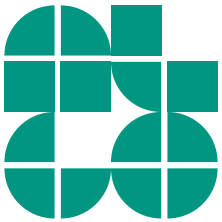
Übung Algorithmische Kartografie

Übungsblatt 10

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

Benjamin Niedermann

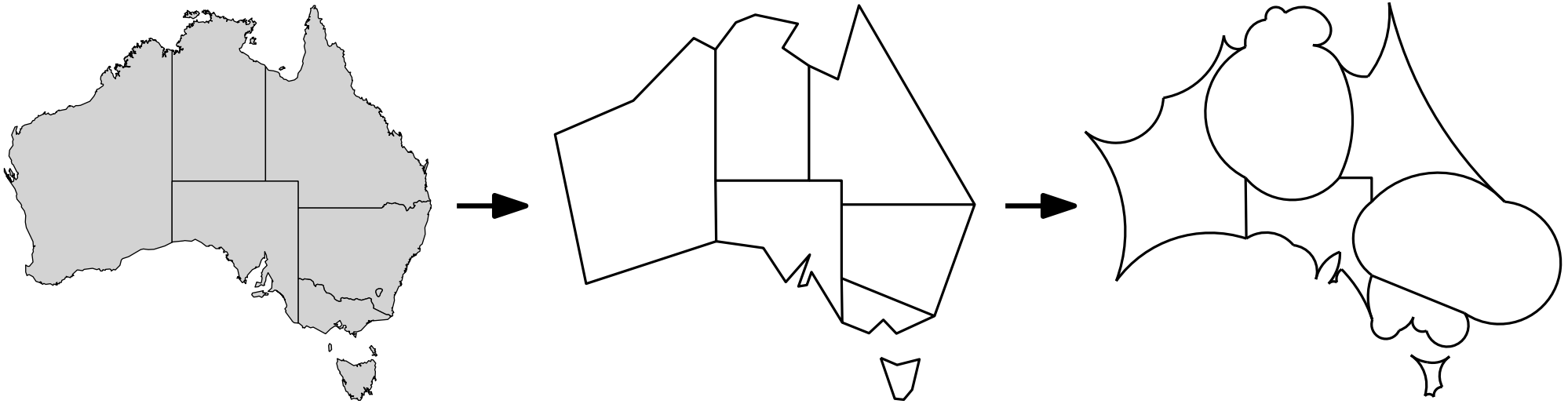
11.07.2013



Circular-Arc Cartograms [Kämper et al. '13]

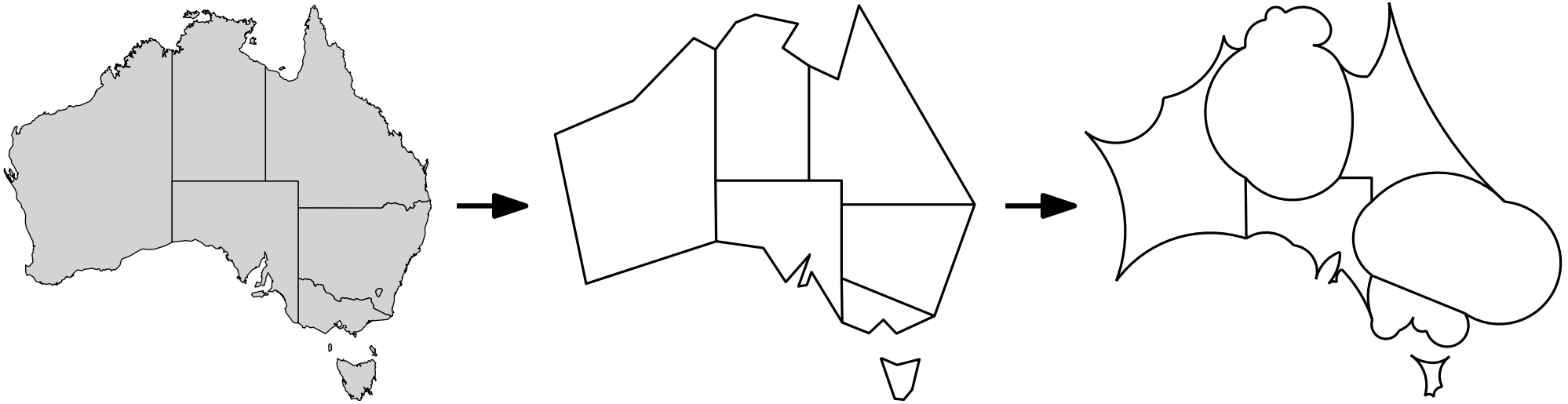
Idea:

- take region boundaries of M
- simplify them
- transform every polygon edge into a circular arc
- adjust the arc radii to obtain target areas



Circular-Arc Cartograms [Kämper et al. '13]

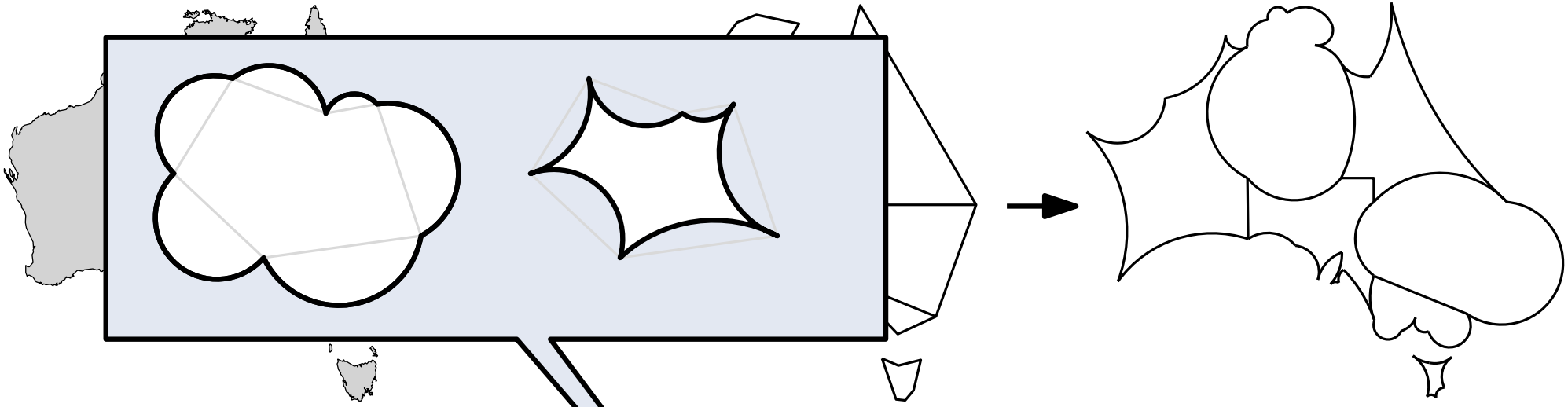
Idea:
- take region boundaries of M
- simplify them
- transform every polygon edge into a circular arc
- adjust the arc radii to obtain target areas



Properties:
- vertices keep positions $\rightarrow$ no displacement
- correct adjacencies
- similar shapes for moderate area changes
- intuitive cloud and snowflake look
- the more simplified the more area potential

Circular-Arc Cartograms [Kämper et al. '13]

