of n points in the plane
- distance threshold ε

Find: Min. size subsequence P' of P, s.t. $d_X(P', P) \leq \varepsilon$ (keep start and end point).

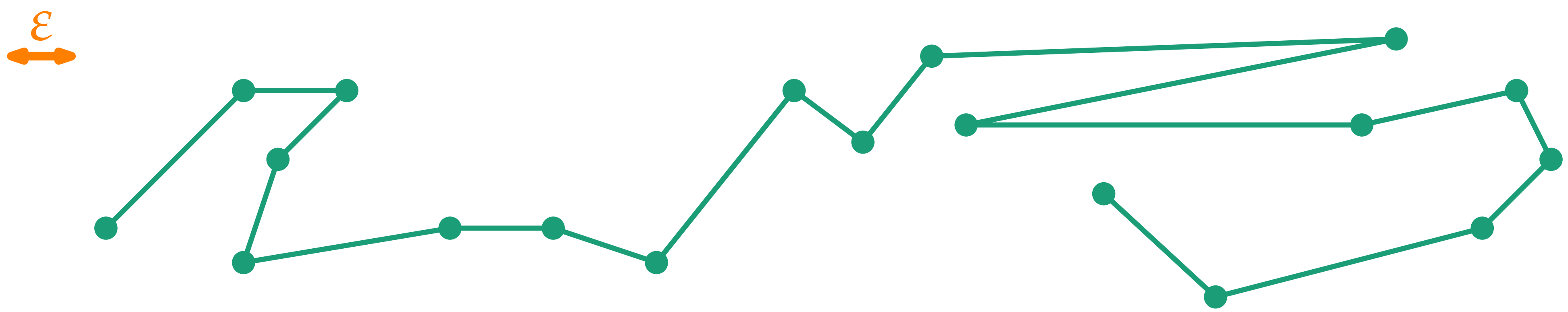


Definition: Polyline Simplification

distance measure for comparing two curves

Given:
- polyline P as a sequence of n points in the plane
- distance threshold ε

Find: Min. size subsequence P' of P, s.t. $d_X(P', P) \leq \varepsilon$ (keep start and end point).

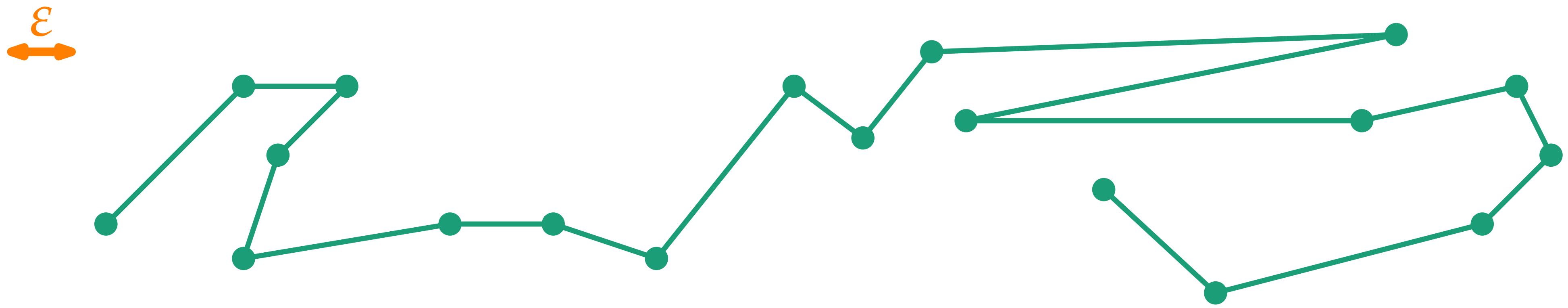


Definition: Polyline Simplification

Given:
- polyline P as a sequence of n points in the plane
- distance threshold ε

distance measure for comparing two curves

Find: Min. size subsequence P' of P, s.t. $d_X(P', P) \leq \varepsilon$ (keep start and end point).



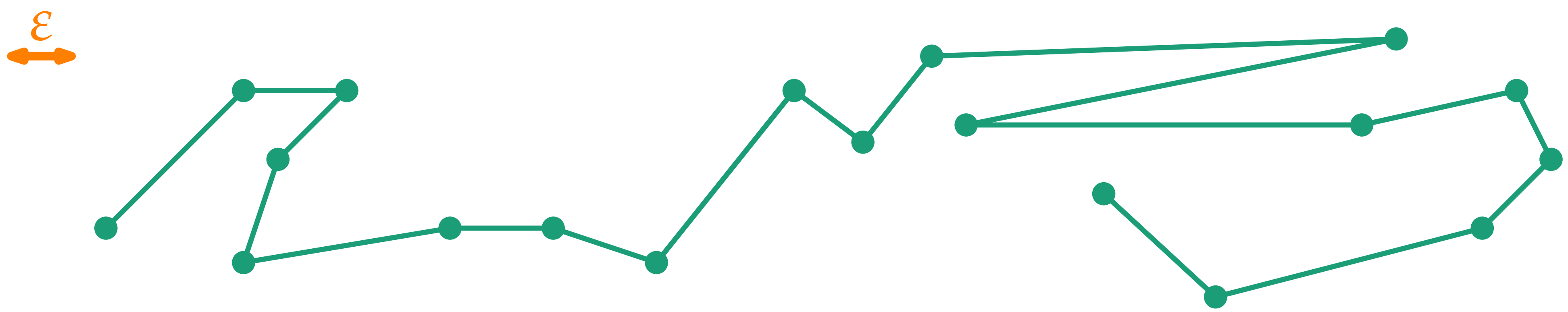
Typical distance measures d_X:

Definition: Polyline Simplification

distance measure for comparing two curves

Given:
- polyline P as a sequence of n points in the plane
- distance threshold ε

Find: Min. size subsequence P' of P, s.t. $d_X(P', P) \leq \varepsilon$ (keep start and end point).



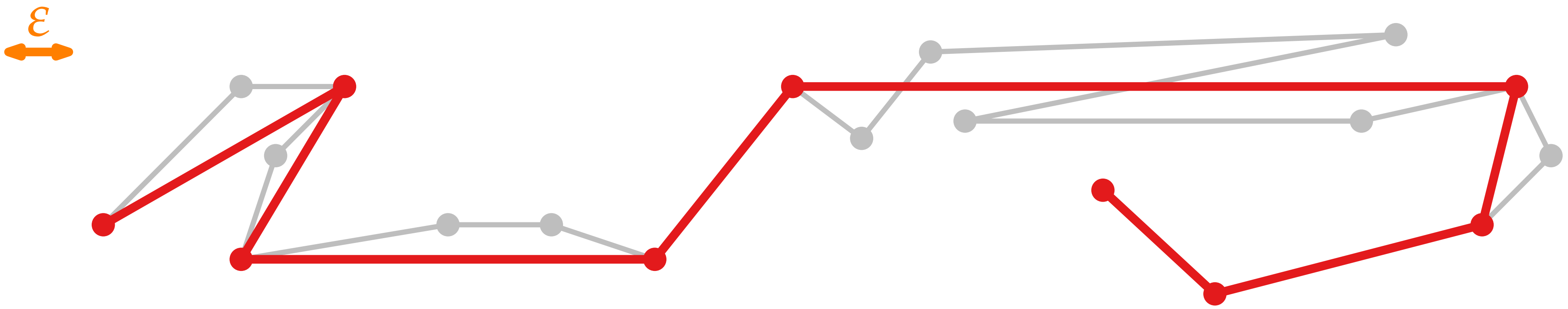
Typical distance measures d_X:

- *Local Hausdorff distance:* for each line segment $\overline{uv} \in P'$: $d_H(\overline{uv}, P[u,v]) \leq \varepsilon$
  where $d_H(L_1, L_2)$ is the undirected Hausdorff distance between curves L_1 and L_2

Definition: Polyline Simplification

Given:
- polyline P as a sequence of n points in the plane
- distance threshold ε

Find: Min. size subsequence P' of P, s.t. $d_X(P', P) \leq \varepsilon$ (keep start and end point).



Typical distance measures d_X:

- *Local Hausdorff distance:* for each line segment $\overline{uv} \in P'$: $d_H(\overline{uv}, P[u,v]) \leq \varepsilon$

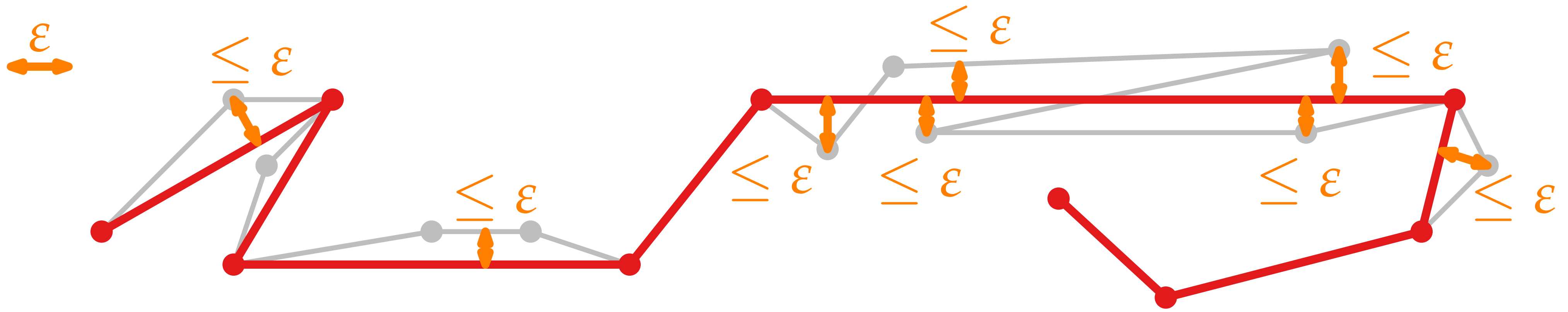
  where $d_H(L_1, L_2)$ is the undirected Hausdorff distance between curves L_1 and L_2

Definition: Polyline Simplification

Given:
- polyline P as a sequence of n points in the plane
- distance threshold ε

Find: Min. size subsequence P' of P, s.t. $d_X(P', P) \le \varepsilon$ (keep start and end point).

Typical distance measures d_X:



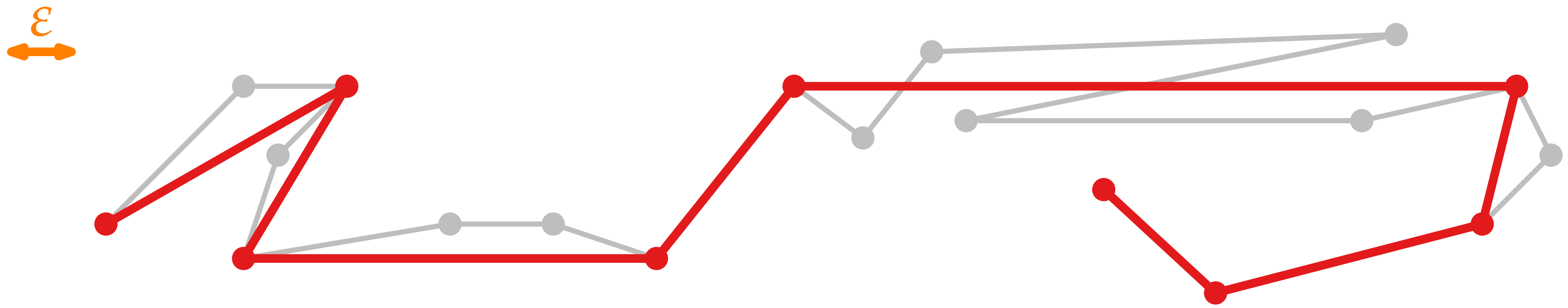
- *Local Hausdorff distance:* for each line segment $\overline{uv} \in P'$: $d_H(\overline{uv}, P[u, v]) \le \varepsilon$
  where $d_H(L_1, L_2)$ is the undirected Hausdorff distance between curves L_1 and L_2

Definition: Polyline Simplification

Given:
- polyline P as a sequence of n points in the plane
- distance threshold ε

Find: Min. size subsequence P' of P, s.t. $d_X(P', P) \leq \varepsilon$ (keep start and end point).

Typical distance measures d_X:

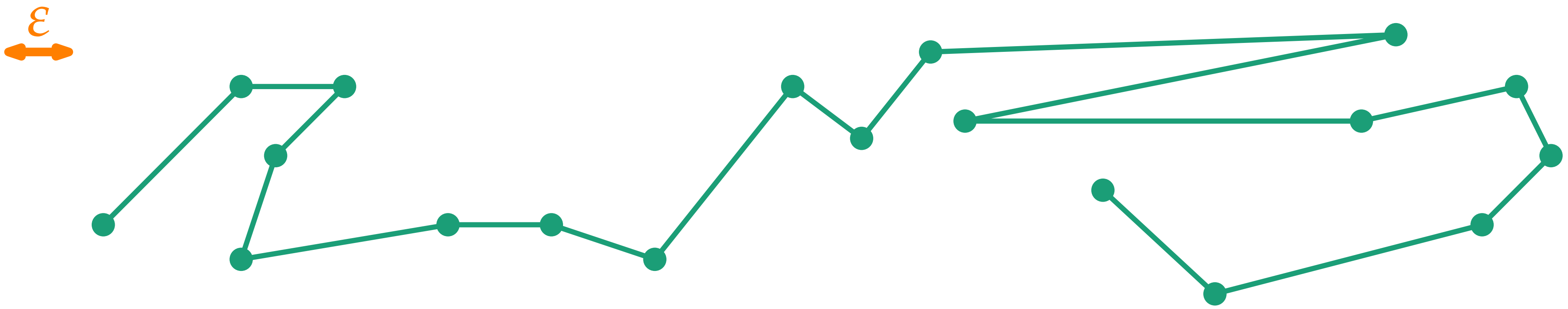


- *Local Hausdorff distance:* for each line segment $\overline{uv} \in P'$: $d_H(\overline{uv}, P[u, v]) \leq \varepsilon$
  where $d_H(L_1, L_2)$ is the undirected Hausdorff distance between curves L_1 and L_2

Definition: Polyline Simplification

Given:
- polyline P as a sequence of n points in the plane
- distance threshold ε

Find: Min. size subsequence P' of P, s.t. $d_X(P', P) \leq \varepsilon$ (keep start and end point).



Typical distance measures d_X:

- *Local Hausdorff distance:* for each line segment $\overline{uv} \in P'$: $d_H(\overline{uv}, P[u, v]) \leq \varepsilon$
  where $d_H(L_1, L_2)$ is the undirected Hausdorff distance between curves L_1 and L_2
- *Local Fréchet distance:* for each line segment $\overline{uv} \in P'$: $d_F(\overline{uv}, P[u, v]) \leq \varepsilon$
  where $d_F(L_1, L_2)$ is the Fréchet distance between curves L_1 and L_2

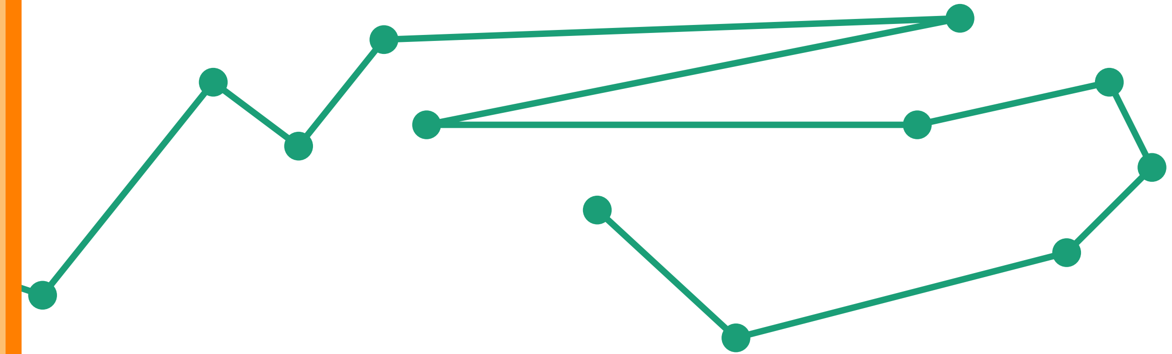
Definition: Polyline Simplification

Given:
- polyline P as a sequence of n points in the plane
- distance threshold ε

Find: M

$_{\mathrm{X}}(P', P) \leq \varepsilon$ (keep start and end point).

Typical measure

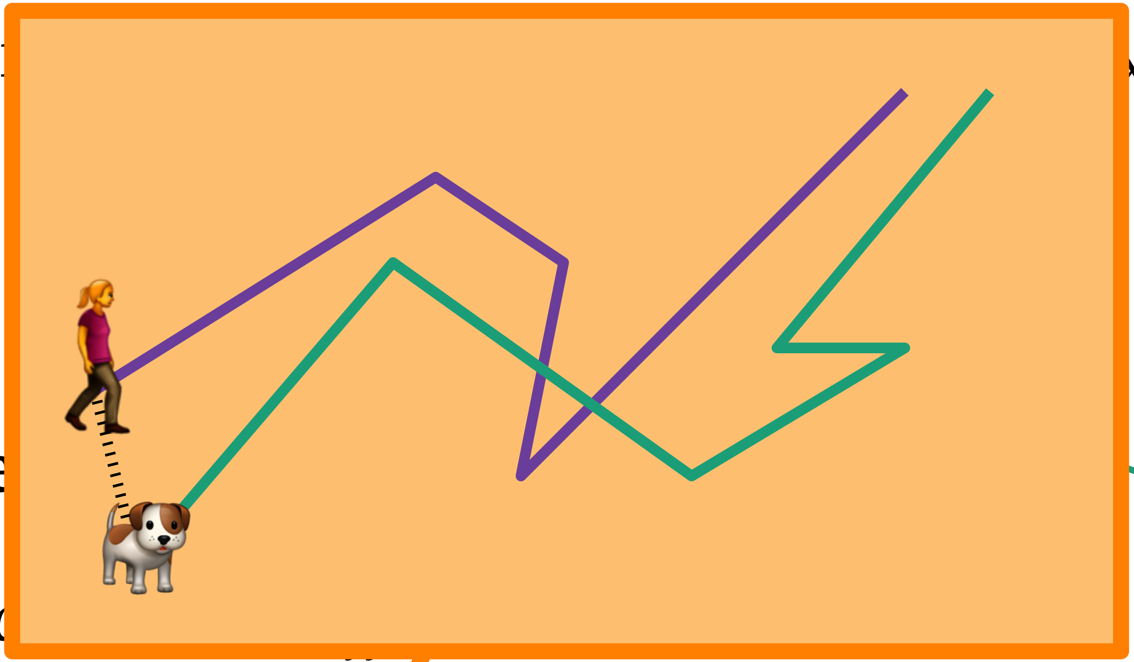


- *Loc* segment $\overline{uv} \in P'$: $d_{\mathrm{H}}(\overline{uv}, P[u,v]) \leq \varepsilon$
  where $d_{\mathrm{H}}(L_1, L_2)$ is the undirected Hausdorff distance between curves L_1 and L_2
- *Local Fréchet distance:* for each line segment $\overline{uv} \in P'$: $d_{\mathrm{F}}(\overline{uv}, P[u,v]) \leq \varepsilon$
  where $d_{\mathrm{F}}(L_1, L_2)$ is the Fréchet distance between curves L_1 and L_2

Definition: Polyline Simplification

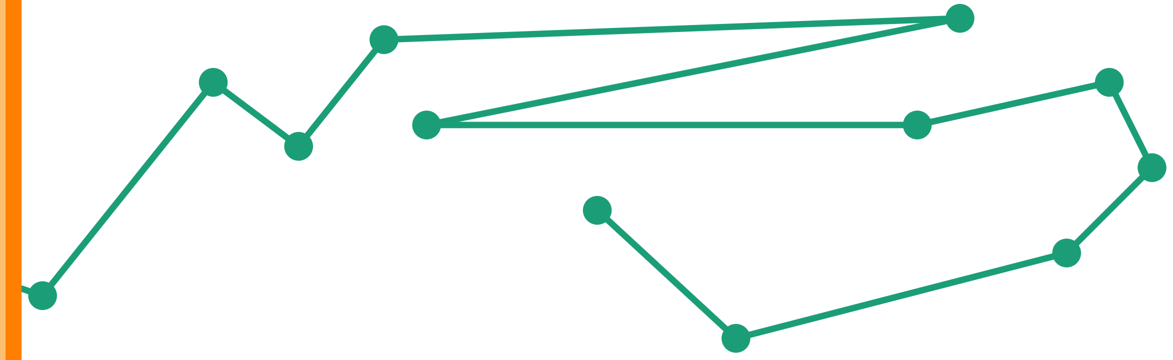
Given:
- polyline P as a sequence of n points in the plane
- distance threshold ε

Find: M



x$(P', P) \leq \varepsilon$ (keep start and end point).

Typical measure

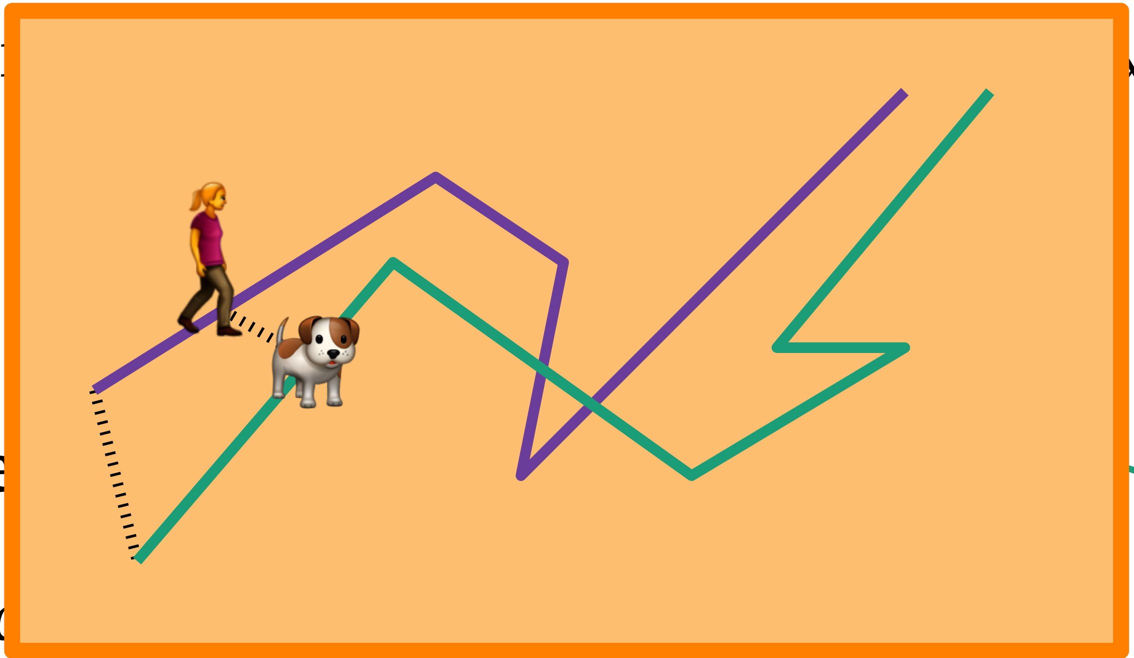


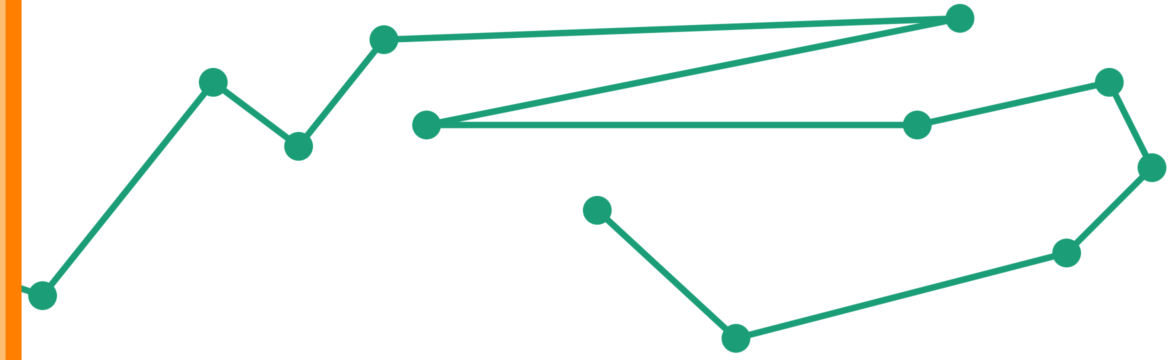
- *Loc* segment $\overline{uv} \in P'$: $d_{\mathrm{H}}(\overline{uv}, P[u,v]) \leq \varepsilon$
  where $d_{\mathrm{H}}(L_1, L_2)$ is the undirected Hausdorff distance between curves L_1 and L_2
- *Local Fréchet distance:* for each line segment $\overline{uv} \in P'$: $d_{\mathrm{F}}(\overline{uv}, P[u,v]) \leq \varepsilon$
  where $d_{\mathrm{F}}(L_1, L_2)$ is the Fréchet distance between curves L_1 and L_2

Definition: Polyline Simplification

Given:
- polyline P as a sequence of n points in the plane
- distance threshold ε

Find: M ... $_X(P', P) \leq \varepsilon$ (keep start and end point).





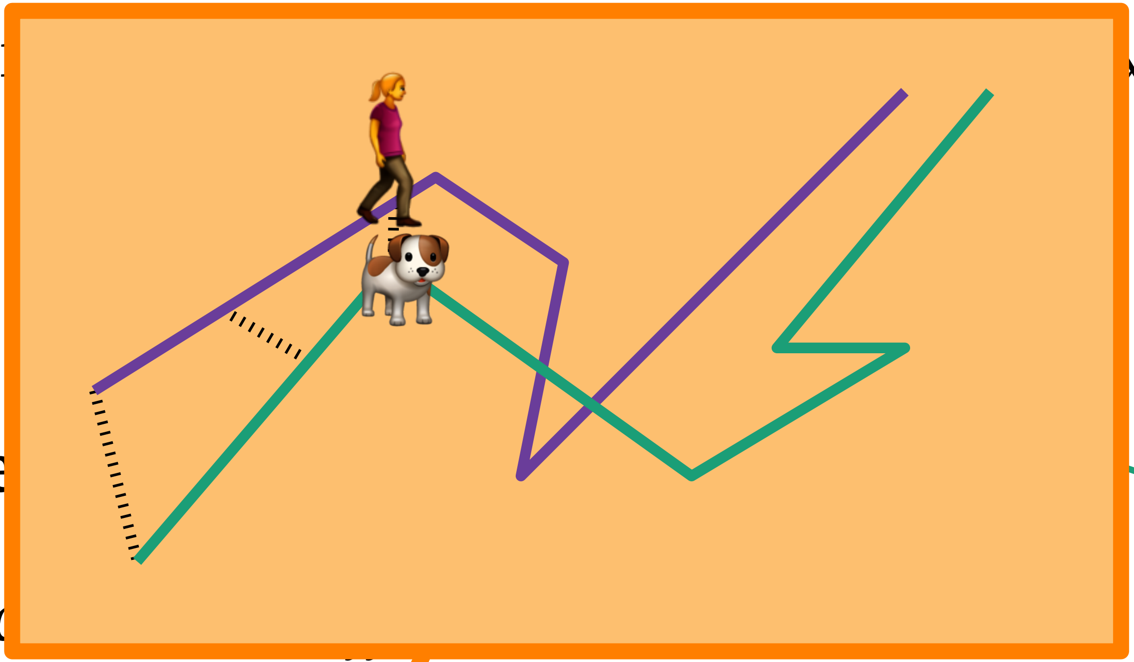
Typical measure

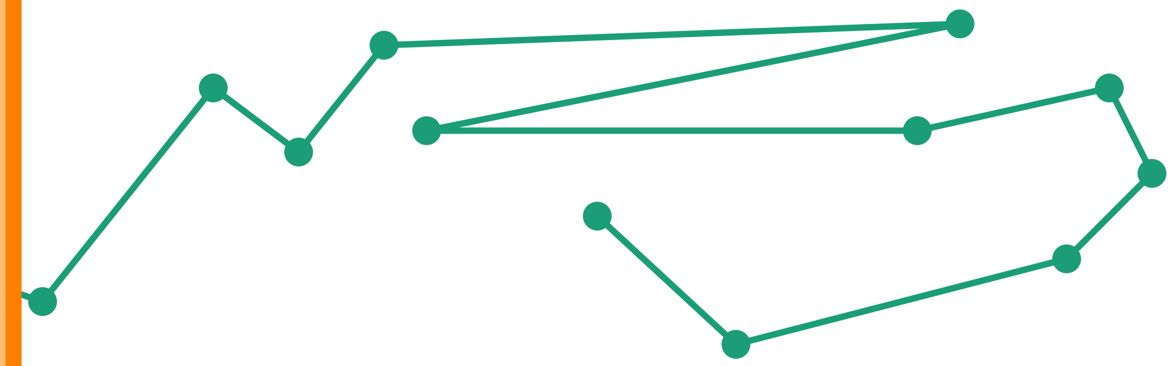
- *Loc* ... segment $\overline{uv} \in P'$: $d_H(\overline{uv}, P[u,v]) \leq \varepsilon$
  where $d_H(L_1, L_2)$ is the undirected Hausdorff distance between curves L_1 and L_2
- *Local Fréchet distance:* for each line segment $\overline{uv} \in P'$: $d_F(\overline{uv}, P[u,v]) \leq \varepsilon$
  where $d_F(L_1, L_2)$ is the Fréchet distance between curves L_1 and L_2

Definition: Polyline Simplification

Given:
- polyline P as a sequence of n points in the plane
- distance threshold ε

Find: M $_X(P', P) \leq \varepsilon$ (keep start and end point).





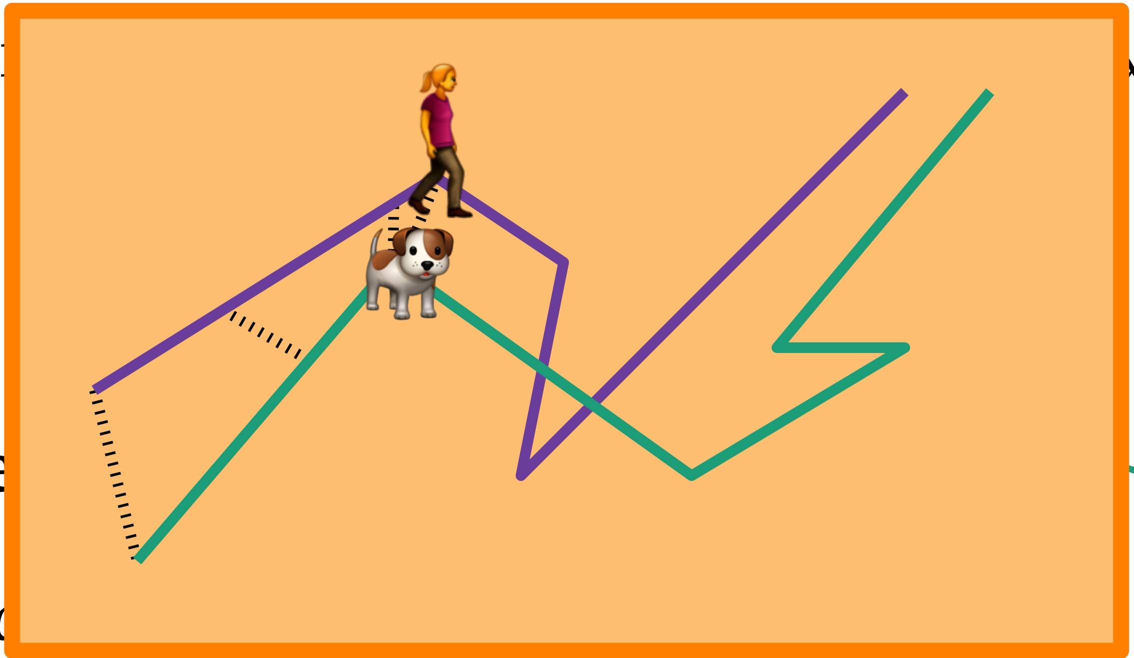
Typical measure

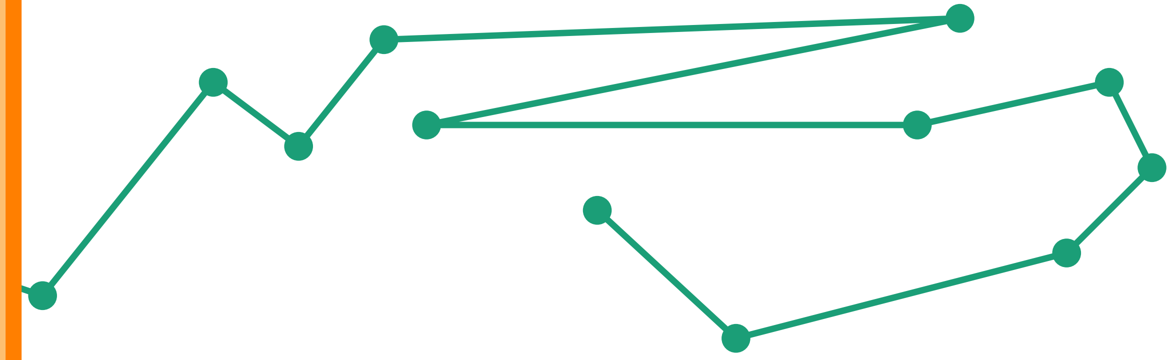
- *Loc* segment $\overline{uv} \in P'$: $d_\text{H}(\overline{uv}, P[u,v]) \leq \varepsilon$
  where $d_\text{H}(L_1, L_2)$ is the undirected Hausdorff distance between curves L_1 and L_2
- *Local Fréchet distance:* for each line segment $\overline{uv} \in P'$: $d_\text{F}(\overline{uv}, P[u,v]) \leq \varepsilon$
  where $d_\text{F}(L_1, L_2)$ is the Fréchet distance between curves L_1 and L_2

Definition: Polyline Simplification

Given:
- polyline P as a sequence of n points in the plane
- distance threshold ε

Find: M … $x(P', P) \leq \varepsilon$ (keep start and end point).





