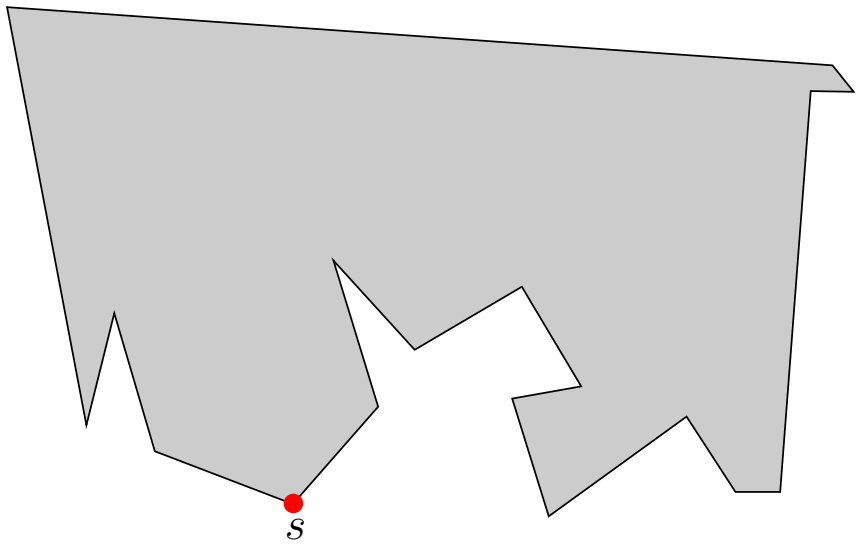
Offline Bewegungsplanung: SWR und Touring

Elmar Langetepe
University of Bonn

Was ist wichtig? Def. 1.25

Was ist wichtig? Def. 1.25

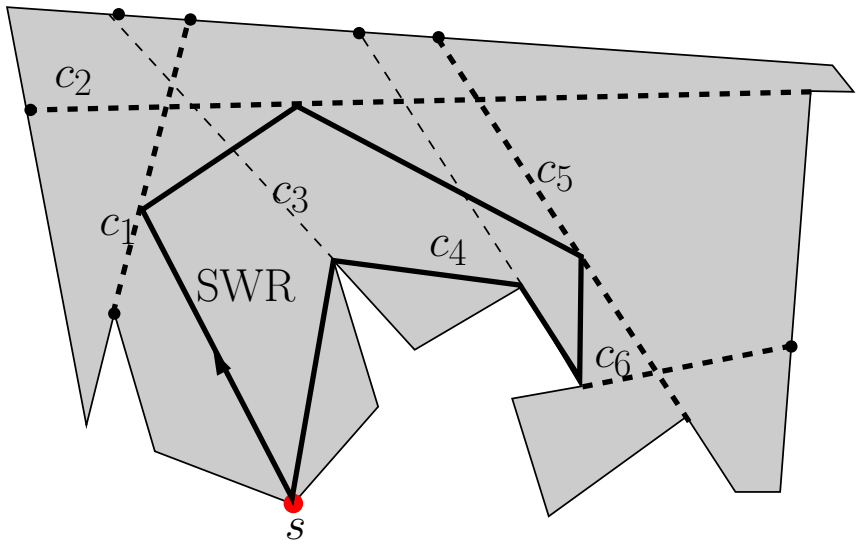
a) (Cuts) Extensionen der reflexen Ecken



Was ist wichtig? Def. 1.25

a) (Cuts) Extensionen der reflexen Ecken

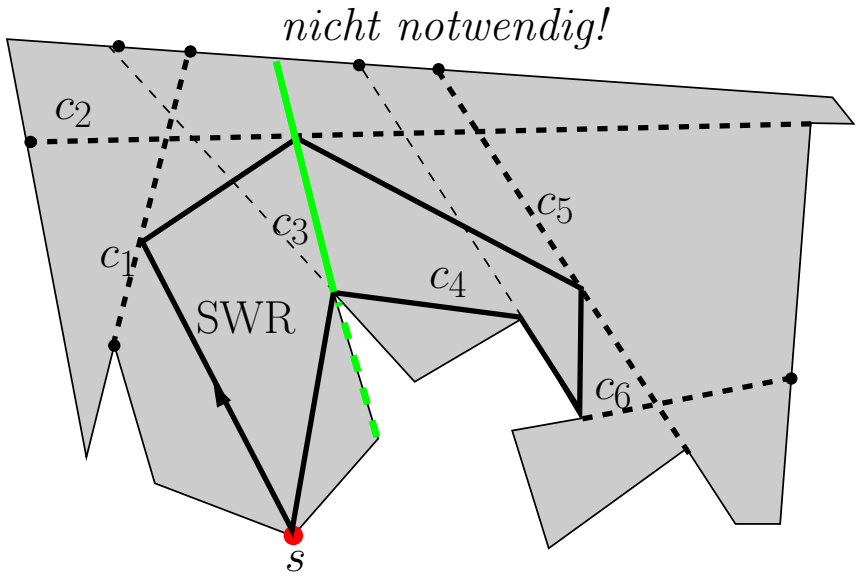
b) Notwendige Cuts (bezüglich s)



Was ist wichtig? Def. 1.25

a) (Cuts) Extensionen der reflexen Ecken

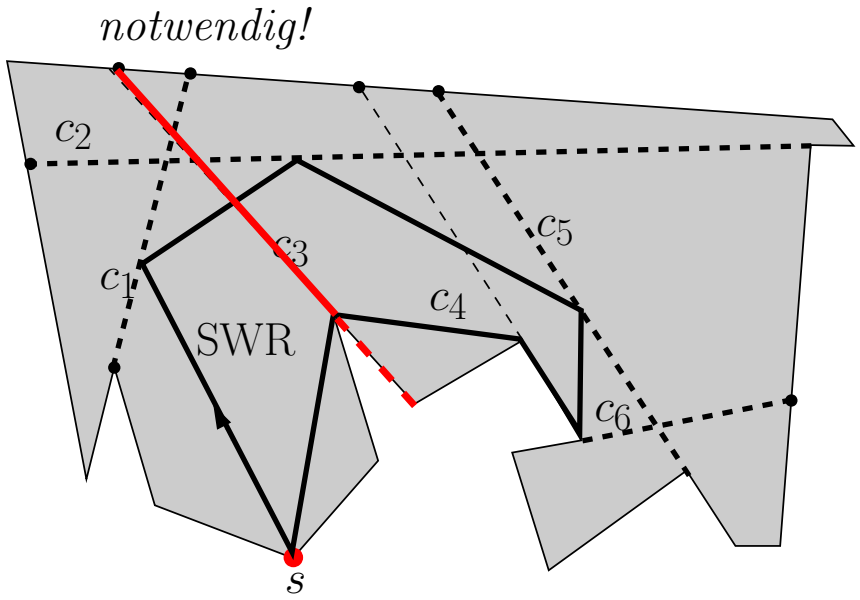
b) Notwendige Cuts (bezüglich s)



Was ist wichtig? Def. 1.25

a) (Cuts) Extensionen der reflexen Ecken

b) Notwendige Cuts (bezüglich s)

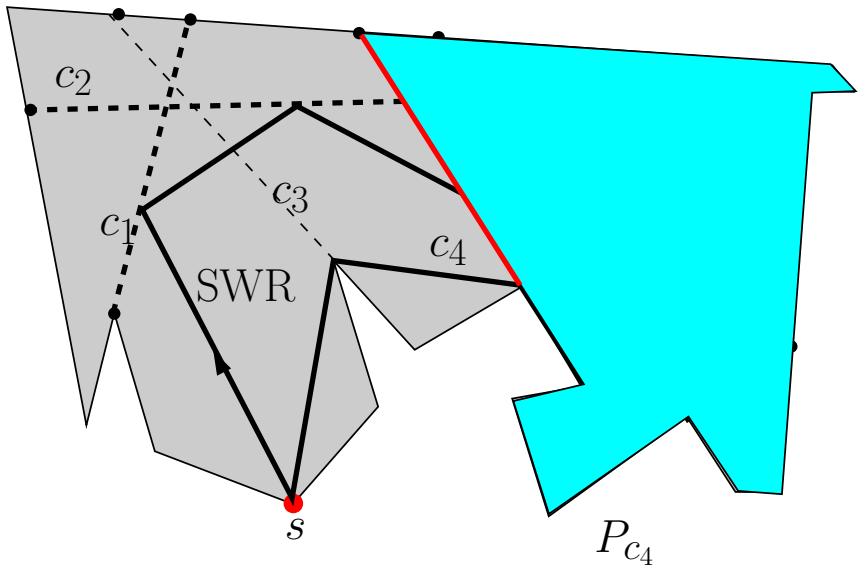


Was ist wichtig? Def. 1.25

a) (Cuts) Extensionen der reflexen Ecken

b) Notwendige Cuts (bezüglich s)

c) Dominanz-Beziehung $P_{c_i} \subseteq P_{c_j}$

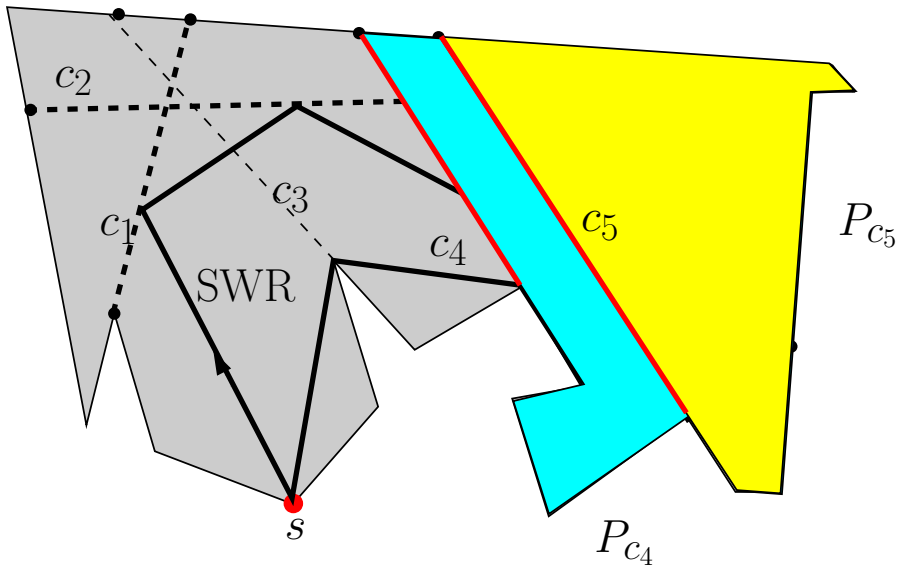


Was ist wichtig? Def. 1.25

a) (Cuts) Extensionen der reflexen Ecken

b) Notwendige Cuts (bezüglich s)

c) Dominanz-Beziehung $P_{c_i} \subseteq P_{c_j}$

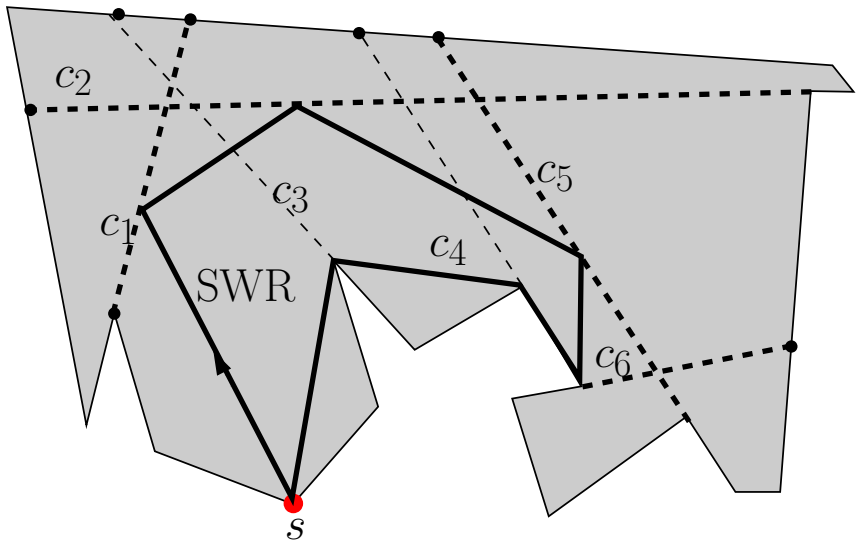


Was ist wichtig? Def. 1.25

a) (Cuts) Extensionen der reflexen Ecken

b) Notwendige Cuts (bezüglich s)

c) Dominanz-Beziehung $P_{c_i} \subseteq P_{c_j}$



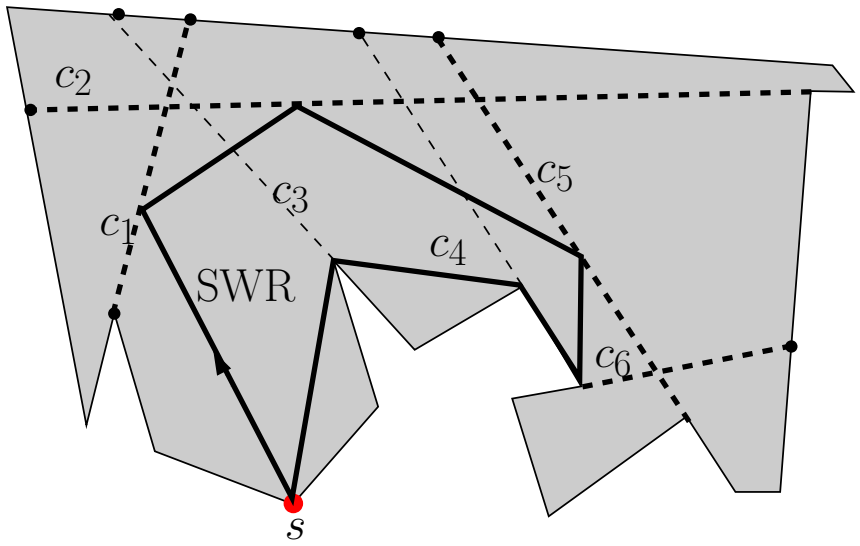
Was ist wichtig? Def. 1.25

a) (Cuts) Extensionen der reflexen Ecken

b) Notwendige Cuts (bezüglich s)

c) Dominanz-Beziehung $P_{c_i} \subseteq P_{c_j}$

d) Wesentliche Cuts



Was ist wichtig? Def. 1.25

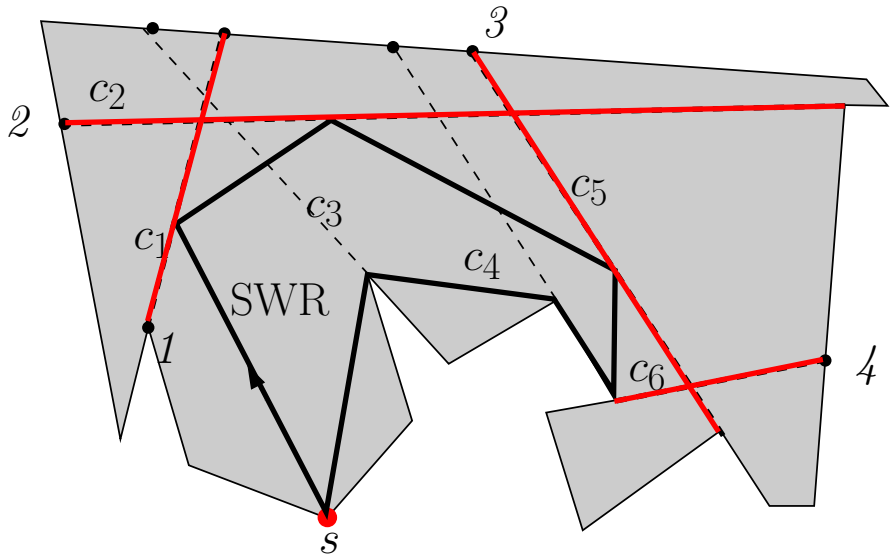
a) (Cuts) Extensionen der reflexen Ecken

b) Notwendige Cuts (bezüglich s)

c) Dominanz-Beziehung $P_{c_i} \subseteq P_{c_j}$

d) Wesentliche Cuts

e) Ordnung der wesentlichen Cuts



Was ist wichtig? Def. 1.25

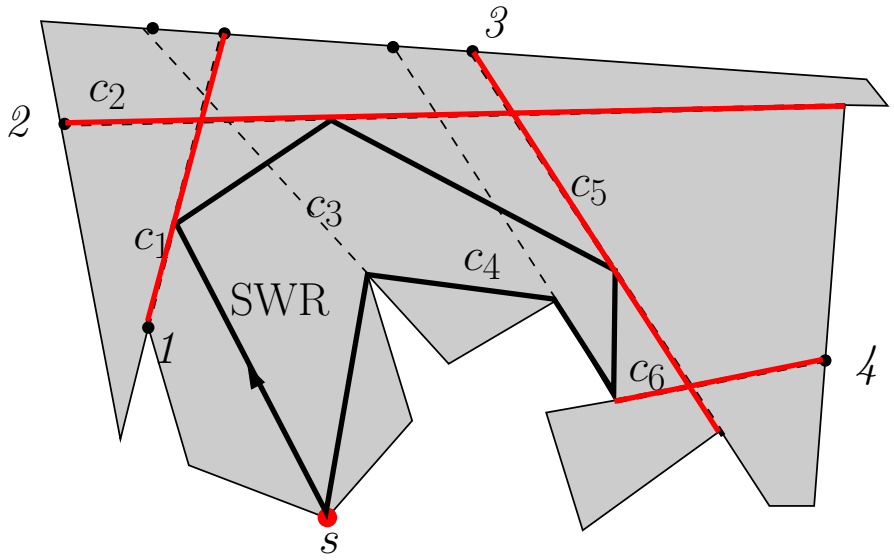
a) (Cuts) Extensionen der reflexen Ecken

b) Notwendige Cuts (bezüglich s)

c) Dominanz-Beziehung $P_{c_i} \subseteq P_{c_j}$

d) Wesentliche Cuts

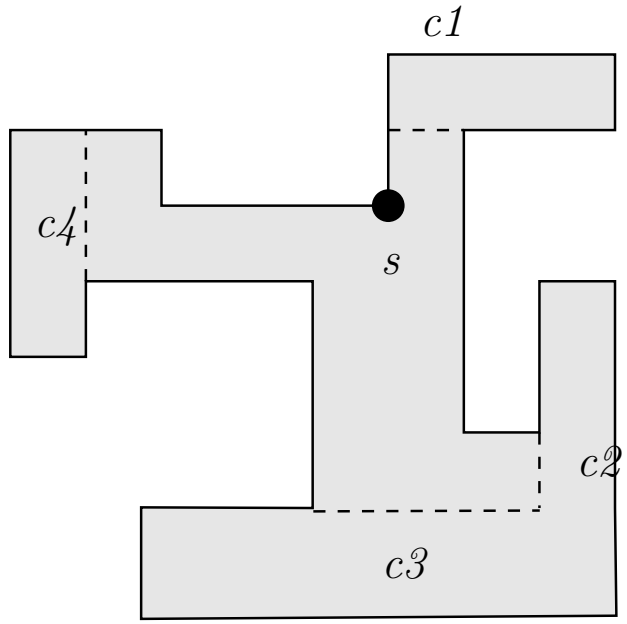
e) Ordnung der wesentlichen Cuts



Lemma 1.26 Ordnung entlang des Randes

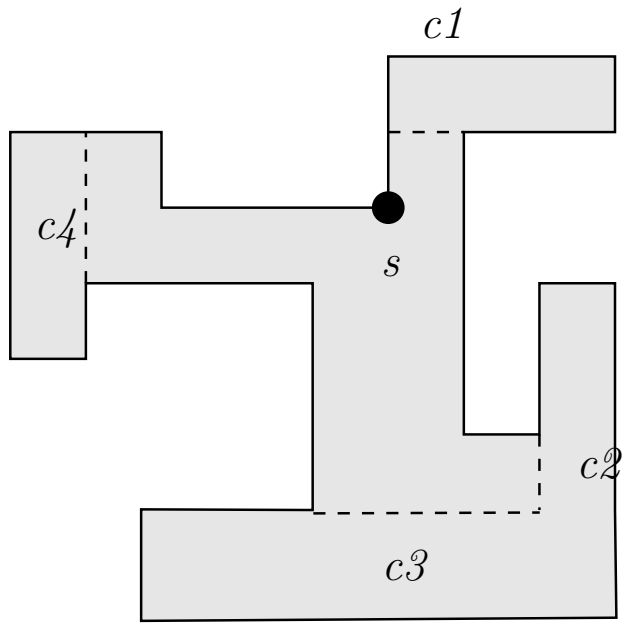
Lemma 1.26 Ordnung entlang des Randes

- Rechtwinkeliges Polygon



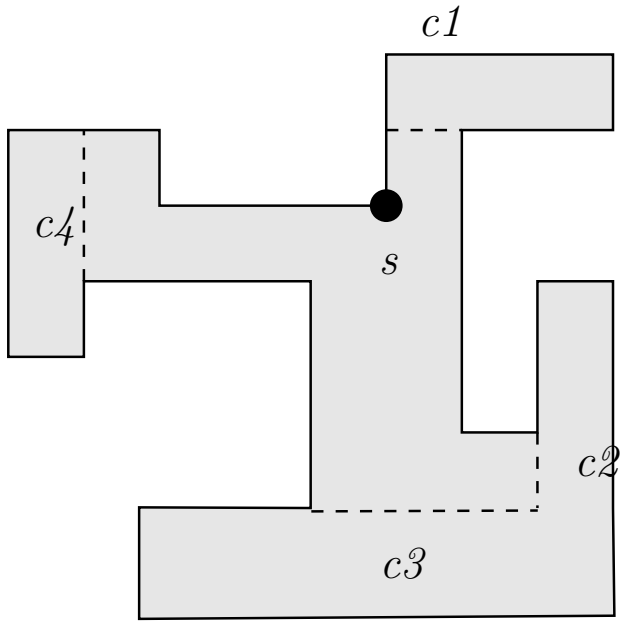
Lemma 1.26 Ordnung entlang des Randes

- Rechtwinkeliges Polygon
- Wesentliche Cuts werden nie überschritten!



