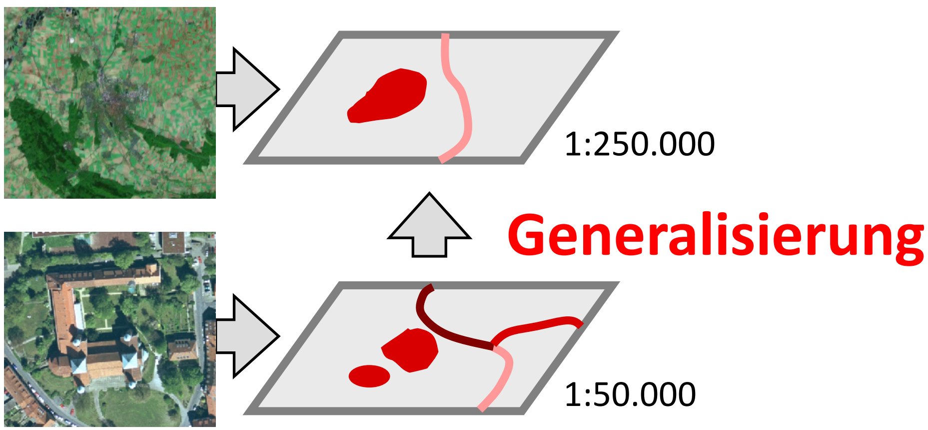
Algorithmen für Geographische Informationssysteme

Vereinfachung von Streckenzügen

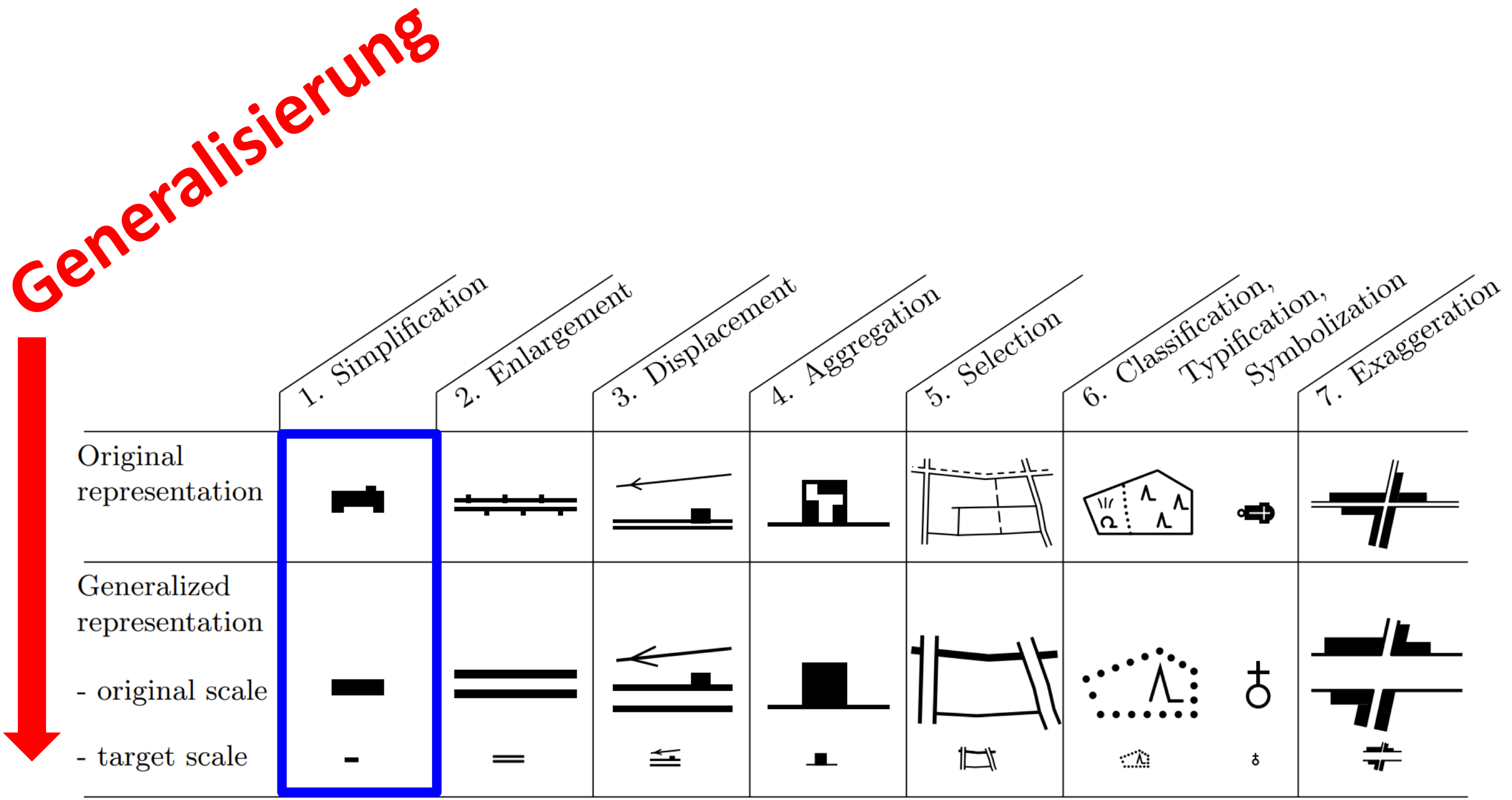
Alexander Wolff

4. Vorlesung

Kartografische Generalisierung



Kartografische Generalisierung



Hake et al. (2002)

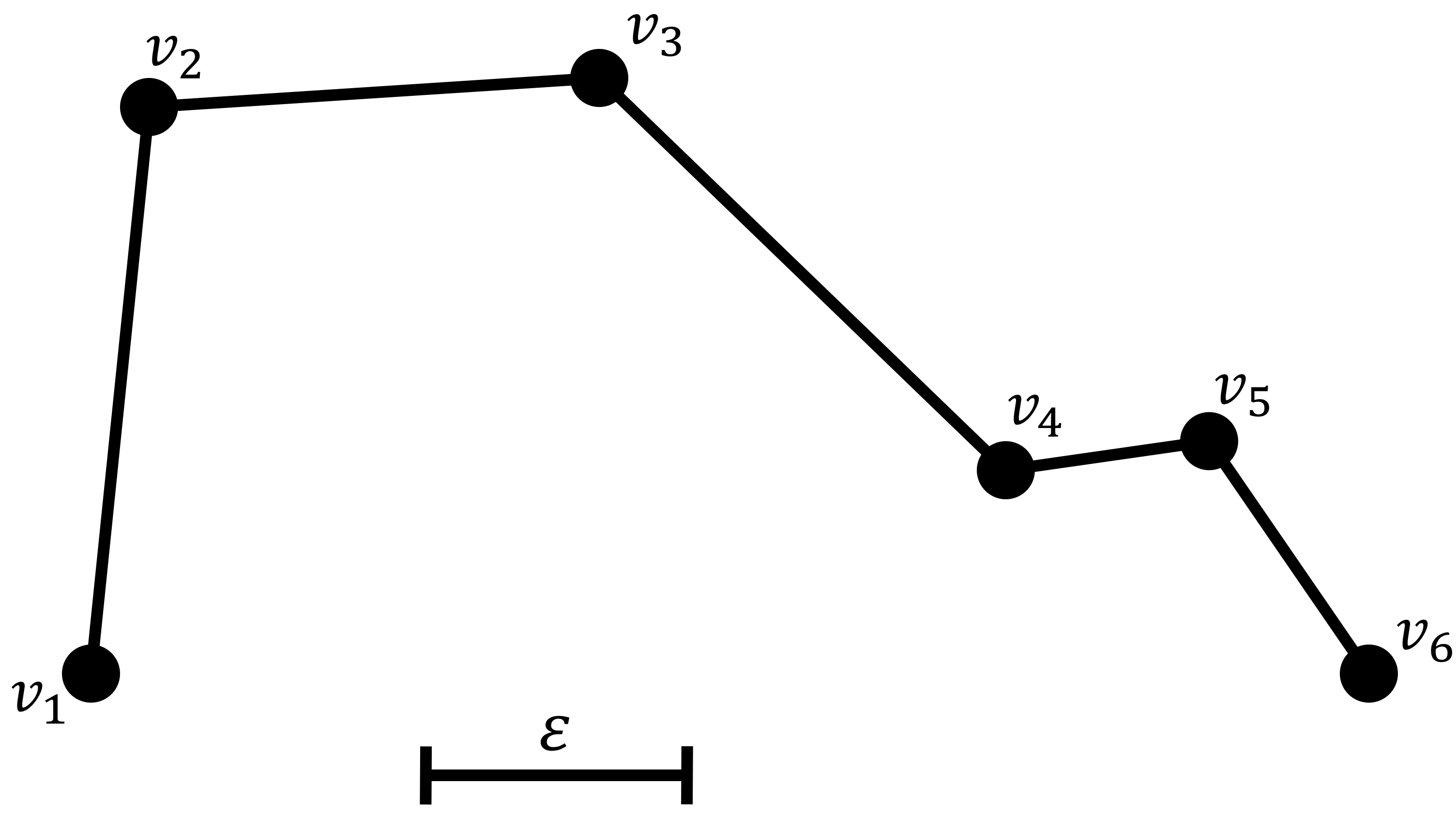
Gliederung

- **Linienvereinfachung**
  - mit dem Douglas-Peucker-Algorithmus
  - durch Optimierung
- **Vereinfachung von Gebäudegrundrissen**

Linienvereinfachung

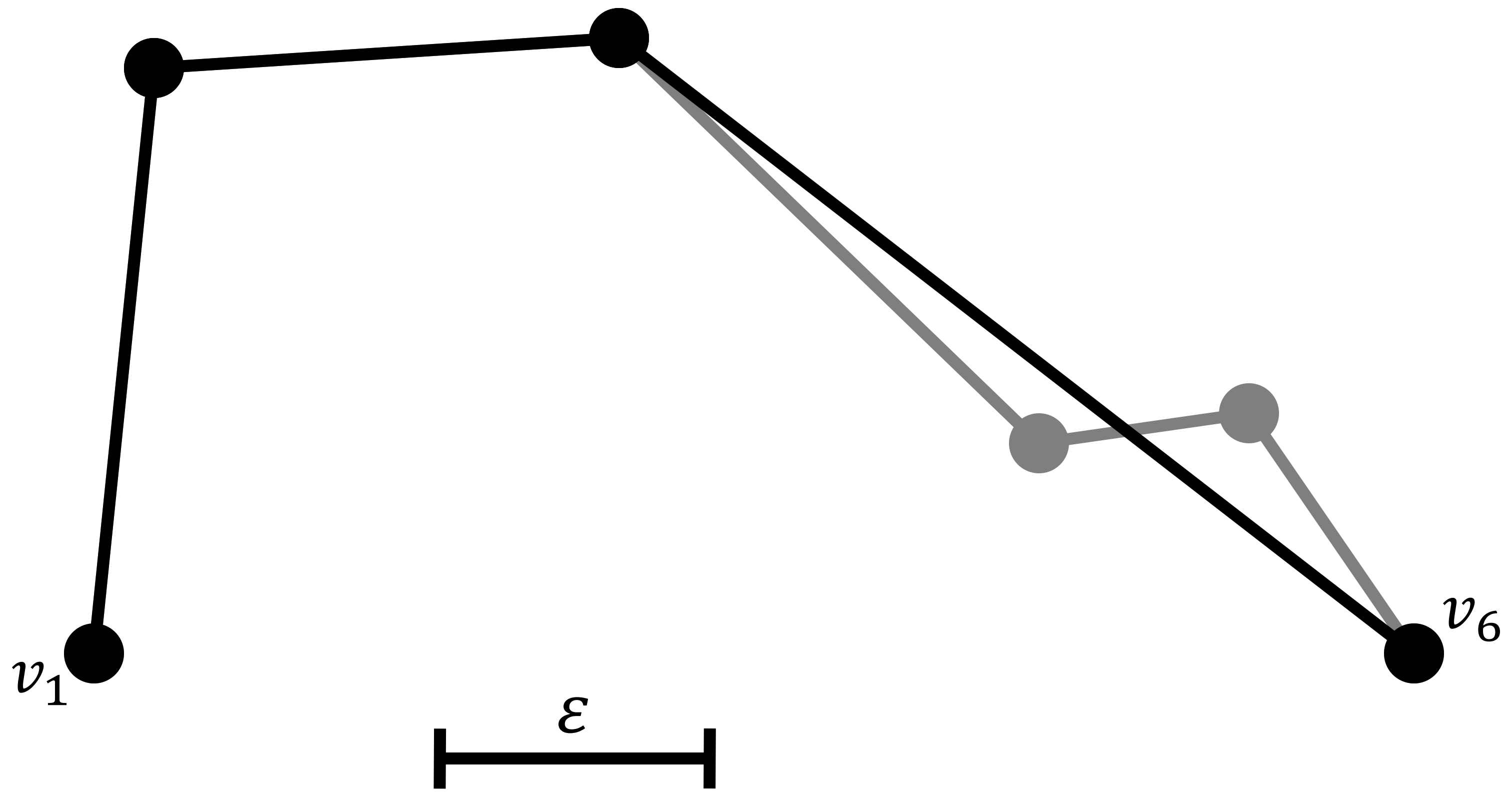
Linienvereinfachung
mit dem Douglas-Peucker-Algorithmus (1973)

Gegeben:
- eine kartographische Linie (eine Folge $V = (v_1, \ldots, v_n)$)
- eine geometrische Toleranz ε



Linienvereinfachung
mit dem Douglas-Peucker-Algorithmus (1973)

Ausgabe: • Teilfolge V' von V (von v_1 nach v_n)



Linienvereinfachung

mit dem Douglas-Peucker-Algorithmus (1973)

Method *douglasPeucker*(V', v_i, v_j)

$d_{\max} = -\infty,\ K = 0$

for $k = i + 1$ **to** $j - 1$

$\quad d\ =$ Abstand v_k zu Segment (v_i, v_j)

$\quad$ **if** $d > d_{\max}$ **then** $d_{\max} = d,\ K = k$

if $d_{\max} > \varepsilon$ **then**

$\quad$ add v_K to V'

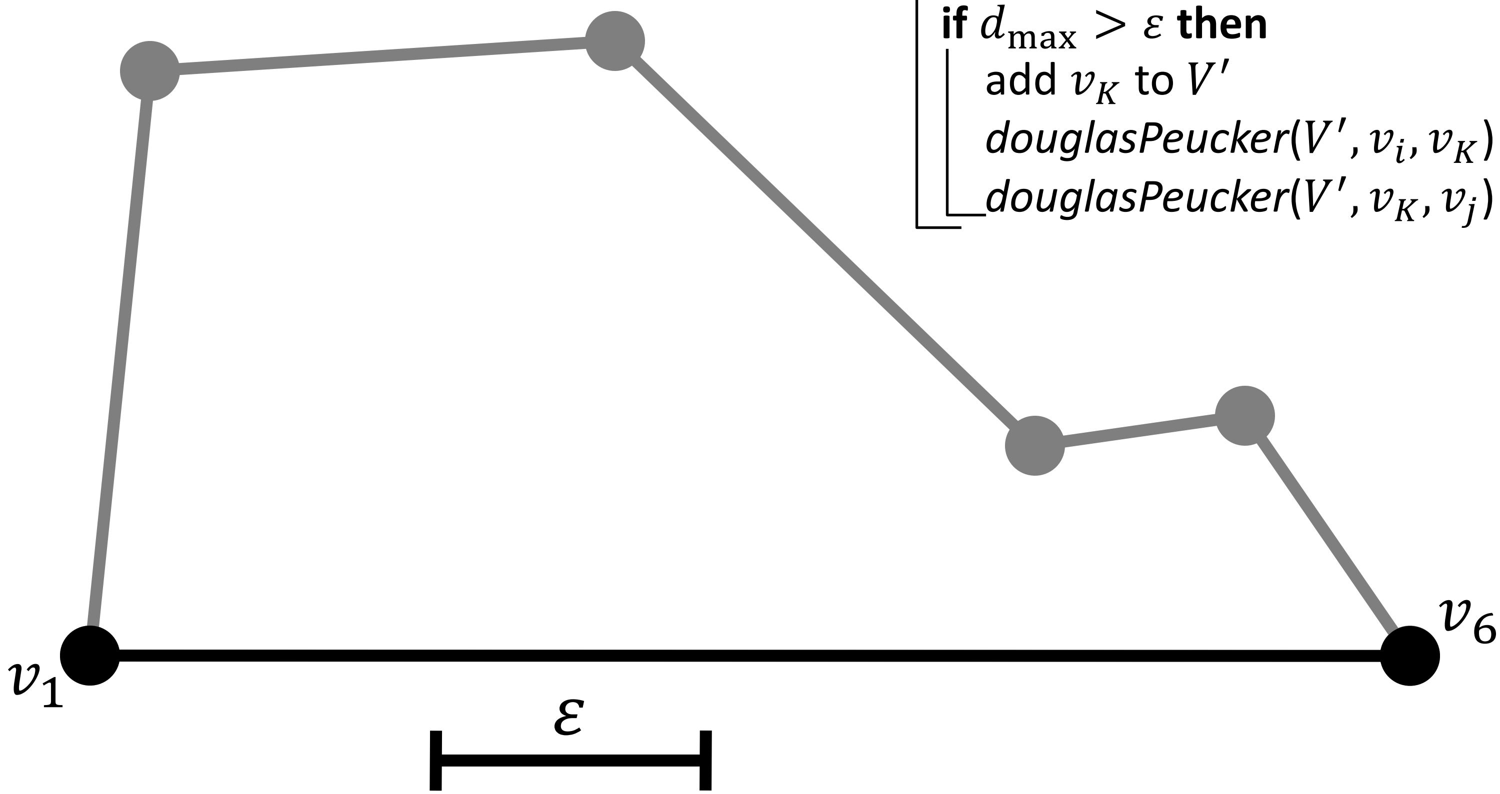
$\quad$ *douglasPeucker*(V', v_i, v_K)

