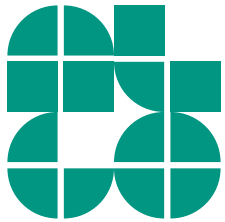
Vorlesung Algorithmische Kartografie

Linienvereinfachung III

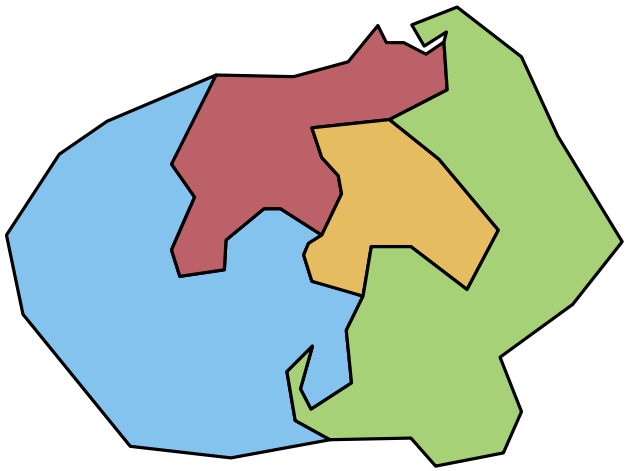
INSTITUT FÜR THEORETISCHE INFORMATIK · FAKULTÄT FÜR INFORMATIK

Benjamin Niedermann · **Martin Nöllenburg**

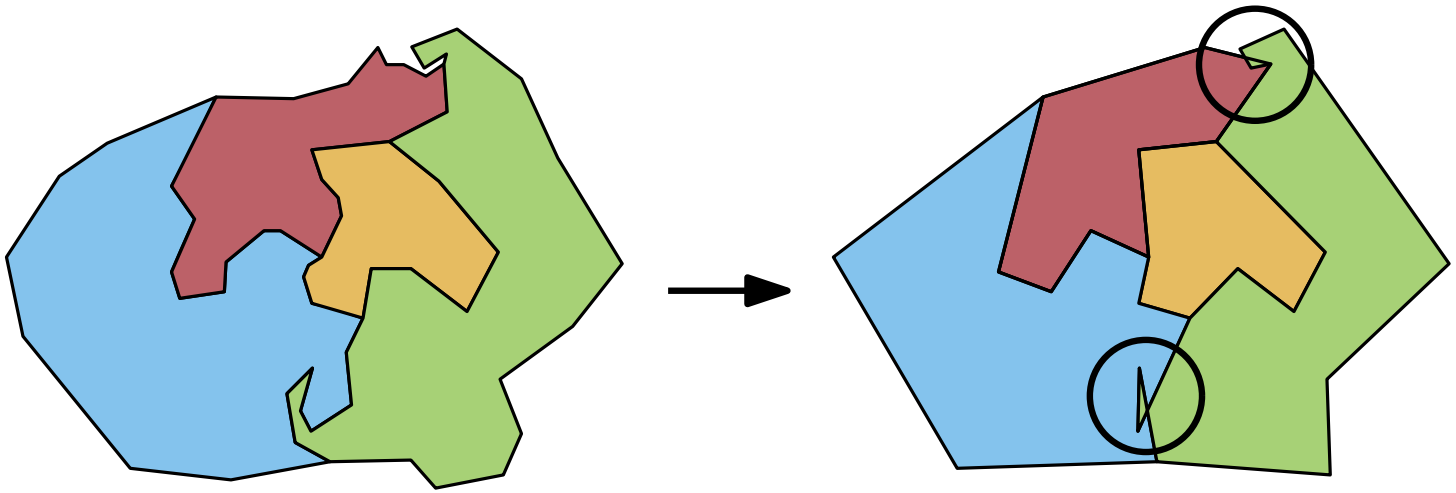
23.04.2015



Pfadvereinfachung für Landkarten

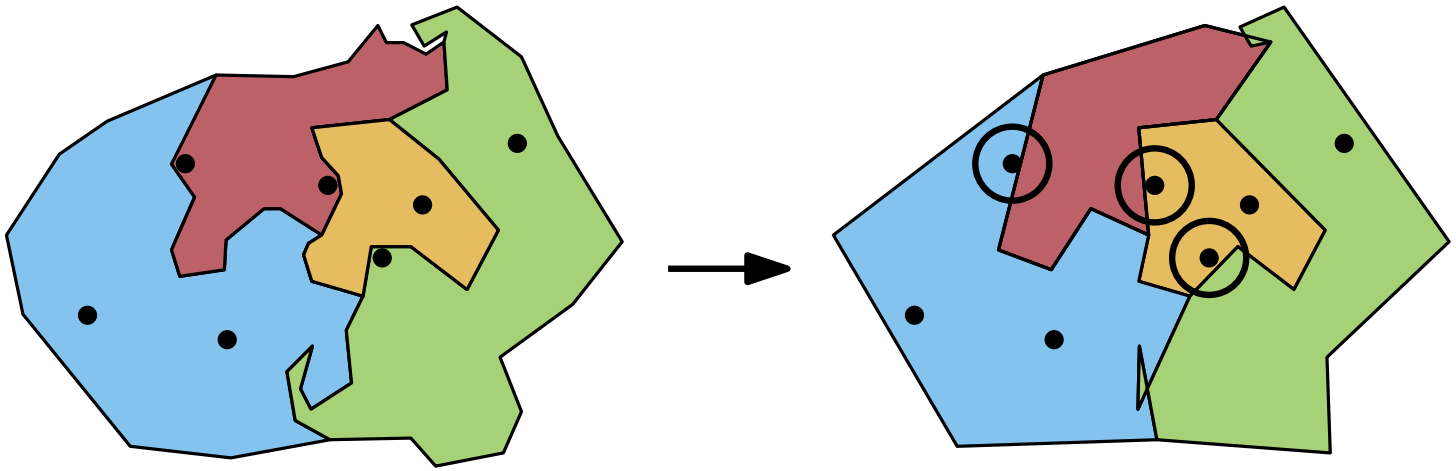


Pfadvereinfachung für Landkarten



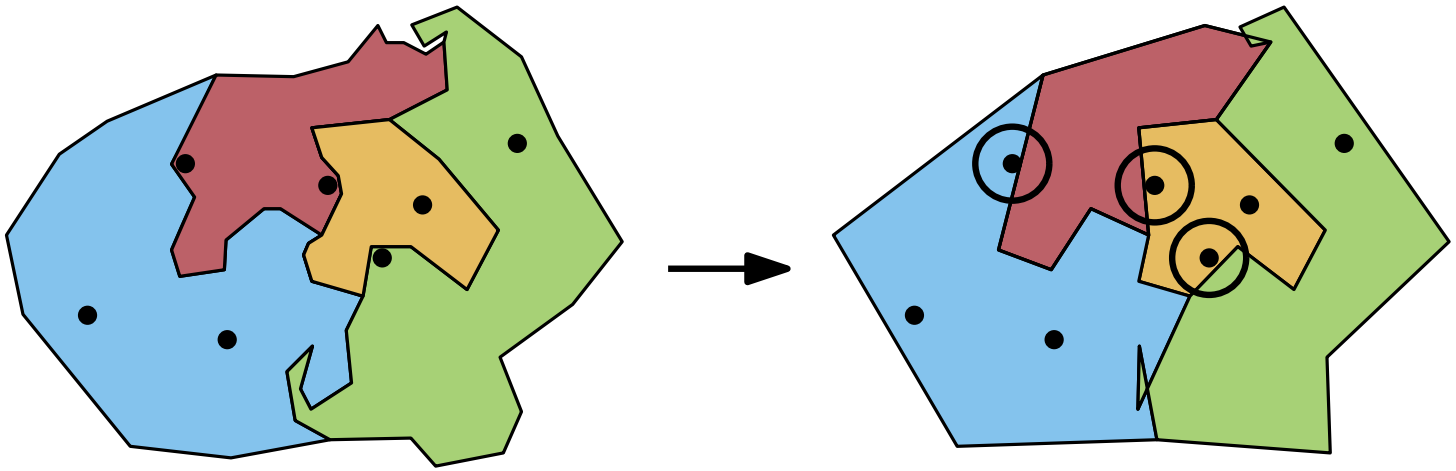
- Selbstschnitte und Schnitte mit anderen Pfaden möglich

Pfadvereinfachung für Landkarten



- Selbstschnitte und Schnitte mit anderen Pfaden möglich
- Punkte (z.B. Städte) können auf falsche Seite gelangen

Pfadvereinfachung für Landkarten



- Selbstschnitte und Schnitte mit anderen Pfaden möglich
- Punkte (z.B. Städte) können auf falsche Seite gelangen

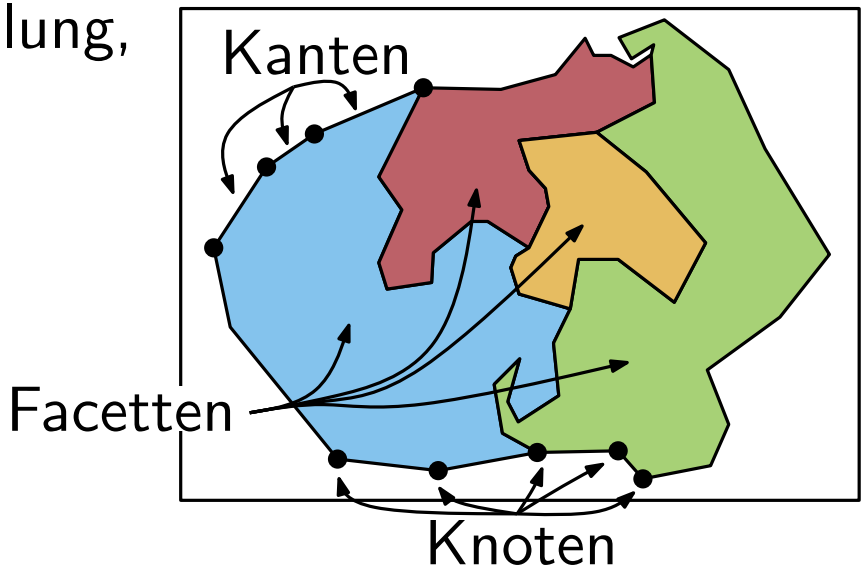
$\rightarrow$ benötigen topologisch korrekten Vereinfachungsalgorithmus

Ansatz 1: nachträgliches Korrigieren

Ansatz 2: Vermeiden durch „schlauere“ Algorithmen

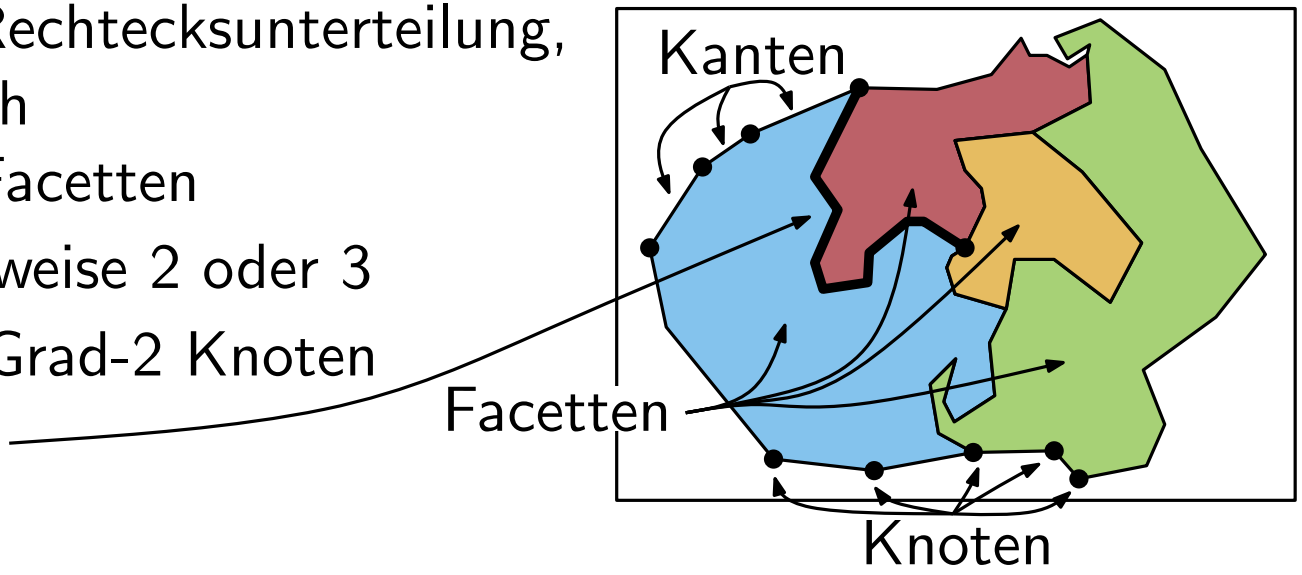
Ein paar Begriffe

- Karte modelliert als Rechtecksunterteilung, d.h. als planarer Graph
  $\rightarrow$ Knoten, Kanten, Facetten



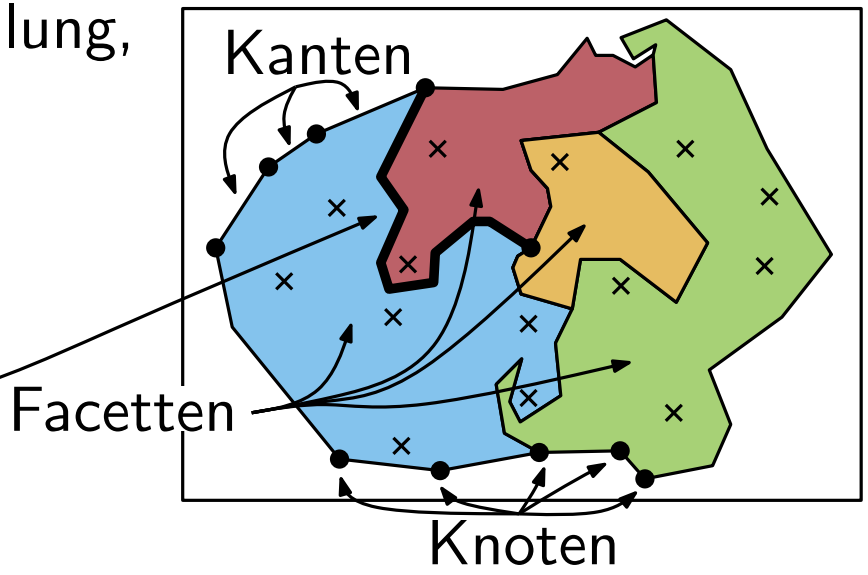
Ein paar Begriffe

- Karte modelliert als Rechtecksunterteilung, d.h. als planarer Graph
  $\rightarrow$ Knoten, Kanten, Facetten
- Knotengrade üblicherweise 2 oder 3
- maximale Pfade von Grad-2 Knoten bilden Kantenzüge