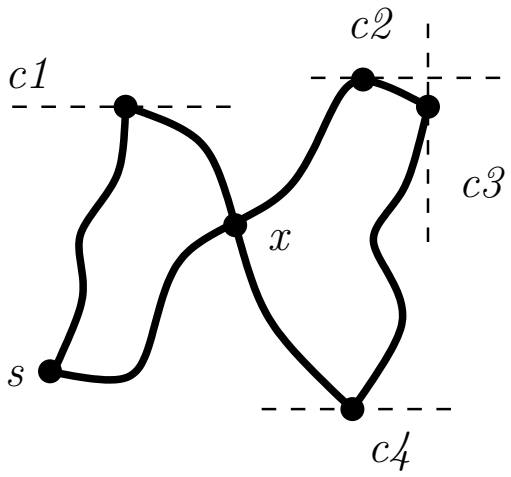
Lemma 1.26 Ordnung entlang des Randes

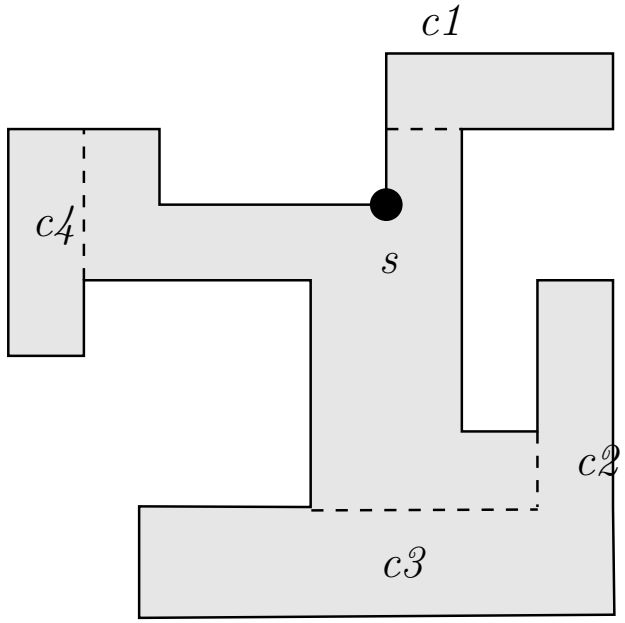
- Rechtwinkeliges Polygon
- Wesentliche Cuts werden nie überschritten!
- SWR besucht Cuts gemäß Ordnung



Lemma 1.26 Ordnung entlang des Randes

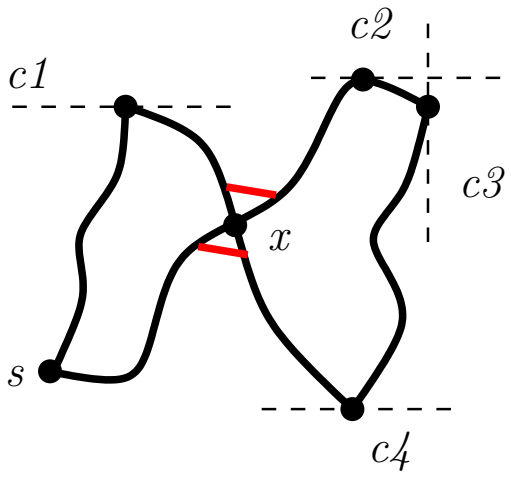
- Rechtwinkeliges Polygon
- Wesentliche Cuts werden nie überschritten!
- SWR besucht Cuts gemäß Ordnung
- Widerspruch! Abkürzung!

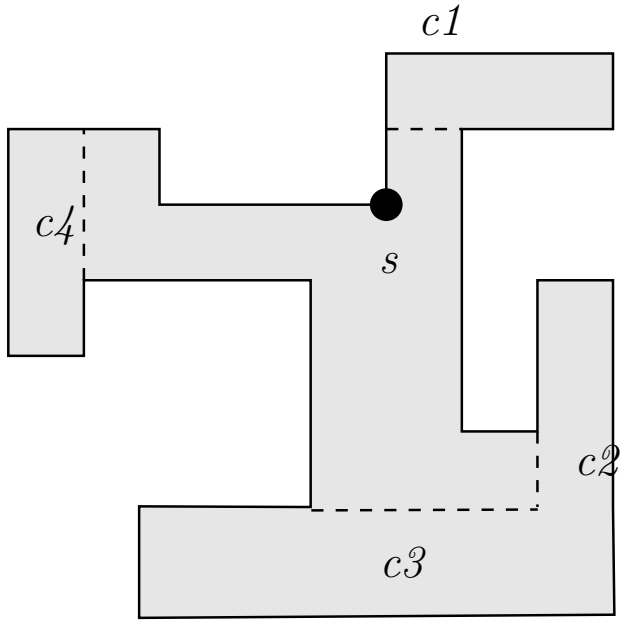




Lemma 1.26 Ordnung entlang des Randes

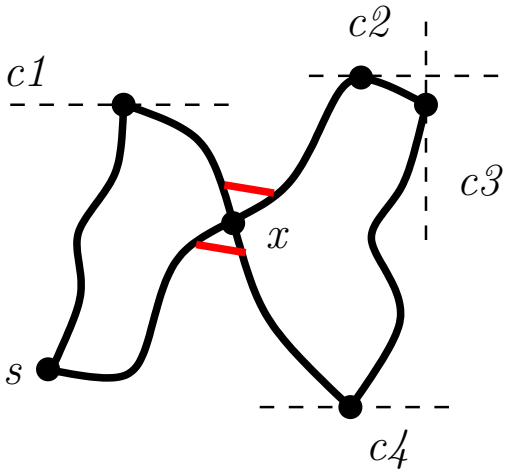
- Rechtwinkeliges Polygon
- Wesentliche Cuts werden nie überschritten!
- SWR besucht Cuts gemäß Ordnung
- Widerspruch! Abkürzung!

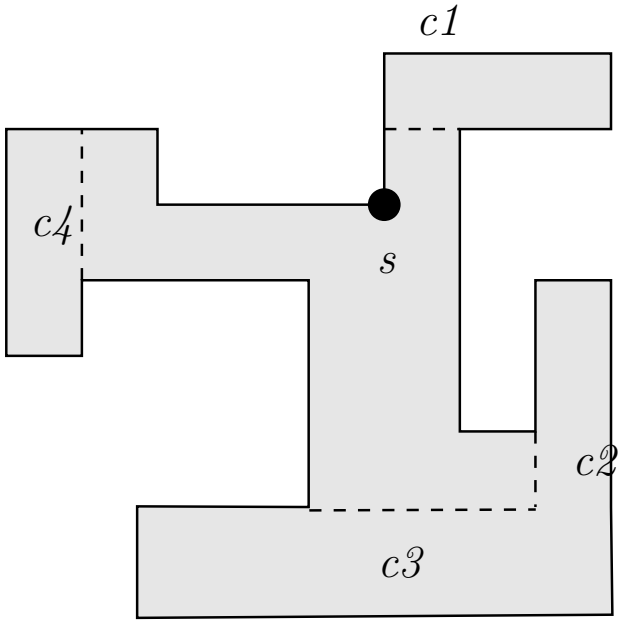




Lemma 1.26 Ordnung entlang des Randes

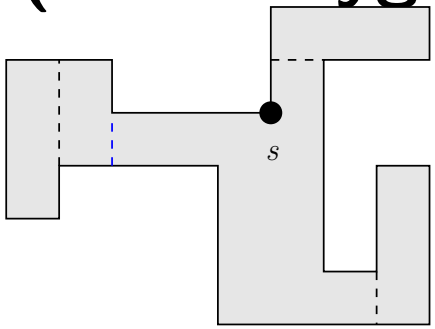
- Rechtwinkeliges Polygon
- Wesentliche Cuts werden nie überschritten!
- SWR besucht Cuts gemäß Ordnung
- Widerspruch! Abkürzung!
- $O(n)$ Algorithmus!!



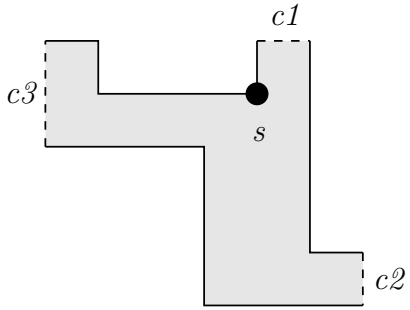


SWR (RW Polygon) Alg. 1.9/Th. 1.27: $O(n)$

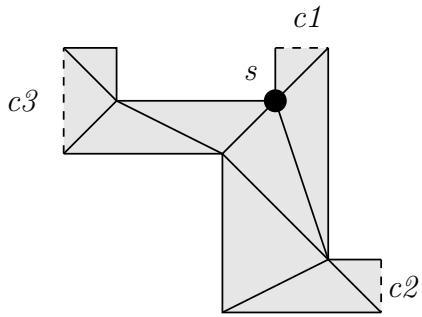
SWR (RW Polygon) Alg. 1.9/Th. 1.27: $O(n)$



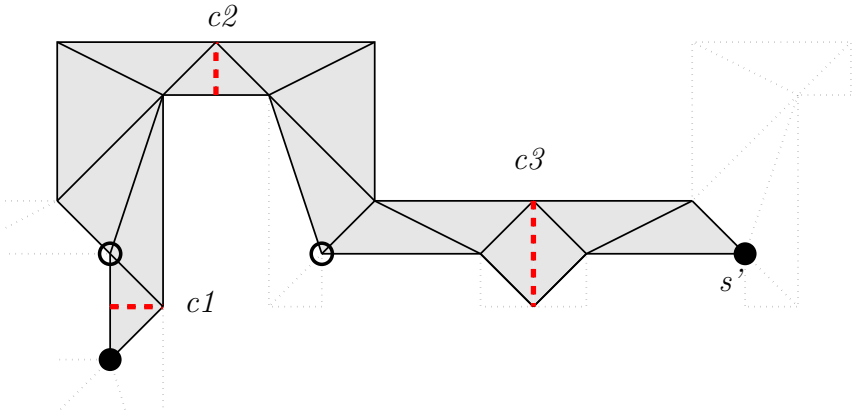
(i) Wesentliche Cuts

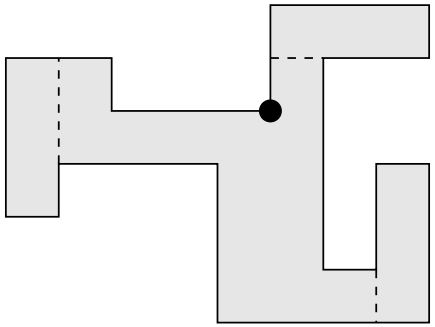


(ii) Abschneiden!



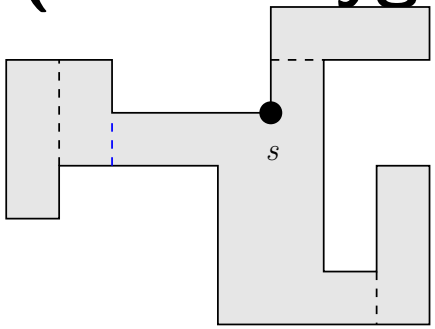
(iii) Triangulation



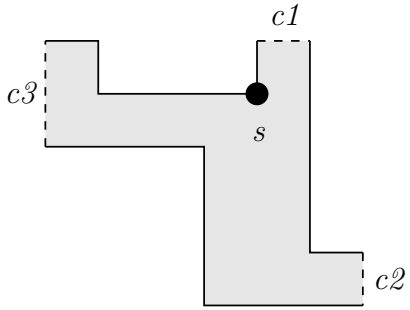


(iv) Spiegeln und Ausrollen!!

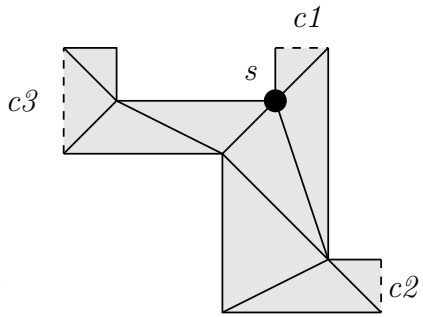
SWR (RW Polygon) Alg. 1.9/Th. 1.27: $O(n)$



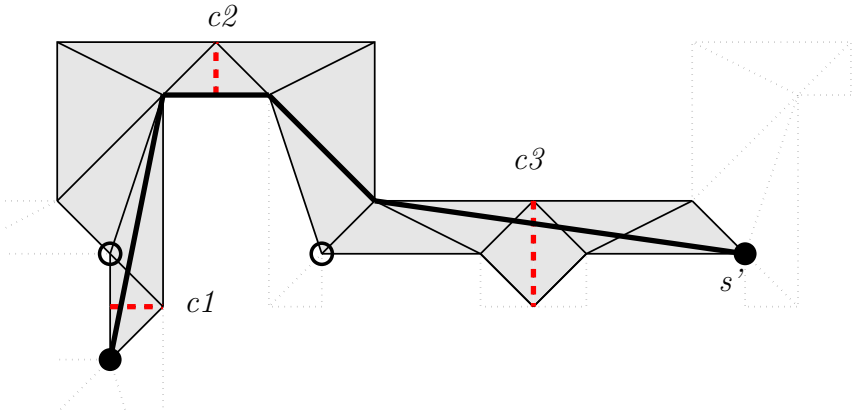
(i) Wesentliche Cuts

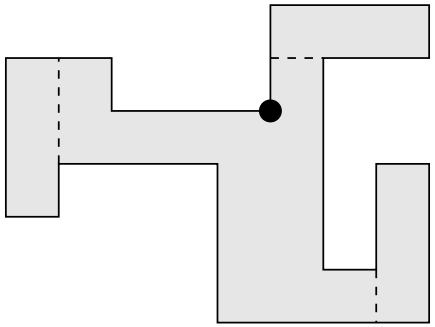


(ii) Abschneiden!



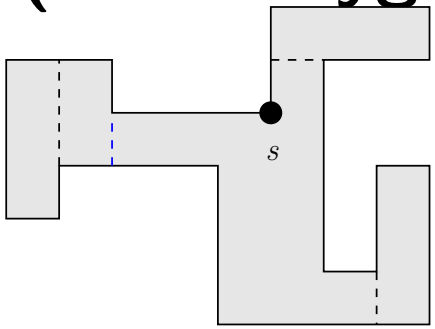
(iii) Triangulation



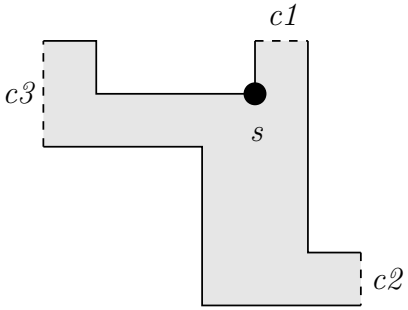


(iv) Spiegeln und Ausrollen!!
(v) Weg berechnen

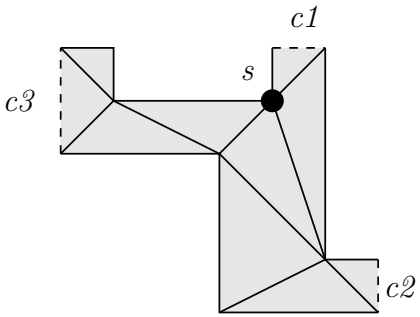
SWR (RW Polygon) Alg. 1.9/Th. 1.27: $O(n)$



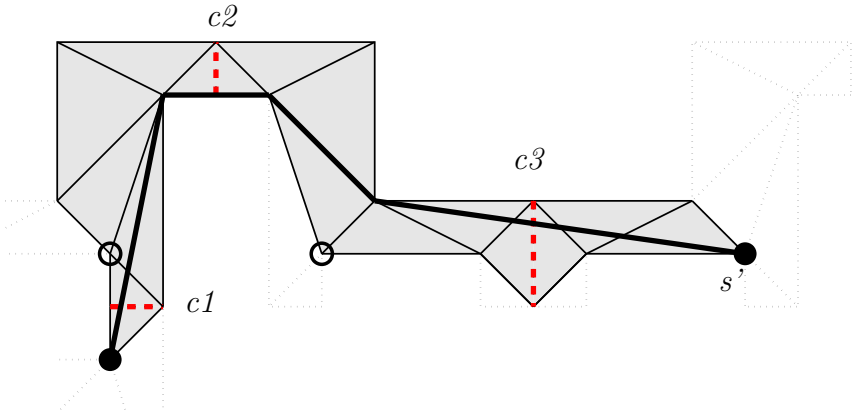
(i) Wesentliche Cuts



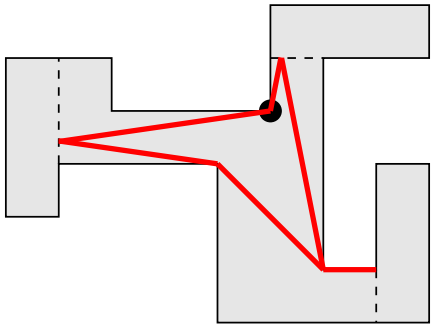
(ii) Abschneiden!



(iii) Triangulation

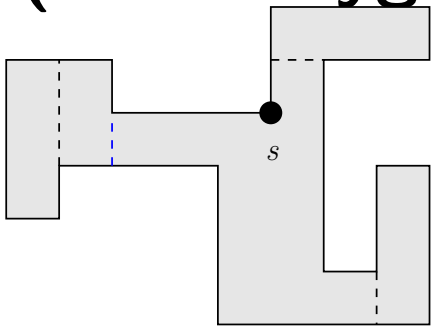


(iv) Spiegeln und Ausrollen!!
(v) Weg berechnen

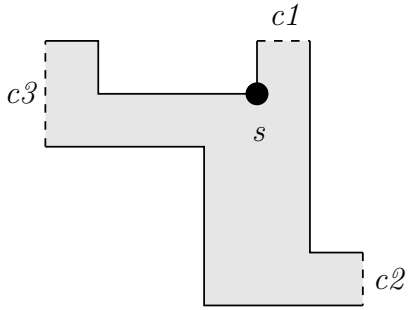


(vi) Zusammenklappen!

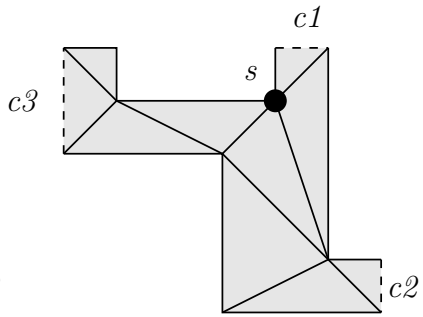
SWR (RW Polygon) Alg. 1.9/Th. 1.27: $O(n)$



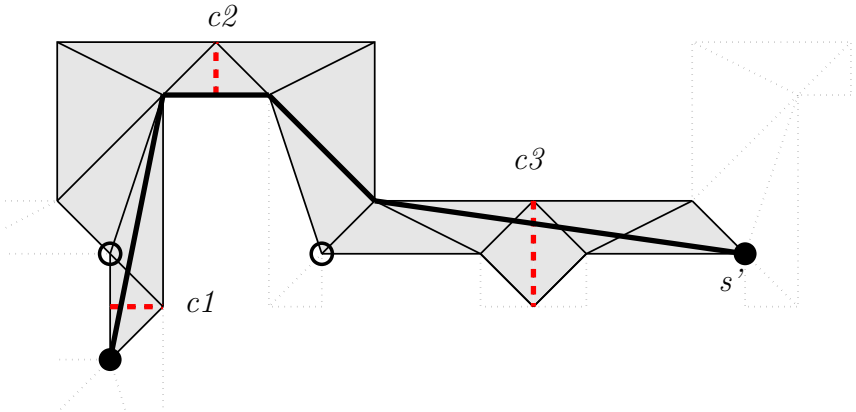
(i) Wesentliche Cuts
O(n)



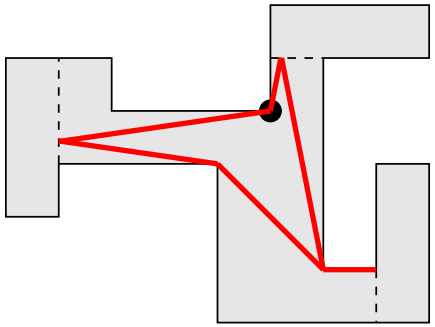
(ii) Abschneiden!



(iii) Triangulation

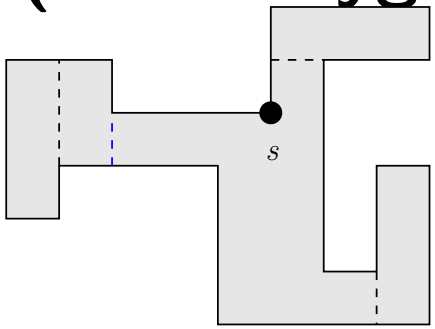


(iv) Spiegeln und Ausrollen!!
(v) Weg berechnen

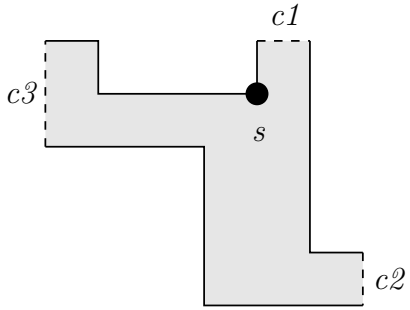


(vi) Zusammenklappen!

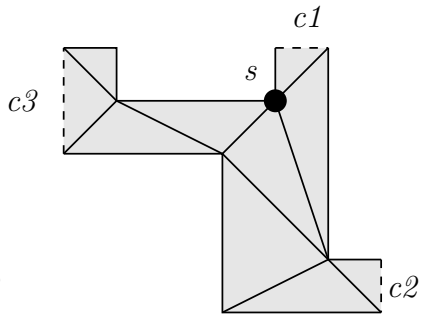
SWR (RW Polygon) Alg. 1.9/Th. 1.27: $O(n)$



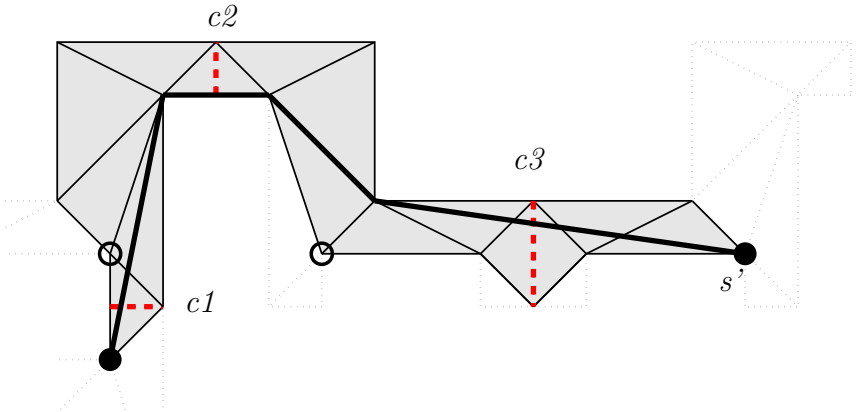
(i) Wesentliche Cuts
O(n)



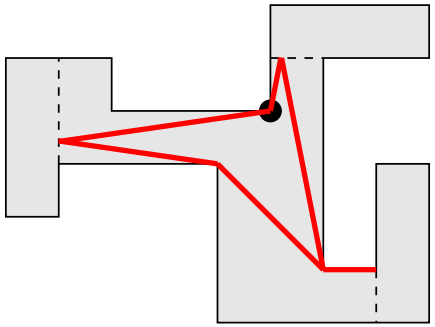
(ii) Abschneiden!
O(n)



(iii) Triangulation

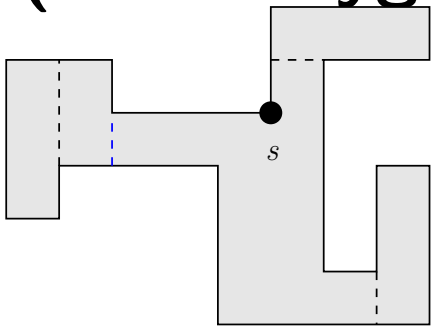


(iv) Spiegeln und Ausrollen!!
(v) Weg berechnen

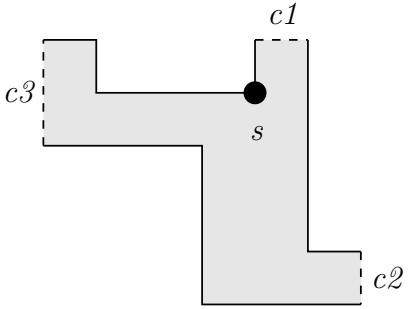


(vi) Zusammenklappen!