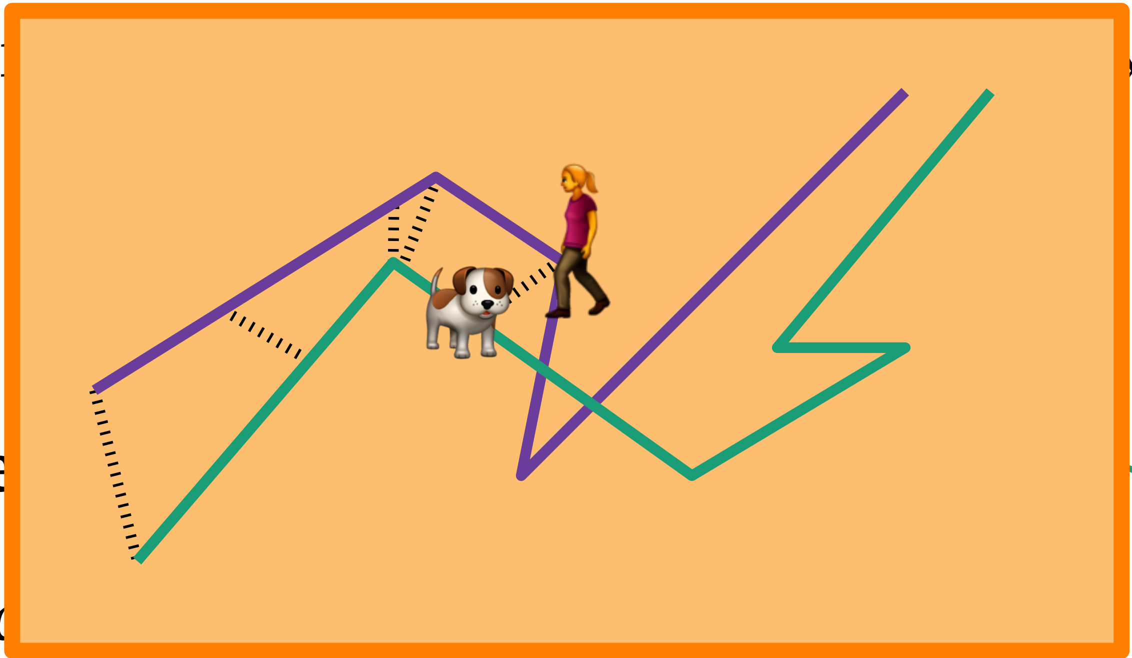
Typical measure…

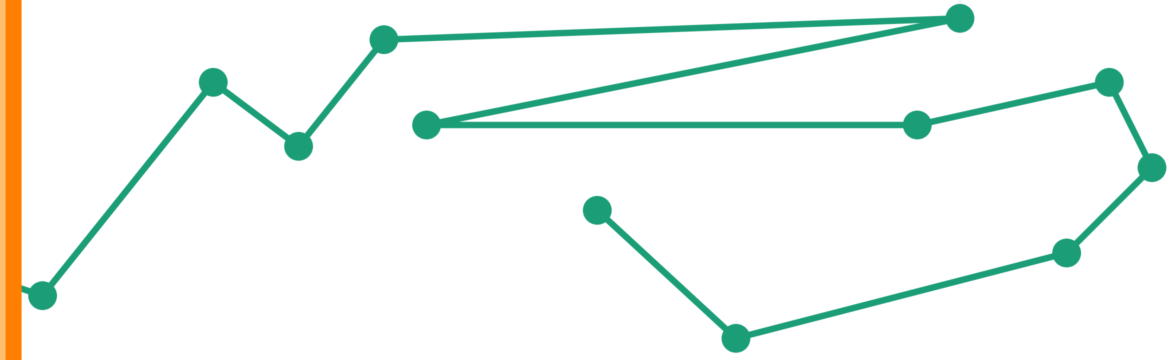
- *Loc*… segment $\overline{uv} \in P'$: $d_{\mathrm{H}}(\overline{uv}, P[u,v]) \leq \varepsilon$
  where $d_{\mathrm{H}}(L_1, L_2)$ is the undirected Hausdorff distance between curves L_1 and L_2
- *Local Fréchet distance:* for each line segment $\overline{uv} \in P'$: $d_{\mathrm{F}}(\overline{uv}, P[u,v]) \leq \varepsilon$
  where $d_{\mathrm{F}}(L_1, L_2)$ is the Fréchet distance between curves L_1 and L_2

Definition: Polyline Simplification

Given:
- polyline P as a sequence of n points in the plane
- distance threshold ε

Find: M $_X(P', P) \le \varepsilon$ (keep start and end point).



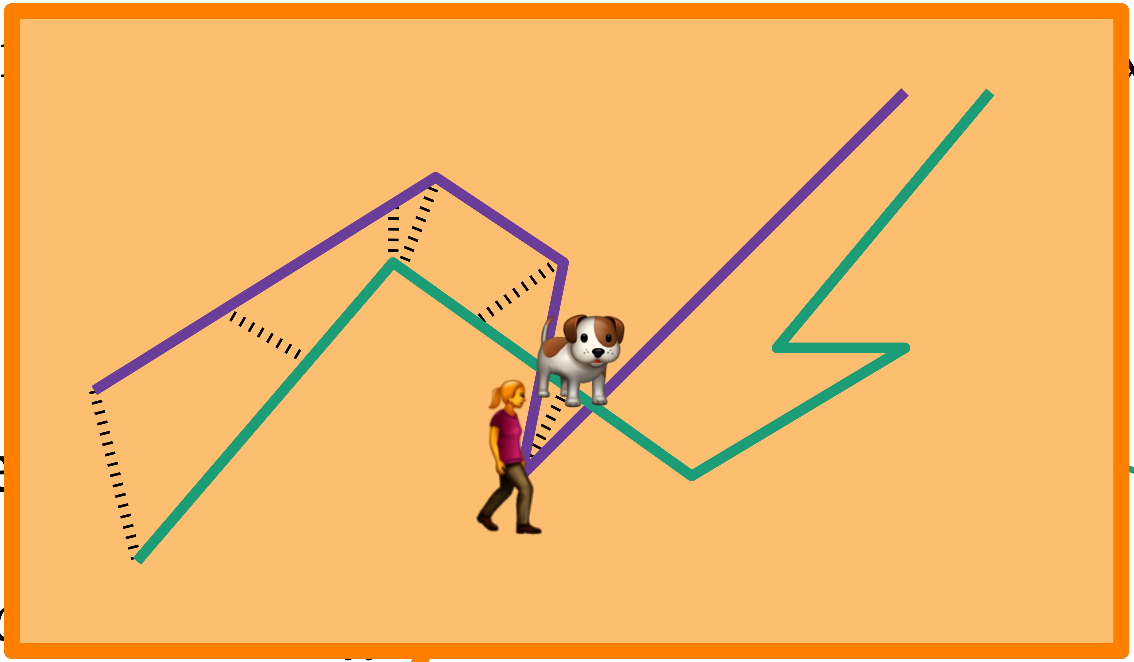


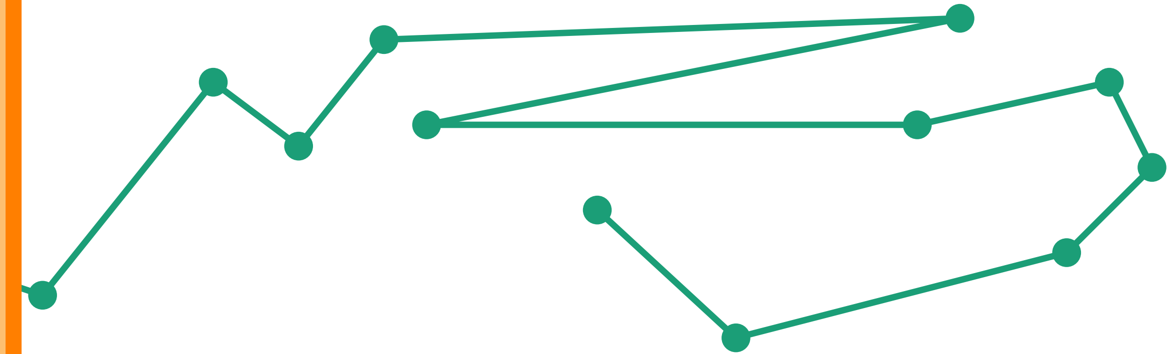
Typical measure

- *Loc* segment $\overline{uv} \in P'$: $d_\mathrm{H}(\overline{uv}, P[u,v]) \le \varepsilon$
  where $d_\mathrm{H}(L_1, L_2)$ is the undirected Hausdorff distance between curves L_1 and L_2
- *Local Fréchet distance:* for each line segment $\overline{uv} \in P'$: $d_\mathrm{F}(\overline{uv}, P[u,v]) \le \varepsilon$
  where $d_\mathrm{F}(L_1, L_2)$ is the Fréchet distance between curves L_1 and L_2

Definition: Polyline Simplification

Given:
- polyline P as a sequence of n points in the plane
- distance threshold ε

Find: M  $_x(P', P) \leq \varepsilon$ (keep start and end point).



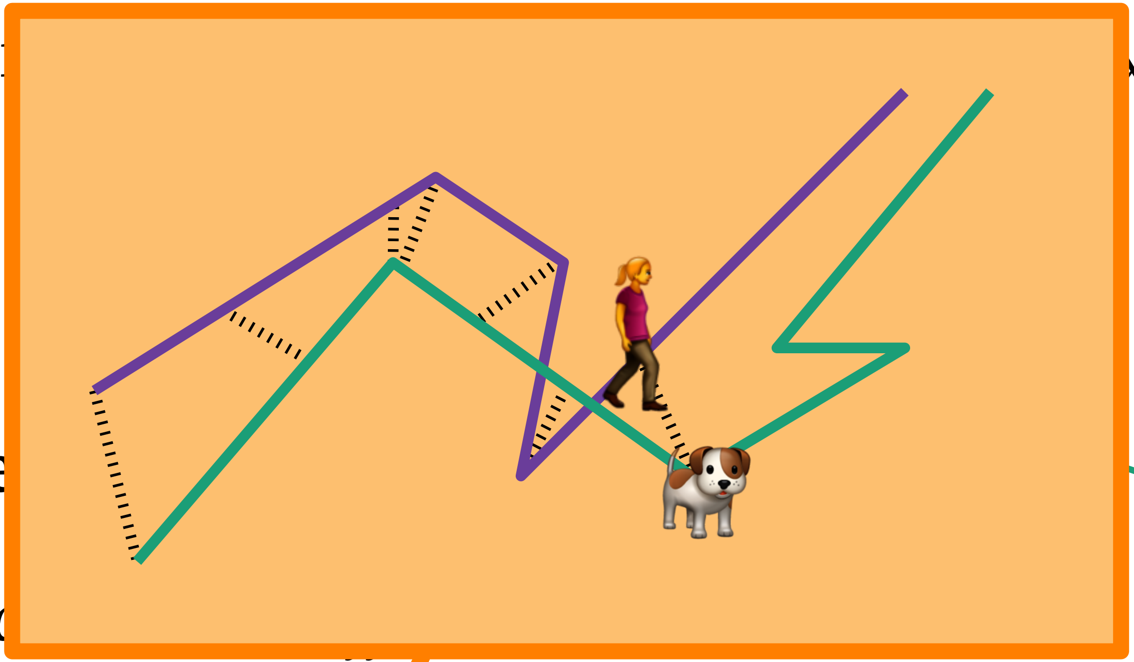
Typical measure

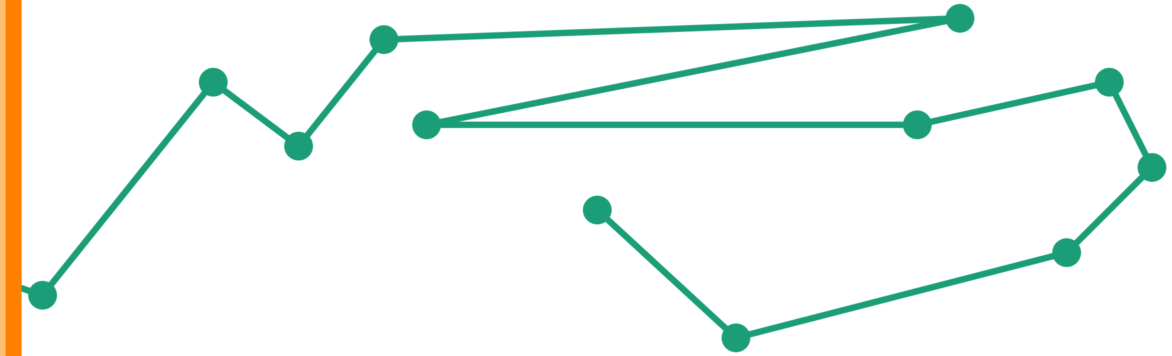
- *Lo* segment $\overline{uv} \in P'$: $d_{\mathrm{H}}(\overline{uv}, P[u,v]) \leq \varepsilon$
  where $d_{\mathrm{H}}(L_1, L_2)$ is the undirected Hausdorff distance between curves L_1 and L_2
- *Local Fréchet distance:* for each line segment $\overline{uv} \in P'$: $d_{\mathrm{F}}(\overline{uv}, P[u,v]) \leq \varepsilon$
  where $d_{\mathrm{F}}(L_1, L_2)$ is the Fréchet distance between curves L_1 and L_2

Definition: Polyline Simplification

Given:
- polyline P as a sequence of n points in the plane
- distance threshold ε

Find: M $_x(P', P) \leq \varepsilon$ (keep start and end point).





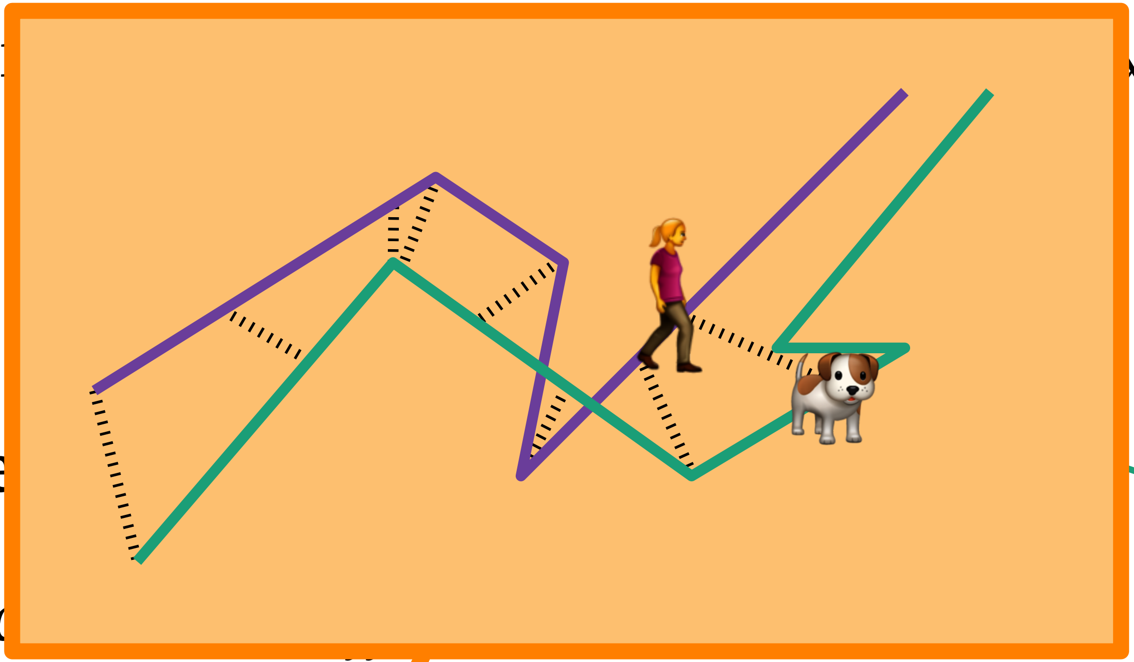
Typical measure

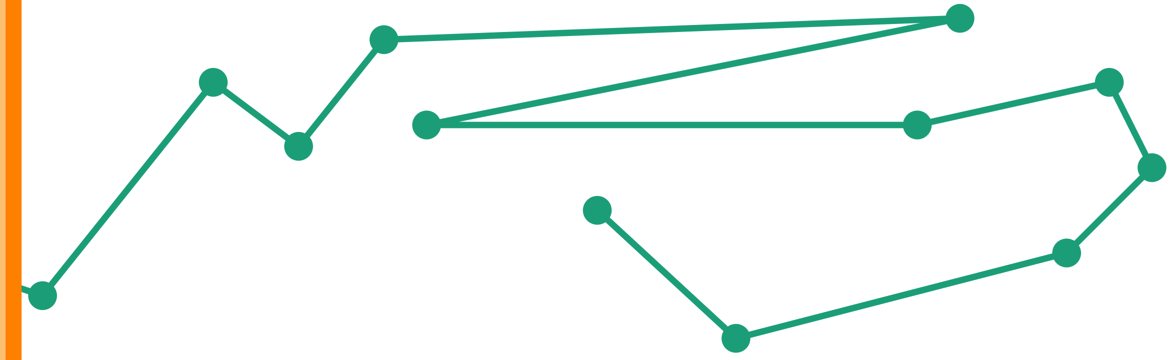
- *Loc* segment $\overline{uv} \in P'$: $d_H(\overline{uv}, P[u,v]) \leq \varepsilon$
  where $d_H(L_1, L_2)$ is the undirected Hausdorff distance between curves L_1 and L_2
- *Local Fréchet distance:* for each line segment $\overline{uv} \in P'$: $d_F(\overline{uv}, P[u,v]) \leq \varepsilon$
  where $d_F(L_1, L_2)$ is the Fréchet distance between curves L_1 and L_2

Definition: Polyline Simplification

Given:
- polyline P as a sequence of n points in the plane
- distance threshold ε

Find: M $_{x}(P', P) \leq \varepsilon$ (keep start and end point).





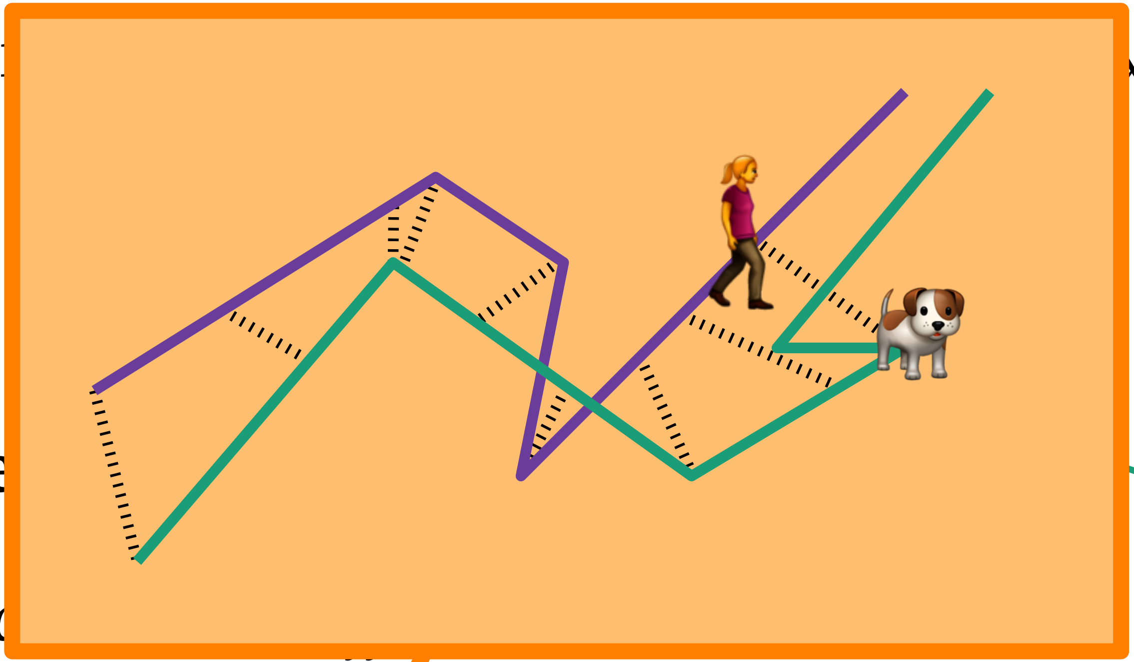
Typical measure

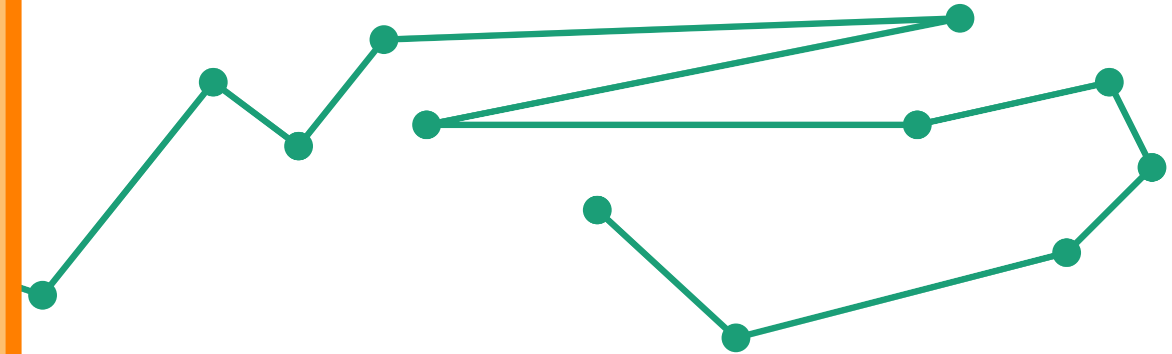
- *Loc* segment $\overline{uv} \in P'$: $d_{\mathrm{H}}(\overline{uv}, P[u,v]) \leq \varepsilon$
  where $d_{\mathrm{H}}(L_1, L_2)$ is the undirected Hausdorff distance between curves L_1 and L_2
- *Local Fréchet distance:* for each line segment $\overline{uv} \in P'$: $d_{\mathrm{F}}(\overline{uv}, P[u,v]) \leq \varepsilon$
  where $d_{\mathrm{F}}(L_1, L_2)$ is the Fréchet distance between curves L_1 and L_2

Definition: Polyline Simplification

Given:
- polyline P as a sequence of n points in the plane
- distance threshold ε

Find: M $_X(P', P) \leq \varepsilon$ (keep start and end point).





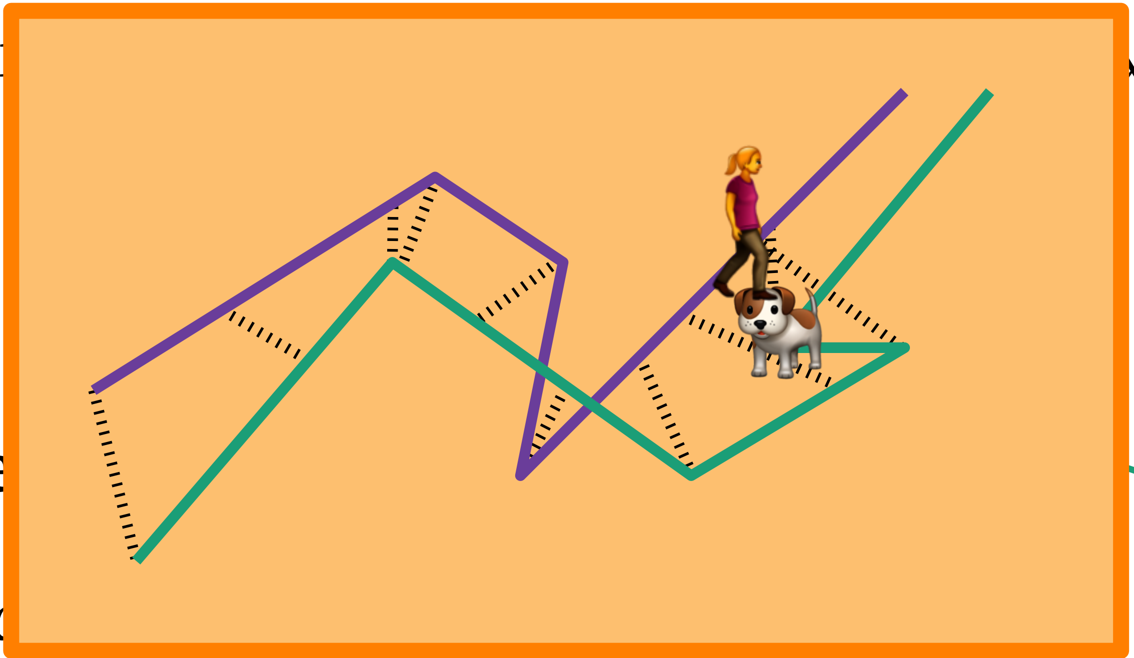
Typical measure

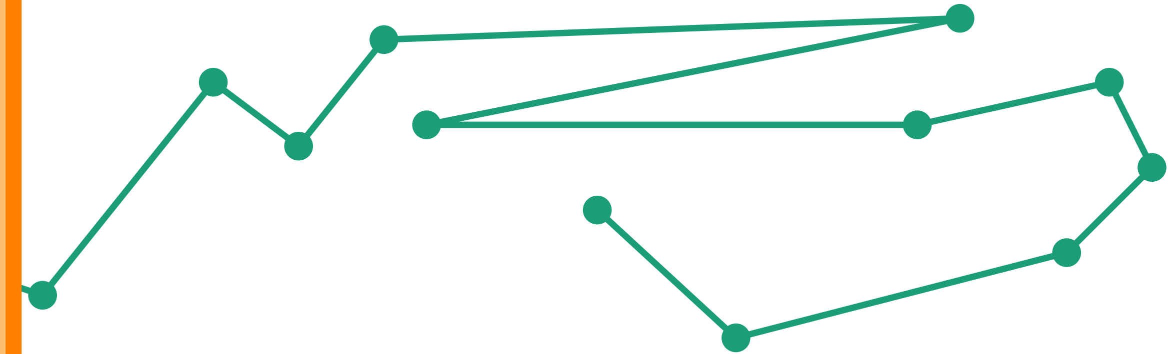
- *Loc* segment $\overline{uv} \in P'$: $d_{\mathrm{H}}(\overline{uv}, P[u,v]) \leq \varepsilon$
  where $d_{\mathrm{H}}(L_1, L_2)$ is the undirected Hausdorff distance between curves L_1 and L_2
- *Local Fréchet distance:* for each line segment $\overline{uv} \in P'$: $d_{\mathrm{F}}(\overline{uv}, P[u,v]) \leq \varepsilon$
  where $d_{\mathrm{F}}(L_1, L_2)$ is the Fréchet distance between curves L_1 and L_2

Definition: Polyline Simplification

Given:
- polyline P as a sequence of n points in the plane
- distance threshold ε

Find: M $_{\text{X}}(P', P) \leq \varepsilon$ (keep start and end point).





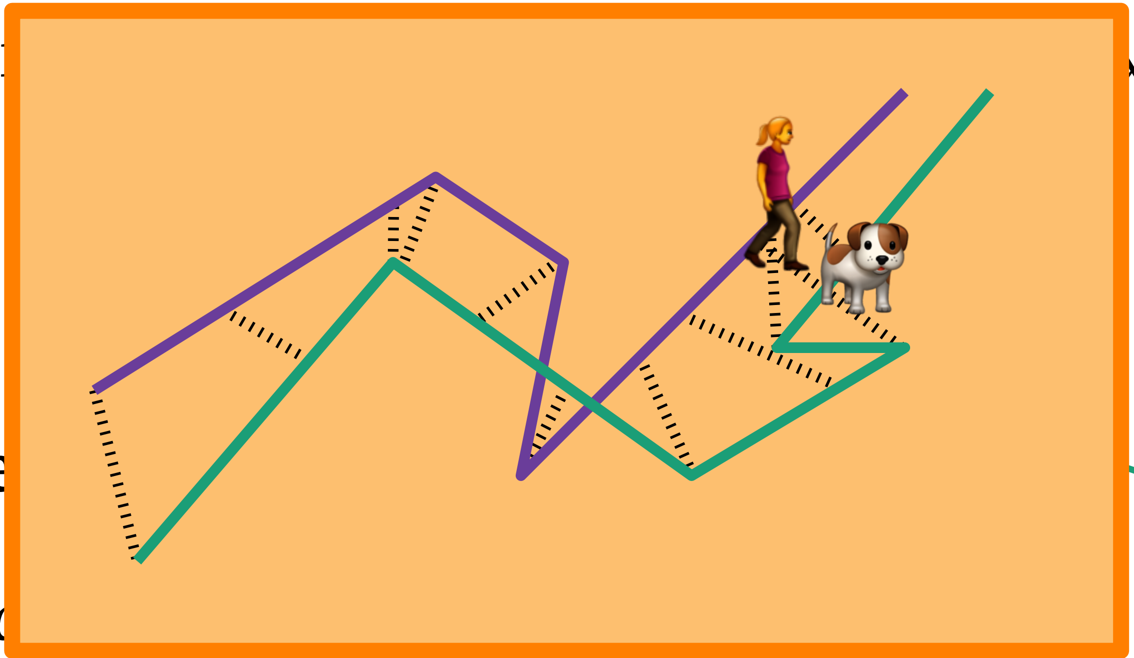
Typical measure

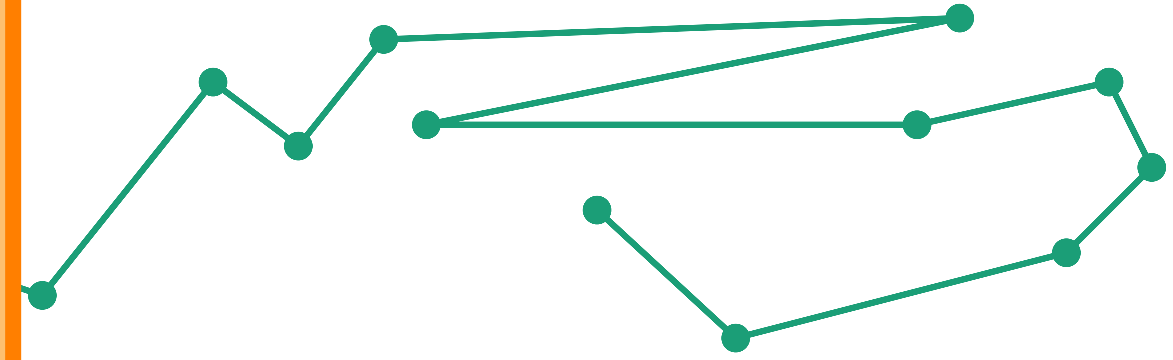
- *Lo* segment $\overline{uv} \in P'$: $d_{\mathrm{H}}(\overline{uv}, P[u,v]) \leq \varepsilon$
  where $d_{\mathrm{H}}(L_1, L_2)$ is the undirected Hausdorff distance between curves L_1 and L_2
- *Local Fréchet distance:* for each line segment $\overline{uv} \in P'$: $d_{\mathrm{F}}(\overline{uv}, P[u,v]) \leq \varepsilon$
  where $d_{\mathrm{F}}(L_1, L_2)$ is the Fréchet distance between curves L_1 and L_2

Definition: Polyline Simplification

Given:
- polyline P as a sequence of n points in the plane
- distance threshold ε

Find: M $_X(P', P) \leq \varepsilon$ (keep start and end point).





