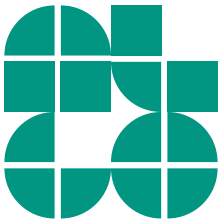
Vorlesung Algorithmische Kartografie

Einführung & Linienvereinfachung

LEHRSTUHL FÜR ALGORITHMIK I · INSTITUT FÜR THEORETISCHE INFORMATIK · FAKULTÄT FÜR INFORMATIK

Martin Nöllenburg

16.04.2013



AlgoKarto-Team

Dozent

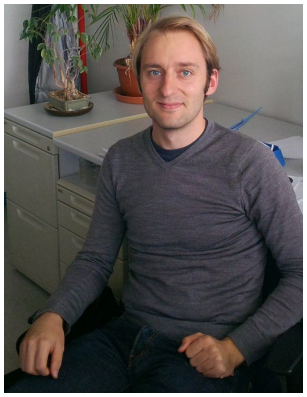
- Martin Nöllenburg
- noellenburg@kit.edu
- Raum 319
- Sprechzeiten: individuell per Mail vereinbaren

AlgoKarto-Team

Dozent

- Martin Nöllenburg
- noellenburg@kit.edu
- Raum 319
- Sprechzeiten: individuell per Mail vereinbaren

Übungsleiter



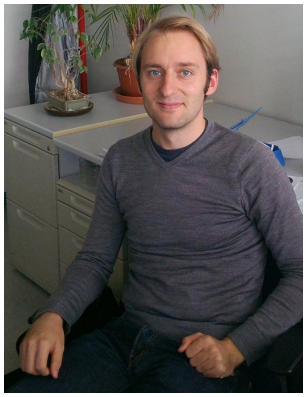
- Benjamin Niedermann
- niedermann@kit.edu
- Raum 322
- Sprechzeiten: individuell per Mail vereinbaren

AlgoKarto-Team

Dozent

- Martin Nöllenburg
- noellenburg@kit.edu
- Raum 319
- Sprechzeiten: individuell per Mail vereinbaren

Übungsleiter



- Benjamin Niedermann
- niedermann@kit.edu
- Raum 322
- Sprechzeiten: individuell per Mail vereinbaren

Termine

- Vorlesung: Di 9:45 – 11:15 Uhr, Raum 301
- Übung: Do 10:15 – 11:00 Uhr, Raum 301 (ab 25.04.)

Organisatorisches

Webseite

`i11www.iti.kit.edu/teaching/sommer2013/algokartografie`

- aktuelle Informationen
- Vorlesungsfolien
- Übungsblätter
- Zusatzmaterial

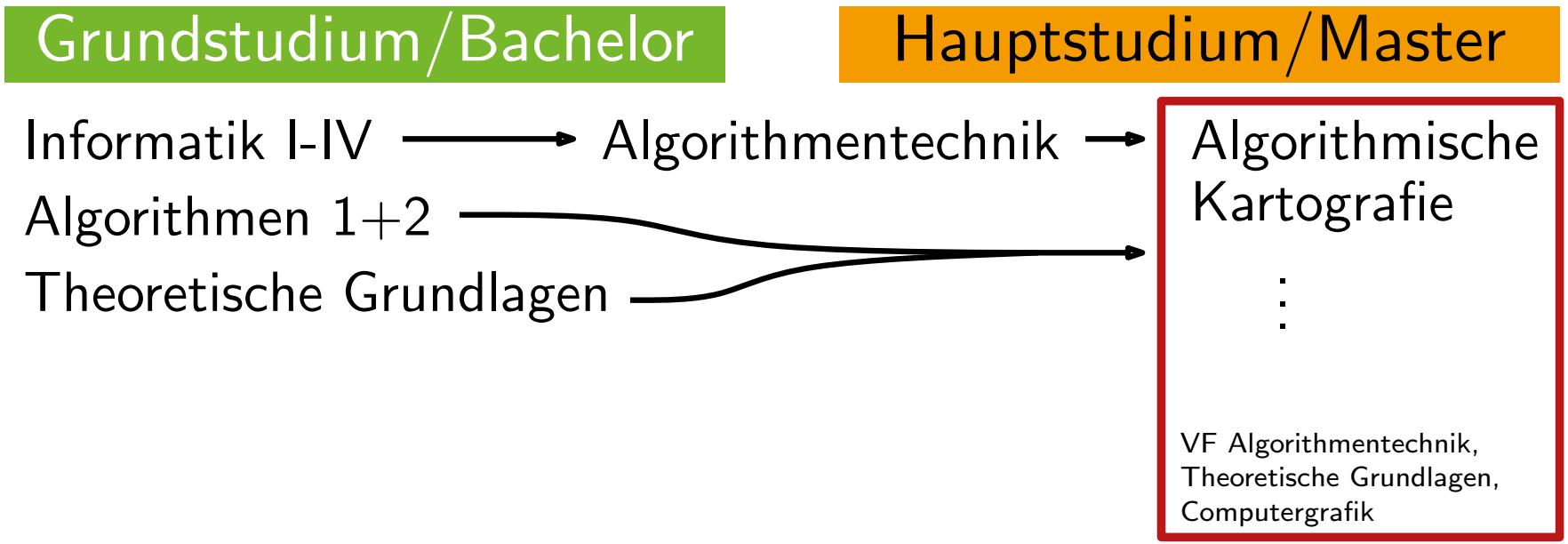
Organisatorisches

Webseite

`i11www.iti.kit.edu/teaching/sommer2013/algokartografie`

- aktuelle Informationen
- Vorlesungsfolien
- Übungsblätter
- Zusatzmaterial

Algorithmische Kartografie im Master-/Diplom-Studium



Algorithmische Kartografie

Karlsruhe Institute of Technology

Lernziele: Am Ende der Vorlesung können Sie

- Begriffe, Strukturen und Problemdefinitionen erklären
- behandelte Algorithmen ausführen, erklären und analysieren
- geeignete Algorithmen und Datenstrukturen auswählen und anpassen
- neue kartografische/geometrische Probleme analysieren und eigene effiziente Lösungen entwerfen

Algorithmische Kartografie

Karlsruhe Institute of Technology

Lernziele: Am Ende der Vorlesung können Sie

- Begriffe, Strukturen und Problemdefinitionen erklären
- behandelte Algorithmen ausführen, erklären und analysieren
- geeignete Algorithmen und Datenstrukturen auswählen und anpassen
- neue kartografische/geometrische Probleme analysieren und eigene effiziente Lösungen entwerfen

Vorkenntnisse: Algorithmik und elementare Geometrie

hilfreich: Algorithmische Geometrie bzw. Algorithmen zur Visualisierung von Graphen

Algorithmische Kartografie

Karlsruhe Institute of Technology

Lernziele: Am Ende der Vorlesung können Sie

- Begriffe, Strukturen und Problemdefinitionen erklären
- behandelte Algorithmen ausführen, erklären und analysieren
- geeignete Algorithmen und Datenstrukturen auswählen und anpassen
- neue kartografische/geometrische Probleme analysieren und eigene effiziente Lösungen entwerfen

Vorkenntnisse: Algorithmik und elementare Geometrie

hilfreich: Algorithmische Geometrie bzw. Algorithmen zur Visualisierung von Graphen

Anforderungen/Prüfung: 5LP = 150h

- aktive Teilnahme an Vorlesung und Übung — ca. 50h
- Bearbeiten der Übungsblätter — ca. 50h
- mündliche Prüfung (einzeln oder in größeren Modulen) — ca. 50h

Prüfbarkeit

Master Informatik

- Algorithm Engineering und Anwendungen (IN4INAEA)
- Design und Analyse von Algorithmen (IN4INDAA)
- Algorithmen der Computergrafik (IN4INACG, 3LP nur VL)
- Algorithmische Kartografie (Einzelmodul 5LP)

Master Informationswirtschaft

- Advanced Algorithms: Engineering and Applications (IW4INAALGOB)
- Advanced Algorithms: Design and Analysis (IW4INAADA)
- Algorithmen der Computergrafik (IW4INACG)

Diplom Informatik

- Vertiefungsfächer Algorithmik, Theoretische Grundlagen, Computergrafik (je 2 SWS)

Übungsbetrieb

Ausgabe Blatt 1
for Woche $i = 2 \ldots 14$ **do**
    if Tag == Dienstag **then**
        Ausgabe Blatt i
        Abgabe Blatt $(i - 1)$
    if Tag == Donnerstag & !isFeiertag(Tag) **then**
        Besprechung Blatt $(i - 1)$

- Bearbeitung der Aufgaben und Abgabe der Lösungen in Zweiergruppen erwünscht
- Übung in der Regel wöchentlich ca. 45 Minuten
- abweichende Regelung an Feiertagen

Werbung: Lehre am Lehrstuhl Algorithmik I

Karlsruhe Institute of Technology

Vertiefungsvorlesungen

- Algorithmen für planare Graphen
  (Di 14:00 Uhr, Do 14:00 Uhr)
- Algorithmen für Routenplanung
  (Mo 14:00 Uhr, Mi 11:30 Uhr)

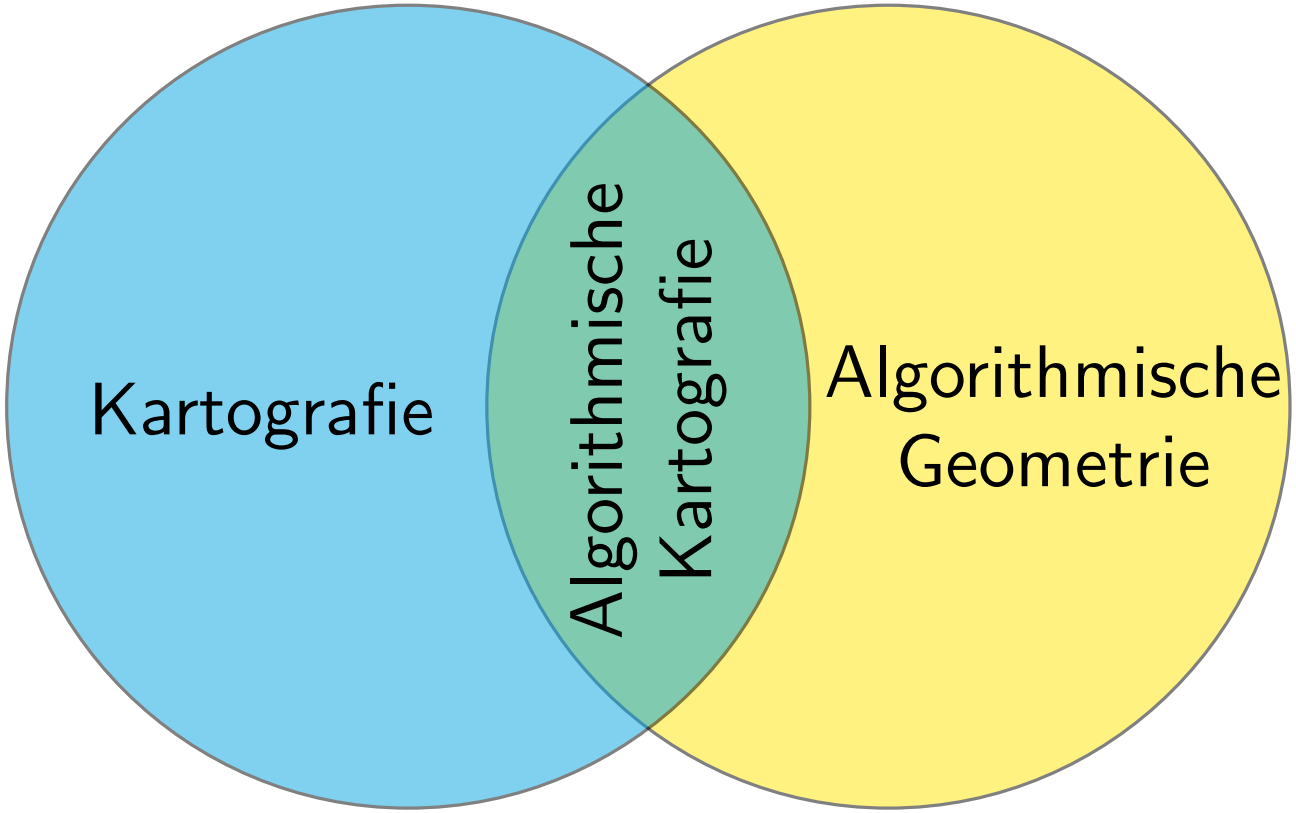
Seminare

- Algorithmen für Energienetze
  Vorbesprechung Di, 23.04. 15:45 Uhr
  bitte per Email anmelden

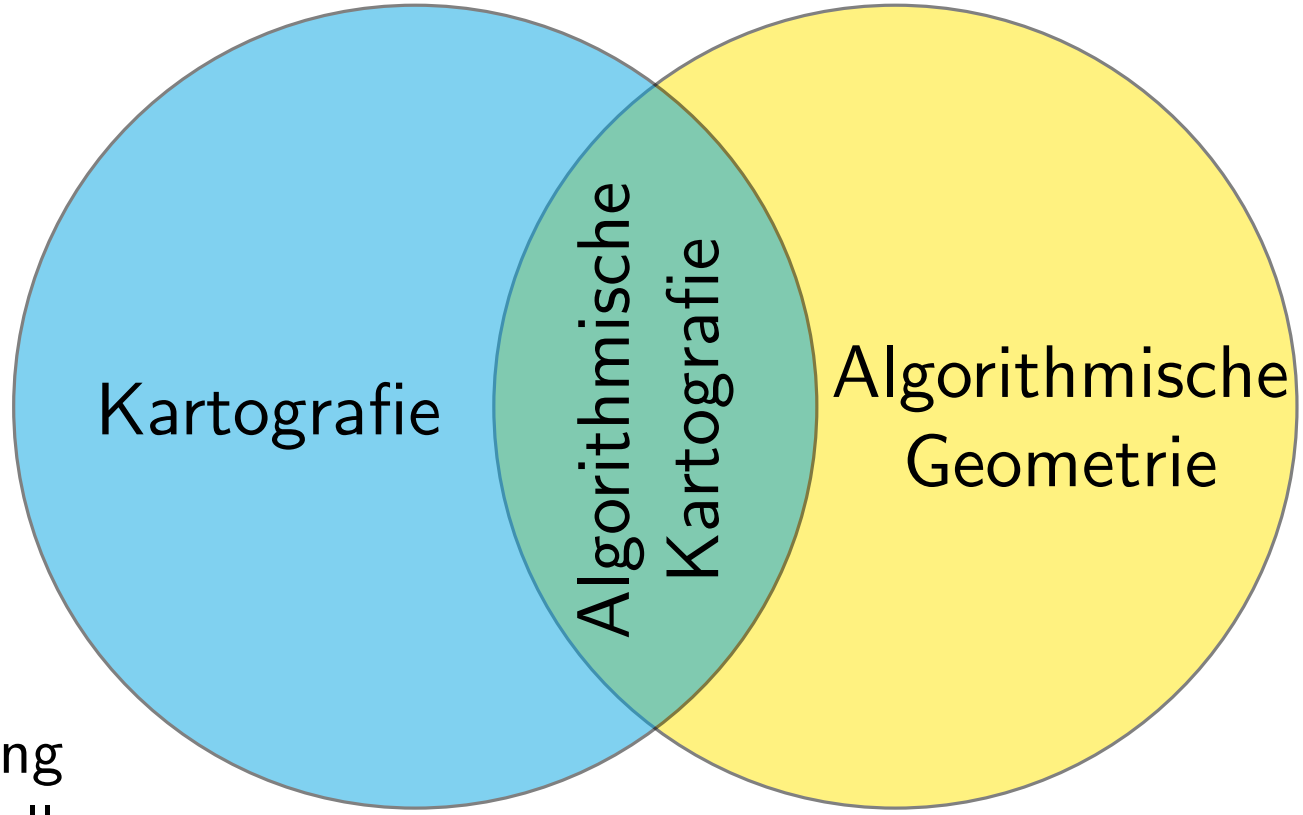
Was ist algorithmische Kartografie?

Karlsruhe Institute of Technology

Was ist algorithmische Kartografie?



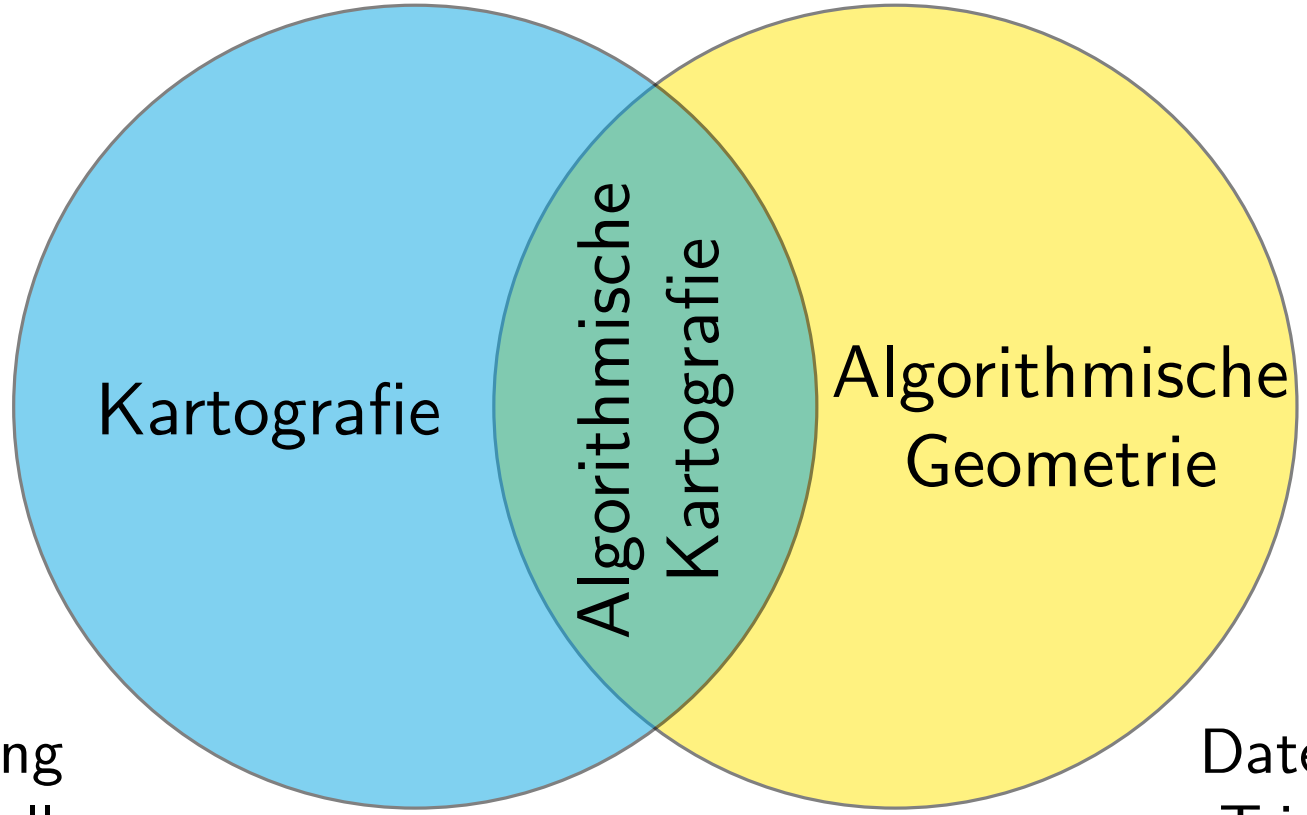
Was ist algorithmische Kartografie?



GIS
Fernerkundung
Geländemodelle
Kartenprojektionen
...

→ Studiengang Geodäsie und Geoinformatik

Was ist algorithmische Kartografie?



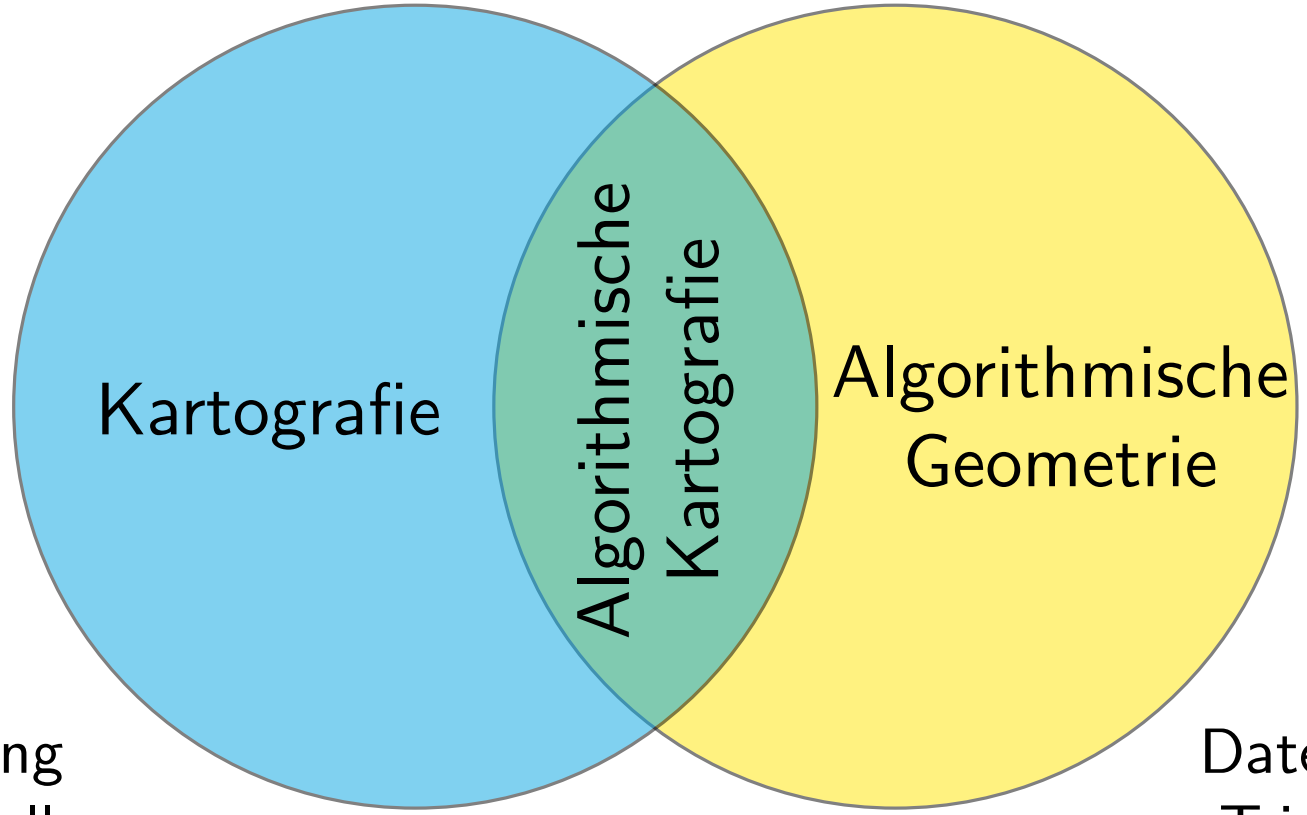
GIS
Fernerkundung
Geländemodelle
Kartenprojektionen
...