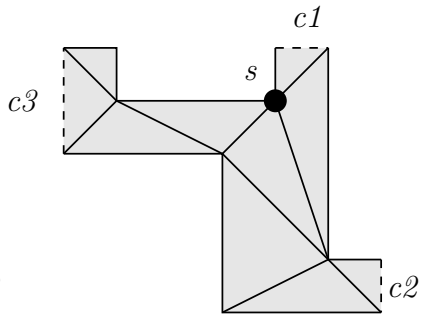
SWR (RW Polygon) Alg. 1.9/Th. 1.27: $O(n)$



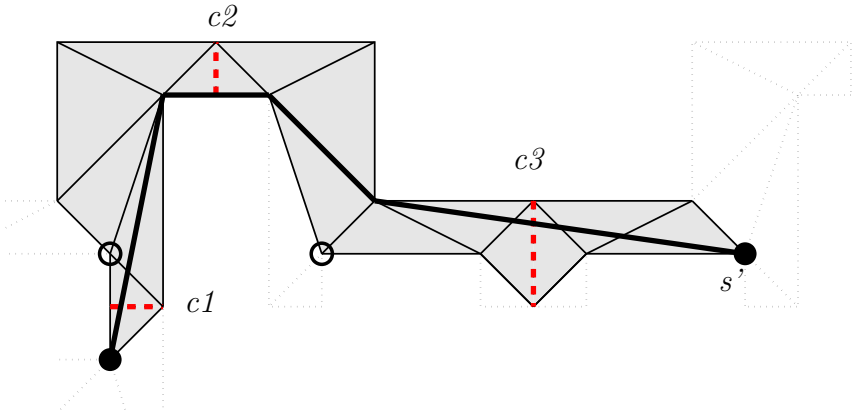
(i) Wesentliche Cuts
O(n)



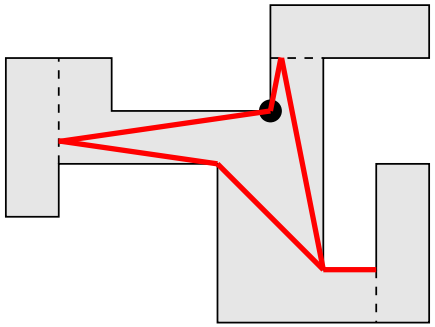
(ii) Abschneiden!
O(n)



(iii) Triangulation
O(n)

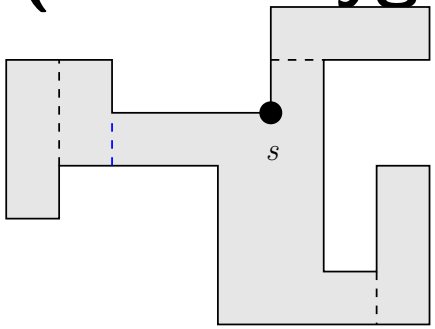


(iv) Spiegeln und Ausrollen!!
(v) Weg berechnen



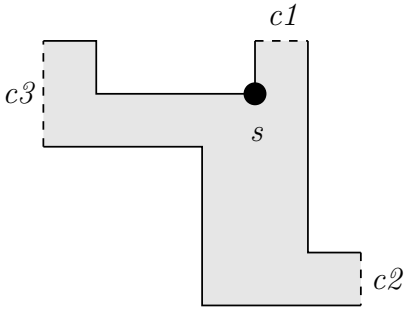
(vi) Zusammenklappen!

SWR (RW Polygon) Alg. 1.9/Th. 1.27: $O(n)$



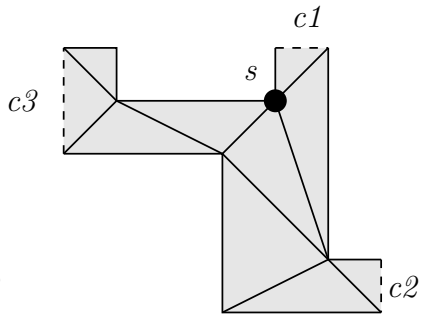
(i) Wesentliche Cuts

O(n)



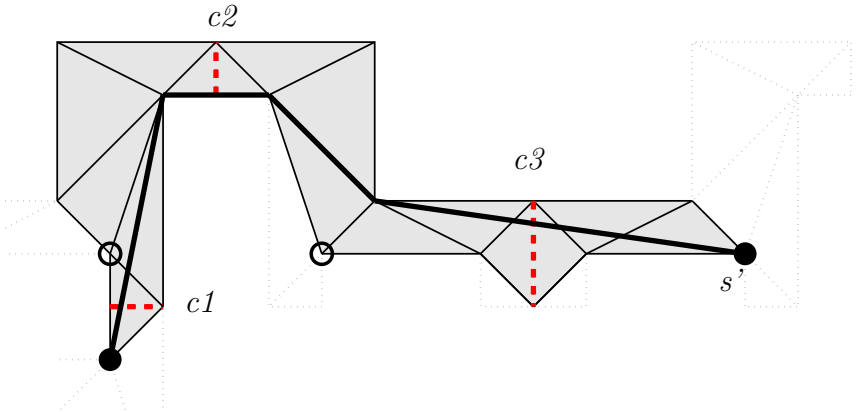
(ii) Abschneiden!

O(n)



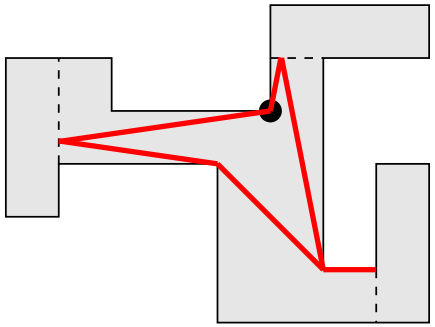
(iii) Triangulation

O(n)



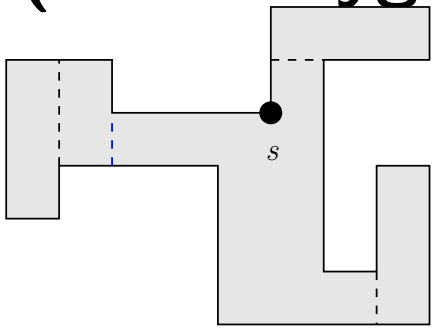
(iv) Spiegeln und Ausrollen!!
(v) Weg berechnen

O(n)



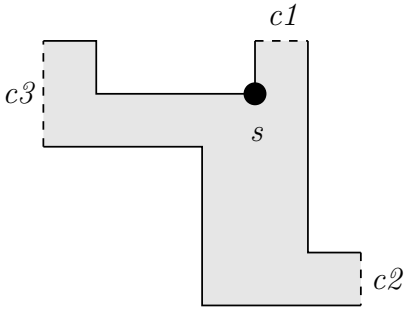
(vi) Zusammenklappen!

SWR (RW Polygon) Alg. 1.9/Th. 1.27: $O(n)$



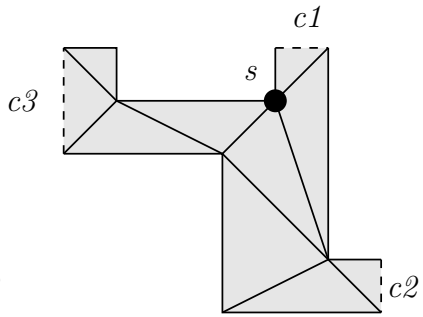
(i) Wesentliche Cuts

O(n)



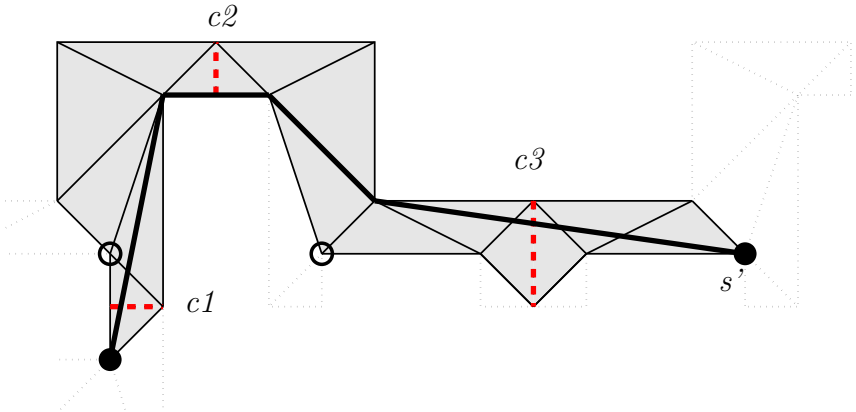
(ii) Abschneiden!

O(n)



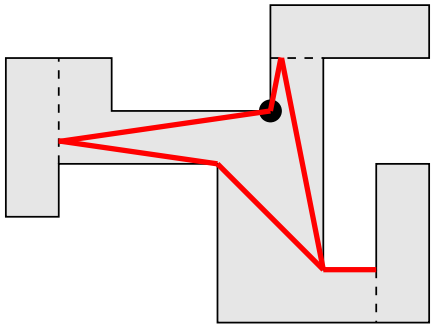
(iii) Triangulation

O(n)



(iv) Spiegeln und Ausrollen!!
(v) Weg berechnen

O(n)



(vi) Zusammenklappen!

O(n)

SWR (Allgemeiner Fall)

SWR (Allgemeiner Fall)

- Corner Problem!!

SWR (Allgemeiner Fall)

- Corner Problem!!
- Def. 1.28 Folge wes. Cuts, die sich sukzessive schneiden

SWR (Allgemeiner Fall)

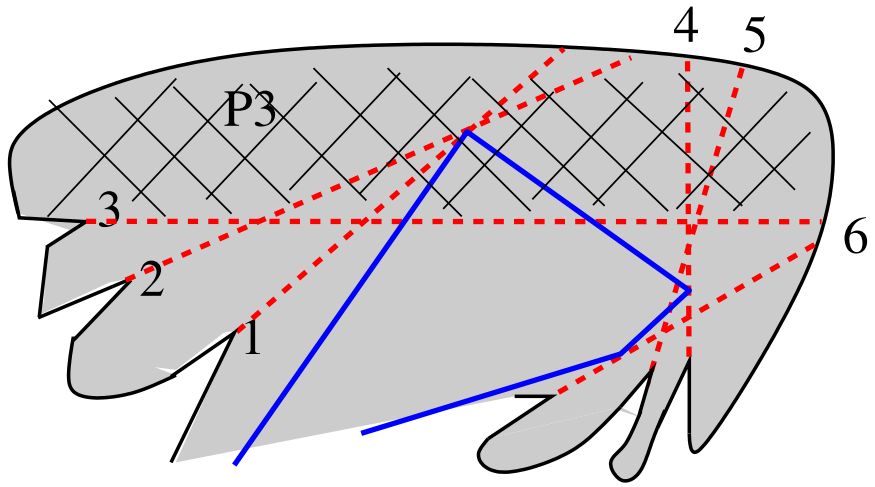
- Corner Problem!!
- Def. 1.28 Folge wes. Cuts, die sich sukzessive schneiden
- Müssen nicht gemäß Ordnung besucht werden

SWR (Allgemeiner Fall)

- Corner Problem!!
- Def. 1.28 Folge wes. Cuts, die sich sukzessive schneiden
- Müssen nicht gemäß Ordnung besucht werden
- Aber die zugehörigen P_{c_i}!!!

SWR (Allgemeiner Fall)

- Corner Problem!!
- Def. 1.28 Folge wes. Cuts, die sich sukzessive schneiden
- Müssen nicht gemäß Ordnung besucht werden
- Aber die zugehörigen P_{c_i}!!!



Lemma 1.29

Lemma 1.29

Die SWR besucht die Corner gemäß der Ordnung entlang des Randes!

Lemma 1.29

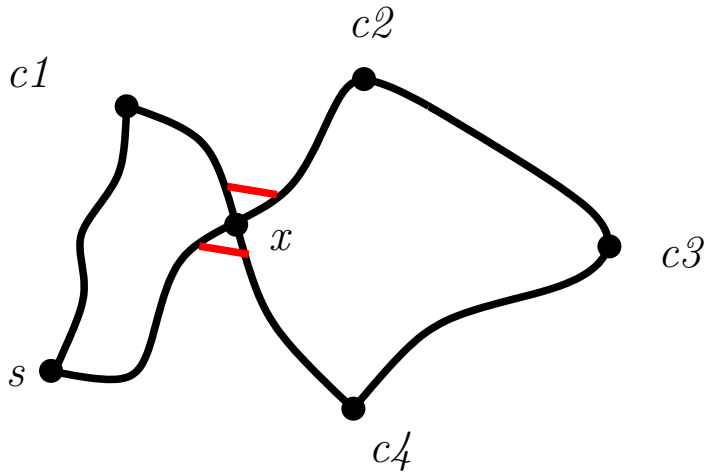
Die SWR besucht die Corner gemäß der Ordnung entlang des Randes!

Beweis: Genauso wie Lemma 1.26!

Lemma 1.29

Die SWR besucht die Corner gemäß der Ordnung entlang des Randes!

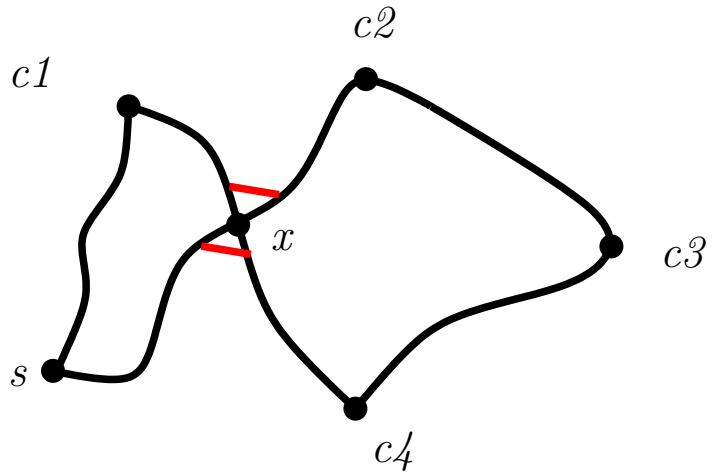
Beweis: Genauso wie Lemma 1.26! Anpassungen im Corner!



Lemma 1.29

Die SWR besucht die Corner gemäß der Ordnung entlang des Randes!

Beweis: Genauso wie Lemma 1.26! Anpassungen im Corner!



Anpassungen innerhalb der Corner: Fehleranfällig!!

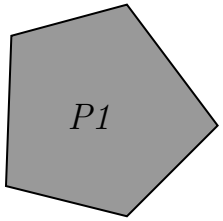
Touring a sequence of polygons (TPP)

Touring a sequence of polygons (TPP)

- Sequenz konvexer Polygone

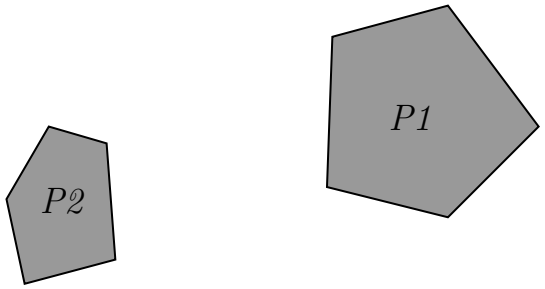
Touring a sequence of polygons (TPP)

- Sequenz konvexer Polygone



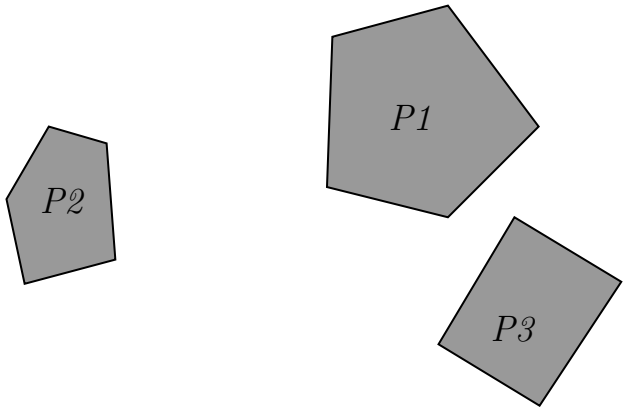
Touring a sequence of polygons (TPP)

- Sequenz konvexer Polygone



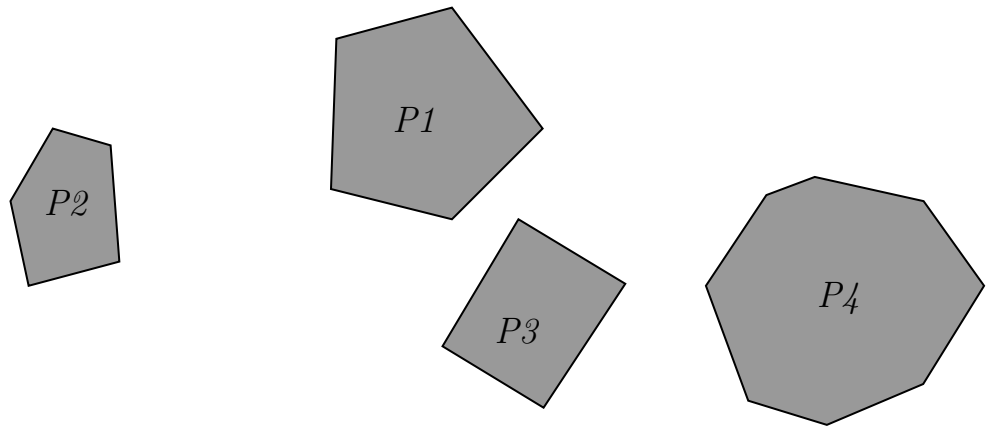
Touring a sequence of polygons (TPP)

- Sequenz konvexer Polygone



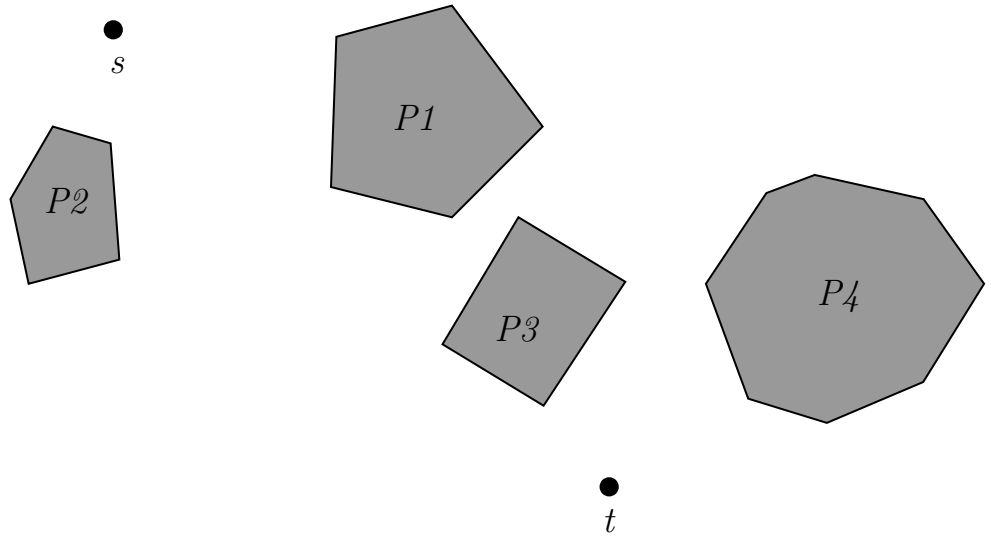
Touring a sequence of polygons (TPP)

- Sequenz konvexer Polygone



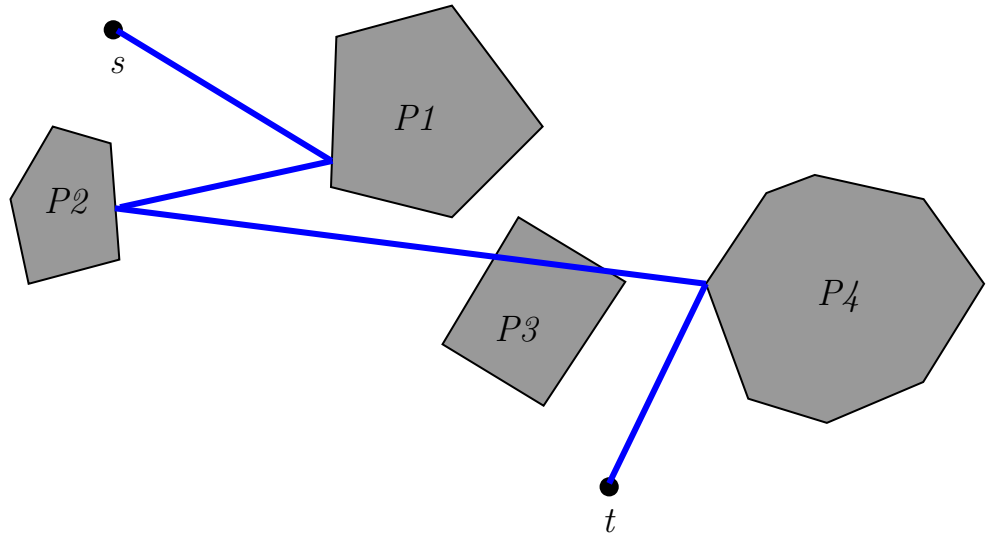
Touring a sequence of polygons (TPP)

- Sequenz konvexer Polygone
- Start s und Ziel t



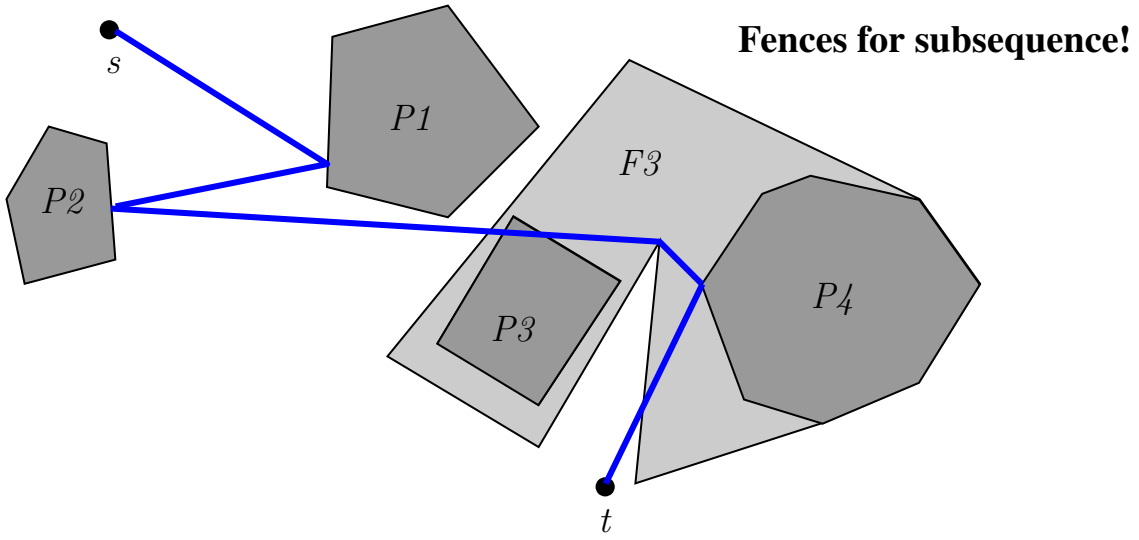
Touring a sequence of polygons (TPP)

- Sequenz konvexer Polygone
- Start s und Ziel t
- Besuche Polygone nach geg. Reihenfolge, Shortest Path



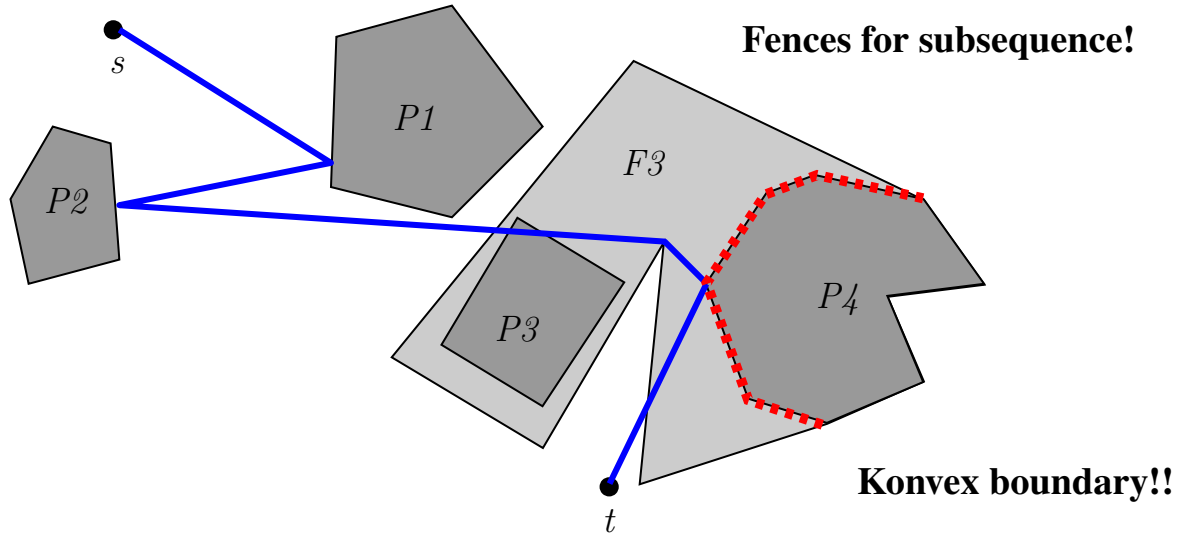
Touring a sequence of polygons (TPP)