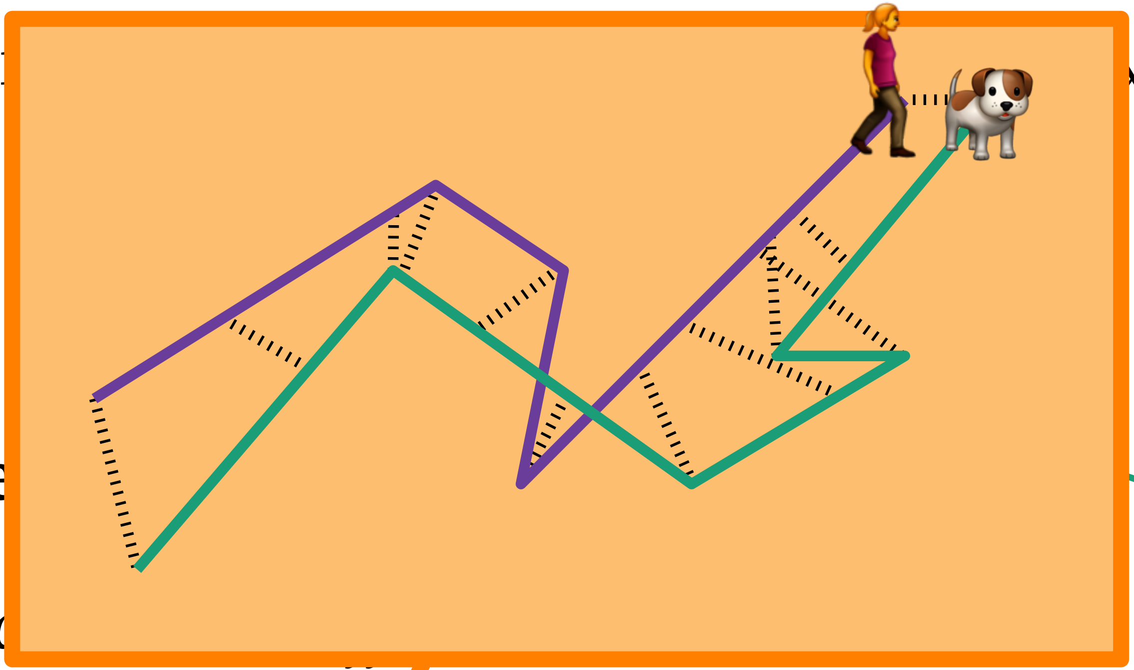
Typical measure

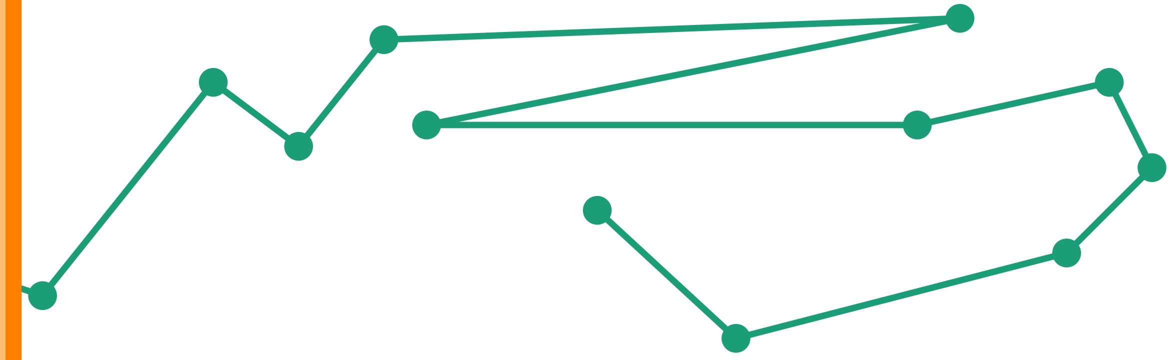
- *Lo* segment $\overline{uv} \in P'$: $d_H(\overline{uv}, P[u,v]) \leq \varepsilon$
  where $d_H(L_1, L_2)$ is the undirected Hausdorff distance between curves L_1 and L_2
- *Local Fréchet distance:* for each line segment $\overline{uv} \in P'$: $d_F(\overline{uv}, P[u,v]) \leq \varepsilon$
  where $d_F(L_1, L_2)$ is the Fréchet distance between curves L_1 and L_2

Definition: Polyline Simplification

Given: ■ polyline P as a sequence of n points in the plane

■ distance threshold ε

Find: M  $_X(P', P) \leq \varepsilon$ (keep start and end point).



Typical measure

■ *Lo* segment $\overline{uv} \in P'$: $d_{\mathrm{H}}(\overline{uv}, P[u,v]) \leq \varepsilon$

where $d_{\mathrm{H}}(L_1, L_2)$ is the undirected Hausdorff distance between curves L_1 and L_2

■ *Local Fréchet distance:* for each line segment $\overline{uv} \in P'$: $d_{\mathrm{F}}(\overline{uv}, P[u,v]) \leq \varepsilon$

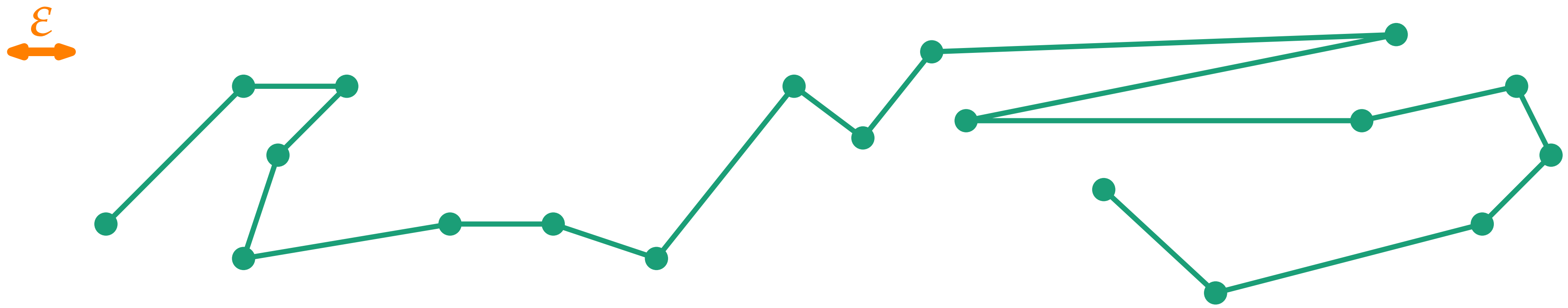
where $d_{\mathrm{F}}(L_1, L_2)$ is the Fréchet distance between curves L_1 and L_2

Definition: Polyline Simplification

Given:
- polyline P as a sequence of n points in the plane
- distance threshold ε

Find: Min. size subsequence P' of P, s.t. $d_X(P', P) \leq \varepsilon$ (keep start and end point).

Typical distance measures d_X:

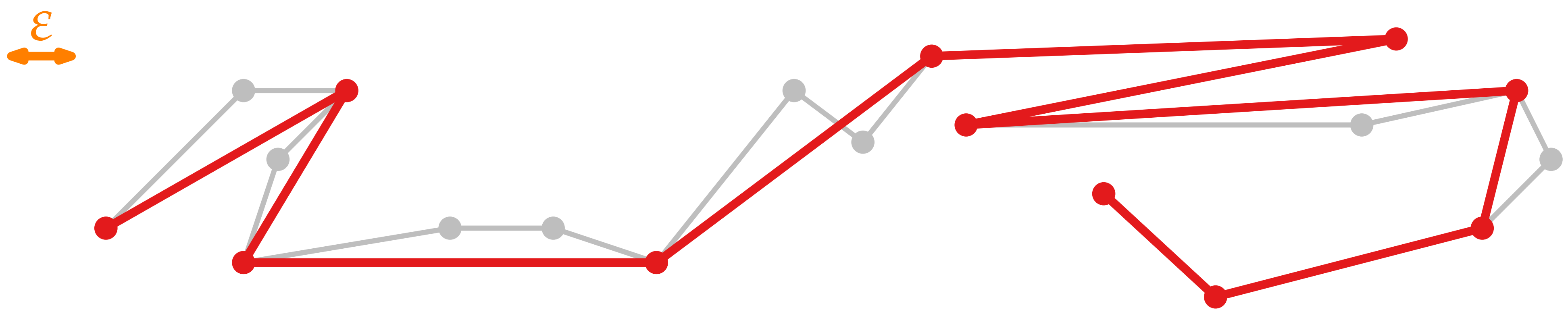


- *Local Hausdorff distance:* for each line segment $\overline{uv} \in P'$: $d_H(\overline{uv}, P[u, v]) \leq \varepsilon$
  where $d_H(L_1, L_2)$ is the undirected Hausdorff distance between curves L_1 and L_2
- *Local Fréchet distance:* for each line segment $\overline{uv} \in P'$: $d_F(\overline{uv}, P[u, v]) \leq \varepsilon$
  where $d_F(L_1, L_2)$ is the Fréchet distance between curves L_1 and L_2

Definition: Polyline Simplification

Given:
- polyline P as a sequence of n points in the plane
- distance threshold ε

Find: Min. size subsequence P' of P, s.t. $d_X(P', P) \leq \varepsilon$ (keep start and end point).



Typical distance measures d_X:

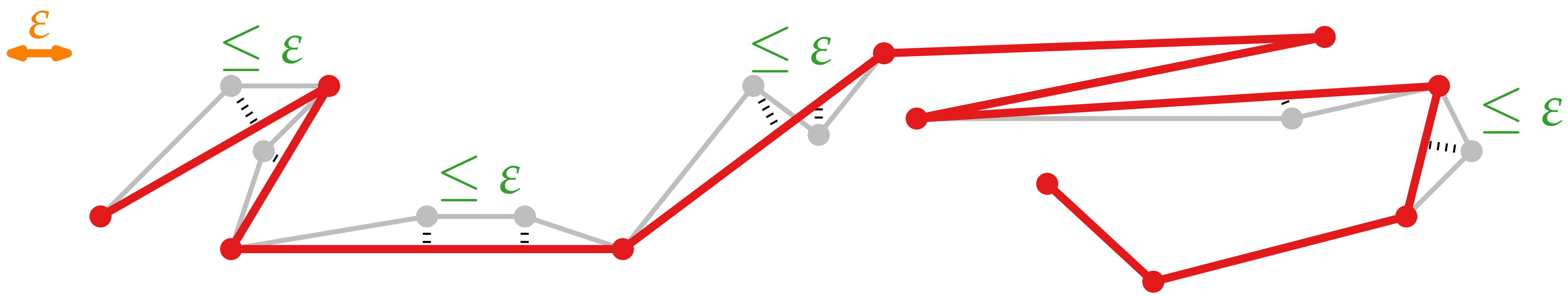
- *Local Hausdorff distance:* for each line segment $\overline{uv} \in P'$: $d_H(\overline{uv}, P[u,v]) \leq \varepsilon$
  where $d_H(L_1, L_2)$ is the undirected Hausdorff distance between curves L_1 and L_2
- *Local Fréchet distance:* for each line segment $\overline{uv} \in P'$: $d_F(\overline{uv}, P[u,v]) \leq \varepsilon$
  where $d_F(L_1, L_2)$ is the Fréchet distance between curves L_1 and L_2

Definition: Polyline Simplification

Given:
- polyline P as a sequence of n points in the plane
- distance threshold ε

Find: Min. size subsequence P' of P, s.t. $d_X(P', P) \leq \varepsilon$ (keep start and end point).

Typical distance measures d_X:

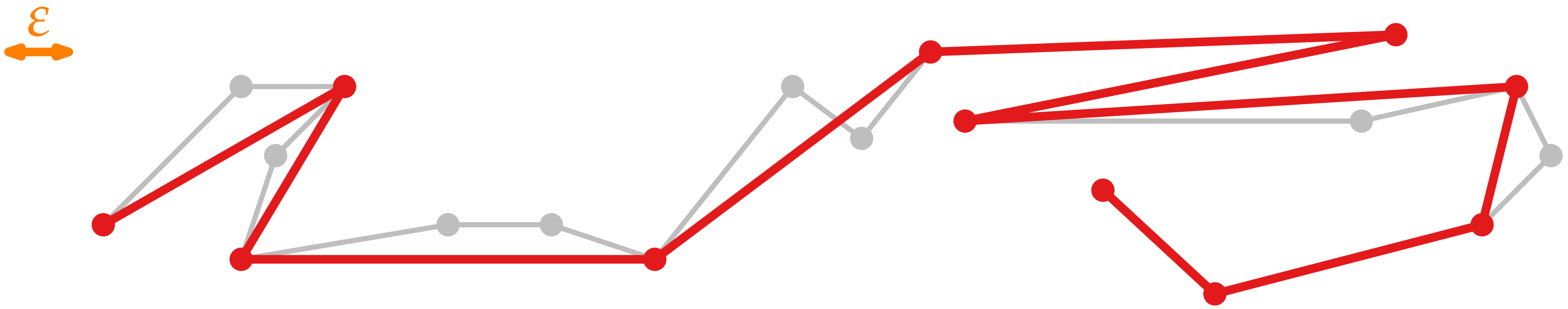


- *Local Hausdorff distance:* for each line segment $\overline{uv} \in P'$: $d_H(\overline{uv}, P[u,v]) \leq \varepsilon$
  where $d_H(L_1, L_2)$ is the undirected Hausdorff distance between curves L_1 and L_2
- *Local Fréchet distance:* for each line segment $\overline{uv} \in P'$: $d_F(\overline{uv}, P[u,v]) \leq \varepsilon$
  where $d_F(L_1, L_2)$ is the Fréchet distance between curves L_1 and L_2

Definition: Polyline Simplification

Given:
- polyline P as a sequence of n points in the plane
- distance threshold ε

Find: Min. size subsequence P' of P, s.t. $d_X(P', P) \leq \varepsilon$ (keep start and end point).



Typical distance measures d_X:

- *Local Hausdorff distance:* for each line segment $\overline{uv} \in P'$: $d_H(\overline{uv}, P[u,v]) \leq \varepsilon$
  where $d_H(L_1, L_2)$ is the undirected Hausdorff distance between curves L_1 and L_2
- *Local Fréchet distance:* for each line segment $\overline{uv} \in P'$: $d_F(\overline{uv}, P[u,v]) \leq \varepsilon$
  where $d_F(L_1, L_2)$ is the Fréchet distance between curves L_1 and L_2

Related Work & Contribution

Running time for finding an optimal polyline simplification …

Related Work & Contribution

minimum size

Running time for finding an optimal polyline simplification . . .

Related Work & Contribution

minimum size

Running time for finding an optimal polyline simplification …

… under the local Hausdorff distance:

Related Work & Contribution

minimum size

Running time for finding an optimal polyline simplification . . .

. . . under the local Hausdorff distance:

- $O(n^3)$ [Imai, Iri ’88]

Related Work & Contribution

minimum size

Running time for finding an optimal polyline simplification . . .

. . . under the local Hausdorff distance:

- $O(n^3)$ [Imai, Iri '88]
- $O(n^2 \log n)$ [Melkman, O'Rourke '88]

Related Work & Contribution

minimum size

Running time for finding an optimal polyline simplification …

… under the local Hausdorff distance:

- $O(n^3)$ [Imai, Iri '88]
- $O(n^2 \log n)$ [Melkman, O'Rourke '88]
- $O(n^2)$ time [Chan, Chin '96]

Related Work & Contribution

minimum size

Running time for finding an optimal polyline simplification …

…under the local Hausdorff distance:

- $O(n^3)$ [Imai, Iri '88]
- $O(n^2 \log n)$ [Melkman, O'Rourke '88]
- $O(n^2)$ time [Chan, Chin '96]

…under the local Fréchet distance:

Related Work & Contribution

minimum size

Running time for finding an optimal polyline simplification …

… under the local Hausdorff distance:

- $O(n^3)$ [Imai, Iri ’88]
- $O(n^2 \log n)$ [Melkman, O’Rourke ’88]
- $O(n^2)$ time [Chan, Chin ’96]

… under the local Fréchet distance:

- $O(n^3)$ [Godau ’91]

Related Work & Contribution

minimum size

Running time for finding an optimal polyline simplification …

… under the local Hausdorff distance:

- $O(n^3)$ [Imai, Iri ’88]
- $O(n^2 \log n)$ [Melkman, O’Rourke ’88]
- $O(n^2)$ time [Chan, Chin ’96]

… under the local Fréchet distance:

- $O(n^3)$ [Godau ’91]
- ?

Related Work & Contribution

minimum size

Running time for finding an optimal polyline simplification …

… under the local Hausdorff distance:

- $O(n^3)$ [Imai, Iri ’88]
- an, O’Rourke ’88]
- Chin ’96]

… under the local Fréchet distance:

- $O(n^3)$ [Godau ’91]
- ?

Algorithmica (2005) 42: 203–219
DOI: 10.1007/s00453-005-1165-y

Algorithmica

Near-Linear Time Approximation Algorithms for Curve Simplification[1]

Pankaj K. Agarwal,[2] Sariel Har-Peled,[3] Nabil H. Mustafa,[2] and Yusu Wang[2]

5. Conclusions. In this paper we presented near-linear approximation algorithms for curve simplification under the Hausdorff and Fréchet error measures. We presented the first efficient approximation algorithm for Fréchet simplifications of a curve in dimension higher than two. Our experimental results demonstrate that our algorithms are efficient.

We conclude by mentioning a few open problems:

(i) Does there exist a near-linear algorithm for computing an ε-simplification of size at most $c\kappa_M(\varepsilon, P)$ for a polygonal curve P, where $c \geq 1$ is a constant?
(ii) Is it possible to compute the optimal ε-simplification under the Hausdorff error measure in near-linear time, or under the Fréchet error measure in subcubic time?
(iii) Is there any provably efficient exact/approximation algorithm for curve simplification in $\mathbb{R}^2$ that returns a simple curve if the input curve is simple.

2005

Related Work & Contribution

minimum size

Running time for finding an optimal polyline simplification …

… under the local Hausdorff distance:

- $O(n^3)$ [Imai, Iri '88]

… under the local Fréchet distance:

- $O(n^3)$ [Godau '91]
- ?

Algorithmica (2005) 42: 203–219
DOI: 10.1007/s00453-005-1165-y
Algorithmica

Near-Linear Time Approximation Algorithms for Curve Simplification[1]

Pankaj K. Agarwal,[2] Sariel Har-Peled,[3] Nabil H. Mustafa,[2] and Yusu Wang[2]

5. Conclusions. In this paper we presented near-linear approximation algorithms for curve simplification under the Hausdorff and Fréchet error measures. We presented the first efficient approximation algorithm for Fréchet simplifications of a curve in dimension higher than two. Our experimental results demonstrate that our algorithms are efficient.

We conclude by mentioning a few open problems:

(i) Does there exist a near-linear algorithm for computing an ε-simplification of size at most $c\kappa_M(\varepsilon, P)$ for a polygonal curve P, where $c \geq 1$ is a constant?
(ii) Is it possible to compute the optimal ε-simplification under the Hausdorff error measure in near-linear time, or under the Fréchet error measure in subcubic time?
(iii) Is there any provably efficient exact/approximation algorithm for curve simplification in $\mathbb{R}^2$ that returns a simple curve if the input curve is simple.

2005

Polyline Simplification has Cubic Complexity

Karl Bringmann
Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany
kbringma@mpi-inf.mpg.de
Bhaskar Ray Chaudhury
Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany
Graduate School of Computer Science Saarbrücken, Saarland Informatics Campus, Germany
braycha@mpi-inf.mpg.de

35th International Symposium on Computational Geometry (SoCG 2019).
Editors: Gill Barequet and Yusu Wang; Article No. 18; pp. 18:1–18:16
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The classic $\hat{O}(n^3)$[1] time algorithm by Imai and Iri [18] was designed for Local-Hausdorff simplification. By changing the distance computation in this algorithm for the Fréchet distance, one can obtain an $\hat{O}(n^3)$-time algorithm for Local-Fréchet [15]. There are improvements for Local-Hausdorff simplification in small dimension d [19, 8, 6]; the fastest running times are $2^{O(d)}n^2$ for L_1-norm, $\hat{O}(n^2)$ for L_∞-norm, and $\hat{O}(n^{3-\Omega(1/d)})$ for L_2-norm [6].

2019

Related Work & Contribution

minimum size

Running time for finding an optimal polyline simplification …

… under the local Hausdorff distance:

- $O(n^3)$ [Imai, Iri '88]

… under the local Fréchet distance:

- $O(n^3)$ [Godau '91]

Algorithmica (2005) 42: 203–219
DOI: 10.1007/s00453-005-1165-y
Algorithmica

Near-Linear Time Approximation Algorithms for Curve Simplification[1]

Pankaj K. Agarwal,[2] Sariel Har-Peled,[3] Nabil H. Mustafa,[2] and Yusu Wang[2]

5. Conclusions. In this paper we presented near-linear approximation algorithms for curve simplification under the Hausdorff and Fréchet error measures. We presented the first efficient approximation algorithm for Fréchet simplifications of a curve in dimension higher than two. Our experimental results demonstrate that our algorithms are efficient.

We conclude by mentioning a few open problems:

(i) Does there exist a near-linear algorithm for computing an ε-simplification of size at most $c\kappa_M(\varepsilon, P)$ for a polygonal curve P, where $c \geq 1$ is a constant?
(ii) Is it possible to compute the optimal ε-simplification under the Hausdorff error measure in near-linear time, or under the Fréchet error measure in subcubic time?
(iii) Is there any provably efficient exact/approximation algorithm for curve simplification in $\mathbb{R}^2$ that returns a simple curve if the input curve is simple.

2005

Polyline Simplification has Cubic Complexity

Karl Bringmann
Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany
kbringma@mpi-inf.mpg.de
Bhaskar Ray Chaudhury
Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany
Graduate School of Computer Science Saarbrücken, Saarland Informatics Campus, Germany
braycha@mpi-inf.mpg.de

35th International Symposium on Computational Geometry (SoCG 2019).
Editors: Gill Barequet and Yusu Wang; Article No. 18; pp. 18:1–18:16
LIPICS Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The classic $\hat{O}(n^3)^1$ time algorithm by Imai and Iri [18] was designed for Local-Hausdorff simplification. By changing the distance computation in this algorithm for the Fréchet distance, one can obtain an $\hat{O}(n^3)$-time algorithm for Local-Fréchet [15]. There are improvements for Local-Hausdorff simplification in small dimension d [19, 8, 6]; the fastest running times are $2^{O(d)}n^2$ for L_1-norm, $\hat{O}(n^2)$ for L_∞-norm, and $\hat{O}(n^{3-\Omega(1/d)})$ for L_2-norm [6].

2019

Journal of Computational Geometry jocg.org

ON OPTIMAL POLYLINE SIMPLIFICATION USING THE HAUSDORFF AND FRÉCHET DISTANCE*

Marc van Kreveld,† *Maarten Löffler,*† *and Lionov Wiratma*†‡

Abstract. We revisit the classical polygonal line simplification problem and study it using the Hausdorff distance and Fréchet distance. Interestingly, no previous authors studied line simplification under these measures in its pure form, namely: for a given $\varepsilon > 0$, choose a minimum size subsequence of the vertices of the input such that the Hausdorff or Fréchet distance between the input and output polylines is at most ε.

Table 1: Algorithmic results.

	Douglas-Peucker	Imai-Iri	Optimal
Hausdorff distance	$O(n \log n)$ [19]	$O(n^2)$ [10]	NP-hard *(this paper)*
Fréchet distance	$O(n^2)$ *(easy)*	$O(n^3)$ [17]	$O(kn^5)$ *(this paper)*

n: vertices in the input polyline; k: output complexity of the simplified polyline

2020

Related Work & Contribution

minimum size

Running time for finding an optimal polyline simplification …

…under the local Hausdorff distance:

- $O(n^3)$ [Imai, Iri ’88]
- $O(n^2 \log n)$ [Melkman, O’Rourke ’88]
- $O(n^2)$ time [Chan, Chin ’96]

…under the local Fréchet distance:

- $O(n^3)$ [Godau ’91]
- ?

Related Work & Contribution

minimum size

Running time for finding an optimal polyline simplification …

…under the local Hausdorff distance:

- $O(n^3)$ [Imai, Iri '88]
- $O(n^2 \log n)$ [Melkman, O'Rourke '88]
- $O(n^2)$ time [Chan, Chin '96]

…under the local Fréchet distance:

- $O(n^3)$ [Godau '91]
- $O(n^2 \log n)$

our contribution

Related Work & Contribution

minimum size

Running time for finding an optimal polyline simplification …

… under the local Hausdorff distance:

L_2 norm

- $O(n^3)$ [Imai, Iri '88]
- $O(n^2 \log n)$ [Melkman, O'Rourke '88]
- $O(n^2)$ time [Chan, Chin '96]

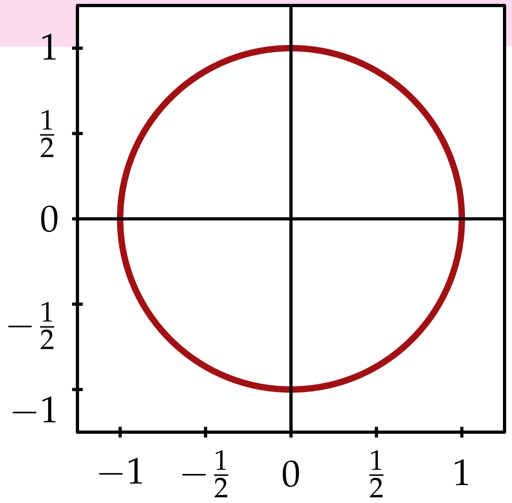
… under the local Fréchet distance:

- $O(n^3)$ [Godau '91]
- $O(n^2 \log n)$

our contribution

L_p:

$p = 2$



Related Work & Contribution

minimum size

Running time for finding an optimal polyline simplification …

… under the local Hausdorff distance:

… under the local Fréchet distance:

L_2 norm

- $O(n^3)$ [Imai, Iri '88] → $O(n^3)$ [Godau '91]
- $O(n^2 \log n)$ [Melkman, O'Rourke '88] → $O(n^2 \log n)$
- $O(n^2)$ time [Chan, Chin '96]

L_1 norm

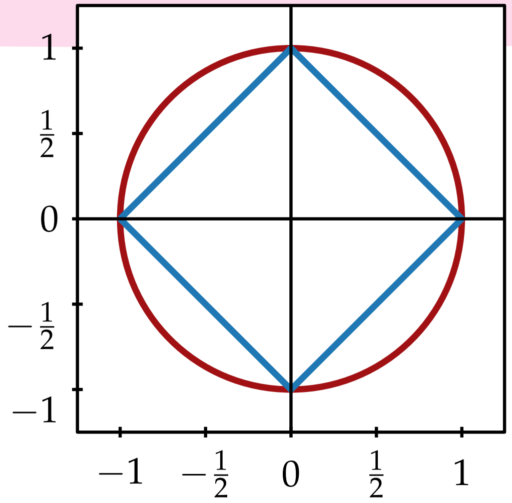
- $O(n^2)$

our contribution

L_p:

$p = 2$

$p = 1$



Related Work & Contribution

minimum size

Running time for finding an optimal polyline simplification …

… under the local Hausdorff distance:

L_2 norm

- $O(n^3)$ [Imai, Iri '88]
- $O(n^2 \log n)$ [Melkman, O'Rourke '88]
- $O(n^2)$ time [Chan, Chin '96]

… under the local Fréchet distance:

- $O(n^3)$ [Godau '91]
- $O(n^2 \log n)$

L_1 norm

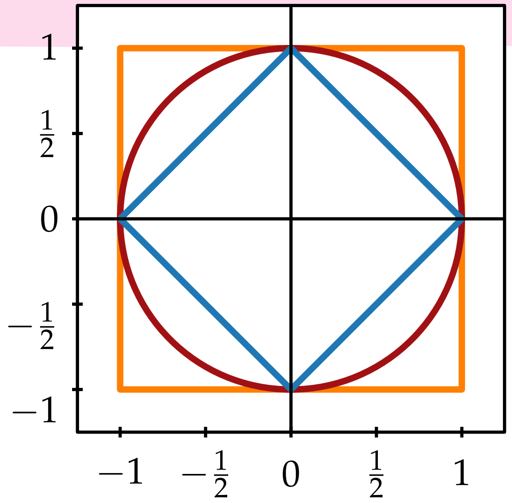
- $O(n^2)$

L_∞ norm

- $O(n^2)$

our contribution

L_p:
$p = \infty$
$p = 2$
$p = 1$



Related Work & Contribution

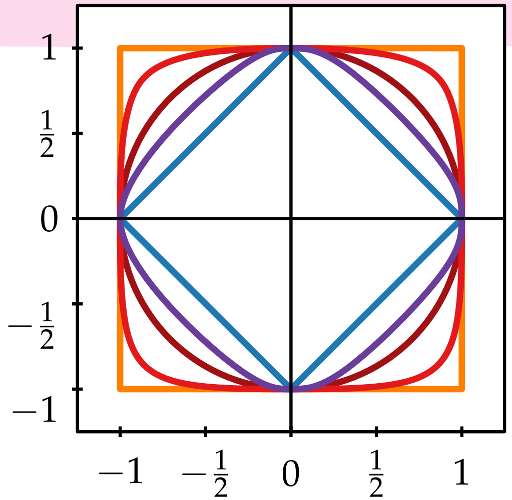
Running time for finding an optimal polyline simplification …

minimum size

… under the local Hausdorff distance:		… under the local Fréchet distance:
L_2 norm		
■ $O(n^3)$ [Imai, Iri '88]		■ $O(n^3)$ [Godau '91]
■ $O(n^2 \log n)$ [Melkman, O'Rourke '88]		■ $O(n^2 \log n)$
■ $O(n^2)$ time [Chan, Chin '96]	**L_1 norm**	■ $O(n^2)$
	L_∞ norm	■ $O(n^2)$
	$L_{p\in(1,\infty)}$ norm	■ $O(n^2 \log n)$