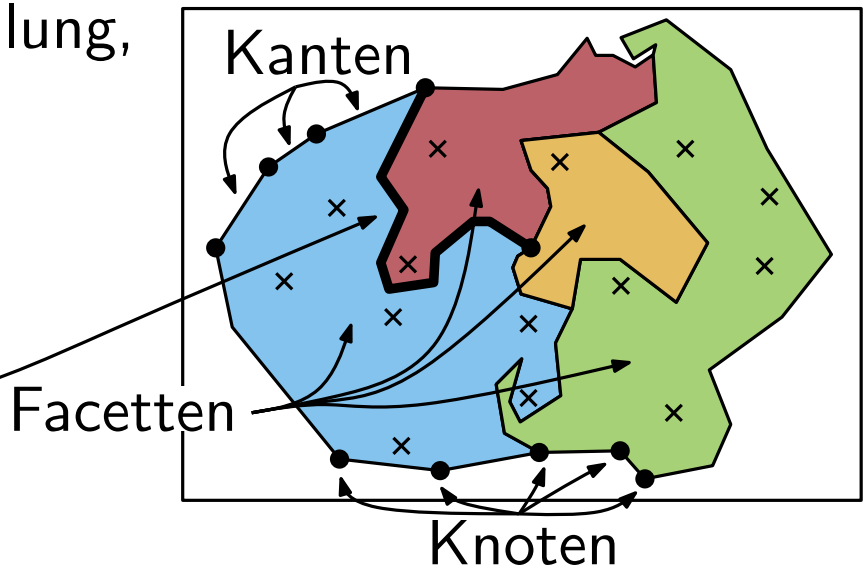
Ein paar Begriffe

- Karte modelliert als Rechtecksunterteilung, d.h. als planarer Graph
  $\rightarrow$ Knoten, Kanten, Facetten
- Knotengrade üblicherweise 2 oder 3
- maximale Pfade von Grad-2 Knoten bilden Kantenzüge
- zusätzlich Menge P von Punkten (x)



Ein paar Begriffe

- Karte modelliert als Rechtecksunterteilung, d.h. als planarer Graph
  $\rightarrow$ Knoten, Kanten, Facetten
- Knotengrade üblicherweise 2 oder 3
- maximale Pfade von Grad-2 Knoten bilden Kantenzüge
- zusätzlich Menge P von Punkten (x)



Ziel: ersetze alle Kantenzüge C durch vereinfachte Kantenzüge C' mit

- kein Punkt von C ist weiter als ε von C' entfernt
- C' ist einfach (d.h. schnittfrei)
- C' schneidet keinen weiteren Kantenzug
- alle Punkte in P liegen auf den gleichen Seiten bzgl. C und C', d.h. Punkte bleiben in den richtigen Facetten

$\rightarrow$ solch ein C' heißt **konsistente** Vereinfachung von C

Algorithmus von de Berg et al. (1998)

Idee:

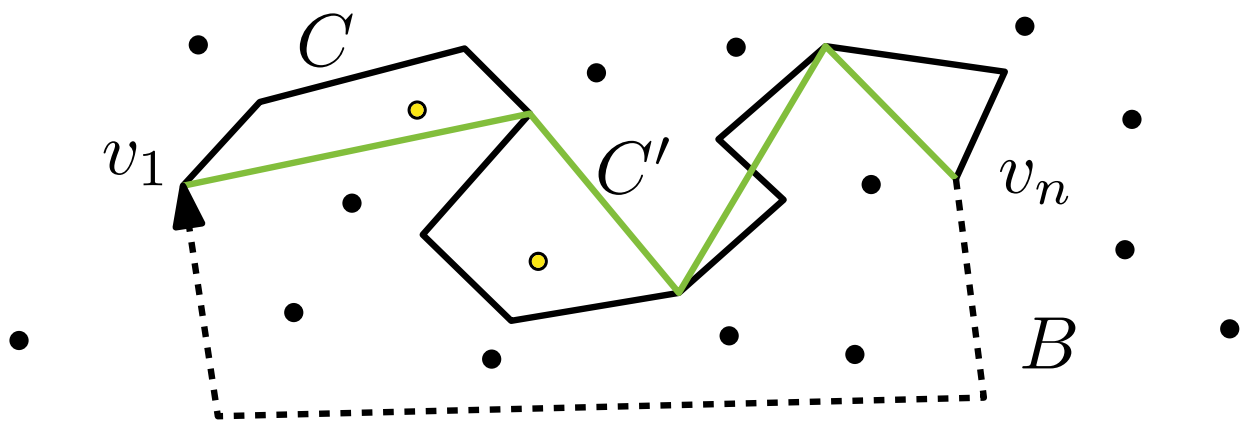
- graphbasierter Algorithmus (Imai & Iri) als Grundlage
- **zulässige** shortcuts bzgl. Fehler ε bilden $G_1 = (V, A_1)$
- **konsistente** shortcuts bzgl. P bilden $G_2 = (V, A_2)$
- kürzester Weg in Graph $G = (V, A_1 \cap A_2)$ liefert optimale zulässige und konsistente Vereinfachung

Algorithmus von de Berg et al. (1998)

Idee:

- graphbasierter Algorithmus (Imai & Iri) als Grundlage
- **zulässige** shortcuts bzgl. Fehler ε bilden $G_1 = (V, A_1)$
- **konsistente** shortcuts bzgl. P bilden $G_2 = (V, A_2)$
- kürzester Weg in Graph $G = (V, A_1 \cap A_2)$ liefert optimale zulässige und konsistente Vereinfachung

Def: Kantenzug $C = (v_1, \ldots, v_n)$ und Vereinfachung C' heißen **konsistent bzgl.** P, wenn es Kantenzug B von v_n nach v_1 gibt, der C und C' zu einfachen Polygonen abschließt, die die gleiche Teilmenge von P einschließen.



Algorithmus von de Berg et al. (1998)

Karlsruhe Institute of Technology

Idee:

- graphbasierter Algorithmus (Imai & Iri) als Grundlage
- **zulässige** shortcuts bzgl. Fehler ε bilden $G_1 = (V, A_1)$
- **konsistente** shortcuts bzgl. P bilden $G_2 = (V, A_2)$
- kürzester Weg in Graph $G = (V, A_1 \cap A_2)$ liefert optimale zulässige und konsistente Vereinfachung

Def: Kantenzug $C = (v_1, \ldots, v_n)$ und Vereinfachung C' heißen **konsistent bzgl.** P, wenn es Kantenzug B von v_n nach v_1 gibt, der C und C' zu einfachen Polygonen abschließt, die die gleiche Teilmenge von P einschließen.

Einschränkung: x-monotone Kantenzüge

x-monotone Kantenzüge

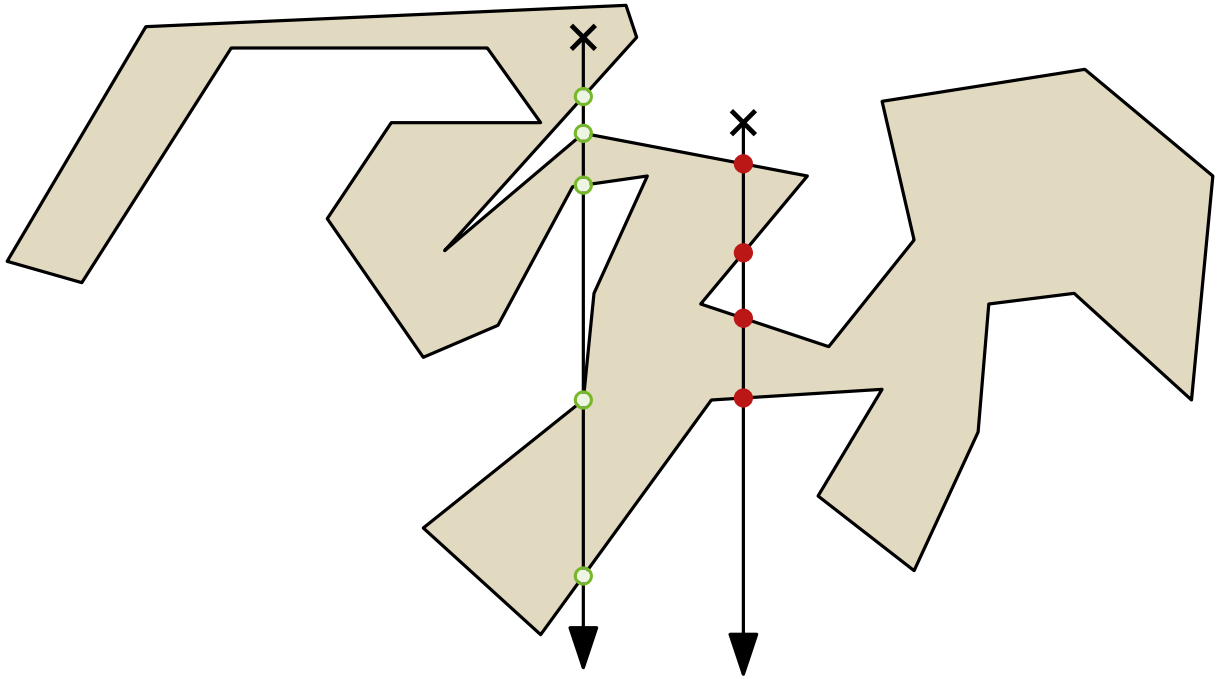
Lemma 1: C' konsistente Vereinfachung von C bzgl. $P \Leftrightarrow$ kein Punkt von P liegt in Facette von „$C \cup C'$“

x-monotone Kantenzüge

Lemma 1: C' konsistente Vereinfachung von C bzgl. $P \Leftrightarrow$ kein Punkt von P liegt in Facette von „$C \cup C'$“

Punkt-in-Polygon-Kriterium (PIP)

Punkt $p \in$ Polygon $Q \Leftrightarrow$ (negativer vertikaler) Strahl schneidet Q ungerade oft



x-monotone Kantenzüge

Lemma 1: C' konsistente Vereinfachung von C bzgl. $P \Leftrightarrow$ kein Punkt von P liegt in Facette von „$C \cup C'$“

Bestimmung aller konsistenten shortcuts von v_i

- C_{ij}: Kantenzug von v_i bis v_j
- Q_{ij}: Polygon $C_{ij} \cup \overline{v_i v_j}$
- $(v_i, v_j) \in A \Leftrightarrow Q_{ij}$ enthält keine Punkte aus P

x-monotone Kantenzüge

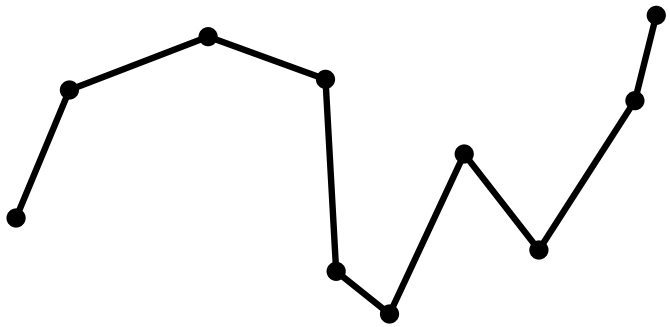
Lemma 1: C' konsistente Vereinfachung von C bzgl. $P \Leftrightarrow$ kein Punkt von P liegt in Facette von „$C \cup C'$“

Bestimmung aller konsistenten shortcuts von v_i

- C_{ij}: Kantenzug von v_i bis v_j
- Q_{ij}: Polygon $C_{ij} \cup \overline{v_i v_j}$
- $(v_i, v_j) \in A \Leftrightarrow Q_{ij}$ enthält keine Punkte aus P

Vorgehen für festes v_i in 3 Schritten

1. berechne Tangentensegmente und induzierte Unterteilung S_i
2. verteile P auf Regionen von S_i
3. berechne konsistente shortcuts



x-monotone Kantenzüge

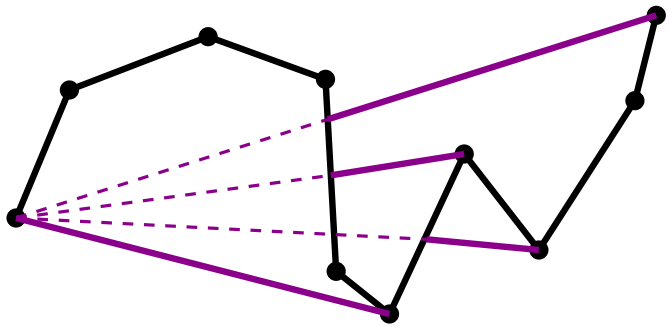
Lemma 1: C' konsistente Vereinfachung von C bzgl. $P \Leftrightarrow$ kein Punkt von P liegt in Facette von „$C \cup C'$“

Bestimmung aller konsistenten shortcuts von v_i

- C_{ij}: Kantenzug von v_i bis v_j
- Q_{ij}: Polygon $C_{ij} \cup \overline{v_i v_j}$
- $(v_i, v_j) \in A \Leftrightarrow Q_{ij}$ enthält keine Punkte aus P