$\quad$ *douglasPeucker*(V', v_K, v_j)

Algorithmus:

$V' = (v_1, v_n)$

douglasPeucker(V', v_1, v_n)



Linienvereinfachung

mit dem Douglas-Peucker-Algorithmus (1973)

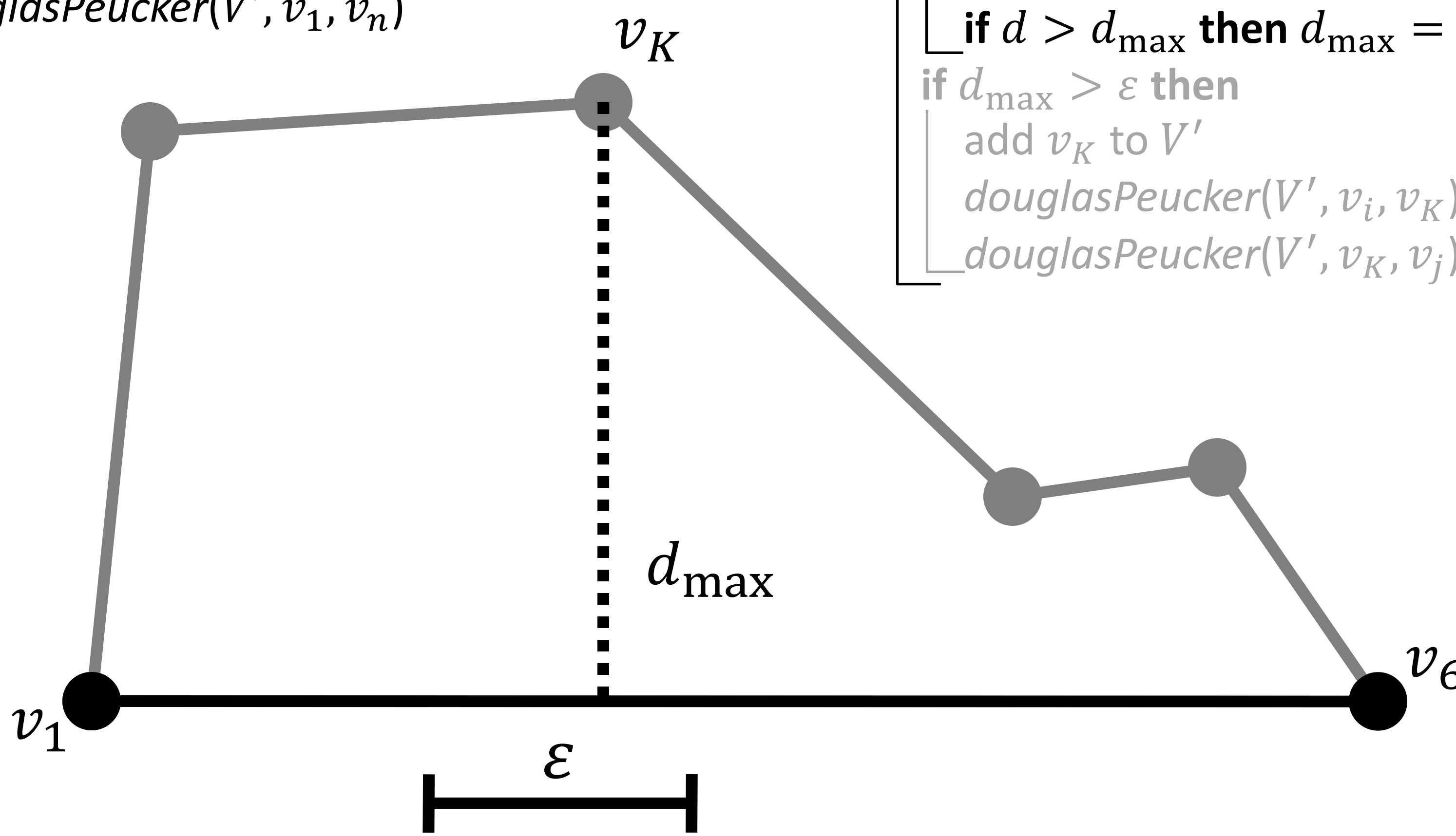
Method *douglasPeucker*(V', v_i, v_j)

- $d_{\max} = -\infty,\ K = 0$
- **for** $k = i+1$ **to** $j-1$
  - $d\ = $ Abstand v_k zu Segment (v_i, v_j)
  - **if** $d > d_{\max}$ **then** $d_{\max} = d,\ K = k$
- **if** $d_{\max} > \varepsilon$ **then**
  - add v_K to V'
  - *douglasPeucker*(V', v_i, v_K)
  - *douglasPeucker*(V', v_K, v_j)

Algorithmus:

$V' = (v_1, v_n)$

douglasPeucker(V', v_1, v_n)



Linienvereinfachung

mit dem Douglas-Peucker-Algorithmus (1973)

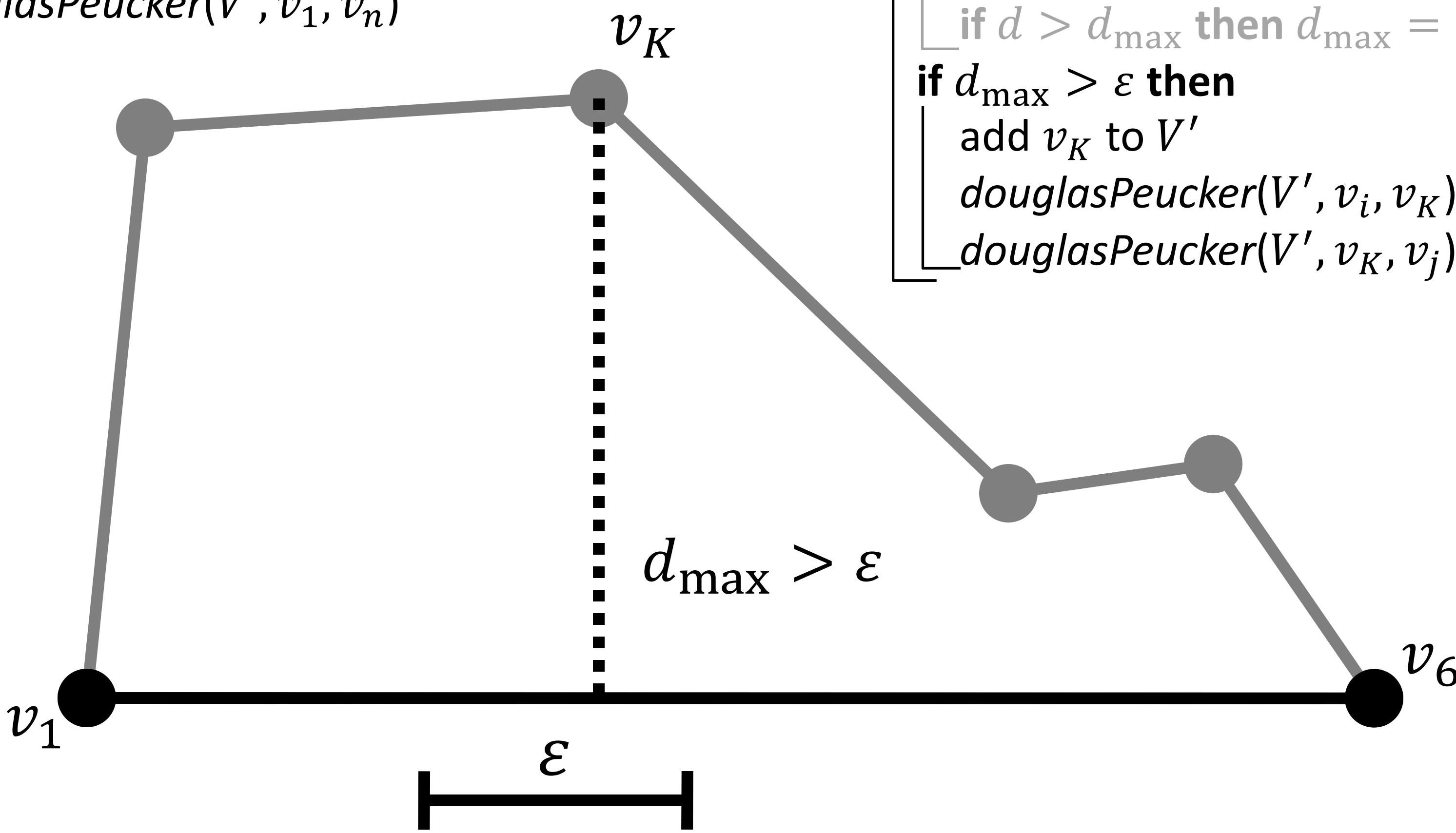
Method *douglasPeucker*(V', v_i, v_j)

- $d_{\max} = -\infty,\ K = 0$
- **for** $k = i + 1$ **to** $j - 1$
  - $d\ \ =$ Abstand v_k zu Segment (v_i, v_j)
  - **if** $d > d_{\max}$ **then** $d_{\max} = d,\ K = k$
- **if** $d_{\max} > \varepsilon$ **then**
  - add v_K to V'
  - *douglasPeucker*(V', v_i, v_K)
  - *douglasPeucker*(V', v_K, v_j)

Algorithmus:

$V' = (v_1, v_n)$

douglasPeucker(V', v_1, v_n)

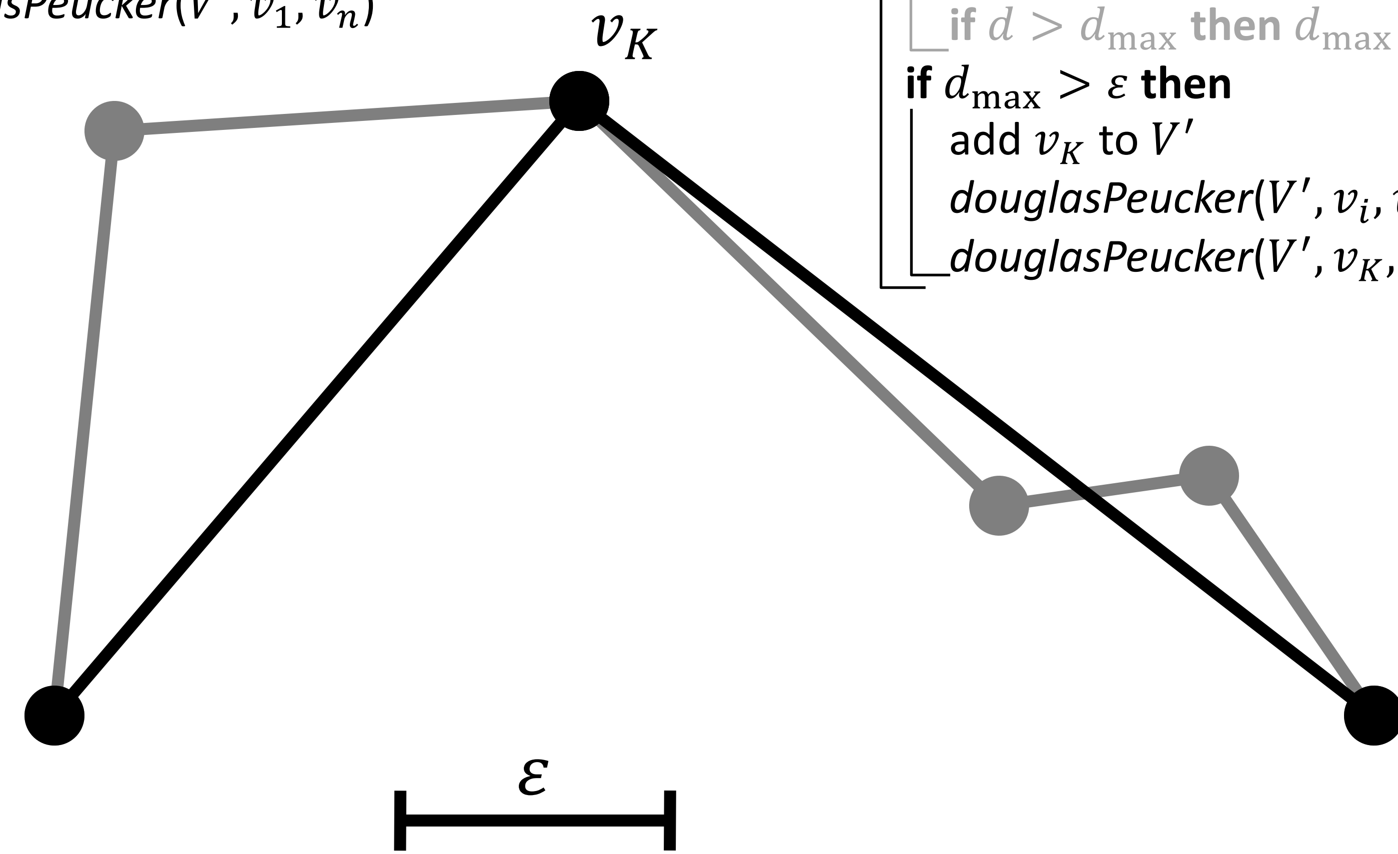


Linienvereinfachung
mit dem Douglas-Peucker-Algorithmus (1973)

Algorithmus:

$V' = (v_1, v_n)$
douglasPeucker(V', v_1, v_n)

Method *douglasPeucker*(V', v_i, v_j)
- $d_{\max} = -\infty,\ K = 0$
- **for** $k = i + 1$ **to** $j - 1$
  - $d\ = $ Abstand v_k zu Segment (v_i, v_j)
  - **if** $d > d_{\max}$ **then** $d_{\max} = d,\ K = k$
- **if** $d_{\max} > \varepsilon$ **then**
  - add v_K to V'
  - *douglasPeucker*(V', v_i, v_K)
  - *douglasPeucker*(V', v_K, v_j)