- Sequenz konvexer Polygone
- Start s und Ziel t
- Besuche Polygone nach geg. Reihenfolge, Shortest Path



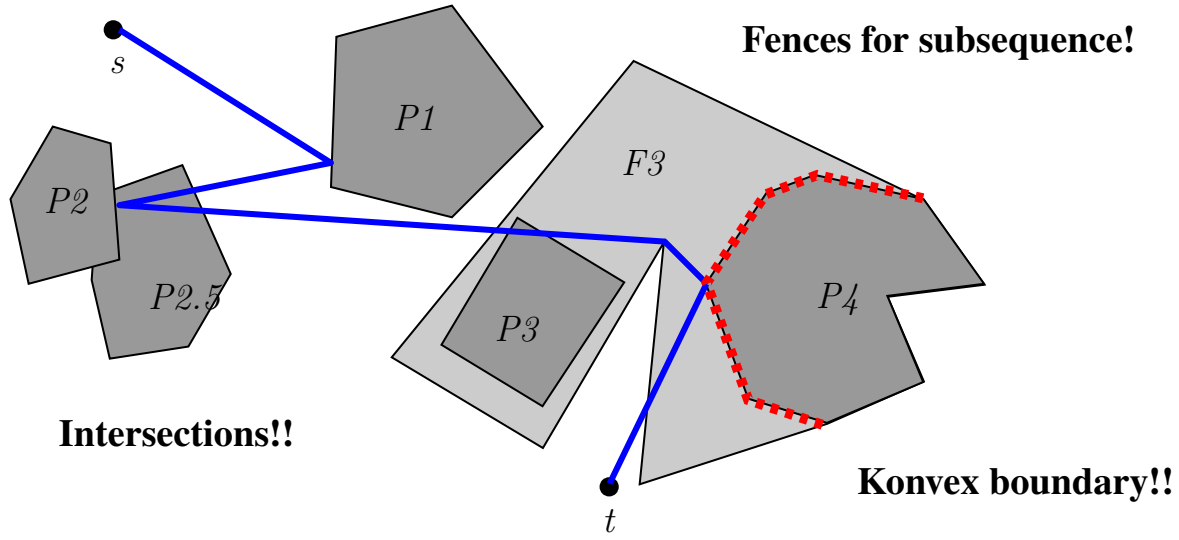
Touring a sequence of polygons (TPP)

- Sequenz konvexer Polygone
- Start s und Ziel t
- Besuche Polygone nach geg. Reihenfolge, Shortest Path



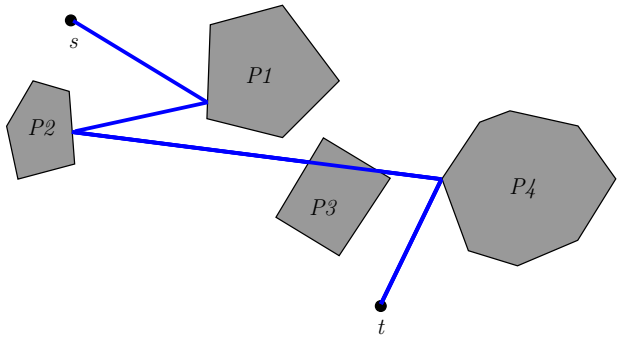
Touring a sequence of polygons (TPP)

- Sequenz konvexer Polygone
- Start s und Ziel t
- Besuche Polygone nach geg. Reihenfolge, Shortest Path

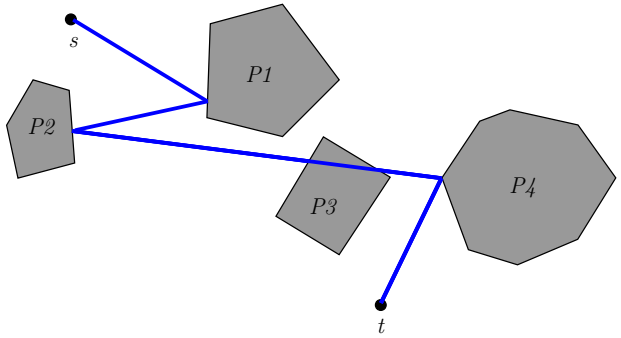


TPP

TPP

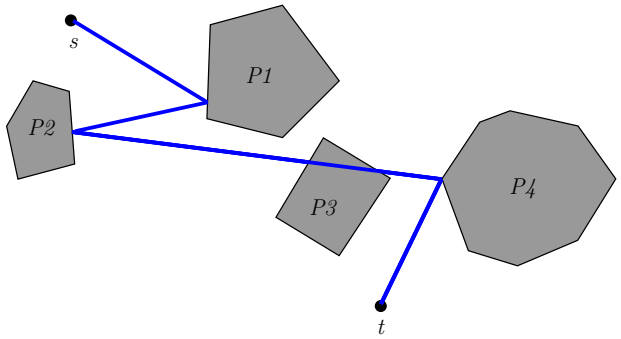


TPP



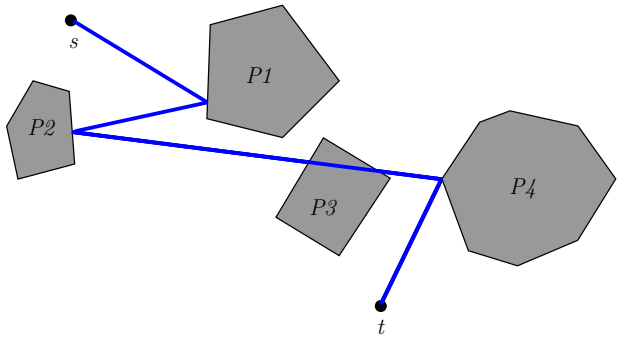
- Einfache Version:
- $O(nk \log \frac{n}{k})$

TPP



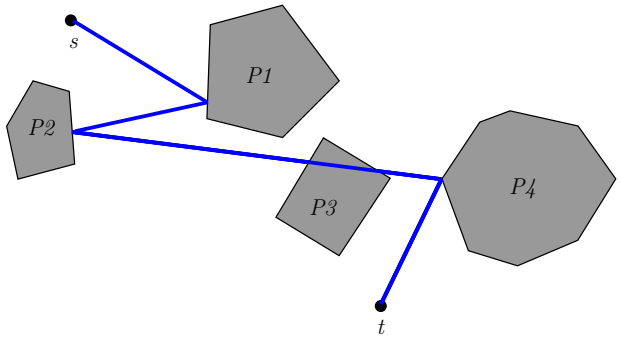
- Einfache Version:
- $O(nk \log \frac{n}{k})$
- Build(Query): $O(nk \log \frac{n}{k})$

TPP



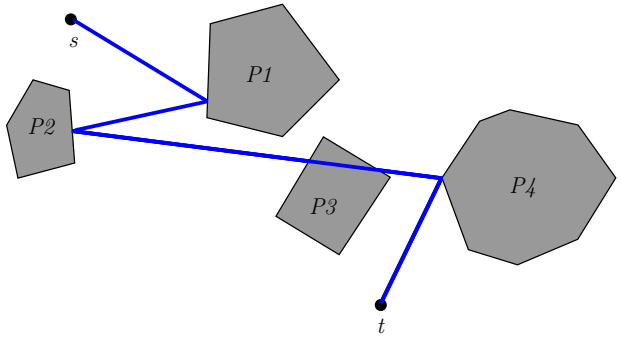
- Einfache Version:
- $O(nk \log \frac{n}{k})$
- Build(Query): $O(nk \log \frac{n}{k})$
- Kompl.: $O(n)$

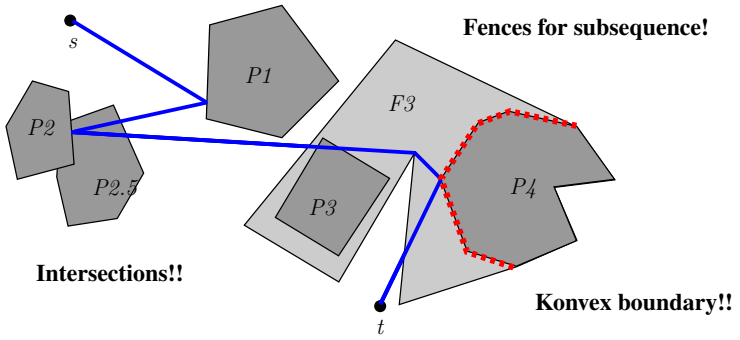
TPP



- Einfache Version:
- $O(nk \log \frac{n}{k})$
- Build(Query): $O(nk \log \frac{n}{k})$
- Kompl.: $O(n)$
- Query (festes s): $O(k \log \frac{n}{k})$

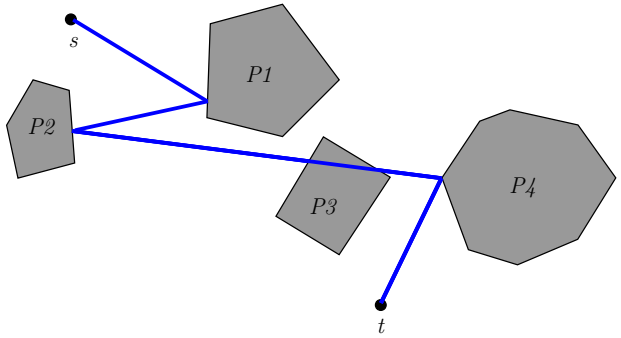
TPP



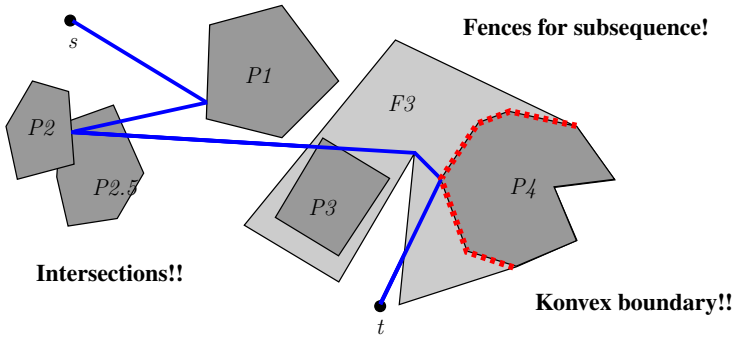


- Einfache Version:
- $O(nk \log \frac{n}{k})$
- Build(Query): $O(nk \log \frac{n}{k})$
- Kompl.: $O(n)$
- Query (festes s): $O(k \log \frac{n}{k})$

TPP

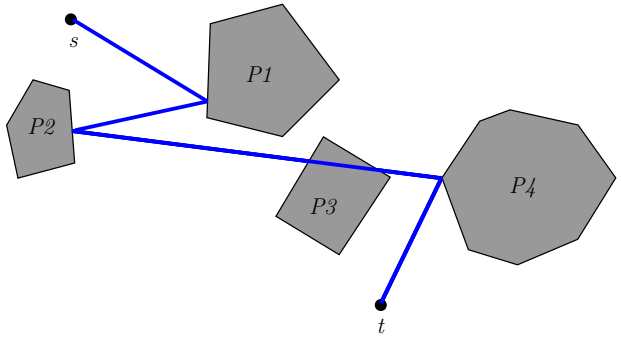


- Einfache Version:
- $O(nk \log \frac{n}{k})$
- Build(Query): $O(nk \log \frac{n}{k})$
- Kompl.: $O(n)$
- Query (festes s): $O(k \log \frac{n}{k})$

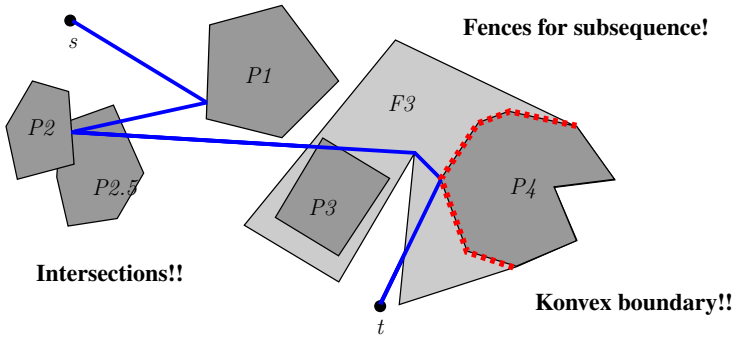


- Allgemeine Version:
- Zäune, Rand konvex, etc.

TPP

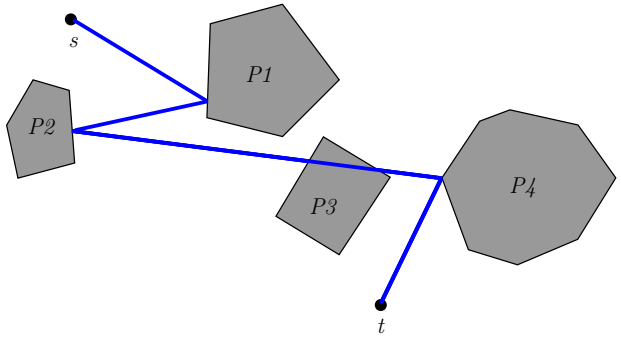


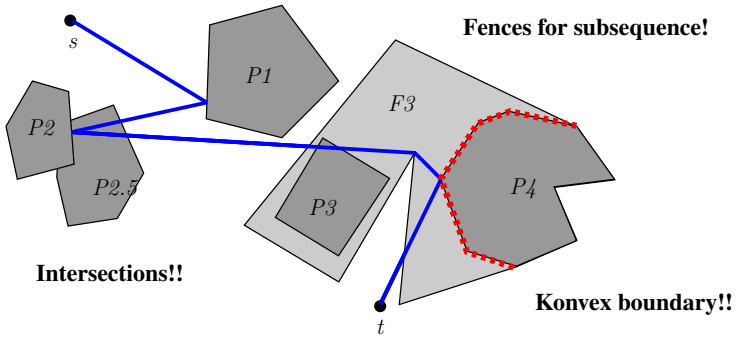
- Einfache Version:
- $O(nk \log \frac{n}{k})$
- Build(Query): $O(nk \log \frac{n}{k})$
- Kompl.: $O(n)$
- Query (festes s): $O(k \log \frac{n}{k})$



- Allgemeine Version:
- Zäune, Rand konvex, etc.
- $O(nk^2 \log n)$

TPP

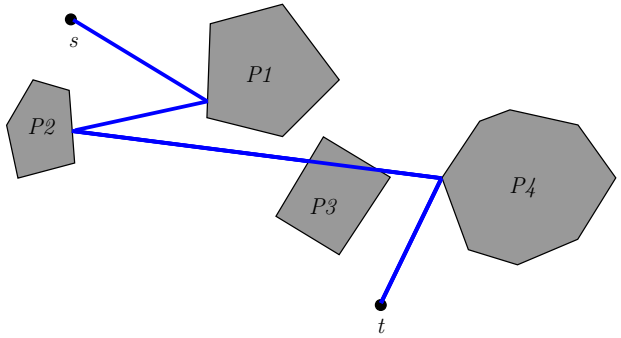


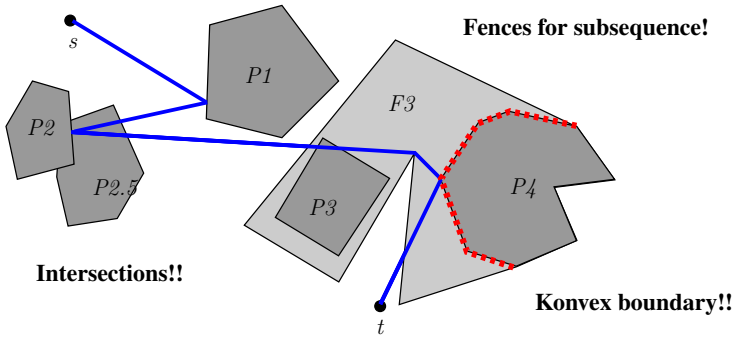


- Einfache Version:
- $O(nk \log \frac{n}{k})$
- Build(Query): $O(nk \log \frac{n}{k})$
- Kompl.: $O(n)$
- Query (festes s): $O(k \log \frac{n}{k})$

- Allgemeine Version:
- Zäune, Rand konvex, etc.
- $O(nk^2 \log n)$
- Build(Query): $O(nk^2 \log n)$

TPP

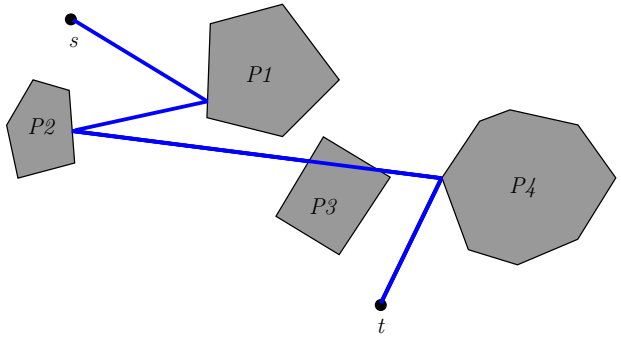


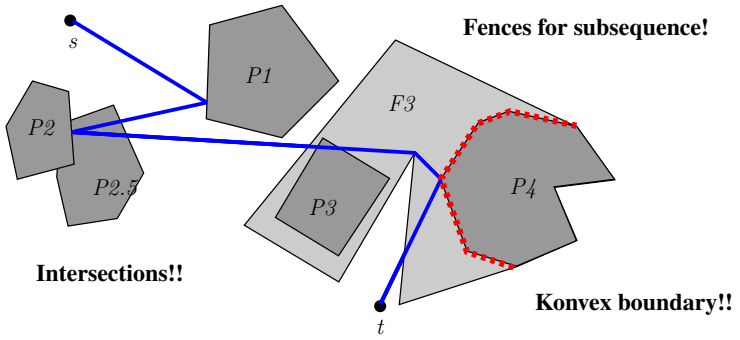


- Einfache Version:
- $O(nk \log \frac{n}{k})$
- Build(Query): $O(nk \log \frac{n}{k})$
- Kompl.: $O(n)$
- Query (festes s): $O(k \log \frac{n}{k})$

- Allgemeine Version:
- Zäune, Rand konvex, etc.
- $O(nk^2 \log n)$
- Build(Query): $O(nk^2 \log n)$
- Kompl.: $O(nk)$

TPP

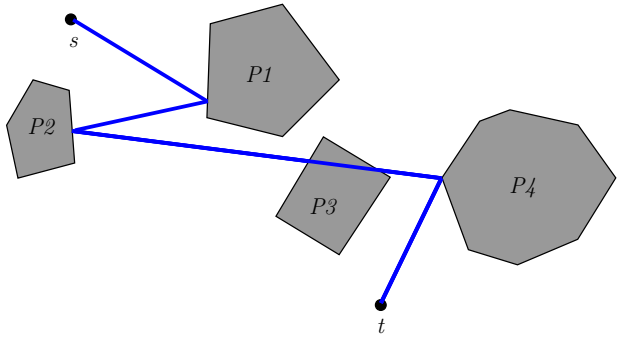


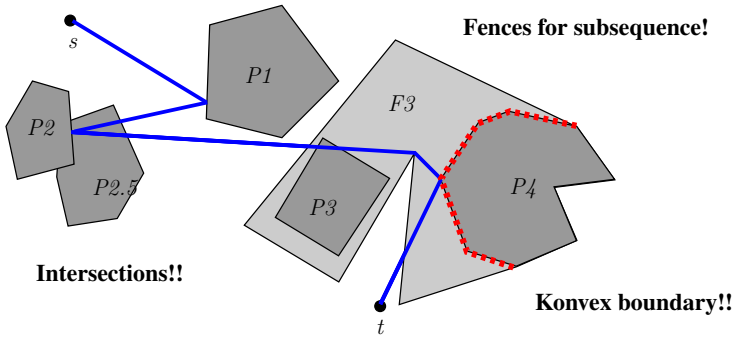


- Einfache Version:
- $O(nk \log \frac{n}{k})$
- Build(Query): $O(nk \log \frac{n}{k})$
- Kompl.: $O(n)$
- Query (festes s): $O(k \log \frac{n}{k})$

- Allgemeine Version:
- Zäune, Rand konvex, etc.
- $O(nk^2 \log n)$
- Build(Query): $O(nk^2 \log n)$
- Kompl.: $O(nk)$
- Query (festes s): $O(k \log n + m)$

TPP





- Einfache Version:
- $O(nk \log \frac{n}{k})$
- Build(Query): $O(nk \log \frac{n}{k})$
- Kompl.: $O(n)$
- Query (festes s): $O(k \log \frac{n}{k})$

- Allgemeine Version:
- Zäune, Rand konvex, etc.
- $O(nk^2 \log n)$
- Build(Query): $O(nk^2 \log n)$
- Kompl.: $O(nk)$
- Query (festes s): $O(k \log n + m)$

Ergebnisse von: Dror, Efrat, Lubiw, Mitchell 2003!!

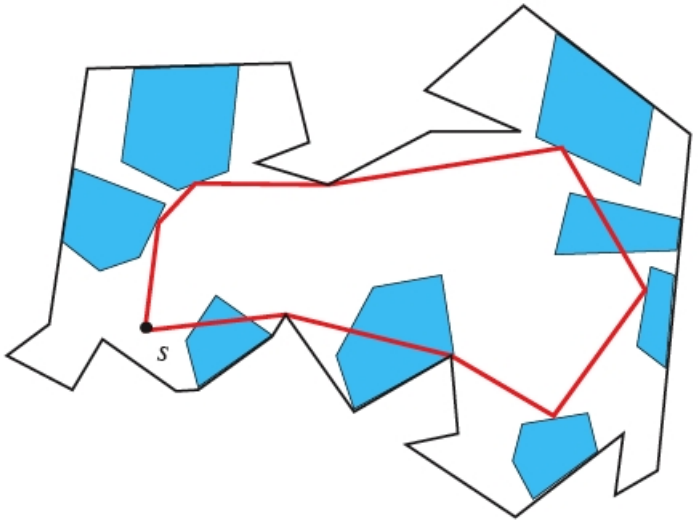
Anwendung: Safari-Problem

Anwendung: Safari-Problem

Kürzester Weg,
innerhalb eines Polygones, der eine
Menge von schnittfreien Polygonen
besucht.