Vorgehen für festes v_i in 3 Schritten

1. berechne Tangentensegmente und induzierte Unterteilung S_i
2. verteile P auf Regionen von S_i
3. berechne konsistente shortcuts



x-monotone Kantenzüge

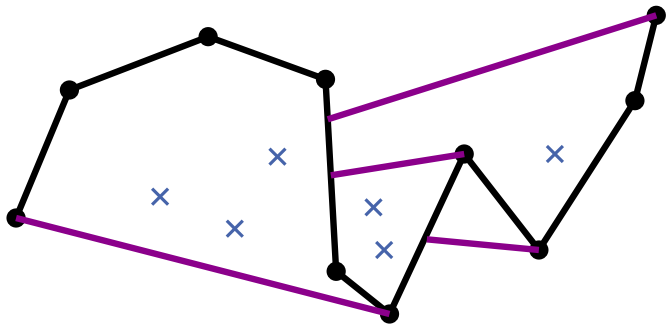
Lemma 1: C' konsistente Vereinfachung von C bzgl. $P \Leftrightarrow$ kein Punkt von P liegt in Facette von „$C \cup C'$“

Bestimmung aller konsistenten shortcuts von v_i

- C_{ij}: Kantenzug von v_i bis v_j
- Q_{ij}: Polygon $C_{ij} \cup \overline{v_i v_j}$
- $(v_i, v_j) \in A \Leftrightarrow Q_{ij}$ enthält keine Punkte aus P

Vorgehen für festes v_i in 3 Schritten

1. berechne Tangentensegmente und induzierte Unterteilung S_i
2. verteile P auf Regionen von S_i
3. berechne konsistente shortcuts



x-monotone Kantenzüge

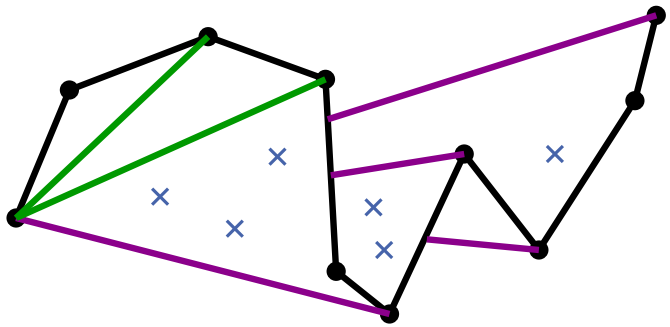
Lemma 1: C' konsistente Vereinfachung von C bzgl. $P \Leftrightarrow$ kein Punkt von P liegt in Facette von „$C \cup C'$“

Bestimmung aller konsistenten shortcuts von v_i

- C_{ij}: Kantenzug von v_i bis v_j
- Q_{ij}: Polygon $C_{ij} \cup \overline{v_i v_j}$
- $(v_i, v_j) \in A \Leftrightarrow Q_{ij}$ enthält keine Punkte aus P

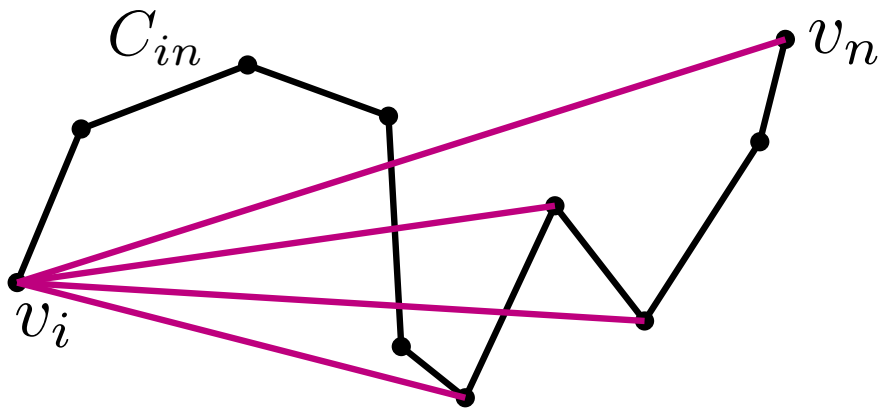
Vorgehen für festes v_i in 3 Schritten

1. berechne Tangentensegmente und induzierte Unterteilung S_i
2. verteile P auf Regionen von S_i
3. berechne konsistente shortcuts



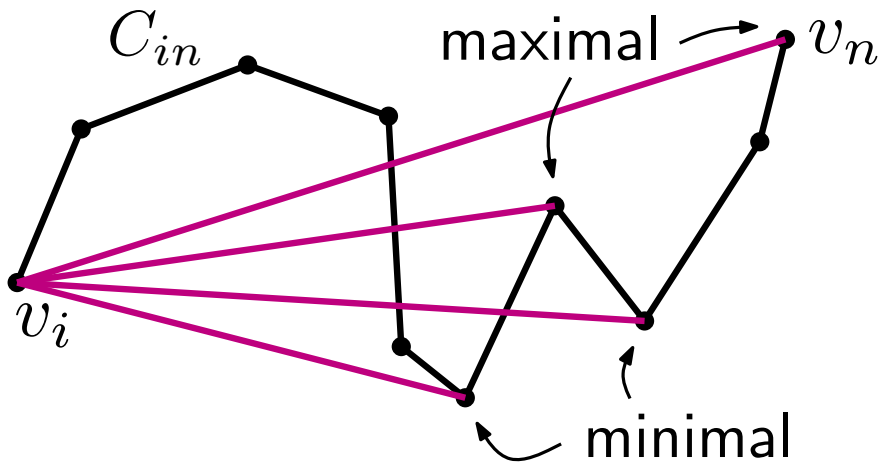
1) Tangentensegmente

- shortcut $v_i v_j$ ist **Tangente** an C_{in}, wenn v_{j-1} und v_{j+1} auf gleicher Seite von $v_i v_j$ liegen (oder $j = n$)



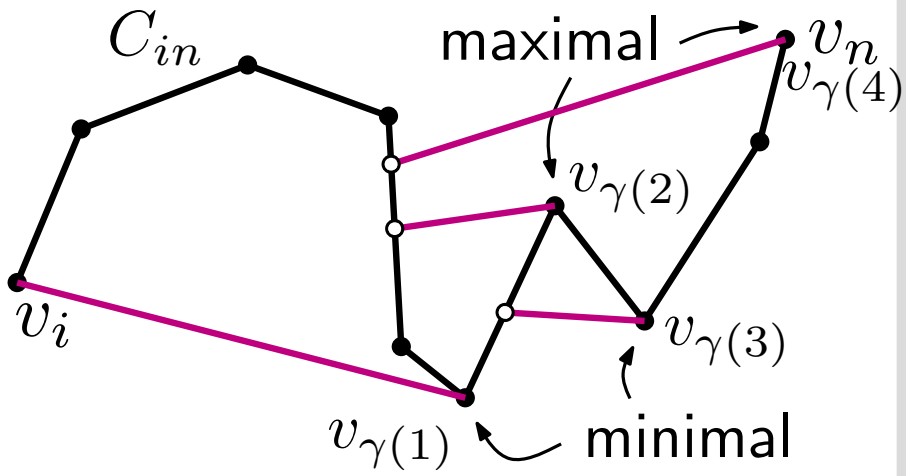
1) Tangentensegmente

- shortcut $v_i v_j$ ist **Tangente** an C_{in}, wenn v_{j-1} und v_{j+1} auf gleicher Seite von $v_i v_j$ liegen (oder $j = n$)
- Tangente ist **minimal/maximal**, wenn v_{j-1} oberhalb/unterhalb $v_i v_j$



1) Tangentensegmente

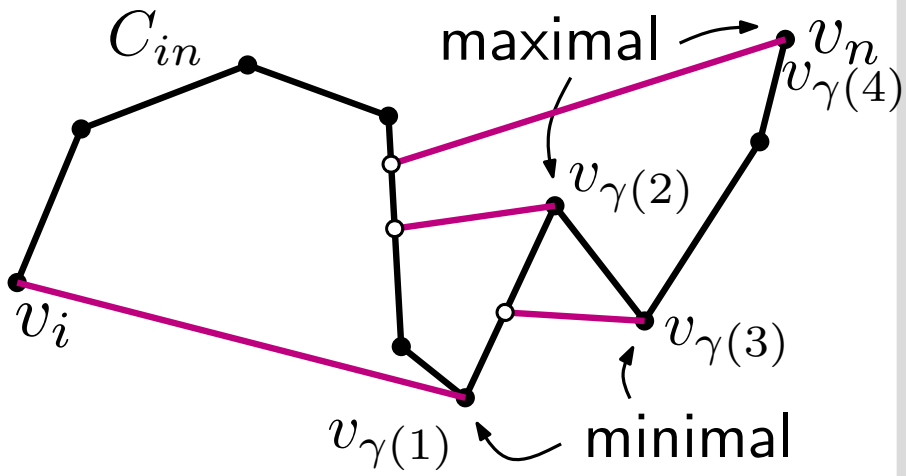
- shortcut $v_i v_j$ ist **Tangente** an C_{in}, wenn v_{j-1} und v_{j+1} auf gleicher Seite von $v_i v_j$ liegen (oder $j = n$)
- Tangente ist **minimal/maximal**, wenn v_{j-1} oberhalb/unterhalb $v_i v_j$
- **Tangentensegment** $\overline{w_j v_j}$, wobei w_j rechtester Punkt in $C_{in} \cap \overline{v_i v_j}$



- Tangenten $v_i v_{\gamma(1)}, \ldots, v_i v_{\gamma(r)}$ definieren Unterteilung S_i mit r Facetten
- Knoten mit höchstem Index jeder Facette definiert Tangentensegment und Facettenindex

1) Tangentensegmente

- shortcut v_iv_j ist **Tangente** an C_{in}, wenn v_{j-1} und v_{j+1} auf gleicher Seite von v_iv_j liegen (oder $j = n$)
- Tangente ist **minimal/maximal**, wenn v_{j-1} oberhalb/unterhalb v_iv_j
- **Tangentensegment** $\overline{w_jv_j}$, wobei w_j rechtester Punkt in $C_{in} \cap \overline{v_iv_j}$

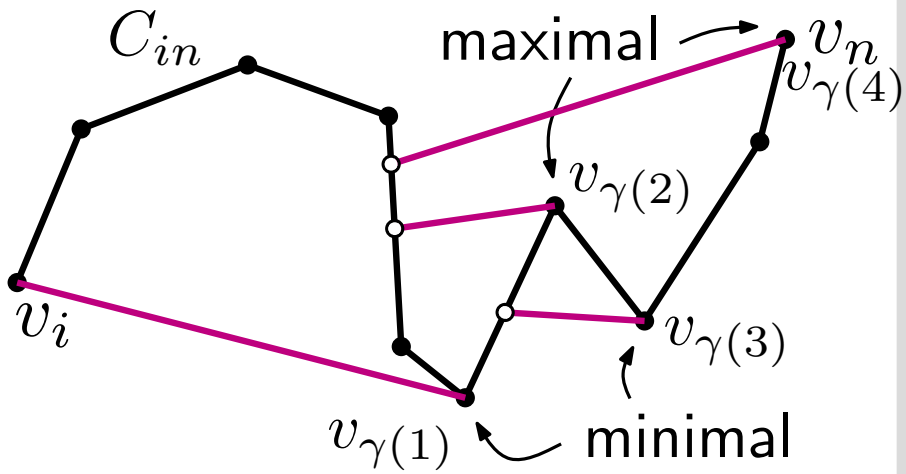


- Tangenten $v_iv_{\gamma(1)}, \ldots, v_iv_{\gamma(r)}$ definieren Unterteilung S_i mit r Facetten
- Knoten mit höchstem Index jeder Facette definiert Tangentensegment und Facettenindex

Lemma 2: Jede begrenzte Facette f von S_i ist θ-**monoton** bzgl. v_i, d.h. $|r \cap f| \in \{0, 1\}$ für jeden Strahl r von v_i

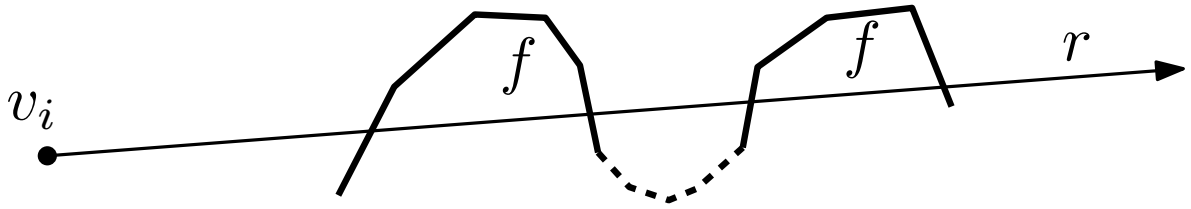
1) Tangentensegmente

- shortcut v_iv_j ist **Tangente** an C_{in}, wenn v_{j-1} und v_{j+1} auf gleicher Seite von v_iv_j liegen (oder $j = n$)
- Tangente ist **minimal/maximal**, wenn v_{j-1} oberhalb/unterhalb v_iv_j
- **Tangentensegment** $\overline{w_jv_j}$, wobei w_j rechtester Punkt in $C_{in} \cap \overline{v_iv_j}$



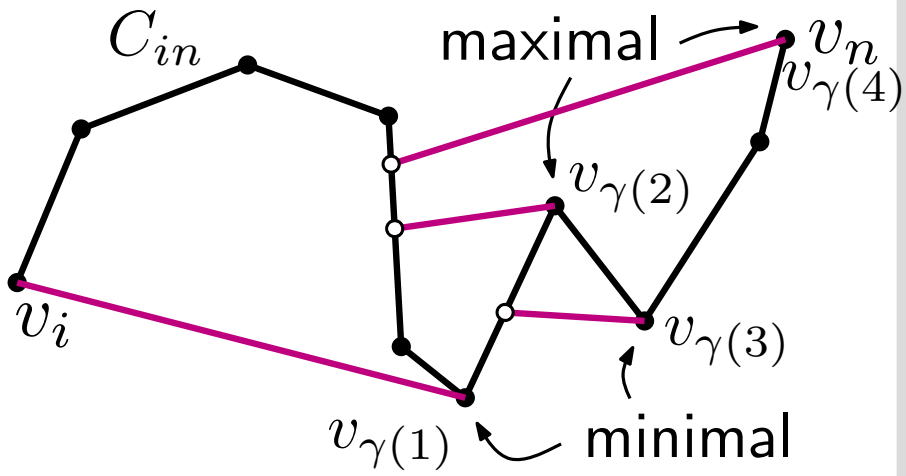
- Tangenten $v_iv_{\gamma(1)}, \ldots, v_iv_{\gamma(r)}$ definieren Unterteilung S_i mit r Facetten
- Knoten mit höchstem Index jeder Facette definiert Tangentensegment und Facettenindex