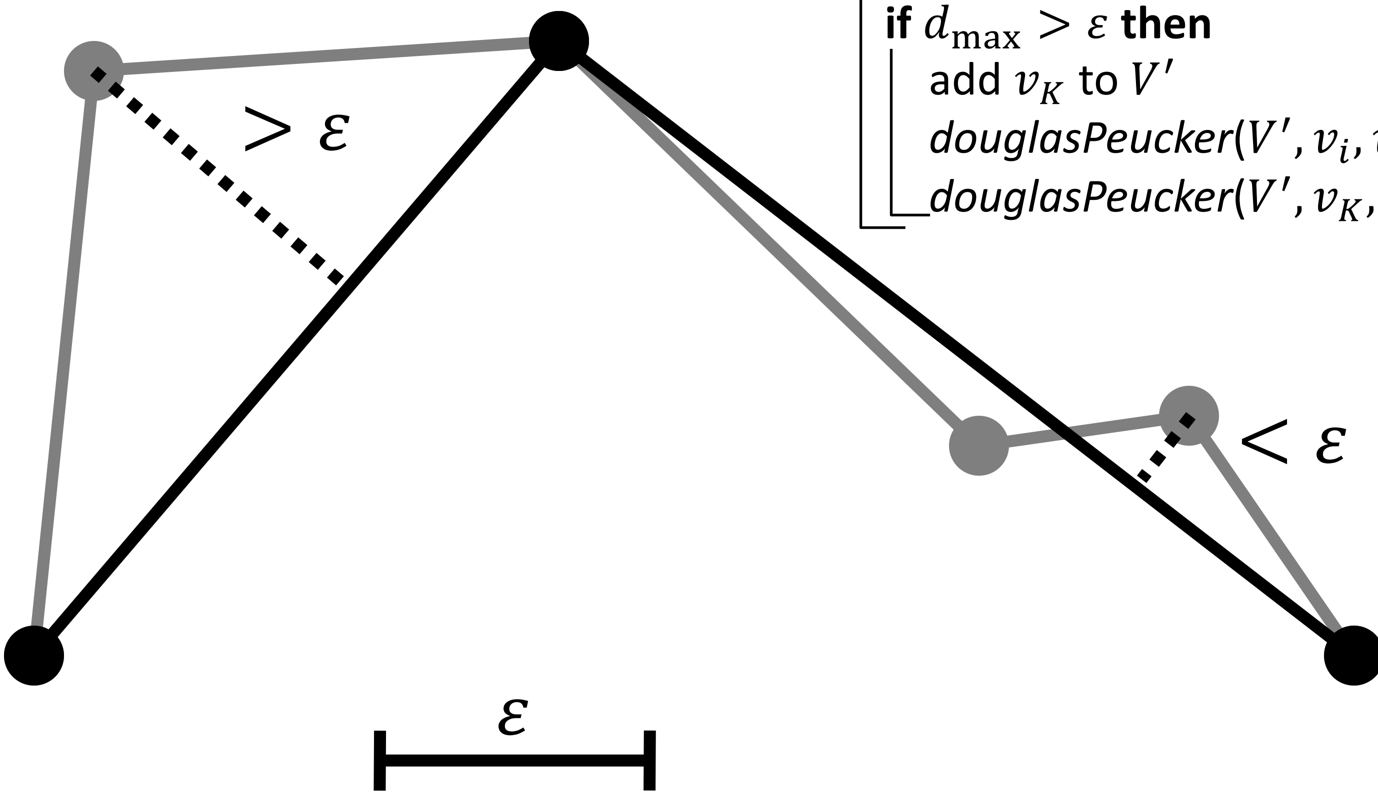
Linienvereinfachung
mit dem Douglas-Peucker-Algorithmus (1973)

Method *douglasPeucker*(V', v_i, v_j)
- $d_{\max} = -\infty$, $K = 0$
- **for** $k = i + 1$ **to** $j - 1$
  - d = Abstand v_k zu Segment (v_i, v_j)
  - **if** $d > d_{\max}$ **then** $d_{\max} = d$, $K = k$
- **if** $d_{\max} > \varepsilon$ **then**
  - add v_K to V'
  - *douglasPeucker*(V', v_i, v_K)
  - *douglasPeucker*(V', v_K, v_j)

Algorithmus:

$V' = (v_1, v_n)$

douglasPeucker(V', v_1, v_n)



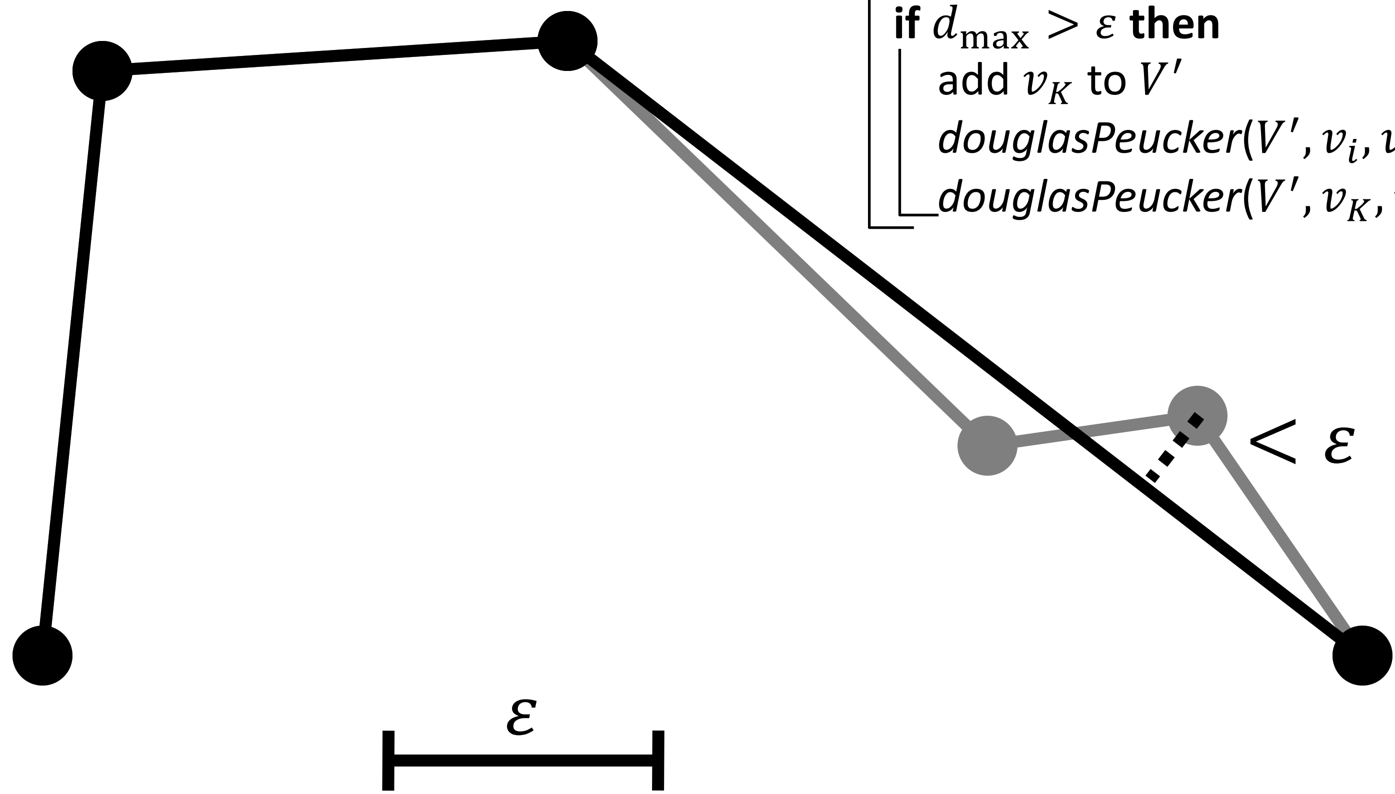
Linienvereinfachung

mit dem Douglas-Peucker-Algorithmus (1973)

Algorithmus:

$V' = (v_1, v_n)$
douglasPeucker(V', v_1, v_n)

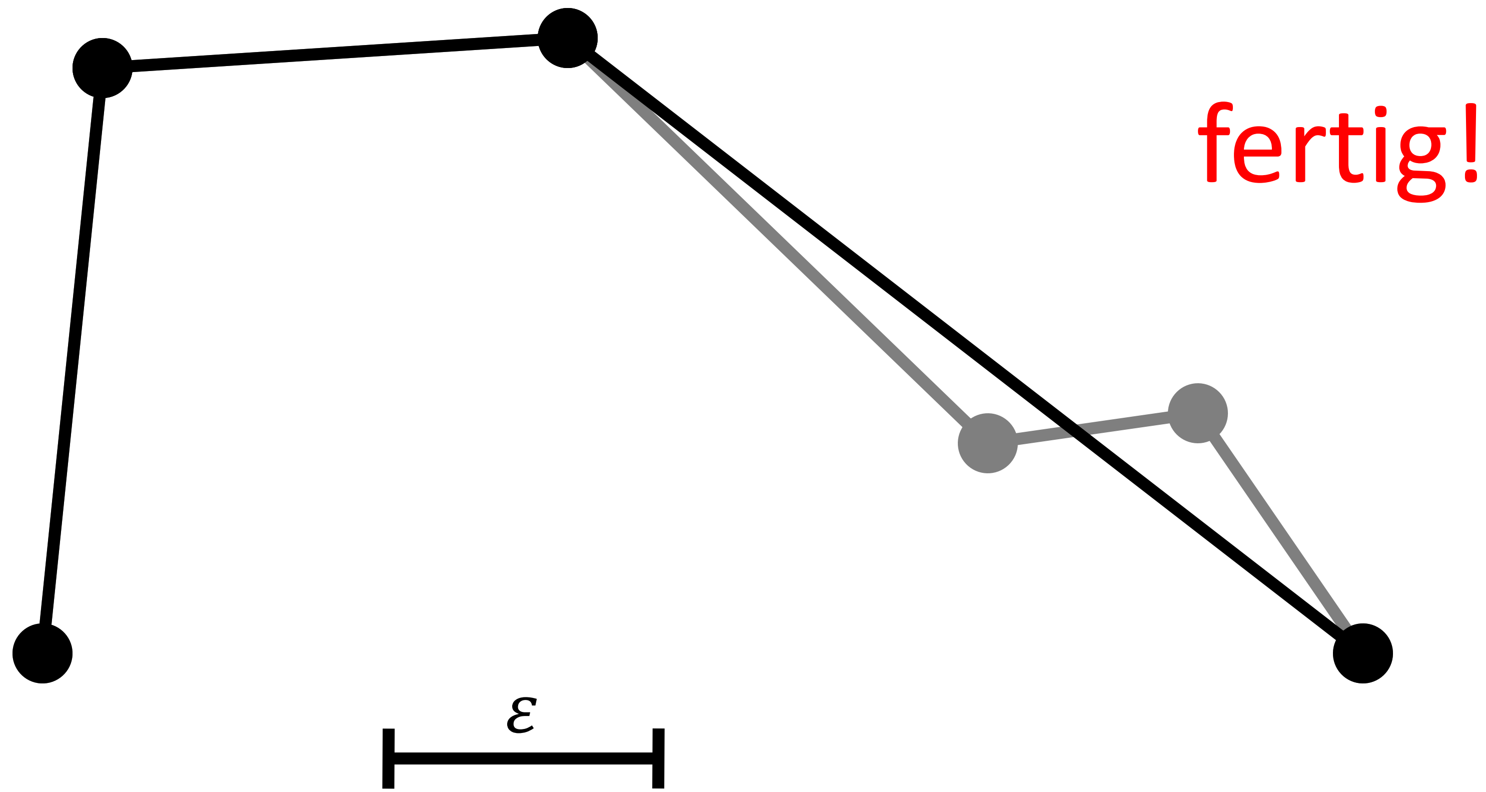
Method *douglasPeucker*(V', v_i, v_j)
$d_{\max} = -\infty$, $K = 0$
for $k = i + 1$ **to** $j - 1$
d = Abstand v_k zu Segment (v_i, v_j)
if $d > d_{\max}$ **then** $d_{\max} = d$, $K = k$
if $d_{\max} > \varepsilon$ **then**
add v_K to V'
douglasPeucker(V', v_i, v_K)
douglasPeucker(V', v_K, v_j)



Linienvereinfachung

mit dem Douglas-Peucker-Algorithmus (1973)

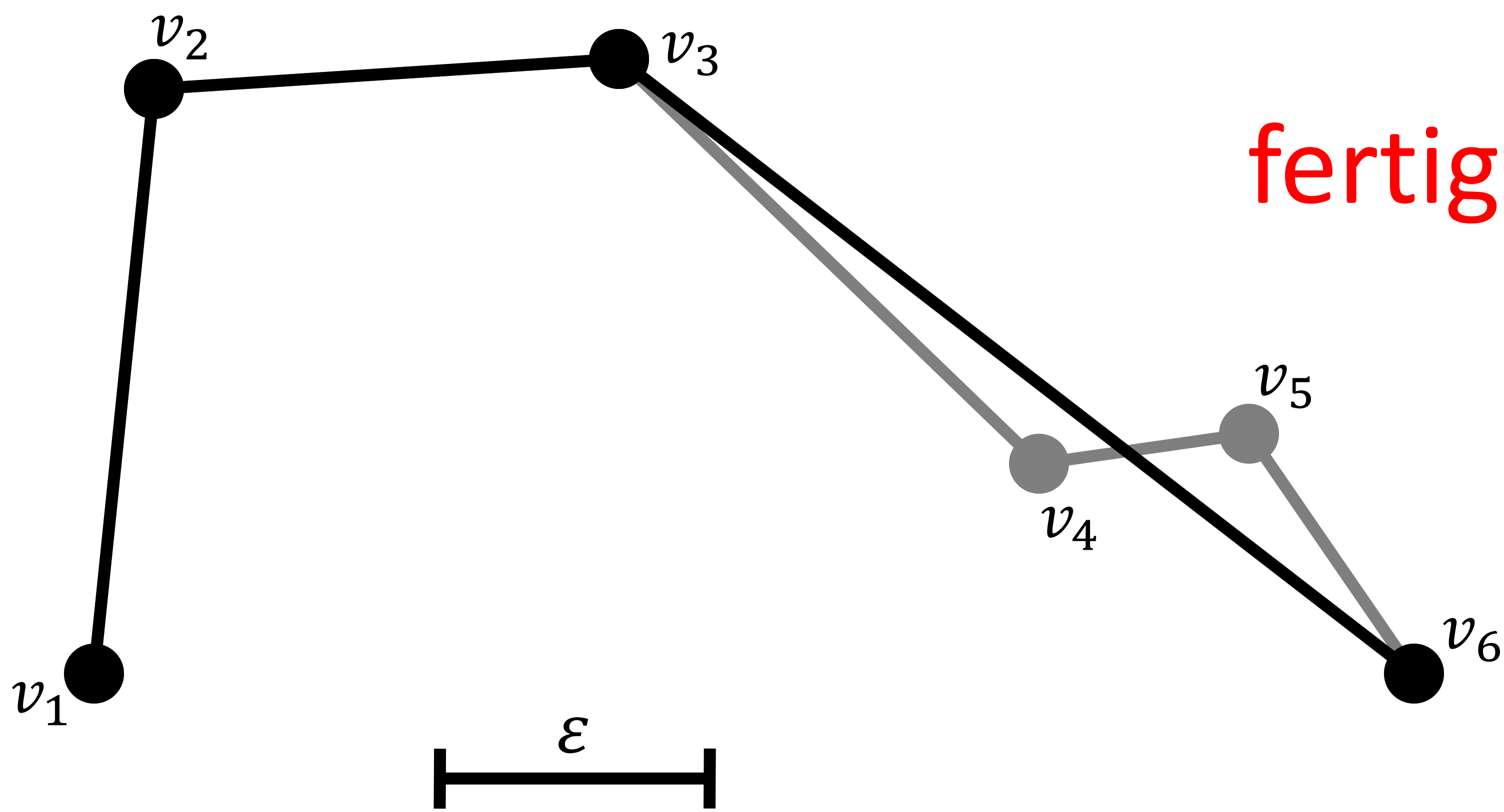
Ausgabe: • Teilfolge V' von V (von v_1 nach v_n)



Linienvereinfachung

mit dem Douglas-Peucker-Algorithmus (1973)

Ausgabe:
- Teilfolge V' von V (von v_1 nach v_n)
- für jedes Liniensegment $S = (v_i, v_j)$ in V' und jede Ecke v_k mit $i < k < j$ gilt, dass $d(S, v_k) \leq \varepsilon$.