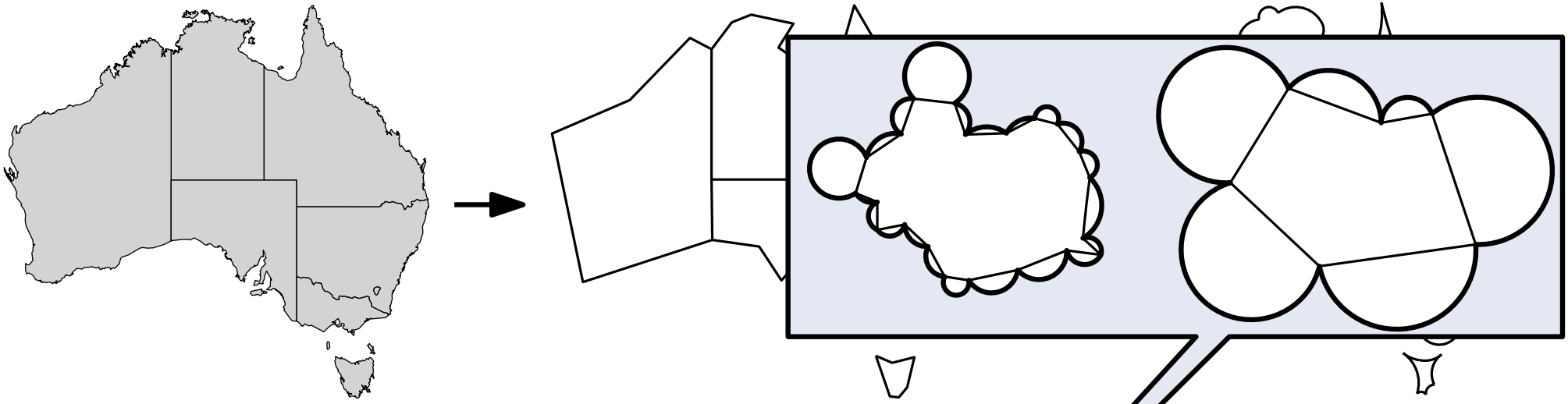
Idea:
- take region boundaries of M
- simplify them
- transform every polygon edge into a circular arc
- adjust the arc radii to obtain target areas



Properties:
- vertices keep positions → no displacement
- correct adjacencies
- similar shapes for moderate area changes
- intuitive cloud and snowflake look
- the more simplified the more area potential

Circular-Arc Cartograms [Kämper et al. '13]

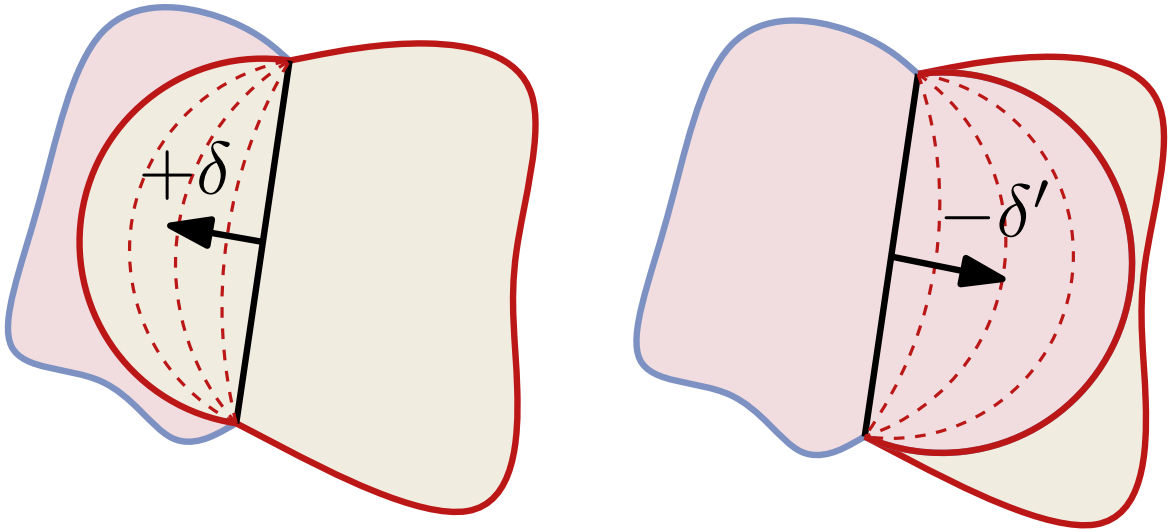
Idea:
- take region boundaries of M
- simplify them
- transform every polygon edge into a circular arc
- adjust the arc radii to obtain target areas