→ Studiengang Geodäsie und Geoinformatik

Algorithmen
Datenstrukturen
Triangulationen
Bereichsabfragen
Schnitte, Hüllen ...

→ Vorlesung Algorithmische Geometrie

Was ist algorithmische Kartografie?



GIS
Fernerkundung
Geländemodelle
Kartenprojektionen
...

→ Studiengang Geodäsie und Geoinformatik

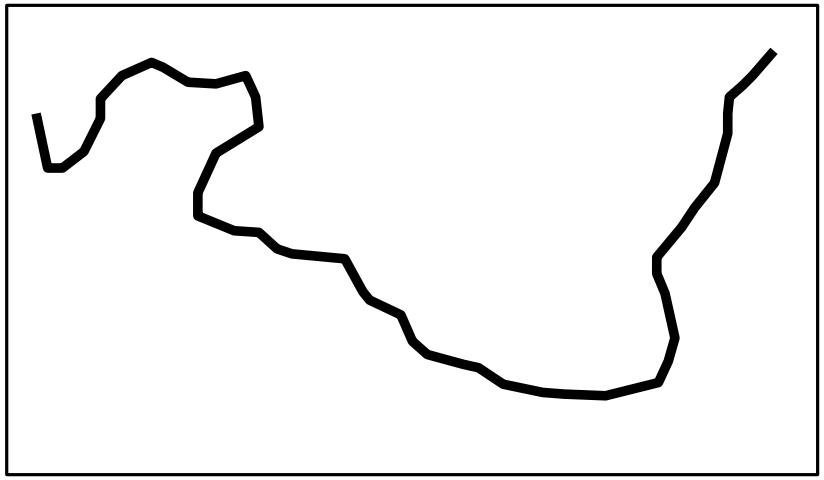
geometrische Algorithmen für kartografische Probleme

Algorithmen
Datenstrukturen
Triangulationen
Bereichsabfragen
Schnitte, Hüllen ...

→ Vorlesung Algorithmische Geometrie

Beispiel 1: Vereinfachung von Kantenzügen

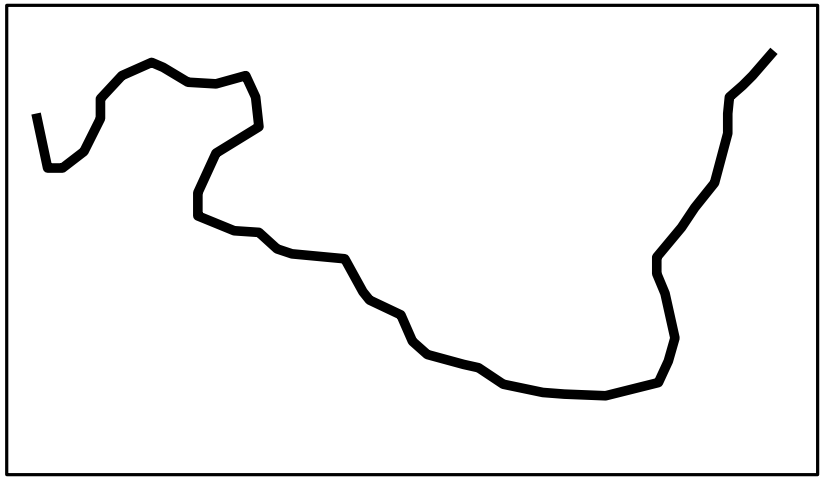
- viele Objekte in Landkarten werden durch Polygone oder Kantenzüge repräsentiert
- Detailgrad abhängig vom Maßstab (Generalisierung)



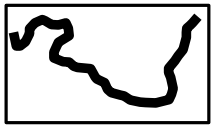
Maßstab $1 : X$

Beispiel 1: Vereinfachung von Kantenzügen

- viele Objekte in Landkarten werden durch Polygone oder Kantenzüge repräsentiert
- Detailgrad abhängig vom Maßstab (Generalisierung)



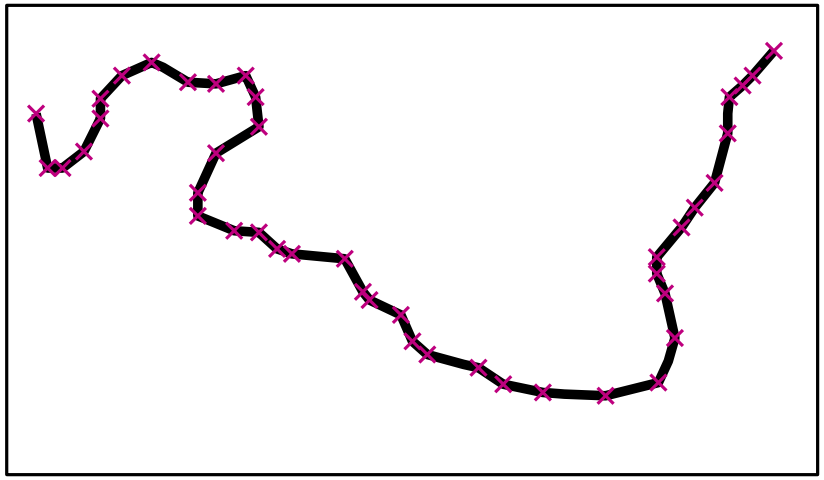
Maßstab $1 : X$



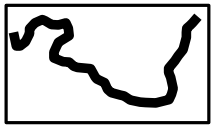
Maßstab $1 : (4X)$

Beispiel 1: Vereinfachung von Kantenzügen

- viele Objekte in Landkarten werden durch Polygone oder Kantenzüge repräsentiert
- Detailgrad abhängig vom Maßstab (Generalisierung)



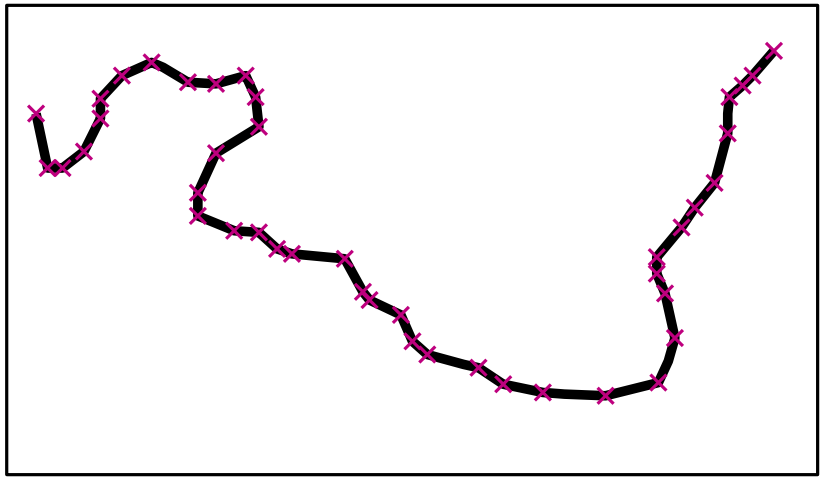
Maßstab $1 : X$



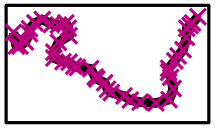
Maßstab $1 : (4X)$

Beispiel 1: Vereinfachung von Kantenzügen

- viele Objekte in Landkarten werden durch Polygone oder Kantenzüge repräsentiert
- Detailgrad abhängig vom Maßstab (Generalisierung)



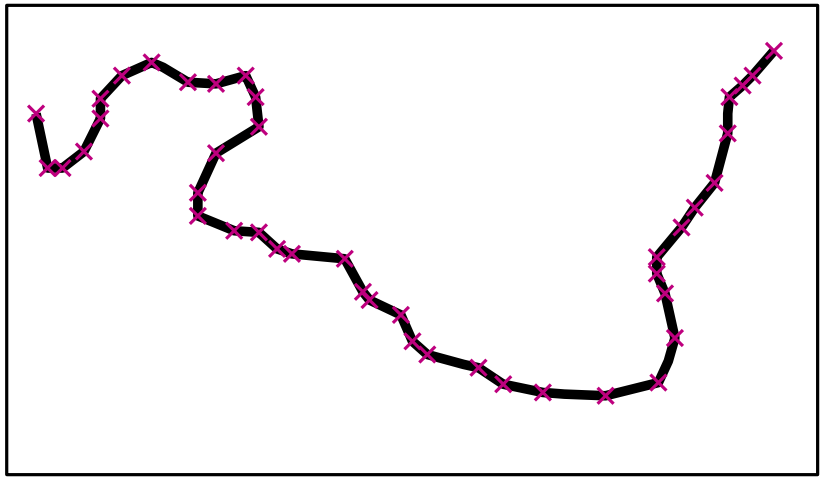
Maßstab $1 : X$



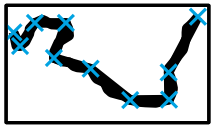
Maßstab $1 : (4X)$

Beispiel 1: Vereinfachung von Kantenzügen

- viele Objekte in Landkarten werden durch Polygone oder Kantenzüge repräsentiert
- Detailgrad abhängig vom Maßstab (Generalisierung)



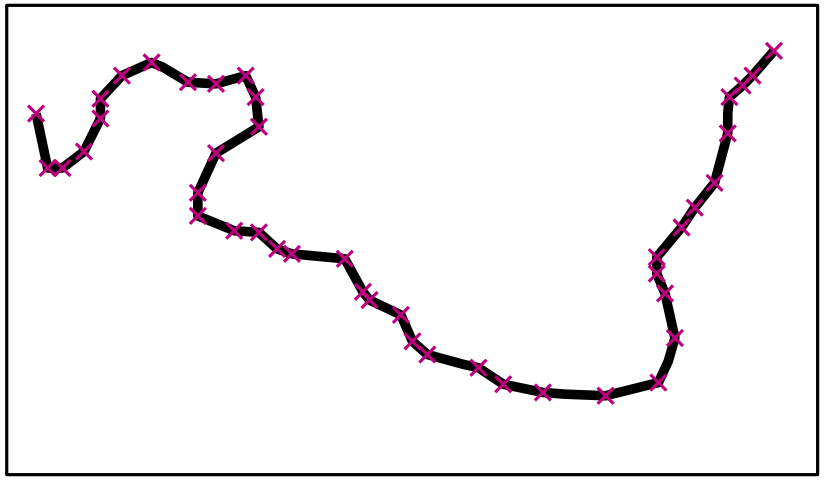
Maßstab $1 : X$



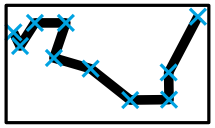
Maßstab $1 : (4X)$

Beispiel 1: Vereinfachung von Kantenzügen

- viele Objekte in Landkarten werden durch Polygone oder Kantenzüge repräsentiert
- Detailgrad abhängig vom Maßstab (Generalisierung)



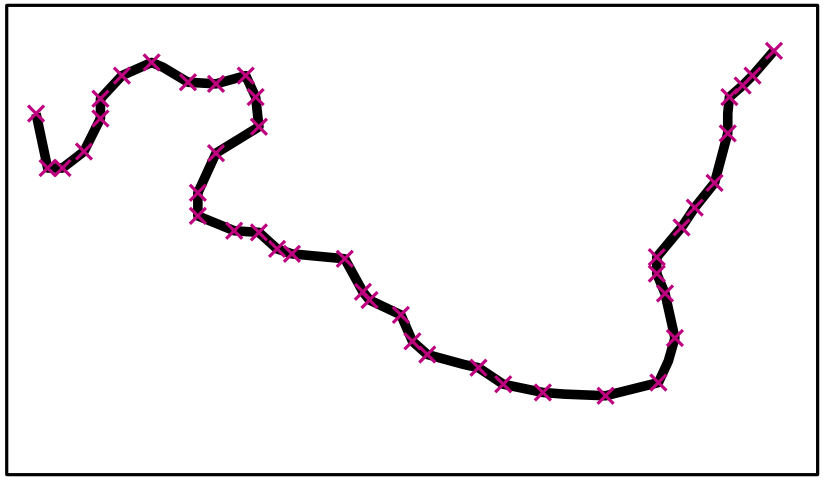
Maßstab $1 : X$



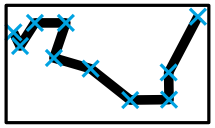
Maßstab $1 : (4X)$

Beispiel 1: Vereinfachung von Kantenzügen

- viele Objekte in Landkarten werden durch Polygone oder Kantenzüge repräsentiert
- Detailgrad abhängig vom Maßstab (Generalisierung)



Maßstab $1 : X$



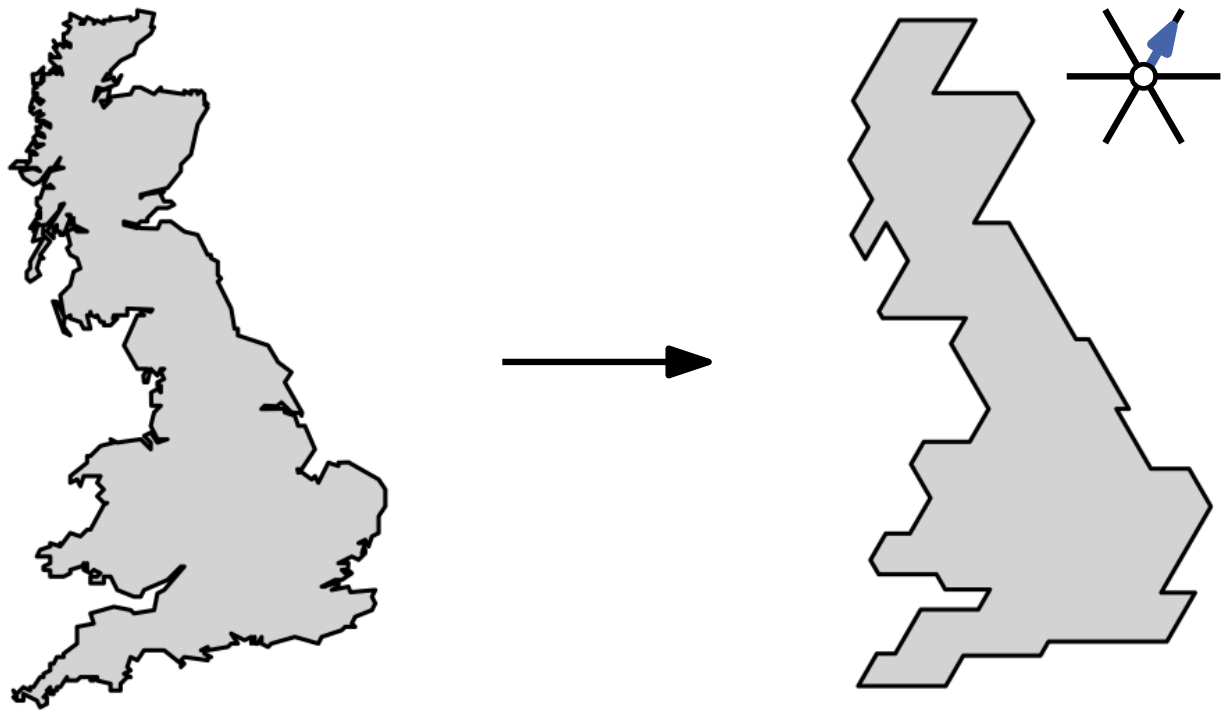
Maßstab $1 : (4X)$

geg: Kantenzug P

ges: Kantenzug Q mit weniger Knoten und kleinem Fehler $|P - Q|$

Beispiel 2: Schematisierung von Polygonen

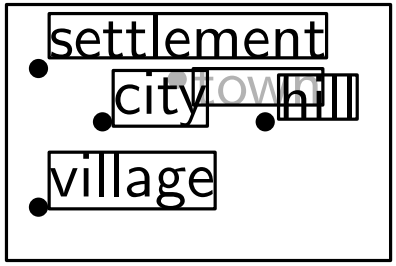
- schematische Karten nutzen oft eingeschränkte Kantenrichtungen
- Polygonflächen müssen geeignet schematisiert werden



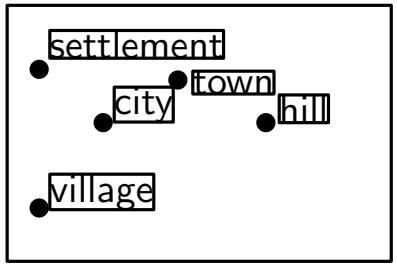
geg: Polygon P mit Fläche A, Richtungsmenge $\mathcal{C}$
ges: $\mathcal{C}$-orientiertes Polygon Q mit gleicher Fläche A und kleinem Fehler $|P - Q|$

Beispiel 3: Beschriftung von Landkarten

- Objekte in Karten benötigen meist einen eindeutig zugeordneten Namen (Label)
- verschiedene interne und externe Labelpositionen möglich

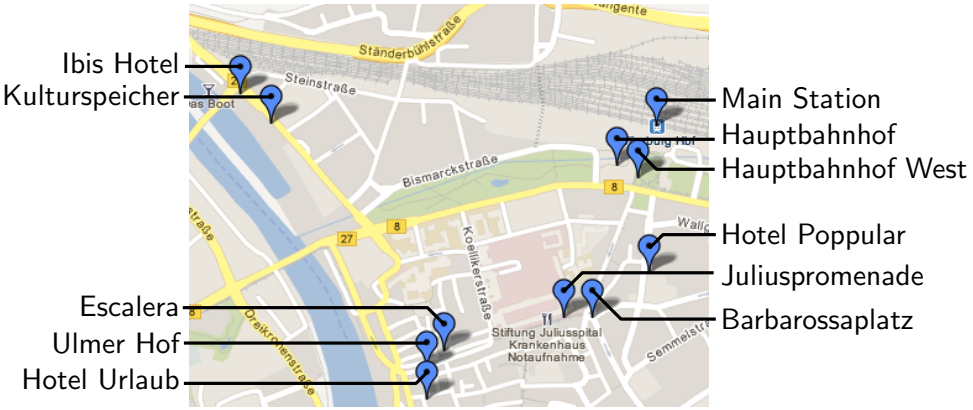


MaxNumber



MaxSize

interne Label



externe Label

geg: Punktmenge P mit Labelmenge L

ges: gültige, optimale Beschriftung von P mit L, je nach Beschriftungsmodell

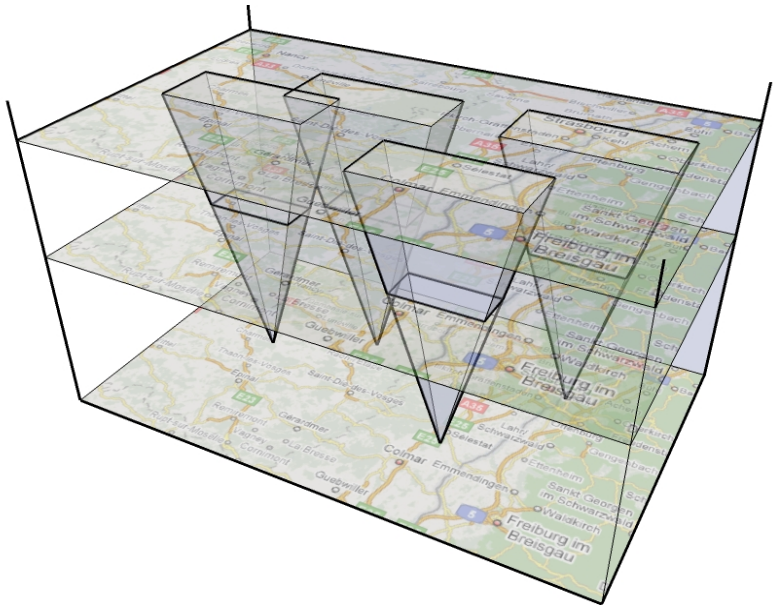
Beispiel 4: Dynamische Karten

- moderne elektronische Karten sind interaktiv & dynamisch
- Formen und Beschriftungen müssen sich kontinuierlich an die Kartenansicht anpassen

z.B. Beschriftung:

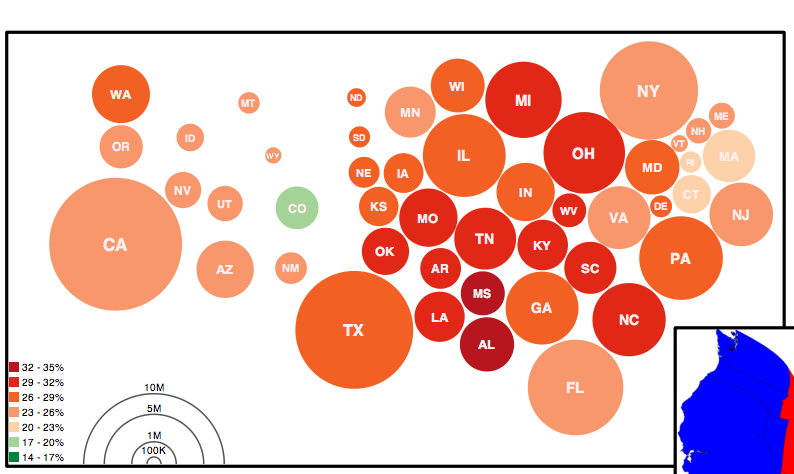
geg: Punktmenge P mit Labelmenge L

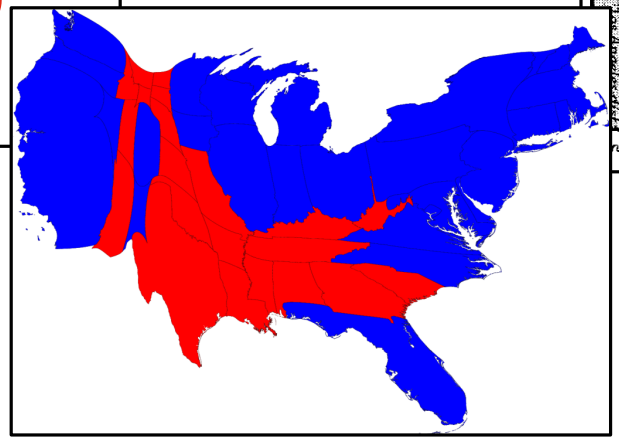
ges: gültige, optimale und **konsistente** Beschriftung von P mit L unter Zoom, Drehen etc.