Lemma 2: Jede begrenzte Facette f von S_i ist θ-**monoton** bzgl. v_i, d.h. $|r \cap f| \in \{0, 1\}$ für jeden Strahl r von v_i



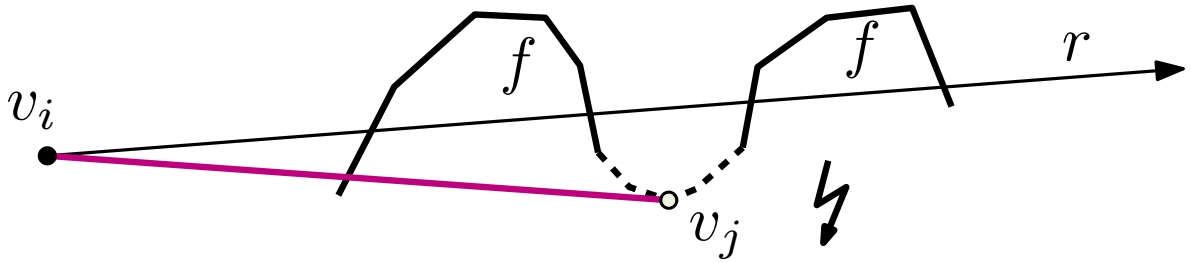
1) Tangentensegmente

- shortcut v_iv_j ist **Tangente** an C_{in}, wenn v_{j-1} und v_{j+1} auf gleicher Seite von v_iv_j liegen (oder $j = n$)
- Tangente ist **minimal/maximal**, wenn v_{j-1} oberhalb/unterhalb v_iv_j
- **Tangentensegment** $\overline{w_jv_j}$, wobei w_j rechtester Punkt in $C_{in} \cap \overline{v_iv_j}$



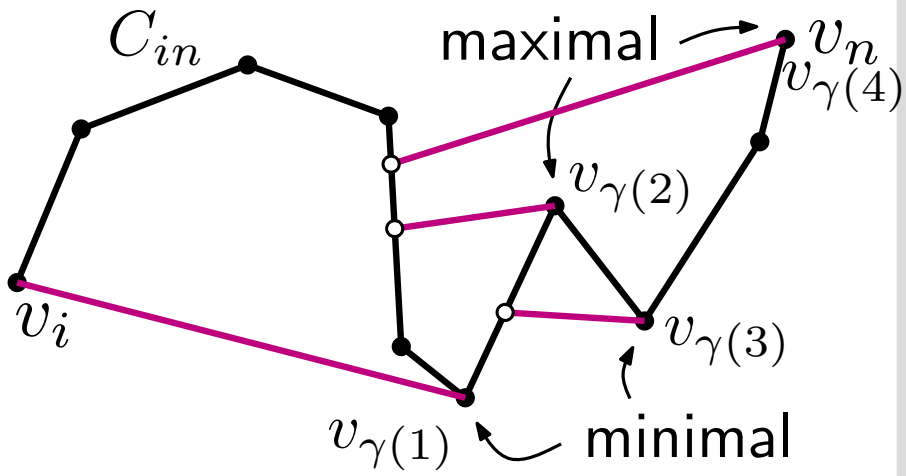
- Tangenten $v_iv_{\gamma(1)}, \ldots, v_iv_{\gamma(r)}$ definieren Unterteilung S_i mit r Facetten
- Knoten mit höchstem Index jeder Facette definiert Tangentensegment und Facettenindex

Lemma 2: Jede begrenzte Facette f von S_i ist θ-**monoton** bzgl. v_i, d.h. $|r \cap f| \in \{0, 1\}$ für jeden Strahl r von v_i



1) Tangentensegmente

- shortcut v_iv_j ist **Tangente** an C_{in}, wenn v_{j-1} und v_{j+1} auf gleicher Seite von v_iv_j liegen (oder $j = n$)
- Tangente ist **minimal/maximal**, wenn v_{j-1} oberhalb/unterhalb v_iv_j
- **Tangentensegment** $\overline{w_jv_j}$, wobei w_j rechtester Punkt in $C_{in} \cap \overline{v_iv_j}$



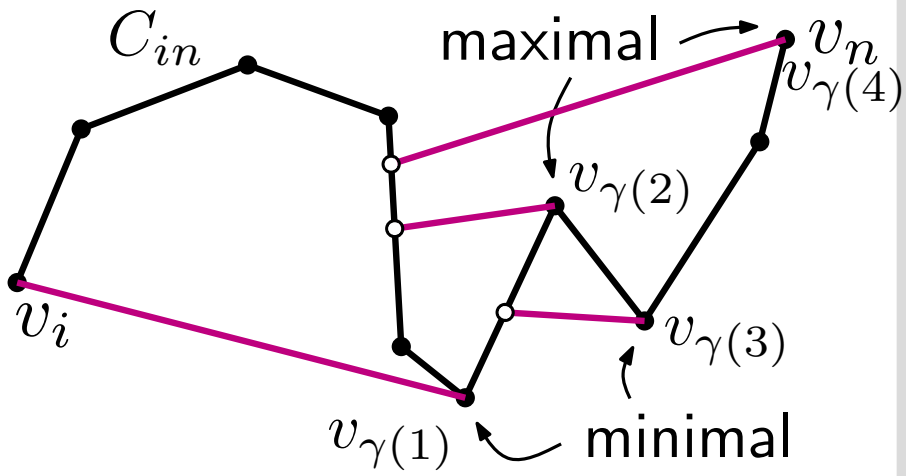
- Tangenten $v_iv_{\gamma(1)}, \ldots, v_iv_{\gamma(r)}$ definieren Unterteilung S_i mit r Facetten
- Knoten mit höchstem Index jeder Facette definiert Tangentensegment und Facettenindex

Lemma 2: Jede begrenzte Facette f von S_i ist θ-**monoton** bzgl. v_i, d.h. $|r \cap f| \in \{0, 1\}$ für jeden Strahl r von v_i

Lemma 3: Jede begrenzte Facette f von S_i besitzt genau einen zshg. Kantenzug, über den Strahlen r von v_i Facette f verlassen

1) Tangentensegmente

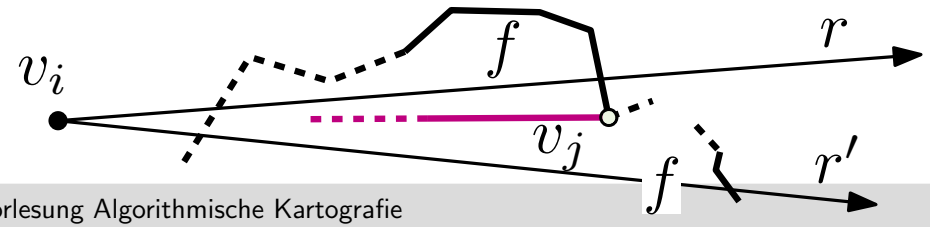
- shortcut v_iv_j ist **Tangente** an C_{in}, wenn v_{j-1} und v_{j+1} auf gleicher Seite von v_iv_j liegen (oder $j = n$)
- Tangente ist **minimal/maximal**, wenn v_{j-1} oberhalb/unterhalb v_iv_j
- **Tangentensegment** $\overline{w_jv_j}$, wobei w_j rechtester Punkt in $C_{in} \cap \overline{v_iv_j}$



- Tangenten $v_iv_{\gamma(1)}, \ldots, v_iv_{\gamma(r)}$ definieren Unterteilung S_i mit r Facetten
- Knoten mit höchstem Index jeder Facette definiert Tangentensegment und Facettenindex

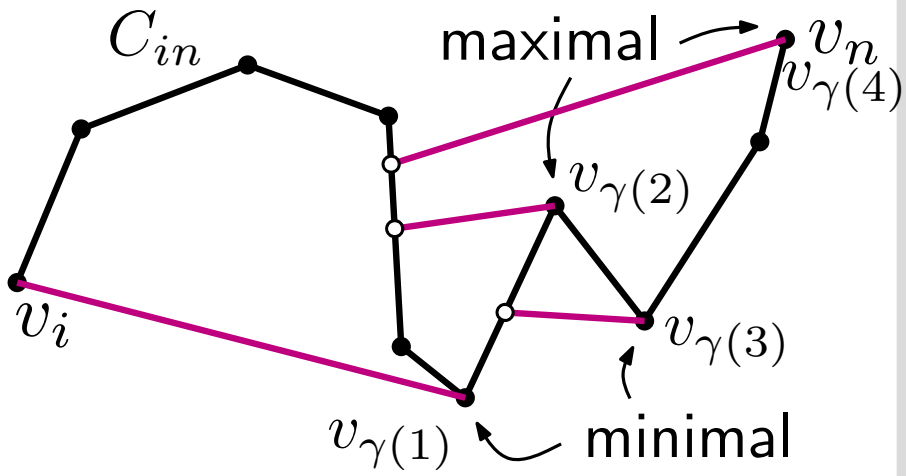
Lemma 2: Jede begrenzte Facette f von S_i ist θ-**monoton** bzgl. v_i, d.h. $|r \cap f| \in \{0, 1\}$ für jeden Strahl r von v_i

Lemma 3: Jede begrenzte Facette f von S_i besitzt genau einen zshg. Kantenzug, über den Strahlen r von v_i Facette f verlassen



1) Tangentensegmente

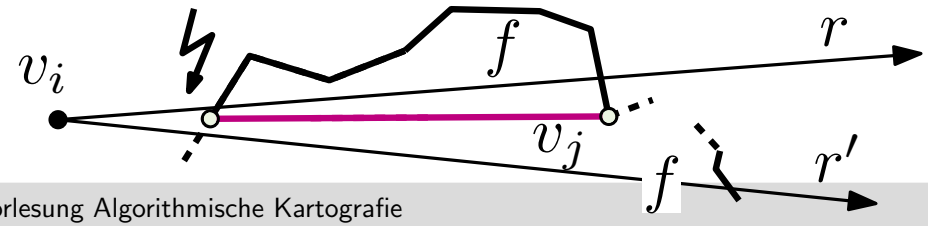
- shortcut v_iv_j ist **Tangente** an C_{in}, wenn v_{j-1} und v_{j+1} auf gleicher Seite von v_iv_j liegen (oder $j = n$)
- Tangente ist **minimal/maximal**, wenn v_{j-1} oberhalb/unterhalb v_iv_j
- **Tangentensegment** $\overline{w_jv_j}$, wobei w_j rechtester Punkt in $C_{in} \cap \overline{v_iv_j}$



- Tangenten $v_iv_{\gamma(1)}, \ldots, v_iv_{\gamma(r)}$ definieren Unterteilung S_i mit r Facetten
- Knoten mit höchstem Index jeder Facette definiert Tangentensegment und Facettenindex

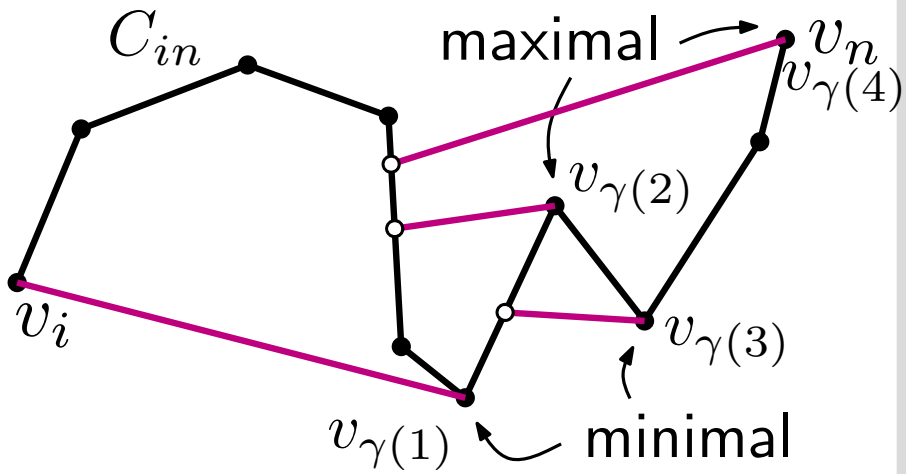
Lemma 2: Jede begrenzte Facette f von S_i ist θ-**monoton** bzgl. v_i, d.h. $|r \cap f| \in \{0, 1\}$ für jeden Strahl r von v_i

Lemma 3: Jede begrenzte Facette f von S_i besitzt genau einen zshg. Kantenzug, über den Strahlen r von v_i Facette f verlassen



1) Tangentensegmente

- shortcut v_iv_j ist **Tangente** an C_{in}, wenn v_{j-1} und v_{j+1} auf gleicher Seite von v_iv_j liegen (oder $j = n$)
- Tangente ist **minimal/maximal**, wenn v_{j-1} oberhalb/unterhalb v_iv_j
- **Tangentensegment** $\overline{w_jv_j}$, wobei w_j rechtester Punkt in $C_{in} \cap \overline{v_iv_j}$



- Tangenten $v_iv_{\gamma(1)}, \ldots, v_iv_{\gamma(r)}$ definieren Unterteilung S_i mit r Facetten
- Knoten mit höchstem Index jeder Facette definiert Tangentensegment und Facettenindex