Properties:
- vertices keep positions → no displacement
- correct adjacencies
- similar shapes for moderate area changes
- intuitive cloud and snowflake look
- the more simplified the more area potential

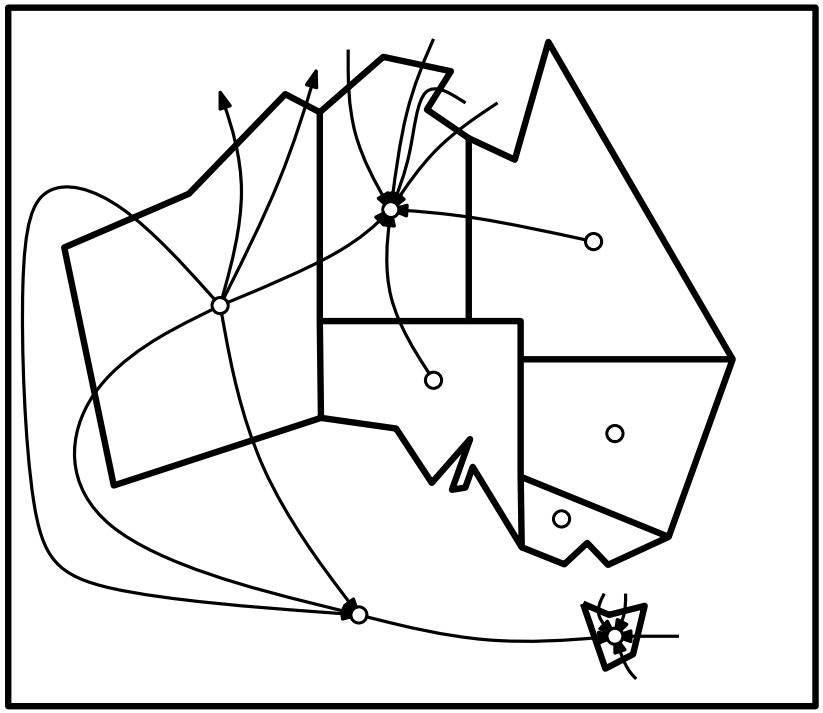
Heuristic Algorithm

Idea: bending an edge transfers area from one face to another



Heuristic Algorithm

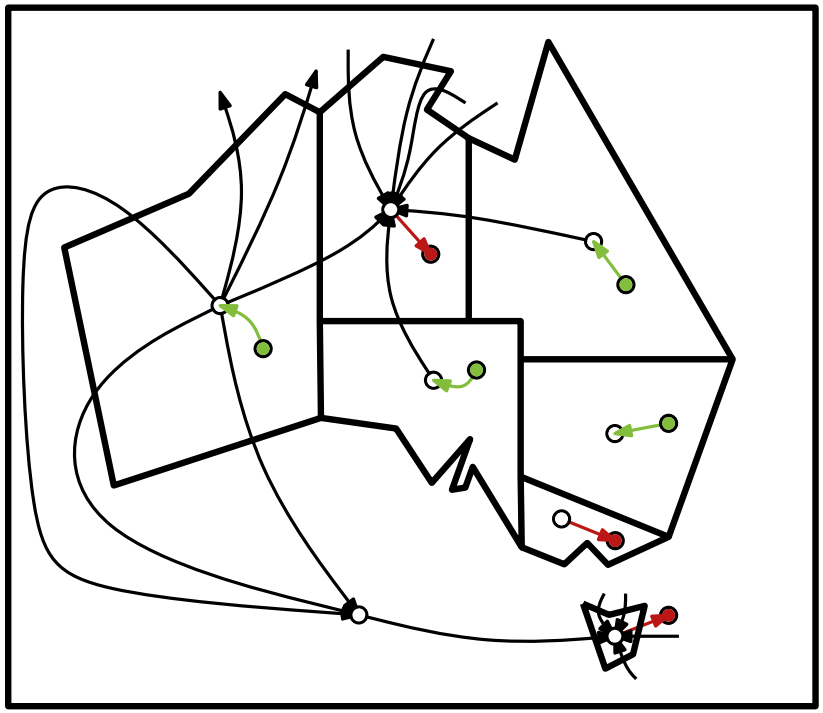
Idea: bending an edge transfers area from one face to another
$\rightarrow$ define a **network flow model** s.t. valid maximum flow corresponds to circular-arc cartogram with small error