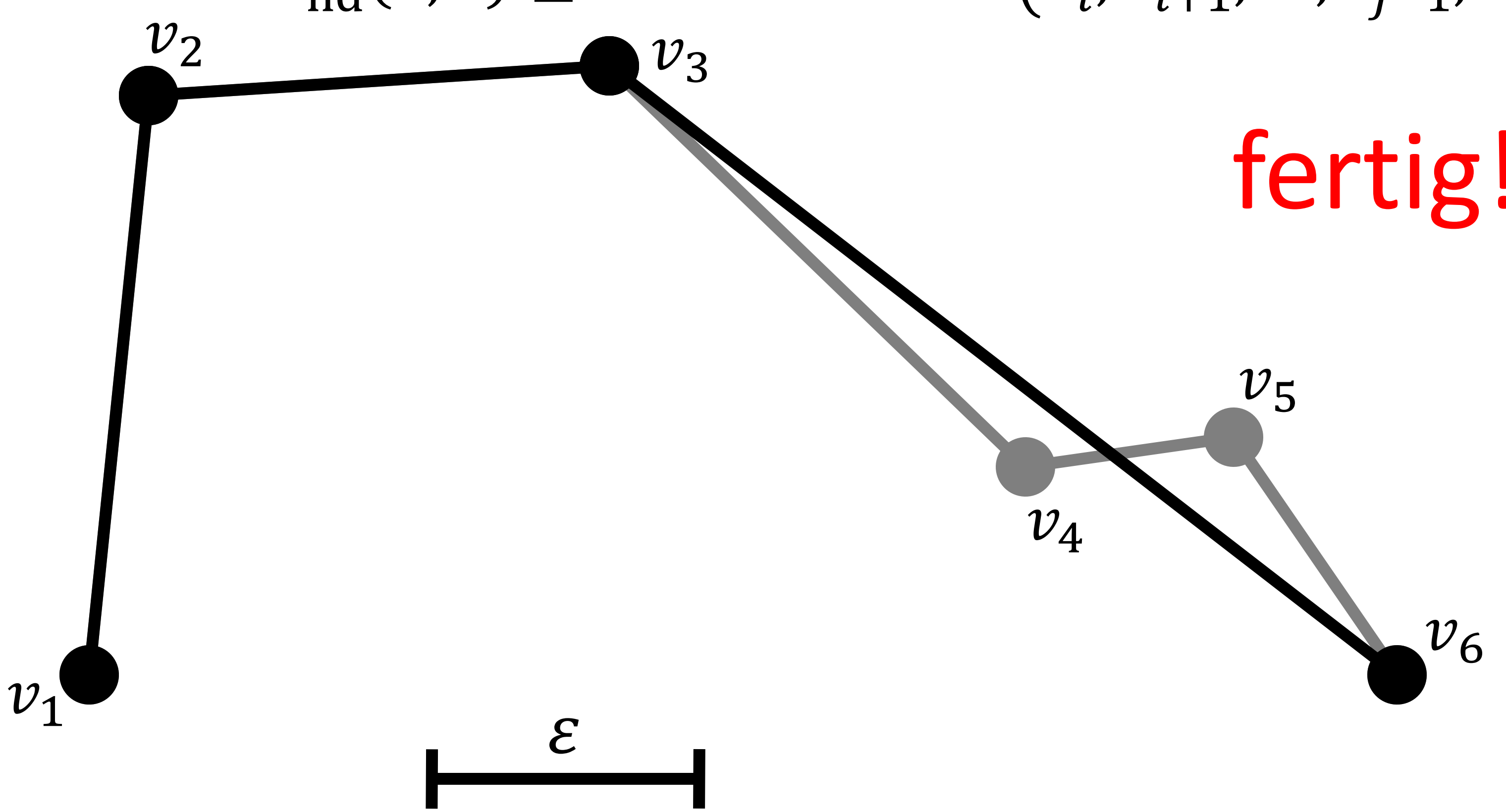
fertig!

Linienvereinfachung

mit dem Douglas-Peucker-Algorithmus (1973)

Ausgabe:
- Teilfolge V' von V (von v_1 nach v_n)
- für jedes Liniensegment $S = (v_i, v_j)$ in V'

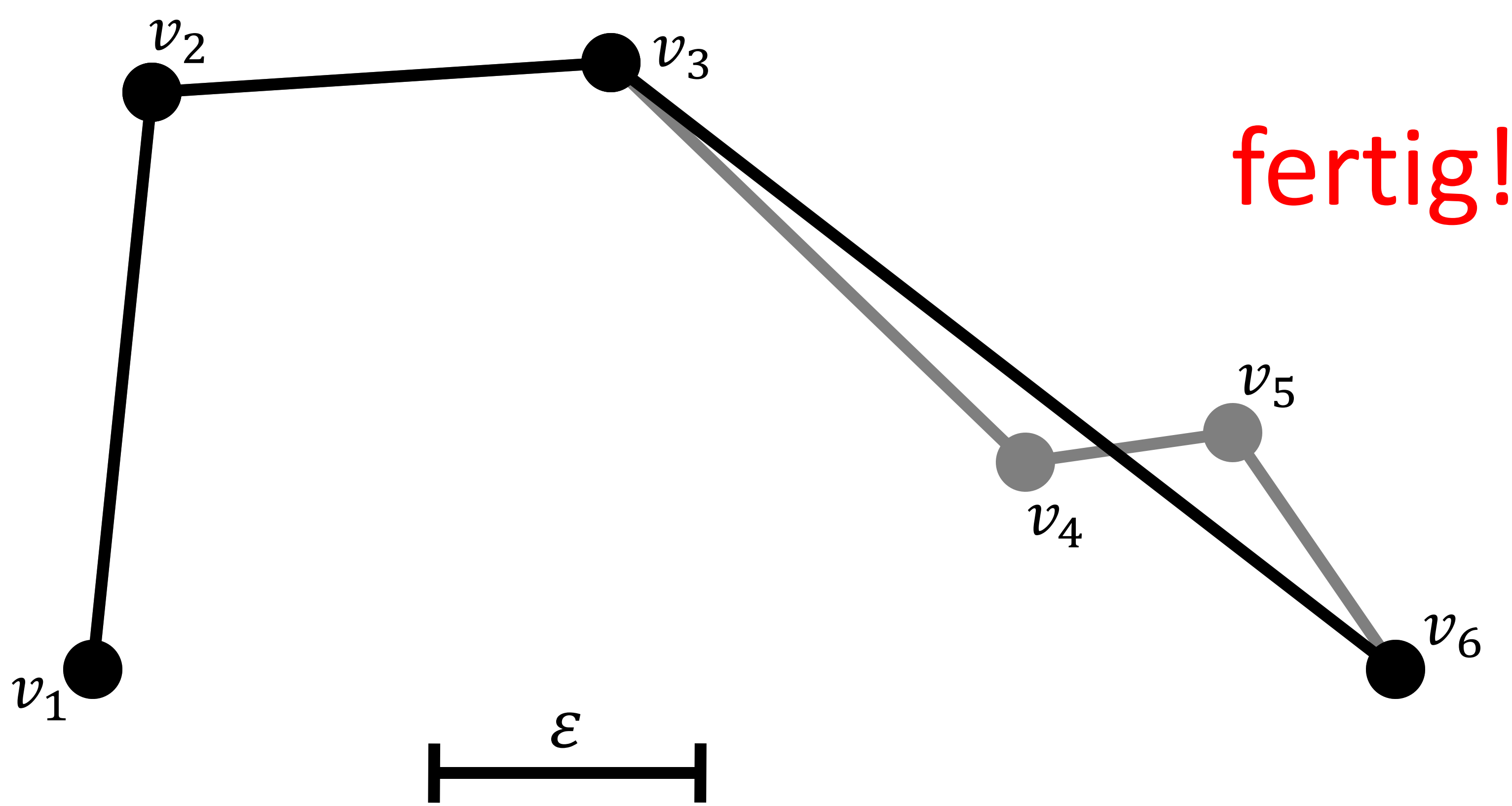
  $\vec{d}_{\text{hd}}(S, L) \leq \varepsilon$ mit $L = (v_i, v_{i+1}, \ldots, v_{j-1}, v_j)$.



Linienvereinfachung

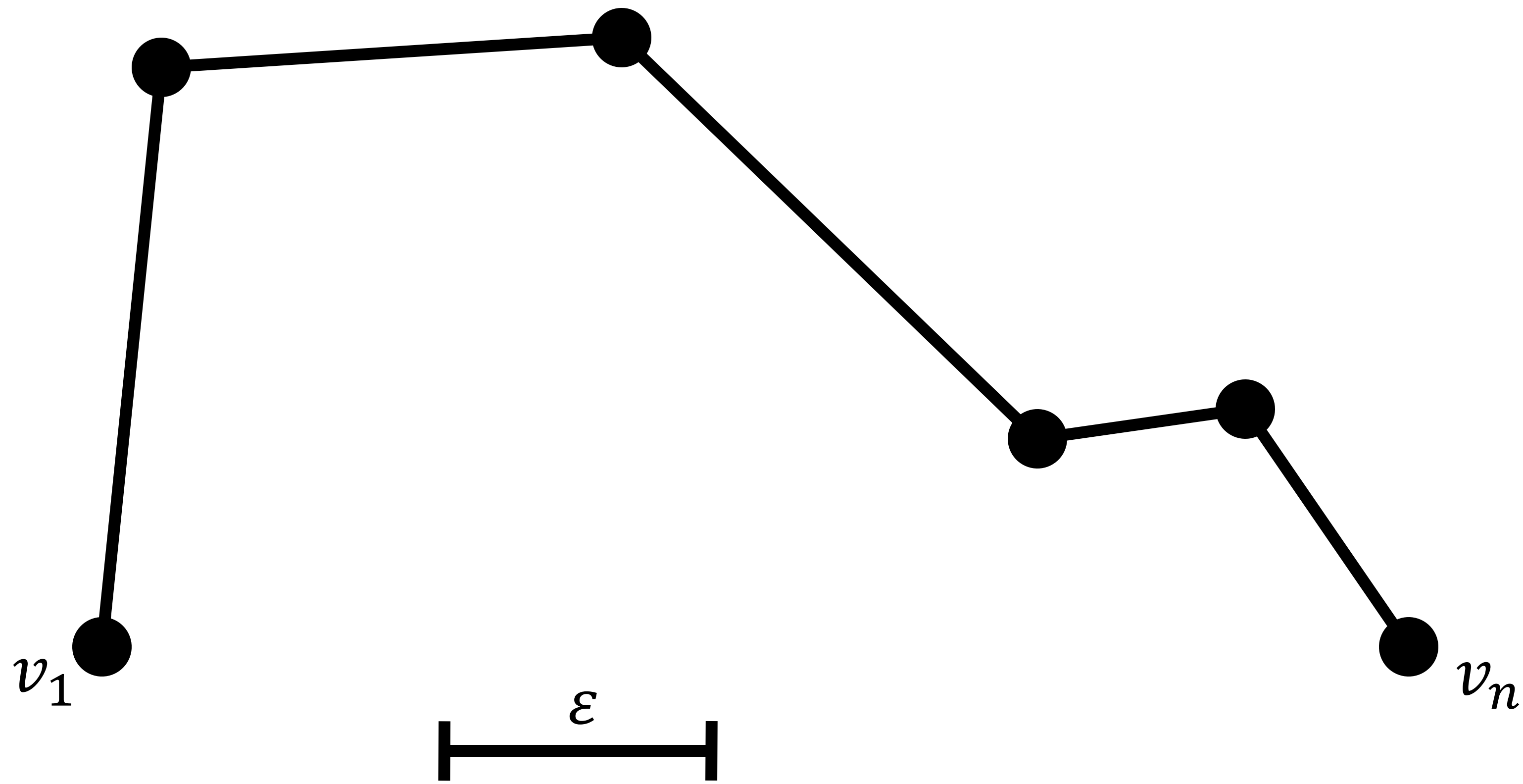
mit dem Douglas-Peucker-Algorithmus (1973)

Ausgabe:
- Teilfolge V' von V (von v_1 nach v_n)
- $\vec{d}_{\mathrm{hd}}(V', V) \leq \varepsilon$



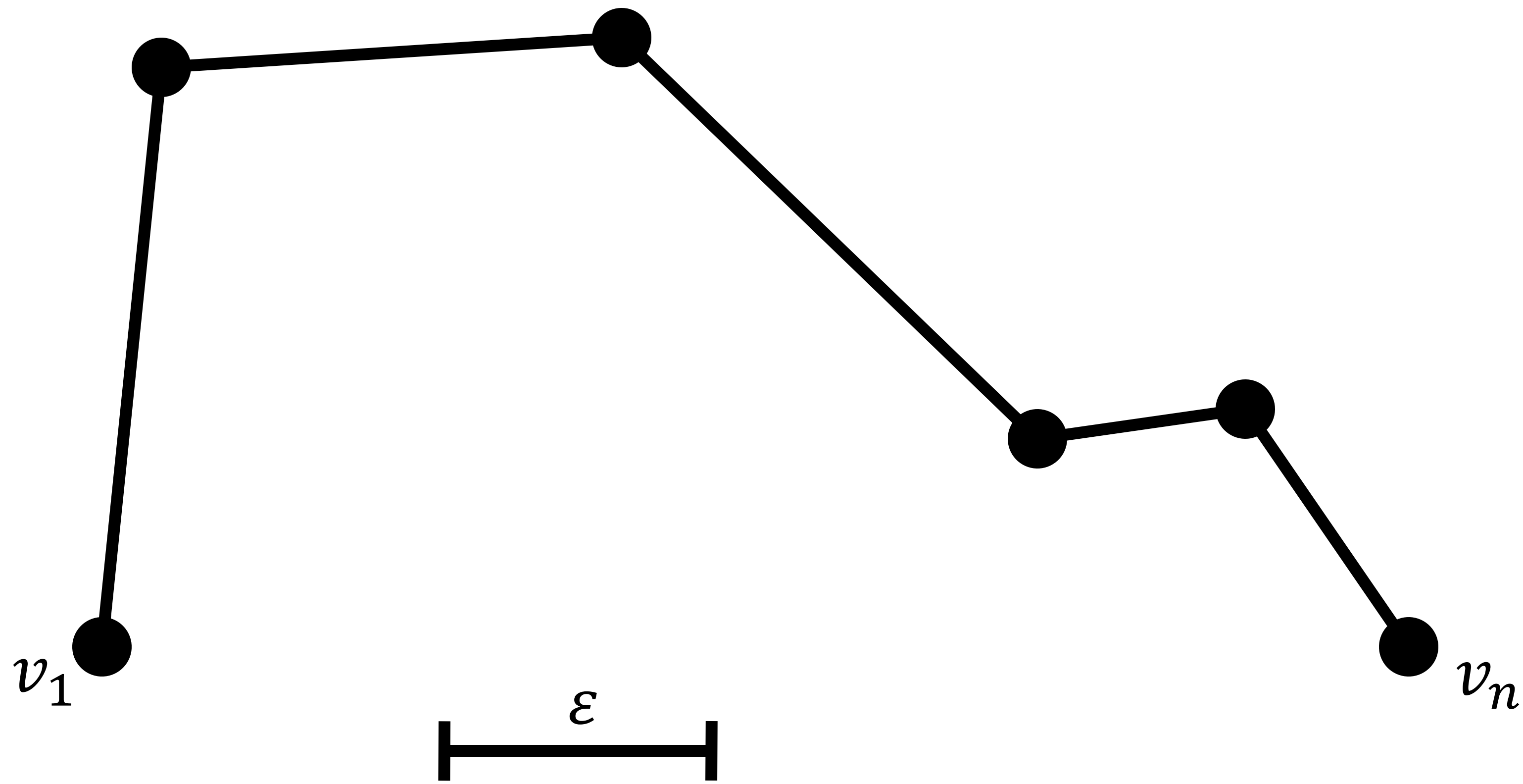
Linienvereinfachung
durch Optimierung (Imai & Iri, 1988)

Gegeben:
- eine kartographische Linie (eine Folge $V = (v_1, \dots, v_n)$)
- eine geometrische Toleranz ε



Linienvereinfachung
durch Optimierung (Imai & Iri, 1988)

Ausgabe:
- ***kleinste*** Teilfolge V' von V (von v_1 nach v_n), so dass
- für jedes Liniensegment $S = \left(v_i, v_j\right)$ in V'
  $\vec{d}_{\text{hd}}(S, L) \leq \varepsilon$ mit $L = \left(v_i, v_{i+1}, \ldots, v_{j-1}, v_j\right)$.