Anwendung: Safari-Problem

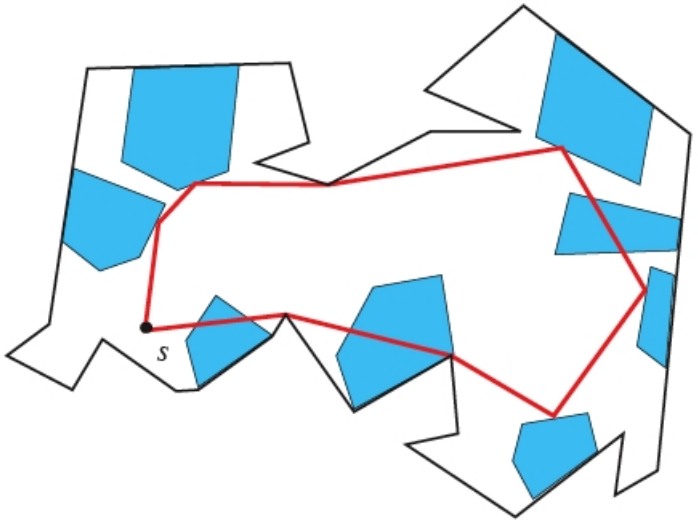
Kürzester Weg,
innerhalb eines Polygones, der eine
Menge von schnittfreien Polygonen
besucht.



Ordnung entlang des Randes!

Anwendung: Safari-Problem

Kürzester Weg,
innerhalb eines Polygones, der eine
Menge von schnittfreien Polygonen
besucht.



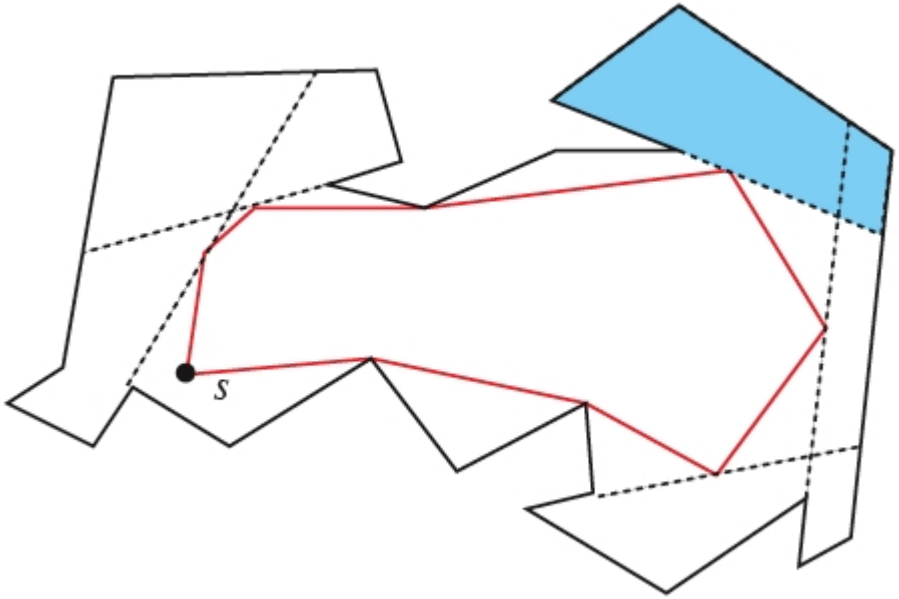
- $O(n^2)$ '92
- $O(n^3)$ '94
- $O(n^3)$ Tan Hirata '01
- $O(n^2 \log n)$ jetzt! Anpassen!!

Ordnung entlang des Randes!

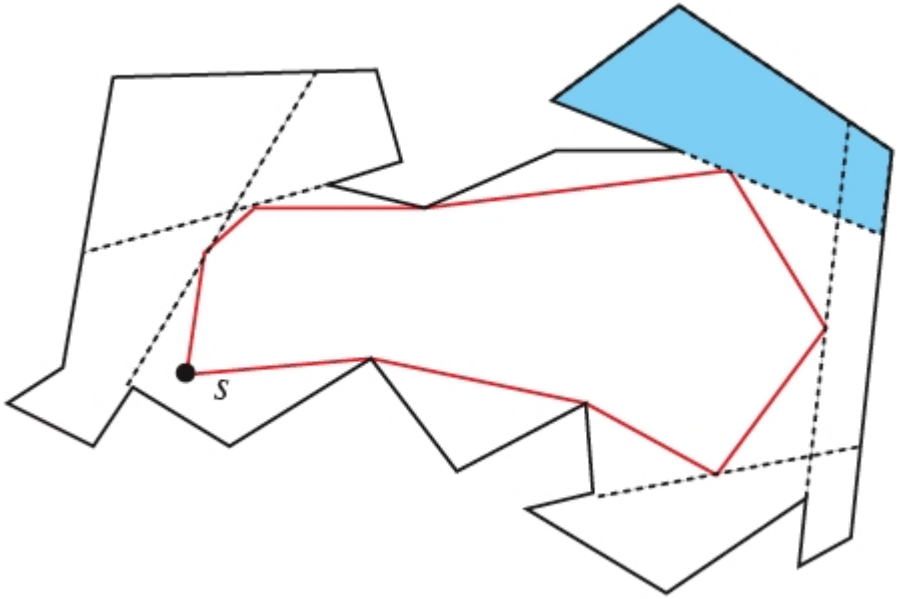
Anwendung: SWR

Anwendung: SWR

Wesentlichen Teile!



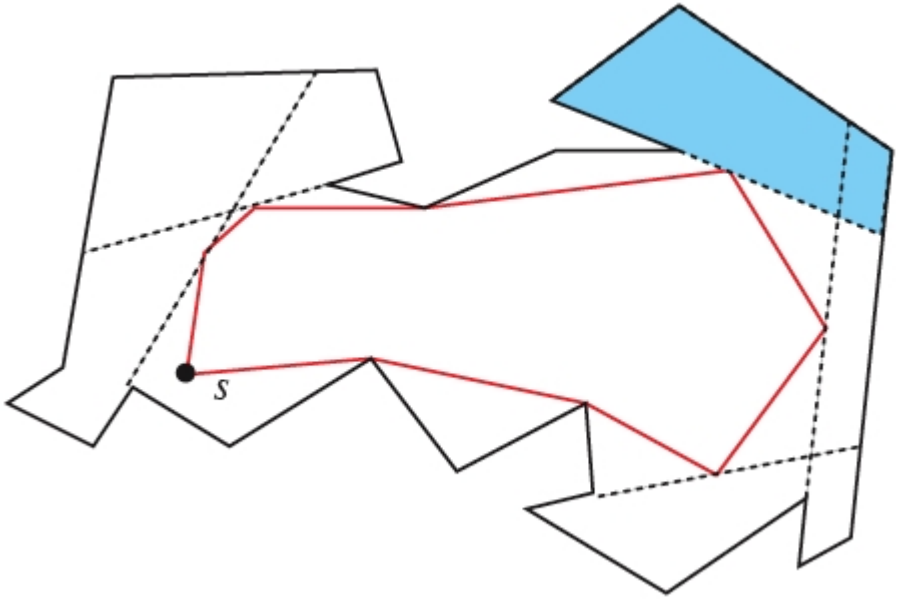
Anwendung: SWR



Wesentlichen Teile!

Ordnung entlang des Randes!

Anwendung: SWR

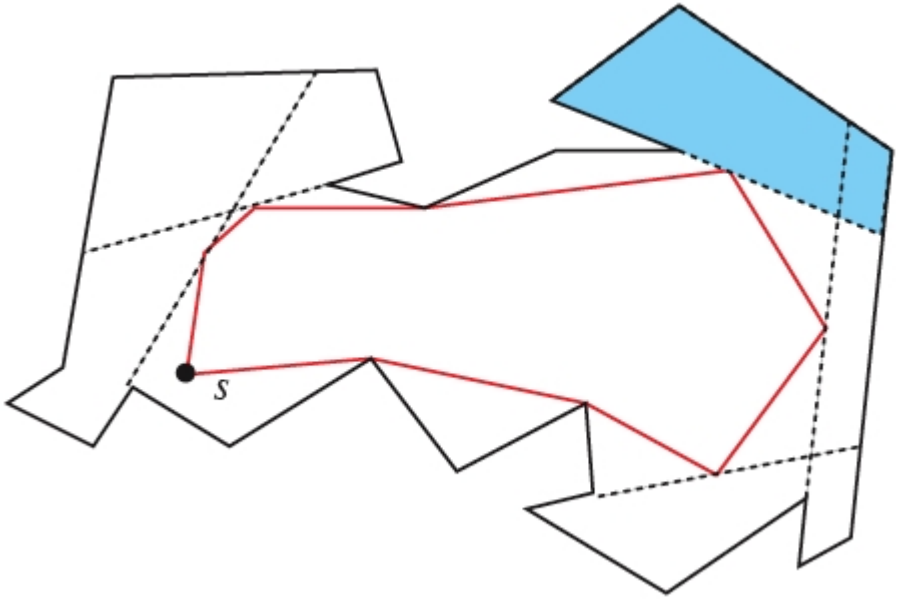


Wesentlichen Teile!

Ordnung entlang des Randes!

Ein gemeinsamer Zaun! Schnitte!

Anwendung: SWR



Wesentlichen Teile!

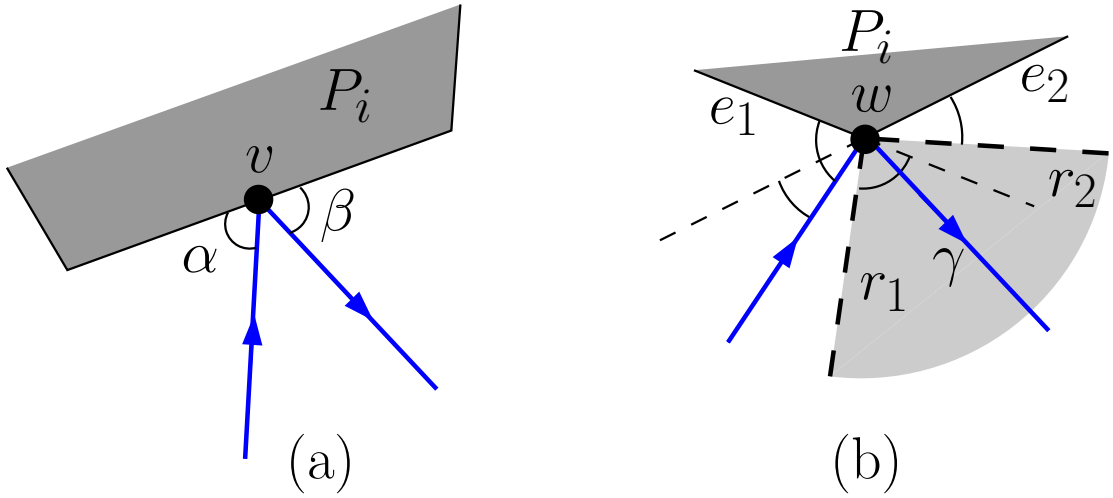
Ordnung entlang des Randes!

Ein gemeinsamer Zaun! Schnitte!

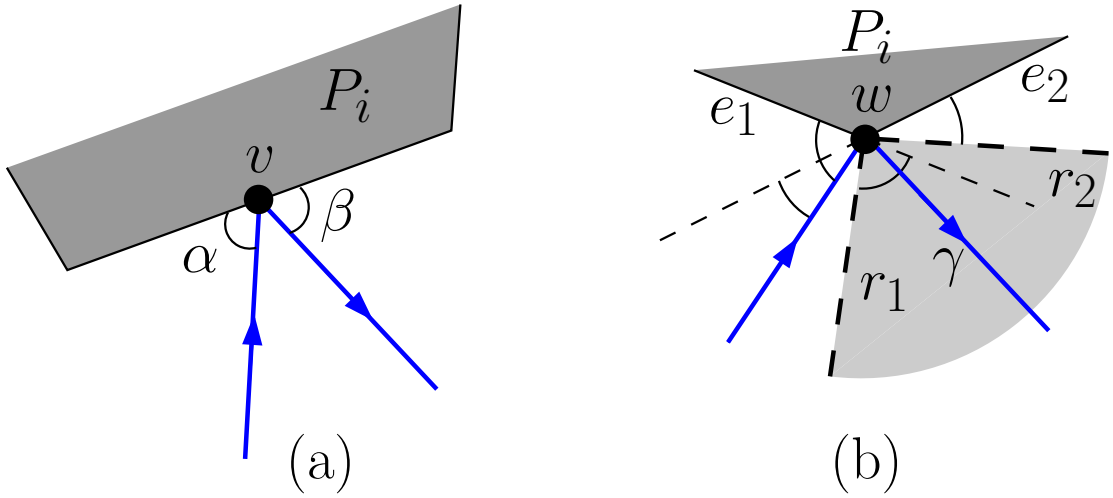
- $O(n^4)$ '91
- $O(n^4)$ Tan et al. '99
- $O(n^3 \log n)$ jetzt!

Lokale Eigenschaften: Lemma 1.31(i)

Lokale Eigenschaften: Lemma 1.31(i)

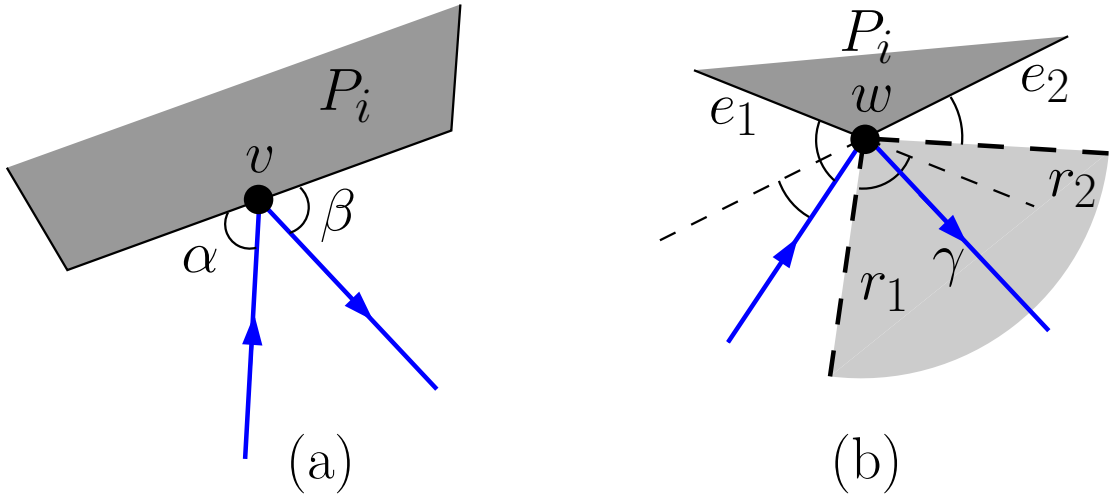


Lokale Eigenschaften: Lemma 1.31(i)



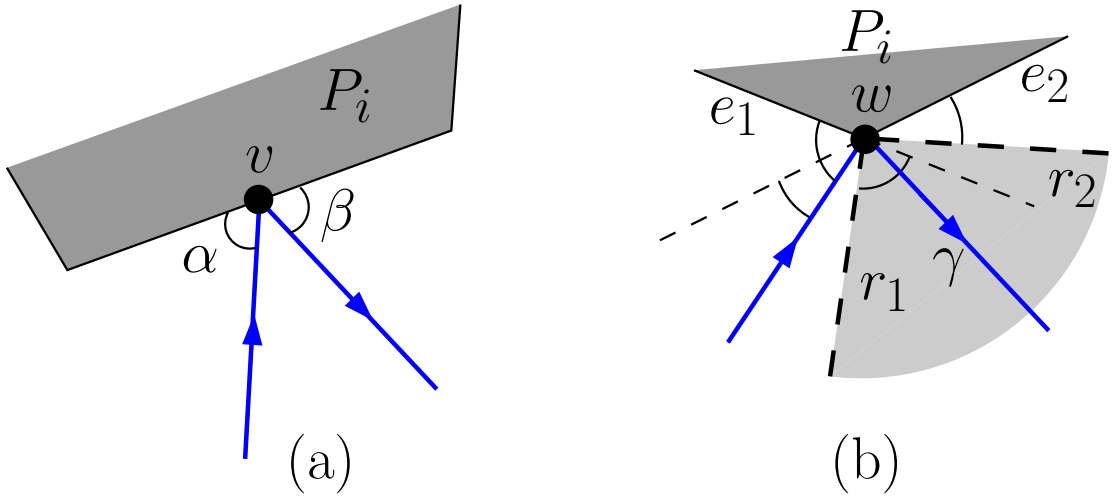
- (a) Reflektion an Kante e:

Lokale Eigenschaften: Lemma 1.31(i)



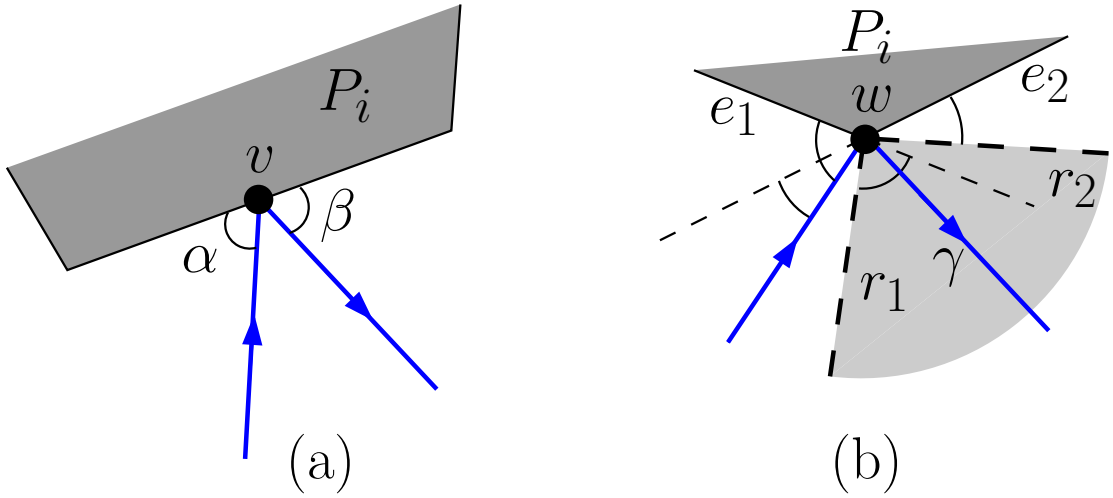
- (a) Reflektion an Kante e: $\alpha = \beta$

Lokale Eigenschaften: Lemma 1.31(i)



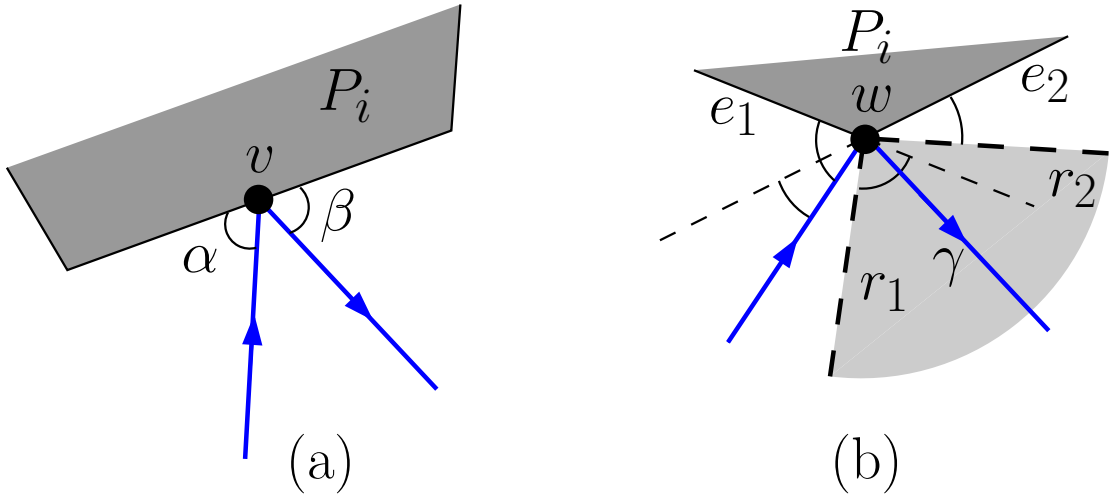
- (a) Reflektion an Kante e: $\alpha = \beta$
- (b) Reflektion an Knoten w:

Lokale Eigenschaften: Lemma 1.31(i)



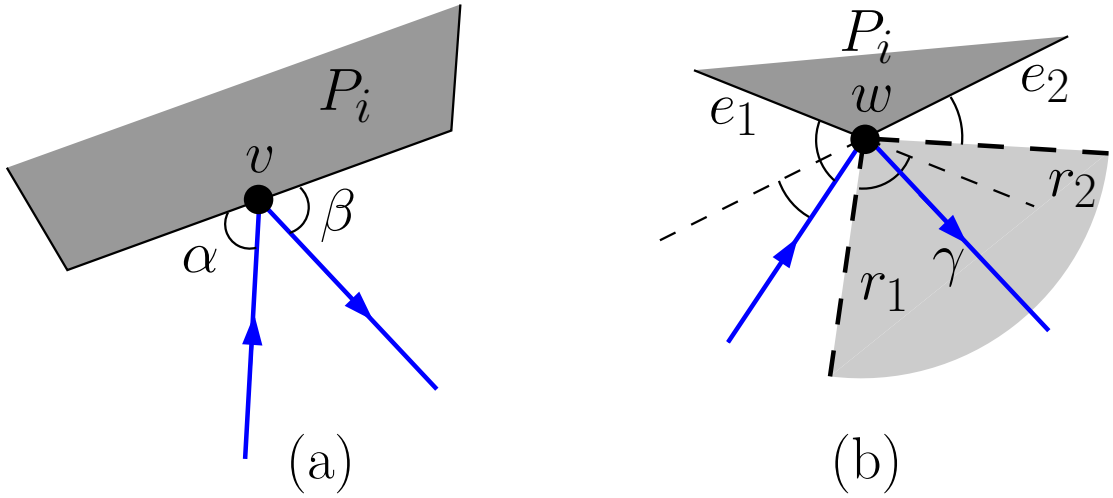
- (a) Reflektion an Kante e: $\alpha = \beta$
- (b) Reflektion an Knoten w: Innerhalb Winkelbereich γ

Lokale Eigenschaften: Lemma 1.31(i)



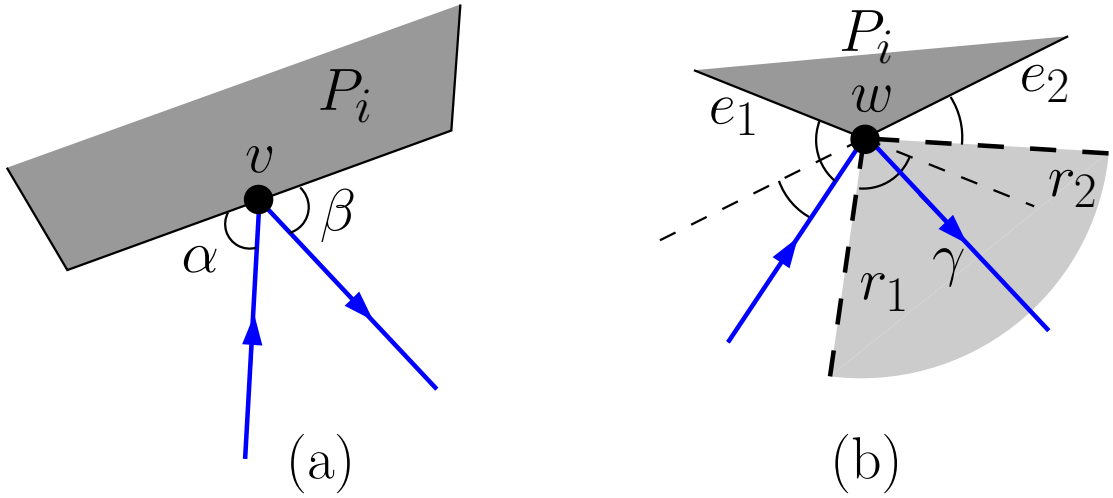
- (a) Reflektion an Kante e: $\alpha = \beta$
- (b) Reflektion an Knoten w: Innerhalb Winkelbereich γ
- Sonst nicht optimal