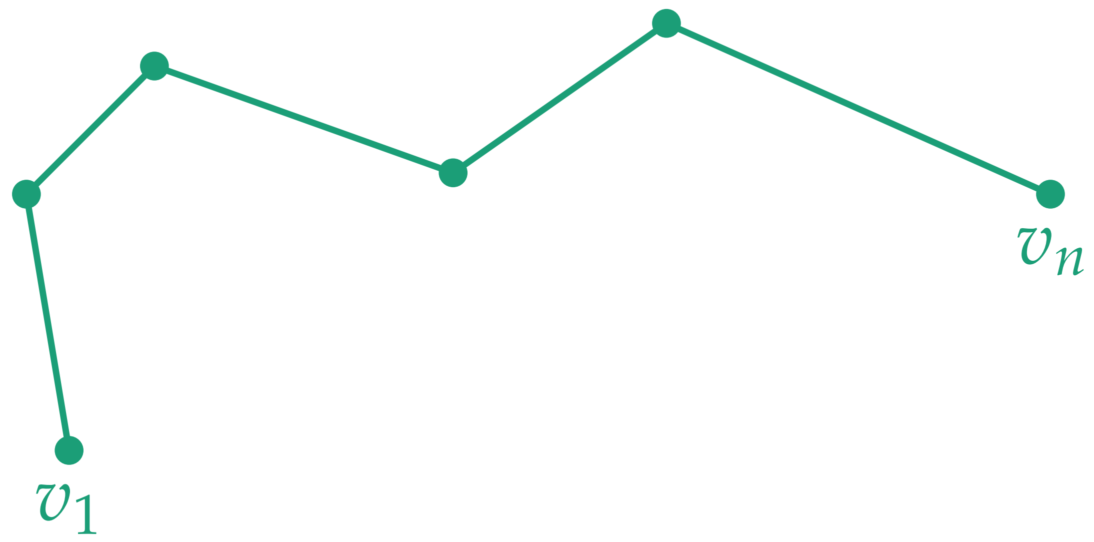
our contribution

L_p:
$p = \infty$
$p = 4$
$p = 2$
$p = 1.5$
$p = 1$



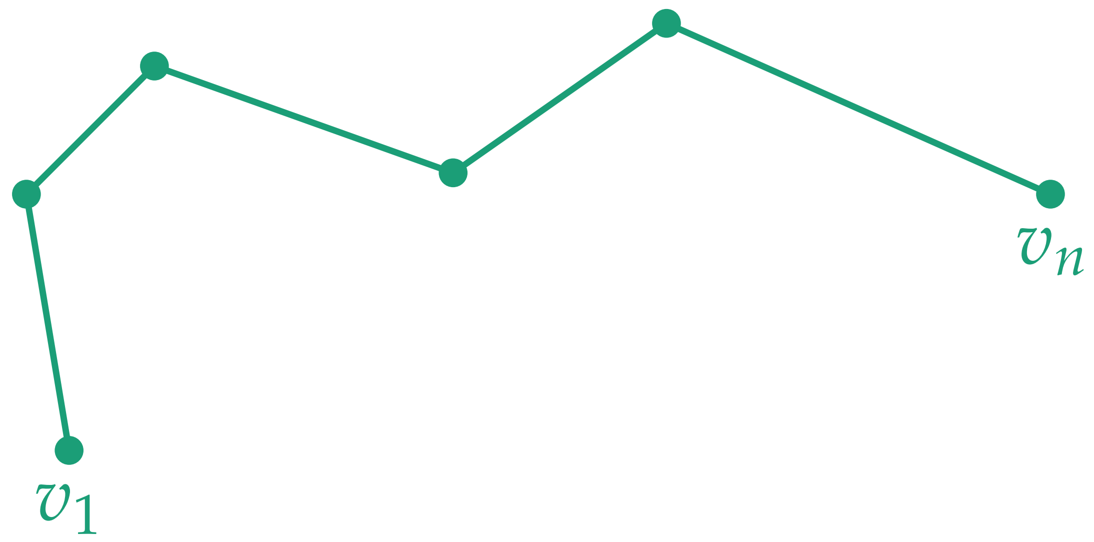
Algorithm by Imai and Iri

- proceeds in two phases



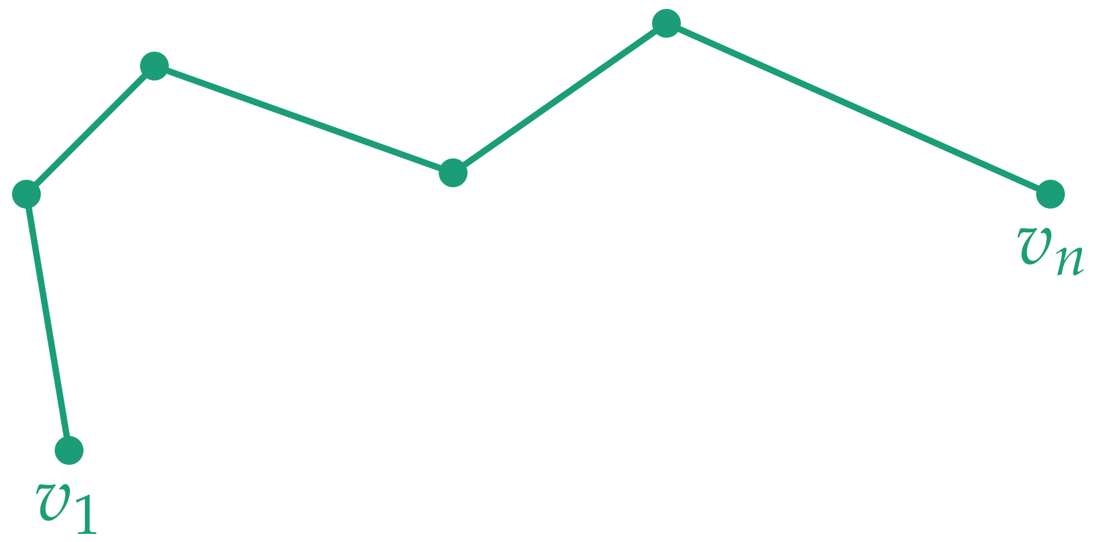
Algorithm by Imai and Iri

- proceeds in two phases
- First phase:



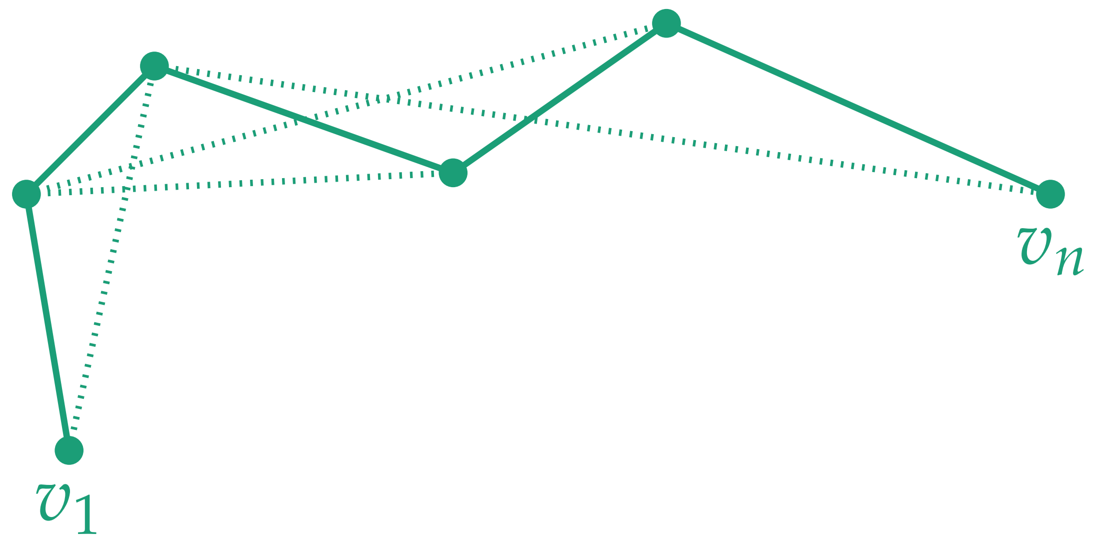
Algorithm by Imai and Iri

- proceeds in two phases
- First phase:
  - determine valid shortcuts brute-force



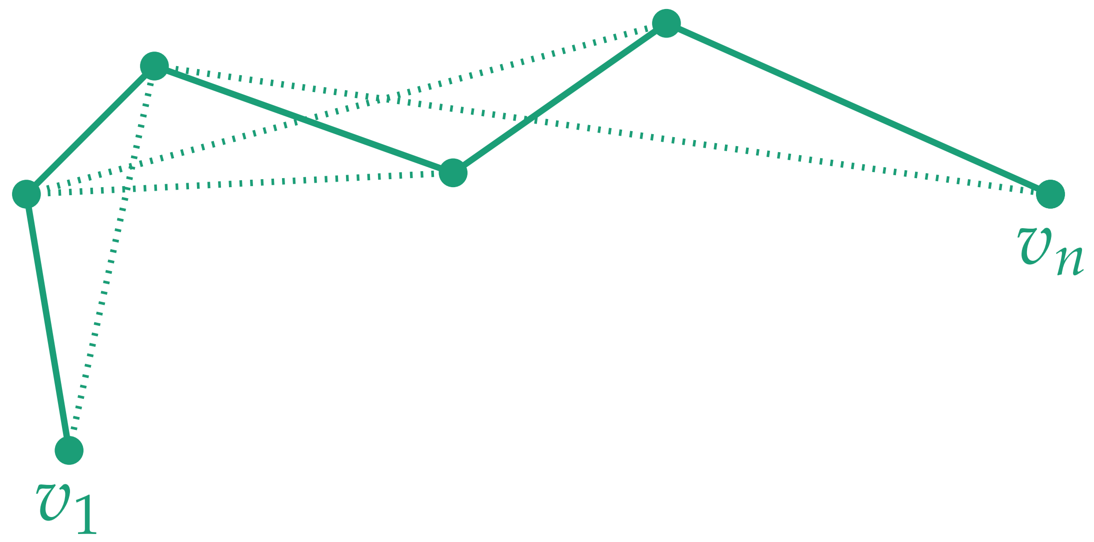
Algorithm by Imai and Iri

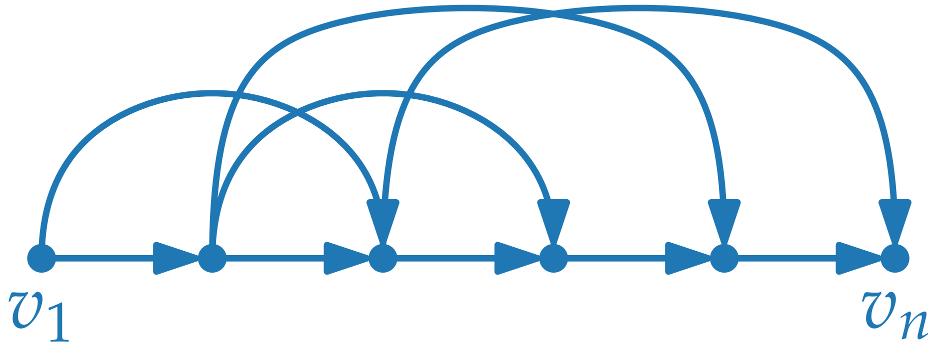
- proceeds in two phases
- First phase:
  - determine valid shortcuts brute-force



Algorithm by Imai and Iri

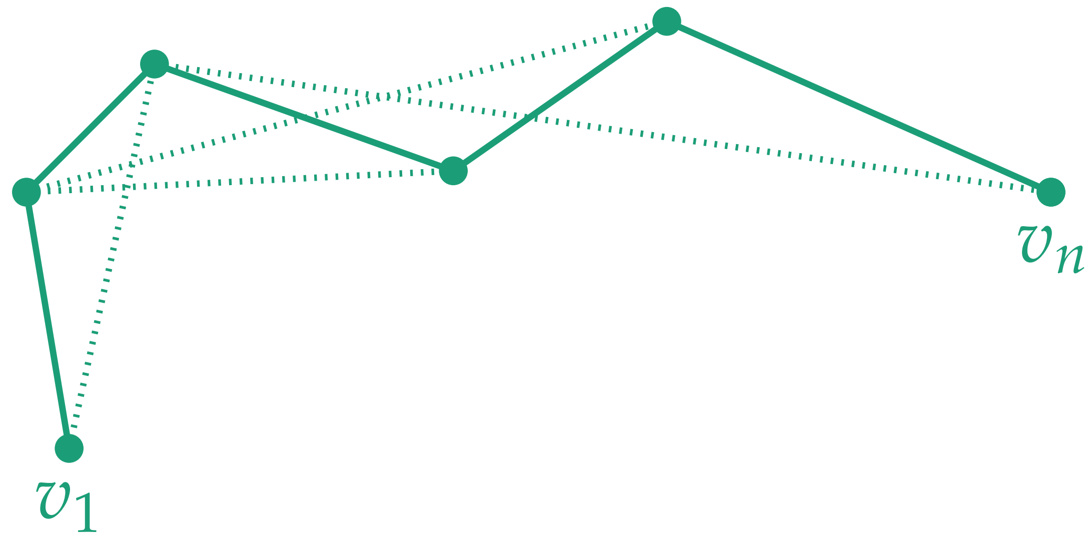
- proceeds in two phases
- First phase:
  - determine valid shortcuts brute-force
  - build shortcut graph

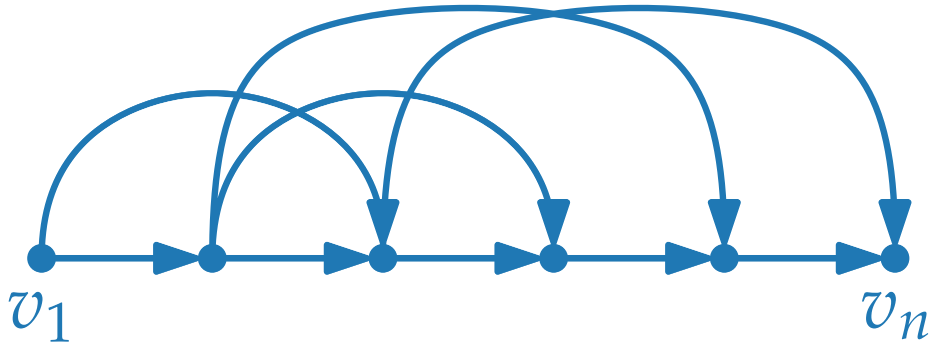




Algorithm by Imai and Iri

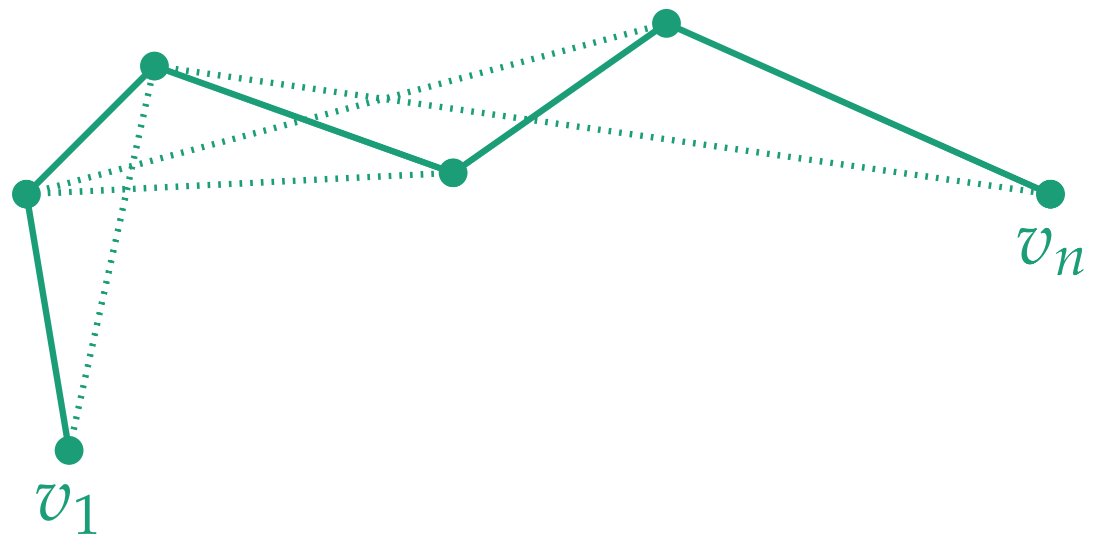
- proceeds in two phases
- First phase:
  - determine valid shortcuts brute-force
  - build shortcut graph
- Second phase:

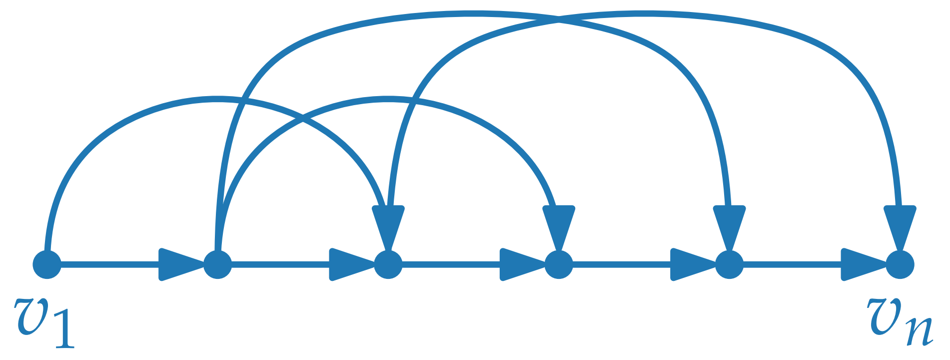




Algorithm by Imai and Iri

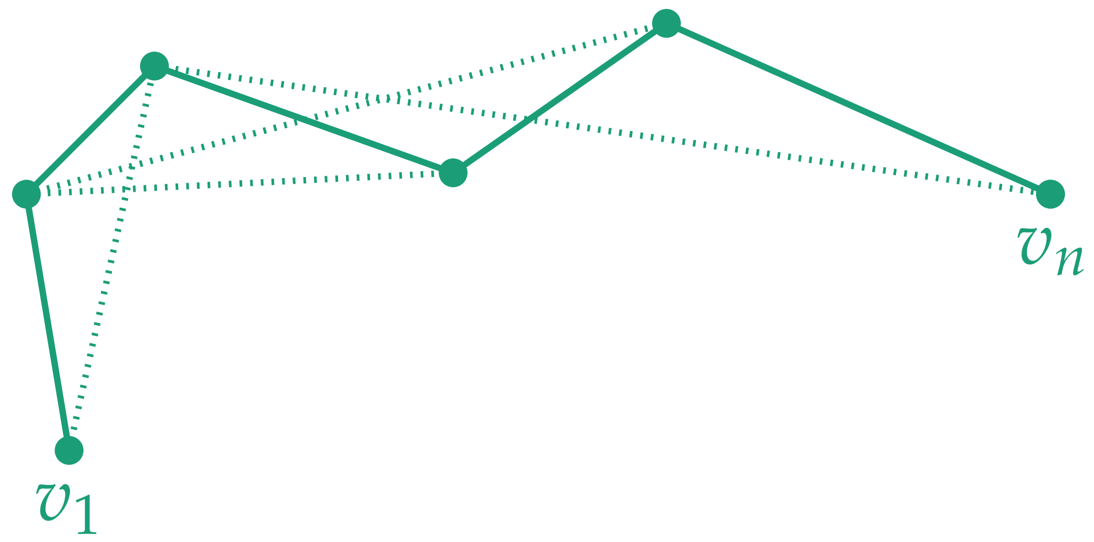
- proceeds in two phases
- First phase:
  - determine valid shortcuts brute-force
  - build shortcut graph
- Second phase:
  - find shortest path in shortcut graph

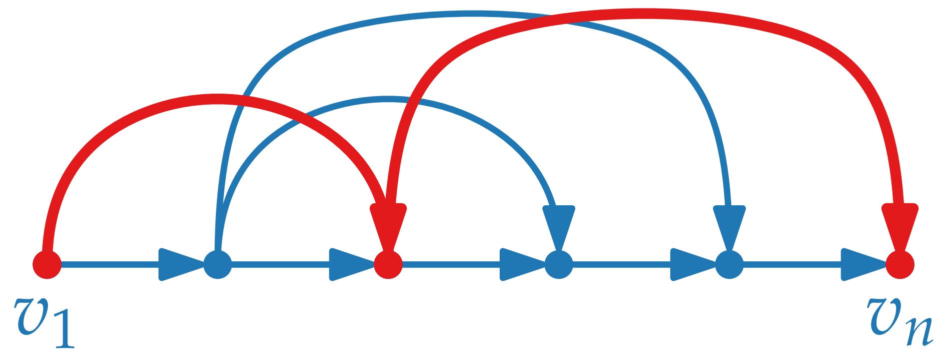




Algorithm by Imai and Iri

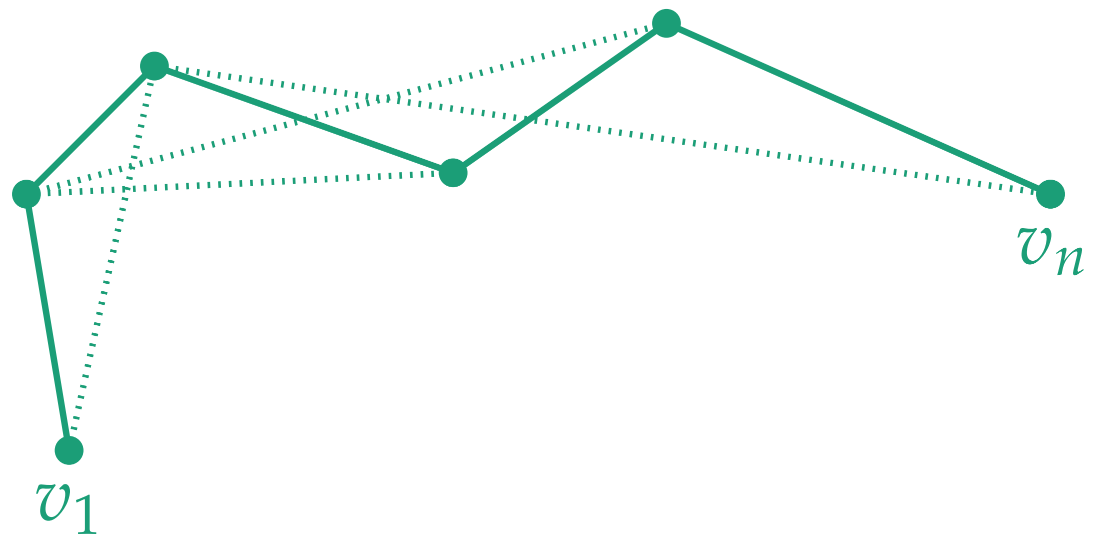
- proceeds in two phases
- First phase:
  - determine valid shortcuts brute-force
  - build shortcut graph
- Second phase:
  - find shortest path in shortcut graph

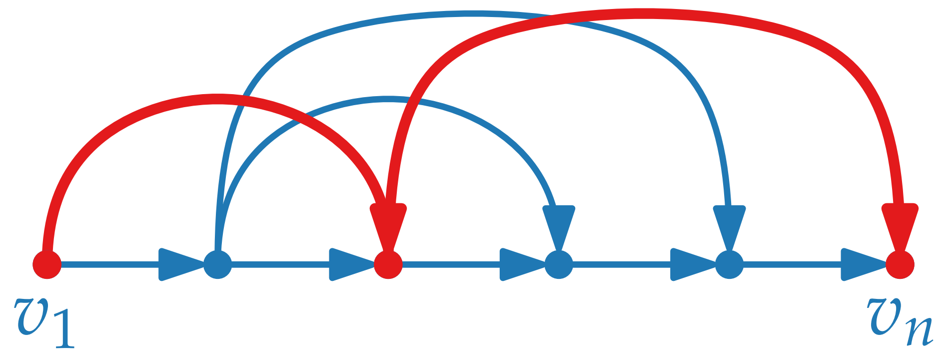




Algorithm by Imai and Iri

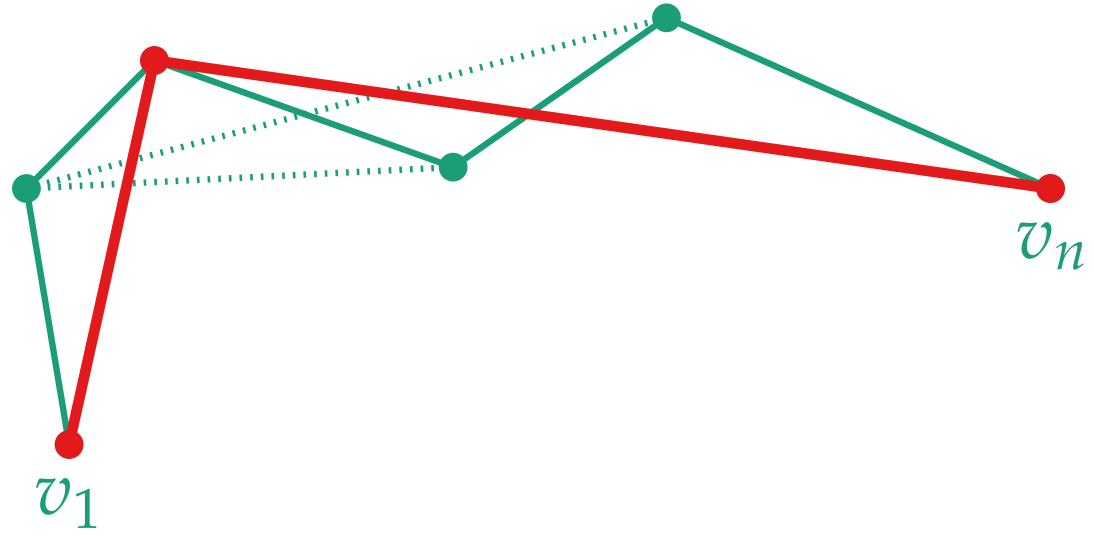
- proceeds in two phases
- First phase:
  - determine valid shortcuts brute-force
  - build shortcut graph
- Second phase:
  - find shortest path in shortcut graph
  - return optimal simplification

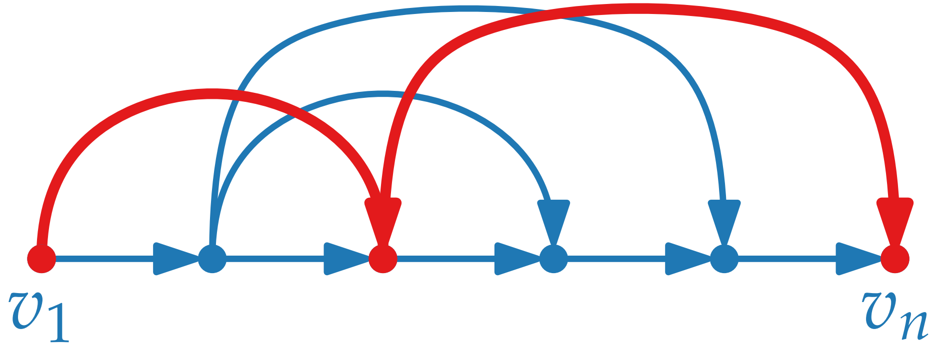




Algorithm by Imai and Iri

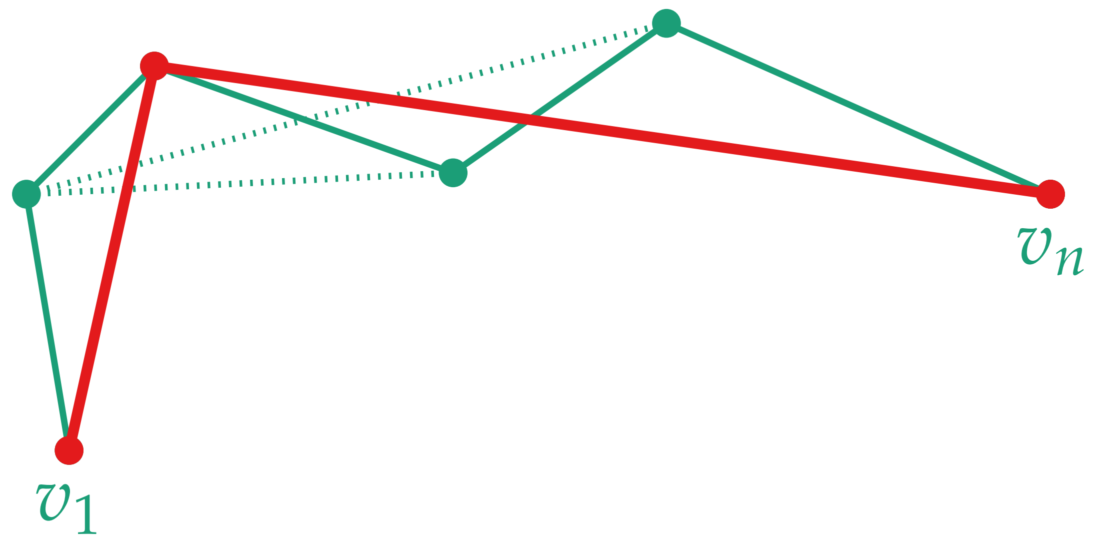
- proceeds in two phases
- First phase:
  - determine valid shortcuts brute-force
  - build shortcut graph
- Second phase:
  - find shortest path in shortcut graph
  - return optimal simplification

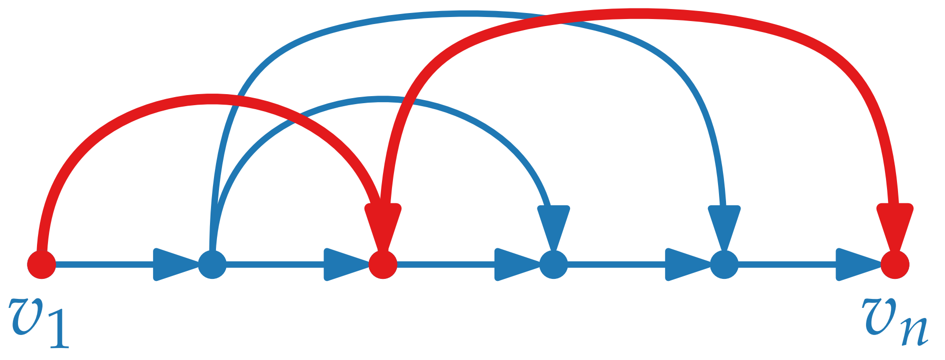




Algorithm by Imai and Iri

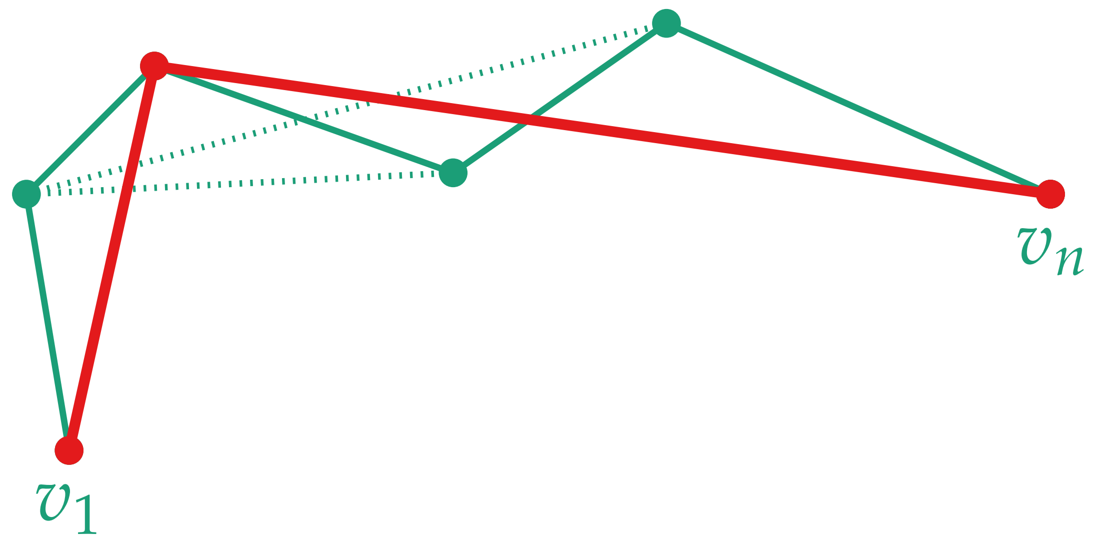
- proceeds in two phases
- First phase: $O(n^3)$
  - determine valid shortcuts brute-force
  - build shortcut graph
- Second phase:
  - find shortest path in shortcut graph
  - return optimal simplification