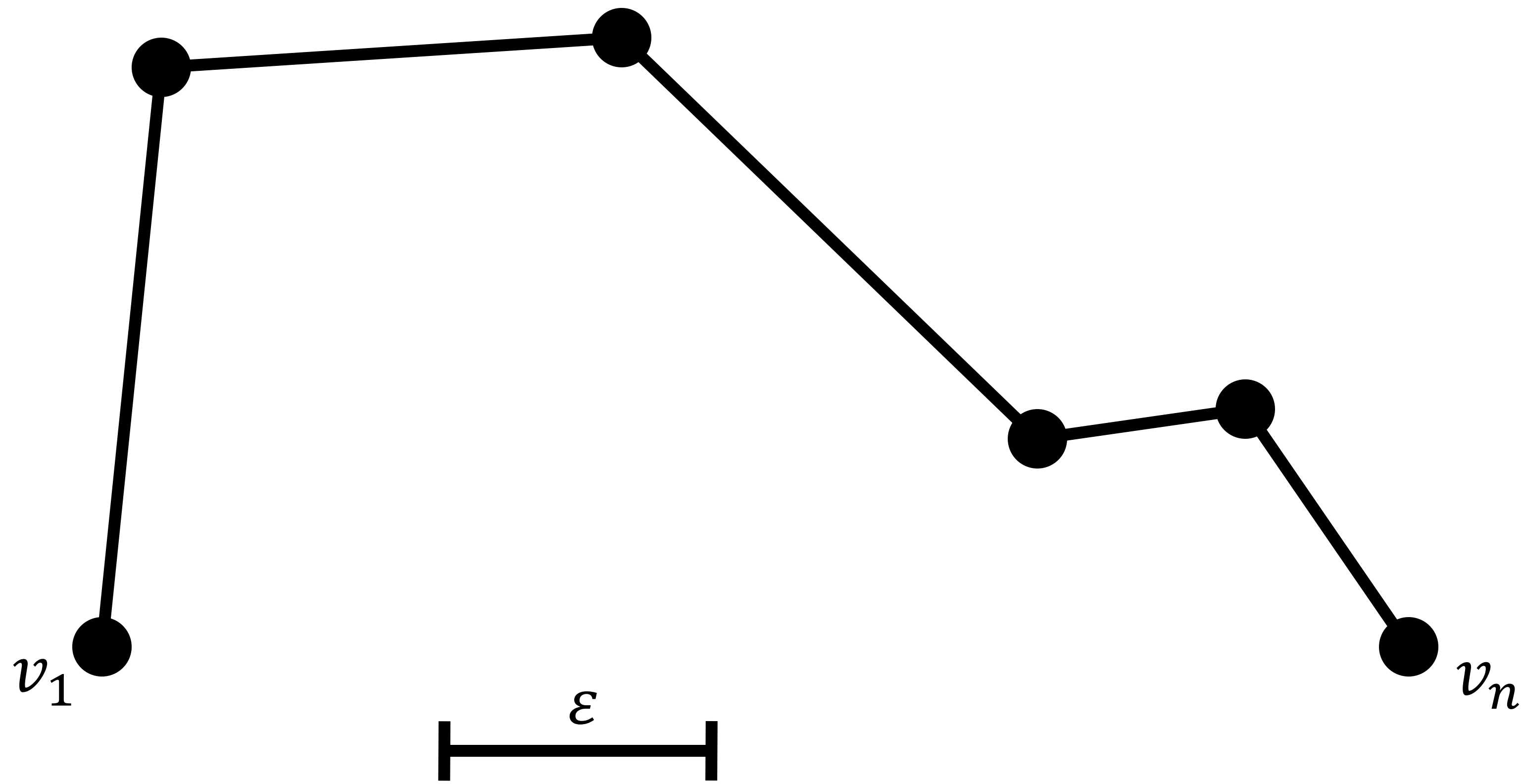


Linienvereinfachung

durch Optimierung (Imai & Iri, 1988)

Ausgabe:

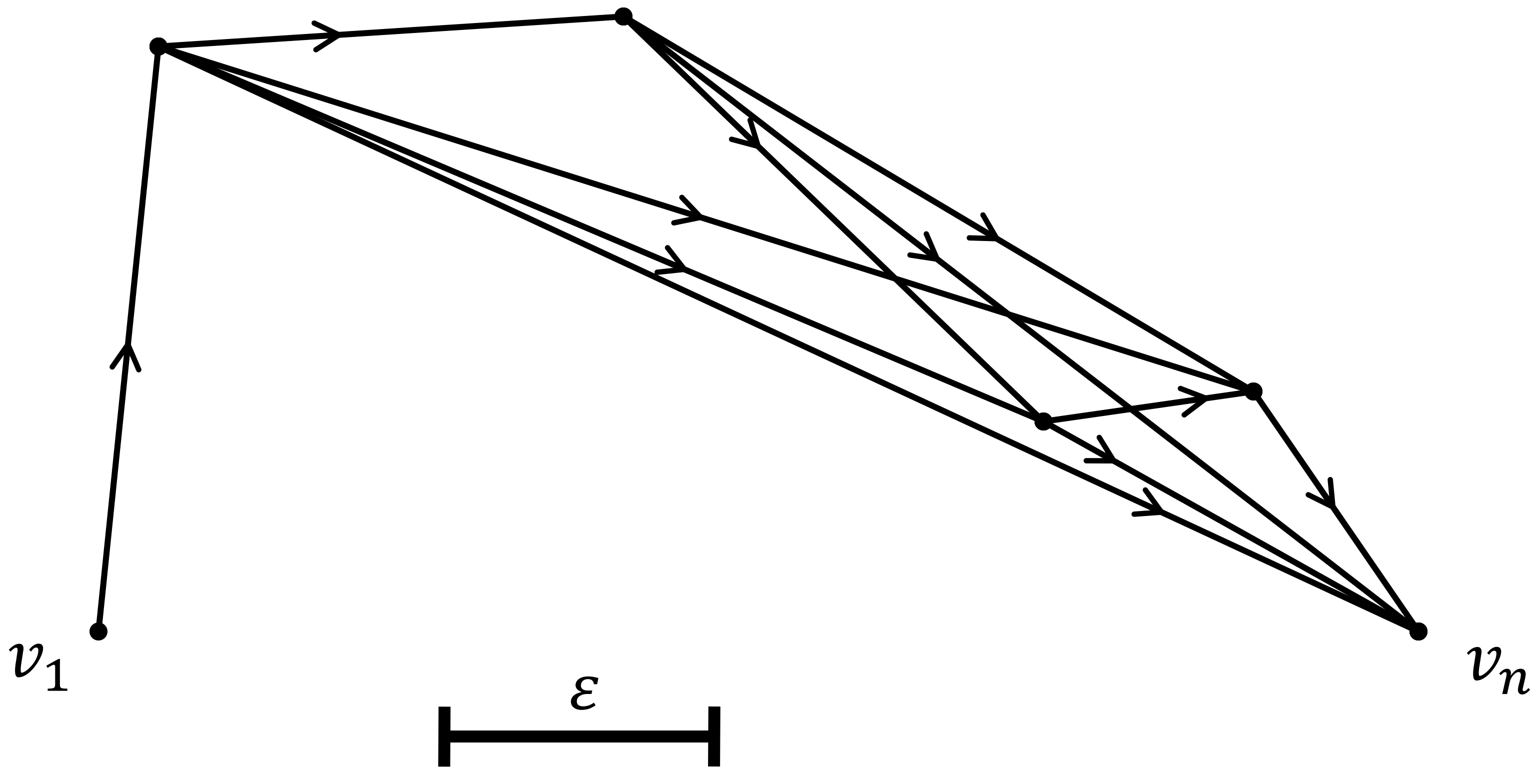
- ***kleinste*** Teilfolge V' von V (von v_1 nach v_n), so dass
- für jedes Liniensegment $S = (v_i, v_j)$ in V' und jede Ecke v_k mit $i < k < j$ gilt, dass $d(S, v_k) \leq \varepsilon$.



Linienvereinfachung
durch Optimierung (Imai & Iri, 1988)

Lösung:

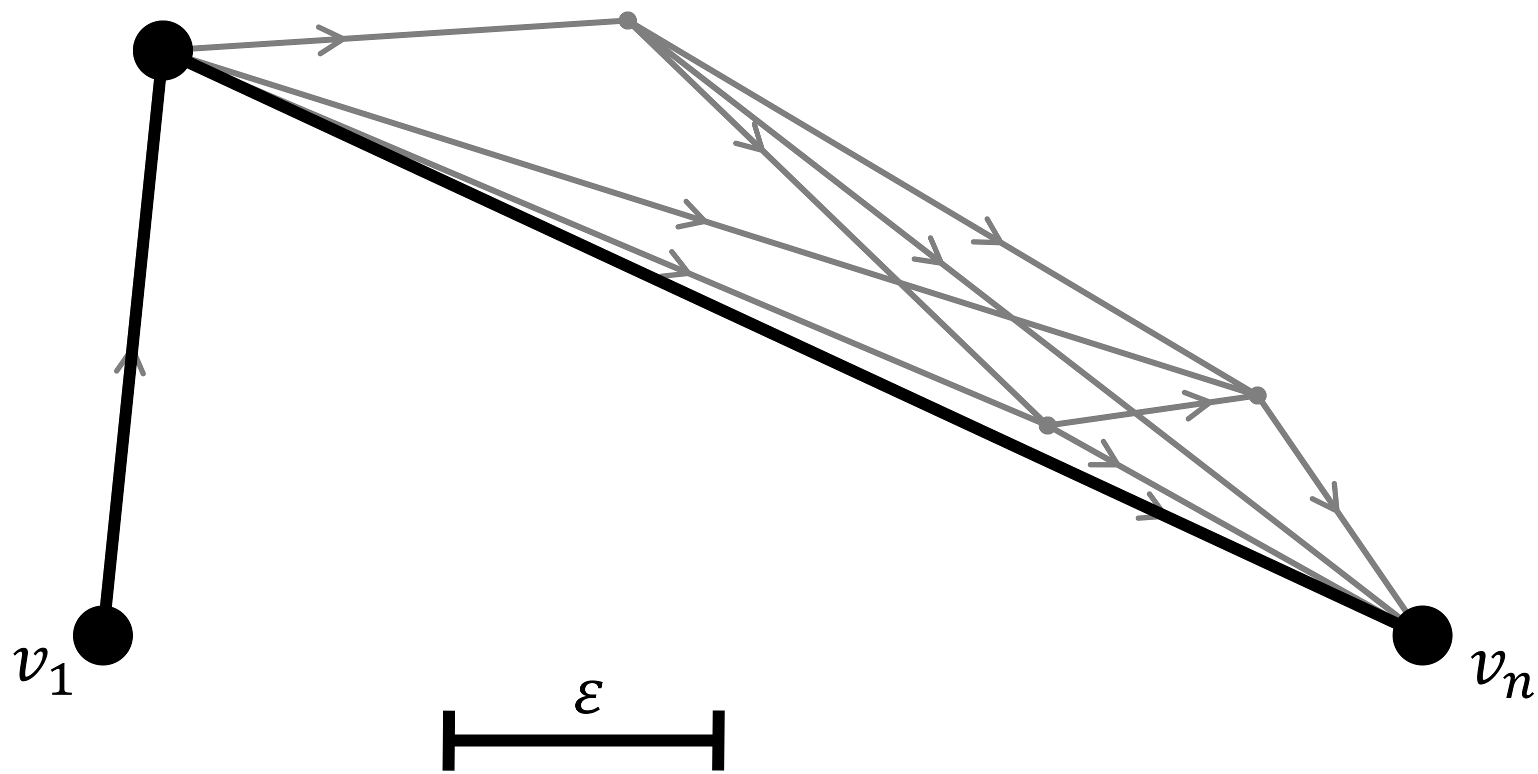
- definiere Graph $G = (V, A)$ mit zulässigen Abkürzungen A (shortcuts)



Linienvereinfachung
durch Optimierung (Imai & Iri, 1988)

Lösung:

- definiere Graph $G = (V, A)$ mit zulässigen Abkürzungen A (shortcuts)
- finde kürzesten Weg in G von v_1 nach v_n



Linienvereinfachung
durch Optimierung (Imai & Iri, 1988)

Aufgabe:
Berechne Abkürzungsgraph G

Lösung (einfacher Algorithmus):

1. Für jede Ecke $v \in \{v_1, \dots, v_n\}$ definiere v als Knoten von G.
2. Für jedes Paar $v_i, v_j \in \{v_1, \dots, v_n\}$ mit $i < j$
   teste für $k = i + 1, \dots, j - 1$, ob $\operatorname{dist}\left(v_k, \overline{v_i v_j}\right) \leq \varepsilon$
   falls ja für alle k: definiere $\left(v_i, v_j\right)$ als Kante von G.

Laufzeit? $O(n^3)$

Linienvereinfachung

durch Optimierung (Imai & Iri, 1988)

$m = O(n^2)$

Aufgabe:
Finde kürzesten Weg in gerichtetem azyklischen Graphen mit n Ecken und m Kanten.

Lösung:

1. Bringe die Ecken in topologische Ordnung. Hier durch $v_1 \dots v_n$ bereits gegeben.
2. Löse das Problem durch dynamische Programmierung:

$d(v_1) = 0$
for $j = 2$ **to** n // Berechne $d(v_j)$ = Länge eines kürzesten v_1-v_j-Pfades
  $d(v_j) = \infty$
  foreach $v_i : (v_i, v_j) \in A$
    if $d(v_i) + \ell(v_i, v_j) < d(v_j)$ **then** // $\ell(v_i, v_j)$ = Länge der Kante $(v_i, v_j) = 1$
      $d(v_j) = d(v_i) + \ell(v_i, v_j)$
      $predecessor(j) = i$

Laufzeit: $O(m + n)$
$= O(n^2)$

Linienvereinfachung
durch Optimierung (Imai & Iri, 1988)

Lösung:

- definiere Graph $G = (V, A)$ mit zulässigen Abkürzungen A (shortcuts)

Laufzeit:

- finde kürzesten Weg in G von v_1 nach v_n

Laufzeit:

Gesamtlaufzeit:

Linienvereinfachung
durch Optimierung (Imai & Iri, 1988)

Lösung:

- definiere Graph $G = (V, A)$ mit zulässigen Abkürzungen A (shortcuts)

Laufzeit: $O(n^3)$

- finde kürzesten Weg in G von v_1 nach v_n

Laufzeit: $O(n^2)$

Gesamtlaufzeit: $O(n^3)$

Linienvereinfachung

durch Optimierung (Imai & Iri, 1988)

Aufgabe:
Berechne Abkürzungsgraph G

Laufzeit: $O(n^2)$

Lösung (besserer Algorithmus):

Chan & Chin, 1996:
Approximation of polygonal curves with minimum number of line segments or min. error,
Int. Journal of Computational Geometry and Applications.