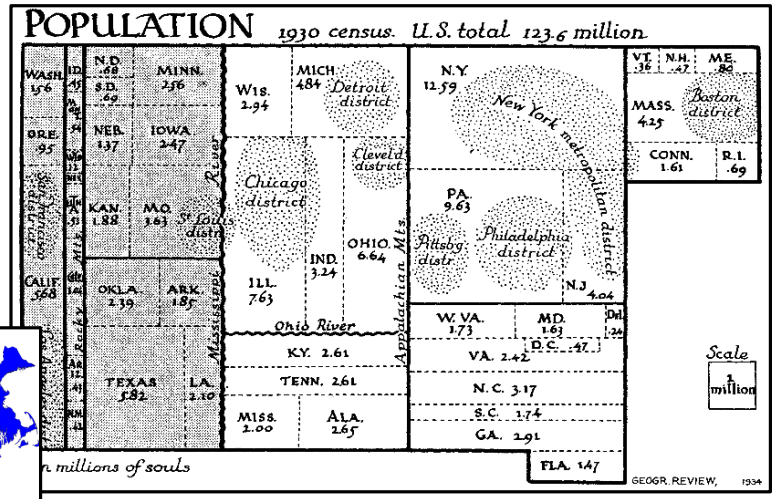


Beispiel 5: Flächenkartogramme

- abstrakte statistische thematische Karten nutzen u.a. verzerrte proportionale Flächen
- proportionale Kontaktrepräsentation des Dualgraphs







geg: gewichtete politische Karte (Unterteilung der Ebene)
ges: entsprechende verzerrte Karte, deren Flächen proportional zu den Gewichten sind

Vorlesungsthemen

Ziel: Überblick über aktuelle algorithmische Forschungsthemen in der Kartografie

Themenliste (vorläufig und unvollständig)

- Linien- und Polygonvereinfachung
- Schematisierung von Linien und Polygonen
- Beschriftung von Landkarten (intern, extern)
- Algorithmen für dynamische Landkarten
- Flächenkartogramme
- Flächenaggregation
- Flusskarten
- Algorithmen für Terrainmodelle
- Gebäudegeneralisierung
- Map Matching
- ...

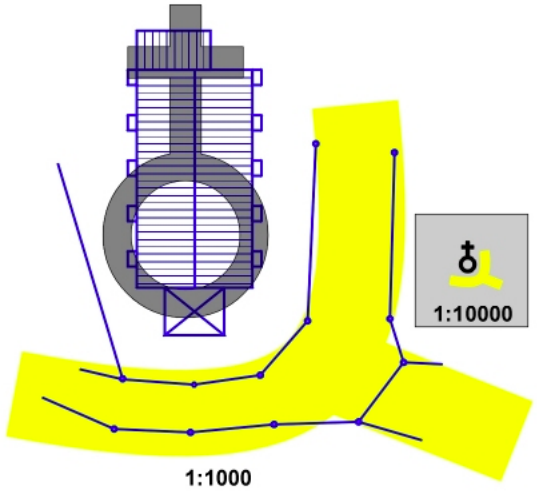
Algorithmen zur Linienvereinfachung

Generalisierung

Generalisierung in der Kartografie bezeichnet Verfahren zur Vereinfachung des Karteninhalts für die Verbesserung der Lesbarkeit in kleineren Maßstäben.

Beispiele:

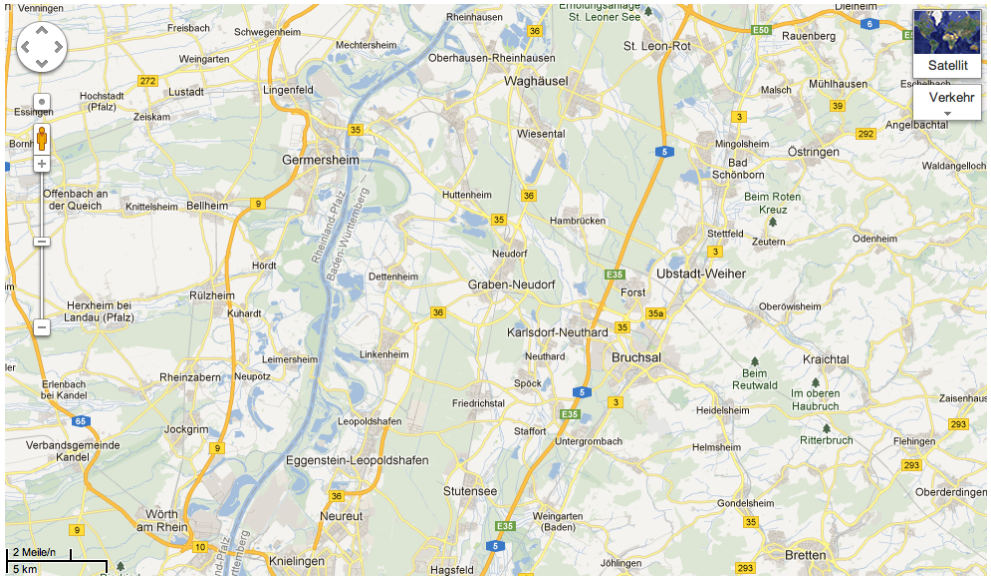
- Auswahl/Filtern von Objekten
- Vereinfachung
- Symbolisierung
- Aggregation
- Glättung
- Vergrößerung
- Verdrängung
- ...



Quelle: Wikipedia

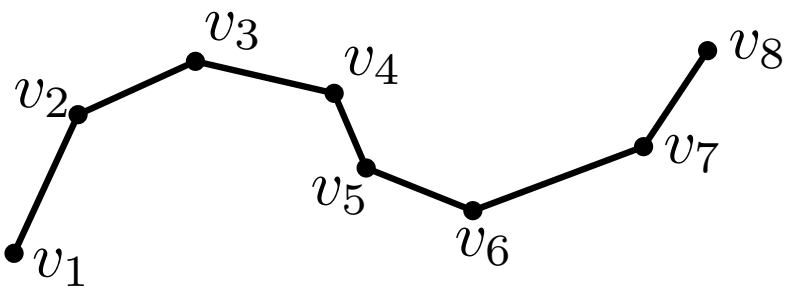
Linienvereinfachung

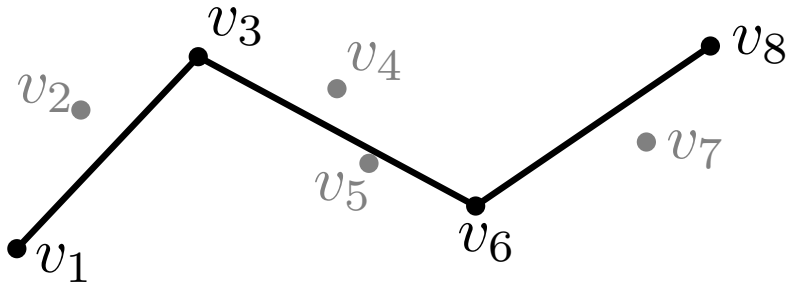
- Karten bestehen zum Großteil aus Polygonzügen
- Linienvereinfachung reduziert Komplexität: visuell und Speicherplatz



geg: Polygonzug $P = (v_1, v_2, \ldots, v_n)$

ges: Polygonzug $Q = (v_1 = v_{i_1}, v_{i_2}, v_{i_3}, \ldots, v_{i_k} = v_n)$ mit $i_1 < i_2 < \cdots < i_k$, so dass Q eine gute Approximation von P und k möglichst klein ist





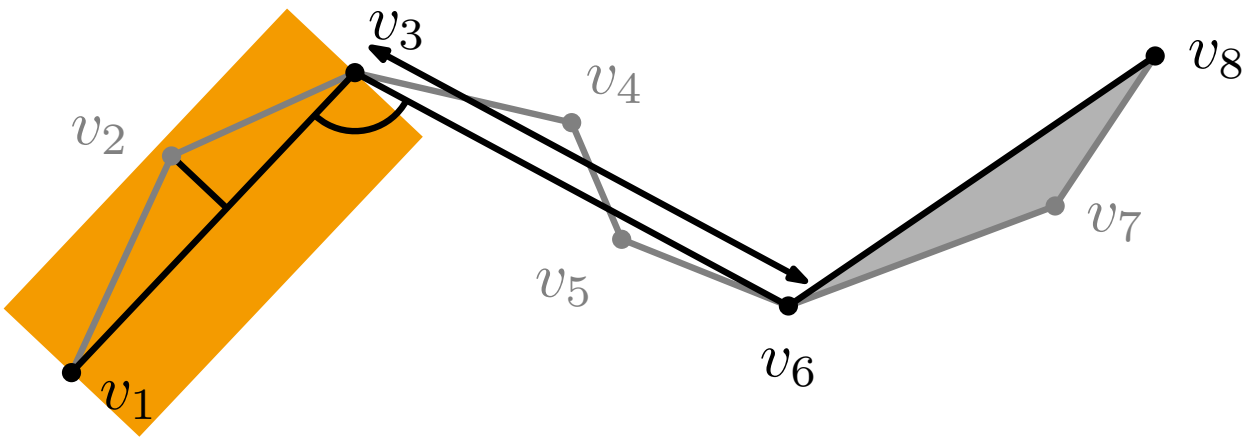
Qualitätskriterien

Überlegen Sie sich mögliche Kriterien für die Approximationsqualität!

Qualitätskriterien

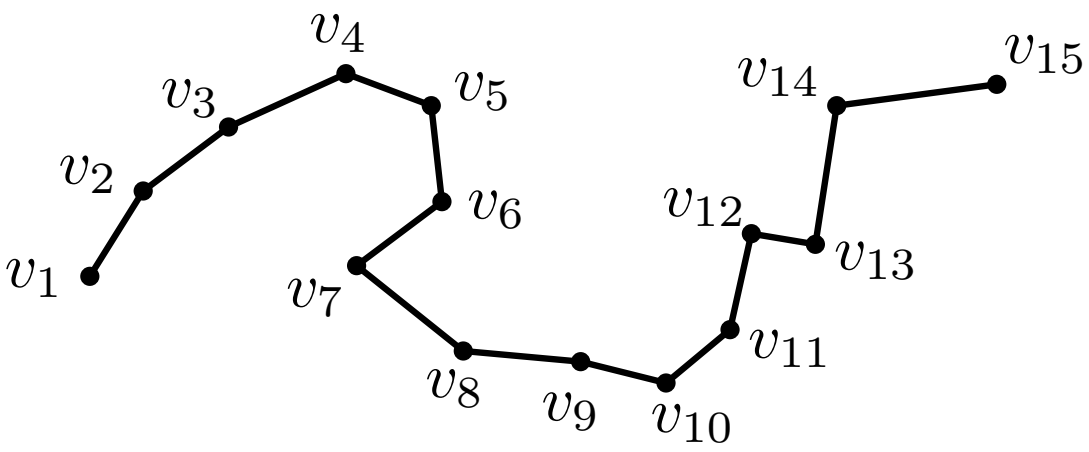
Überlegen Sie sich mögliche Kriterien für die Approximationsqualität!

- ähnliche Länge
- ähnliche Winkel
- geringer Abstand
- ε-Korridor
- kleine Flächenänderung



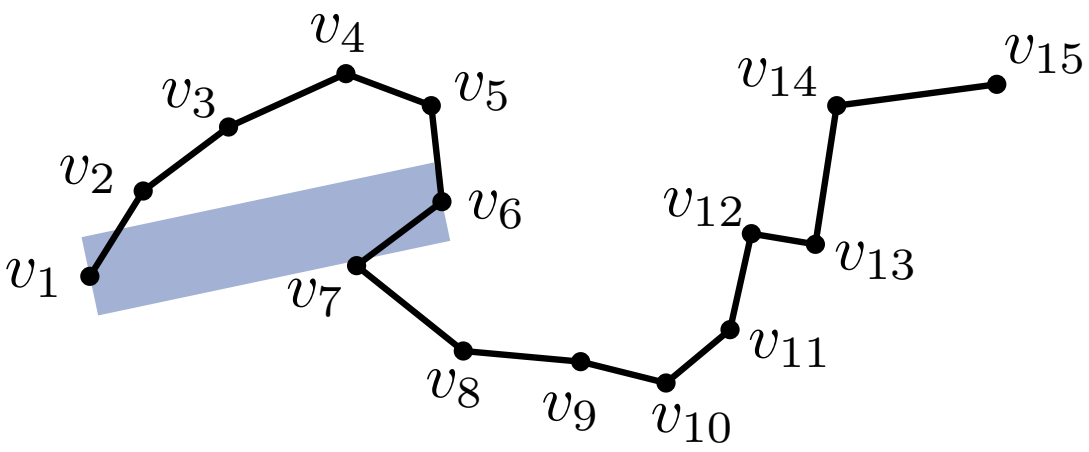
Erste Algorithmen

lokale Suche mit look-ahead s und ε-Korridor



Erste Algorithmen

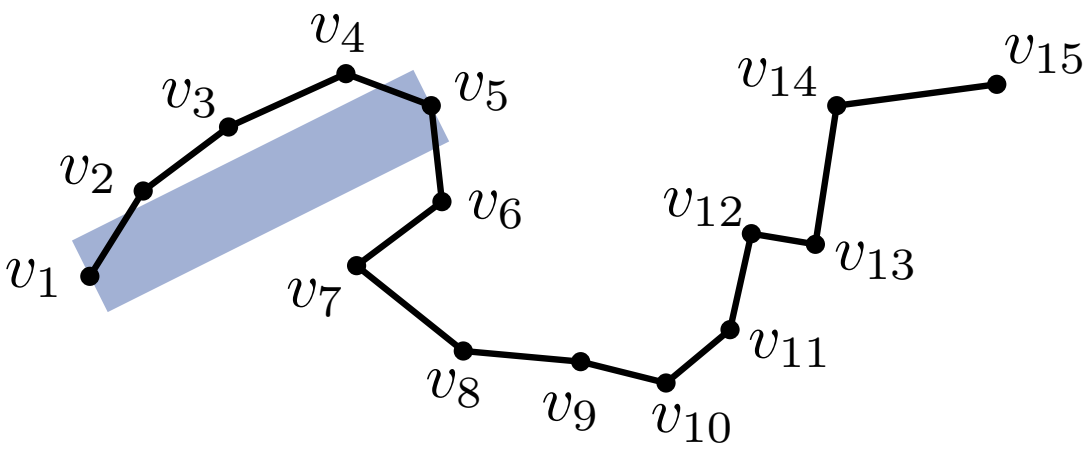
lokale Suche mit look-ahead s und ε-Korridor



$s = 5$

Erste Algorithmen

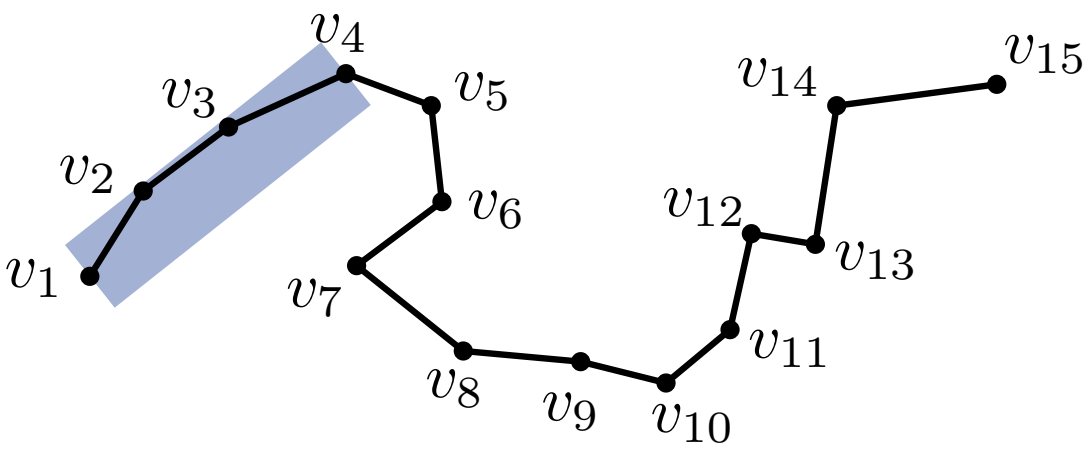
lokale Suche mit look-ahead s und ε-Korridor



$s = 5$

Erste Algorithmen

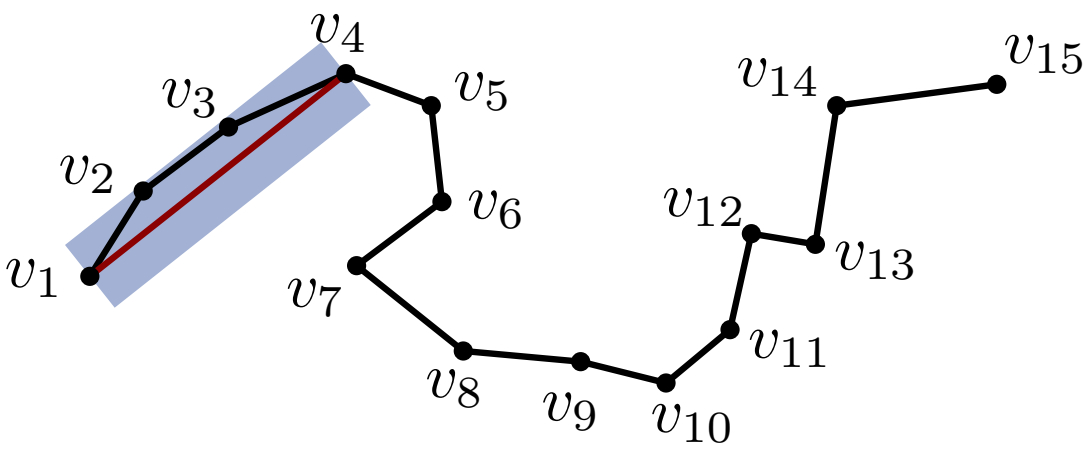
lokale Suche mit look-ahead s und ε-Korridor



$s = 5$

Erste Algorithmen

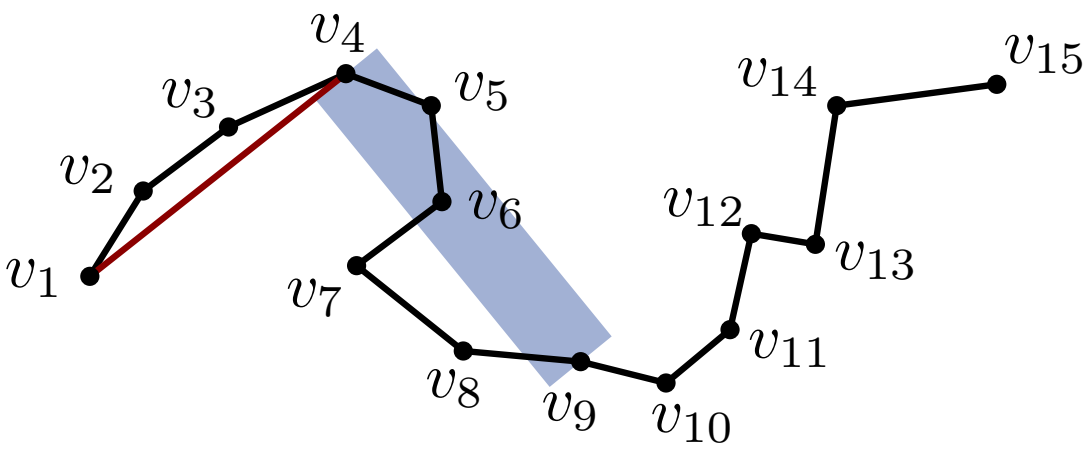
lokale Suche mit look-ahead s und ε-Korridor



$s = 5$

Erste Algorithmen

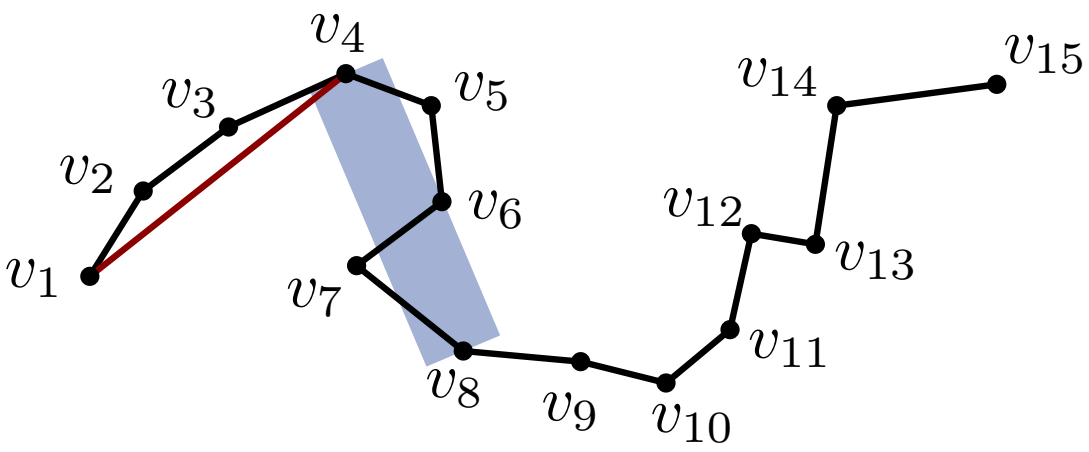
lokale Suche mit look-ahead s und ε-Korridor



$s = 5$

Erste Algorithmen

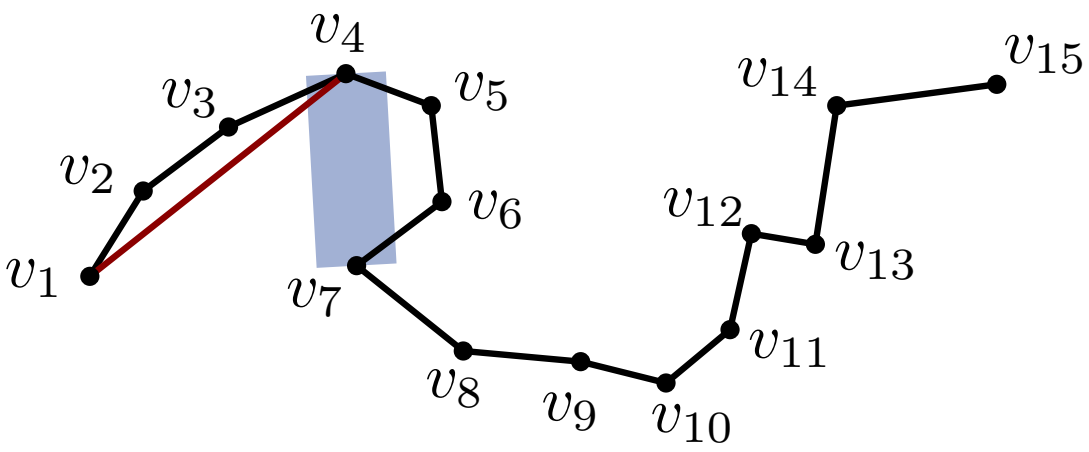
lokale Suche mit look-ahead s und ε-Korridor