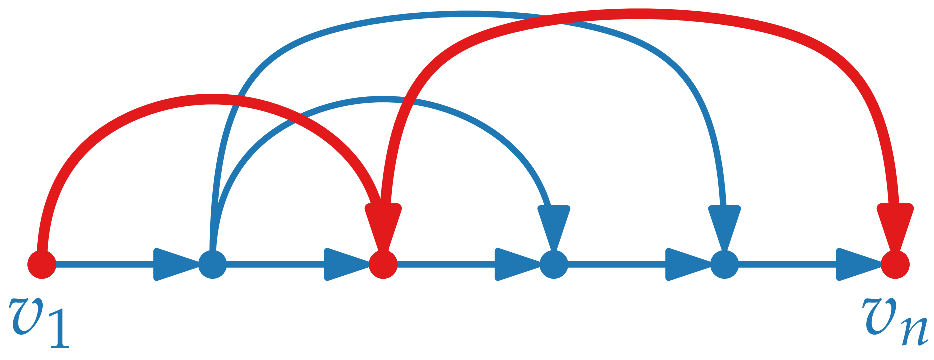




Algorithm by Imai and Iri

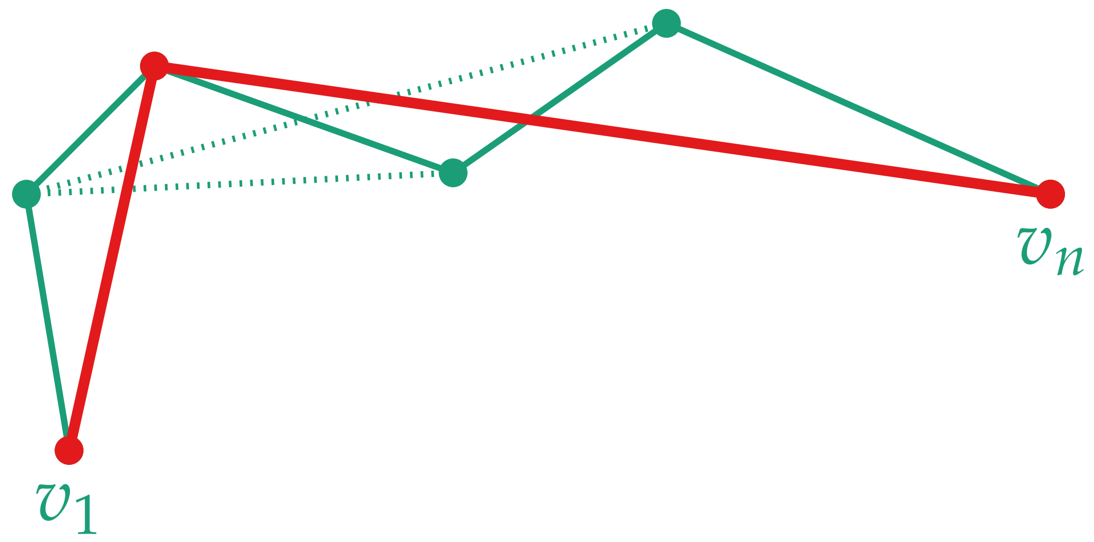
- proceeds in two phases
- First phase: $O(n^3)$
  - determine valid shortcuts brute-force
  - build shortcut graph
- Second phase: $O(n^2)$
  - find shortest path in shortcut graph
  - return optimal simplification

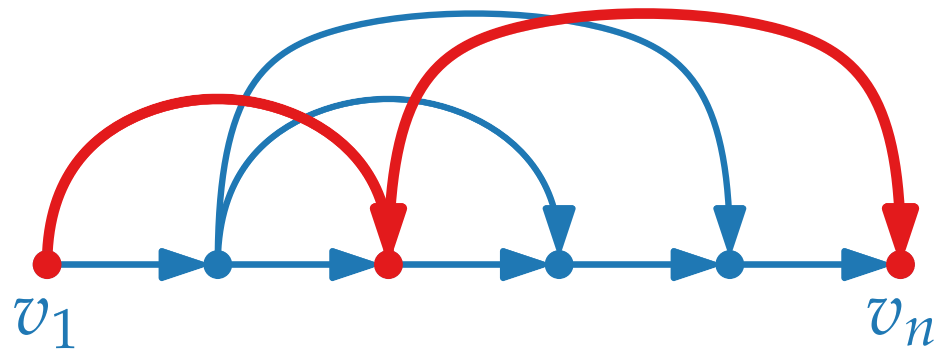




Algorithm by Imai and Iri

- proceeds in two phases
- First phase: $O(n^3)$
  - determine valid shortcuts brute-force
  - build shortcut graph
- Second phase: $O(n^2)$
  - find shortest path in shortcut graph
  - return optimal simplification



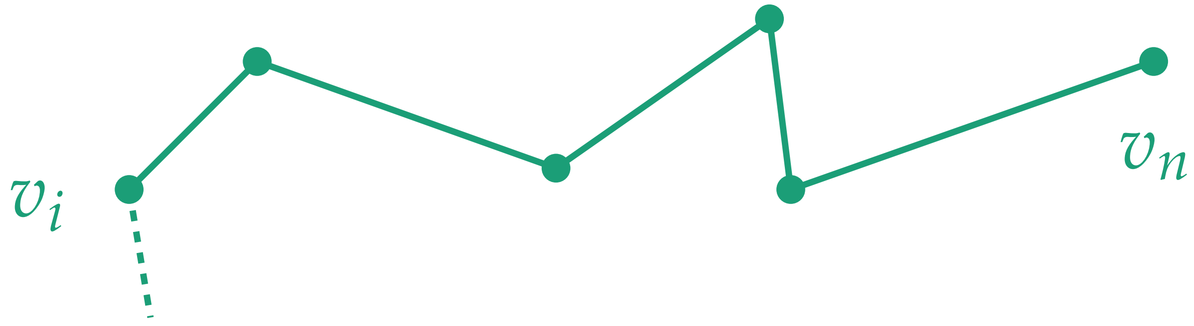


$\Rightarrow$ total running time $O(n^3)$

Algorithm by Melkman & O'Rourke

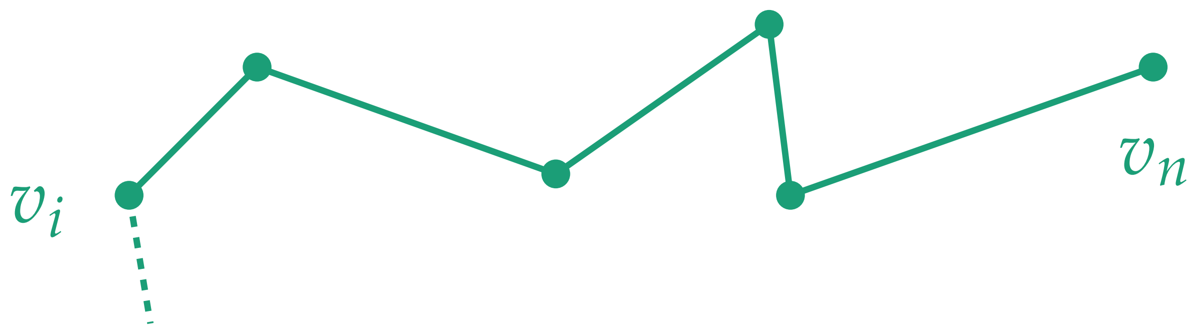
Algorithm by Melkman & O'Rourke

- Starting at each vertex v_i ($i \in \{1, \ldots, n\}$),



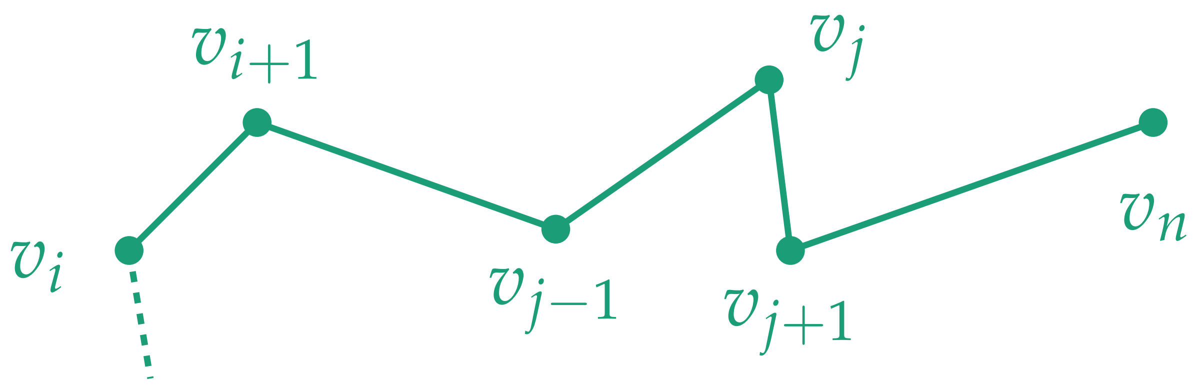
Algorithm by Melkman & O'Rourke

- Starting at each vertex v_i ($i \in \{1, \ldots, n\}$),
    - traverse each subsequent vertex v_j ($j \in \{i+1, \ldots, n\}$)



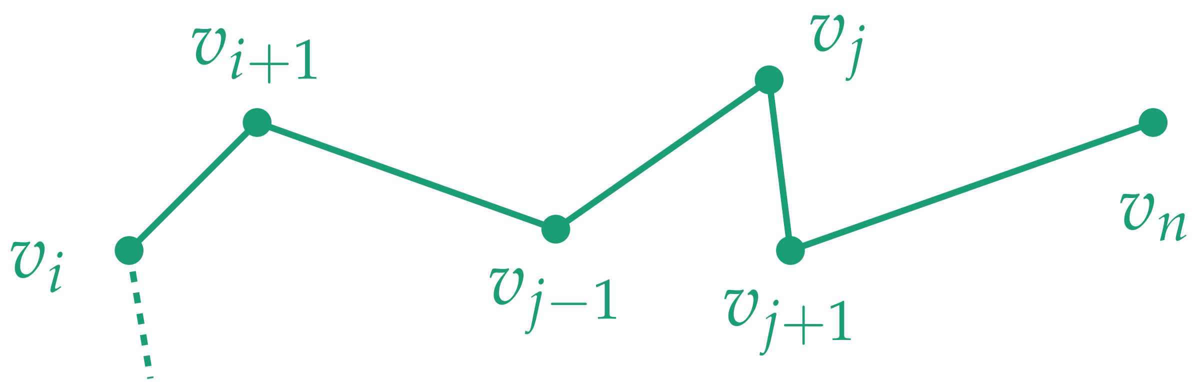
Algorithm by Melkman & O'Rourke

- Starting at each vertex v_i ($i \in \{1, \ldots, n\}$),
  - traverse each subsequent vertex v_j ($j \in \{i+1, \ldots, n\}$)



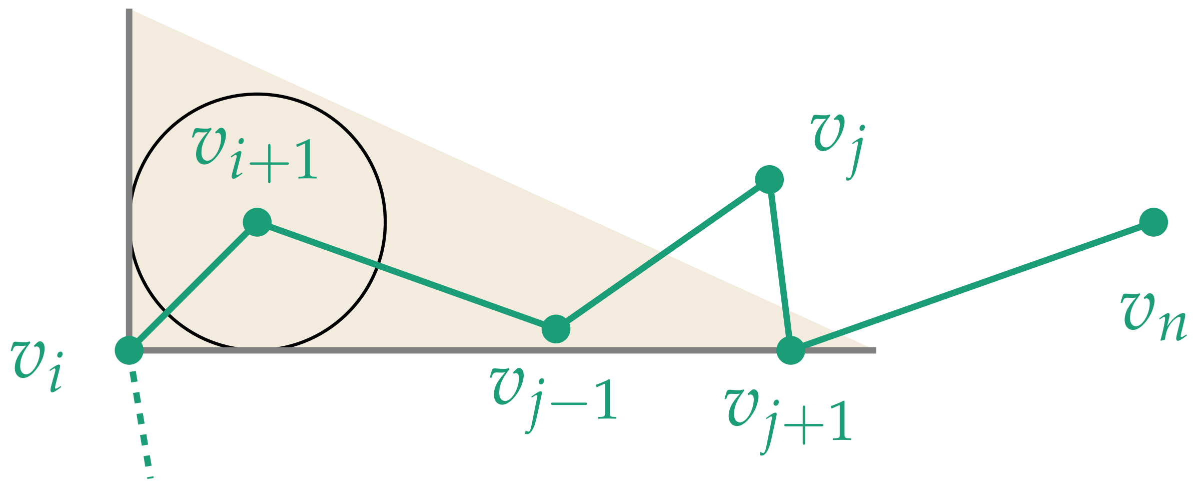
Algorithm by Melkman & O'Rourke

- Starting at each vertex v_i ($i \in \{1, \ldots, n\}$),
  - traverse each subsequent vertex v_j ($j \in \{i+1, \ldots, n\}$)
  - while maintaining a *cone*



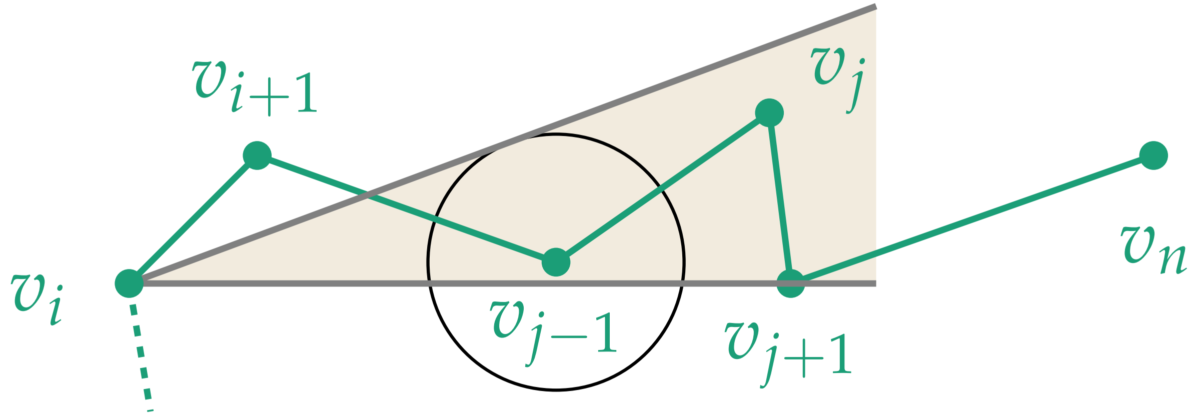
Algorithm by Melkman & O'Rourke

- Starting at each vertex v_i ($i \in \{1, \ldots, n\}$),
  - traverse each subsequent vertex v_j ($j \in \{i+1, \ldots, n\}$)
  - while maintaining a *cone*



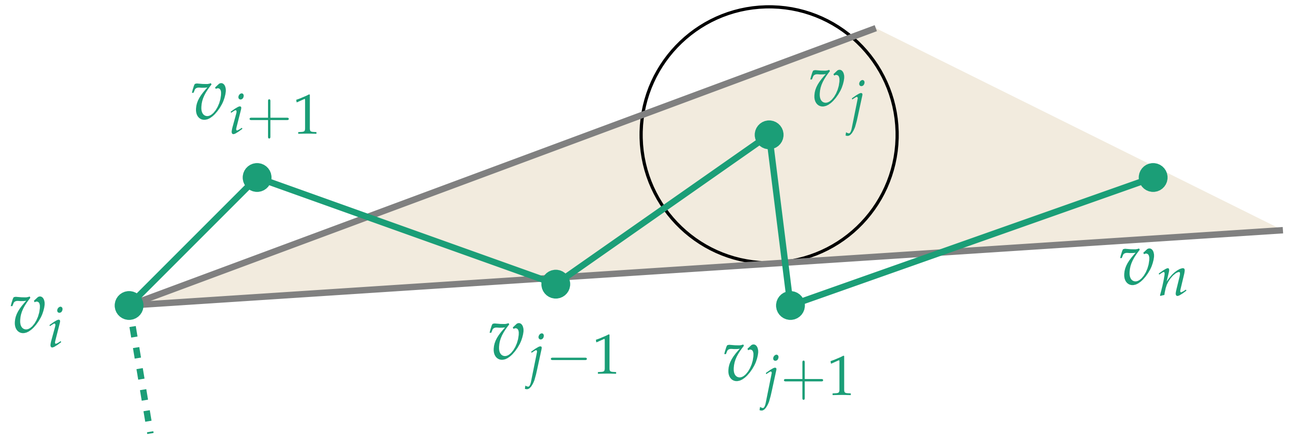
Algorithm by Melkman & O'Rourke

- Starting at each vertex v_i ($i \in \{1, \ldots, n\}$),
  - traverse each subsequent vertex v_j ($j \in \{i+1, \ldots, n\}$)
  - while maintaining a *cone*



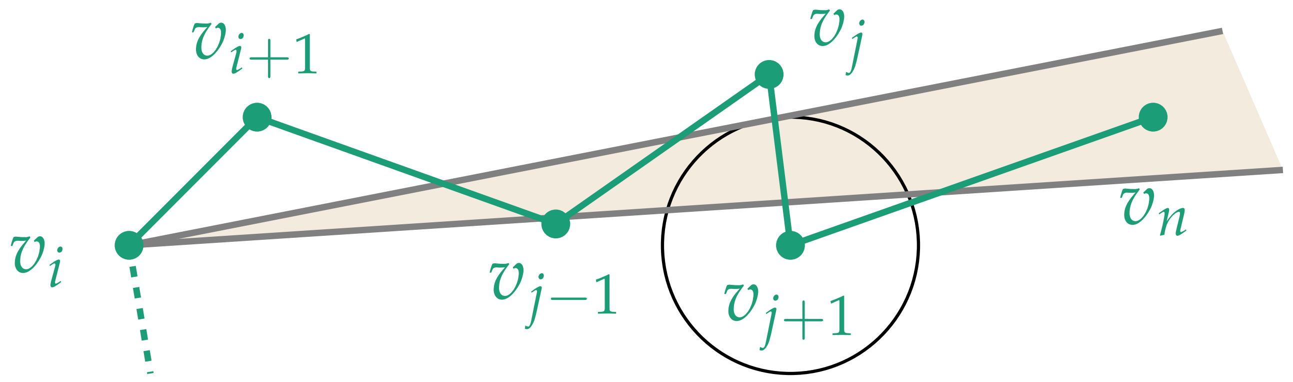
Algorithm by Melkman & O'Rourke

- Starting at each vertex v_i ($i \in \{1, \dots, n\}$),
  - traverse each subsequent vertex v_j ($j \in \{i+1, \dots, n\}$)
  - while maintaining a *cone*



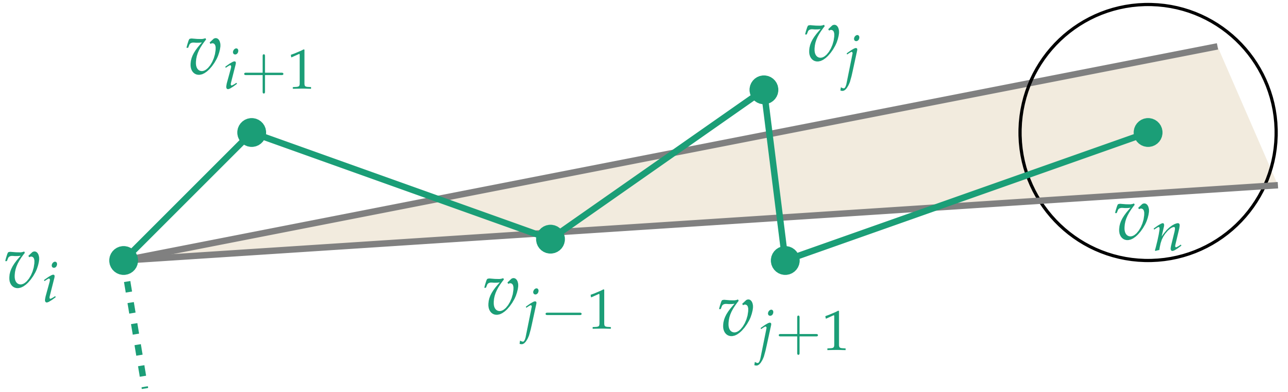
Algorithm by Melkman & O'Rourke

- Starting at each vertex v_i ($i \in \{1, \dots, n\}$),
  - traverse each subsequent vertex v_j ($j \in \{i+1, \dots, n\}$)
  - while maintaining a *cone*



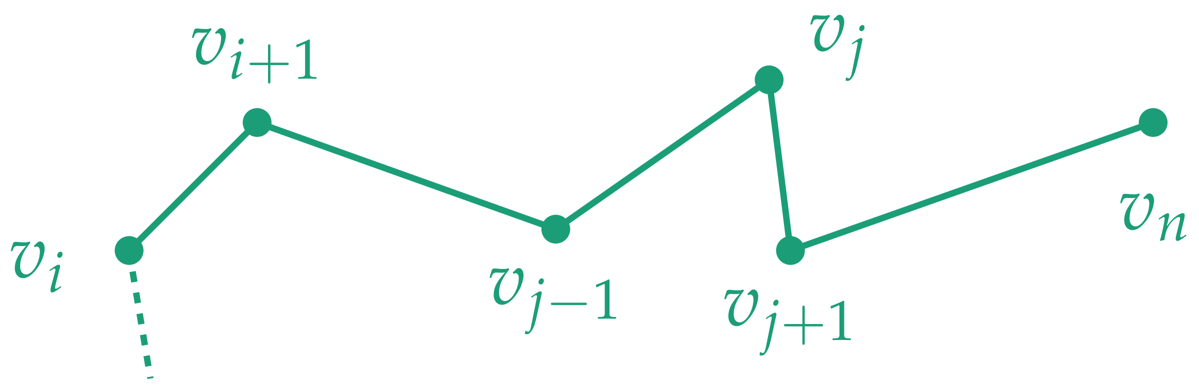
Algorithm by Melkman & O'Rourke

- Starting at each vertex v_i ($i \in \{1, \dots, n\}$),
  - traverse each subsequent vertex v_j ($j \in \{i+1, \dots, n\}$)
  - while maintaining a *cone*



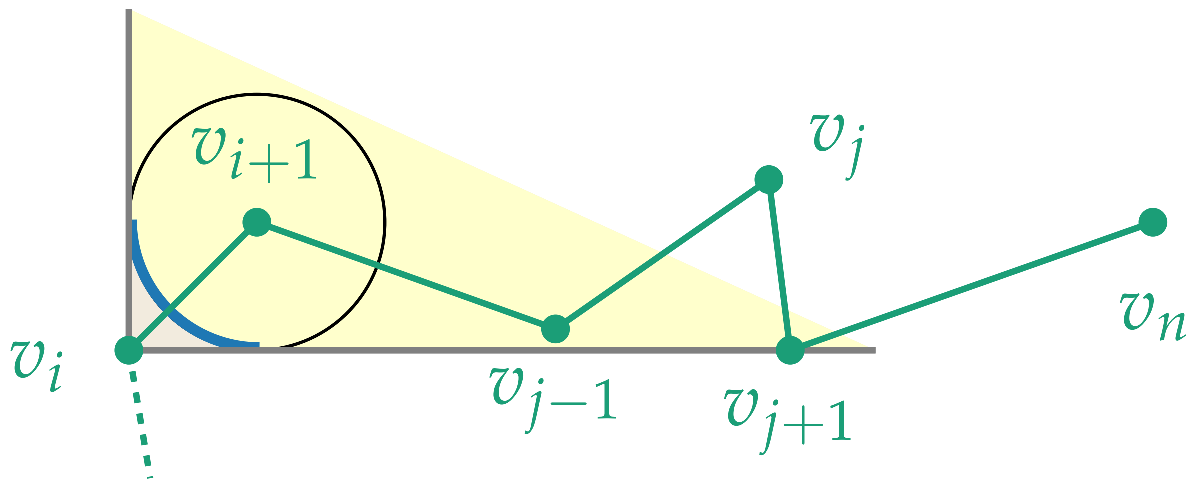
Algorithm by Melkman & O'Rourke

- Starting at each vertex v_i ($i \in \{1, \ldots, n\}$),
  - traverse each subsequent vertex v_j ($j \in \{i+1, \ldots, n\}$)
  - while maintaining a *cone* and a ***wave front***.



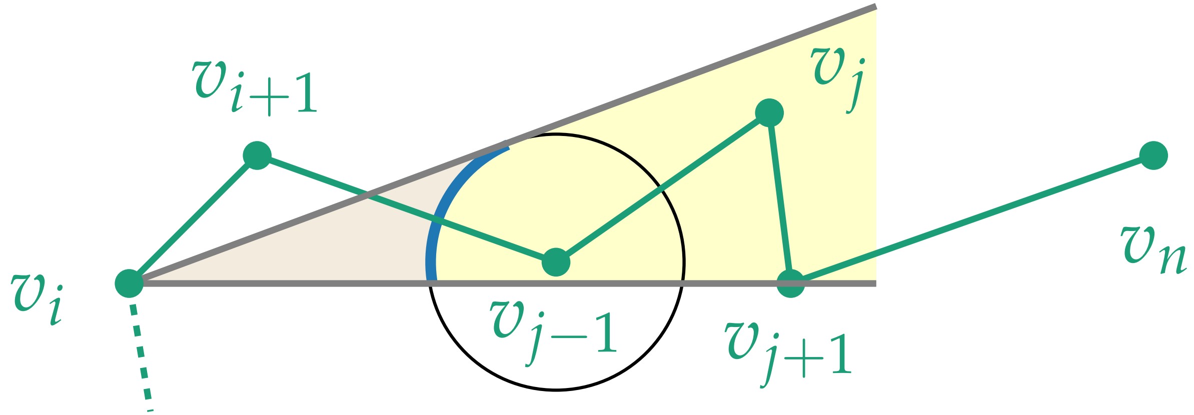
Algorithm by Melkman & O'Rourke

- Starting at each vertex v_i ($i \in \{1, \ldots, n\}$),
  - traverse each subsequent vertex v_j ($j \in \{i+1, \ldots, n\}$)
  - while maintaining a *cone* and a ***wave front***.



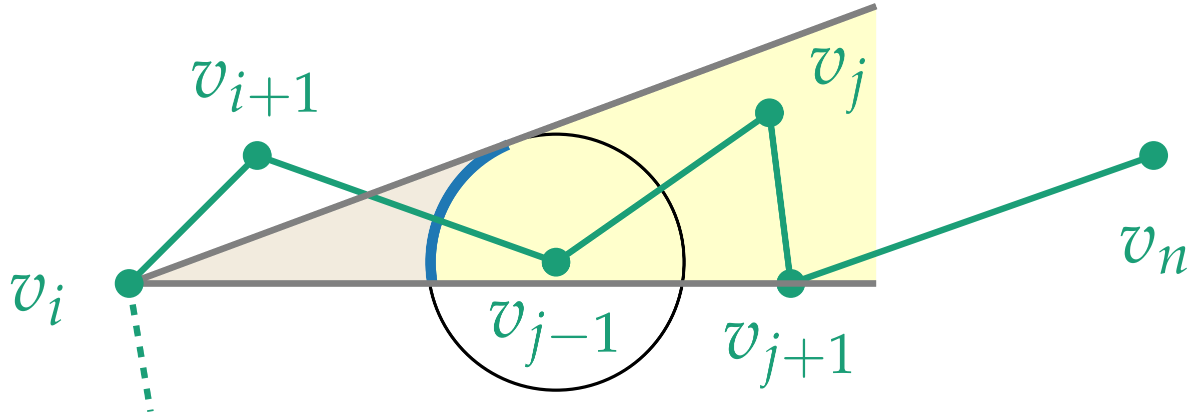
Algorithm by Melkman & O'Rourke

- Starting at each vertex v_i ($i \in \{1, \ldots, n\}$),
    - traverse each subsequent vertex v_j ($j \in \{i+1, \ldots, n\}$)
    - while maintaining a ***cone*** and a ***wave front***.



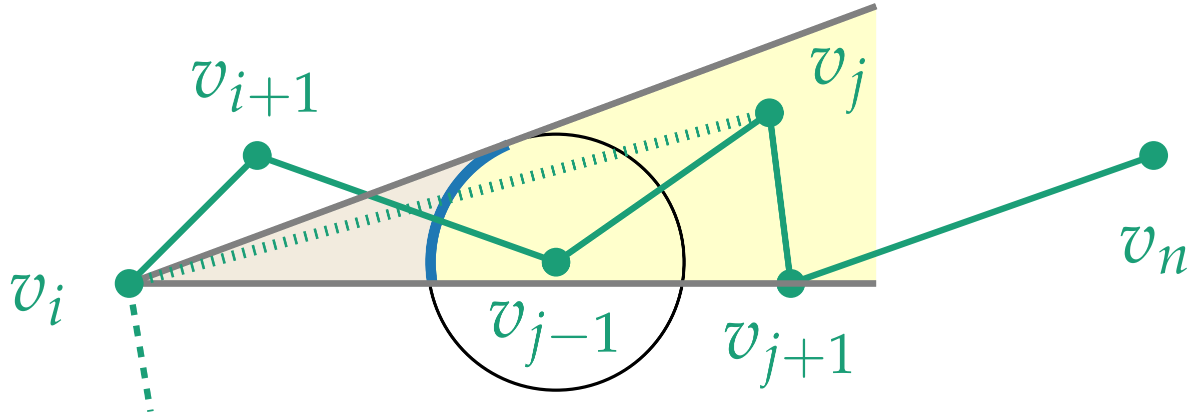
Algorithm by Melkman & O'Rourke

- Starting at each vertex v_i ($i \in \{1, \ldots, n\}$),
  - traverse each subsequent vertex v_j ($j \in \{i+1, \ldots, n\}$)
  - while maintaining a ***cone*** and a ***wave front***.
- $\overline{v_i v_j}$ is a valid shortcut $\Leftrightarrow$ v_j lies in the cone and "behind" the wave front



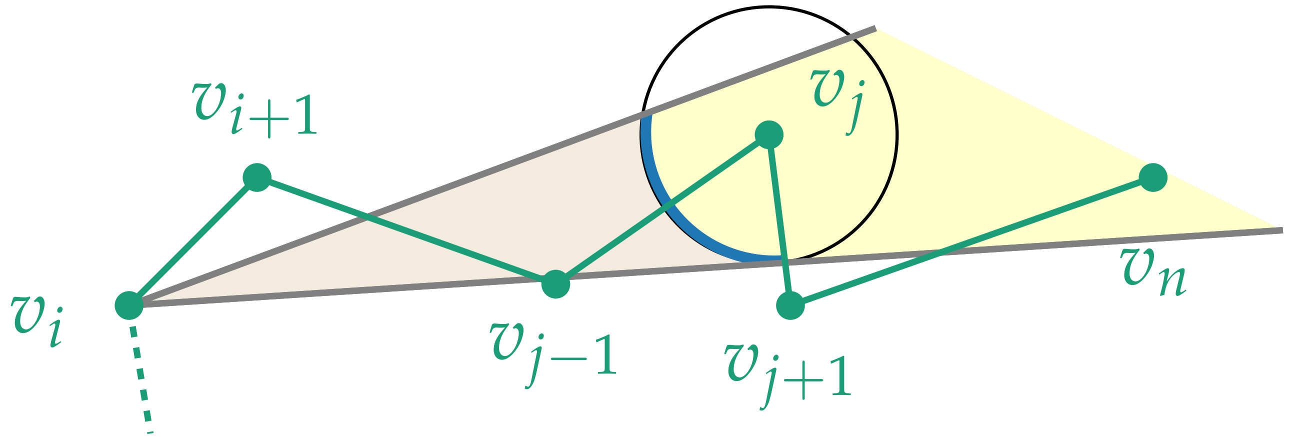
Algorithm by Melkman & O'Rourke

- Starting at each vertex v_i ($i \in \{1, \ldots, n\}$),
  - traverse each subsequent vertex v_j ($j \in \{i+1, \ldots, n\}$)
  - while maintaining a *cone* and a ***wave front***.
- $\overline{v_i v_j}$ is a valid shortcut $\Leftrightarrow$ v_j lies in the cone and "behind" the wave front



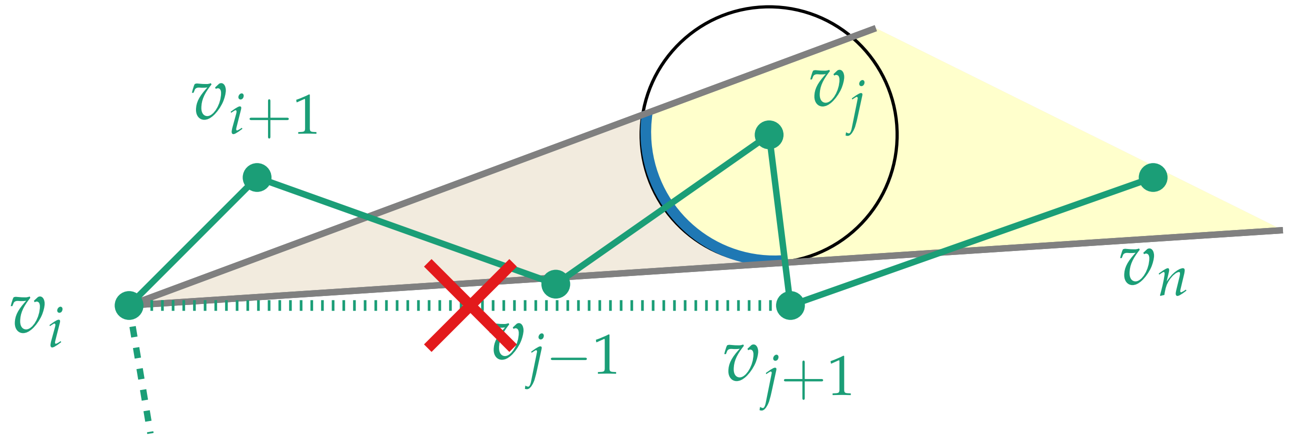
Algorithm by Melkman & O'Rourke

- Starting at each vertex v_i ($i \in \{1, \ldots, n\}$),
  - traverse each subsequent vertex v_j ($j \in \{i+1, \ldots, n\}$)
  - while maintaining a ***cone*** and a ***wave front***.
- $\overline{v_i v_j}$ is a valid shortcut $\Leftrightarrow$ v_j lies in the cone and "behind" the wave front