Lokale Eigenschaften: Lemma 1.31(i)



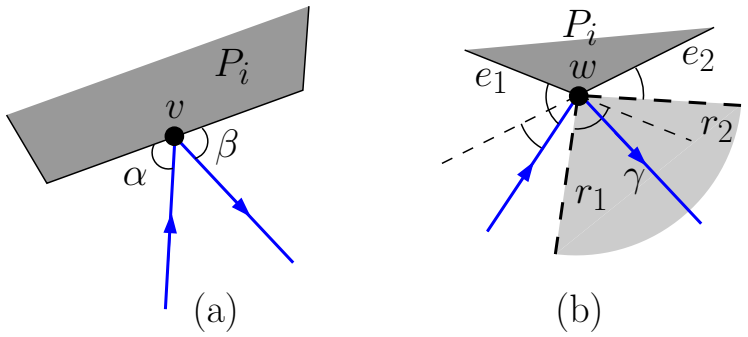
- (a) Reflektion an Kante e: $\alpha = \beta$
- (b) Reflektion an Knoten w: Innerhalb Winkelbereich γ
- Sonst nicht optimal
- Lemma 1.31: Local optimality $\Rightarrow$ Global optimality

Lokale Eigenschaften: Lemma 1.31(i)



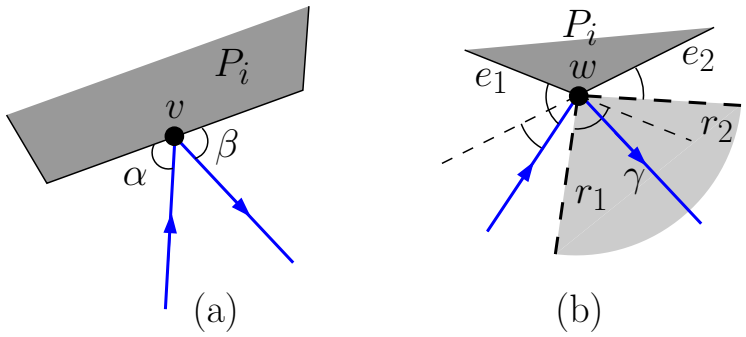
- (a) Reflektion an Kante e: $\alpha = \beta$
- (b) Reflektion an Knoten w: Innerhalb Winkelbereich γ
- Sonst nicht optimal
- Lemma 1.31: Local optimality $\Rightarrow$ Global optimality
- Aufgabe: Berechne sukzessive lokal optimalen Pfad

Lemma 1.31 (ii) und (iii)



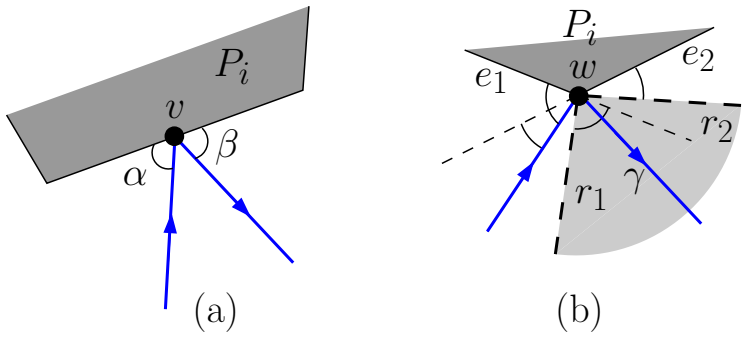
Lemma 1.31 (ii) und (iii)

- Bezeichnung $\pi_i(p)$:



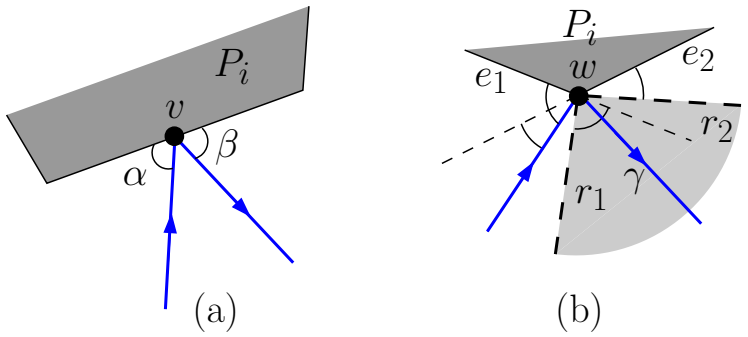
Lemma 1.31 (ii) und (iii)

- Bezeichnung $\pi_i(p)$: lokal opt. Weg von s über $P_1, \ldots, P_i$ bis p



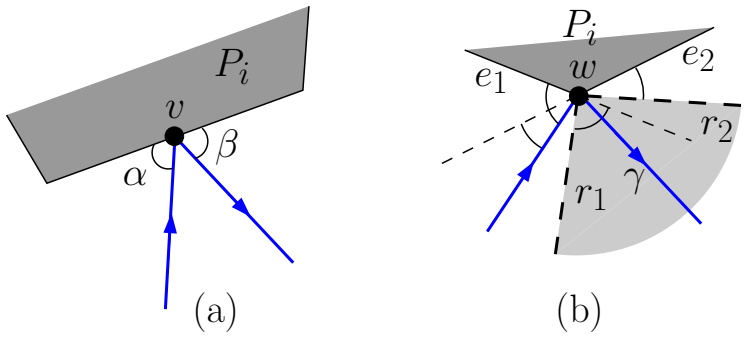
Lemma 1.31 (ii) und (iii)

- Bezeichnung $\pi_i(p)$: lokal opt. Weg von s über $P_1, \ldots, P_i$ bis p
- Lokal optimal: Erfüllt überall die Bedingungen (a) oder (b)



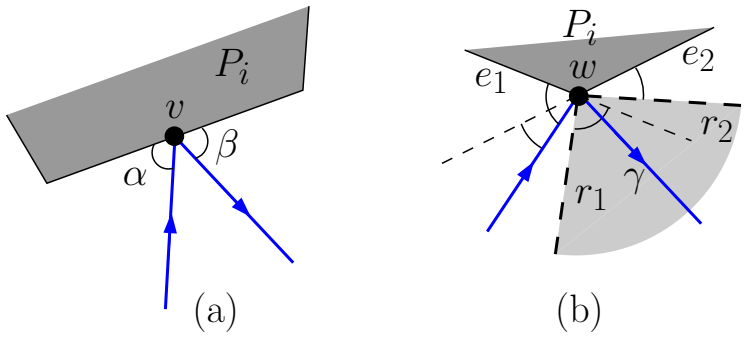
Lemma 1.31 (ii) und (iii)

- Bezeichnung $\pi_i(p)$: lokal opt. Weg von s über $P_1, \ldots, P_i$ bis p
- Lokal optimal: Erfüllt überall die Bedingungen (a) oder (b)
- Über Ecken und Kanten, lokal nicht verkürzbar



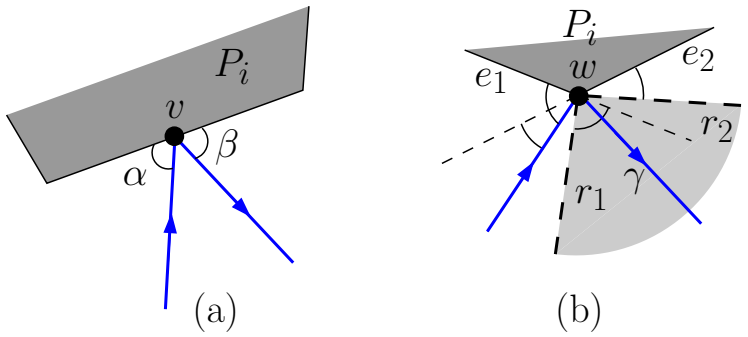
Lemma 1.31 (ii) und (iii)

- Bezeichnung $\pi_i(p)$: lokal opt. Weg von s über $P_1, \ldots, P_i$ bis p
- Lokal optimal: Erfüllt überall die Bedingungen (a) oder (b)
- Über Ecken und Kanten, lokal nicht verkürzbar
- (ii): $\pi_i(p)$ ist stets eindeutig!



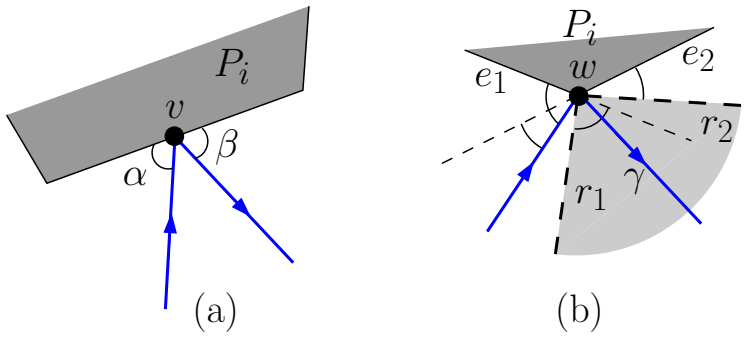
Lemma 1.31 (ii) und (iii)

- Bezeichnung $\pi_i(p)$: lokal opt. Weg von s über $P_1, \ldots, P_i$ bis p
- Lokal optimal: Erfüllt überall die Bedingungen (a) oder (b)
- Über Ecken und Kanten, lokal nicht verkürzbar
- (ii): $\pi_i(p)$ ist stets eindeutig! (Konstruktiv!)



Lemma 1.31 (ii) und (iii)

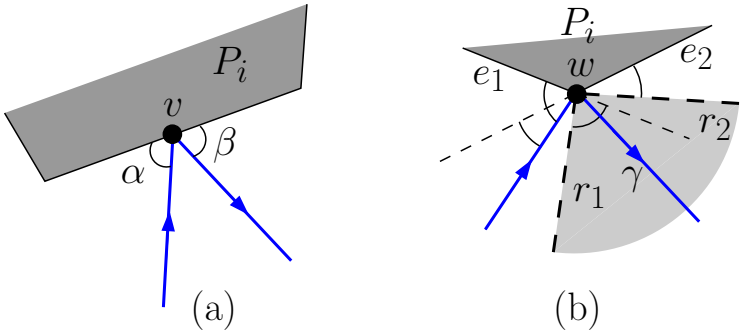
- Bezeichnung $\pi_i(p)$: lokal opt. Weg von s über $P_1, \ldots, P_i$ bis p
- Lokal optimal: Erfüllt überall die Bedingungen (a) oder (b)
- Über Ecken und Kanten, lokal nicht verkürzbar
- (ii): $\pi_i(p)$ ist stets eindeutig! (Konstruktiv!)
- (iii): $\pi_i(p)$ ist auch global optimal!



Lemma 1.31 (ii) und (iii)

- Bezeichnung $\pi_i(p)$: lokal opt. Weg von s über $P_1, \ldots, P_i$ bis p
- Lokal optimal: Erfüllt überall die Bedingungen (a) oder (b)
- Über Ecken und Kanten, lokal nicht verkürzbar
- (ii): $\pi_i(p)$ ist stets eindeutig! (Konstruktiv!)
- (iii): $\pi_i(p)$ ist auch global optimal!

Beweis (iii): Jeder global optimale Weg muss lokal optimal sein!



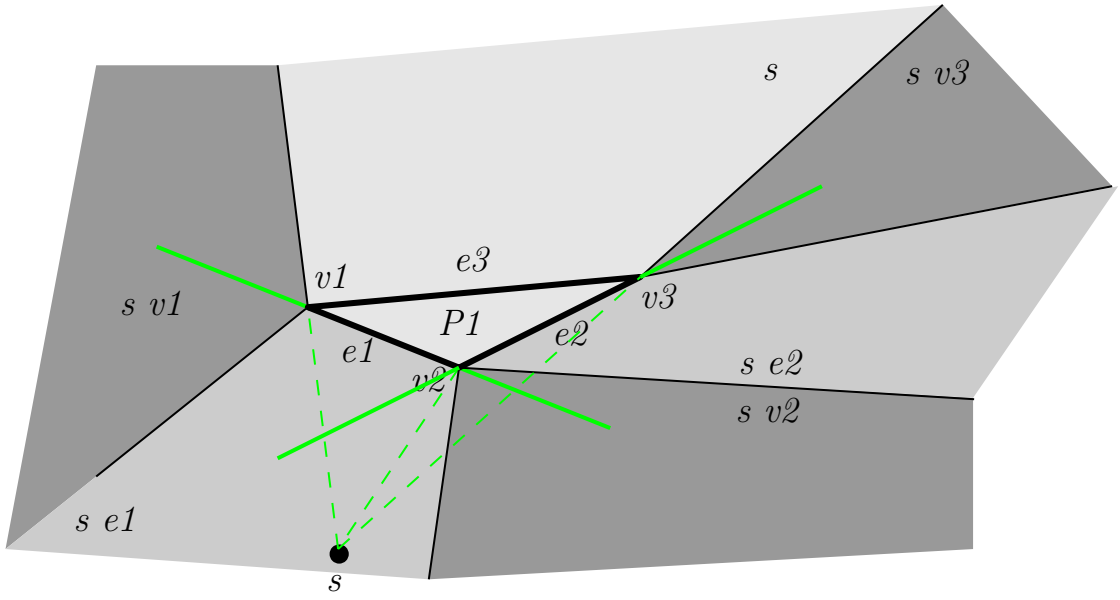
Full comb. shortest path map: Beispiel

Full comb. shortest path map: Beispiel

- Fixiere den Startpunkt s

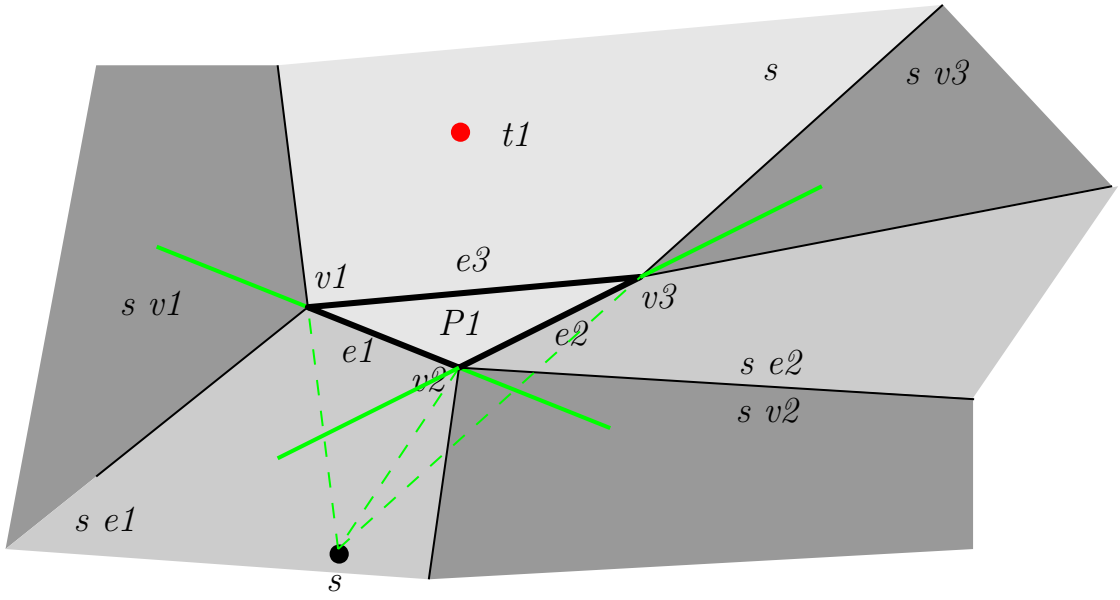
Full comb. shortest path map: Beispiel

- Fixiere den Startpunkt s
- Teile die Ebene in Regionen ein:
- *Kombinatorisch* gleiche Kürzeste Wege



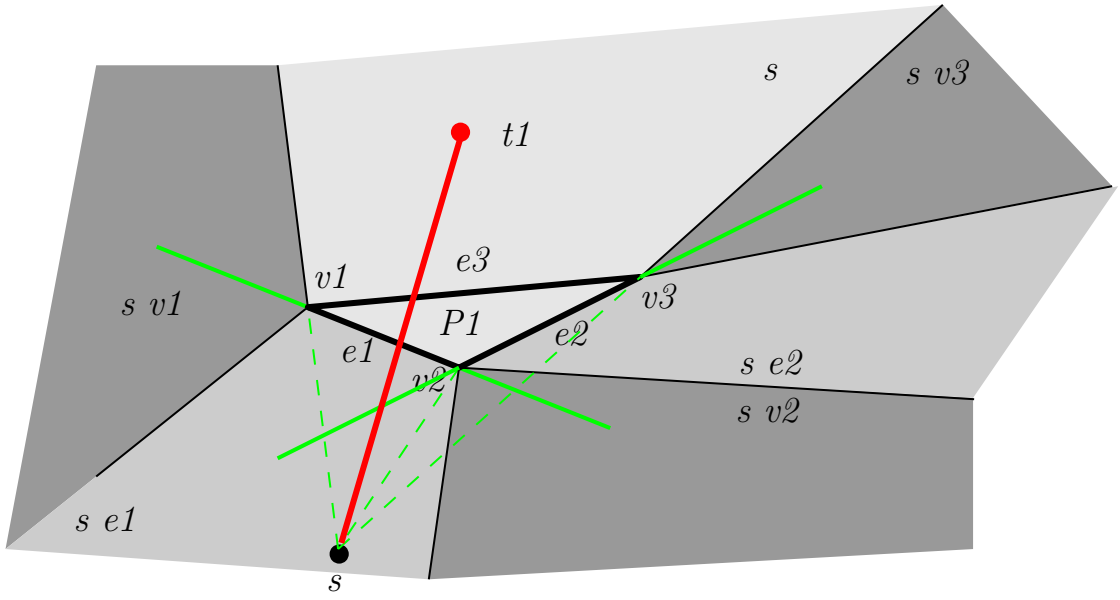
Full comb. shortest path map: Beispiel

- Fixiere den Startpunkt s
- Teile die Ebene in Regionen ein:
- *Kombinatorisch* gleiche Kürzeste Wege



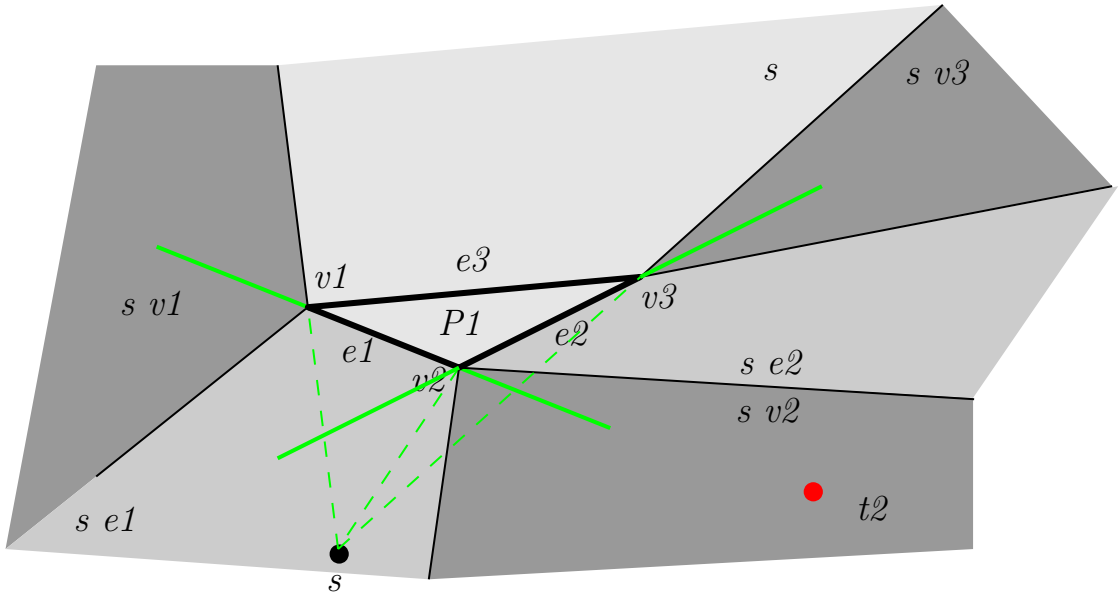
Full comb. shortest path map: Beispiel

- Fixiere den Startpunkt s
- Teile die Ebene in Regionen ein:
- *Kombinatorisch* gleiche Kürzeste Wege



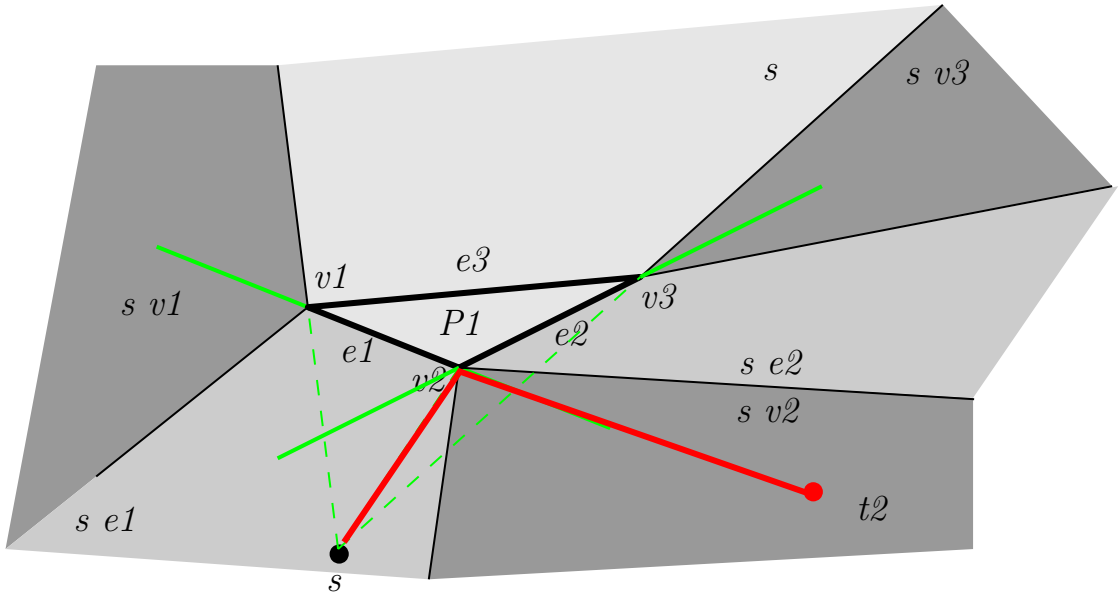
Full comb. shortest path map: Beispiel

- Fixiere den Startpunkt s
- Teile die Ebene in Regionen ein:
- *Kombinatorisch* gleiche Kürzeste Wege



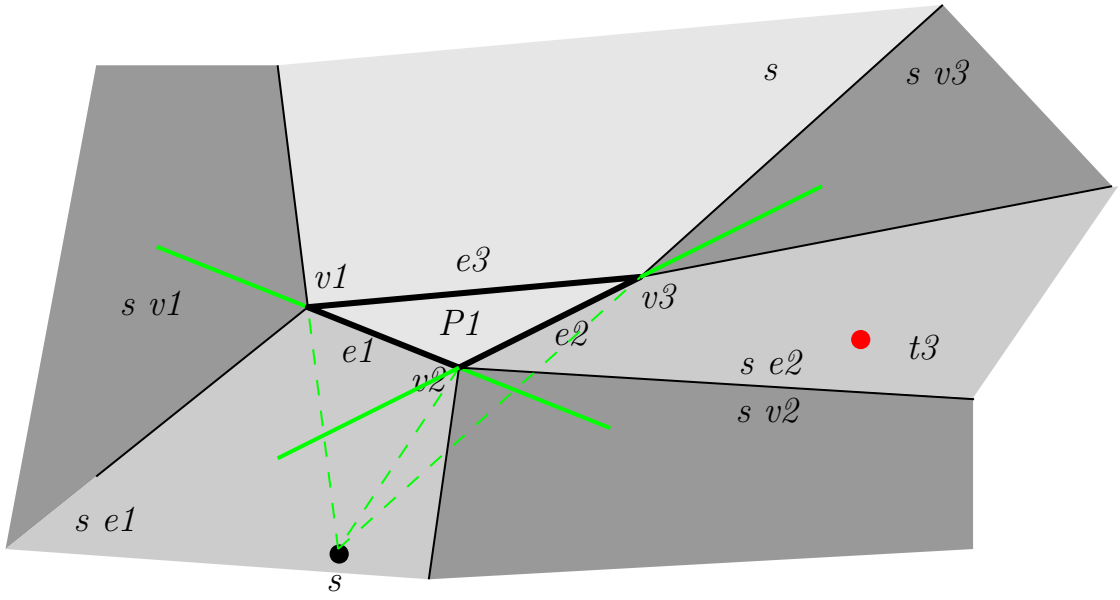
Full comb. shortest path map: Beispiel

- Fixiere den Startpunkt s
- Teile die Ebene in Regionen ein:
- *Kombinatorisch* gleiche Kürzeste Wege