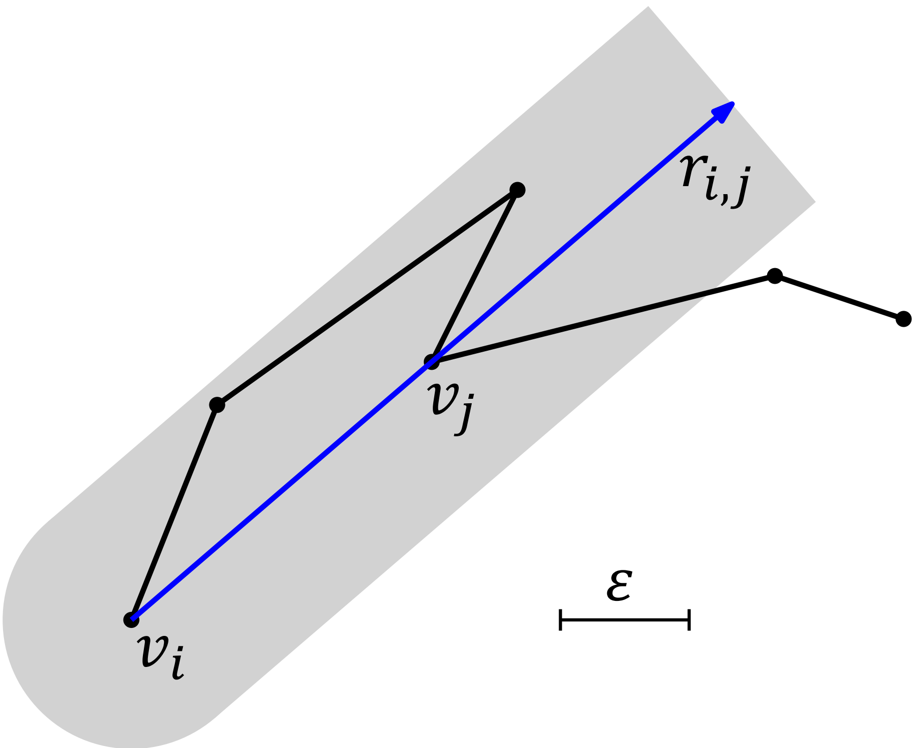
Gesamtlaufzeit: $O(n^2)$

Linienvereinfachung
durch Optimierung (Chan & Chin, 1996)

Aufgabe:
Berechne Abkürzungsgraph G

Lösung (besserer Algorithmus):

Berechne Abkürzungsgraph G',
der Kante (v_i, v_j) enthält,
wenn für jedes $i < k < j$ gilt,
dass $d(v_k, r_{i,j}) \leq \varepsilon$



Definition:
$r_{i,j}$ = Strahl von v_i durch v_j

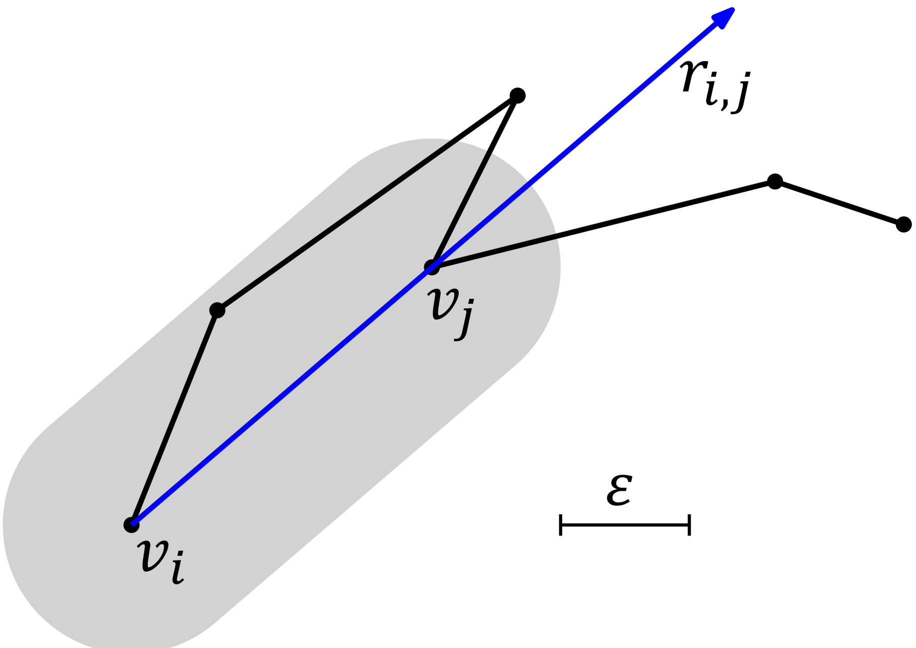
Linienvereinfachung
durch Optimierung (Chan & Chin, 1996)

Aufgabe:
Berechne Abkürzungsgraph G

Lösung (besserer Algorithmus):

Berechne Abkürzungsgraph G', der Kante (v_i, v_j) enthält, wenn für jedes $i < k < j$ gilt, dass $d(v_k, r_{i,j}) \leq \varepsilon$

Achtung: $G' \neq \mathrm{G}$



Definition:
$r_{i,j} =$ Strahl von v_i durch v_j

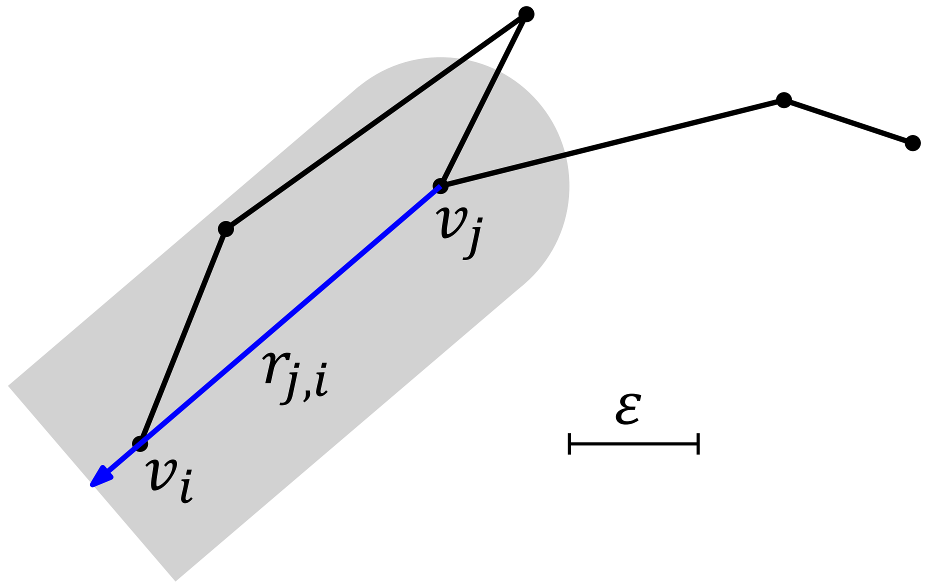
Linienvereinfachung
durch Optimierung (Chan & Chin, 1996)

Aufgabe:
Berechne Abkürzungsgraph G

Lösung (besserer Algorithmus):

Berechne Abkürzungsgraph G',
der Kante (v_i, v_j) enthält,
wenn für jedes $i < k < j$ gilt,
dass $d(v_k, r_{i,j}) \leq \varepsilon$

Berechne Abkürzungsgraph G'',
der Kante (v_i, v_j) enthält,
wenn für jedes $i < k < j$ gilt,
dass $d(v_k, r_{j,i}) \leq \varepsilon$



Definition:
$r_{i,j}$ = Strahl von v_i durch v_j

Linienvereinfachung
durch Optimierung (Chan & Chin, 1996)

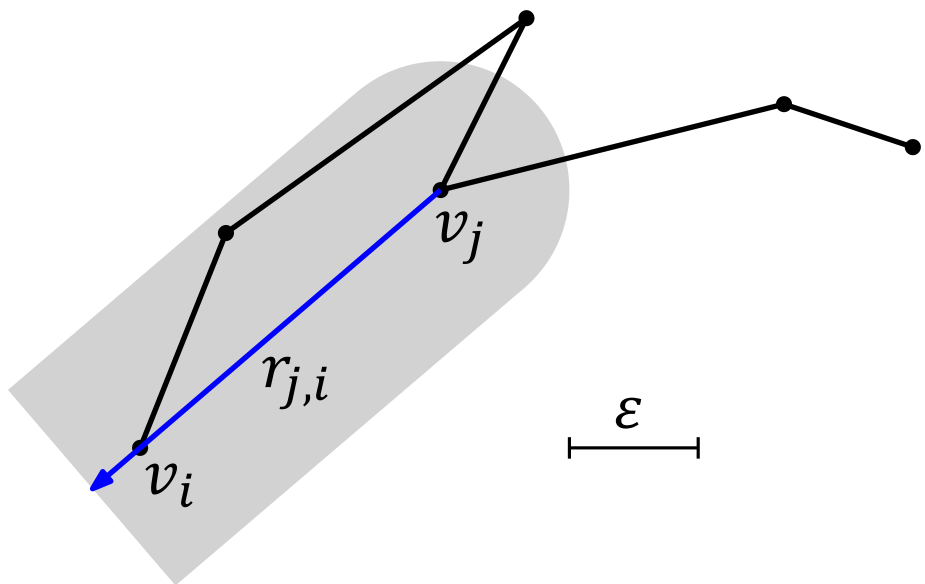
Aufgabe:
Berechne Abkürzungsgraph G

Lösung (besserer Algorithmus):

Berechne Abkürzungsgraph G', der Kante (v_i, v_j) enthält, wenn für jedes $i < k < j$ gilt, dass $d(v_k, r_{i,j}) \leq \varepsilon$

Berechne Abkürzungsgraph G'', der Kante (v_i, v_j) enthält, wenn für jedes $i < k < j$ gilt, dass $d(v_k, r_{j,i}) \leq \varepsilon$

G enthält Kante (v_i, v_j), wenn G' und G'' Kante (v_i, v_j) enthalten.



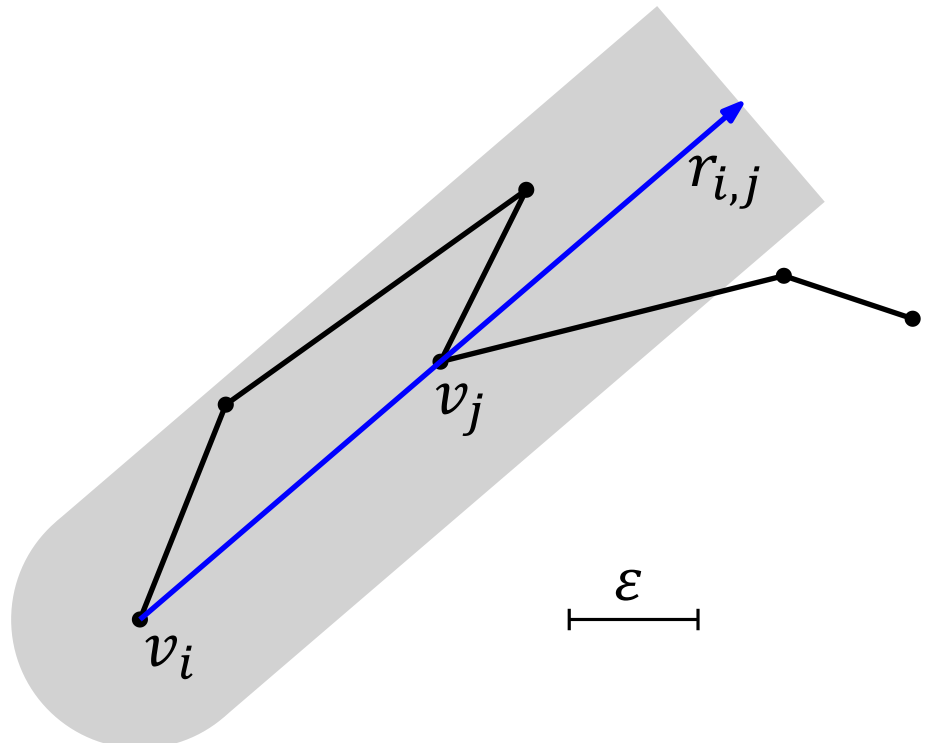
Definition:
$r_{i,j}$ = Strahl von v_i durch v_j

Linienvereinfachung
durch Optimierung (Chan & Chin, 1996)

Aufgabe:
Berechne Abkürzungsgraph G

Lösung (besserer Algorithmus):

Berechne Abkürzungsgraph G',
der Kante (v_i, v_j) enthält,
wenn für jedes $i < k < j$ gilt,
dass $d(v_k, r_{i,j}) \leq \varepsilon$



Definition:
$r_{i,j}$ = Strahl von v_i durch v_j

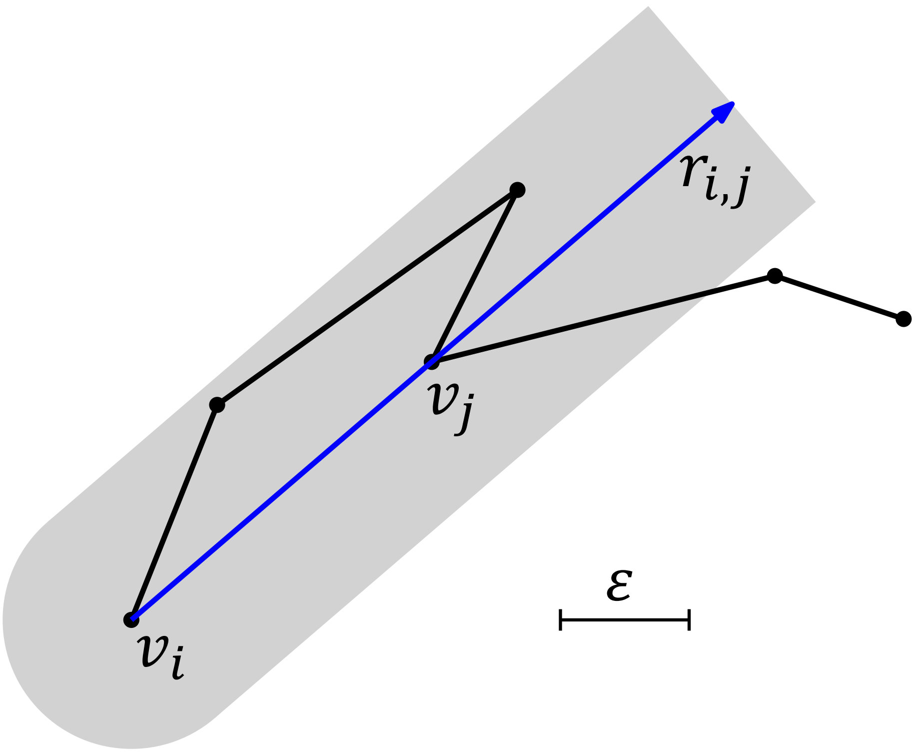
Linienvereinfachung
durch Optimierung (Chan & Chin, 1996)

Aufgabe:
Berechne Abkürzungsgraph G

Lösung (besserer Algorithmus):

Berechne Abkürzungsgraph G', der Kante (v_i, v_j) enthält, wenn für jedes $i < k < j$ gilt, dass $d(v_k, r_{i,j}) \leq \varepsilon$

Algorithmus:
Für jeden Knoten v_i berechne die Kanten (v_i, v_j) in G' in **linearer Zeit**.



Linienvereinfachung
durch Optimierung (Chan & Chin, 1996)

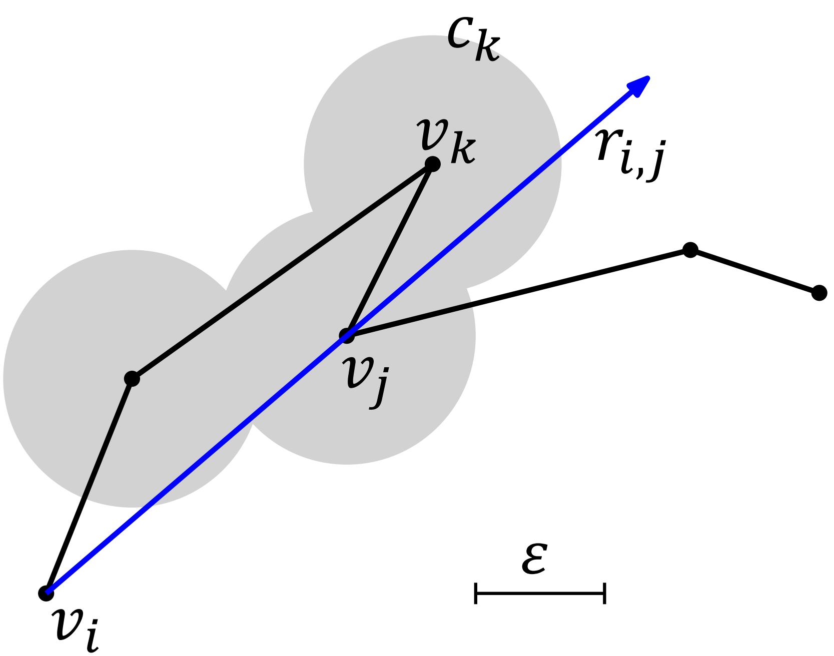
Aufgabe:
Berechne Abkürzungsgraph G

Lösung (besserer Algorithmus):

Berechne Abkürzungsgraph G',
der Kante (v_i, v_j) enthält,
wenn für jedes $i < k < j$ gilt,
dass $d(v_k, r_{i,j}) \leq \varepsilon$

Algorithmus:
Für jeden Knoten v_i berechne die Kanten (v_i, v_j) in G' in **linearer Zeit**.

(v_i, v_j) ist in G', wenn $r_{i,j}$ jeden Kreis c_k mit $i < k \leq j$ schneidet.