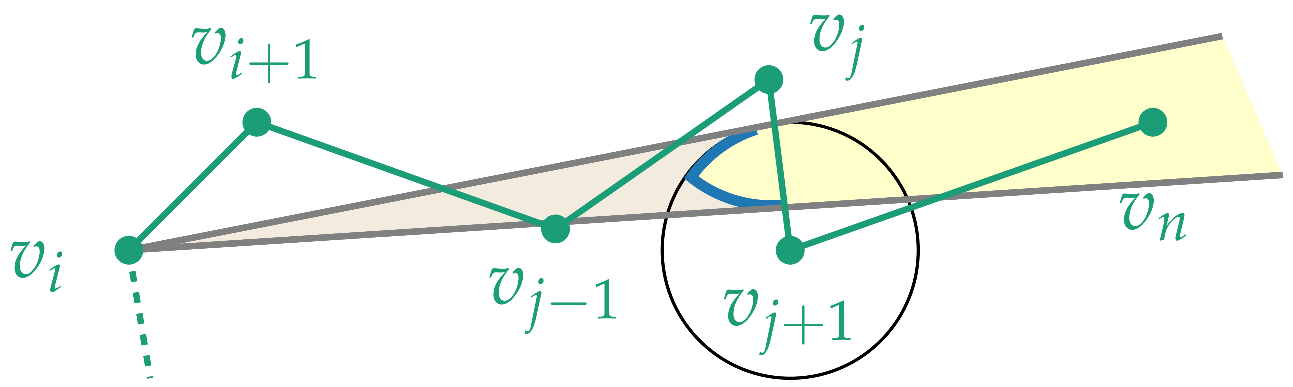
Algorithm by Melkman & O'Rourke

- Starting at each vertex v_i ($i \in \{1, \ldots, n\}$),
  - traverse each subsequent vertex v_j ($j \in \{i+1, \ldots, n\}$)
  - while maintaining a *cone* and a ***wave front***.
- $\overline{v_i v_j}$ is a valid shortcut $\Leftrightarrow$ v_j lies in the cone and "behind" the wave front



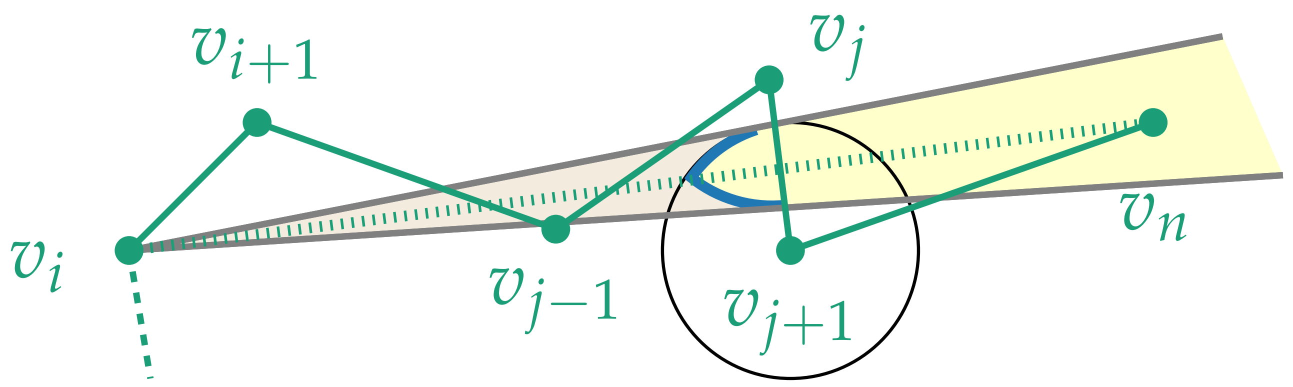
Algorithm by Melkman & O'Rourke

- Starting at each vertex v_i ($i \in \{1, \ldots, n\}$),
  - traverse each subsequent vertex v_j ($j \in \{i+1, \ldots, n\}$)
  - while maintaining a ***cone*** and a ***wave front***.
- $\overline{v_i v_j}$ is a valid shortcut $\Leftrightarrow$ v_j lies in the cone and "behind" the wave front



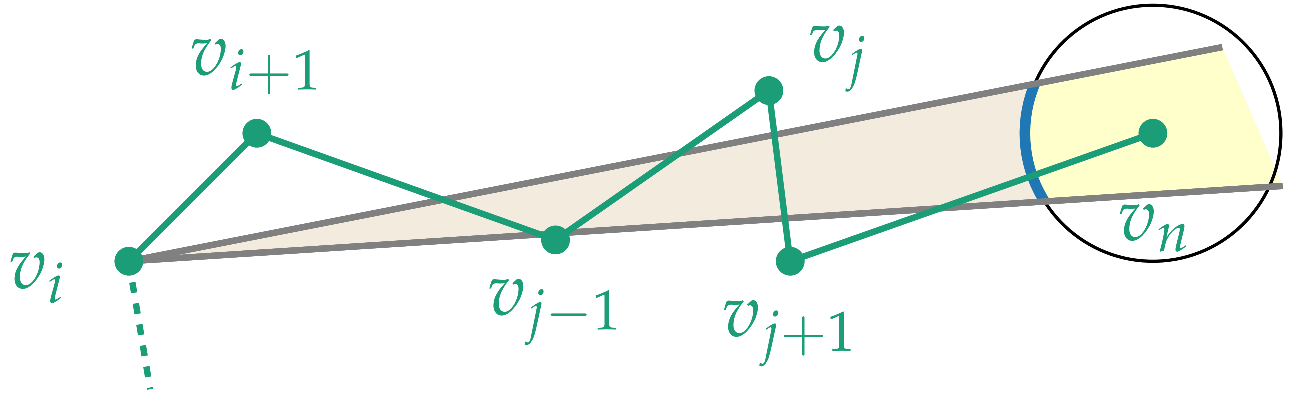
Algorithm by Melkman & O'Rourke

- Starting at each vertex v_i ($i \in \{1, \ldots, n\}$),
  - traverse each subsequent vertex v_j ($j \in \{i+1, \ldots, n\}$)
  - while maintaining a ***cone*** and a ***wave front***.
- $\overline{v_i v_j}$ is a valid shortcut $\Leftrightarrow$ v_j lies in the cone and "behind" the wave front



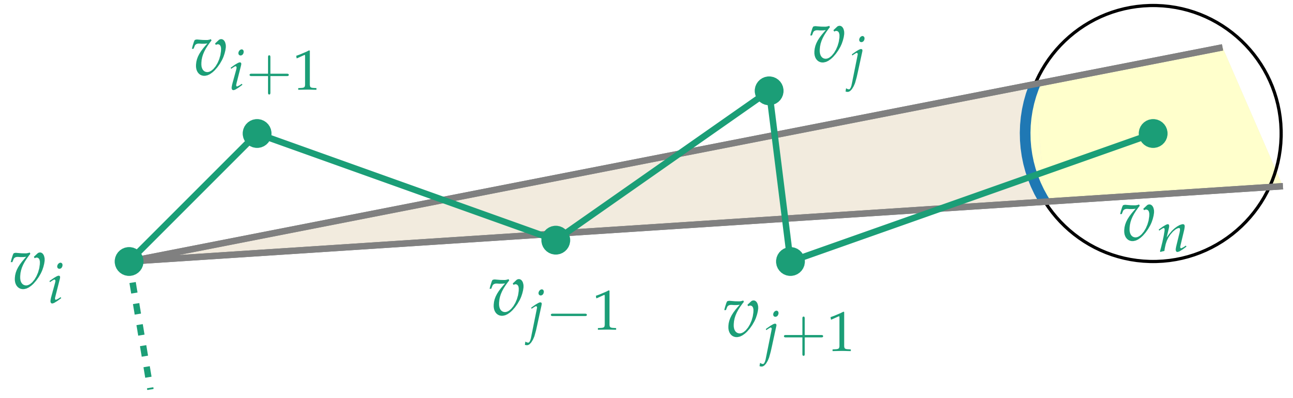
Algorithm by Melkman & O'Rourke

- Starting at each vertex v_i ($i \in \{1, \dots, n\}$),
  - traverse each subsequent vertex v_j ($j \in \{i+1, \dots, n\}$)
  - while maintaining a *cone* and a ***wave front***.
- $\overline{v_i v_j}$ is a valid shortcut $\Leftrightarrow$ v_j lies in the cone and "behind" the wave front



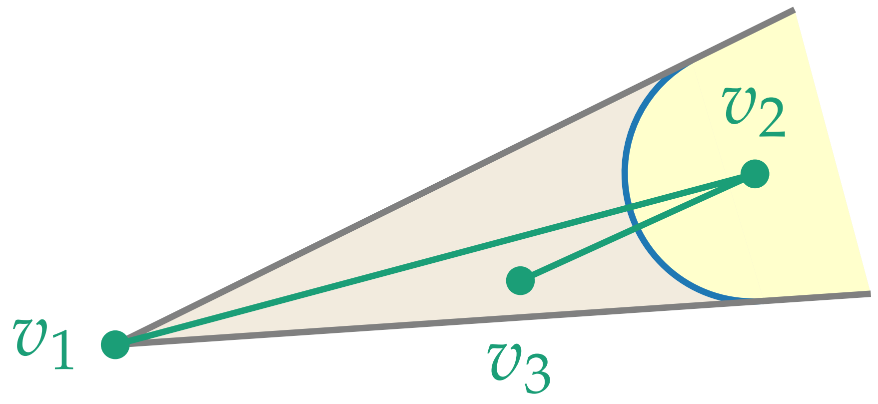
Algorithm by Melkman & O'Rourke

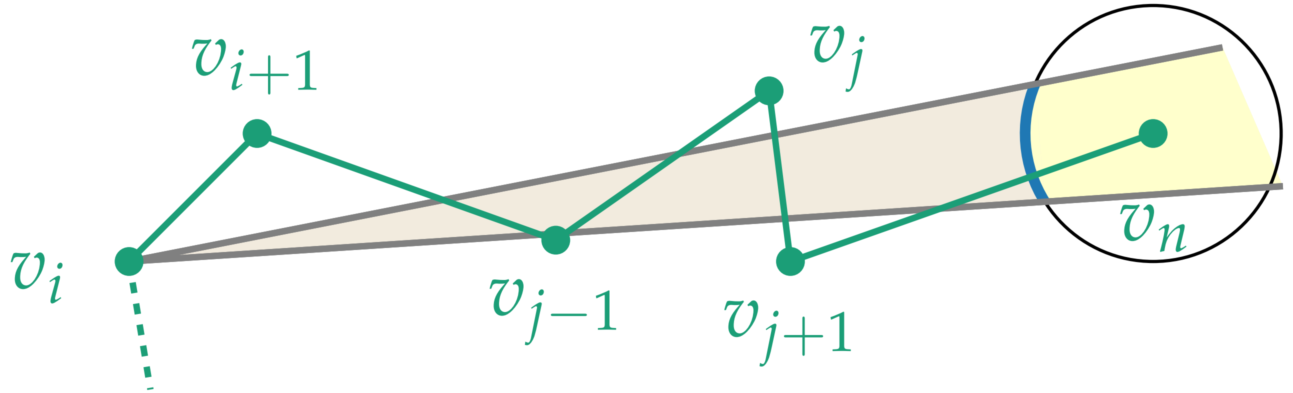
- Starting at each vertex v_i ($i \in \{1, \ldots, n\}$),
  - traverse each subsequent vertex v_j ($j \in \{i+1, \ldots, n\}$)
  - while maintaining a ***cone*** and a ***wave front***.
- $\overline{v_i v_j}$ is a valid shortcut $\Leftrightarrow$ v_j lies in the cone and "behind" the wave front
- Why is the wave front used at all?



Algorithm by Melkman & O'Rourke

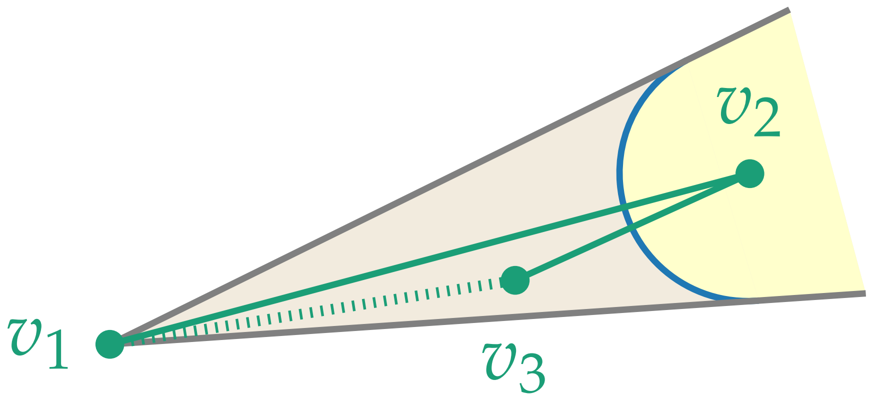
- Starting at each vertex v_i ($i \in \{1, \ldots, n\}$),
  - traverse each subsequent vertex v_j ($j \in \{i+1, \ldots, n\}$)
  - while maintaining a *cone* and a ***wave front***.
- $\overline{v_i v_j}$ is a valid shortcut $\Leftrightarrow$ v_j lies in the cone and "behind" the wave front
- Why is the wave front used at all?

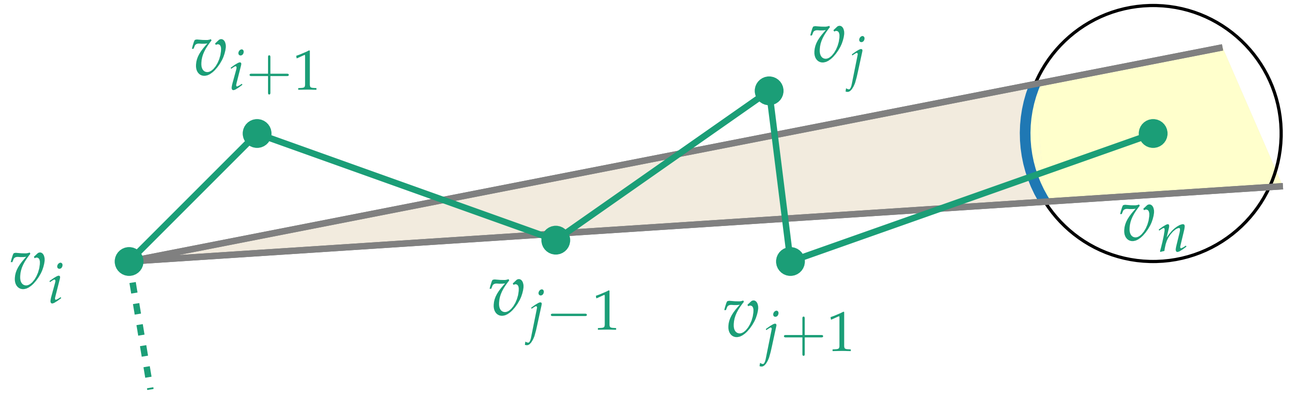




Algorithm by Melkman & O'Rourke

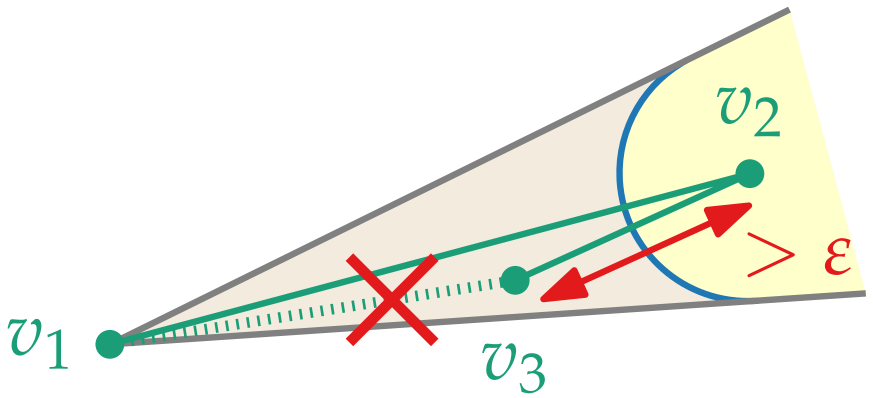
- Starting at each vertex v_i ($i \in \{1, \ldots, n\}$),
  - traverse each subsequent vertex v_j ($j \in \{i+1, \ldots, n\}$)
  - while maintaining a *cone* and a ***wave front***.
- $\overline{v_i v_j}$ is a valid shortcut $\Leftrightarrow$ v_j lies in the cone and "behind" the wave front
- Why is the wave front used at all?

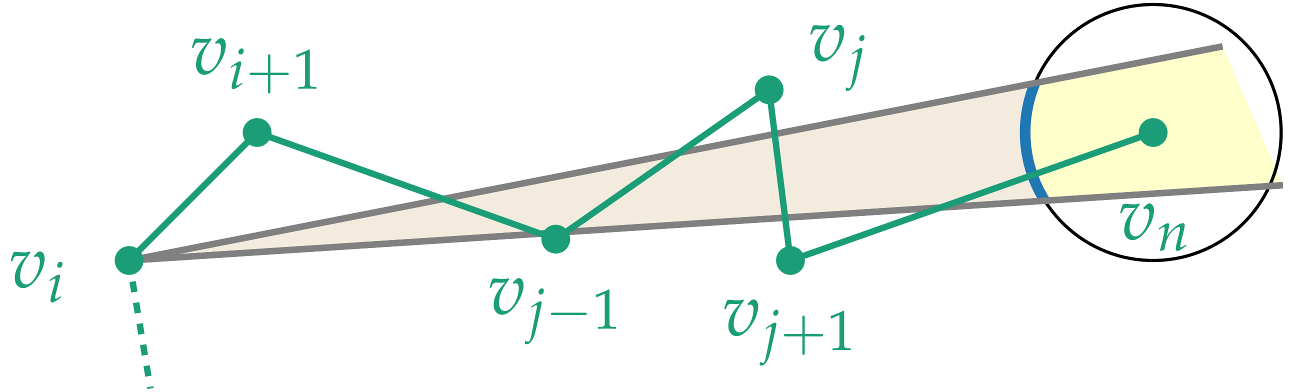




Algorithm by Melkman & O'Rourke

- Starting at each vertex v_i ($i \in \{1, \ldots, n\}$),
  - traverse each subsequent vertex v_j ($j \in \{i+1, \ldots, n\}$)
  - while maintaining a *cone* and a ***wave front***.
- $\overline{v_i v_j}$ is a valid shortcut $\Leftrightarrow$ v_j lies in the cone and "behind" the wave front
- Why is the wave front used at all?





The Wave Front

- is a sequence of circular arcs.

The Wave Front

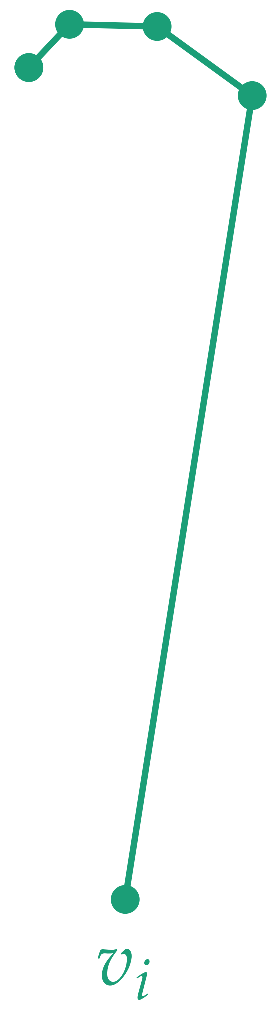
- is a sequence of circular arcs.

How complex can the wave front be?

The Wave Front

- is a sequence of circular arcs.

How complex can the wave front be?

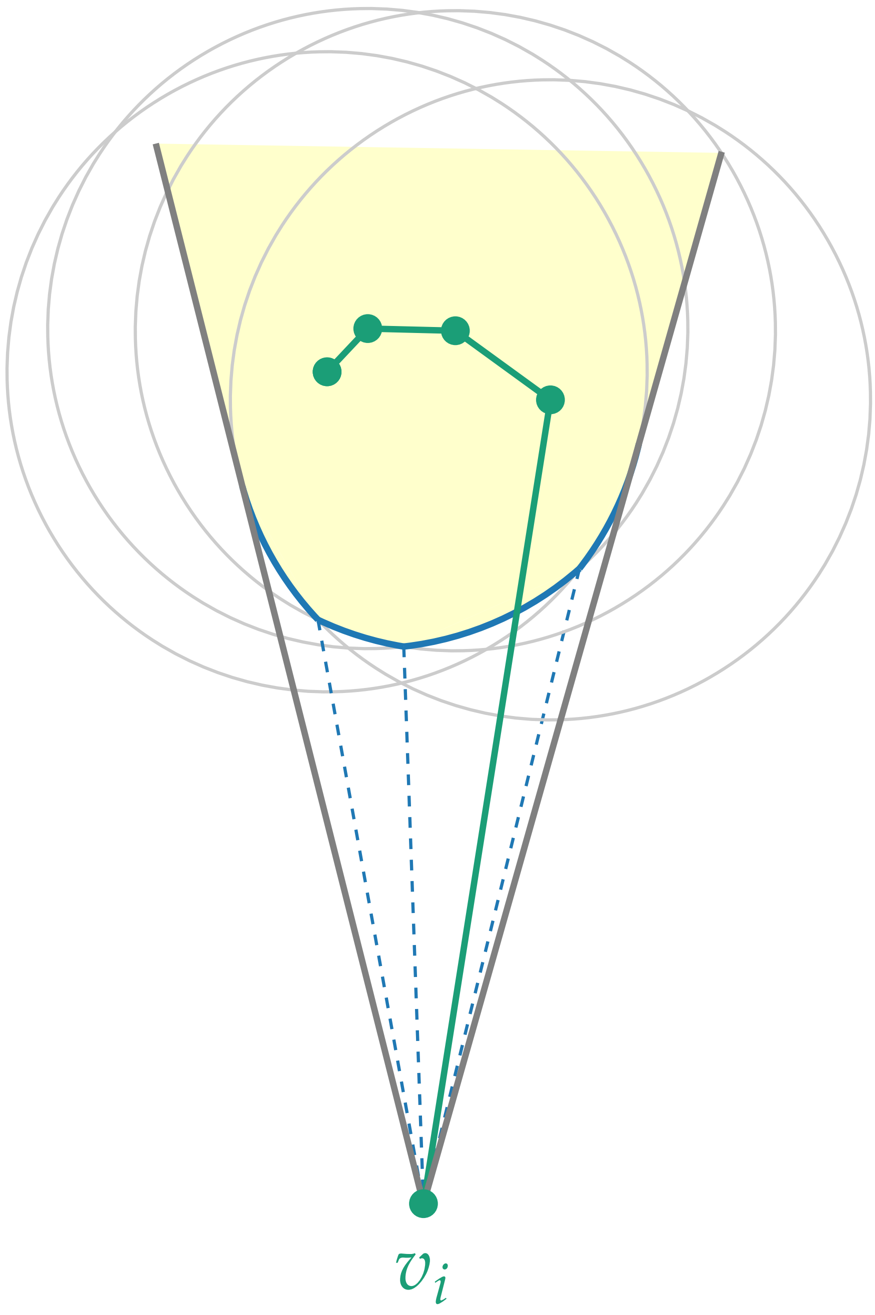


The Wave Front

- is a sequence of circular arcs.

How complex can the wave front be?

- each vertex contributes ≤ 1 arc $\Rightarrow O(n)$ arcs



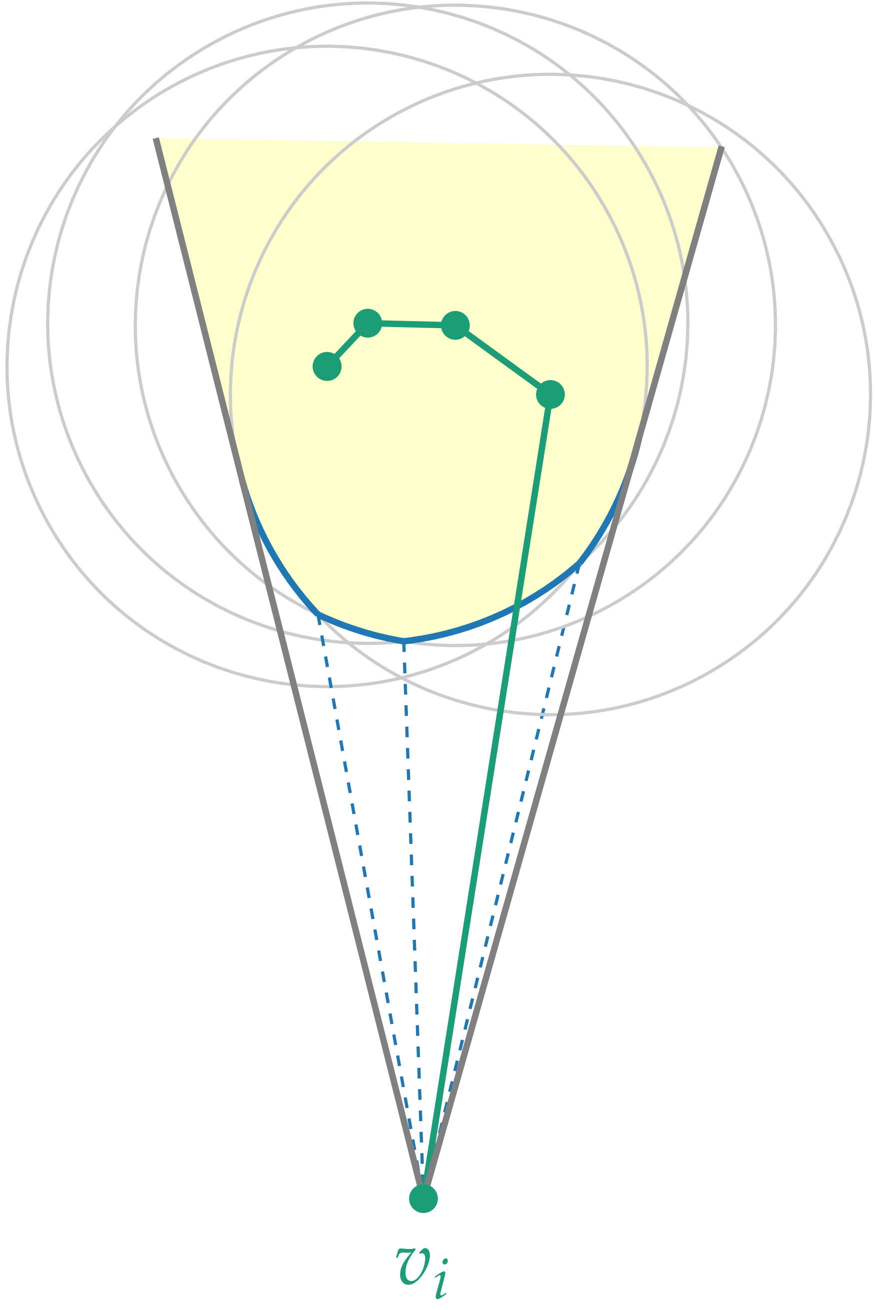
The Wave Front

- is a sequence of circular arcs.

How complex can the wave front be?

- each vertex contributes ≤ 1 arc $\Rightarrow O(n)$ arcs

$\Rightarrow$ stored in a balanced search tree



The Wave Front

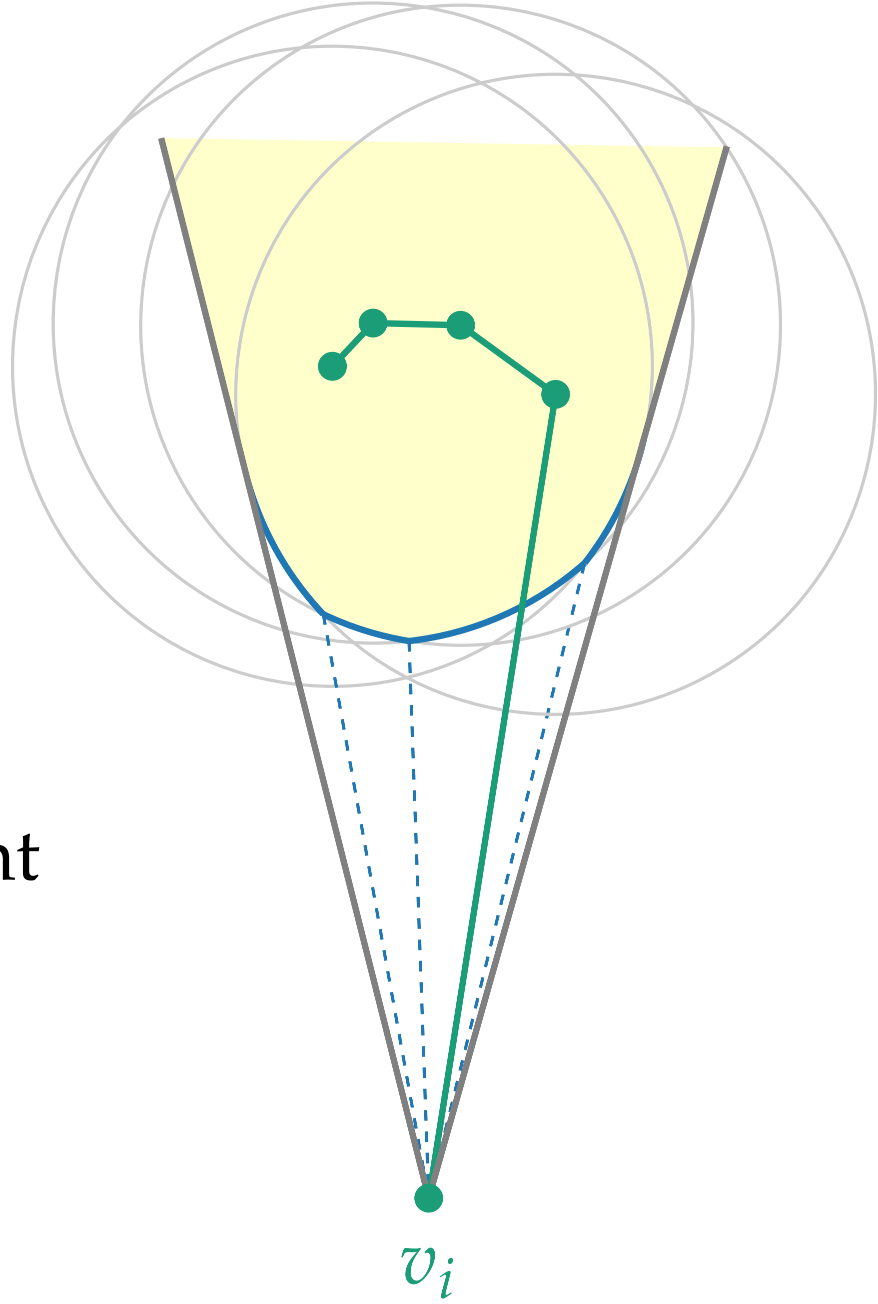
- is a sequence of circular arcs.

How complex can the wave front be?

- each vertex contributes ≤ 1 arc $\Rightarrow O(n)$ arcs

$\Rightarrow$ stored in a balanced search tree

  - finding the intersection between the wave front and a ray emanating at v_i takes $O(\log n)$ time



The Wave Front

- is a sequence of circular arcs.