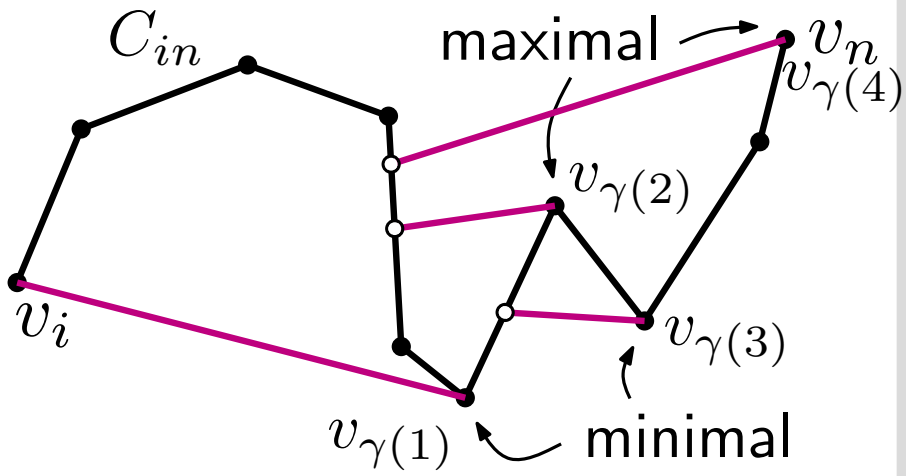
Lemma 2: Jede begrenzte Facette f von S_i ist θ-**monoton** bzgl. v_i, d.h. $|r \cap f| \in \{0, 1\}$ für jeden Strahl r von v_i

Lemma 3: Jede begrenzte Facette f von S_i besitzt genau einen zshg. Kantenzug, über den Strahlen r von v_i Facette f verlassen

Lemma 4: Jeder Strahl von v_i schneidet die Facetten von S_i in aufsteigender Reihenfolge

1) Tangentensegmente

- shortcut v_iv_j ist **Tangente** an C_{in}, wenn v_{j-1} und v_{j+1} auf gleicher Seite von v_iv_j liegen (oder $j = n$)
- Tangente ist **minimal/maximal**, wenn v_{j-1} oberhalb/unterhalb v_iv_j
- **Tangentensegment** $\overline{w_jv_j}$, wobei w_j rechtester Punkt in $C_{in} \cap \overline{v_iv_j}$

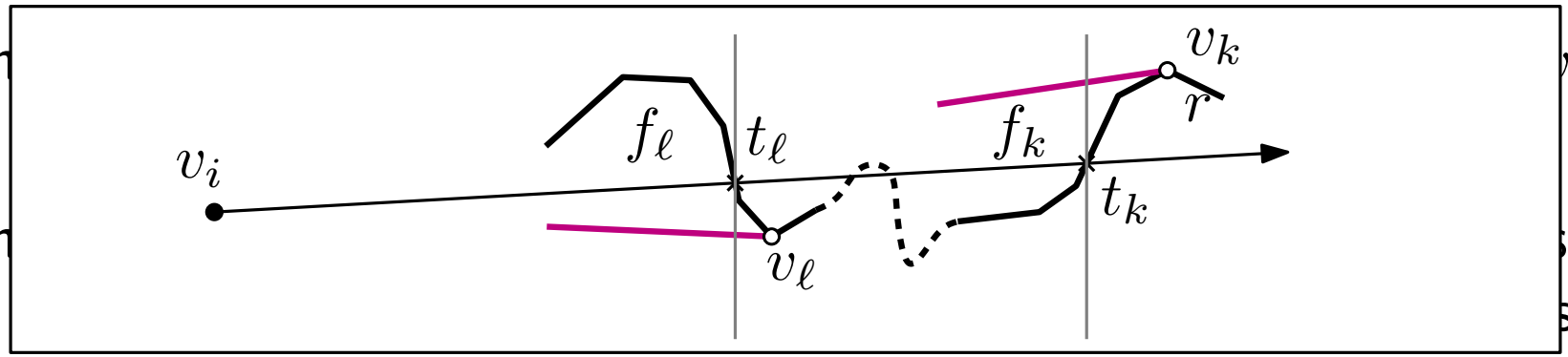


- Tangenten $v_iv_{\gamma(1)}, \ldots, v_iv_{\gamma(r)}$ definieren Unterteilung S_i mit r Facetten
- Knoten mit höchstem Index jeder Facette definiert Tangentensegment und Facettenindex

Lemm ... v_i, d.h.

Lemm ... shg.

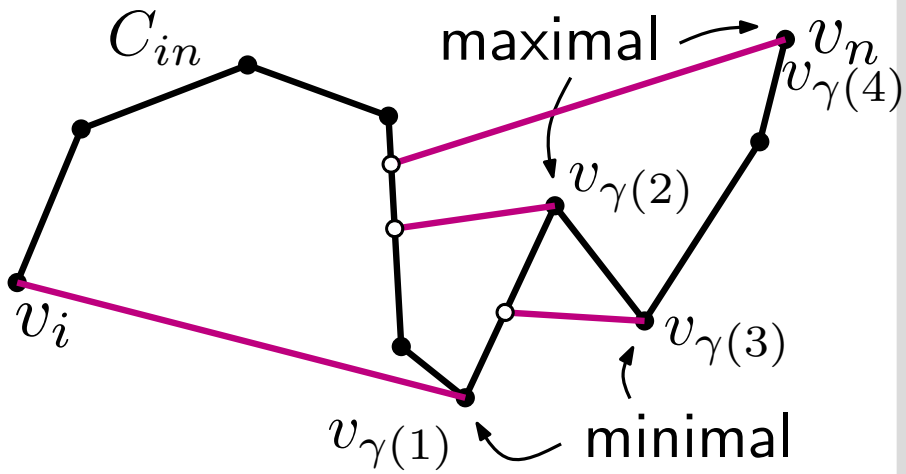
... ssen



Lemma 4: Jeder Strahl von v_i schneidet die Facetten von S_i in aufsteigender Reihenfolge

1) Tangentensegmente

- shortcut v_iv_j ist **Tangente** an C_{in}, wenn v_{j-1} und v_{j+1} auf gleicher Seite von v_iv_j liegen (oder $j = n$)
- Tangente ist **minimal/maximal**, wenn v_{j-1} oberhalb/unterhalb v_iv_j
- **Tangentensegment** $\overline{w_jv_j}$, wobei w_j rechtester Punkt in $C_{in} \cap \overline{v_iv_j}$



- Tangenten $v_iv_{\gamma(1)}, \ldots, v_iv_{\gamma(r)}$ definieren Unterteilung S_i mit r Facetten
- Knoten mit höchstem Index jeder Facette definiert Tangentensegment und Facettenindex

Lemma 2: Jede begrenzte Facette f von S_i ist θ-**monoton** bzgl. v_i, d.h. $|r \cap f| \in \{0, 1\}$ für jeden Strahl r von v_i

Lemma 3: Jede begrenzte Facette f von S_i besitzt genau einen zshg. Kantenzug, über den Strahlen r von v_i Facette f verlassen

Lemma 4: Jeder Strahl von v_i schneidet die Facetten von S_i in aufsteigender Reihenfolge

1) Tangentensegmente: Algorithmus

Karlsruhe Institute of Technology

Tangentensegmente(C, i)

Input: Kantenzug $C = (v_1, \ldots, v_n)$, Index $1 \leq i \leq n$
Output: Tangentensegmente bzgl. v_i
for $j = i + 1, \ldots, n$ **do**
  if $v_i v_j$ maximale Tangente **then**
    $k \leftarrow j - 1$
    while $v_i v_j \cap \overline{v_{k-1} v_k} = \emptyset$ **do**
      $k \leftarrow k - 1$
    $w_j \leftarrow v_i v_j \cap \overline{v_{k-1} v_k}$
  else if $v_i v_j$ minimale Tangente **then**
    ... /* analoges Vorgehen */

1) Tangentensegmente: Algorithmus

Karlsruhe Institute of Technology

Tangentensegmente(C, i)

Input: Kantenzug $C = (v_1, \ldots, v_n)$, Index $1 \leq i \leq n$
Output: Tangentensegmente bzgl. v_i
for $j = i + 1, \ldots, n$ **do**
  if $v_i v_j$ maximale Tangente **then**
    $k \leftarrow j - 1$
    while $v_i v_j \cap \overline{v_{k-1} v_k} = \emptyset$ **do**
      $k \leftarrow k - 1$
    $w_j \leftarrow v_i v_j \cap \overline{v_{k-1} v_k}$
  else if $v_i v_j$ minimale Tangente **then**
    ... /* analoges Vorgehen */

Laufzeit?

1) Tangentensegmente: Algorithmus

Karlsruhe Institute of Technology

Tangentensegmente(C, i)

Input: Kantenzug $C = (v_1, \ldots, v_n)$, Index $1 \leq i \leq n$
Output: Tangentensegmente bzgl. v_i
for $j = i + 1, \ldots, n$ **do**
  if $v_i v_j$ maximale Tangente **then**
    $k \leftarrow j - 1$
    while $v_i v_j \cap \overline{v_{k-1} v_k} = \emptyset$ **do**
      $k \leftarrow k - 1$
      if $v_i v_k$ maximale Tangente **then**
        $k \leftarrow h$, wobei $v_{h-1} v_h$ Punkt w_k enthält
    $w_j \leftarrow v_i v_j \cap \overline{v_{k-1} v_k}$
  else if $v_i v_j$ minimale Tangente **then**
    ... /* analoges Vorgehen */

Laufzeit?

1) Tangentensegmente: Algorithmus

Karlsruhe Institute of Technology

Tangentensegmente(C, i)

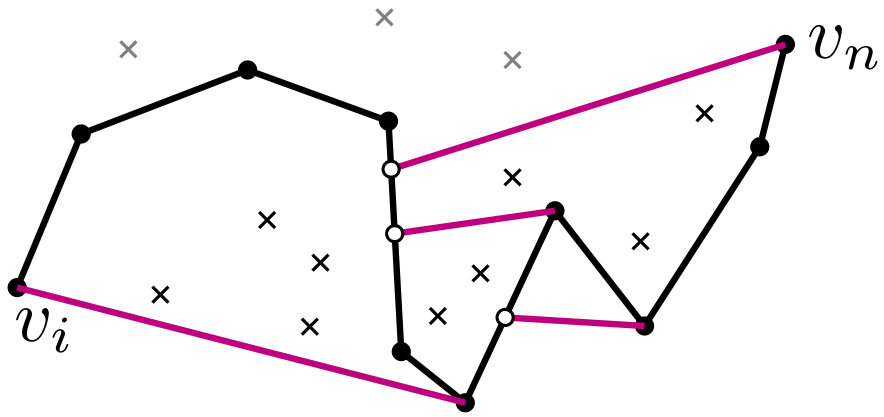
Input: Kantenzug $C = (v_1, \ldots, v_n)$, Index $1 \leq i \leq n$
Output: Tangentensegmente bzgl. v_i
for $j = i + 1, \ldots, n$ **do**
  if $v_i v_j$ maximale Tangente **then**
    $k \leftarrow j - 1$
    while $v_i v_j \cap \overline{v_{k-1} v_k} = \emptyset$ **do**
      $k \leftarrow k - 1$
      if $v_i v_k$ maximale Tangente **then**
        $k \leftarrow h$, wobei $v_{h-1} v_h$ Punkt w_k enthält
    $w_j \leftarrow v_i v_j \cap \overline{v_{k-1} v_k}$
  else if $v_i v_j$ minimale Tangente **then**
    ... `/* analoges Vorgehen */`

Laufzeit?

$\rightarrow$ Tangentensegmente(C, i) benötigt $O(n)$ Zeit

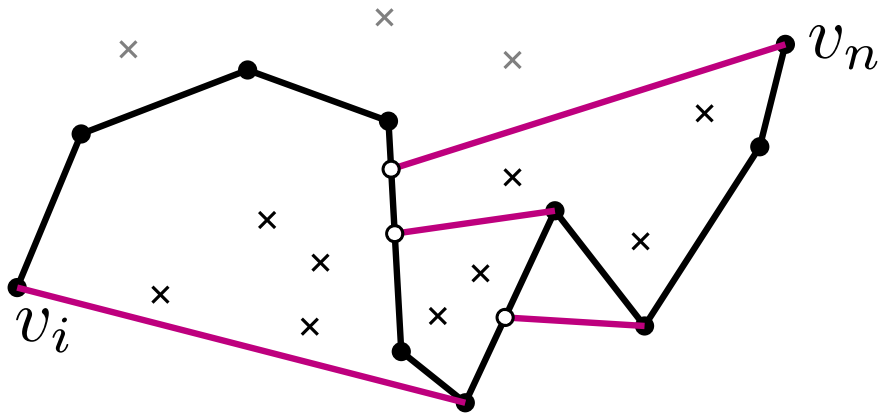
2) Punktzuweisung

- Punkte in äußerer Facette irrelevant
- weise Punkte in P Facetten von S_i zu



2) Punktzuweisung

- Punkte in äußerer Facette irrelevant
- weise Punkte in P Facetten von S_i zu

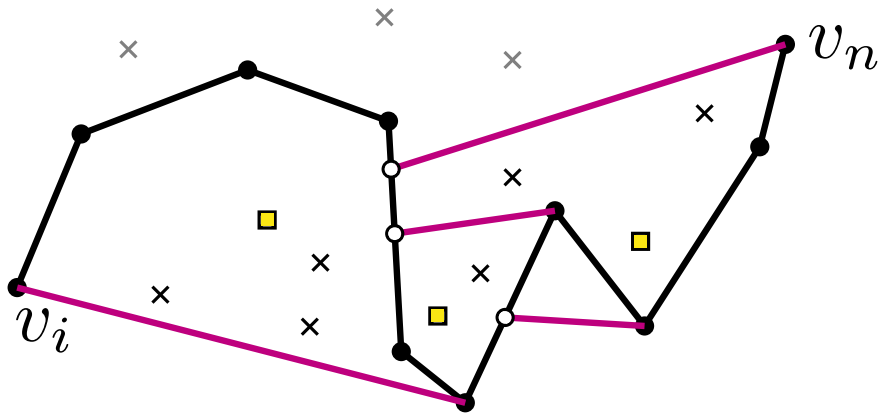


Sind alle verbleibenden Punkte nötig?