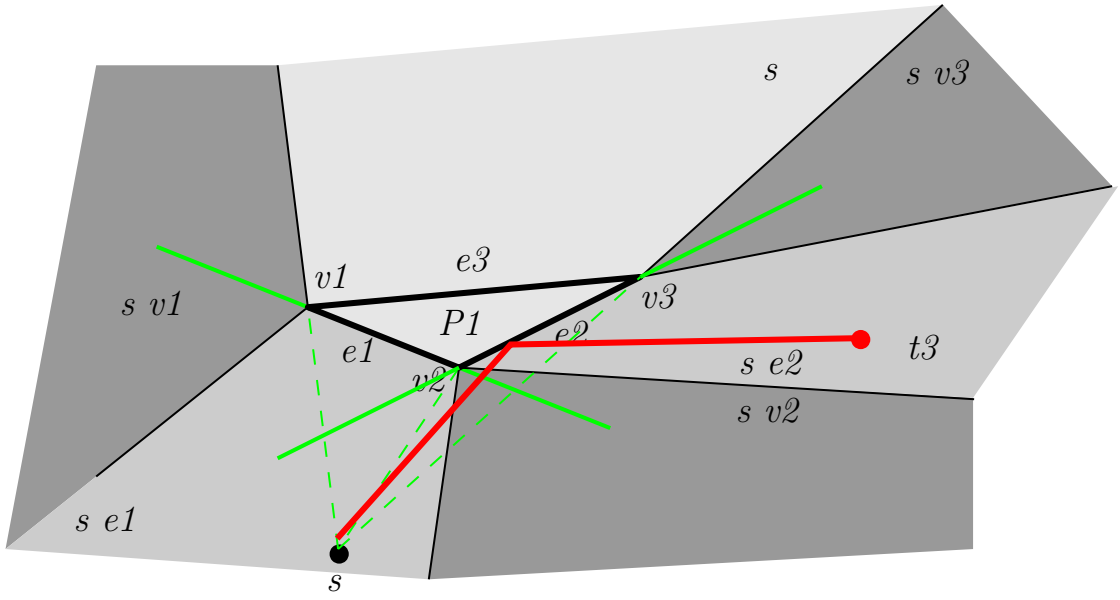
Full comb. shortest path map: Beispiel

- Fixiere den Startpunkt s
- Teile die Ebene in Regionen ein:
- *Kombinatorisch* gleiche Kürzeste Wege



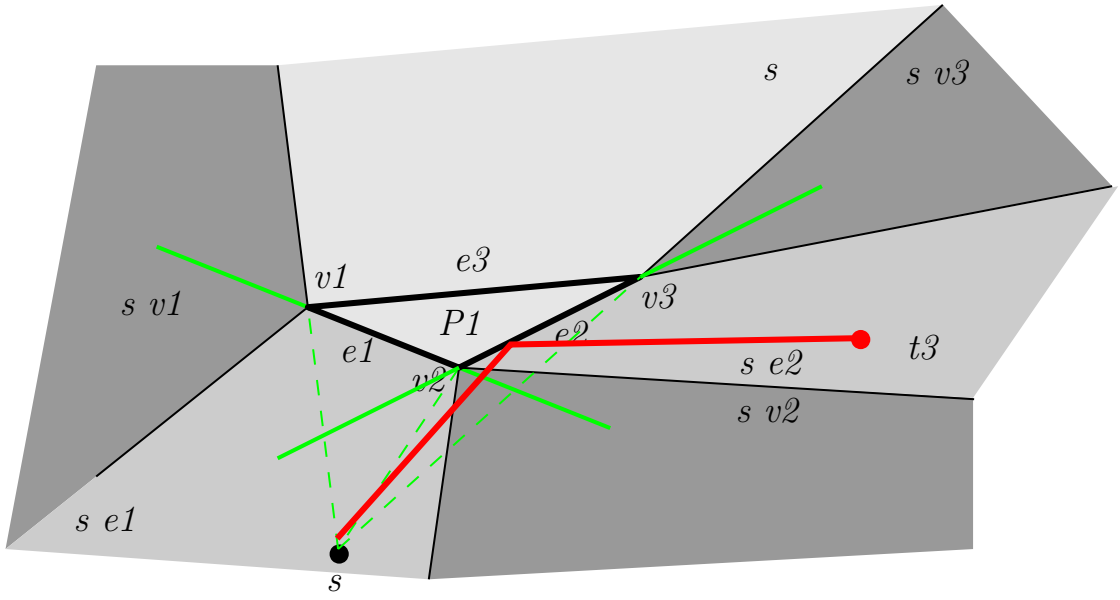
Full comb. shortest path map: Beispiel

- Fixiere den Startpunkt s
- Teile die Ebene in Regionen ein:
- *Kombinatorisch* gleiche Kürzeste Wege



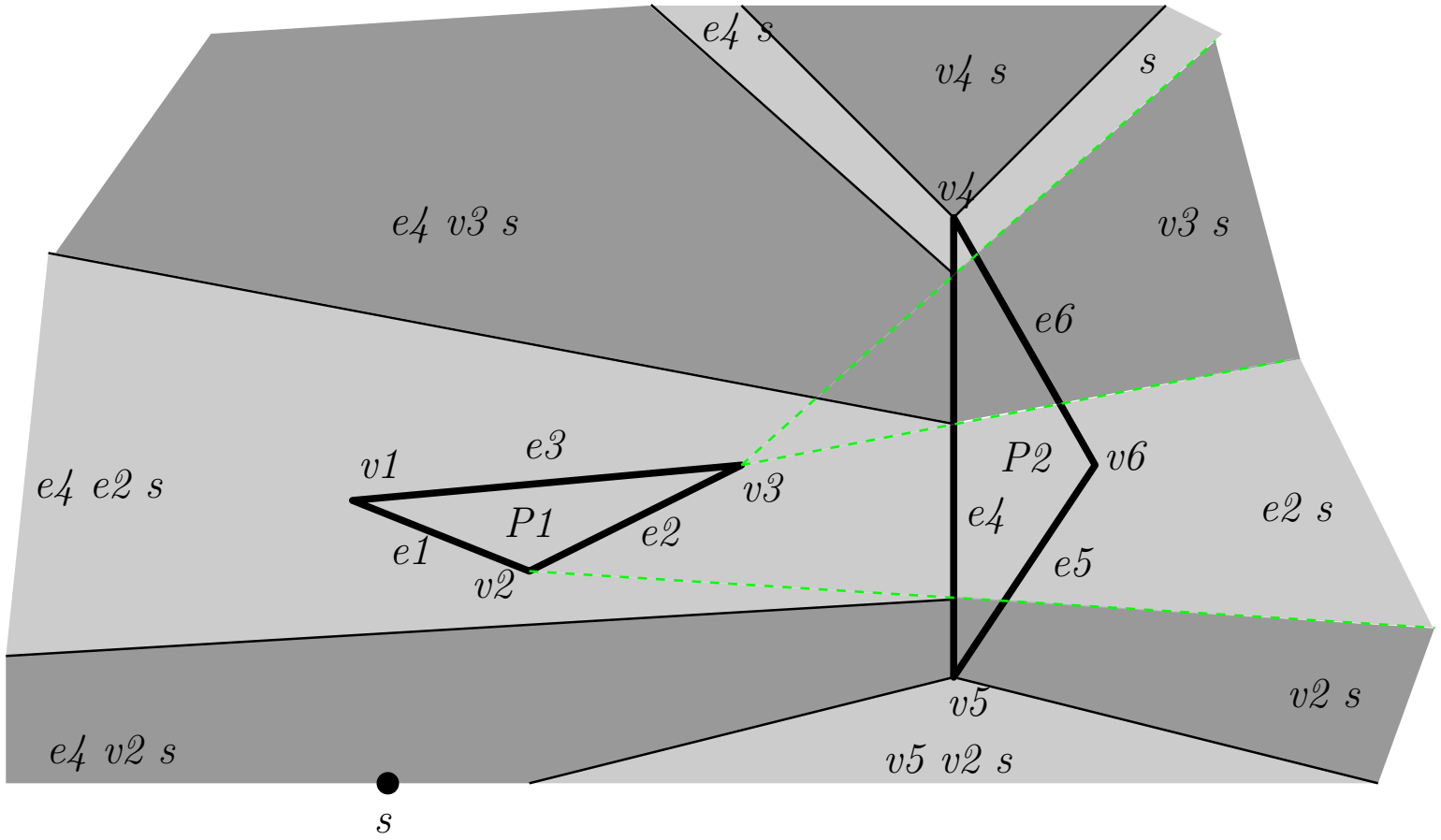
Full comb. shortest path map: Beispiel

- Fixiere den Startpunkt s
- Teile die Ebene in Regionen ein:
- *Kombinatorisch* gleiche Kürzeste Wege

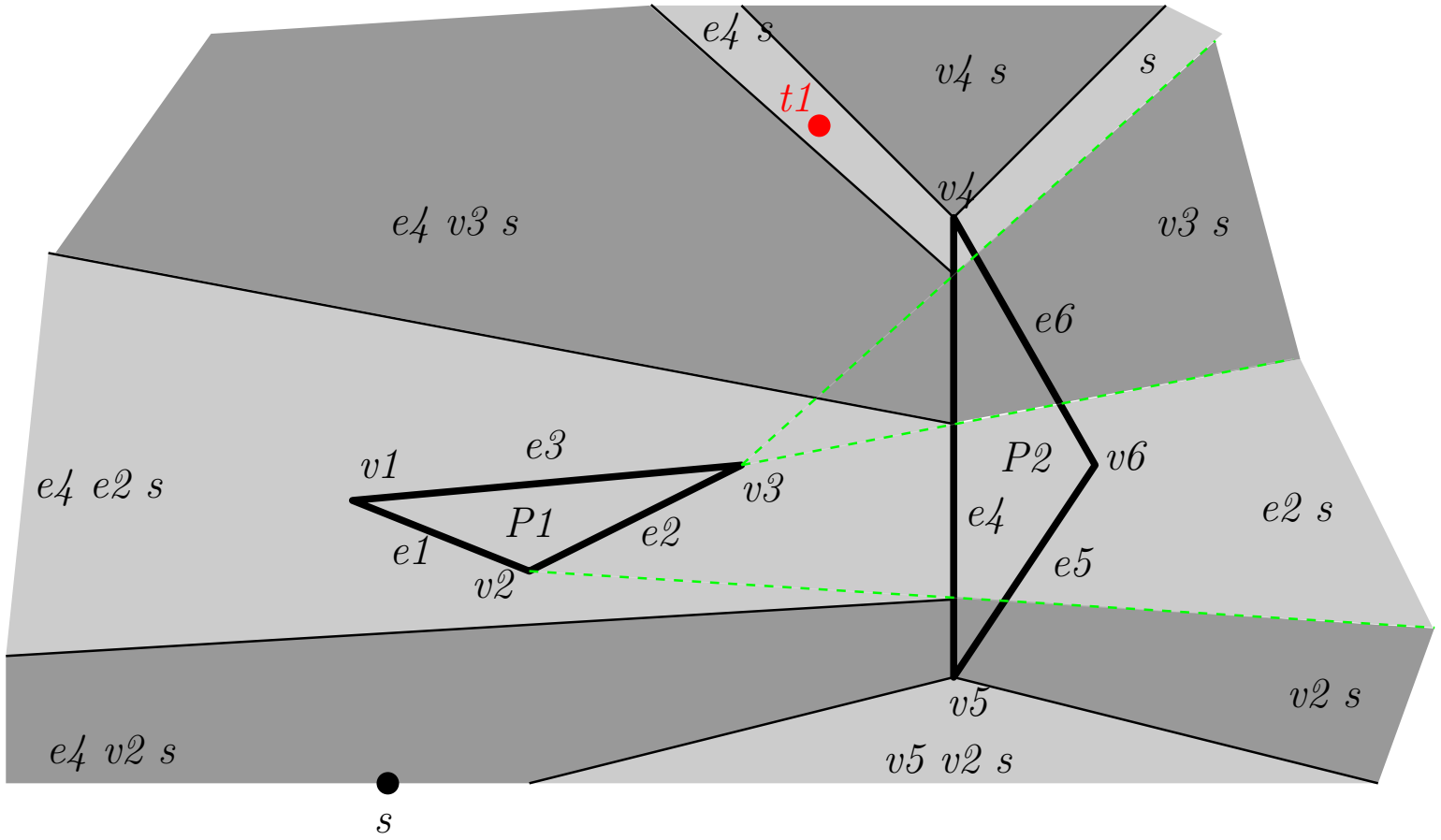


Full comb. shortest path map: Zwei Polygone

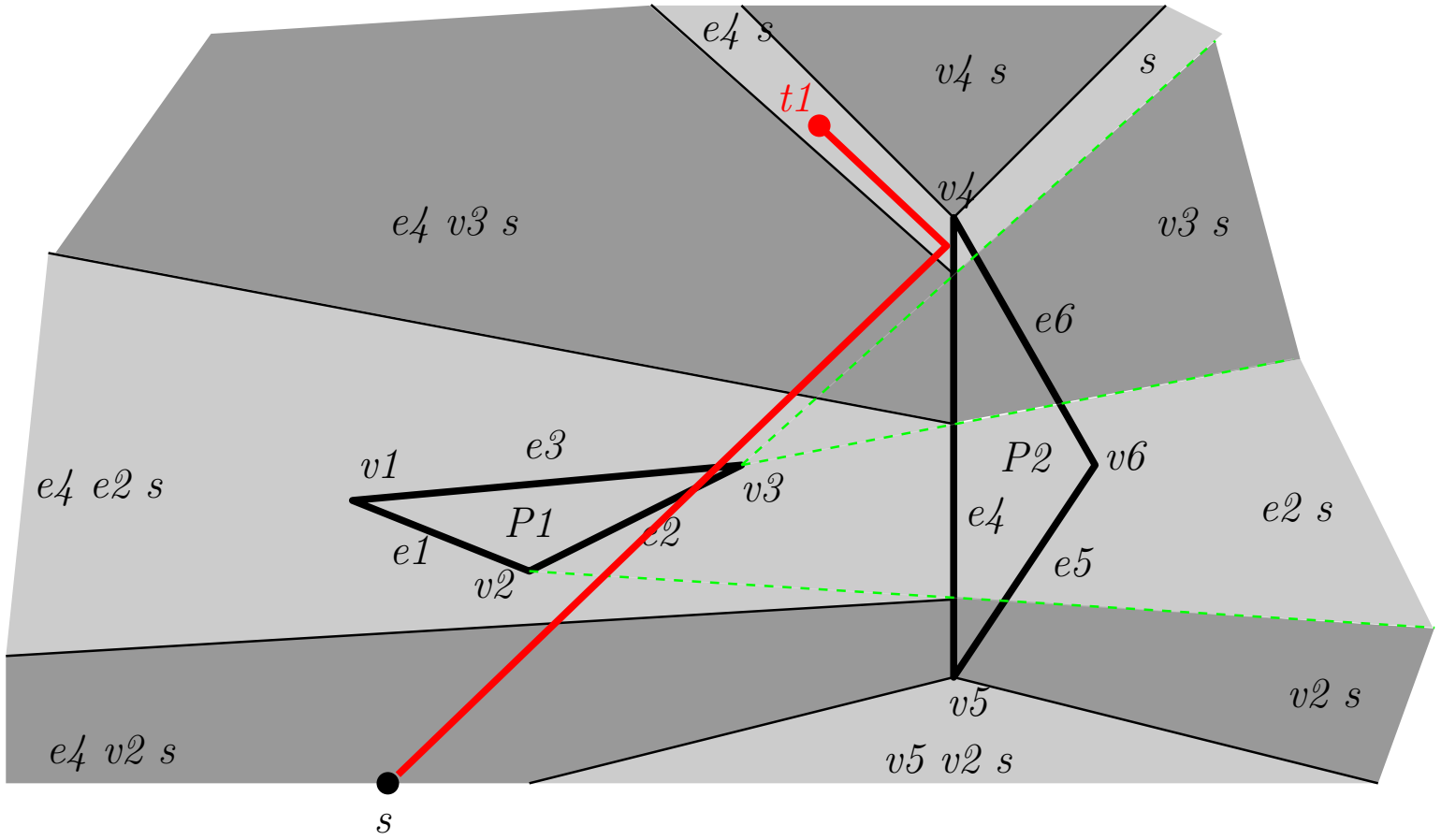
Full comb. shortest path map: Zwei Polygone



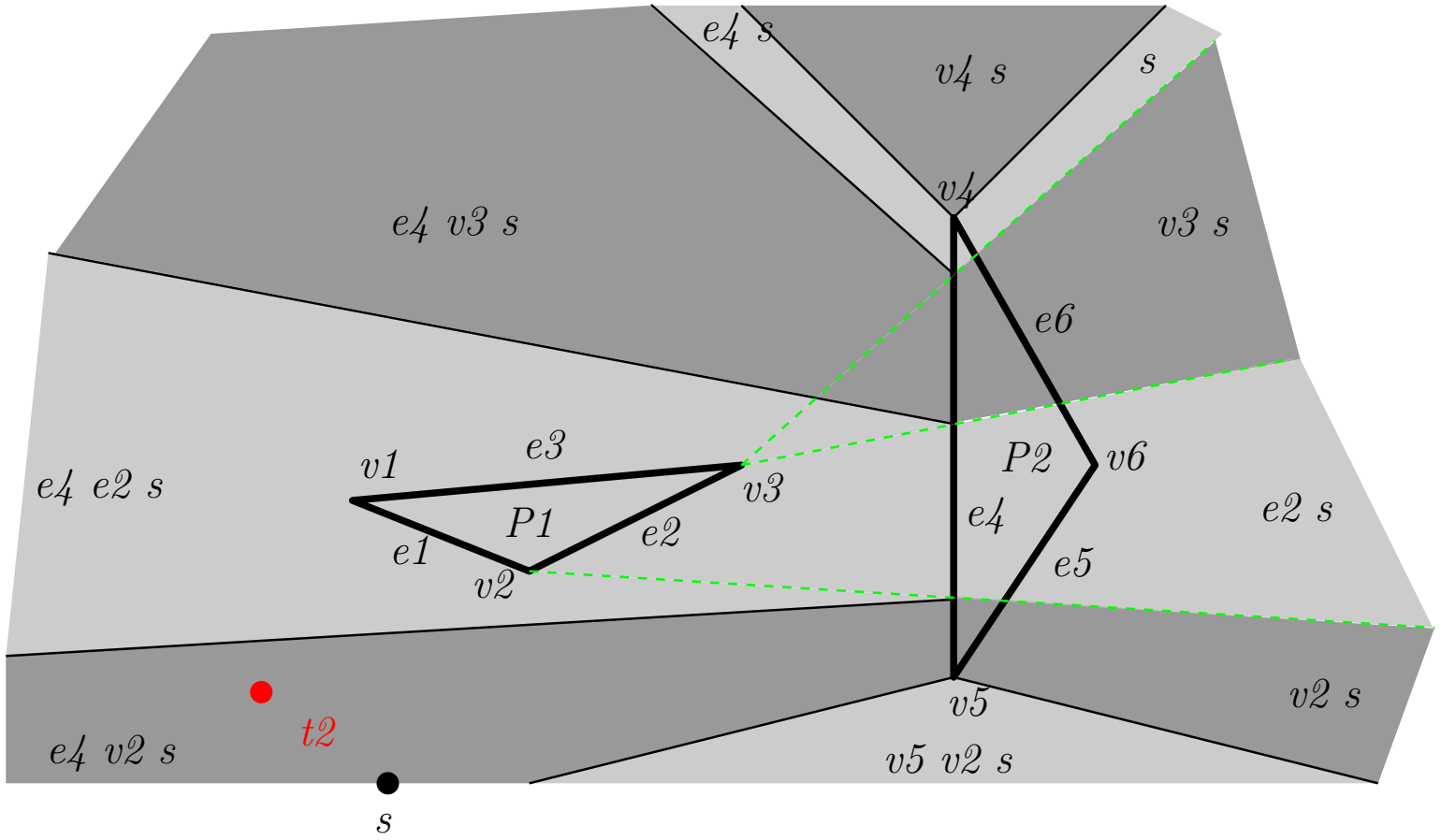
Full comb. shortest path map: Zwei Polygone



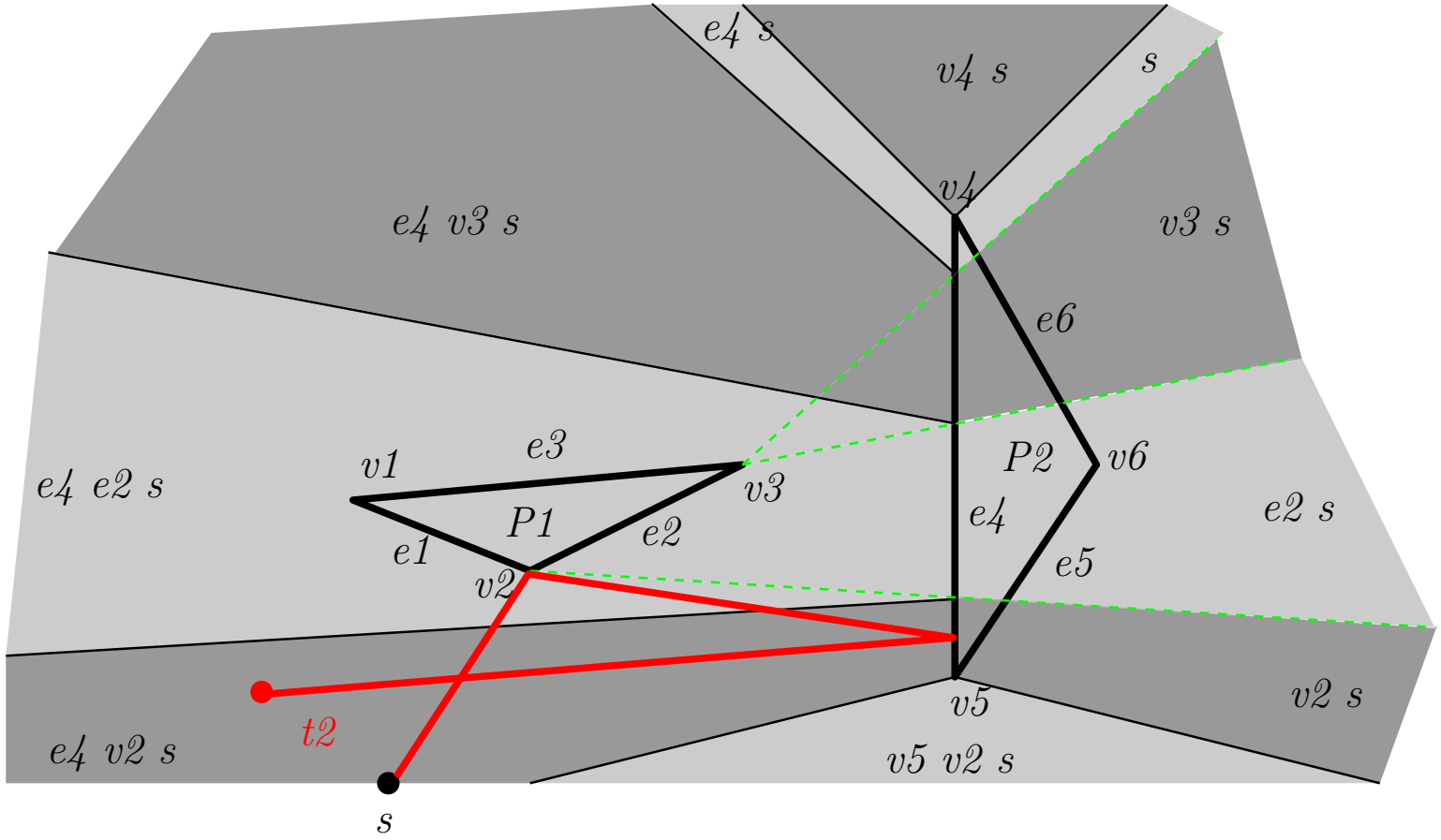
Full comb. shortest path map: Zwei Polygone