Definition:
c_k = Kreis um v_k mit Radius ε

Linienvereinfachung
durch Optimierung (Chan & Chin, 1996)

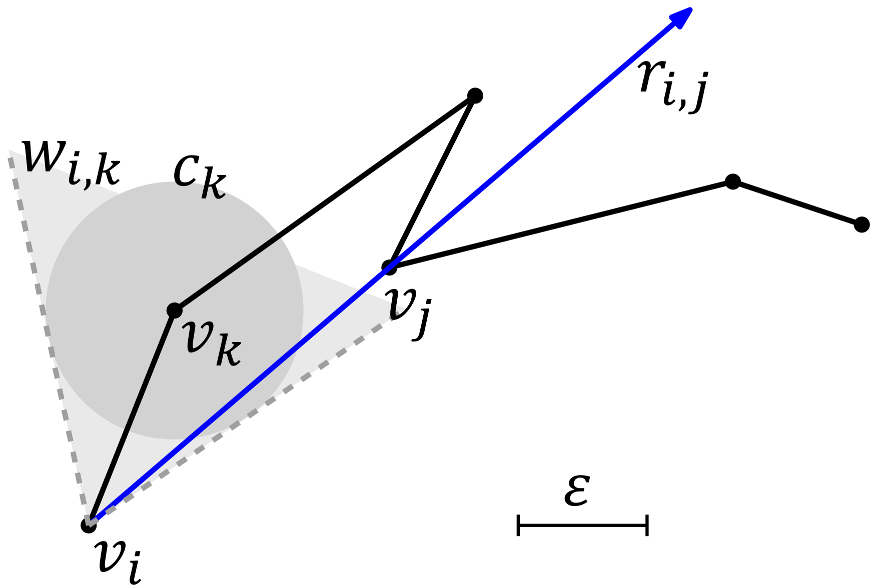
Aufgabe:
Berechne Abkürzungsgraph G

Lösung (besserer Algorithmus):

Berechne Abkürzungsgraph G',
der Kante (v_i, v_j) enthält,
wenn für jedes $i < k < j$ gilt,
dass $d(v_k, r_{i,j}) \leq \varepsilon$

Algorithmus:
Für jeden Knoten v_i berechne die
Kanten (v_i, v_j) in G' in **linearer Zeit**.

(v_i, v_j) ist in G', wenn jeder Keil $w_{i,k}$
mit $i < k \leq j$ den Strahl $r_{i,j}$ enthält.



Definition:
$w_{i,k}$ = kleinster Keil mit Spitze v_i,
der c_k enthält

Linienvereinfachung
durch Optimierung (Chan & Chin, 1996)

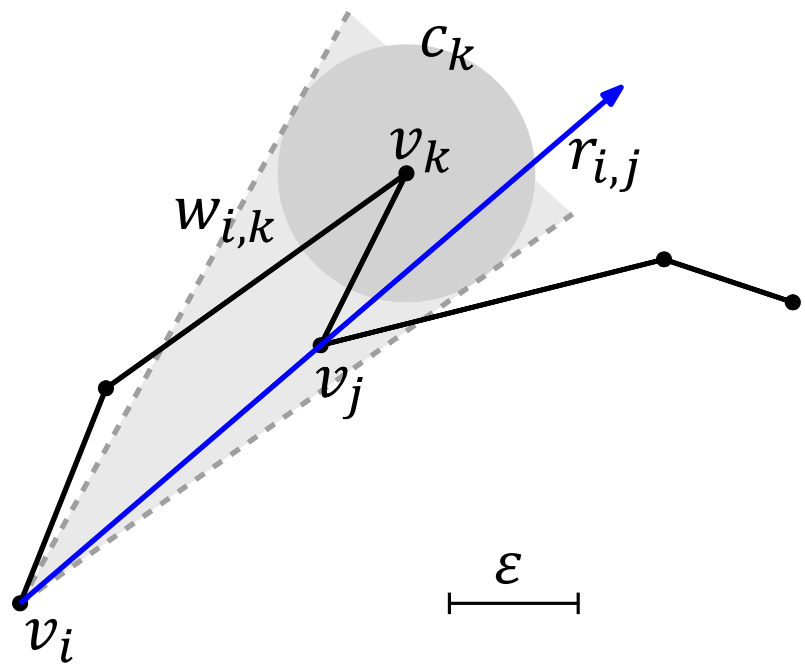
Aufgabe:
Berechne Abkürzungsgraph G

Lösung (besserer Algorithmus):

Berechne Abkürzungsgraph G',
der Kante (v_i, v_j) enthält,
wenn für jedes $i < k < j$ gilt,
dass $d(v_k, r_{i,j}) \leq \varepsilon$

Algorithmus:
Für jeden Knoten v_i berechne die Kanten (v_i, v_j) in G' in **linearer Zeit**.

(v_i, v_j) ist in G', wenn jeder Keil $w_{i,k}$ mit $i < k \leq j$ den Strahl $r_{i,j}$ enthält.



Definition:
$w_{i,k}$ = kleinster Keil mit Spitze v_i, der c_k enthält

Linienvereinfachung
durch Optimierung (Chan & Chin, 1996)

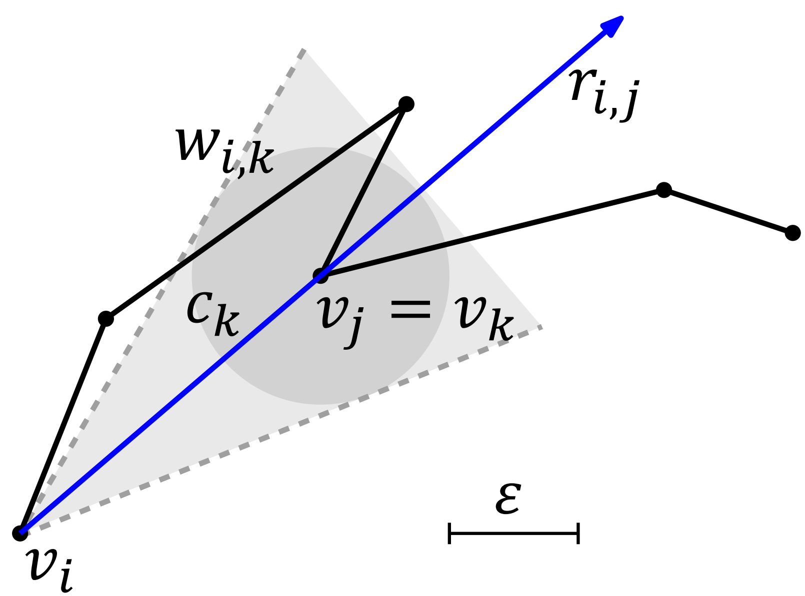
Aufgabe:
Berechne Abkürzungsgraph G

Lösung (besserer Algorithmus):

Berechne Abkürzungsgraph G',
der Kante (v_i, v_j) enthält,
wenn für jedes $i < k < j$ gilt,
dass $d(v_k, r_{i,j}) \leq \varepsilon$

Algorithmus:
Für jeden Knoten v_i berechne die Kanten (v_i, v_j) in G' in **linearer Zeit**.

(v_i, v_j) ist in G', wenn jeder Keil $w_{i,k}$ mit $i < k \leq j$ den Strahl $r_{i,j}$ enthält.



Definition:
$w_{i,k}$ = kleinster Keil mit Spitze v_i, der c_k enthält

Linienvereinfachung
durch Optimierung (Chan & Chin, 1996)

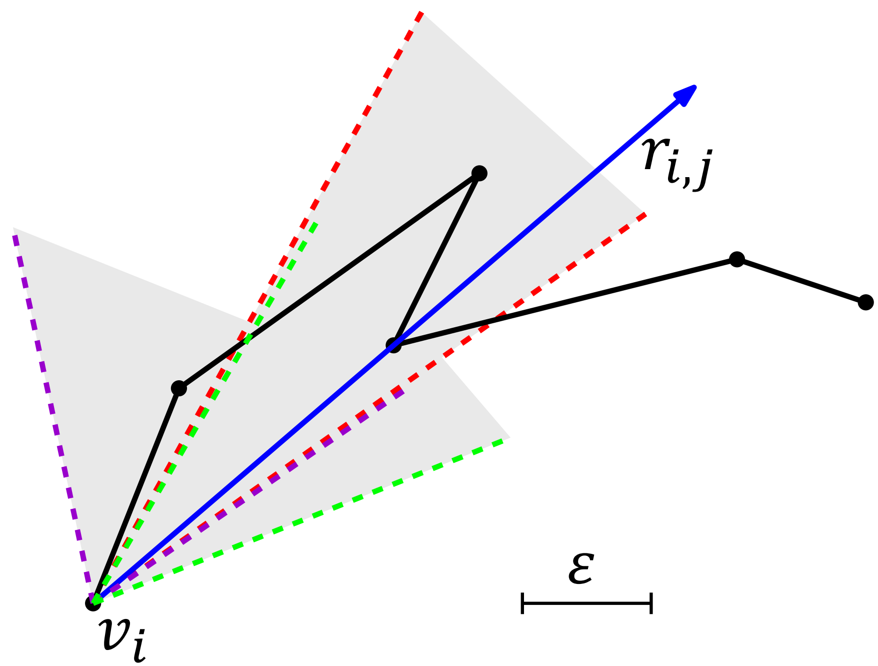
Aufgabe:
Berechne Abkürzungsgraph G

Lösung (besserer Algorithmus):

Berechne Abkürzungsgraph G',
der Kante (v_i, v_j) enthält,
wenn für jedes $i < k < j$ gilt,
dass $d(v_k, r_{i,j}) \leq \varepsilon$

Algorithmus:
Für jeden Knoten v_i berechne die Kanten (v_i, v_j) in G' in **linearer Zeit**.

(v_i, v_j) ist in G', wenn die Schnittmenge $W_{i,j}$ der Keile $w_{i,k}$ mit $i < k \leq j$ den Strahl $r_{i,j}$ enthält.



Definition:
$w_{i,k}$ = kleinster Keil mit Spitze v_i, der c_k enthält

Linienvereinfachung
durch Optimierung (Chan & Chin, 1996)

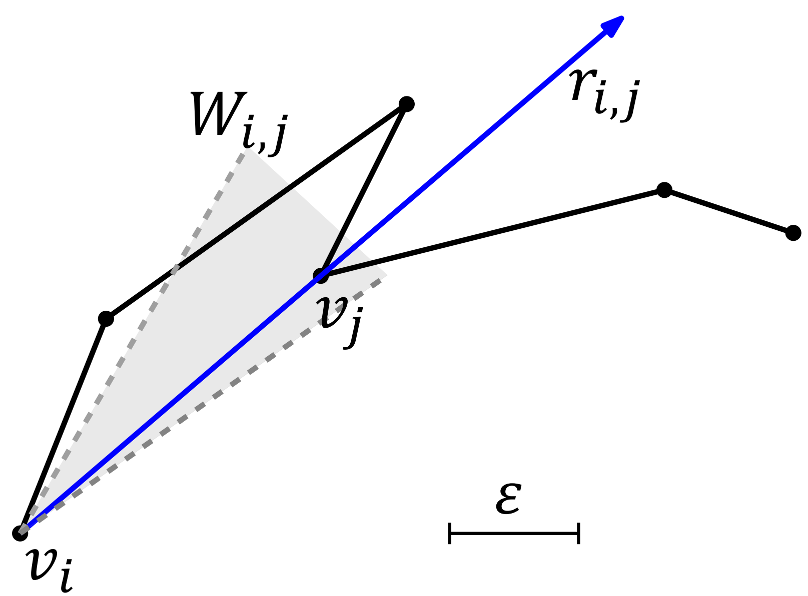
Aufgabe:
Berechne Abkürzungsgraph G

Lösung (besserer Algorithmus):

Berechne Abkürzungsgraph G',
der Kante (v_i, v_j) enthält,
wenn für jedes $i < k < j$ gilt,
dass $d(v_k, r_{i,j}) \leq \varepsilon$

Algorithmus:
Für jeden Knoten v_i berechne die Kanten (v_i, v_j) in G' in **linearer Zeit**.

(v_i, v_j) ist in G', wenn die Schnittmenge $W_{i,j}$ der Keile $w_{i,k}$ mit $i < k \leq j$ den Strahl $r_{i,j}$ enthält.



Definition:
$$W_{i,j} = w_{i,i+1} \cap w_{i,i+2} \cap \cdots \cap w_{i,j}$$

Linienvereinfachung
durch Optimierung (Chan & Chin, 1996)

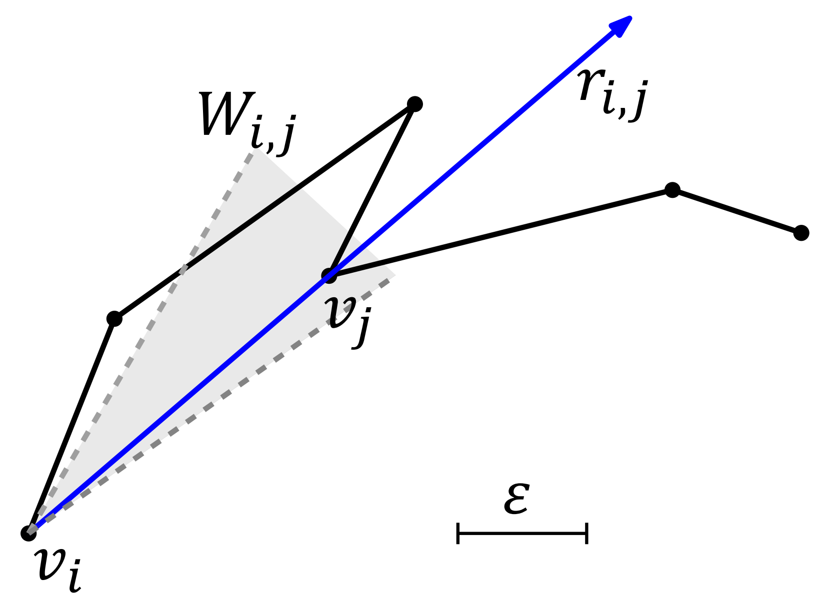
Aufgabe:
Berechne Abkürzungsgraph G

Lösung (besserer Algorithmus):

Berechne Abkürzungsgraph G',
der Kante (v_i, v_j) enthält,
wenn für jedes $i < k < j$ gilt,
dass $d(v_k, r_{i,j}) \leq \varepsilon$

Algorithmus:
Für jeden Knoten v_i berechne die Kanten (v_i, v_j) in G' in **linearer Zeit**.

(v_i, v_j) ist in G', wenn die Schnittmenge $W_{i,j}$ der Keile $w_{i,k}$ mit $i < k \leq j$ den Strahl $r_{i,j}$ enthält.



Definition:

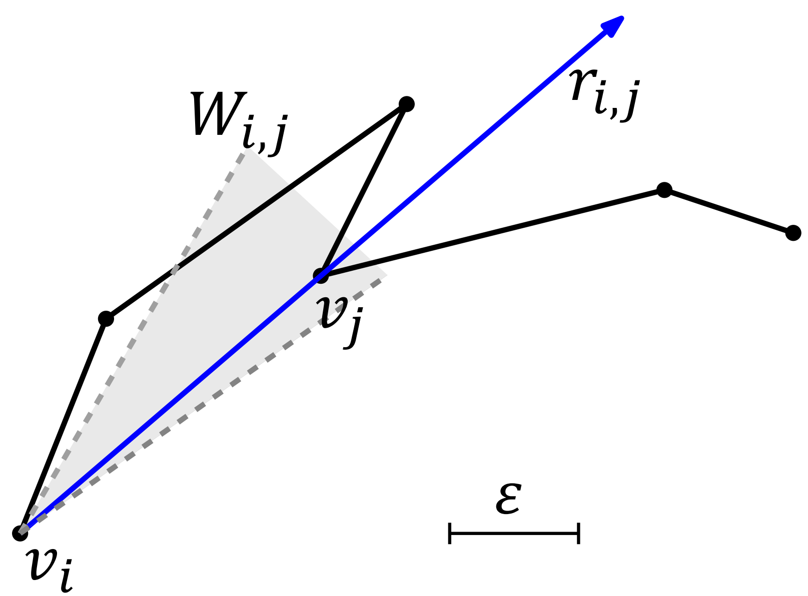
$$W_{i,j} = w_{i,i+1} \cap w_{i,i+2} \cap \cdots \cap w_{i,j}$$
$$= W_{i,j-1} \cap w_{i,j}$$

Linienvereinfachung
durch Optimierung (Chan & Chin, 1996)

Aufgabe:
Berechne Abkürzungsgraph G

Lösung (besserer Algorithmus):

Berechne Abkürzungsgraph G',
der Kante (v_i, v_j) enthält,
wenn für jedes $i < k < j$ gilt,
dass $d(v_k, r_{i,j}) \leq \varepsilon$



Algorithmus:

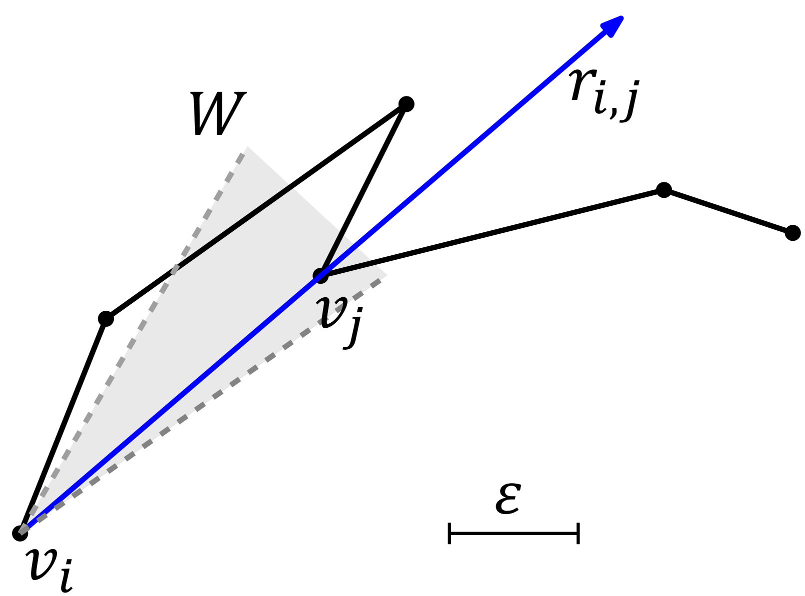
for $i = 1$ **to** n
- $W_{i,i} = \mathbb{R}^2$
- **for** $j = i + 1$ **to** n
  - $W_{i,j} = W_{i,j-1} \cap w_{i,j}$
  - **if** $W_{i,j} = \emptyset$ **then** break
  - **if** $r_{i,j} \subseteq W_{i,j}$ **then** add (v_i, v_j) to G'

Linienvereinfachung
durch Optimierung (Chan & Chin, 1996)

Aufgabe:
Berechne Abkürzungsgraph G

Lösung (besserer Algorithmus):

Berechne Abkürzungsgraph G',
der Kante (v_i, v_j) enthält,
wenn für jedes $i < k < j$ gilt,
dass $d(v_k, r_{i,j}) \leq \varepsilon$



Algorithmus:

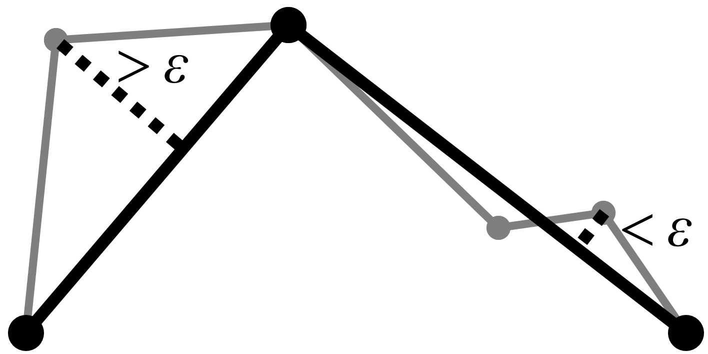
for $i = 1$ **to** n
  $W = \mathbb{R}^2$
  for $j = i + 1$ **to** n
    $W = W \cap w_{i,j}$
    if $W = \emptyset$ **then** break
    if $r_{i,j} \subseteq W$ **then** add (v_i, v_j) to G'

Laufzeit: $O(n^2)$

Douglas-Peucker vs. Optimierung

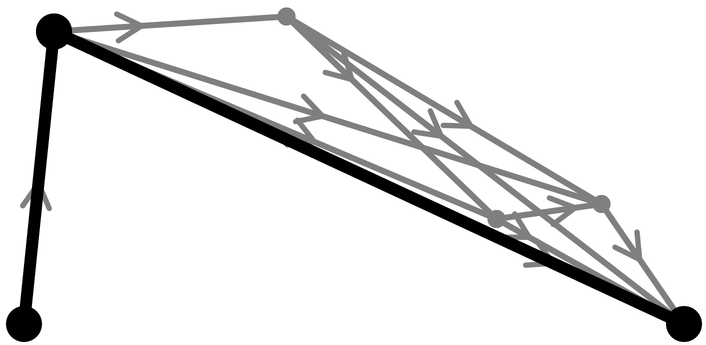
$$O(n^2)$$

(Douglas & Peucker, 1973)



$$O(n^3)$$

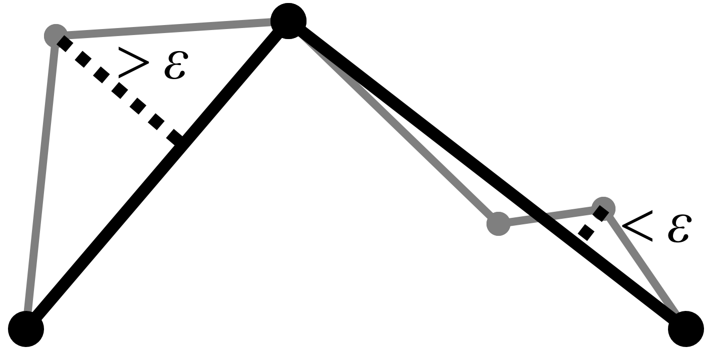
(Imai & Iri, 1988)



Douglas-Peucker vs. Optimierung

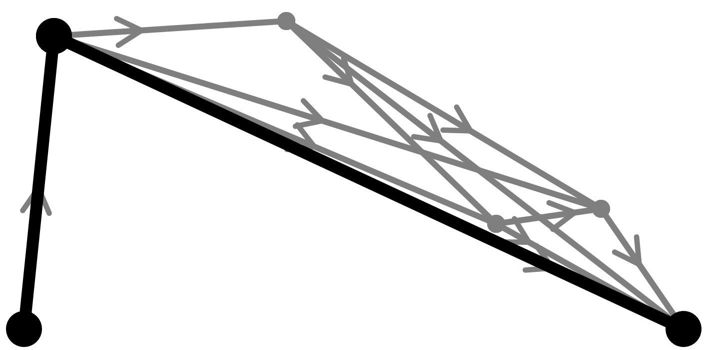
$O(n \log n)$

(Hershberger & Snoeyink, 1992)



$O(n^2)$

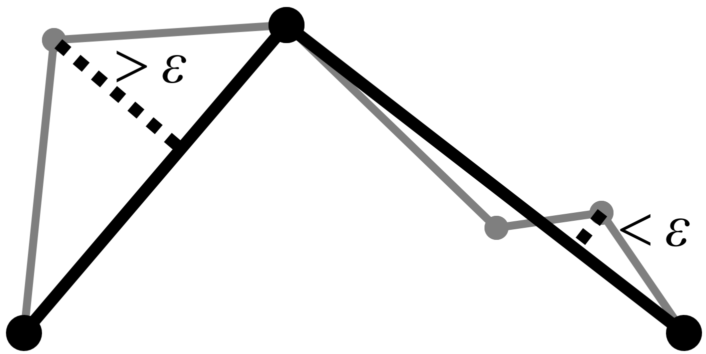
(Chan & Chin, 1996)



Douglas-Peucker vs. Optimierung

$O(n \log n)$

(Hershberger & Snoeyink, 1992)

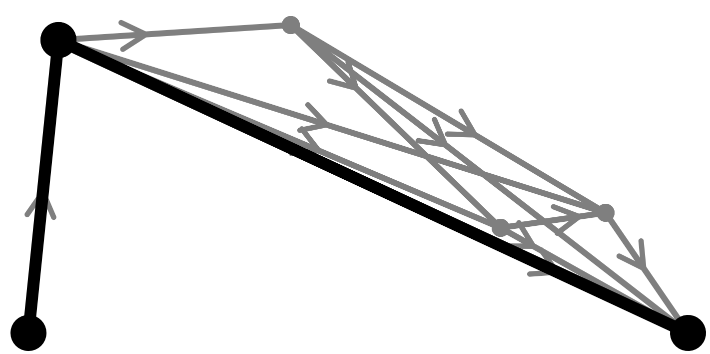


$O(n^2)$

(Chan & Chin, 1996)

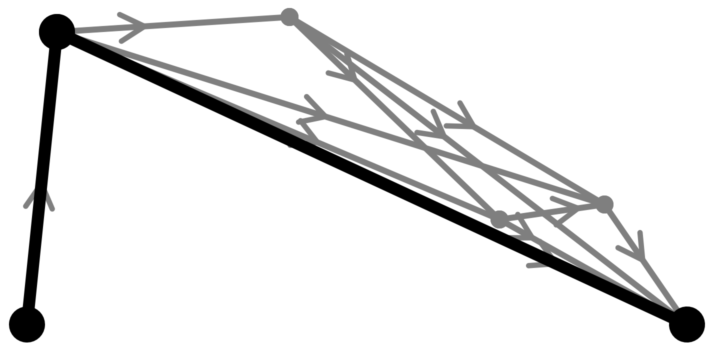
Approximationsalg. mit Laufzeit $O(n)$:

ε-zulässige Vereinfachung mit höchstens so vielen Ecken wie optimale $\varepsilon/2$-zulässige Vereinfachung (Agarwal et al., 2002)



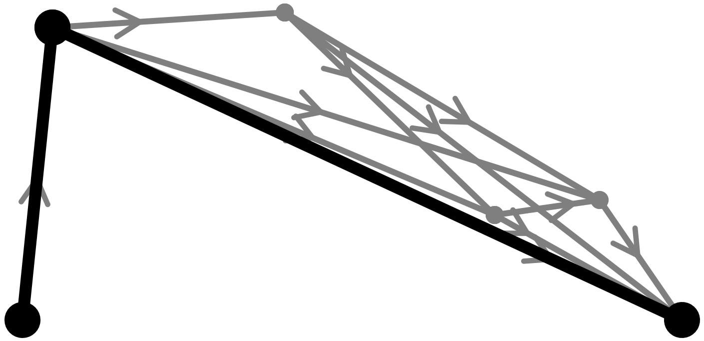
Optimierungsansatz

- Modellierung als Kürzeste-Wege-Problem (Imai & Iri, 1988)



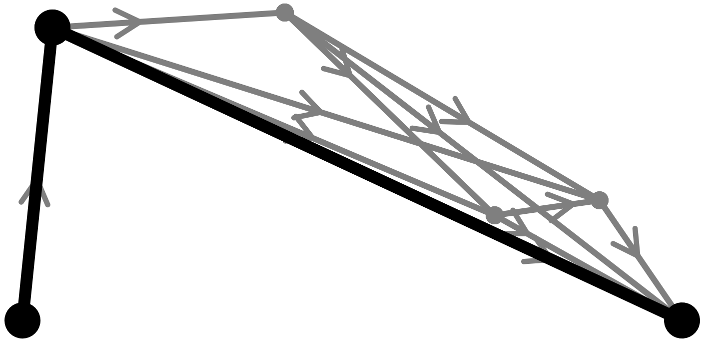
Optimierungsansatz

- Modellierung als Kürzeste-Wege-Problem (Imai & Iri, 1988)
- Vermeidung von Selbstschnitten (de Berg et al., 1998)



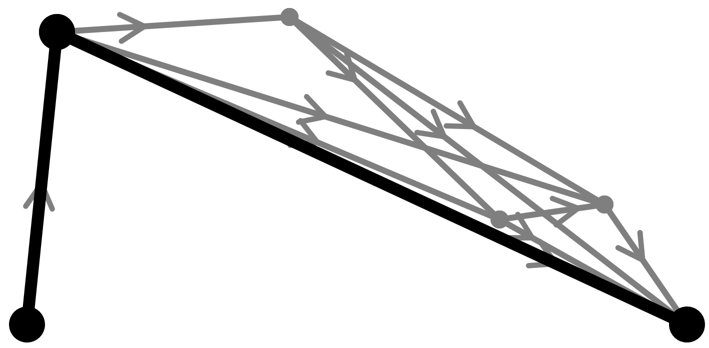
Optimierungsansatz

- Modellierung als Kürzeste-Wege-Problem (Imai & Iri, 1988)
- Vermeidung von Selbstschnitten (de Berg et al., 1998)
- Einhaltung von Winkelbedingungen (Chen et al., 2005)



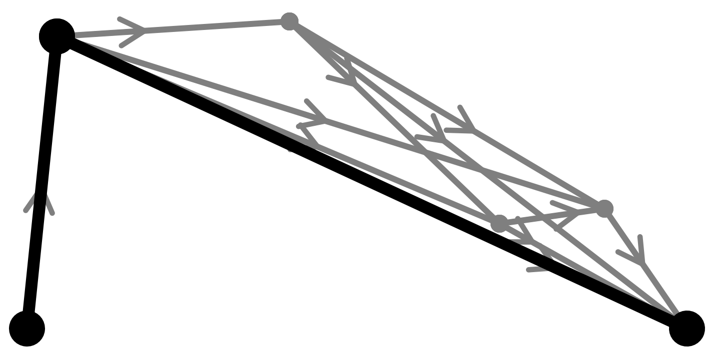
Optimierungsansatz

- Modellierung als Kürzeste-Wege-Problem (Imai & Iri, 1988)
- Vermeidung von Selbstschnitten (de Berg et al., 1998)
- Einhaltung von Winkelbedingungen (Chen et al., 2005)
- Einhaltung von Flächenbedingungen (Bose et al., 2006)



Optimierungsansatz

- Modellierung als Kürzeste-Wege-Problem (Imai & Iri, 1988)
- Vermeidung von Selbstschnitten (de Berg et al., 1998)
- Einhaltung von Winkelbedingungen (Chen et al., 2005)
- Einhaltung von Flächenbedingungen (Bose et al., 2006)
- Einhaltung von Längenbedingungen (Gudmundsson et al., 2007)



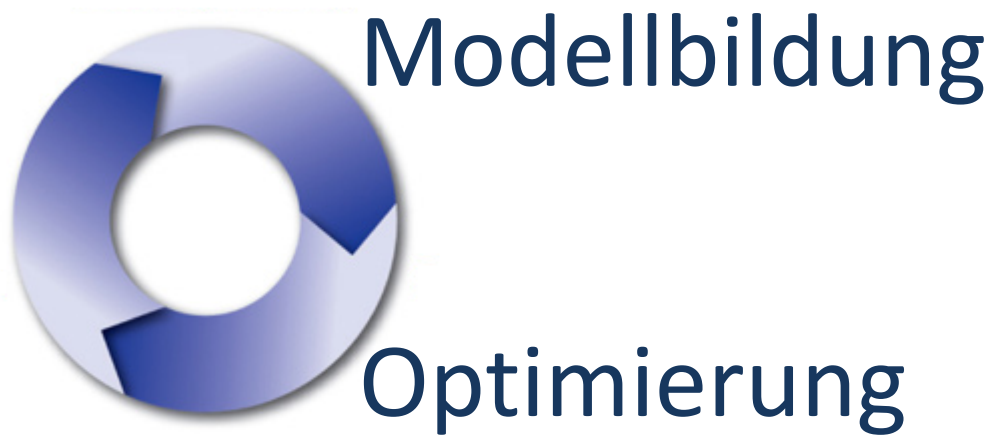
Optimierungsansatz

- Modellierung als Kürzeste-Wege-Problem (Imai & Iri, 1988)
- Vermeidung von Selbstschnitten (de Berg et al., 1998)
- Einhaltung von Winkelbedingungen (Chen et al., 2005)
- Einhaltung von Flächenbedingungen (Bose et al., 2006)
- Einhaltung von Längenbedingungen (Gudmundsson et al., 2007)

Modellbildung

Optimierungsansatz

- Modellierung als Kürzeste-Wege-Problem (Imai & Iri, 1988)
- Vermeidung von Selbstschnitten (de Berg et al., 1998)
- Einhaltung von Winkelbedingungen (Chen et al., 2005)
- Einhaltung von Flächenbedingungen (Bose et al., 2006)
- Einhaltung von Längenbedingungen (Gudmundsson et al., 2007)



Optimierungsansatz

- Modellierung als Kürzeste-Wege-Problem (Imai & Iri, 1988)
- Vermeidung von Selbstschnitten (de Berg et al., 1998)
- Einhaltung von Winkelbedingungen (Chen et al., 2005)
- Einhaltung von Flächenbedingungen (Bose et al., 2006)
- Einhaltung von Längenbedingungen (Gudmundsson et al., 2007)

Optimierungsansatz

- Modellierung als Kürzeste-Wege-Problem (Imai & Iri, 1988)
- Vermeidung von Selbstschnitten (de Berg et al., 1998)
- Einhaltung von Winkelbedingungen (Chen et al., 2005)
- Einhaltung von Flächenbedingungen (Bose et al., 2006)
- Einhaltung von Längenbedingungen (Gudmundsson et al., 2007)

Optimierung bietet Möglichkeit zur schrittweisen Modellierung von Problemen!