2) Punktzuweisung

- Punkte in äußerer Facette irrelevant
- weise Punkte in P Facetten von S_i zu
- für minimale Facette f, behalte Punkt p_f mit maximaler Steigung von $v_i p_f$
- für maximale Facette f, behalte Punkt p_f mit minimaler Steigung von $v_i p_f$

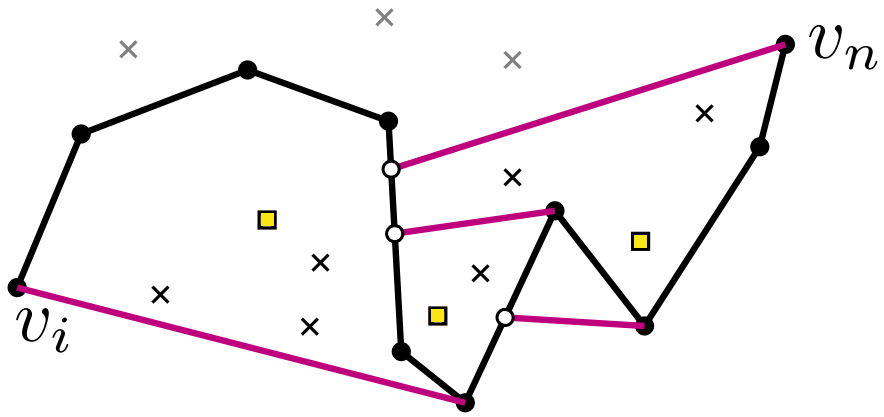
$\rightarrow$ verbleibende Punktmenge $P' \subseteq P$



2) Punktzuweisung

- Punkte in äußerer Facette irrelevant
- weise Punkte in P Facetten von S_i zu
- für minimale Facette f, behalte Punkt p_f mit maximaler Steigung von $v_i p_f$
- für maximale Facette f, behalte Punkt p_f mit minimaler Steigung von $v_i p_f$

$\rightarrow$ verbleibende Punktmenge $P' \subseteq P$



Lemma 5: Ein shortcut $v_i v_j$ ist konsistent für C_{ij} und P gdw. er konsistent für P' ist.

2) Punktzuweisung: Algorithmus

Punktzuweisung(S_i, i, P)

Input: Unterteilung S_i, Index $1 \leq i \leq n$, Punktmenge $P = \{p_1, \ldots, p_m\}$
Output: Punkt p_f für jede Facette f
$P' \leftarrow$ Punkte aus P rechts von v_i
priority queue $Q \leftarrow P' \cup \{v_{i+1}, \ldots, v_n\}$ sortiert nach Winkel bzgl. v_i
$T \leftarrow$ leerer binärer Suchbaum für radialen Sweep
while Q nicht leer **do**
    $p \leftarrow Q$.extractMax
    if p Knoten von C_{in} **then**
        update T mit Kanten von p
    else
        $e \leftarrow$ linkeste Kante $v_k v_{k+1}$ in T rechts von p auf sweep line
        speichere p an e
gehe über alle Kanten von C_{in} und weise Facetten f maximalen/minimalen Punkt p_f zu

2) Punktzuweisung: Algorithmus

Karlsruhe Institute of Technology

Punktzuweisung(S_i, i, P)

Input: Unterteilung S_i, Index $1 \leq i \leq n$, Punktmenge $P = \{p_1, \ldots, p_m\}$
Output: Punkt p_f für jede Facette f
$P' \leftarrow$ Punkte aus P rechts von v_i
priority queue $Q \leftarrow P' \cup \{v_{i+1}, \ldots, v_n\}$ sortiert nach Winkel bzgl. v_i
$T \leftarrow$ leerer binärer Suchbaum für radialen Sweep
while Q nicht leer **do**
    $p \leftarrow Q$.extractMax
    if p Knoten von C_{in} **then**
        update T mit Kanten von p
    else
        $e \leftarrow$ linkeste Kante $v_k v_{k+1}$ in T rechts von p auf sweep line
        speichere p an e
gehe über alle Kanten von C_{in} und weise Facetten f maximalen/minimalen Punkt p_f zu

Laufzeit $O((n+m)\log(n+m))$

2) Punktzuweisung: Algorithmus

Karlsruhe Institute of Technology

Punktzuweisung(S_i, i, P)

Input: Unterteilung S_i, Index $1 \leq i \leq n$, Punktmenge $P = \{p_1, \ldots, p_m\}$
Output: Punkt p_f für jede Facette f
$P' \leftarrow$ Punkte aus P rechts von v_i
priority queue $Q \leftarrow P' \cup \{v_{i+1}, \ldots, v_n\}$ sortiert nach Winkel bzgl. v_i
$T \leftarrow$ leerer binärer Suchbaum für radialen Sweep
while Q nicht leer **do**
    $p \leftarrow Q$.extractMax
    if p Knoten von C_{in} **then**
        update T mit Kanten von p
    else
        $e \leftarrow$ linkeste Kante $v_k v_{k+1}$ in T rechts von p auf sweep line
        speichere p an e
gehe über alle Kanten von C_{in} und weise Facetten f maximalen/minimalen Punkt p_f zu

später

Laufzeit $O((n+m)\log(n+m))$

3) Konsistenzberechnung

Input: Unterteilung S_i mit Facetten $f_1, \ldots, f_k$, Punkt $p_r \in P \cup \{\perp\}$
für jede Facette f_r, Knoten v_i

Output: Menge A aller konsistenten shortcuts von v_i

$Q \leftarrow$ double-ended queue für shortcuts $v_i v_j$ absteigend sortiert nach Steigung mit Pointern zw. $v_i v_j$ und v_j

$j \leftarrow i + 1$

for $h = 1, \ldots, k$ **do**

- **if** $p_h \neq \perp$ **then**
  - **if** f_h maximale Facette **then**
    - entferne shortcuts vom Anfang von Q
    - solange Steigung größer als $v_i p_h$
  - **else**
    - entferne shortcuts vom Ende von Q
    - solange Steigung kleiner als $v_i p_h$
- **while** $v_j \neq v_{\gamma(h)}$ **do**
  - **if** $v_i v_j \in Q$ **then** A.add($v_i v_j$); Q.remove($v_i v_j$)
  - $j \leftarrow j + 1$

3) Konsistenzberechnung

Input: Unterteilung S_i mit Facetten $f_1, \ldots, f_k$, Punkt $p_r \in P \cup \{\perp\}$
für jede Facette f_r, Knoten v_i

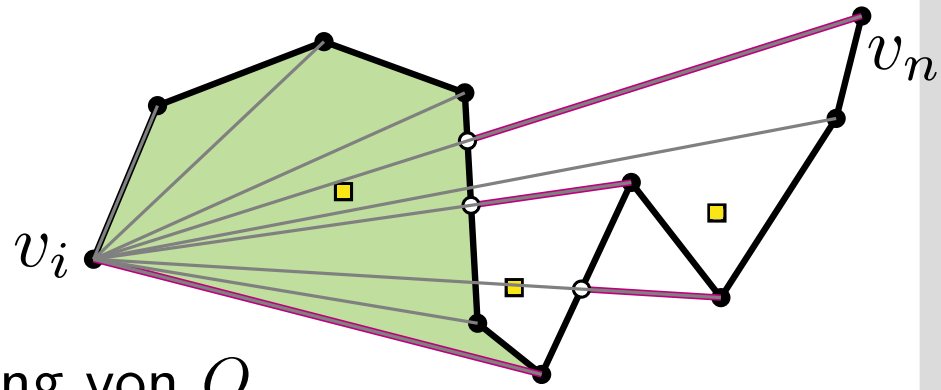
Output: Menge A aller konsistenten shortcuts von v_i

$Q \leftarrow$ double-ended queue für shortcuts $v_i v_j$ absteigend sortiert nach Steigung mit Pointern zw. $v_i v_j$ und v_j

$j \leftarrow i + 1$

for $h = 1, \ldots, k$ **do**

- **if** $p_h \neq \perp$ **then**
  - **if** f_h maximale Facette **then**
    - entferne shortcuts vom Anfang von Q
    - solange Steigung größer als $v_i p_h$
  - **else**
    - entferne shortcuts vom Ende von Q
    - solange Steigung kleiner als $v_i p_h$
- **while** $v_j \neq v_{\gamma(h)}$ **do**
  - **if** $v_i v_j \in Q$ **then** A.add($v_i v_j$); Q.remove($v_i v_j$)
  - $j \leftarrow j + 1$



3) Konsistenzberechnung

Input: Unterteilung S_i mit Facetten $f_1, \ldots, f_k$, Punkt $p_r \in P \cup \{\perp\}$
für jede Facette f_r, Knoten v_i

Output: Menge A aller konsistenten shortcuts von v_i

$Q \leftarrow$ double-ended queue für shortcuts $v_i v_j$ absteigend sortiert nach Steigung mit Pointern zw. $v_i v_j$ und v_j

$j \leftarrow i + 1$

for $h = 1, \ldots, k$ **do**

  if $p_h \neq \perp$ **then**

    if f_h maximale Facette **then**

      entferne shortcuts vom Anfang von Q
      solange Steigung größer als $v_i p_h$

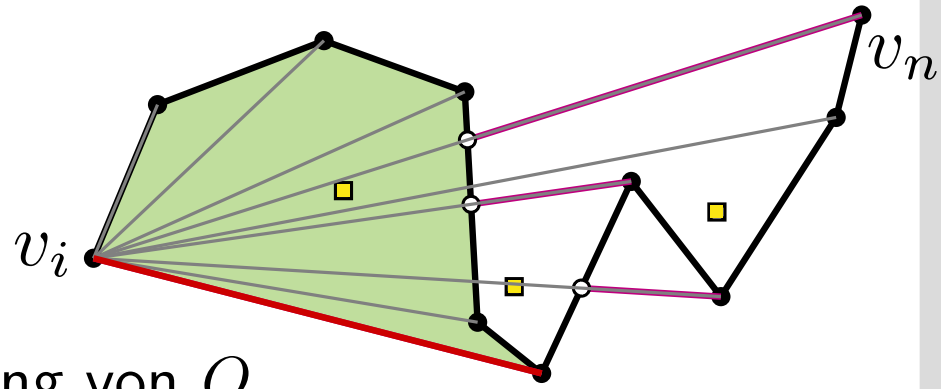
    else

      entferne shortcuts vom Ende von Q
      solange Steigung kleiner als $v_i p_h$

  while $v_j \neq v_{\gamma(h)}$ **do**

    if $v_i v_j \in Q$ **then** A.add($v_i v_j$); Q.remove($v_i v_j$)

    $j \leftarrow j + 1$



3) Konsistenzberechnung

Input: Unterteilung S_i mit Facetten $f_1, \ldots, f_k$, Punkt $p_r \in P \cup \{\perp\}$
für jede Facette f_r, Knoten v_i

Output: Menge A aller konsistenten shortcuts von v_i

$Q \leftarrow$ double-ended queue für shortcuts $v_i v_j$ absteigend sortiert nach Steigung mit Pointern zw. $v_i v_j$ und v_j

$j \leftarrow i + 1$

for $h = 1, \ldots, k$ **do**

  if $p_h \neq \perp$ **then**

    if f_h maximale Facette **then**

      entferne shortcuts vom Anfang von Q
      solange Steigung größer als $v_i p_h$

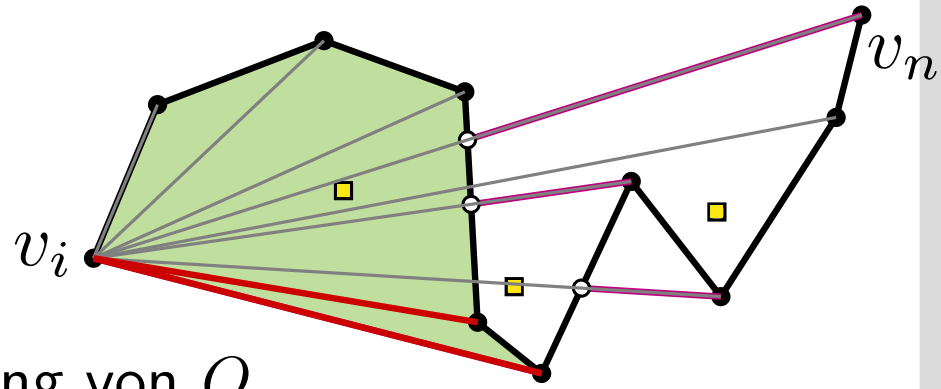
    else

      entferne shortcuts vom Ende von Q
      solange Steigung kleiner als $v_i p_h$

  while $v_j \neq v_{\gamma(h)}$ **do**

    if $v_i v_j \in Q$ **then** A.add($v_i v_j$); Q.remove($v_i v_j$)

    $j \leftarrow j + 1$



3) Konsistenzberechnung

Input: Unterteilung S_i mit Facetten $f_1, \ldots, f_k$, Punkt $p_r \in P \cup \{\perp\}$
für jede Facette f_r, Knoten v_i

Output: Menge A aller konsistenten shortcuts von v_i

$Q \leftarrow$ double-ended queue für shortcuts $v_i v_j$ absteigend sortiert nach Steigung mit Pointern zw. $v_i v_j$ und v_j

$j \leftarrow i + 1$

for $h = 1, \ldots, k$ **do**

  if $p_h \neq \perp$ **then**

    if f_h maximale Facette **then**

      entferne shortcuts vom Anfang von Q
      solange Steigung größer als $v_i p_h$

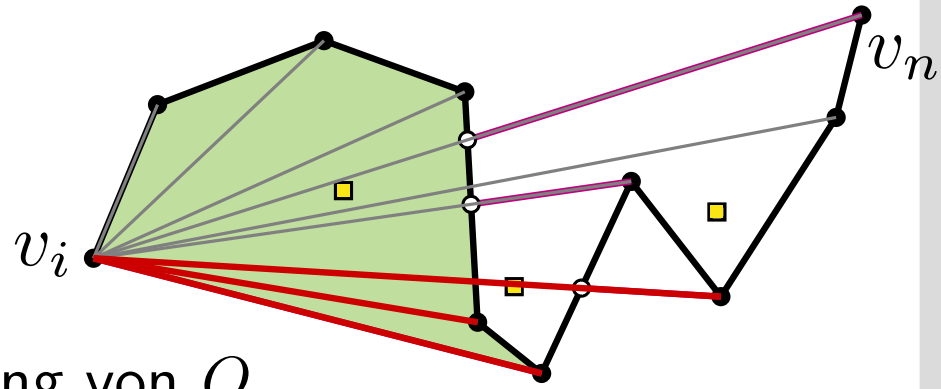
    else

      entferne shortcuts vom Ende von Q
      solange Steigung kleiner als $v_i p_h$

  while $v_j \neq v_{\gamma(h)}$ **do**

    if $v_i v_j \in Q$ **then** A.add($v_i v_j$); Q.remove($v_i v_j$)