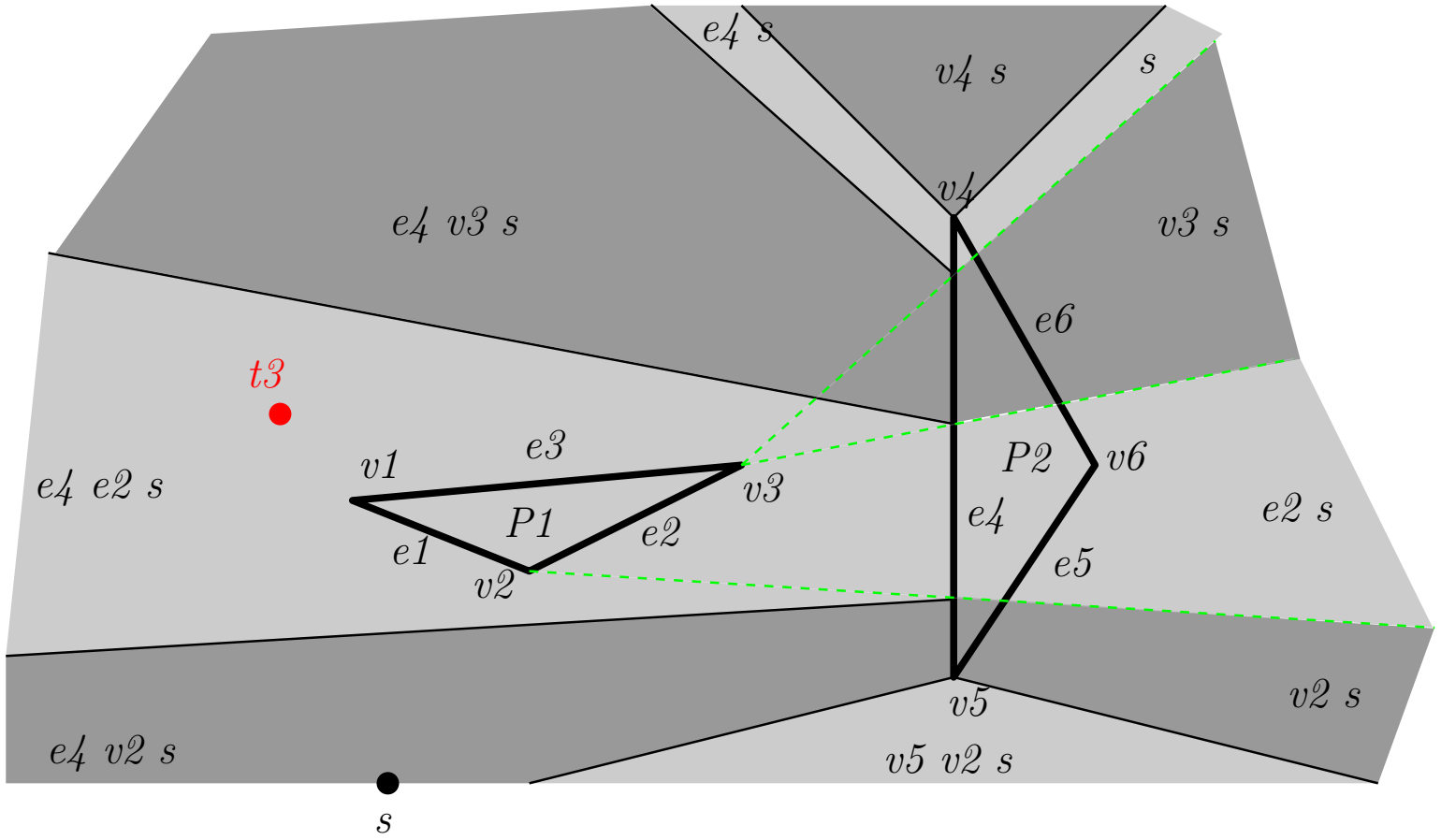
Full comb. shortest path map: Zwei Polygone



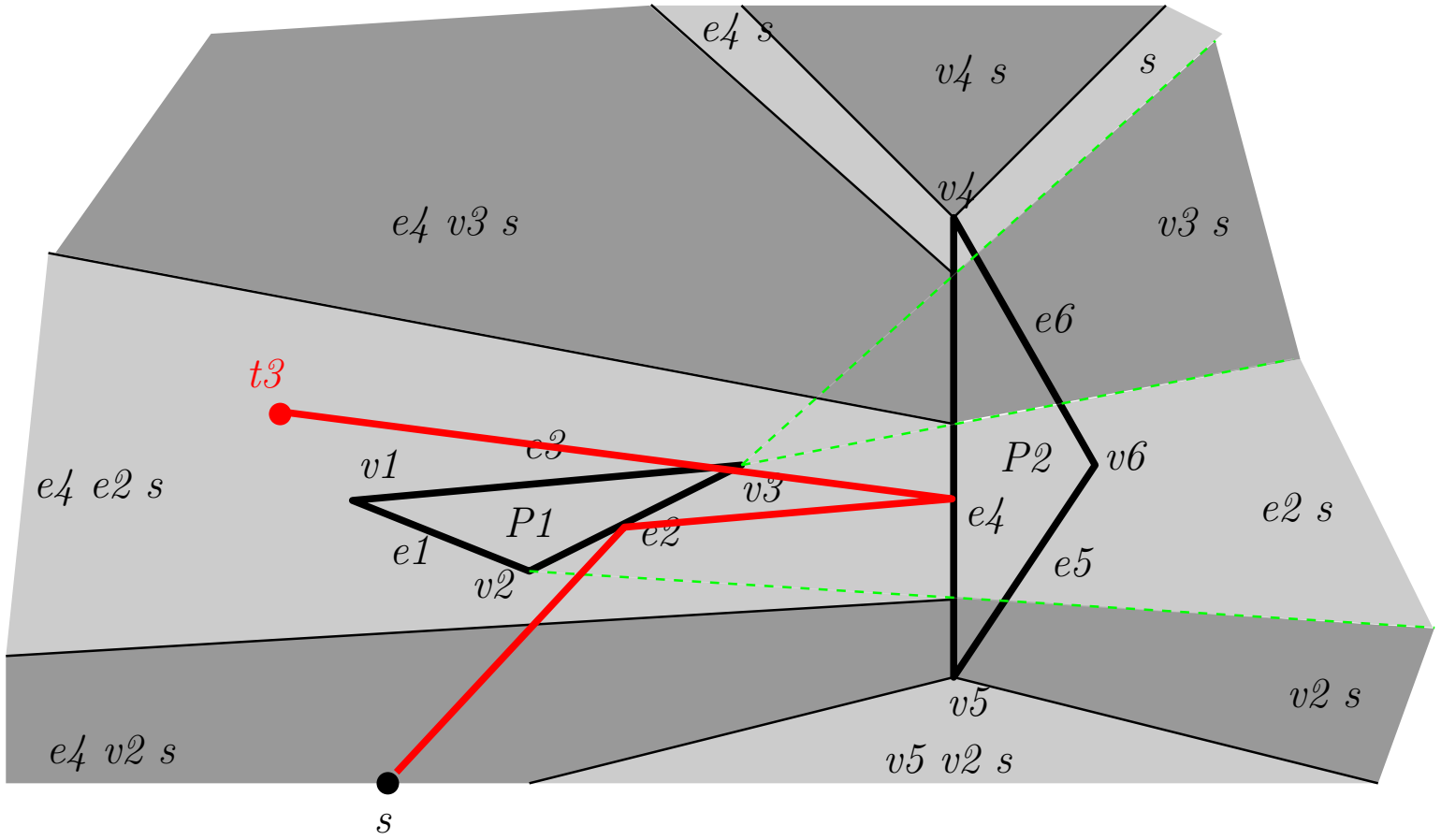
Full comb. shortest path map: Zwei Polygone



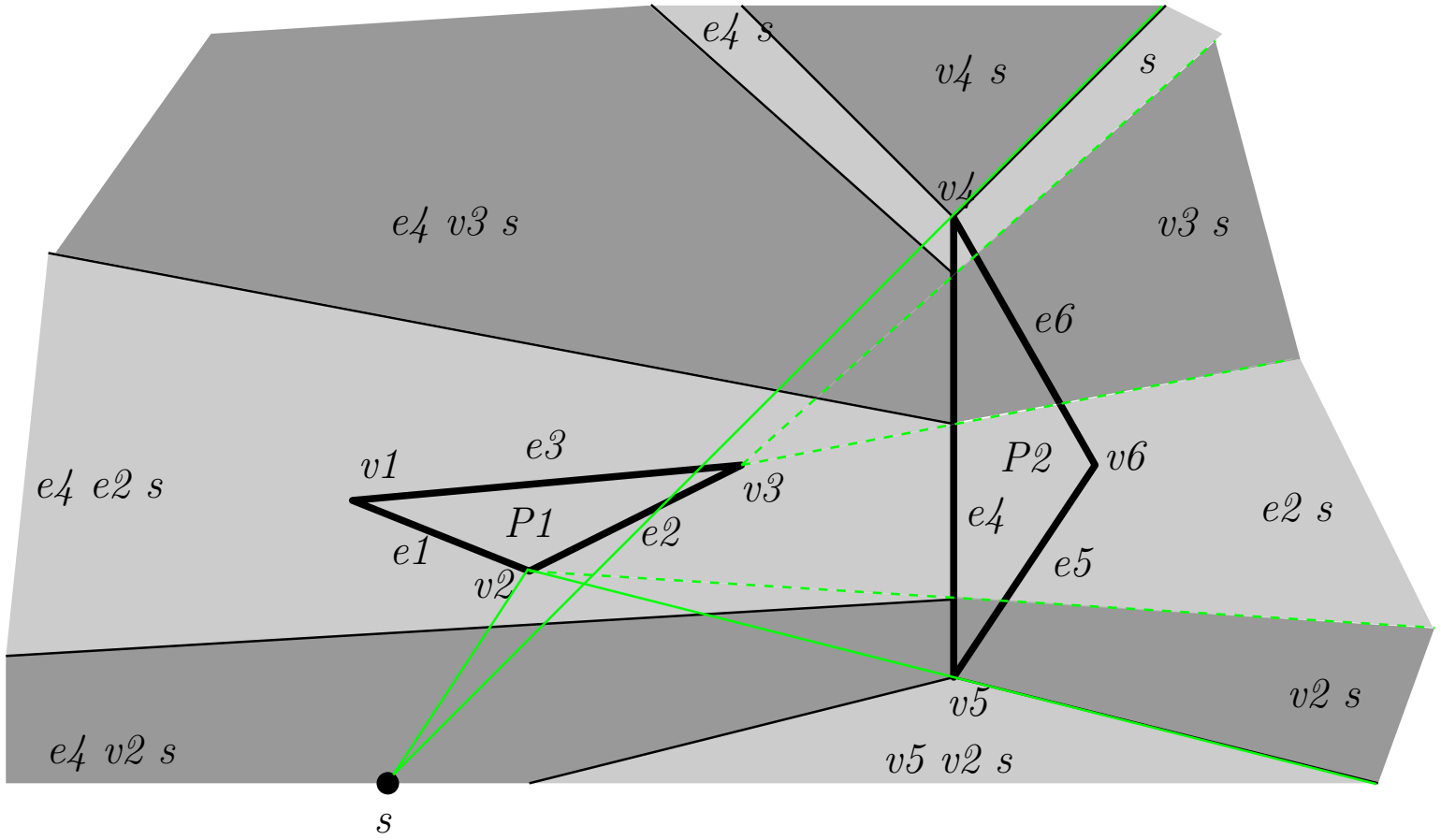
Full comb. shortest path map: Zwei Polygone



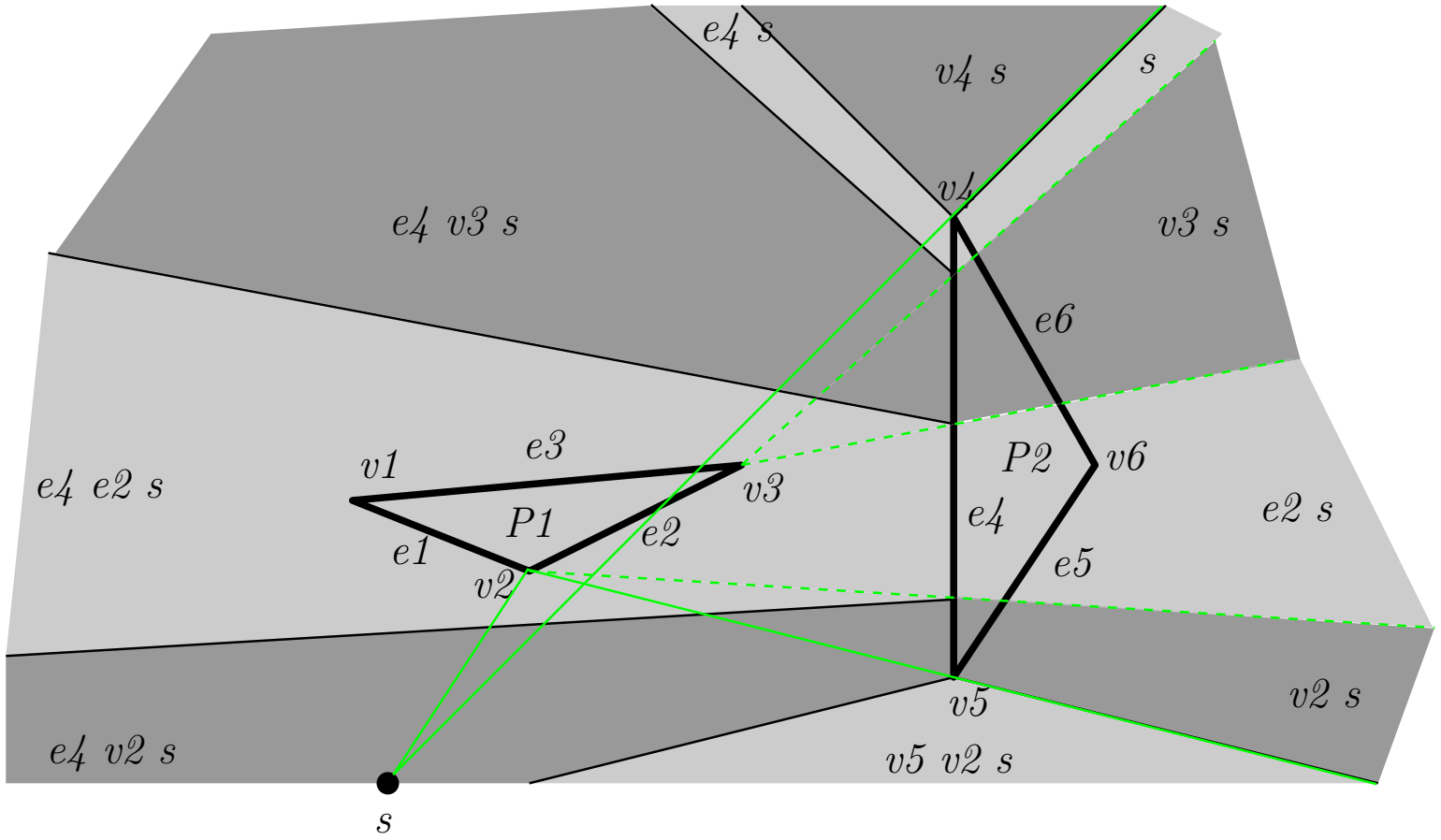
Full comb. shortest path map: Zwei Polygone



Full comb. shortest path map: Zwei Polygone



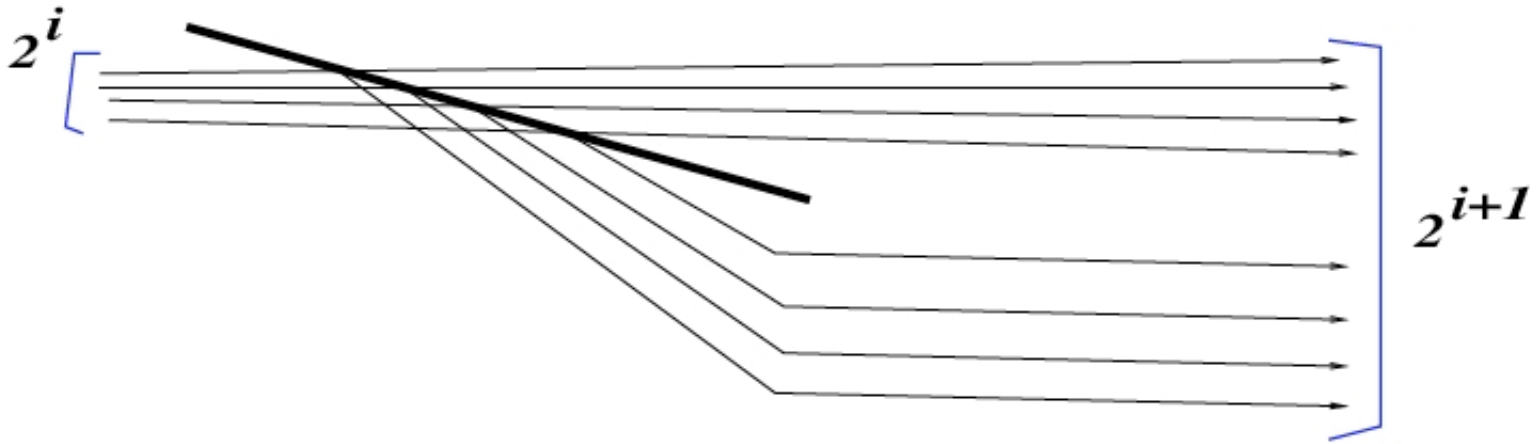
Full comb. shortest path map: Zwei Polygone



Komplexität der FC SPM

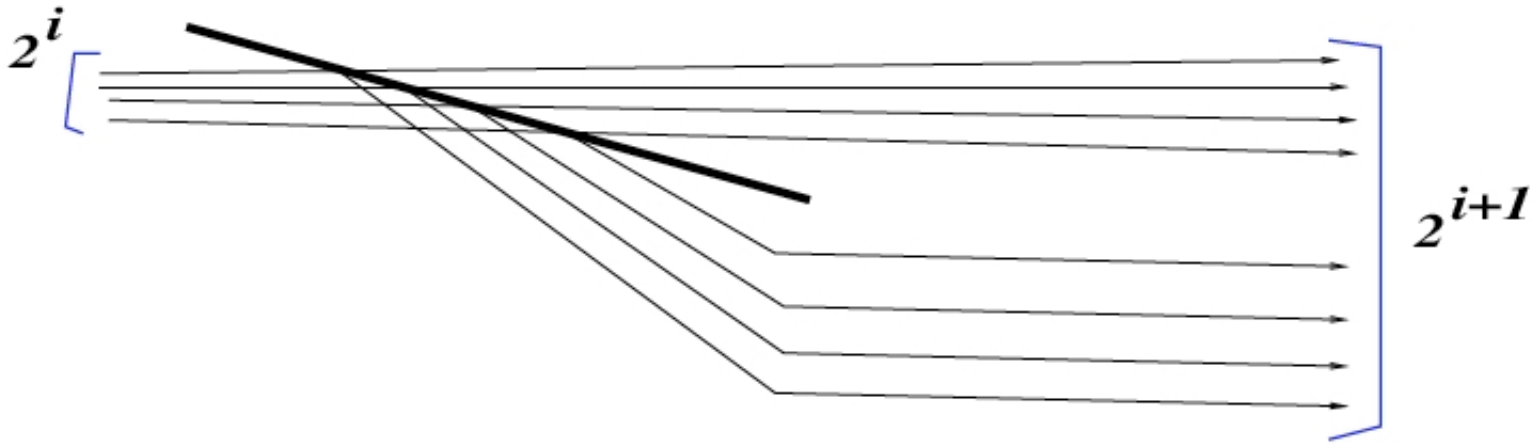
Komplexität der FC SPM

Anzahl der Kanten: Verdoppeln für jedes Polygon!



Komplexität der FC SPM

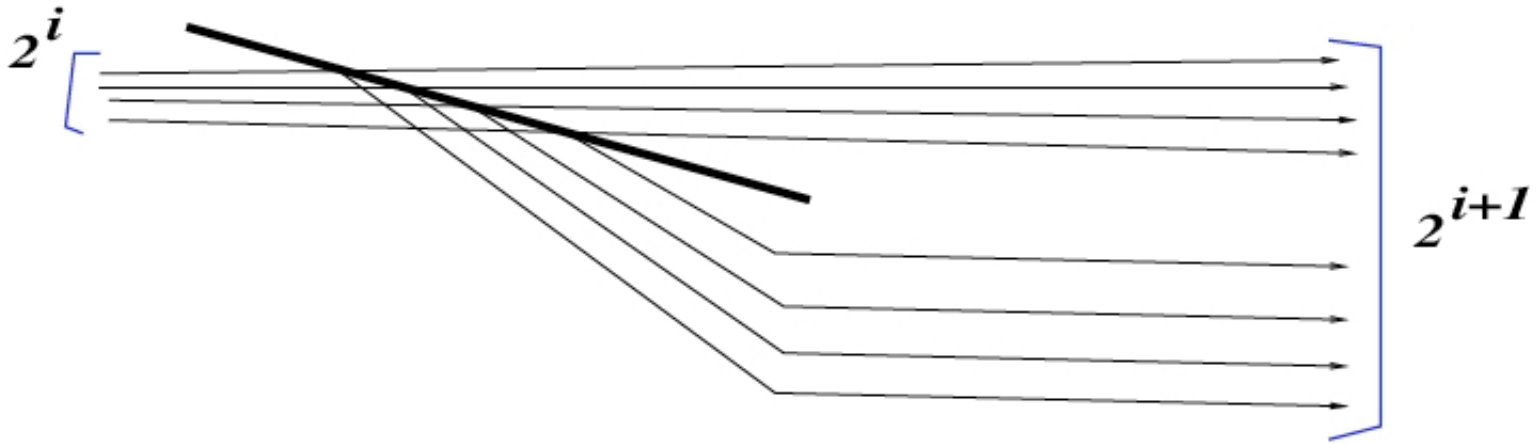
Anzahl der Kanten: Verdoppeln für jedes Polygon!



Mehr als $\Omega((n-k)2^k)$ Kanten!

Komplexität der FC SPM

Anzahl der Kanten: Verdoppeln für jedes Polygon!



Mehr als $\Omega((n-k)2^k)$ Kanten! Zu viele!