$s = 5$

Erste Algorithmen

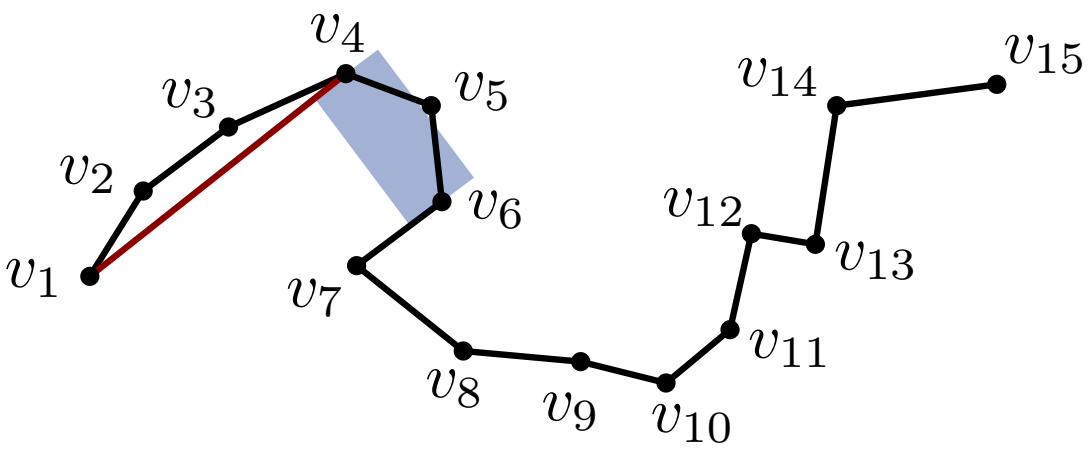
lokale Suche mit look-ahead s und ε-Korridor



$s = 5$

Erste Algorithmen

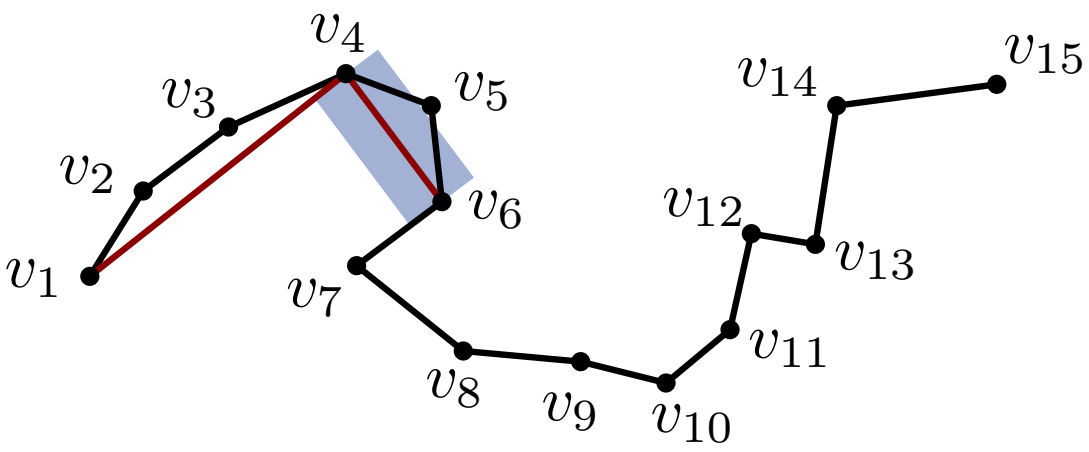
lokale Suche mit look-ahead s und ε-Korridor



$s = 5$

Erste Algorithmen

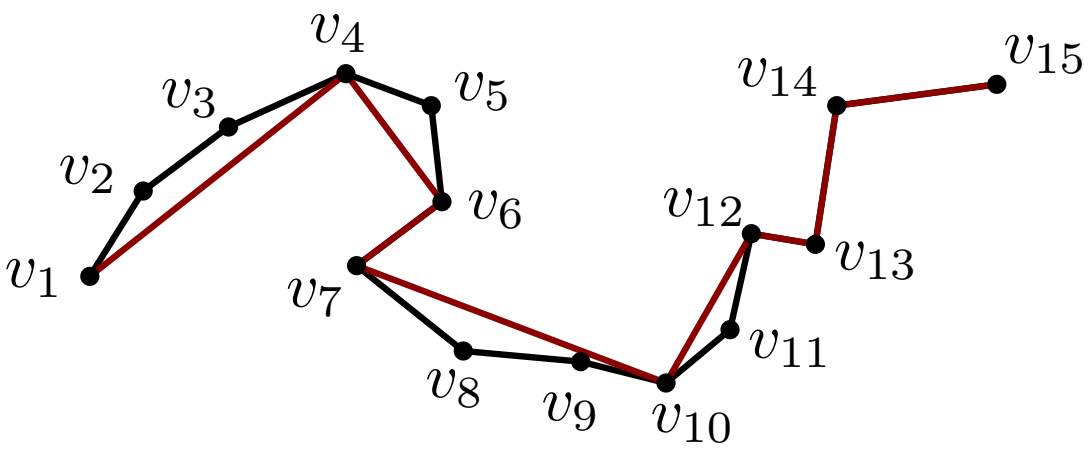
lokale Suche mit look-ahead s und ε-Korridor



$s = 5$

Erste Algorithmen

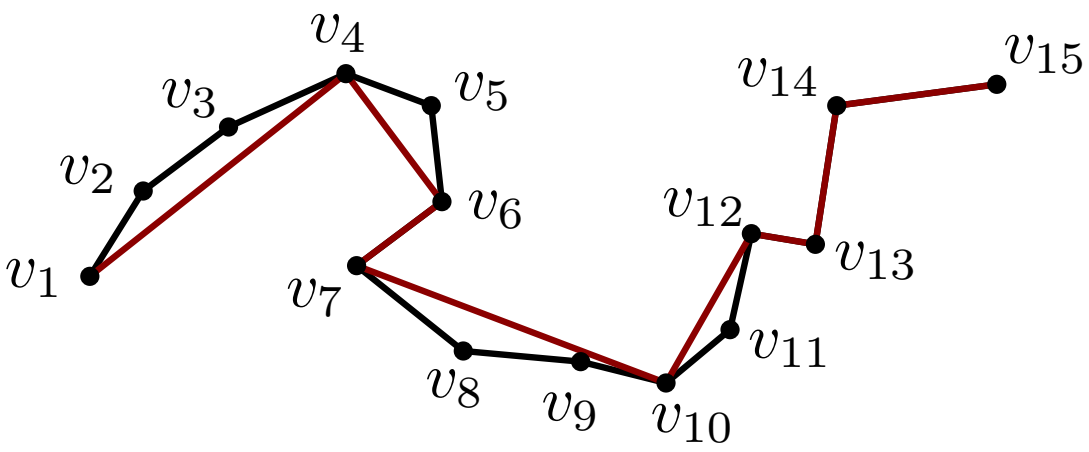
lokale Suche mit look-ahead s und ε-Korridor