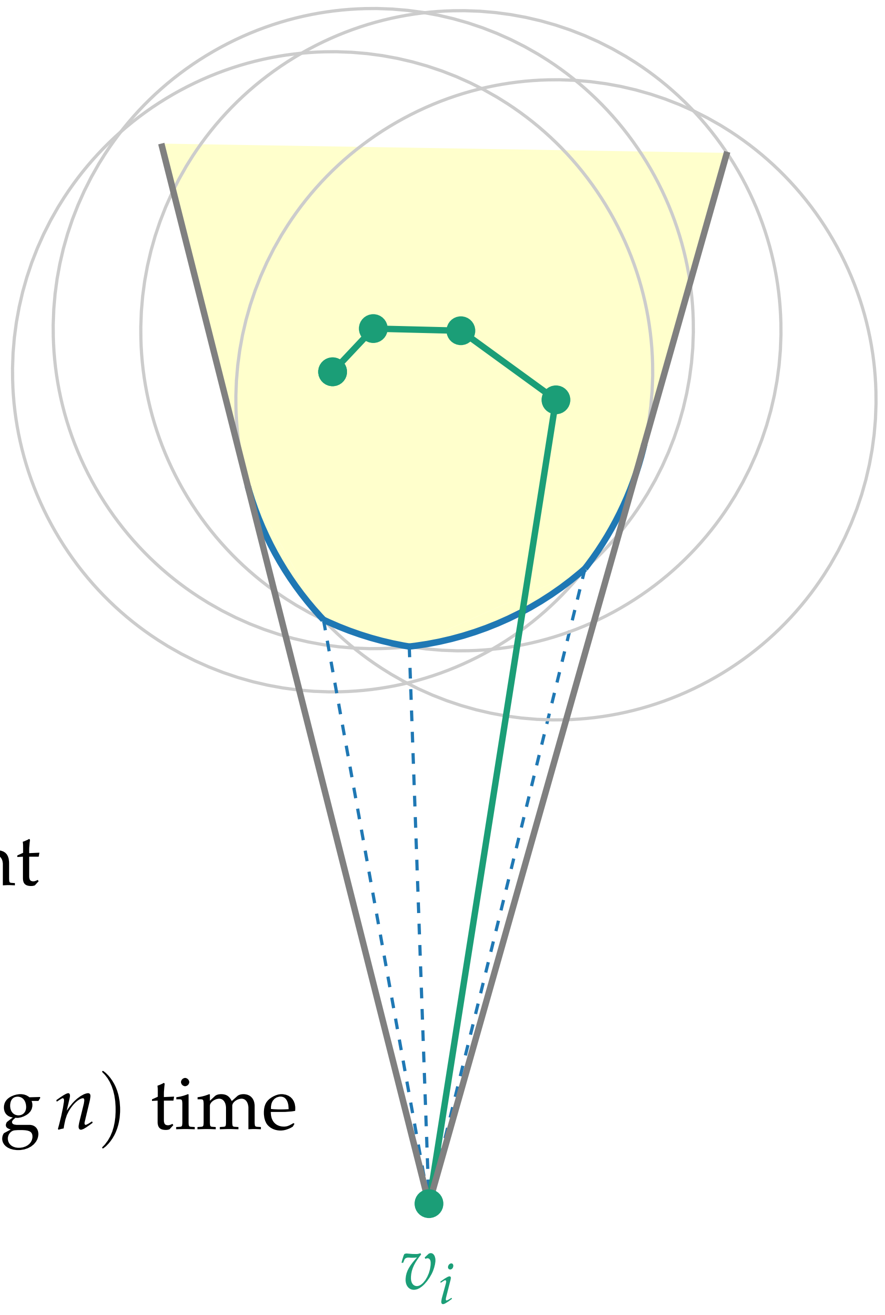
How complex can the wave front be?

- each vertex contributes ≤ 1 arc $\Rightarrow O(n)$ arcs

$\Rightarrow$ stored in a balanced search tree

- finding the intersection between the wave front and a ray emanating at v_i takes $O(\log n)$ time

- updating the wave front takes amortized $O(\log n)$ time



The Wave Front

- is a sequence of circular arcs.

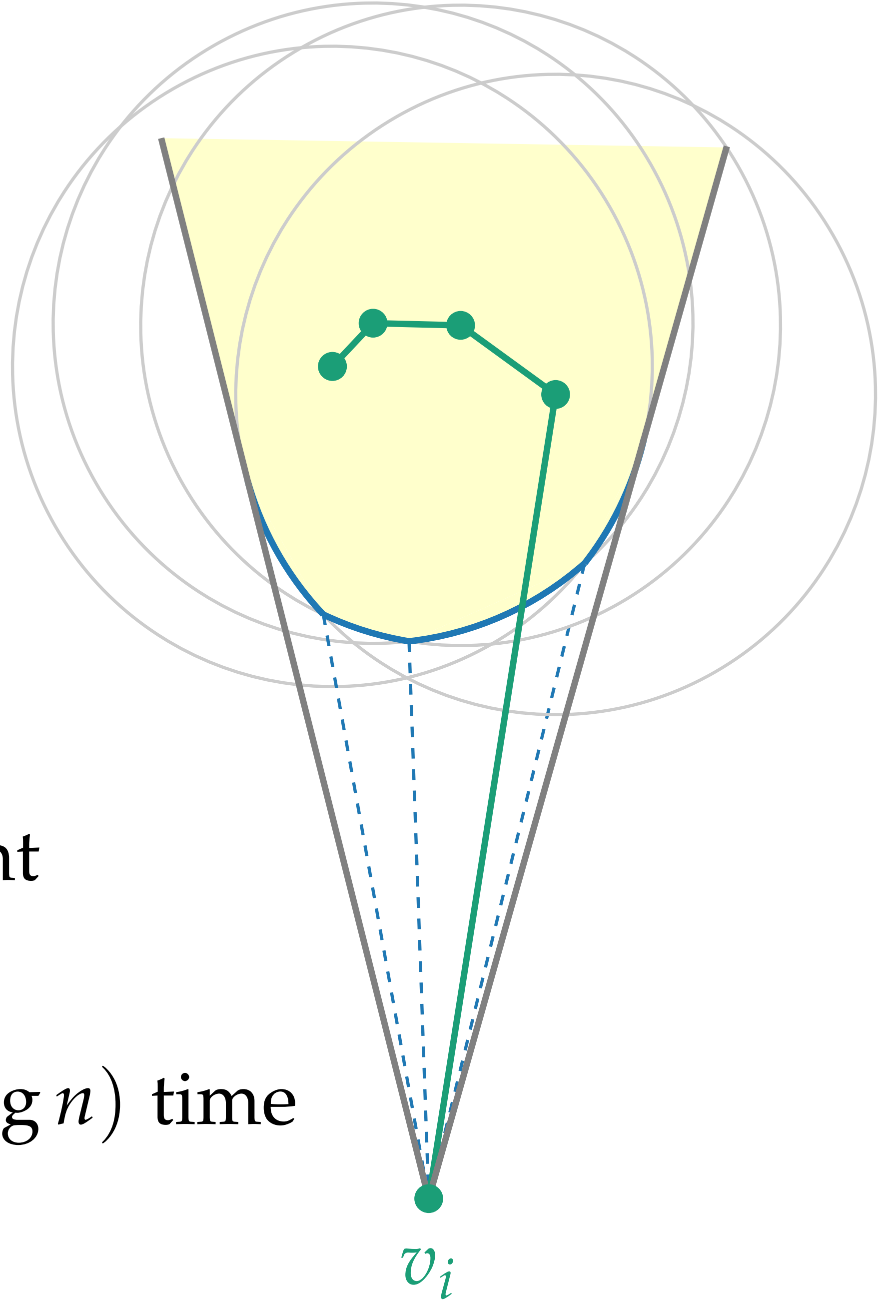
How complex can the wave front be?

- each vertex contributes ≤ 1 arc $\Rightarrow O(n)$ arcs

$\Rightarrow$ stored in a balanced search tree

  - finding the intersection between the wave front and a ray emanating at v_i takes $O(\log n)$ time
  - updating the wave front takes amortized $O(\log n)$ time

$\Rightarrow$ processing a vertex takes amortized $O(\log n)$ time



The Wave Front

- is a sequence of circular arcs.

How complex can the wave front be?

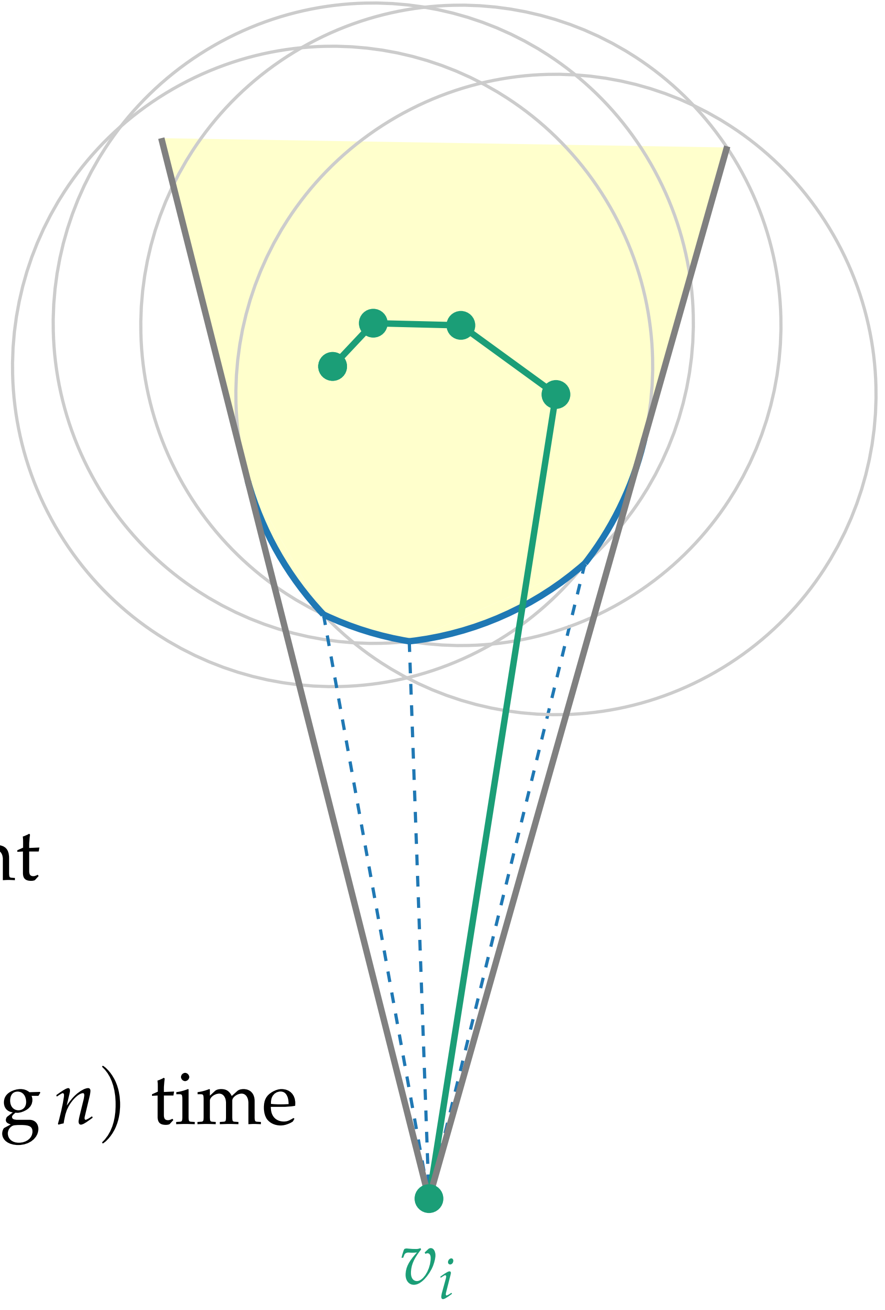
- each vertex contributes ≤ 1 arc $\Rightarrow O(n)$ arcs

$\Rightarrow$ stored in a balanced search tree

  - finding the intersection between the wave front and a ray emanating at v_i takes $O(\log n)$ time
  - updating the wave front takes amortized $O(\log n)$ time

$\Rightarrow$ processing a vertex takes amortized $O(\log n)$ time

$\Rightarrow$ total running time $O(n^2 \log n)$



Our Modifications

- In contrast to the Hausdorff distance, the order of the vertices along a shortcut segment matters for the Fréchet distance.

Our Modifications

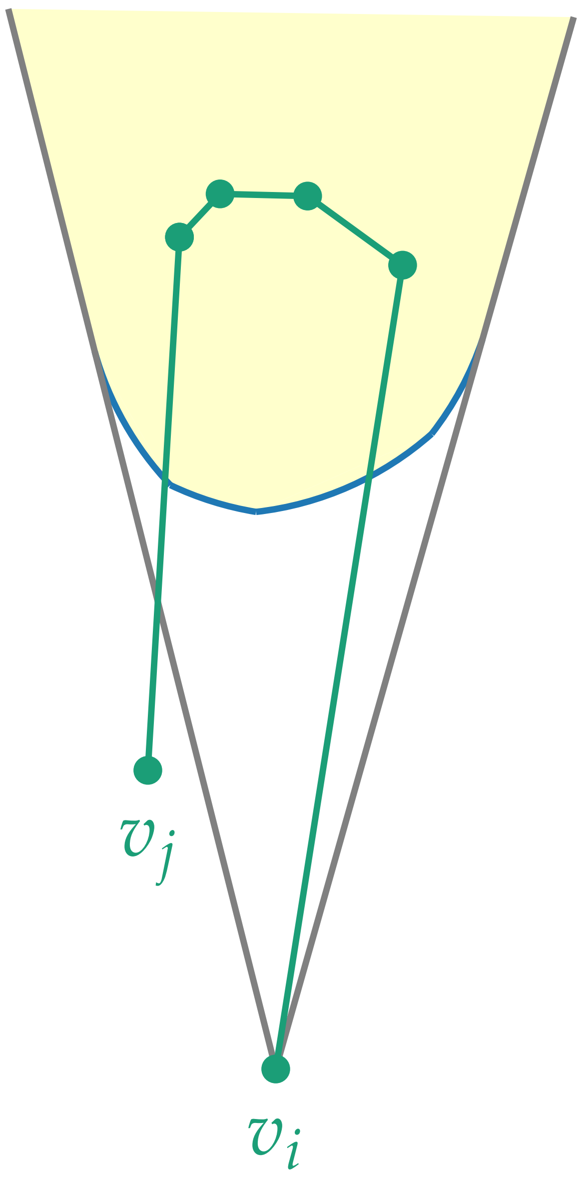
- In contrast to the Hausdorff distance, the order of the vertices along a shortcut segment matters for the Fréchet distance.

⇒ narrow the cone s.t. the ε-circle around v_j contains the whole wave front.

Our Modifications

- In contrast to the Hausdorff distance, the order of the vertices along a shortcut segment matters for the Fréchet distance.

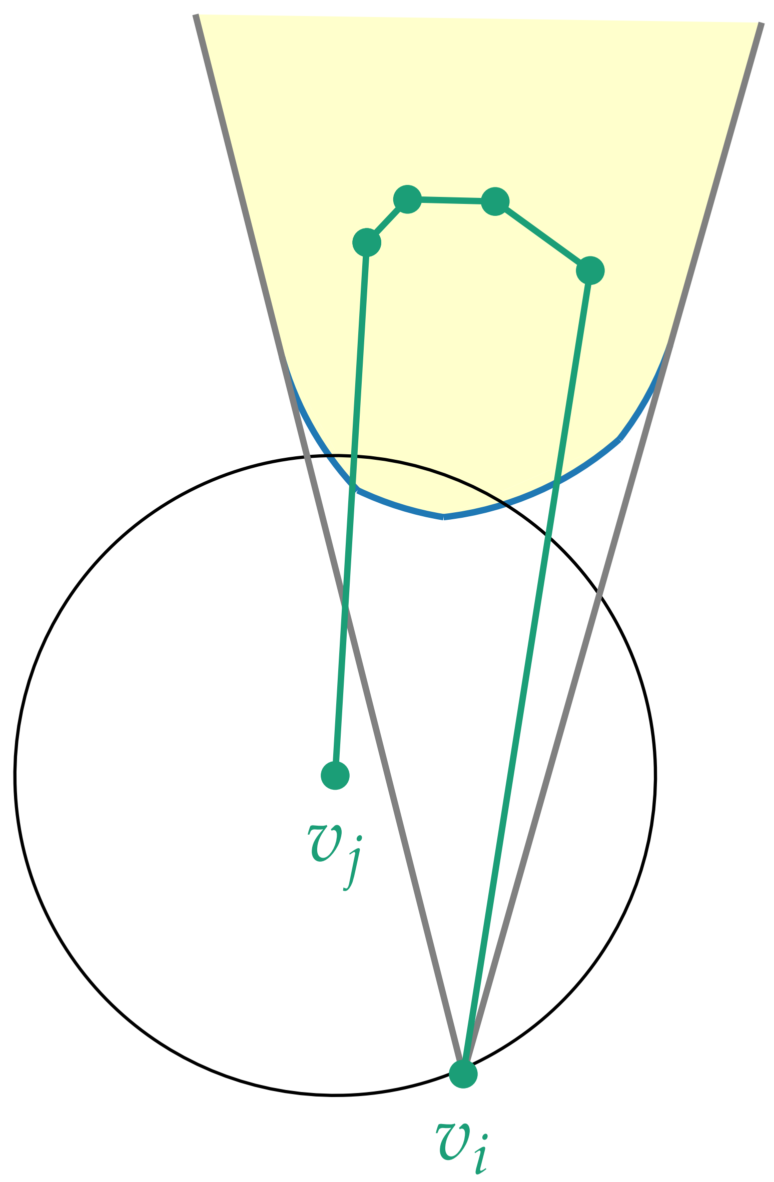
$\Rightarrow$ narrow the cone s.t. the ε-circle around v_j contains the whole wave front.



Our Modifications

- In contrast to the Hausdorff distance, the order of the vertices along a shortcut segment matters for the Fréchet distance.

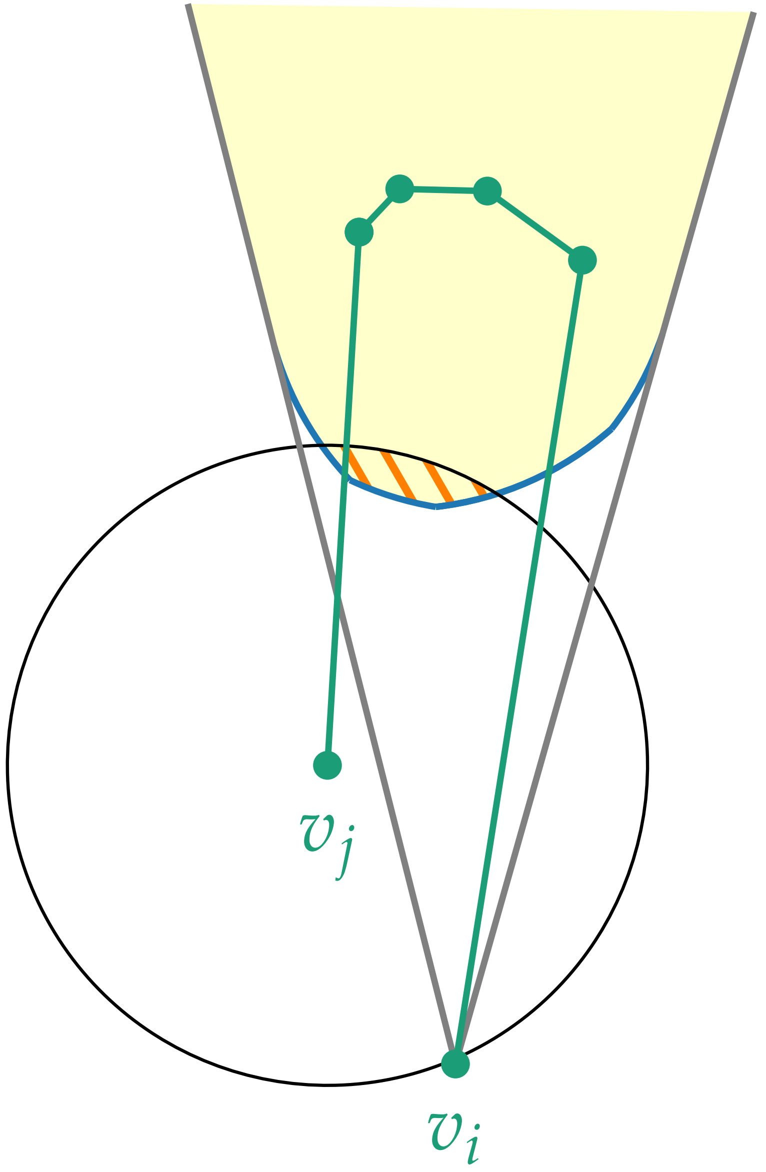
$\Rightarrow$ narrow the cone s.t. the ε-circle around v_j contains the whole wave front.



Our Modifications

- In contrast to the Hausdorff distance, the order of the vertices along a shortcut segment matters for the Fréchet distance.

$\Rightarrow$ narrow the cone s.t. the ε-circle around v_j contains the whole wave front.

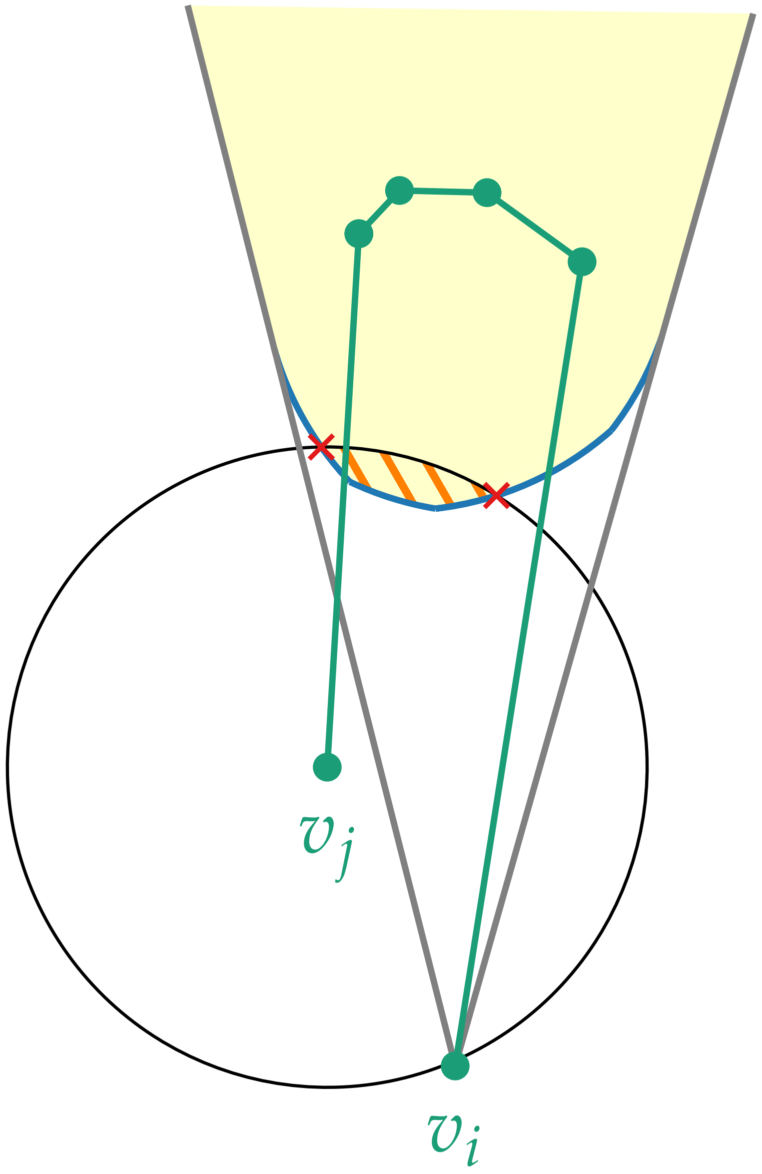


Our Modifications

- In contrast to the Hausdorff distance, the order of the vertices along a short-cut segment matters for the Fréchet distance.

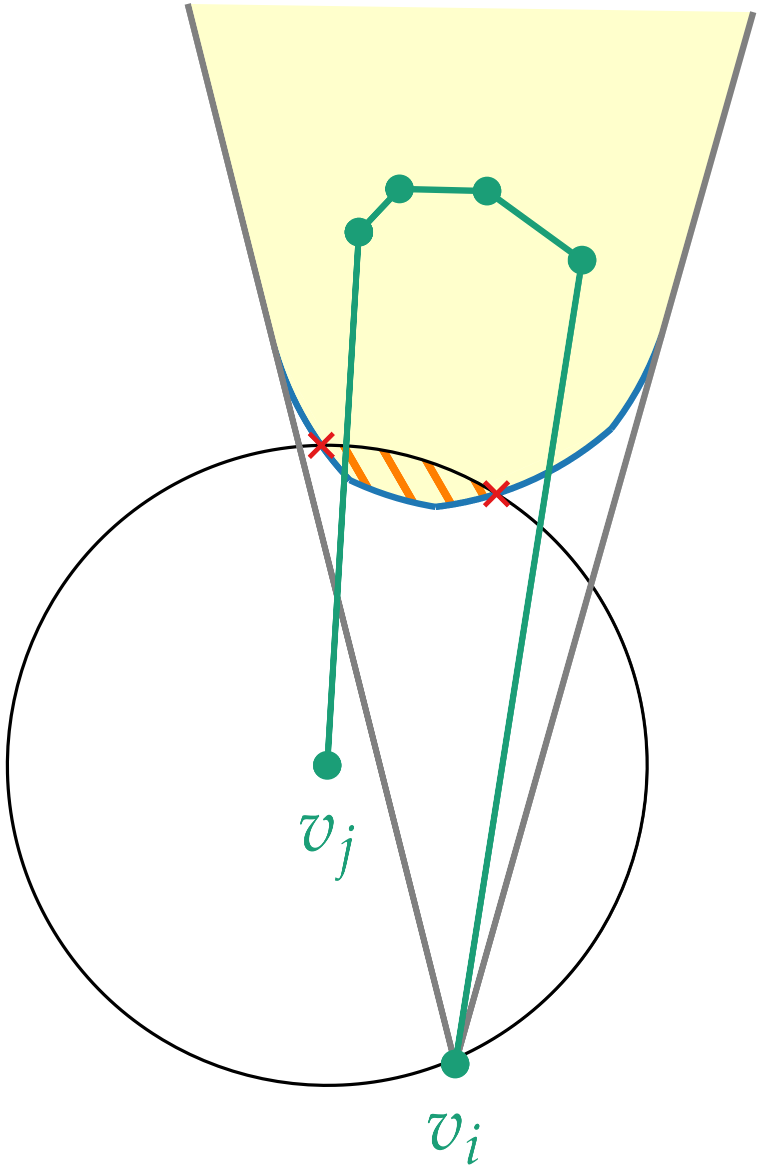
$\Rightarrow$ narrow the cone s.t. the ε-circle around v_j contains the whole wave front.

  - determine the ≤ 2 intersection points between the ε-circle around v_j and the wave front



Our Modifications

- In contrast to the Hausdorff distance, the order of the vertices along a shortcut segment matters for the Fréchet distance.



$\Rightarrow$ narrow the cone s.t. the ε-circle around v_j contains the whole wave front.

- determine the ≤ 2 intersection points between the ε-circle around v_j and the wave front

we show how to do this in $O(\log n)$ time

Our Modifications

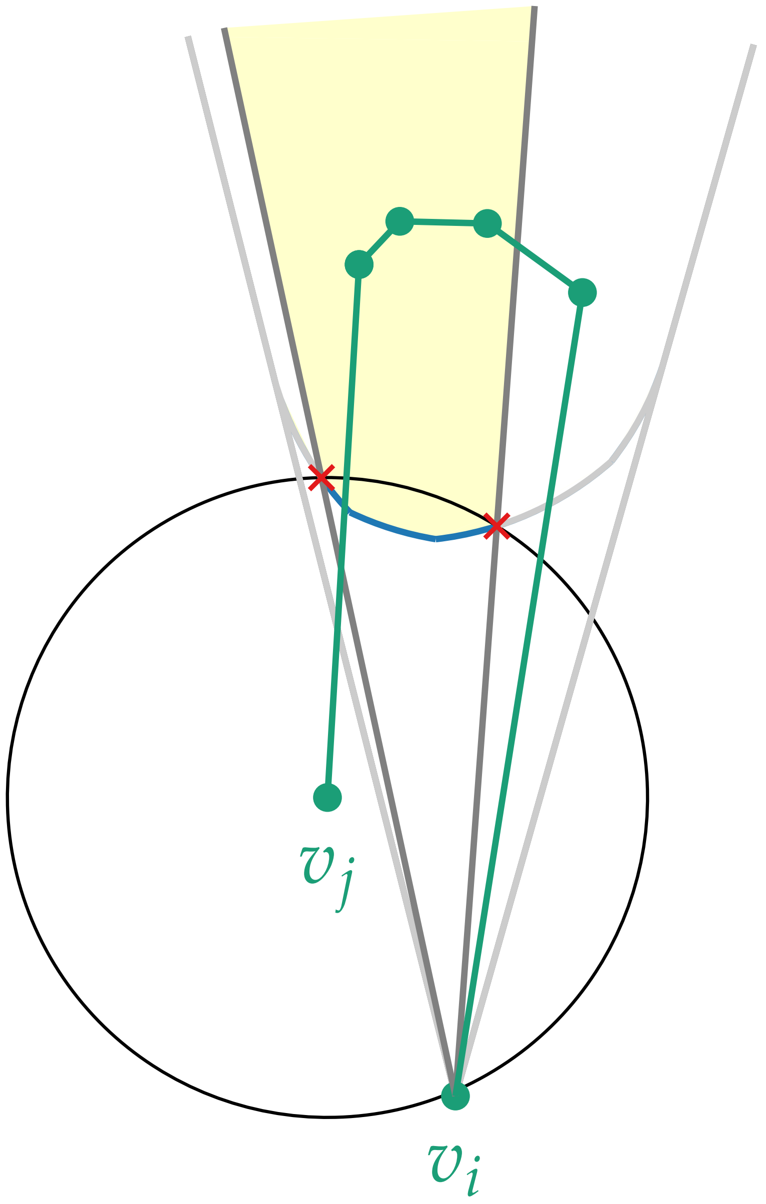
- In contrast to the Hausdorff distance, the order of the vertices along a shortcut segment matters for the Fréchet distance.

$\Rightarrow$ narrow the cone s.t. the ε-circle around v_j contains the whole wave front.

- determine the ≤ 2 intersection points between the ε-circle around v_j and the wave front

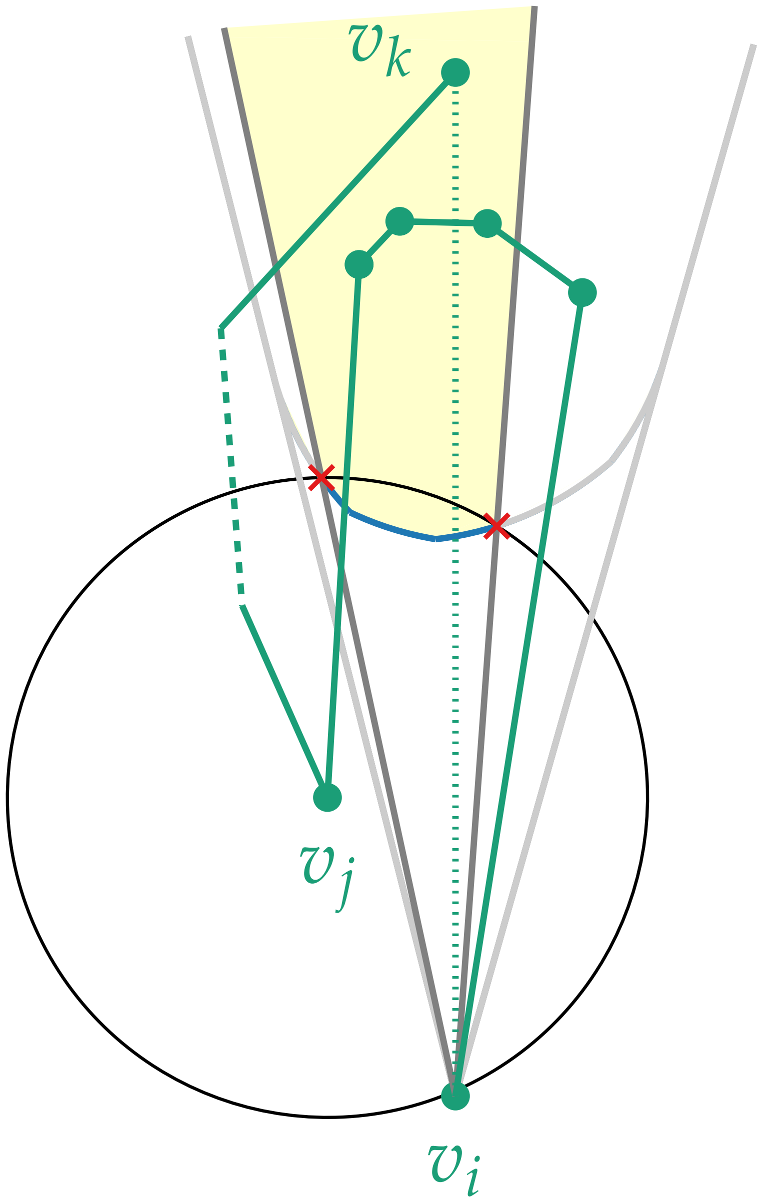
  we show how to do this in $O(\log n)$ time

- narrow the cone accordingly



Our Modifications

- In contrast to the Hausdorff distance, the order of the vertices along a shortcut segment matters for the Fréchet distance.