    $j \leftarrow j + 1$



3) Konsistenzberechnung

Input: Unterteilung S_i mit Facetten $f_1, \ldots, f_k$, Punkt $p_r \in P \cup \{\perp\}$
für jede Facette f_r, Knoten v_i

Output: Menge A aller konsistenten shortcuts von v_i

$Q \leftarrow$ double-ended queue für shortcuts $v_i v_j$ absteigend sortiert nach Steigung mit Pointern zw. $v_i v_j$ und v_j

$j \leftarrow i + 1$

for $h = 1, \ldots, k$ **do**

  if $p_h \neq \perp$ **then**

    if f_h maximale Facette **then**

      entferne shortcuts vom Anfang von Q
      solange Steigung größer als $v_i p_h$

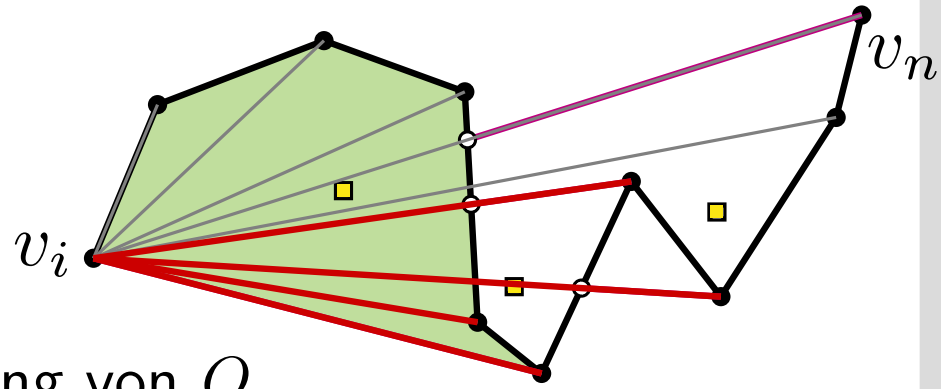
    else

      entferne shortcuts vom Ende von Q
      solange Steigung kleiner als $v_i p_h$

  while $v_j \neq v_{\gamma(h)}$ **do**

    if $v_i v_j \in Q$ **then** A.add($v_i v_j$); Q.remove($v_i v_j$)

    $j \leftarrow j + 1$



3) Konsistenzberechnung

Input: Unterteilung S_i mit Facetten $f_1, \ldots, f_k$, Punkt $p_r \in P \cup \{\perp\}$
für jede Facette f_r, Knoten v_i

Output: Menge A aller konsistenten shortcuts von v_i

$Q \leftarrow$ double-ended queue für shortcuts $v_i v_j$ absteigend sortiert nach Steigung mit Pointern zw. $v_i v_j$ und v_j

$j \leftarrow i + 1$

for $h = 1, \ldots, k$ **do**

  if $p_h \neq \perp$ **then**

    if f_h maximale Facette **then**

      entferne shortcuts vom Anfang von Q
      solange Steigung größer als $v_i p_h$

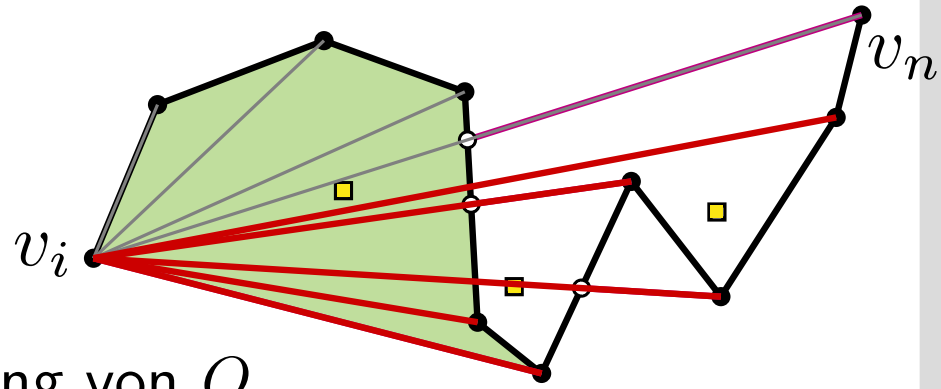
    else

      entferne shortcuts vom Ende von Q
      solange Steigung kleiner als $v_i p_h$

  while $v_j \neq v_{\gamma(h)}$ **do**

    if $v_i v_j \in Q$ **then** A.add($v_i v_j$); Q.remove($v_i v_j$)

    $j \leftarrow j + 1$



3) Konsistenzberechnung

Input: Unterteilung S_i mit Facetten $f_1, \ldots, f_k$, Punkt $p_r \in P \cup \{\perp\}$
für jede Facette f_r, Knoten v_i

Output: Menge A aller konsistenten shortcuts von v_i

$Q \leftarrow$ double-ended queue für shortcuts $v_i v_j$ absteigend sortiert nach Steigung mit Pointern zw. $v_i v_j$ und v_j

$j \leftarrow i + 1$

for $h = 1, \ldots, k$ **do**

  if $p_h \neq \perp$ **then**

    if f_h maximale Facette **then**

      entferne shortcuts vom Anfang von Q
      solange Steigung größer als $v_i p_h$

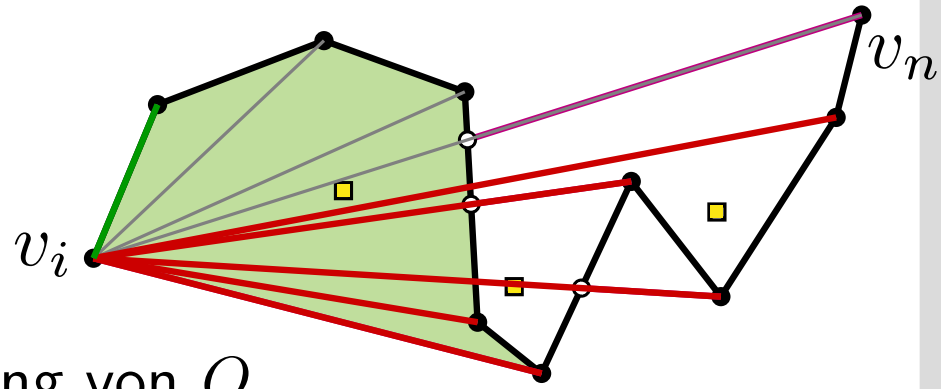
    else

      entferne shortcuts vom Ende von Q
      solange Steigung kleiner als $v_i p_h$

  while $v_j \neq v_{\gamma(h)}$ **do**

    if $v_i v_j \in Q$ **then** A.add($v_i v_j$); Q.remove($v_i v_j$)

    $j \leftarrow j + 1$



3) Konsistenzberechnung

Input: Unterteilung S_i mit Facetten $f_1, \ldots, f_k$, Punkt $p_r \in P \cup \{\perp\}$
für jede Facette f_r, Knoten v_i

Output: Menge A aller konsistenten shortcuts von v_i

$Q \leftarrow$ double-ended queue für shortcuts $v_i v_j$ absteigend sortiert nach Steigung mit Pointern zw. $v_i v_j$ und v_j

$j \leftarrow i + 1$

for $h = 1, \ldots, k$ **do**

  if $p_h \neq \perp$ **then**

    if f_h maximale Facette **then**

      entferne shortcuts vom Anfang von Q
      solange Steigung größer als $v_i p_h$

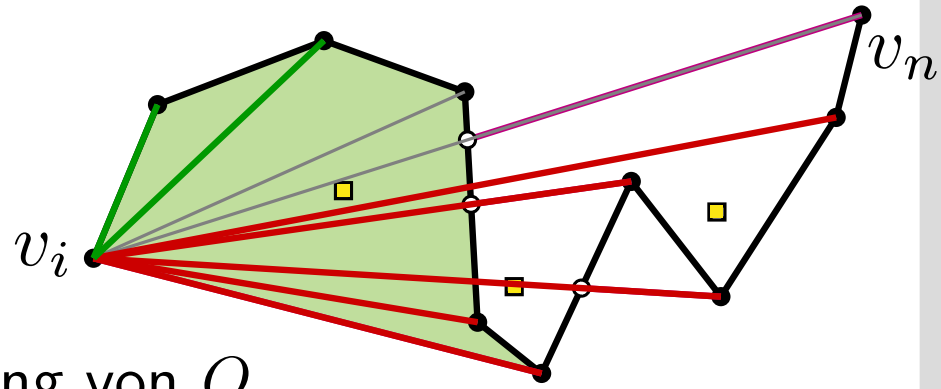
    else

      entferne shortcuts vom Ende von Q
      solange Steigung kleiner als $v_i p_h$

  while $v_j \neq v_{\gamma(h)}$ **do**

    if $v_i v_j \in Q$ **then** A.add($v_i v_j$); Q.remove($v_i v_j$)

    $j \leftarrow j + 1$



3) Konsistenzberechnung

Input: Unterteilung S_i mit Facetten $f_1, \ldots, f_k$, Punkt $p_r \in P \cup \{\perp\}$
für jede Facette f_r, Knoten v_i

Output: Menge A aller konsistenten shortcuts von v_i

$Q \leftarrow$ double-ended queue für shortcuts $v_i v_j$ absteigend sortiert nach Steigung mit Pointern zw. $v_i v_j$ und v_j

$j \leftarrow i + 1$

for $h = 1, \ldots, k$ **do**

  if $p_h \neq \perp$ **then**

    if f_h maximale Facette **then**

      entferne shortcuts vom Anfang von Q
      solange Steigung größer als $v_i p_h$

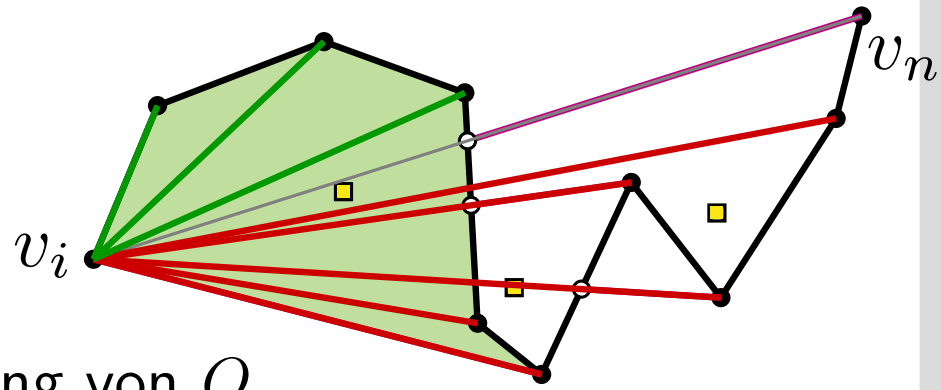
    else

      entferne shortcuts vom Ende von Q
      solange Steigung kleiner als $v_i p_h$

  while $v_j \neq v_{\gamma(h)}$ **do**

    if $v_i v_j \in Q$ **then** A.add($v_i v_j$); Q.remove($v_i v_j$)

    $j \leftarrow j + 1$



3) Konsistenzberechnung

Input: Unterteilung S_i mit Facetten $f_1, \ldots, f_k$, Punkt $p_r \in P \cup \{\perp\}$
für jede Facette f_r, Knoten v_i

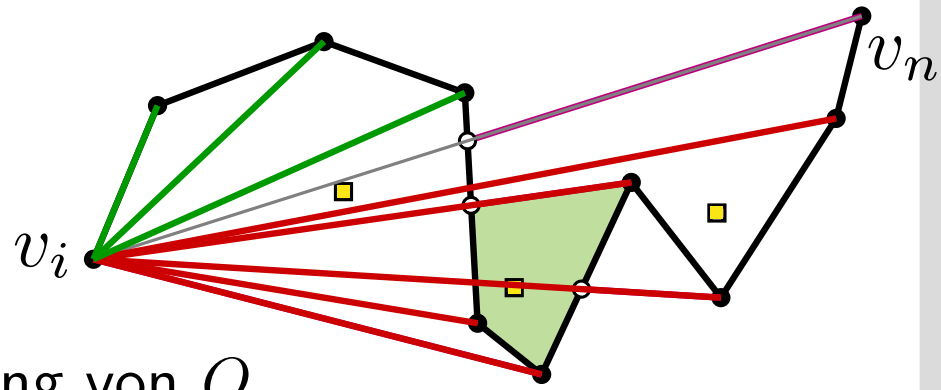
Output: Menge A aller konsistenten shortcuts von v_i

$Q \leftarrow$ double-ended queue für shortcuts $v_i v_j$ absteigend sortiert nach Steigung mit Pointern zw. $v_i v_j$ und v_j

$j \leftarrow i + 1$

for $h = 1, \ldots, k$ **do**

- **if** $p_h \neq \perp$ **then**
  - **if** f_h maximale Facette **then**
    - entferne shortcuts vom Anfang von Q
    - solange Steigung größer als $v_i p_h$
  - **else**
    - entferne shortcuts vom Ende von Q
    - solange Steigung kleiner als $v_i p_h$
- **while** $v_j \neq v_{\gamma(h)}$ **do**
  - **if** $v_i v_j \in Q$ **then** A.add($v_i v_j$); Q.remove($v_i v_j$)
  - $j \leftarrow j + 1$



3) Konsistenzberechnung

Input: Unterteilung S_i mit Facetten $f_1, \ldots, f_k$, Punkt $p_r \in P \cup \{\perp\}$
für jede Facette f_r, Knoten v_i