$s = 5$

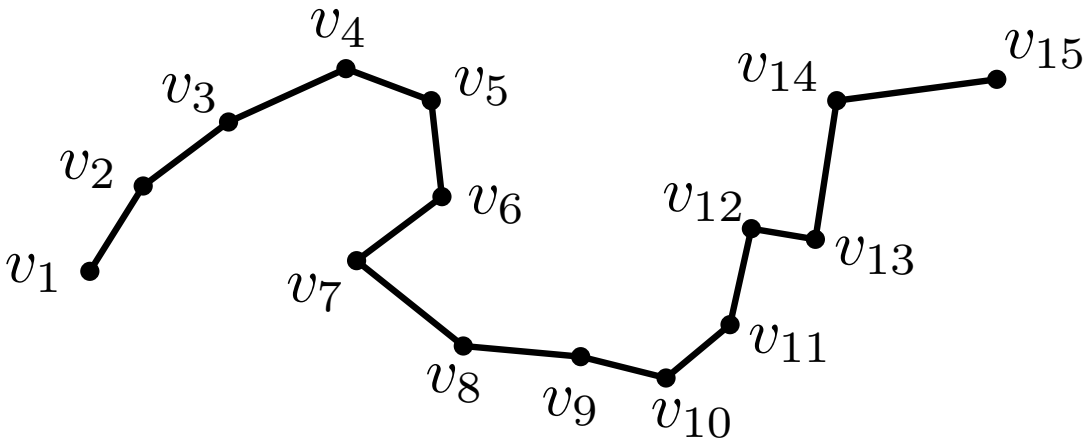
Erste Algorithmen

lokale Suche mit look-ahead s und ε-Korridor



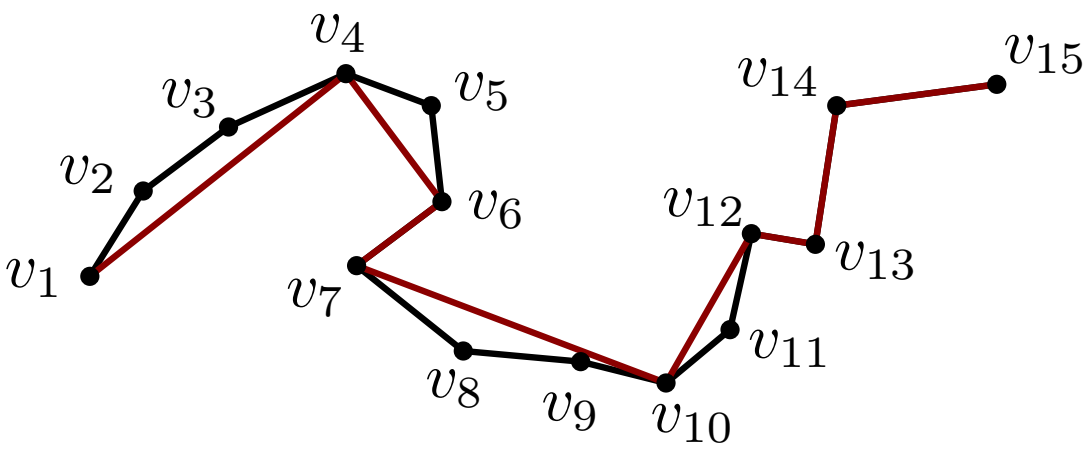
$s = 5$

unbeschränkte greedy Suche mit ε-Korridor



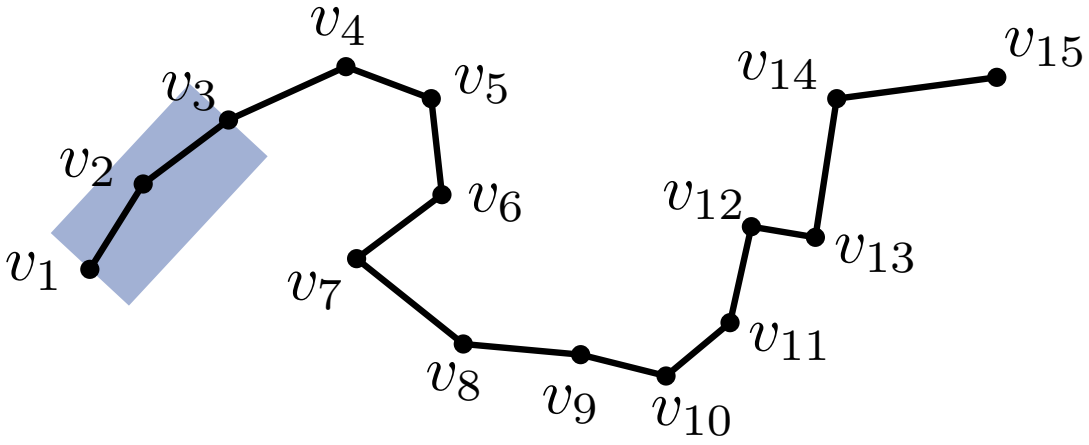
Erste Algorithmen

lokale Suche mit look-ahead s und ε-Korridor



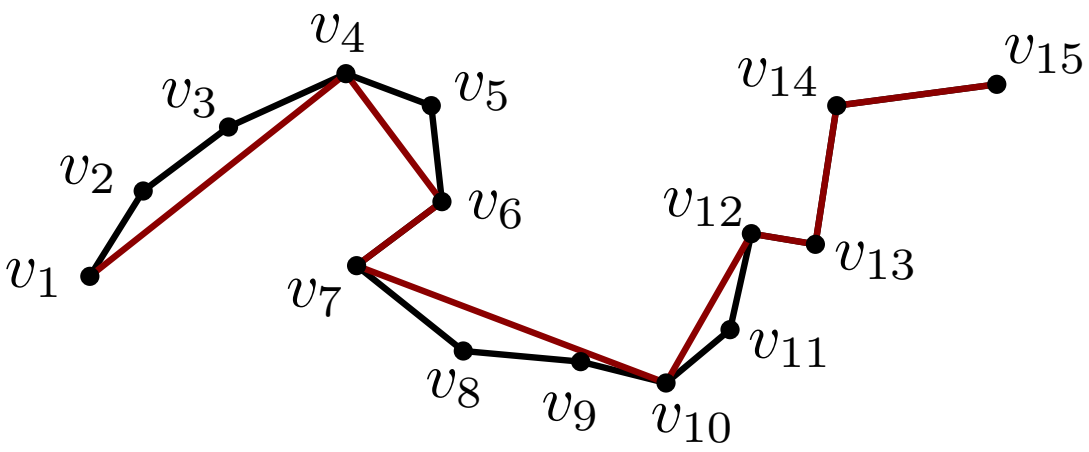
$s = 5$

unbeschränkte greedy Suche mit ε-Korridor



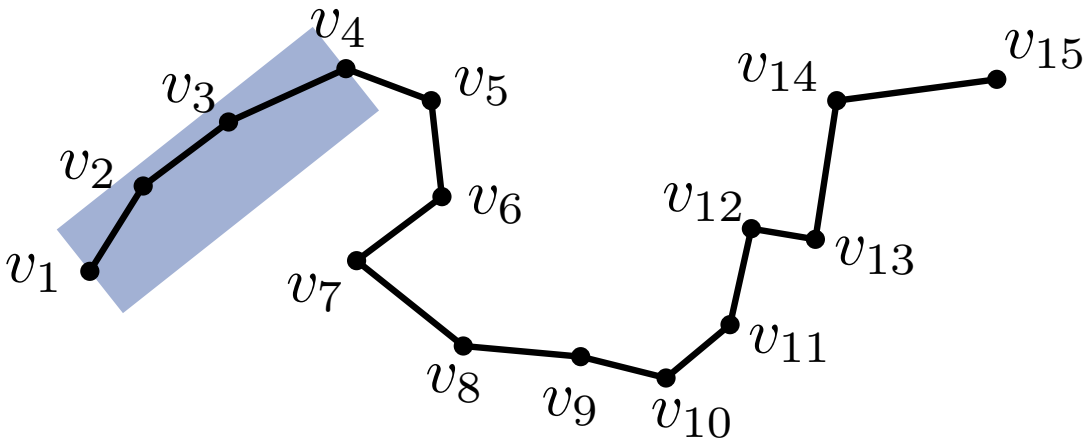
Erste Algorithmen

lokale Suche mit look-ahead s und ε-Korridor



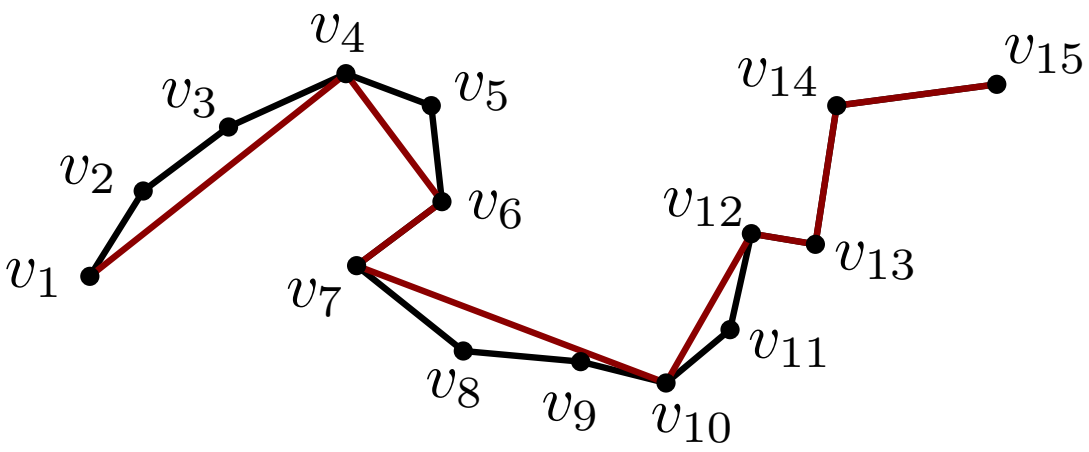
$s = 5$

unbeschränkte greedy Suche mit ε-Korridor



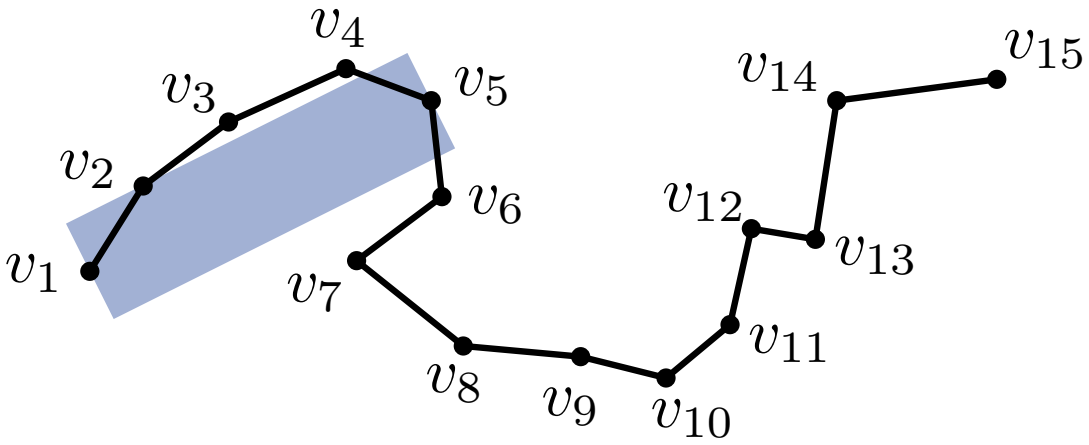
Erste Algorithmen

lokale Suche mit look-ahead s und ε-Korridor



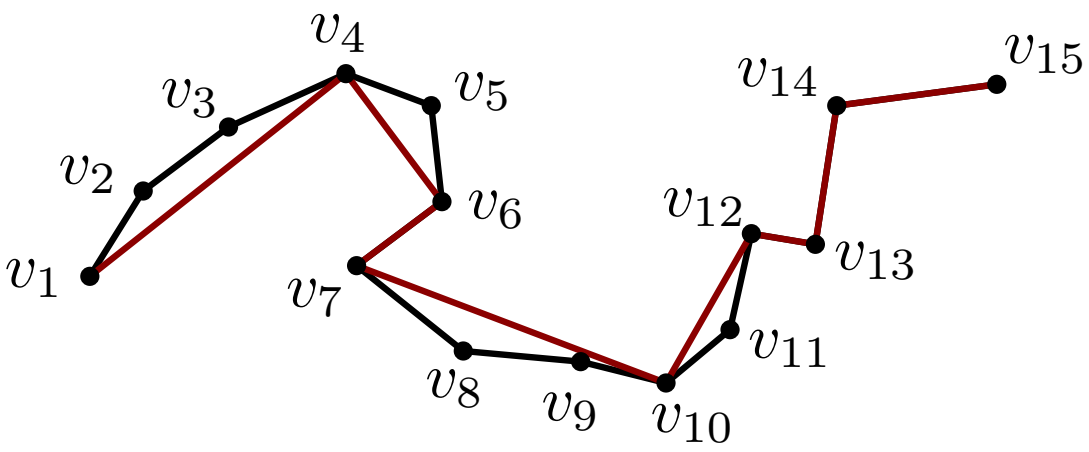
$s = 5$

unbeschränkte greedy Suche mit ε-Korridor



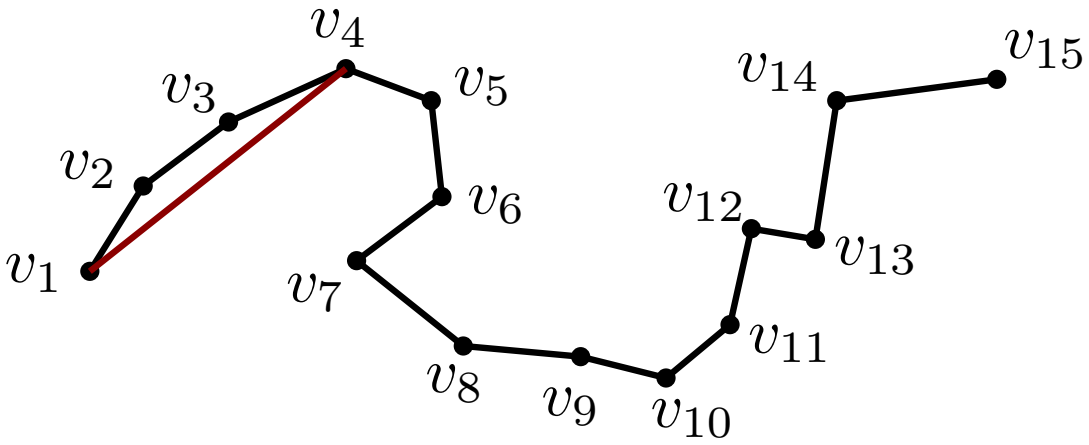
Erste Algorithmen

lokale Suche mit look-ahead s und ε-Korridor



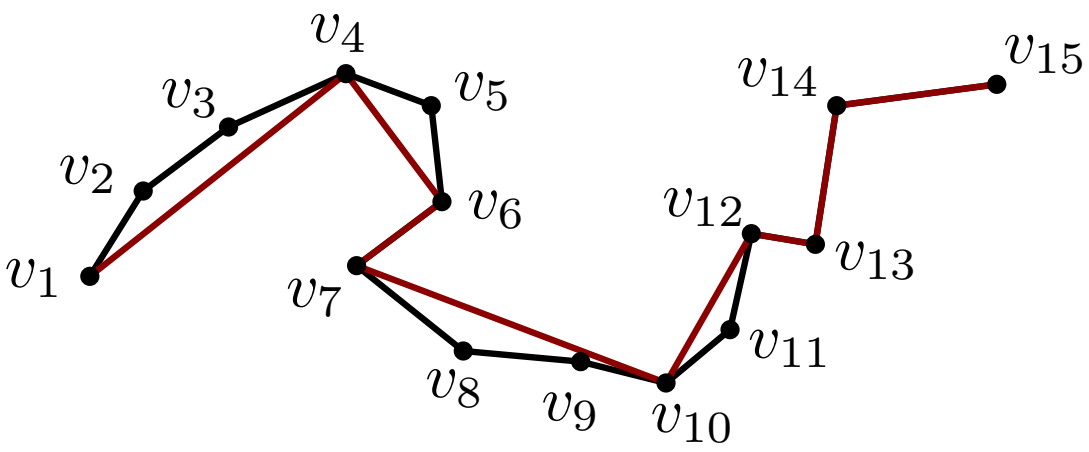
$s = 5$

unbeschränkte greedy Suche mit ε-Korridor



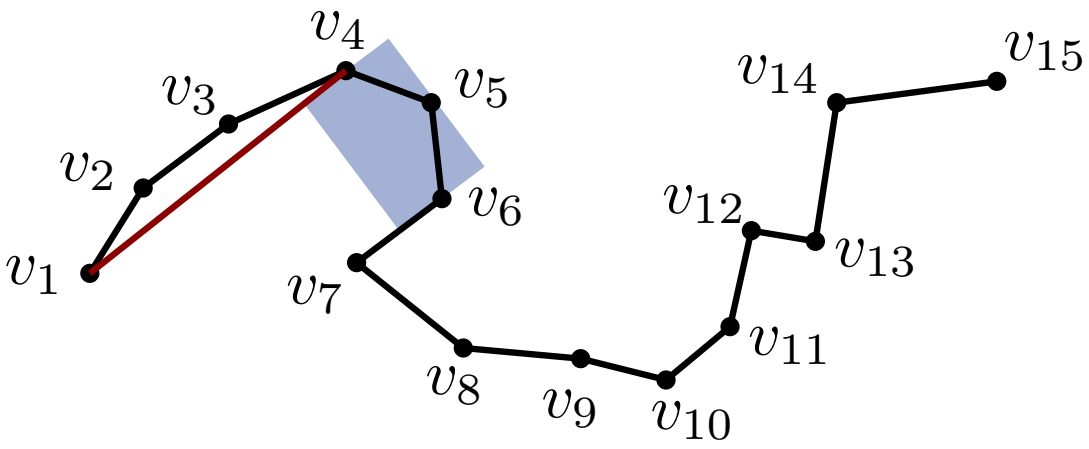
Erste Algorithmen

lokale Suche mit look-ahead s und ε-Korridor



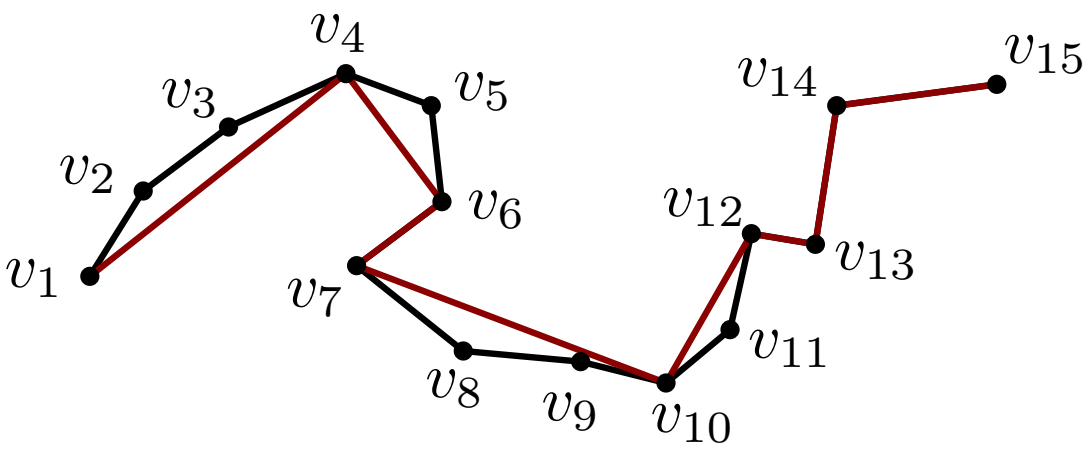
$s = 5$

unbeschränkte greedy Suche mit ε-Korridor



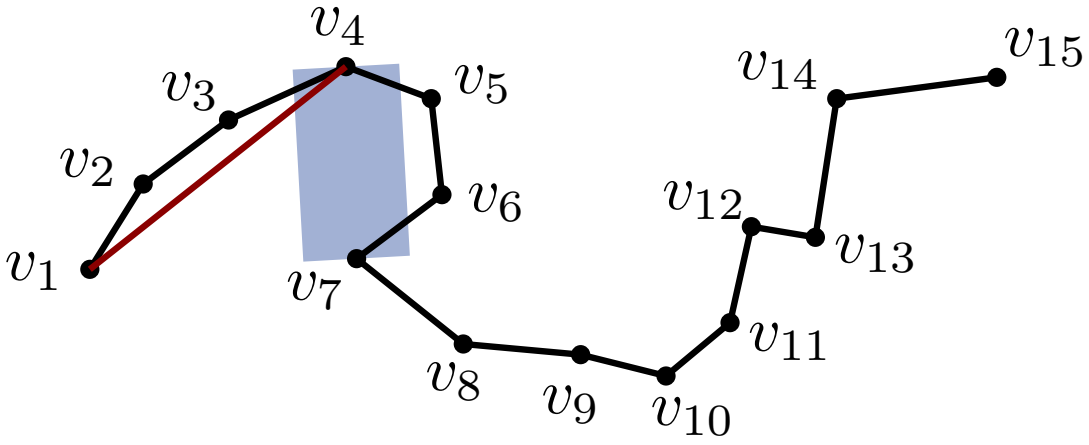
Erste Algorithmen

lokale Suche mit look-ahead s und ε-Korridor



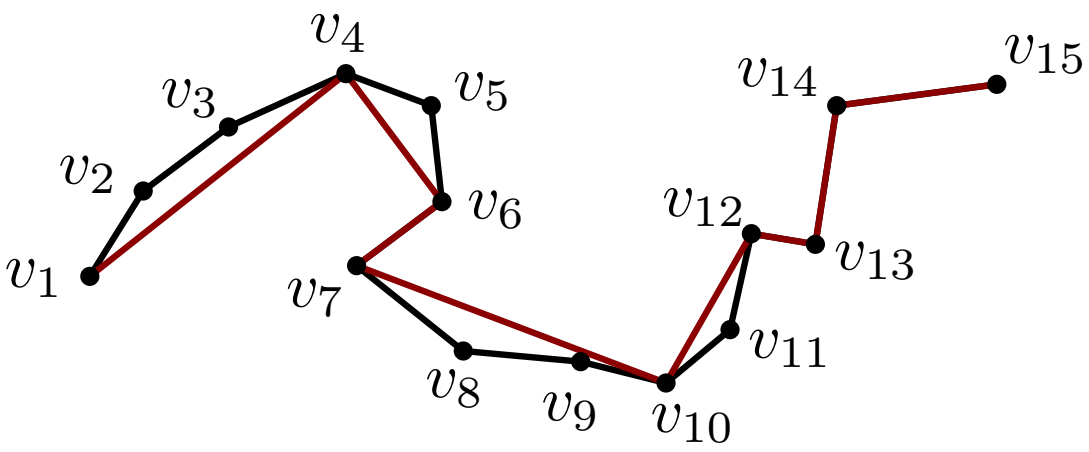
$s = 5$

unbeschränkte greedy Suche mit ε-Korridor



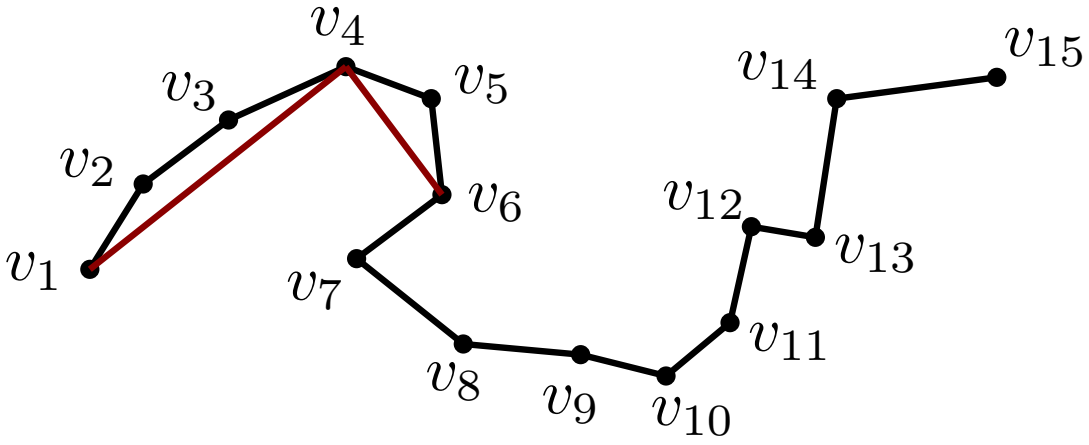
Erste Algorithmen

lokale Suche mit look-ahead s und ε-Korridor



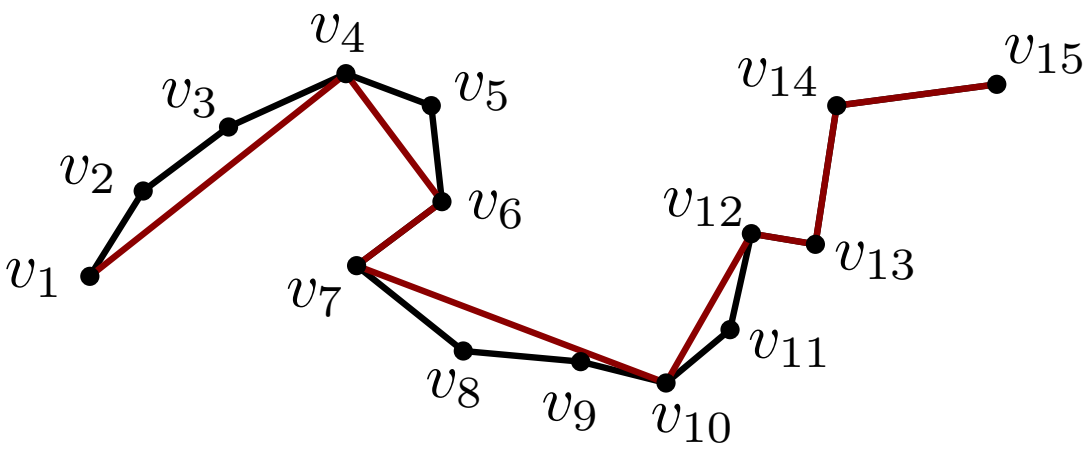
$s = 5$

unbeschränkte greedy Suche mit ε-Korridor



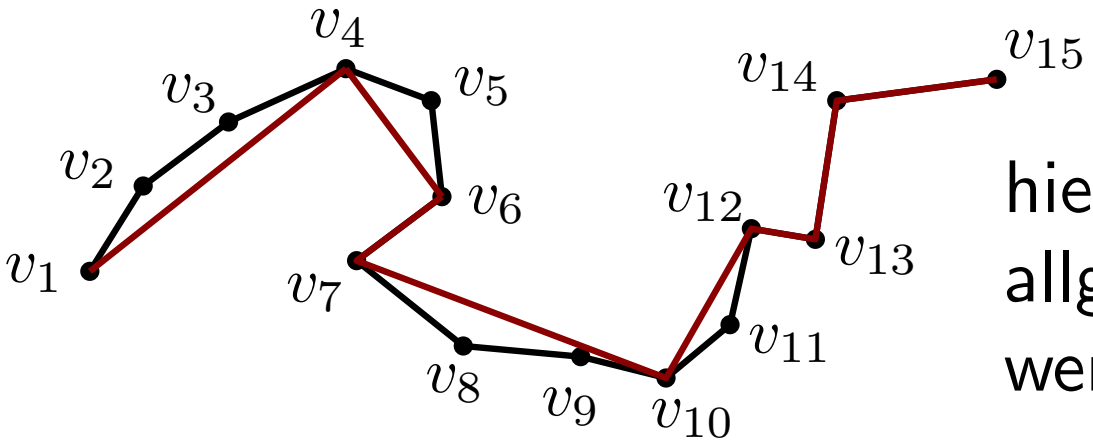
Erste Algorithmen

lokale Suche mit look-ahead s und ε-Korridor



$s = 5$

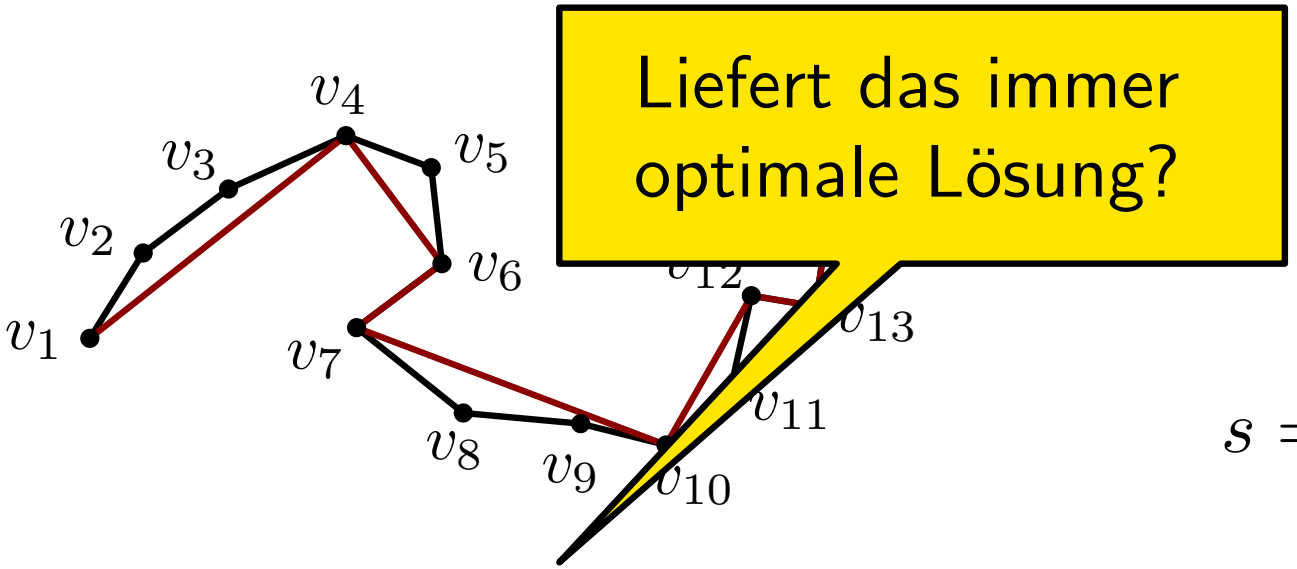
unbeschränkte greedy Suche mit ε-Korridor