$\Rightarrow$ narrow the cone s.t. the ε-circle around v_j contains the whole wave front.

  - determine the ≤ 2 intersection points between the ε-circle around v_j and the wave front

    we show how to do this in $O(\log n)$ time

  - narrow the cone accordingly

- Correctness: for a shortcut segment $\overline{v_i v_k}$, map each intermediate vertex v_j to the intersection point of v_j's wave front and $\overline{v_i v_k}$.

Our Modifications

- In contrast to the Hausdorff distance, the order of the vertices along a shortcut segment matters for the Fréchet distance.

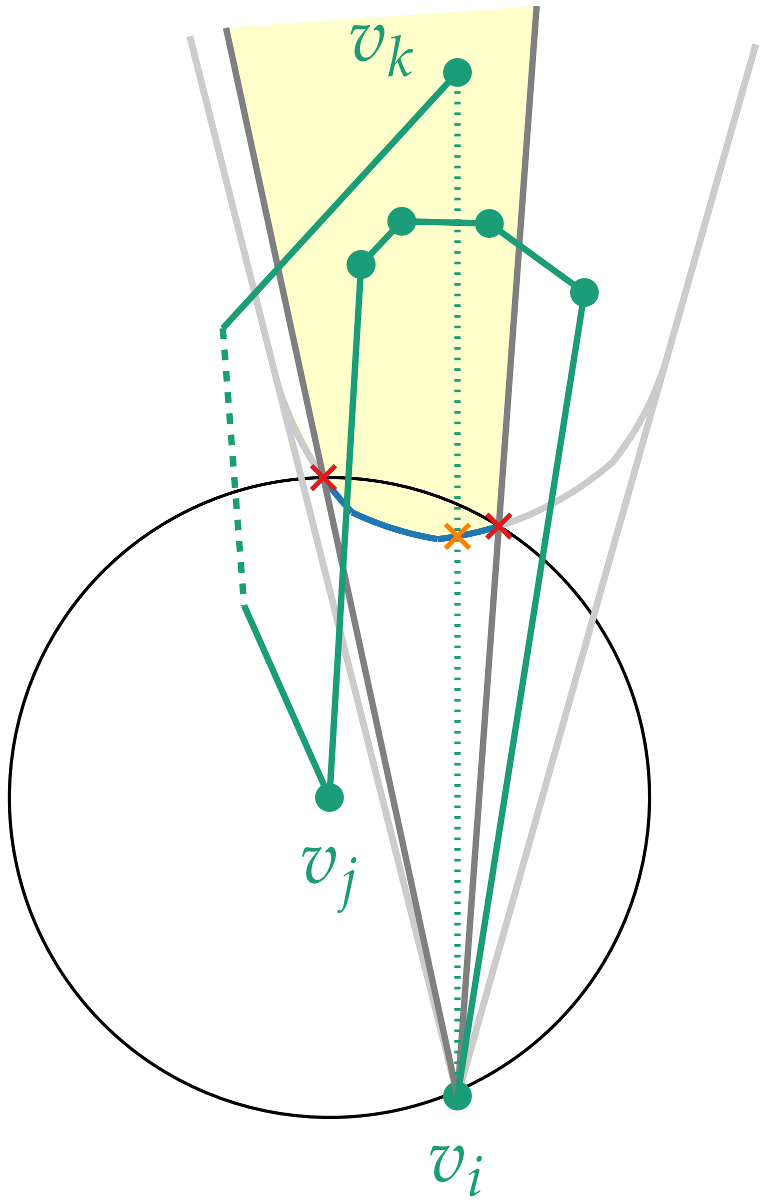
$\Rightarrow$ narrow the cone s.t. the ε-circle around v_j contains the whole wave front.

  - determine the ≤ 2 intersection points between the ε-circle around v_j and the wave front

    we show how to do this in $O(\log n)$ time

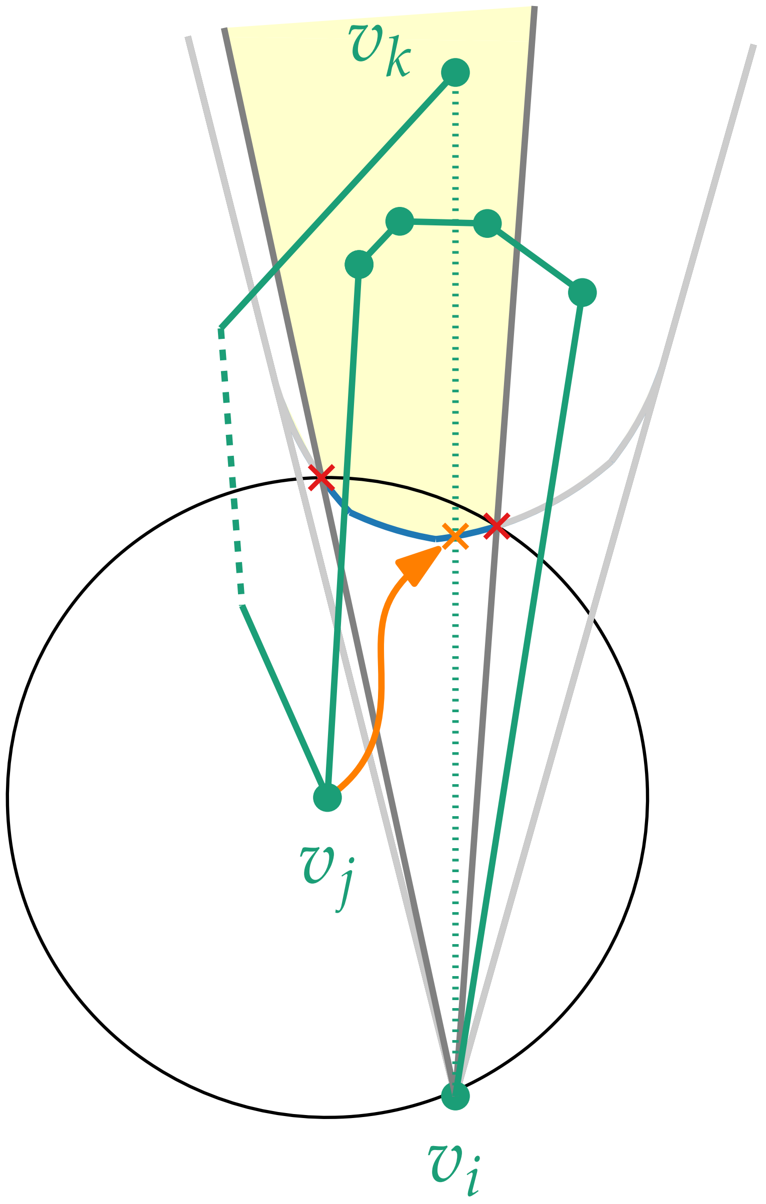
  - narrow the cone accordingly

- Correctness: for a shortcut segment $\overline{v_i v_k}$, map each intermediate vertex v_j to the intersection point of v_j's wave front and $\overline{v_i v_k}$.



Our Modifications

- In contrast to the Hausdorff distance, the order of the vertices along a shortcut segment matters for the Fréchet distance.



$\Rightarrow$ narrow the cone s.t. the ε-circle around v_j contains the whole wave front.

- determine the ≤ 2 intersection points between the ε-circle around v_j and the wave front

  we show how to do this in $O(\log n)$ time

- narrow the cone accordingly

- Correctness: for a shortcut segment $\overline{v_i v_k}$, map each intermediate vertex v_j to the intersection point of v_j's wave front and $\overline{v_i v_k}$.

Our Modifications

- In contrast to the Hausdorff distance, the order of the vertices along a shortcut segment matters for the Fréchet distance.

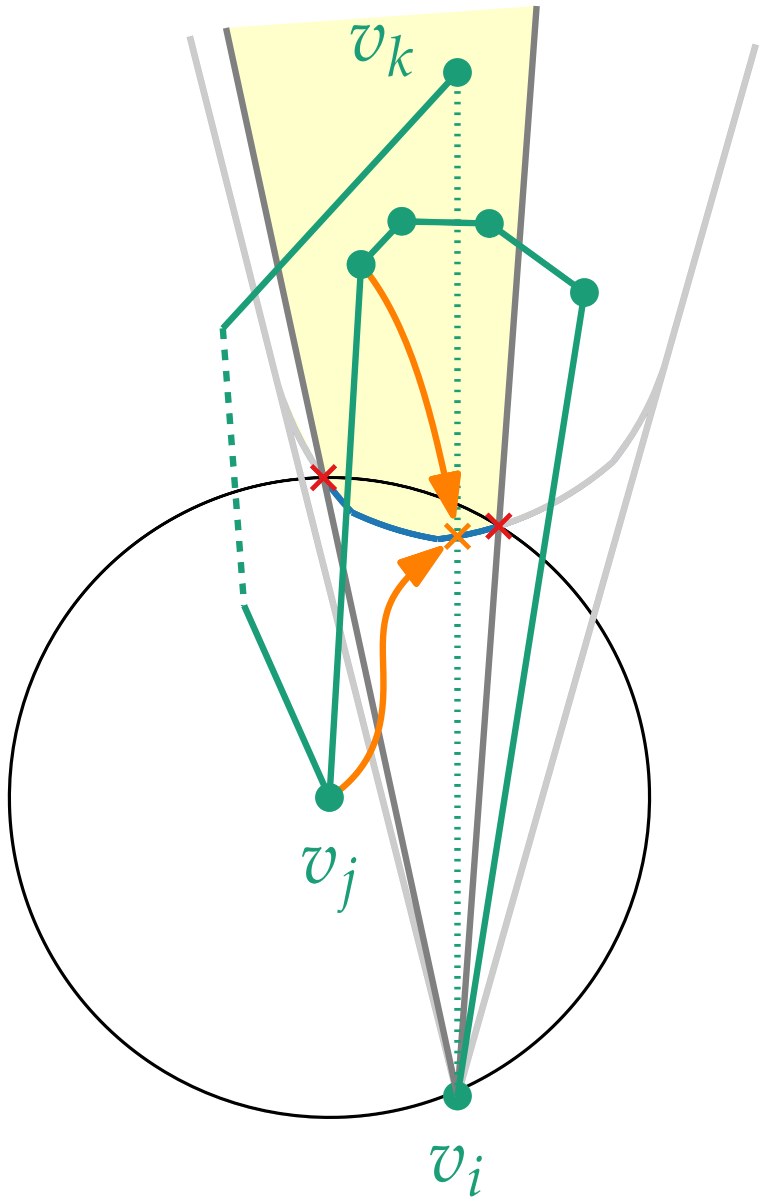
$\Rightarrow$ narrow the cone s.t. the ε-circle around v_j contains the whole wave front.

- determine the ≤ 2 intersection points between the ε-circle around v_j and the wave front

  we show how to do this in $O(\log n)$ time

- narrow the cone accordingly

- Correctness: for a shortcut segment $\overline{v_i v_k}$, map each intermediate vertex v_j to the intersection point of v_j's wave front and $\overline{v_i v_k}$.



L_2 and Other L_p Norms

Theorem: Polyline simplification under the local Fréchet distance can be done in $O(n^2 \log n)$ time in the L_2 norm.

L_2 and Other L_p Norms

Theorem: Polyline simplification under the local Fréchet distance can be done in $O(n^2 \log n)$ time in the L_2 norm.

$\downarrow$

Assumption:
we can determine intersection points between
ε-circle & line and ε-circle & ε-circle
in $O(1)$ time in L_p

L_2 and Other L_p Norms

Theorem: Polyline simplification under the local Fréchet distance can be done in $O(n^2 \log n)$ time in the L_2 norm.

Assumption:
we can determine intersection points between ε-circle & line and ε-circle & ε-circle in $O(1)$ time in L_p

Corollary: Polyline simplification under the local Fréchet distance can be done in $O(n^2 \log n)$ time in the L_p norm where $p \in [1, \infty]$.

L_1 and L_∞ Norm

L_1 and L_∞ Norm

Observation: in L_1 and L_∞, the wave front consist of at most two line segments.

L_1 and L_∞ Norm

Observation: in L_1 and L_∞, the wave front consist of at most two line segments.

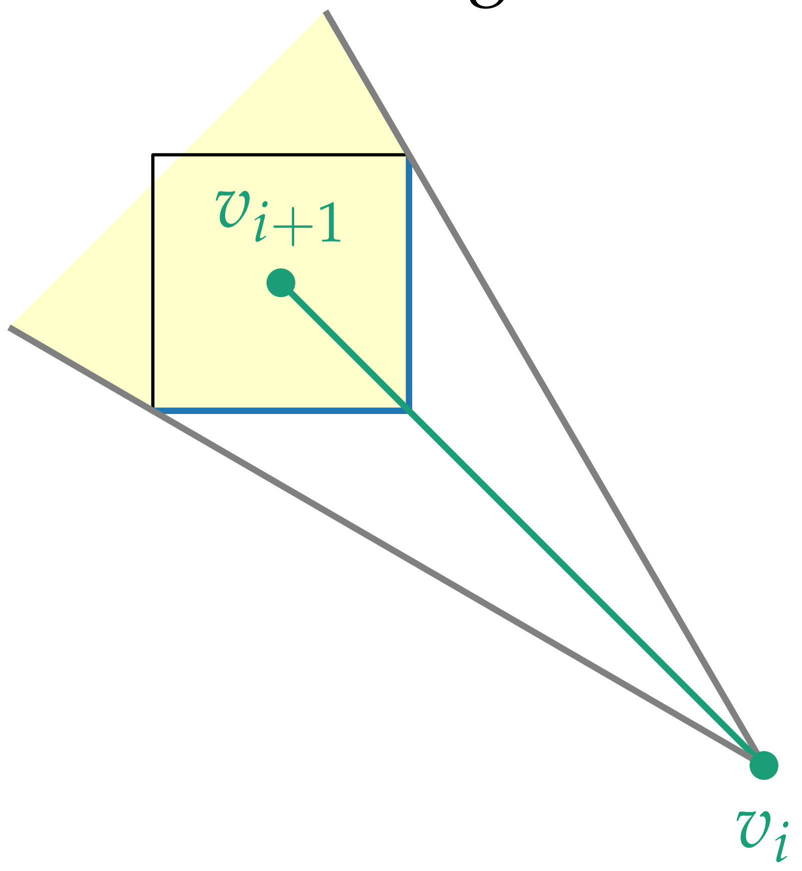
Proof Idea for L_∞: show statement (+ both line segments are axis-parallel) inductively

L_1 and L_∞ Norm

Observation: in L_1 and L_∞, the wave front consist of at most two line segments.

Proof Idea for L_∞: show statement (+ both line segments are axis-parallel) inductively

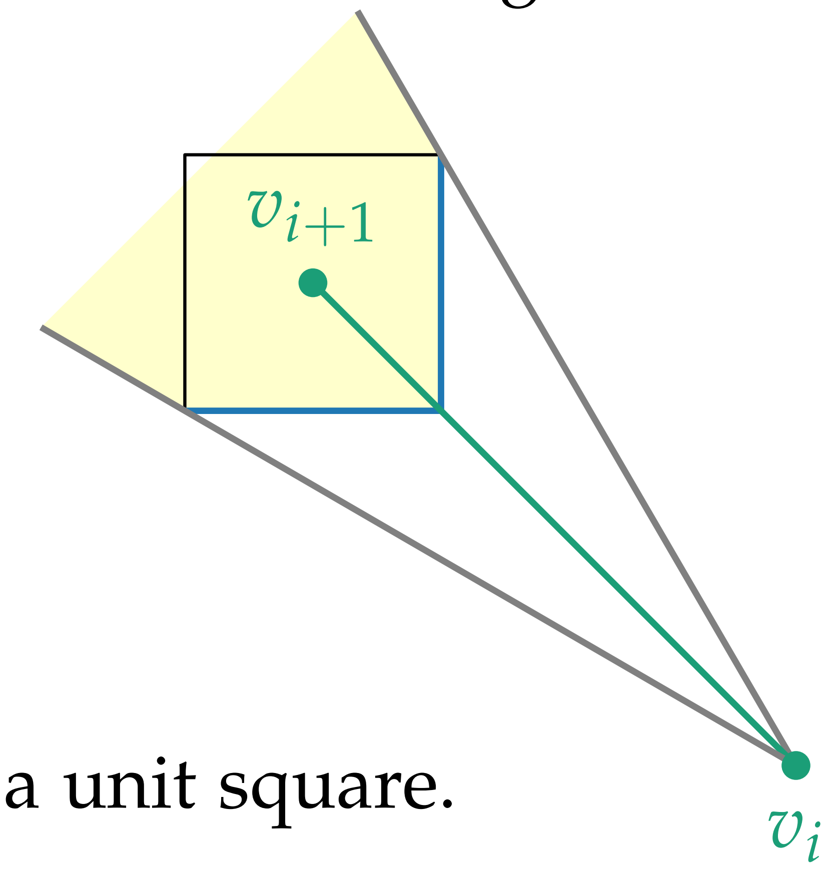
(IB) The first wave front are $\leq$ 2 sides of an axis-parallel square with side length 2ε (*unit square*).



L_1 and L_∞ Norm

Observation: in L_1 and L_∞, the wave front consist of at most two line segments.

Proof Idea for L_∞: show statement (+ both line segments are axis-parallel) inductively



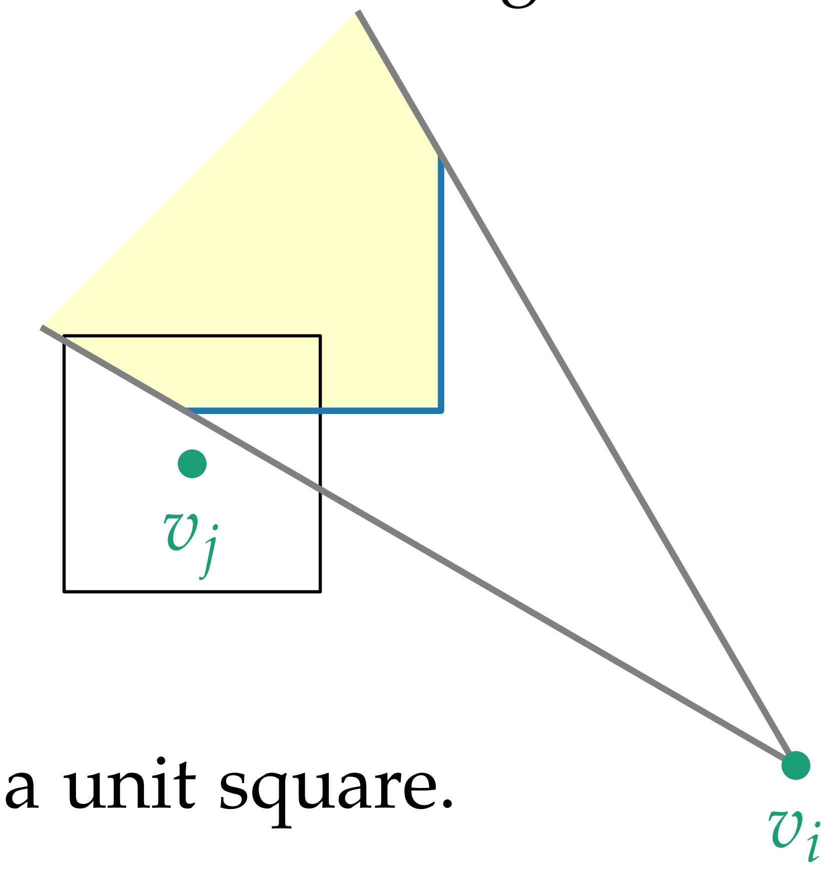
(IB) The first wave front are ≤ 2 sides of an axis-parallel square with side length 2ε (*unit square*).

(IS) To obtain the new wave front, we compute the intersection of two axis-parallel line segments with a unit square.

L_1 and L_∞ Norm

Observation: in L_1 and L_∞, the wave front consist of at most two line segments.

Proof Idea for L_∞: show statement (+ both line segments are axis-parallel) inductively



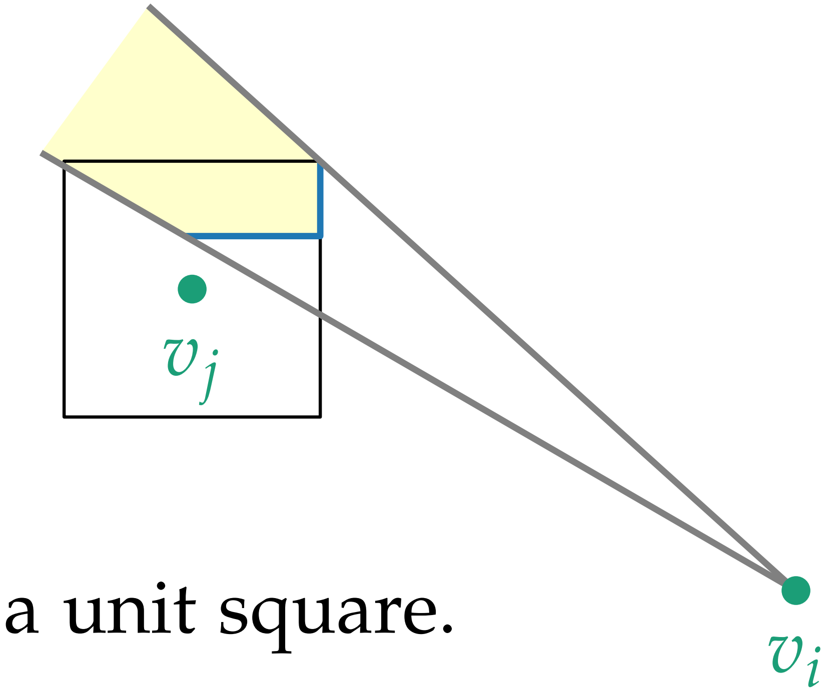
(IB) The first wave front are $\leq$ 2 sides of an axis-parallel square with side length 2ε (*unit square*).

(IS) To obtain the new wave front, we compute the intersection of two axis-parallel line segments with a unit square.

L_1 and L_∞ Norm

Observation: in L_1 and L_∞, the wave front consist of at most two line segments.

Proof Idea for L_∞: show statement (+ both line segments are axis-parallel) inductively



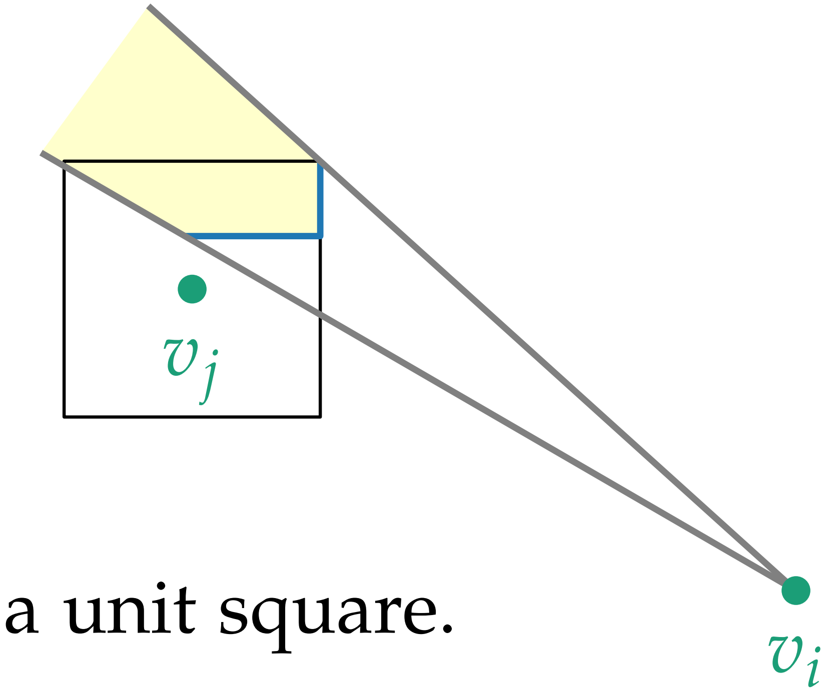
(IB) The first wave front are ≤ 2 sides of an axis-parallel square with side length 2ε (*unit square*).

(IS) To obtain the new wave front, we compute the intersection of two axis-parallel line segments with a unit square.

L_1 and L_∞ Norm

Observation: in L_1 and L_∞, the wave front consist of at most two line segments.

Proof Idea for L_∞: show statement (+ both line segments are axis-parallel) inductively



(IB) The first wave front are $\leq$ 2 sides of an axis-parallel square with side length 2ε (*unit square*).

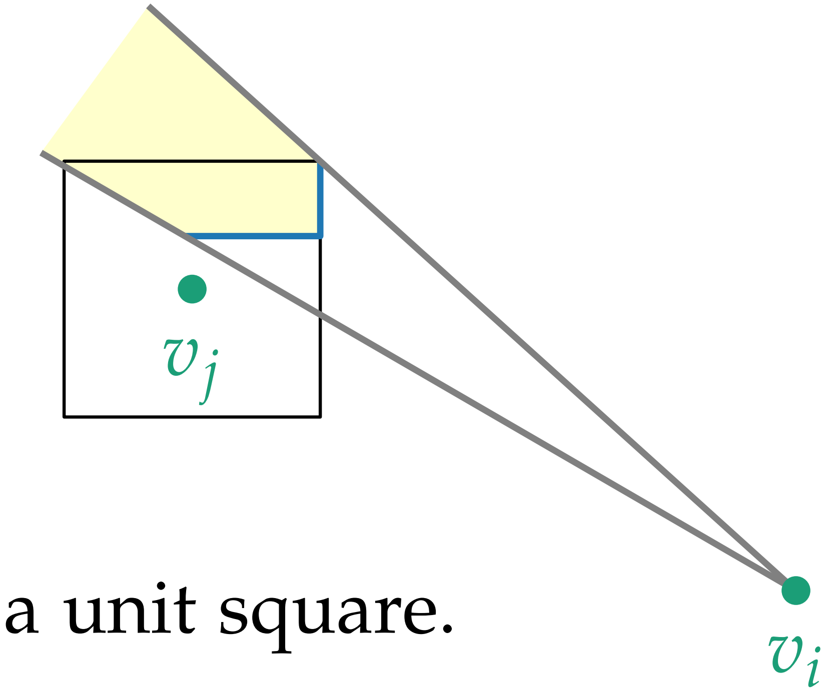
(IS) To obtain the new wave front, we compute the intersection of two axis-parallel line segments with a unit square.

Hence, the resulting wave front has again ≤ 2 axis-parallel line segments.

L_1 and L_∞ Norm

Observation: in L_1 and L_∞, the wave front consist of at most two line segments.

Proof Idea for L_∞: show statement (+ both line segments are axis-parallel) inductively



(IB) The first wave front are $\leq$ 2 sides of an axis-parallel square with side length 2ε (*unit square*).

(IS) To obtain the new wave front, we compute the intersection of two axis-parallel line segments with a unit square.

Hence, the resulting wave front has again ≤ 2 axis-parallel line segments.

Theorem: Polyline simplification under the local Fréchet distance can be done in $O(n^2)$ time in the L_1 and L_∞ norm.

Summary & Outlook

- Using techniques from the algorithm by Melkman and O'Rourke, we have shown how to find an optimal simplification of a polyline under the local Fréchet distance in $O(n^2 \log n)$ (instead of $O(n^3)$) time in the L_2 norm.

Summary & Outlook

- Using techniques from the algorithm by Melkman and O'Rourke, we have shown how to find an optimal simplification of a polyline under the local Fréchet distance in $O(n^2 \log n)$ (instead of $O(n^3)$) time in the L_2 norm.
- Our result generalizes to other L_p norms ($p \in [1, \infty]$).
  In L_1 and L_∞, the runtime improves to $O(n^2)$.

Summary & Outlook

Results

- Using techniques from the algorithm by Melkman and O'Rourke, we have shown how to find an optimal simplification of a polyline under the local Fréchet distance in $O(n^2 \log n)$ (instead of $O(n^3)$) time in the L_2 norm.
- Our result generalizes to other L_p norms ($p \in [1, \infty]$).
  In L_1 and L_∞, the runtime improves to $O(n^2)$.

Summary & Outlook

Results

- Using techniques from the algorithm by Melkman and O'Rourke, we have shown how to find an optimal simplification of a polyline under the local Fréchet distance in $O(n^2 \log n)$ (instead of $O(n^3)$) time in the L_2 norm.
- Our result generalizes to other L_p norms ($p \in [1, \infty]$).
  In L_1 and L_∞, the runtime improves to $O(n^2)$.

Future Work

Summary & Outlook

Results

- Using techniques from the algorithm by Melkman and O'Rourke, we have shown how to find an optimal simplification of a polyline under the local Fréchet distance in $O(n^2 \log n)$ (instead of $O(n^3)$) time in the L_2 norm.
- Our result generalizes to other L_p norms ($p \in [1, \infty]$).
  In L_1 and L_∞, the runtime improves to $O(n^2)$.

Future Work

- The log-factor arises when the wave front has linear size.
  However, this requires a special arrangment of the vertices of the polyline.

Summary & Outlook

Results

- Using techniques from the algorithm by Melkman and O'Rourke, we have shown how to find an optimal simplification of a polyline under the local Fréchet distance in $O(n^2 \log n)$ (instead of $O(n^3)$) time in the L_2 norm.
- Our result generalizes to other L_p norms ($p \in [1, \infty]$).
  In L_1 and L_∞, the runtime improves to $O(n^2)$.

Future Work

- The log-factor arises when the wave front has linear size.
  However, this requires a special arrangment of the vertices of the polyline.
- Test experimentally that in a practical setting such arrangments don't occur.

Summary & Outlook

Results

- Using techniques from the algorithm by Melkman and O'Rourke, we have shown how to find an optimal simplification of a polyline under the local Fréchet distance in $O(n^2 \log n)$ (instead of $O(n^3)$) time in the L_2 norm.
- Our result generalizes to other L_p norms ($p \in [1, \infty]$).
  In L_1 and L_∞, the runtime improves to $O(n^2)$.

Future Work

- The log-factor arises when the wave front has linear size.
  However, this requires a special arrangment of the vertices of the polyline.
- Test experimentally that in a practical setting such arrangments don't occur.
- Investigate conditions that guarantee $O(n^2)$ runtime (e.g. density in each ε-ball, smoothed analysis).

Summary & Outlook

Results

- Using techniques from the algorithm by Melkman and O'Rourke, we have shown how to find an optimal simplification of a polyline under the local Fréchet distance in $O(n^2 \log n)$ (instead of $O(n^3)$) time in the L_2 norm.
- Our result generalizes to other L_p norms ($p \in [1, \infty]$).
  In L_1 and L_∞, the runtime improves to $O(n^2)$.

Future Work

- The log-factor arises when the wave front has linear size.
  However, this requires a special arrangment of the vertices of the polyline.
- Test experimentally that in a practical setting such arrangments don't occur.
- Investigate conditions that guarantee $O(n^2)$ runtime (e.g. density in each ε-ball, smoothed analysis).
- Adjust the algorithm to handle such arrangements faster.