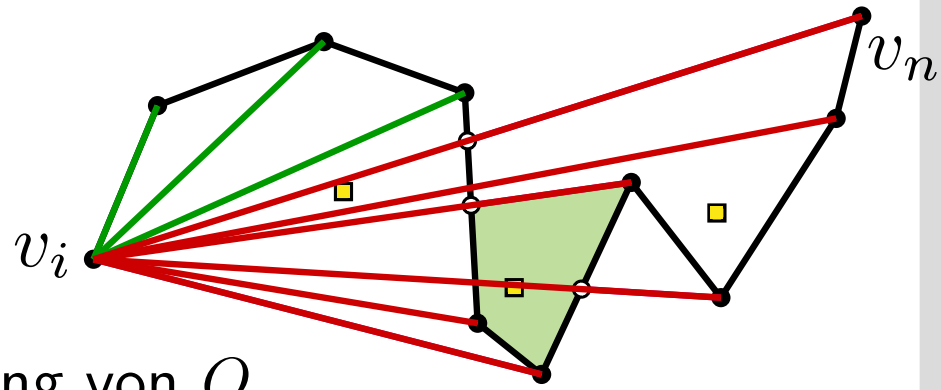
Output: Menge A aller konsistenten shortcuts von v_i

$Q \leftarrow$ double-ended queue für shortcuts $v_i v_j$ absteigend sortiert nach Steigung mit Pointern zw. $v_i v_j$ und v_j

$j \leftarrow i + 1$

for $h = 1, \ldots, k$ **do**

- **if** $p_h \neq \perp$ **then**
  - **if** f_h maximale Facette **then**
    - entferne shortcuts vom Anfang von Q
    - solange Steigung größer als $v_i p_h$
  - **else**
    - entferne shortcuts vom Ende von Q
    - solange Steigung kleiner als $v_i p_h$
- **while** $v_j \neq v_{\gamma(h)}$ **do**
  - **if** $v_i v_j \in Q$ **then** A.add($v_i v_j$); Q.remove($v_i v_j$)
  - $j \leftarrow j + 1$



3) Konsistenzberechnung

Input: Unterteilung S_i mit Facetten $f_1, \ldots, f_k$, Punkt $p_r \in P \cup \{\perp\}$
für jede Facette f_x, Knoten v_i

Output: Menge A aller konsistenten shortcuts von v_i

$Q \leftarrow$ double-ended queue mit shortcuts $v_i v_j$ absteigend sortiert nach Steigung mit Pointern zw. $v_i v_j$ und v_j

$j \leftarrow i + 1$

for $h = 1, \ldots, k$ **do**

  if $p_h \neq \perp$ **then**

    if f_h maximale Facette **then**

      entferne shortcuts vom Anfang von Q

      solange Steigung größer als $v_i p_h$

    else

      entferne shortcuts vom Ende von Q

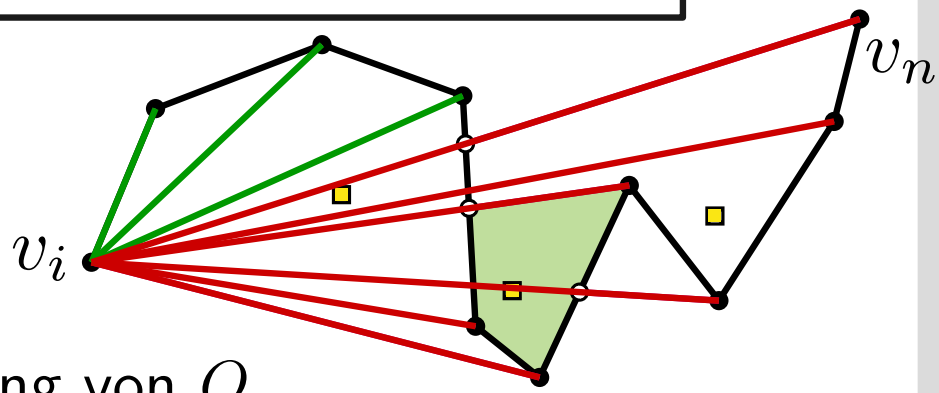
      solange Steigung kleiner als $v_i p_h$

  while $v_j \neq v_{\gamma(h)}$ **do**

    if $v_i v_j \in Q$ **then** A.add($v_i v_j$); Q.remove($v_i v_j$)

    $j \leftarrow j + 1$

Lemma 6: Die Berechnung der shortcuts in A ist korrekt und benötigt $O(n \log n)$ Zeit.



Zusammenfassung

Lemma 7: Für x-monotonen Kantenzug C mit n Knoten und Menge P von m Punkten können alle konsistenten shortcuts für einen Knoten v von C in Zeit $O((n+m)\log n)$ berechnet werden.

Zusammenfassung

Lemma 7: Für x-monotonen Kantenzug C mit n Knoten und Menge P von m Punkten können alle konsistenten shortcuts für einen Knoten v von C in Zeit $O((n+m)\log n)$ berechnet werden.

Beweisskizze:

- Schritt 1: Tangentensegmente berechnen $O(n)$
- Schritt 2: Punktzuweisung $O((n+m)\log n)$
- Schritt 3: Konsistenzberechnung $O(n \log n)$

Zusammenfassung

Lemma 7: Für x-monotonen Kantenzug C mit n Knoten und Menge P von m Punkten können alle konsistenten shortcuts für einen Knoten v von C in Zeit $O((n+m)\log n)$ berechnet werden.

Satz 2: Für x-monotonen Kantenzug C mit n Knoten, Menge P von m Punkten und Fehlerschranke ε kann die kleinste konsistente Vereinfachung C' mit Fehler $\leq \varepsilon$ in Zeit $O(n(n+m)\log n)$ berechnet werden.

Zusammenfassung

Lemma 7: Für x-monotonen Kantenzug C mit n Knoten und Menge P von m Punkten können alle konsistenten shortcuts für einen Knoten v von C in Zeit $O((n+m)\log n)$ berechnet werden.

Satz 2: Für x-monotonen Kantenzug C mit n Knoten, Menge P von m Punkten und Fehlerschranke ε kann die kleinste konsistente Vereinfachung C' mit Fehler $\leq \varepsilon$ in Zeit $O(n(n+m)\log n)$ berechnet werden.

Beweisskizze:

- Graph $G_1 = (V, A_1)$ mit Algorithmus von Imai/Iri $O(n^2)$
- Graph $G_2 = (V, A_2) \leftarrow n$-fache Anwendung von Lemma 7 $O(n(n+m)\log n)$
- Graph $G = (V, A_1 \cap A_2)$ bilden $O(n^2)$
- kürzesten $v_1 v_n$-Pfad in G suchen $O(n^2)$

Übungen

1) Zeigen Sie, dass die Methode *Punktzuweisung* die Laufzeit $O((n+m)\log n)$ zulässt.

2) Überlegen Sie sich basierend auf geometrischen Beobachtungen ein Beschleunigungsverfahren, mit dessen Hilfe die Punktmenge P für die Vereinfachung eines Polygonzugs C sinnvoll eingeschränkt werden kann.

3) Überlegen Sie sich eine Heuristik, mit deren Hilfe die Konsistenzberechnung von shortcuts im Gesamtalgorithmus beschleunigt werden kann.

Zusammenfassung Linienvereinfachung

- typische Qualitätskriterien: Abstandsmaß, Anzahl Segmente

Zusammenfassung Linienvereinfachung

- typische Qualitätskriterien: Abstandsmaß, Anzahl Segmente

- einfache Greedy-Algorithmen
- Douglas-Peucker Algorithmus (und Beschleunigung)
- Dreiecksbasiertes Verfahren von Visvalingam/Whyatt
- exakte Optimierung als kürzeste-Wege-Problem in Graphen
- topologisch konsistente Vereinfachung von de Berg et al.

Zusammenfassung Linienvereinfachung

- typische Qualitätskriterien: Abstandsmaß, Anzahl Segmente

- einfache Greedy-Algorithmen
- Douglas-Peucker Algorithmus (und Beschleunigung)
- Dreiecksbasiertes Verfahren von Visvalingam/Whyatt
- exakte Optimierung als kürzeste-Wege-Problem in Graphen
- topologisch konsistente Vereinfachung von de Berg et al.

- de Berg et al. (1998) stellen noch weitere Anpassungen für einige praktisch relevante Fälle vor (z.B. nicht x-monoton)

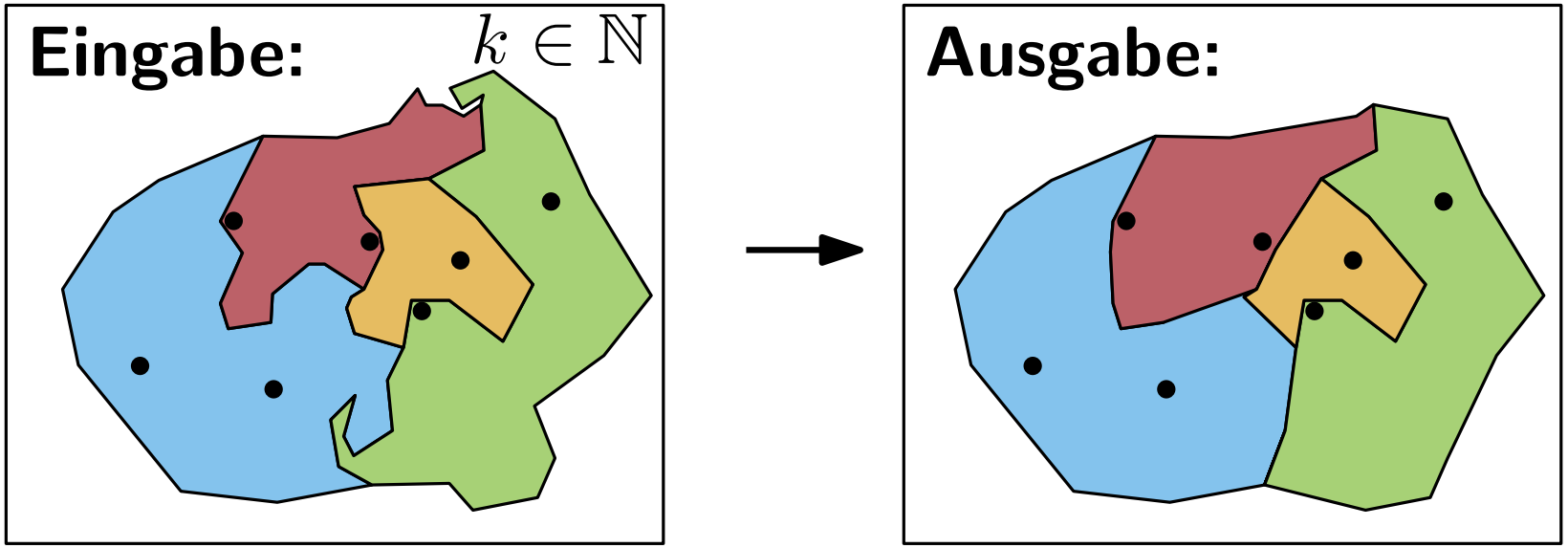
Zusammenfassung Linienvereinfachung

- typische Qualitätskriterien: Abstandsmaß, Anzahl Segmente
- einfache Greedy-Algorithmen
- Douglas-Peucker Algorithmus (und Beschleunigung)
- Dreiecksbasiertes Verfahren von Visvalingam/Whyatt
- exakte Optimierung als kürzeste-Wege-Problem in Graphen
- topologisch konsistente Vereinfachung von de Berg et al.
- de Berg et al. (1998) stellen noch weitere Anpassungen für einige praktisch relevante Fälle vor (z.B. nicht x-monoton)
- für Linienvereinfachung unter Hinzunahme zusätzlicher Knoten ist die Segmentminimierung NP-schwer [Guibas et al. '93]

Projektarbeit: Kartenvereinfachung

Projekt: Vereinfachung von Regionen

Ziel: Algorithmus zur Vereinfachung von Regionen einer Karte.



Anforderungen:

- Topologie bleibt erhalten.
- Vereinfachte Regionen besitzen maximal k Kanten.

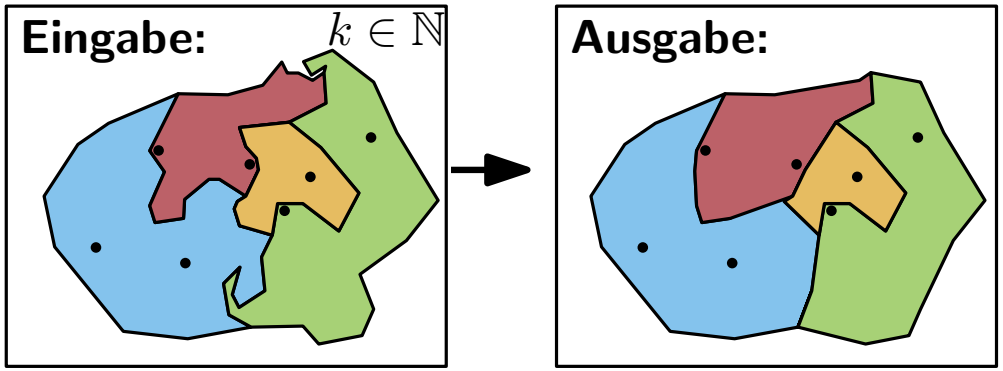
Aufgabe:

- Entwicklung eines Algorithmus.
- Implementierung des Algorithmus.
- Testen der Implementierung.

Eckdaten des Projekts

Team: Jeweils 2 Personen

Programmiersprache:
Frei wählbar.
Empfehlung: Java oder Python



Ablauf:

- Bis 13. Mai: Treffen mit Betreuer – Besprechung des geplanten Algorithmus.
- Bis 21. Juni: Abgabe der Projektdateien.
- Anfang Juli: 10 minütige Präsentation.

Bewertung:

- Präsentation
- Implementierung
  - Alle Anforderungen erfüllt?
  - Wie stark lassen sich Regionen vereinfachen?

Testdaten gibt es auf der Homepage.