hier identisch,
allg. jedoch
weniger Segmente

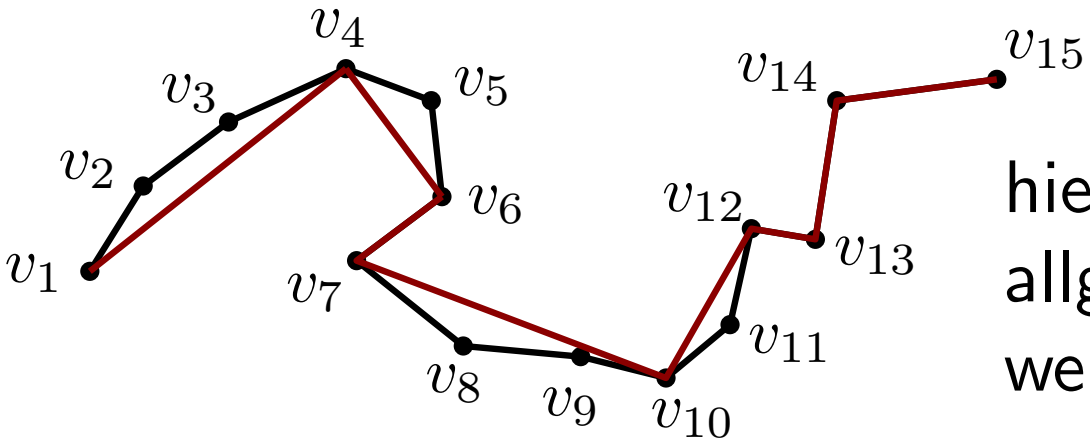
Erste Algorithmen

lokale Suche mit look-ahead s und ε-Korridor



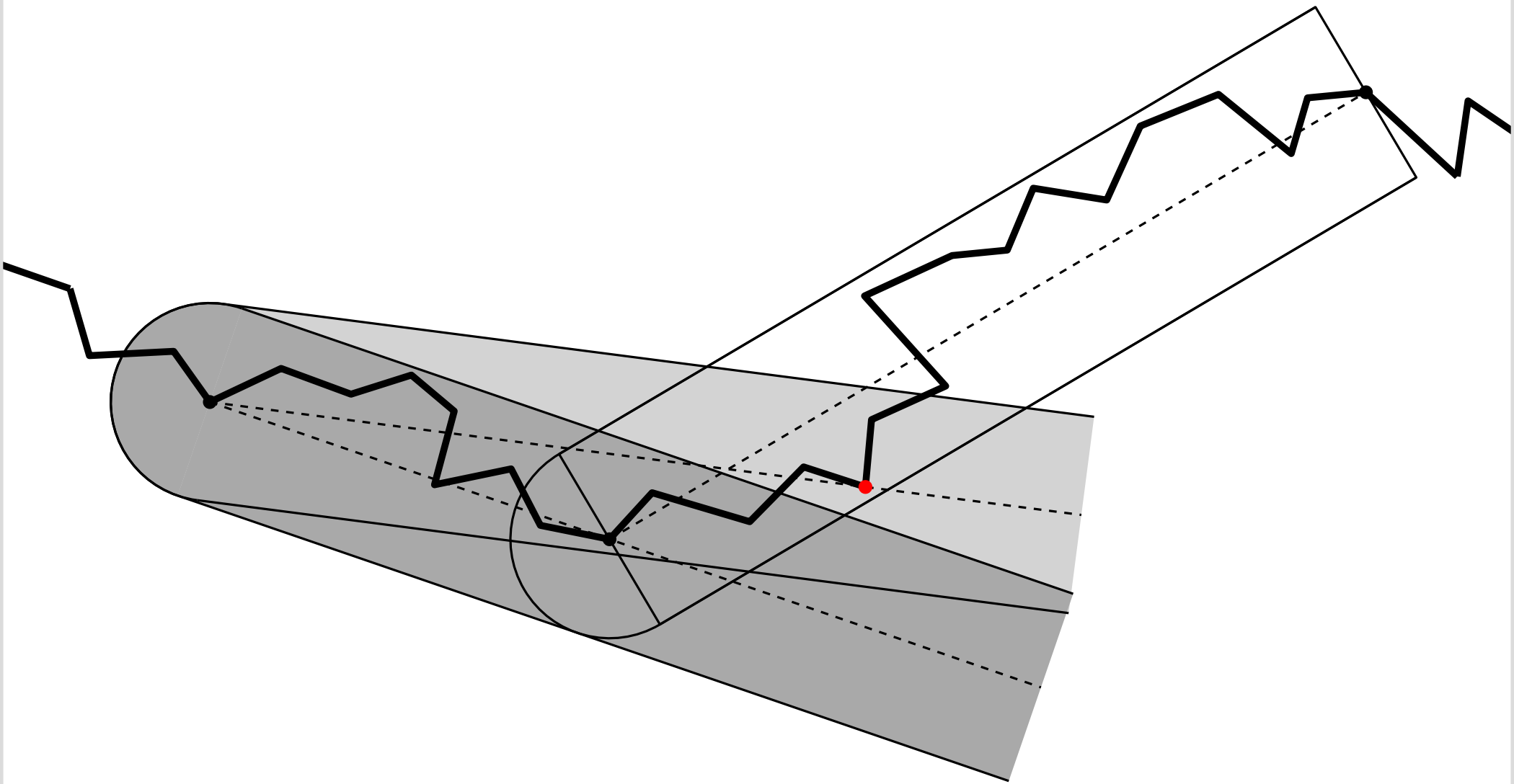
$s = 5$

unbeschränkte greedy Suche mit ε-Korridor

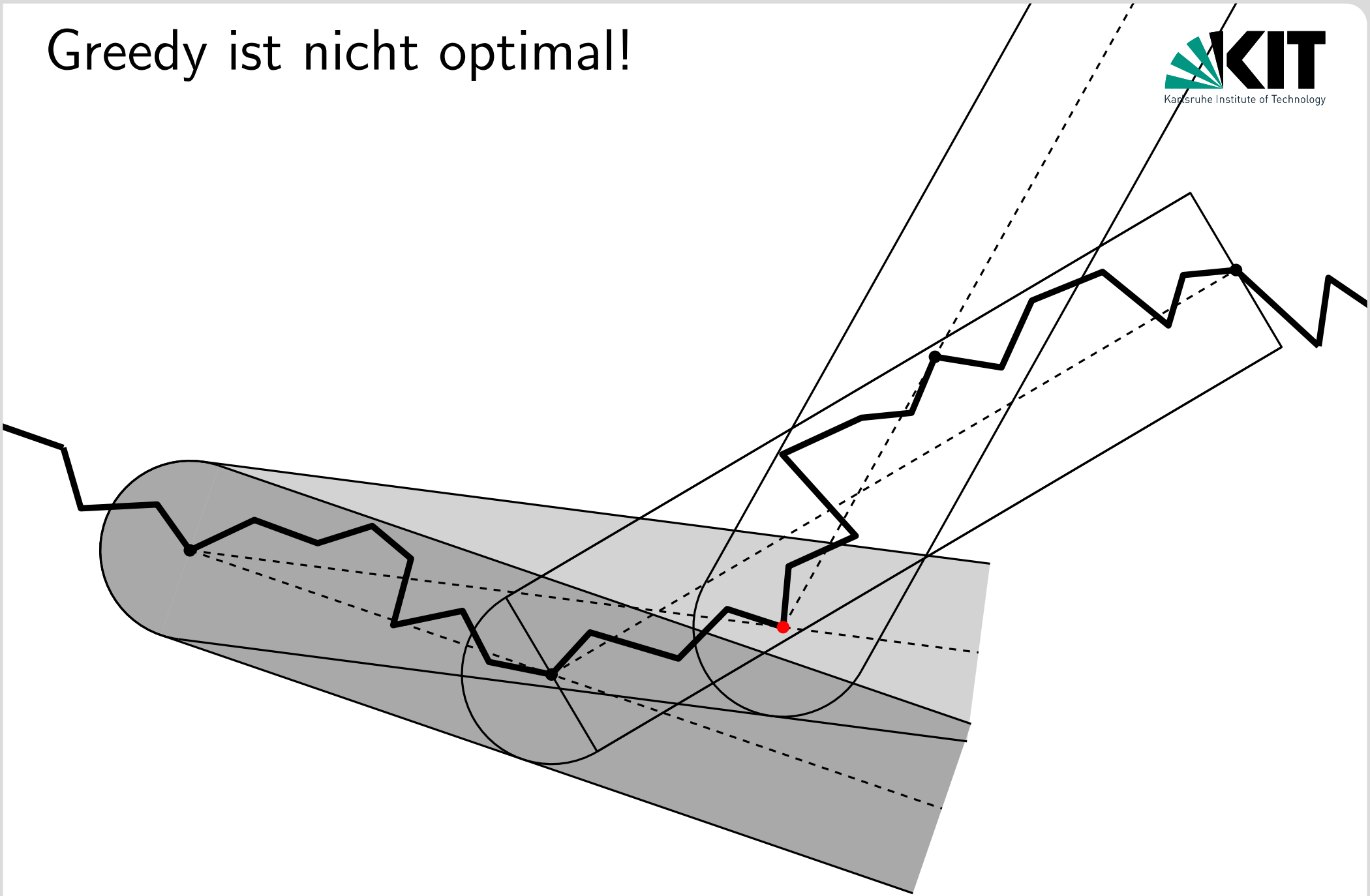


hier identisch, allg. jedoch weniger Segmente

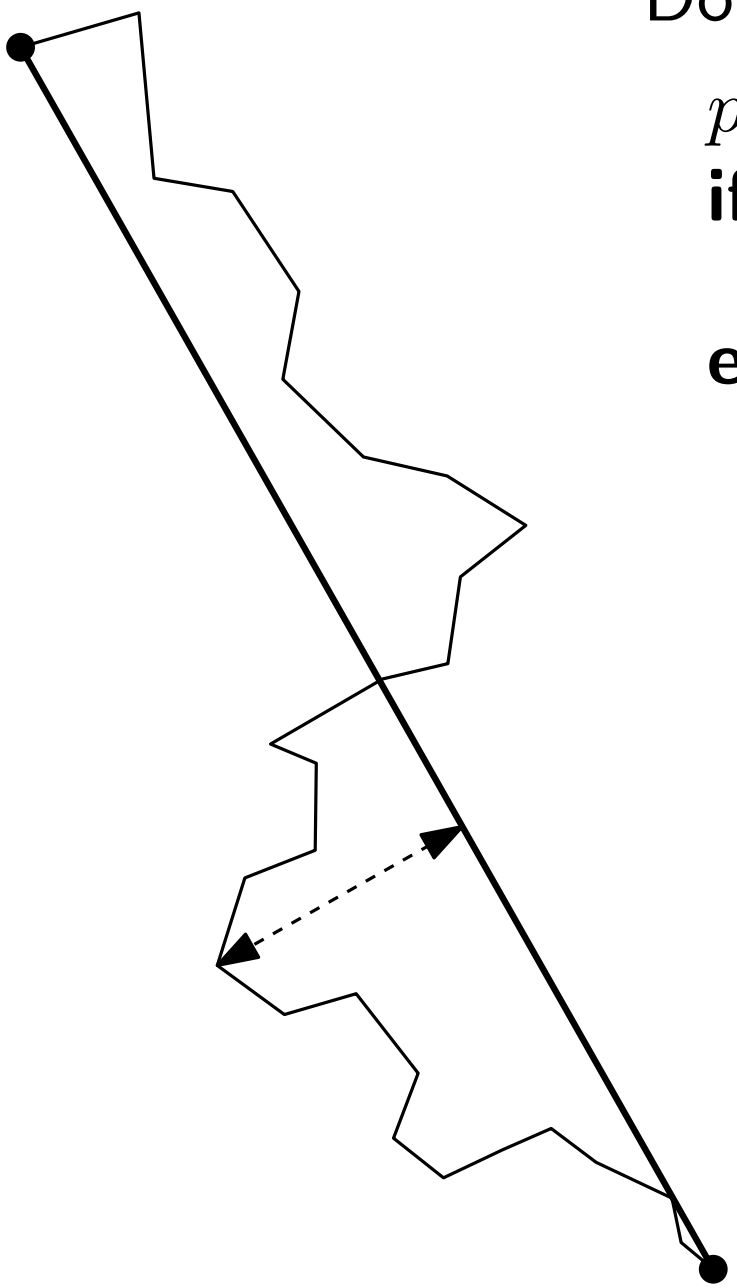
Greedy ist nicht optimal!



Greedy ist nicht optimal!



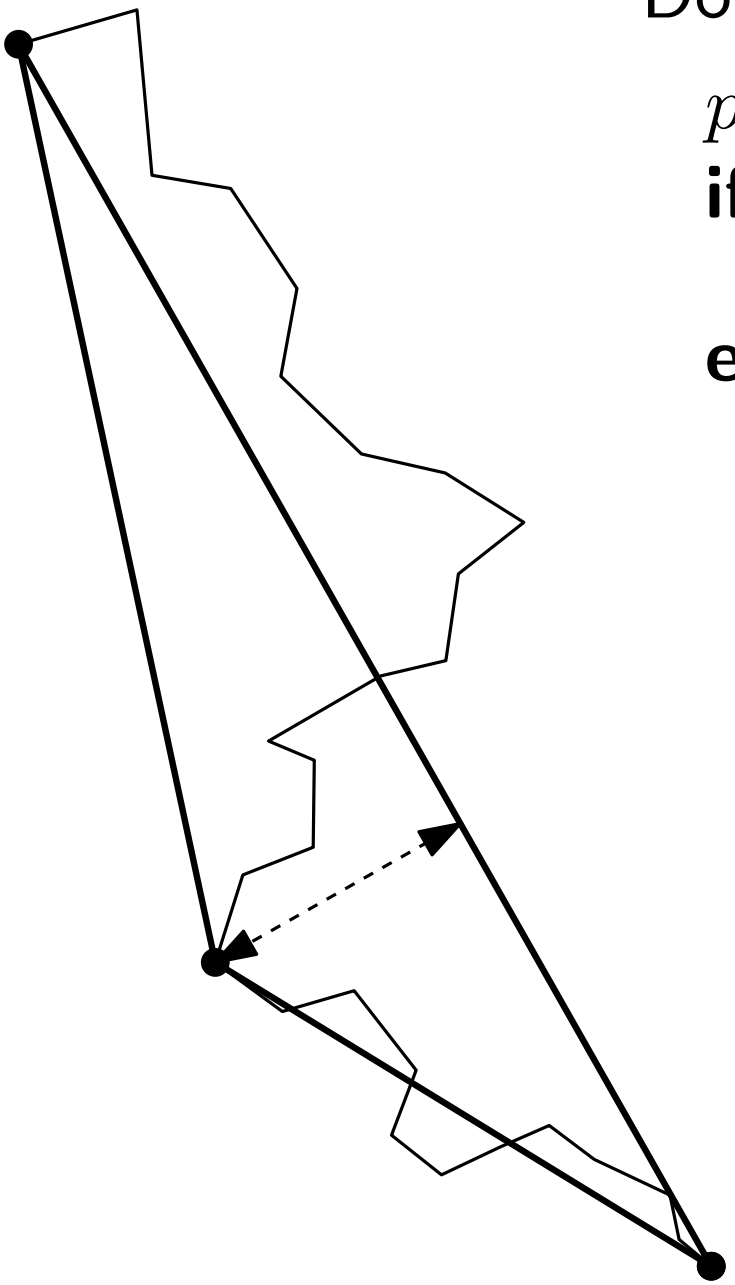
Douglas-Peucker Algorithmus [1973]



DouglasPeucker(P, i, j)

$p_f \leftarrow$ weitester Knoten von $\overline{p_i p_j}$
if dist$(\overline{p_i p_j}, p_f) < \varepsilon$ **then**
  return $\overline{p_i p_j}$
else
  DouglasPeucker(P, i, f)
  DouglasPeucker(P, f, j)

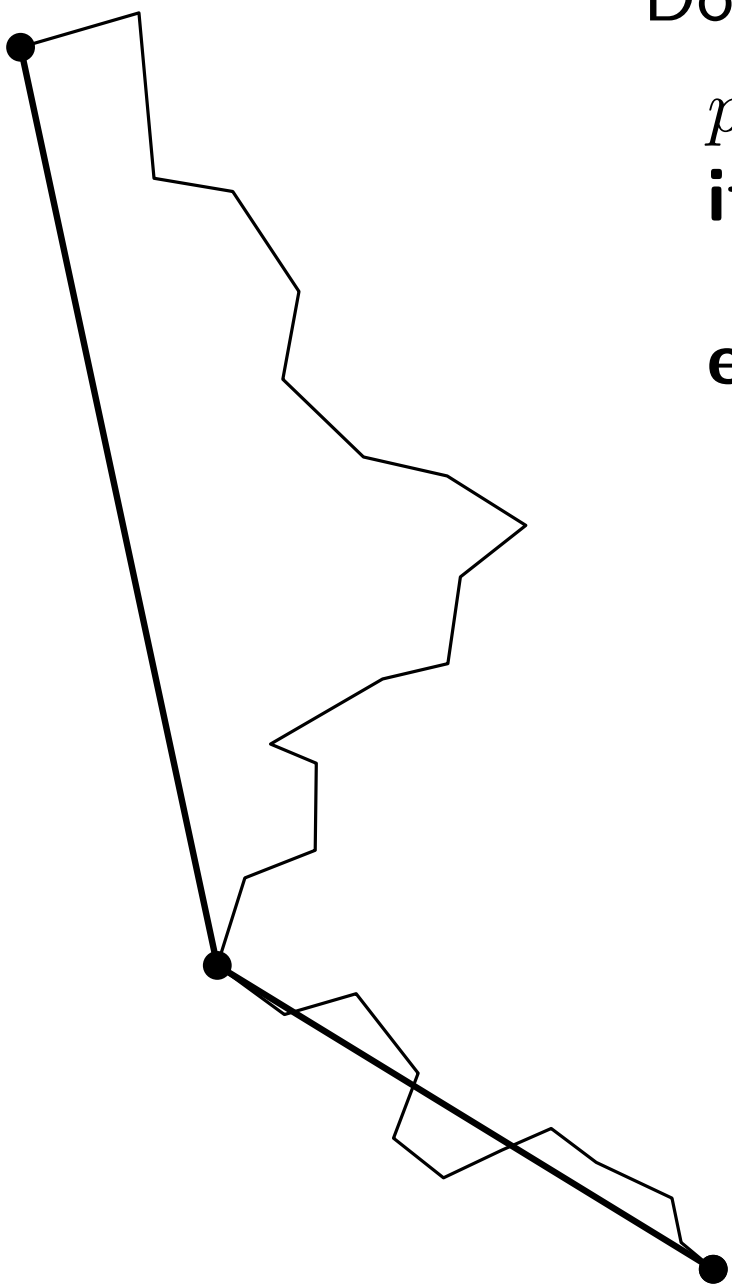
Douglas-Peucker Algorithmus [1973]



DouglasPeucker(P, i, j)

$p_f \leftarrow$ weitester Knoten von $\overline{p_i p_j}$
if dist$(\overline{p_i p_j}, p_f) < \varepsilon$ **then**
  return $\overline{p_i p_j}$
else
  DouglasPeucker(P, i, f)
  DouglasPeucker(P, f, j)

Douglas-Peucker Algorithmus [1973]



DouglasPeucker(P, i, j)

$p_f \leftarrow$ weitester Knoten von $\overline{p_i p_j}$

if dist$(\overline{p_i p_j}, p_f) < \varepsilon$ **then**

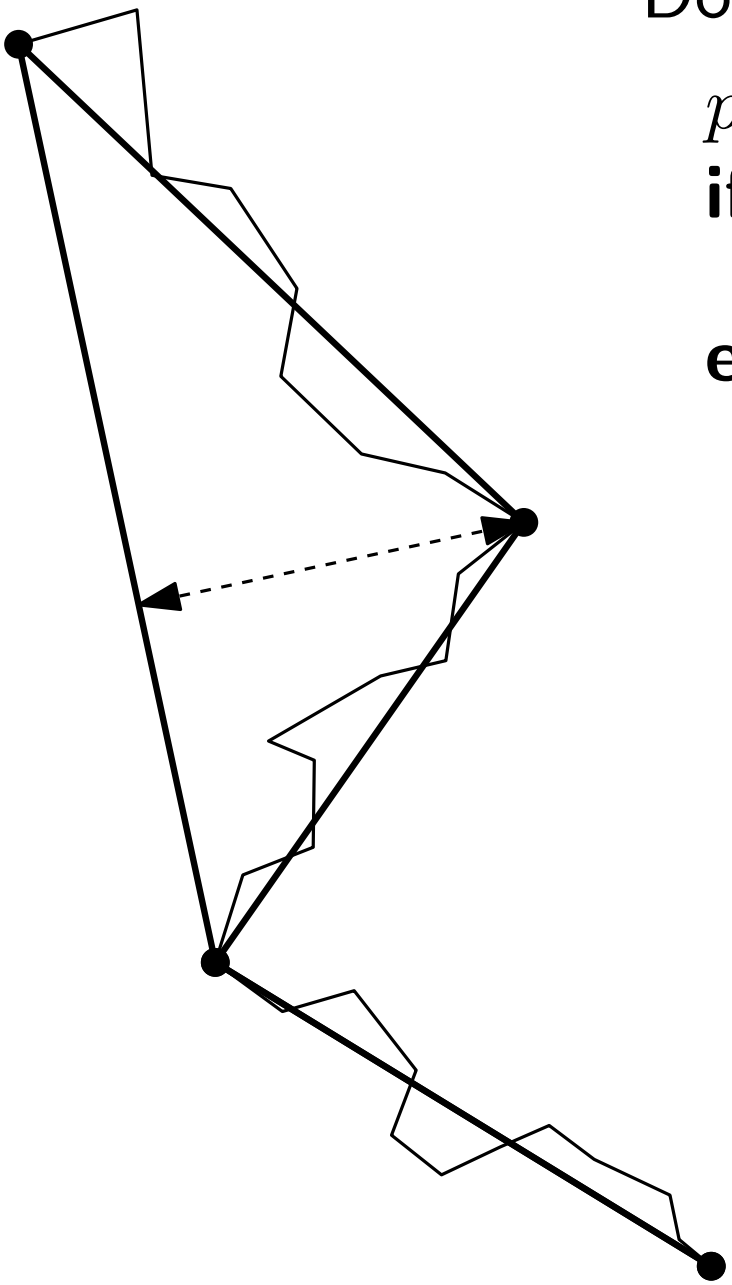
  return $\overline{p_i p_j}$

else

  DouglasPeucker(P, i, f)

  DouglasPeucker(P, f, j)

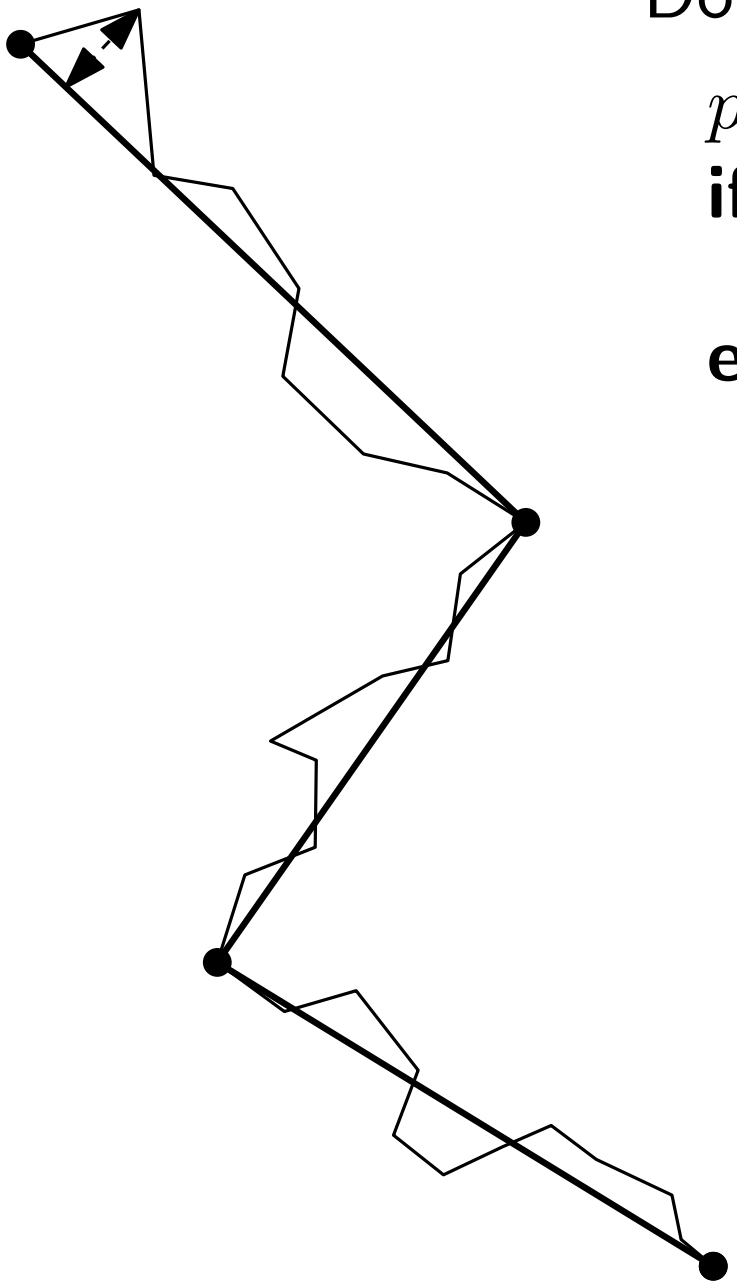
Douglas-Peucker Algorithmus [1973]



DouglasPeucker(P, i, j)

$p_f \leftarrow$ weitester Knoten von $\overline{p_i p_j}$
if dist$(\overline{p_i p_j}, p_f) < \varepsilon$ **then**
  return $\overline{p_i p_j}$
else
  DouglasPeucker(P, i, f)
  DouglasPeucker(P, f, j)

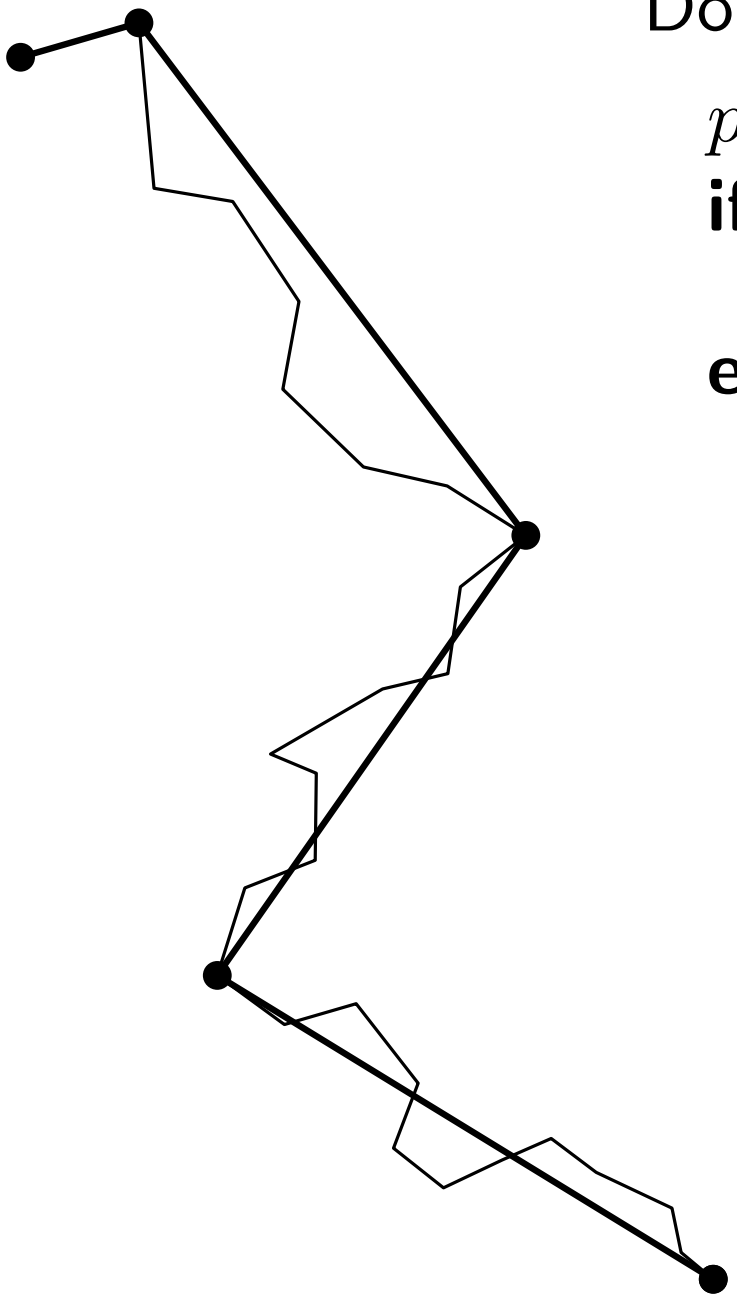
Douglas-Peucker Algorithmus [1973]



DouglasPeucker(P, i, j)

$p_f \leftarrow$ weitester Knoten von $\overline{p_i p_j}$
if dist$(\overline{p_i p_j}, p_f) < \varepsilon$ **then**
  return $\overline{p_i p_j}$
else
  DouglasPeucker(P, i, f)
  DouglasPeucker(P, f, j)

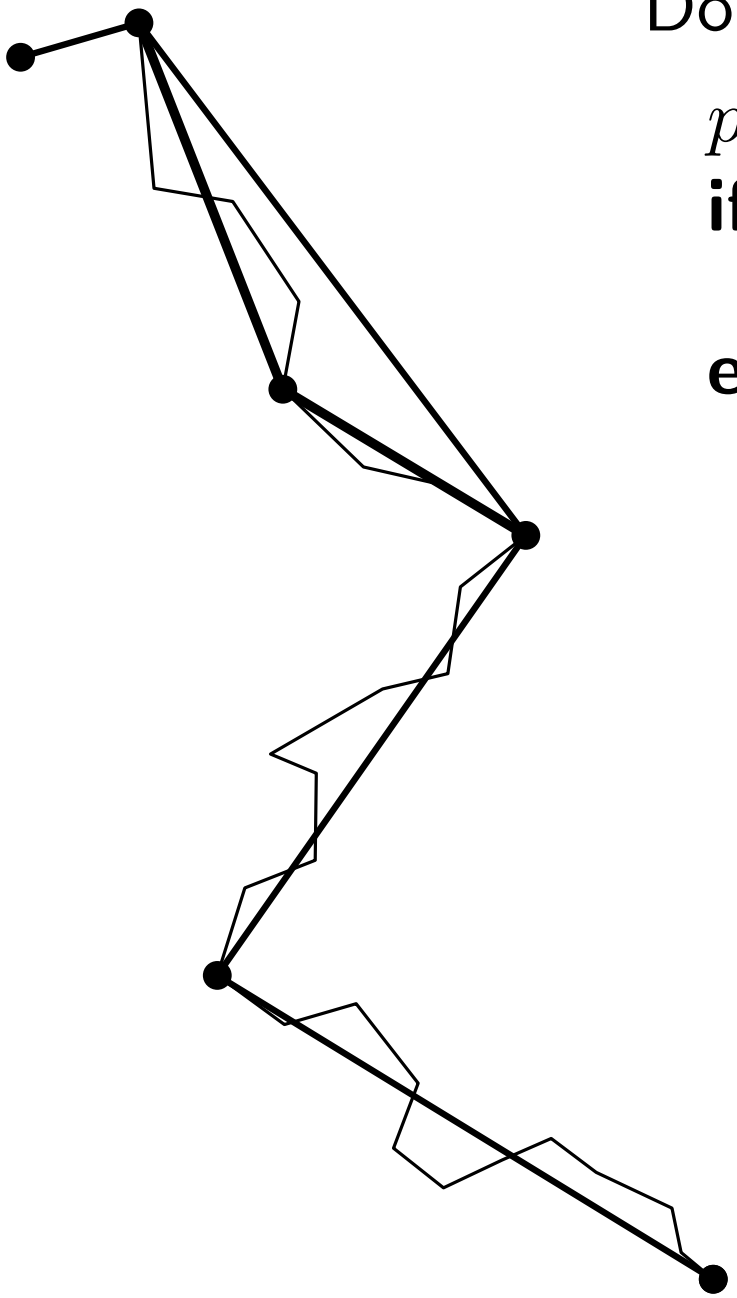
Douglas-Peucker Algorithmus [1973]



DouglasPeucker(P, i, j)

$p_f \leftarrow$ weitester Knoten von $\overline{p_i p_j}$
if dist$(\overline{p_i p_j}, p_f) < \varepsilon$ **then**
  return $\overline{p_i p_j}$
else
  DouglasPeucker(P, i, f)
  DouglasPeucker(P, f, j)

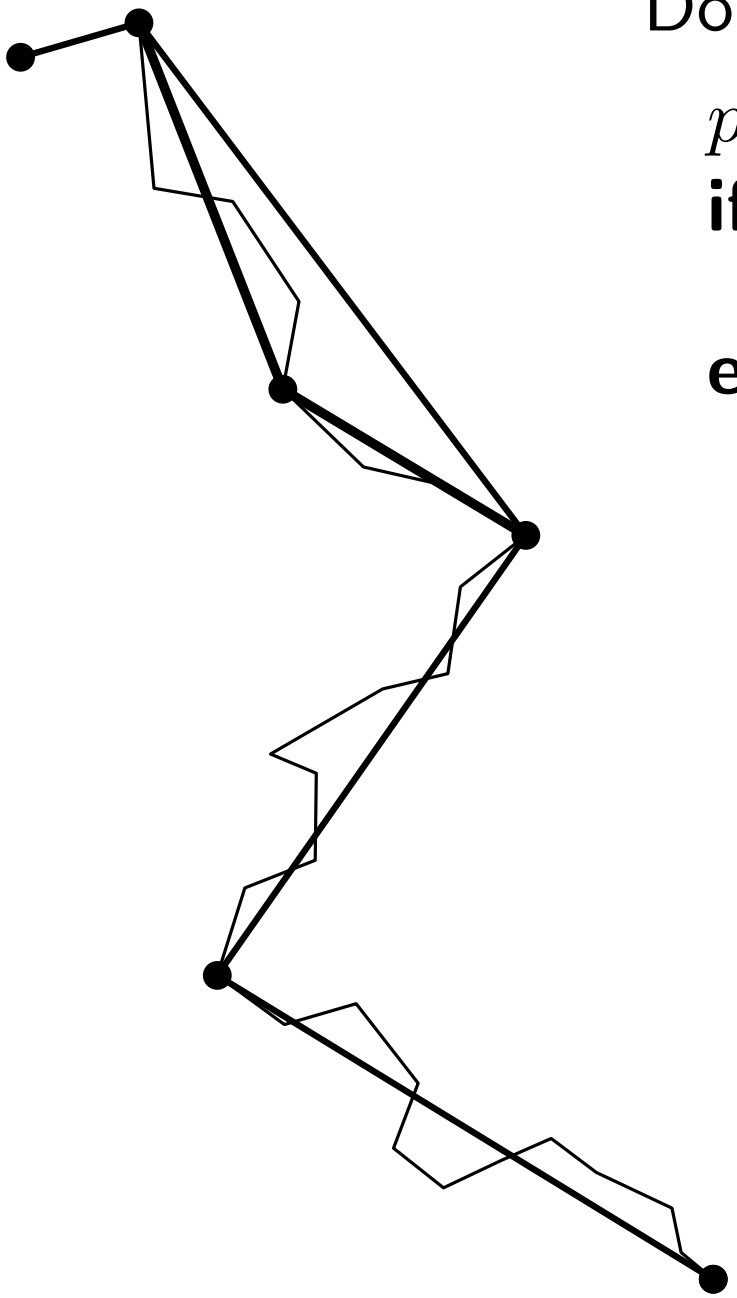
Douglas-Peucker Algorithmus [1973]



DouglasPeucker(P, i, j)

$p_f \leftarrow$ weitester Knoten von $\overline{p_i p_j}$
if dist$(\overline{p_i p_j}, p_f) < \varepsilon$ **then**
  return $\overline{p_i p_j}$
else
  DouglasPeucker(P, i, f)
  DouglasPeucker(P, f, j)

Douglas-Peucker Algorithmus [1973]

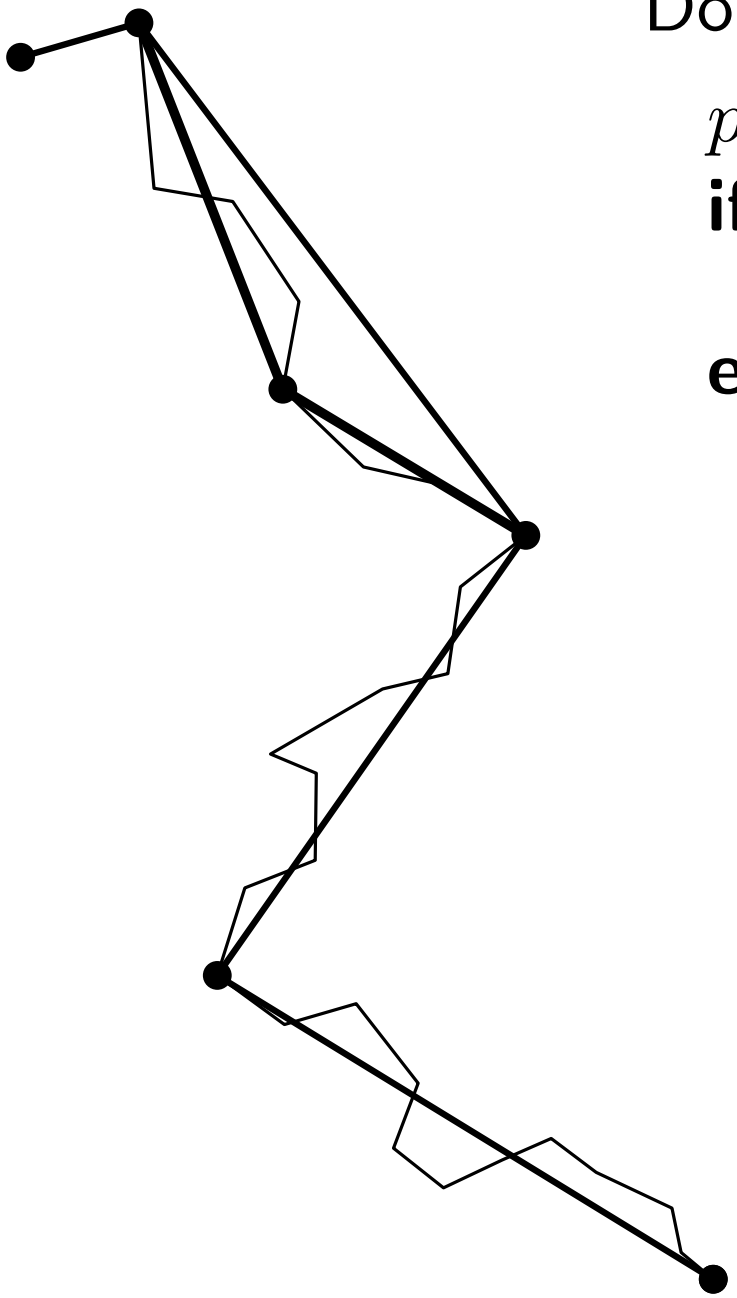


DouglasPeucker(P, i, j)

$p_f \leftarrow$ weitester Knoten von $\overline{p_i p_j}$
if dist$(\overline{p_i p_j}, p_f) < \varepsilon$ **then**
  return $\overline{p_i p_j}$
else
  DouglasPeucker(P, i, f)
  DouglasPeucker(P, f, j)

Laufzeit?

Douglas-Peucker Algorithmus [1973]



DouglasPeucker(P, i, j)

$p_f \leftarrow$ weitester Knoten von $\overline{p_i p_j}$
if dist$(\overline{p_i p_j}, p_f) < \varepsilon$ **then**
  return $\overline{p_i p_j}$
else
  DouglasPeucker(P, i, f)
  DouglasPeucker(P, f, j)

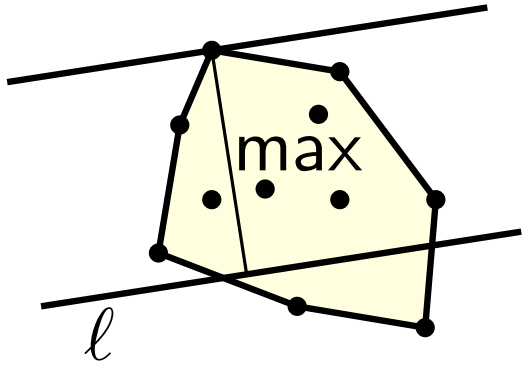
Laufzeit?

- naive Implementierung: $O(n^2)$
- balancierter Fall: $O(n \log n)$

Beschleunigung des DP-Algorithmus

[Hershberger, Snoeyink '92]

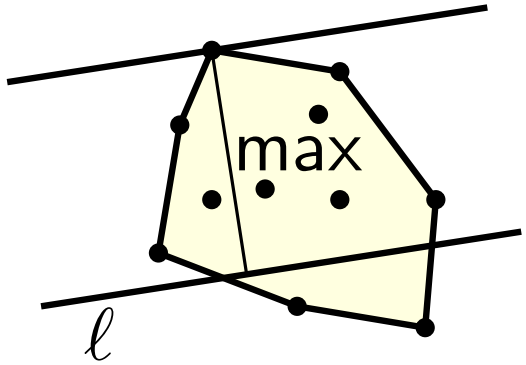
Beobachtung: für Gerade ℓ und Punktmenge P liegt der von ℓ weitest entfernte Punkt auf einer zu ℓ parallelen Tangente an die konvexe Hülle $CH(P)$



Beschleunigung des DP-Algorithmus

[Hershberger, Snoeyink '92]

Beobachtung: für Gerade ℓ und Punktmenge P liegt der von ℓ weitest entfernte Punkt auf einer zu ℓ parallelen Tangente an die konvexe Hülle $CH(P)$

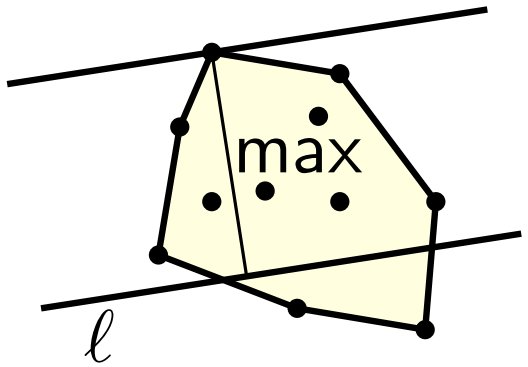


$\rightarrow$ müssen nur solche Punkte als Splitknoten betrachten

Beschleunigung des DP-Algorithmus

[Hershberger, Snoeyink '92]

Beobachtung: für Gerade ℓ und Punktmenge P liegt der von ℓ weitest entfernte Punkt auf einer zu ℓ parallelen Tangente an die konvexe Hülle $CH(P)$



$\rightarrow$ müssen nur solche Punkte als Splitknoten betrachten

Def: Pfadhülle PH für $(p_i, \ldots, p_j)$ besteht aus einem Zwischenknoten p_m und den konvexen Hüllen $CH(p_i, \ldots, p_m)$ und $CH(p_m, \ldots, p_j)$

Algorithmus von Hershberger und Snoeyink

drei Operationen:

- build(i, j, PH): erstelle Pfadhülle von $(p_i, \ldots, p_j)$ mit Zwischenknoten $p_m = p_{\lfloor (i+j)/2 \rfloor}$
- split(PH, k): trenne Pfad $(p_i, \ldots, p_j)$ an Stelle k und gib Pfadhülle für Teilpfad, der p_m enthält zurück
- farthest(PH, ℓ): finde weitesten von ℓ entfernten Knoten in PH

Algorithmus von Hershberger und Snoeyink

drei Operationen:

- build(i, j, PH): erstelle Pfadhülle von $(p_i, \ldots, p_j)$ mit Zwischenknoten $p_m = p_{\lfloor (i+j)/2 \rfloor}$
- split(PH, k): trenne Pfad $(p_i, \ldots, p_j)$ an Stelle k und gib Pfadhülle für Teilpfad, der p_m enthält zurück
- farthest(PH, ℓ): finde weitesten von ℓ entfernten Knoten in PH

DPHull(i, j, PH)

$p_f \leftarrow$ farthest$(PH, \overline{p_i p_j})$
if dist$(\overline{p_i p_j}, p_f) \leq \varepsilon$ **then**
  return $\overline{p_i p_j}$
else
  if $f < m$ **then**
    split(PH, f)
    DPHull(f, j, PH)
    build(i, f, PH)
    DPHull(i, f, PH)
  else
    split(PH, f)
    DPHull(i, f, PH)
    build(f, j, PH)
    DPHull(f, j, PH)

Algorithmus von Hershberger und Snoeyink

drei Operationen:

- build(i, j, PH): erstelle Pfadhülle von $(p_i, \ldots, p_j)$ mit Zwischenknoten $p_m = p_{\lfloor (i+j)/2 \rfloor}$
- split(PH, k): trenne Pfad $(p_i, \ldots, p_j)$ an Stelle k und gib Pfadhülle für Teilpfad, der p_m enthält zurück
- farthest(PH, ℓ): finde weitesten von ℓ entfernten Knoten in PH

Satz: DPHull$(1, n, PH)$ kann mit Laufzeit $O(n \log n)$ implementiert werden.
(Ann: $(p_1, \ldots p_n)$ schnittfrei)

DPHull(i, j, PH)

$p_f \leftarrow$ farthest$(PH, \overline{p_i p_j})$
if dist$(\overline{p_i p_j}, p_f) \leq \varepsilon$ **then**
  return $\overline{p_i p_j}$
else
  if $f < m$ **then**
    split(PH, f)
    DPHull(f, j, PH)
    build(i, f, PH)
    DPHull(i, f, PH)
  else
    split(PH, f)
    DPHull(i, f, PH)
    build(f, j, PH)
    DPHull(f, j, PH)

Alternative Punktauswahl

Entferntesten Punkt wie bei DP zu behalten kann Ausreißer betonen.

Betrachte stattdessen **effektive Fläche** jedes Knotens und entferne iterativ Knoten mit kleinster Fläche.

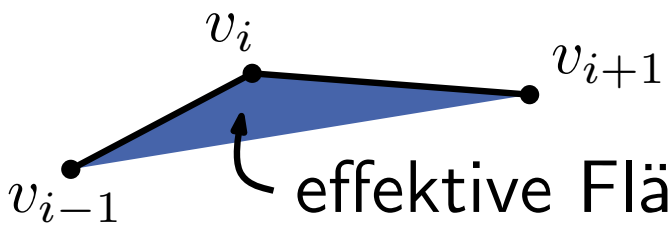
[Visvalingam, Whyatt '93]

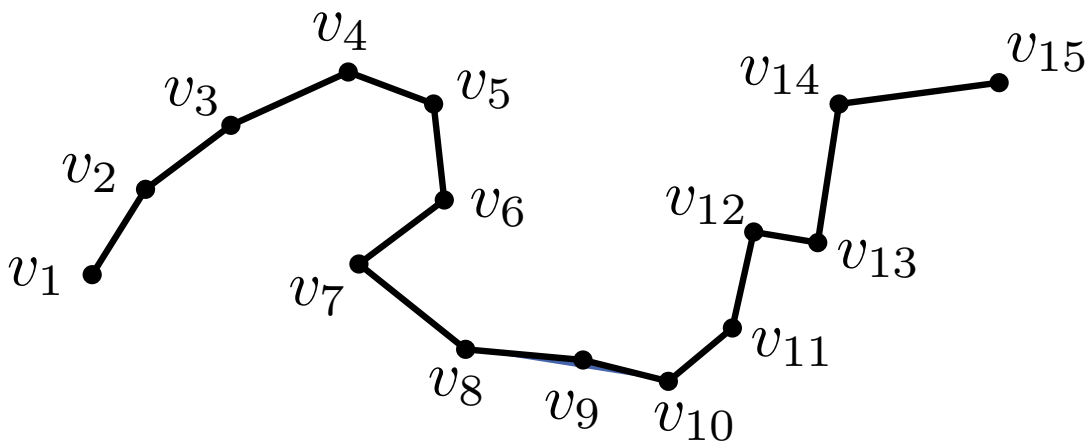
Alternative Punktauswahl

Entferntesten Punkt wie bei DP zu behalten kann Ausreißer betonen.

Betrachte stattdessen **effektive Fläche** jedes Knotens und entferne iterativ Knoten mit kleinster Fläche.

[Visvalingam, Whyatt '93]



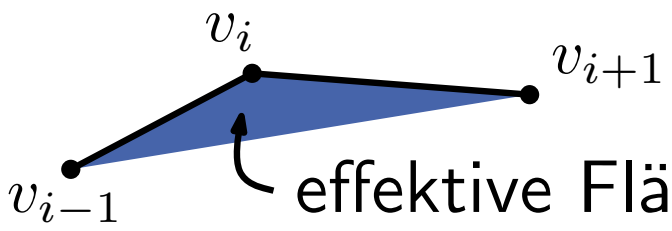


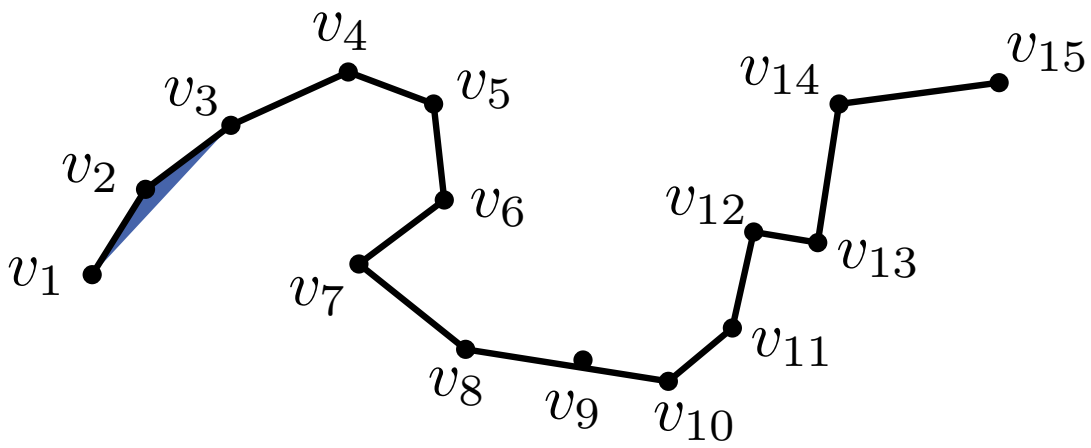
Alternative Punktauswahl

Entferntesten Punkt wie bei DP zu behalten kann Ausreißer betonen.

Betrachte stattdessen **effektive Fläche** jedes Knotens und entferne iterativ Knoten mit kleinster Fläche.

[Visvalingam, Whyatt '93]



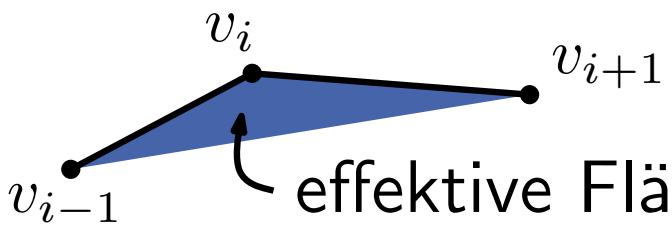


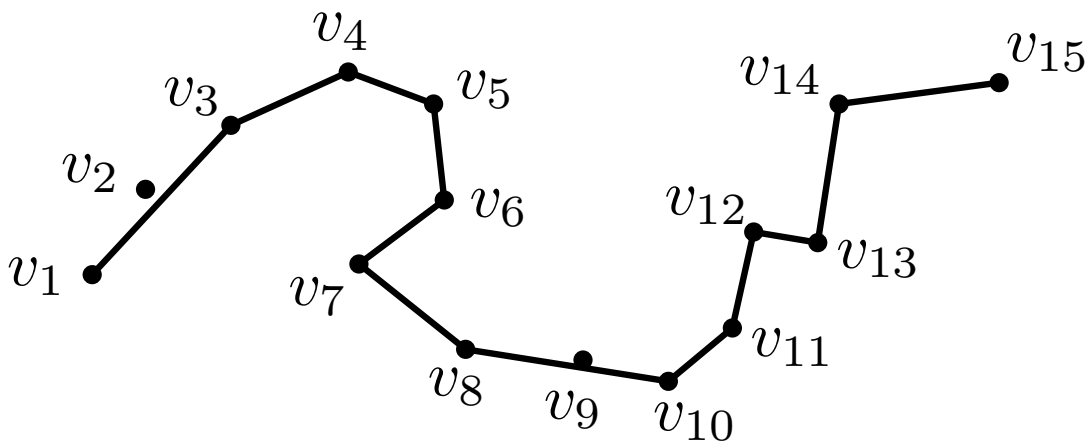
Alternative Punktauswahl

Entferntesten Punkt wie bei DP zu behalten kann Ausreißer betonen.

Betrachte stattdessen **effektive Fläche** jedes Knotens und entferne iterativ Knoten mit kleinster Fläche.

[Visvalingam, Whyatt '93]



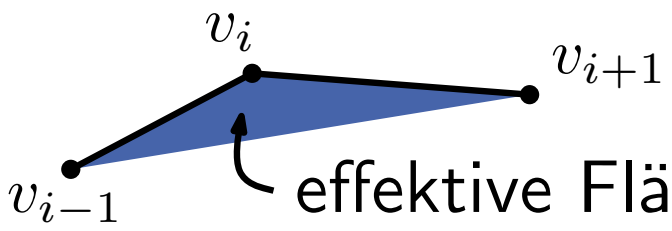


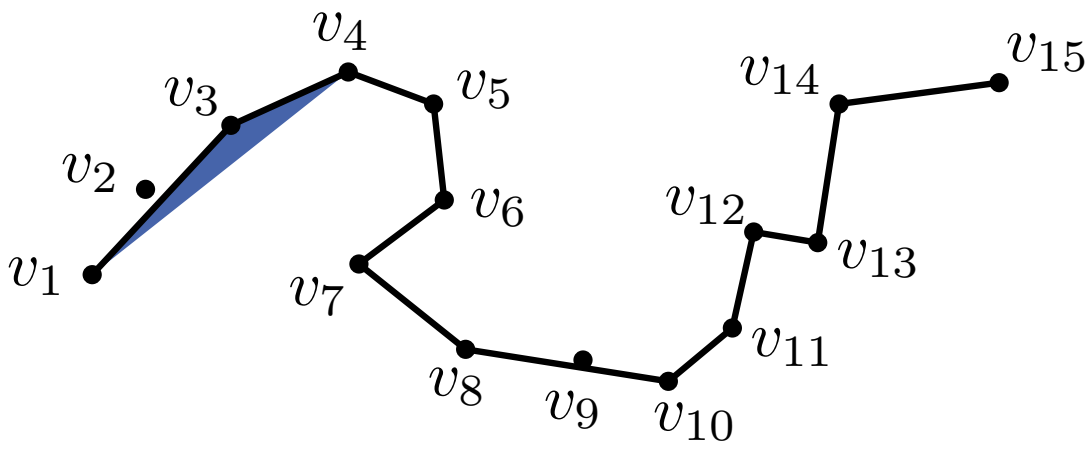
Alternative Punktauswahl

Entferntesten Punkt wie bei DP zu behalten kann Ausreißer betonen.

Betrachte stattdessen **effektive Fläche** jedes Knotens und entferne iterativ Knoten mit kleinster Fläche.

[Visvalingam, Whyatt '93]



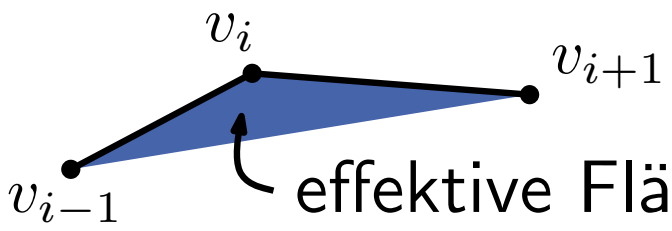


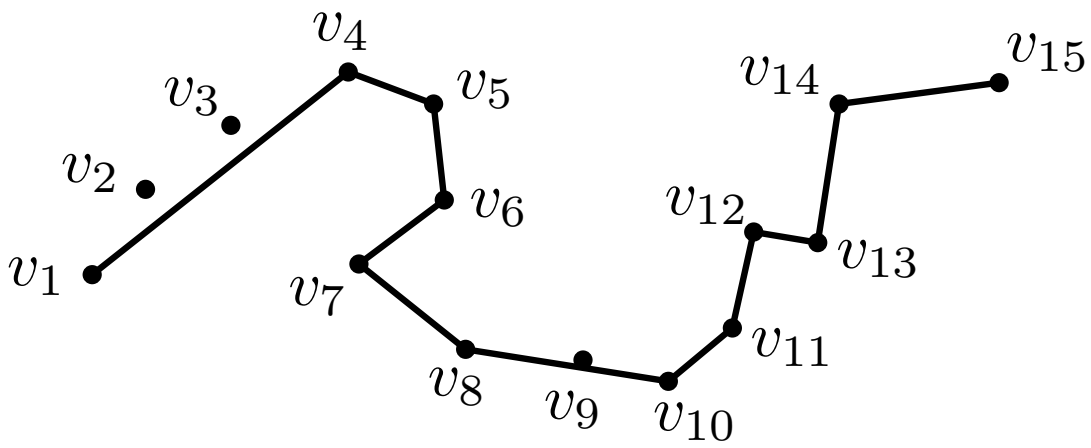
Alternative Punktauswahl

Entferntesten Punkt wie bei DP zu behalten kann Ausreißer betonen.

Betrachte stattdessen **effektive Fläche** jedes Knotens und entferne iterativ Knoten mit kleinster Fläche.

[Visvalingam, Whyatt '93]



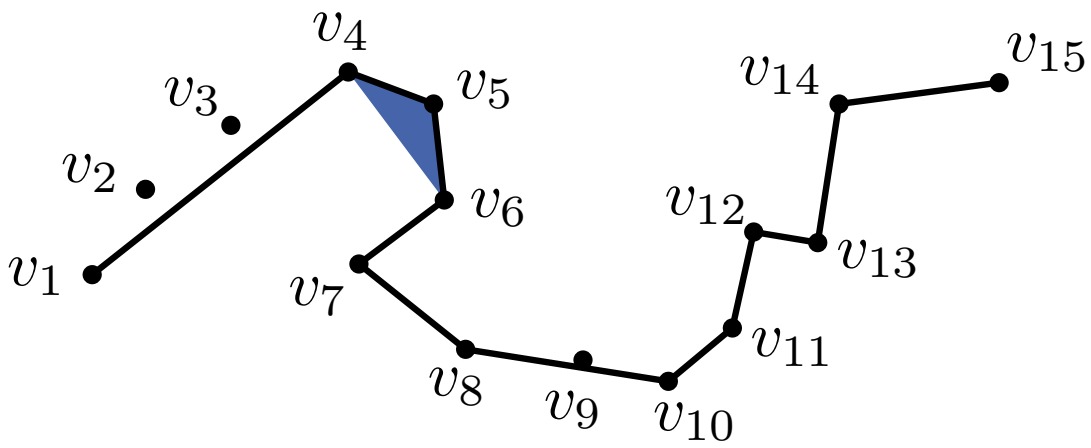


Alternative Punktauswahl

Entferntesten Punkt wie bei DP zu behalten kann Ausreißer betonen.

Betrachte stattdessen **effektive Fläche** jedes Knotens und entferne iterativ Knoten mit kleinster Fläche.

[Visvalingam, Whyatt '93]

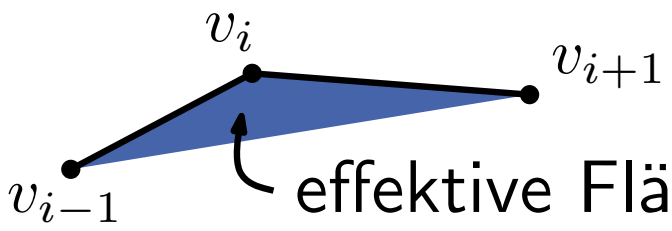


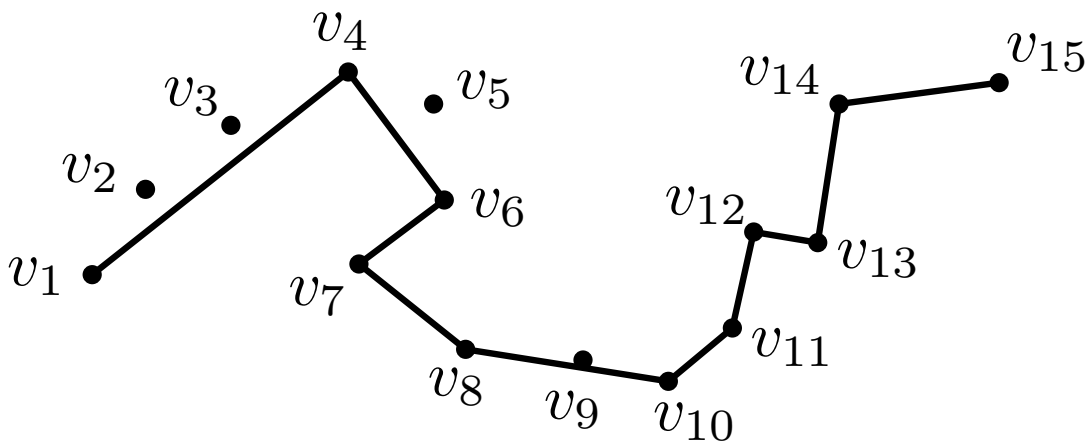
Alternative Punktauswahl

Entferntesten Punkt wie bei DP zu behalten kann Ausreißer betonen.

Betrachte stattdessen **effektive Fläche** jedes Knotens und entferne iterativ Knoten mit kleinster Fläche.

[Visvalingam, Whyatt '93]



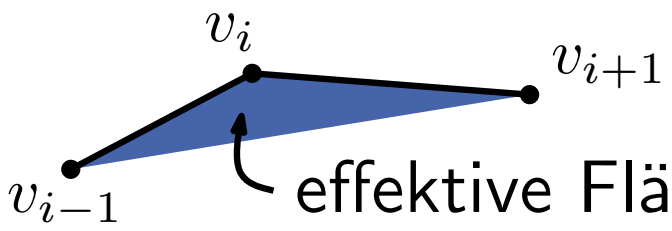


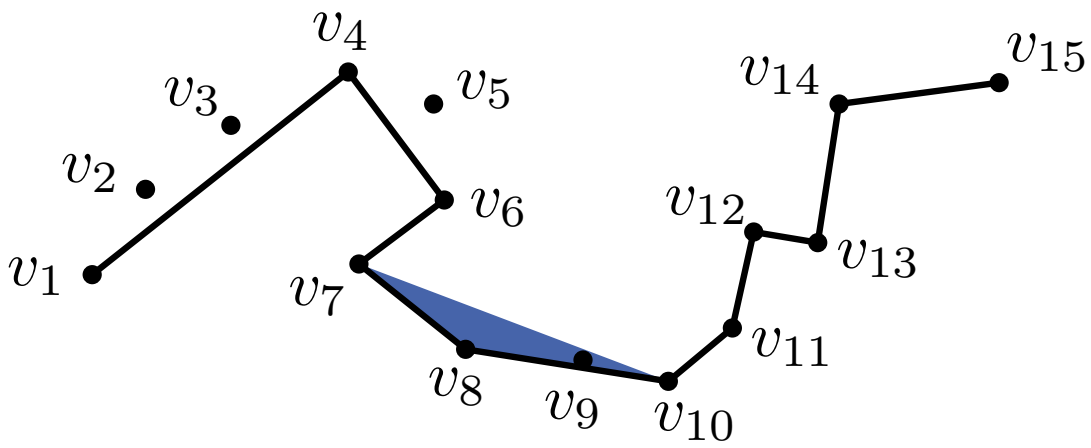
Alternative Punktauswahl

Entferntesten Punkt wie bei DP zu behalten kann Ausreißer betonen.

Betrachte stattdessen **effektive Fläche** jedes Knotens und entferne iterativ Knoten mit kleinster Fläche.

[Visvalingam, Whyatt '93]



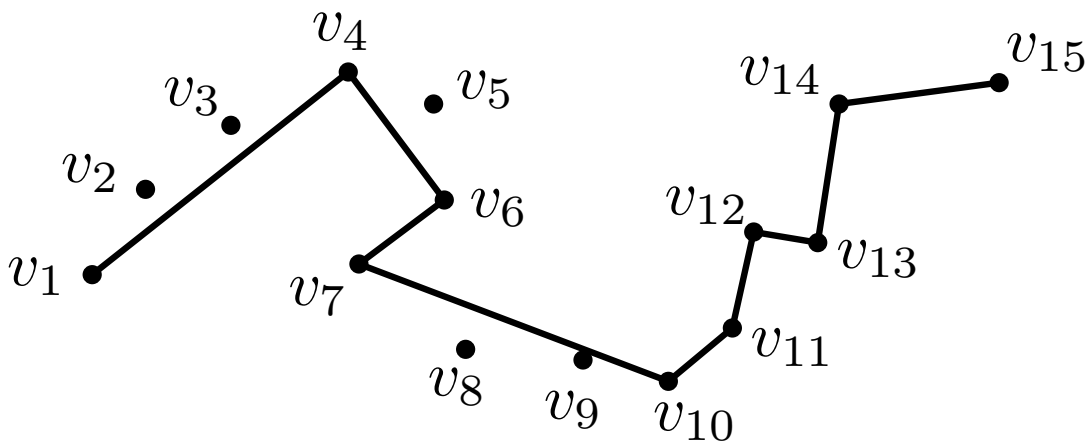


Alternative Punktauswahl

Entferntesten Punkt wie bei DP zu behalten kann Ausreißer betonen.

Betrachte stattdessen **effektive Fläche** jedes Knotens und entferne iterativ Knoten mit kleinster Fläche.

[Visvalingam, Whyatt '93]

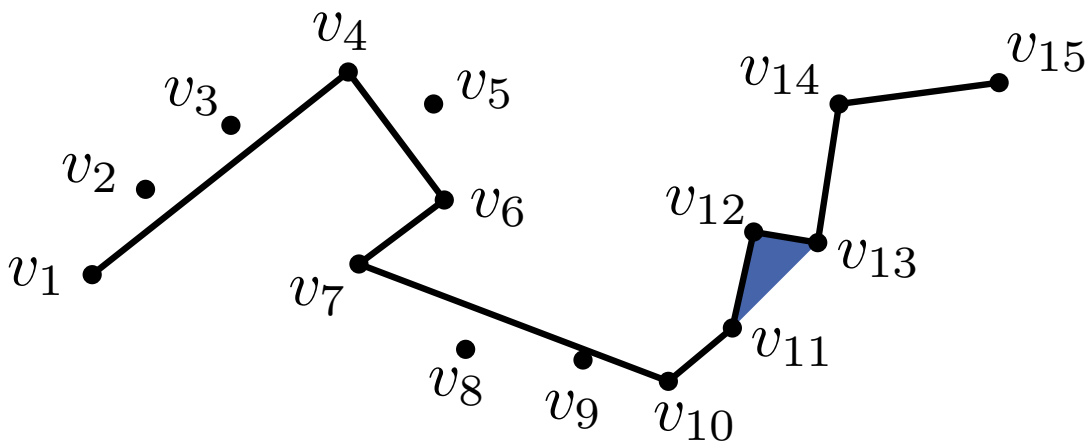


Alternative Punktauswahl

Entferntesten Punkt wie bei DP zu behalten kann Ausreißer betonen.

Betrachte stattdessen **effektive Fläche** jedes Knotens und entferne iterativ Knoten mit kleinster Fläche.

[Visvalingam, Whyatt '93]

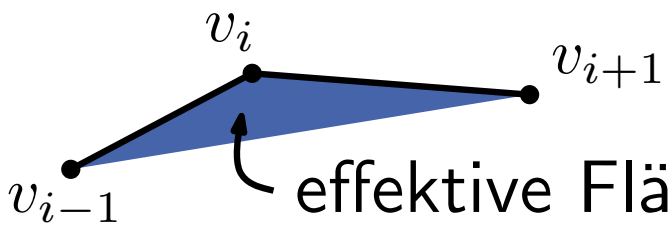


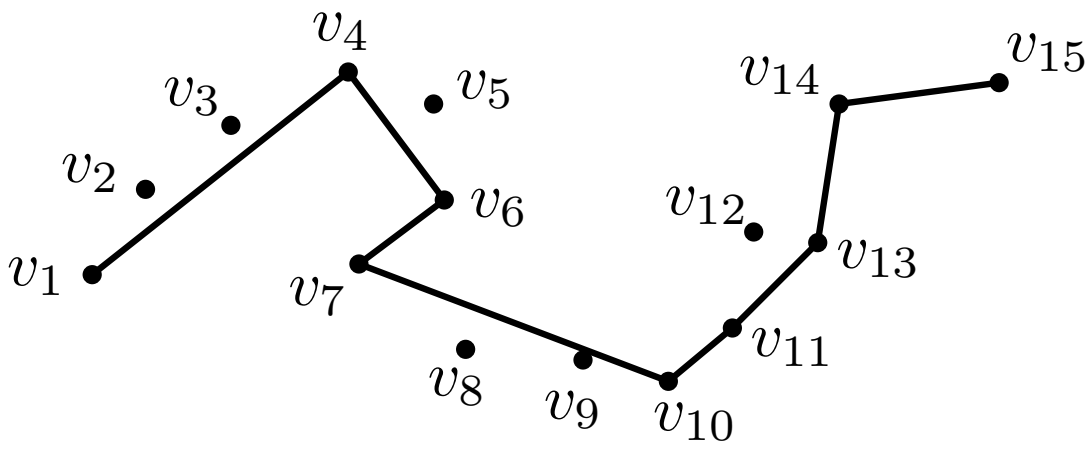
Alternative Punktauswahl

Entferntesten Punkt wie bei DP zu behalten kann Ausreißer betonen.

Betrachte stattdessen **effektive Fläche** jedes Knotens und entferne iterativ Knoten mit kleinster Fläche.

[Visvalingam, Whyatt '93]



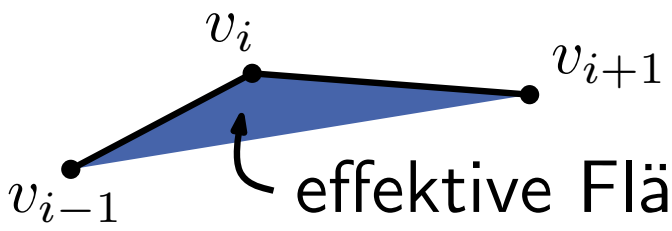


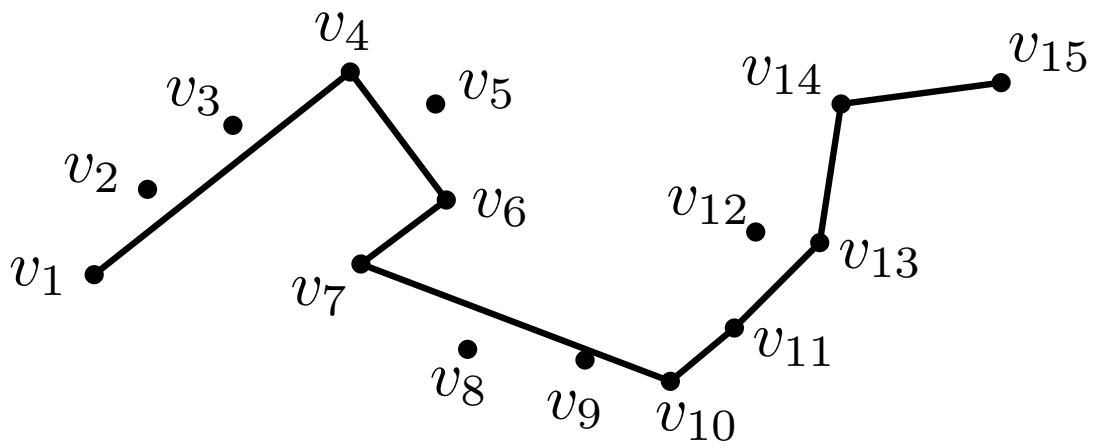
Alternative Punktauswahl

Entferntesten Punkt wie bei DP zu behalten kann Ausreißer betonen.

Betrachte stattdessen **effektive Fläche** jedes Knotens und entferne iterativ Knoten mit kleinster Fläche.

[Visvalingam, Whyatt '93]





Laufzeit?

Vereinfachung als Optimierungsproblem

Karlsruhe Institute of Technology

Bisherige Algorithmen liefern heuristisch gute Approximationen an die Eingabe, jedoch nicht notwenig eine optimale Lösung.

Formulierung als Graphenproblem:

- vollständiger DAG G auf $\{v_1, \ldots, v_n\}$ mit Kante (v_i, v_j) für alle $i < j$
- Kosten c_{ij} für das Ersetzen des Teilpfads $(v_i, v_{i+1}, \ldots, v_j)$ durch Kante (v_i, v_j)
- finde kostenminimalen (kürzesten) Weg von v_1 nach v_n in G mit maximal k Kanten

Vereinfachung als Optimierungsproblem

Bisherige Algorithmen liefern heuristisch gute Approximationen an die Eingabe, jedoch nicht notwenig eine optimale Lösung.

Formulierung als Graphenproblem:

- vollständiger DAG G auf $\{v_1, \ldots, v_n\}$ mit Kante (v_i, v_j) für alle $i < j$
- Kosten c_{ij} für das Ersetzen des Teilpfads $(v_i, v_{i+1}, \ldots, v_j)$ durch Kante (v_i, v_j)
- finde kostenminimalen (kürzesten) Weg von v_1 nach v_n in G mit maximal k Kanten

Solche „constrained shortest path“ Probleme sind i. Allg. NP-schwer, hier jedoch nicht.

$\rightarrow$ mehr dazu in der Übung!