- construct directed **dual graph**:
  one dual vertex v_i per face f_i,
  one dual edge per polygon edge
- **strong CAC:** direct edges outward if $|t_i - a_i| > 0$ & inward if $|t_i - a_i| < 0$
- **weak CAC:** bidirected pairs of edges

Heuristic Algorithm

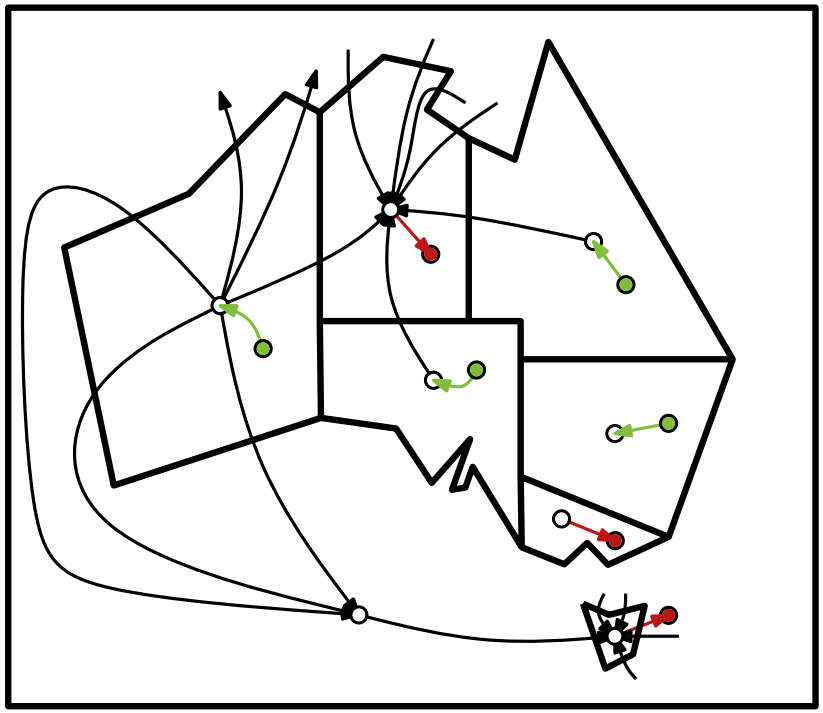
Idea: bending an edge transfers area from one face to another
$\rightarrow$ define a **network flow model** s.t. valid maximum flow corresponds to circular-arc cartogram with small error



- construct directed **dual graph**:
  one dual vertex v_i per face f_i,
  one dual edge per polygon edge
- **strong CAC:** direct edges outward if $t_i - a_i > 0$ & inward if $t_i - a_i < 0$
- **weak CAC:** bidirected pairs of edges
- add another vertex v_i' for each v_i
- if $\Delta_i = t_i - a_i > 0$ make v_i' a source and add (v_i', v_i) with capacity Δ_i
- if $\Delta_i = t_i - a_i < 0$ make v_i' a sink and add (v_i, v_i') with capacity $-\Delta_i$

Heuristic Algorithm

Idea: bending an edge transfers area from one face to another
$\rightarrow$ define a **network flow model** s.t. valid maximum flow corresponds to circular-arc cartogram with small error



- construct directed **dual graph**:
  one dual vertex v_i per face f_i,
  one dual edge per polygon edge
- **strong CAC:** direct edges outward if $t_i - a_i > 0$ & inward if $t_i - a_i < 0$
- **weak CAC:** bidirected pairs of edges
- add another vertex v_i' for each v_i
- if $\Delta_i = t_i - a_i > 0$ make v_i' a source and add (v_i', v_i) with capacity Δ_i
- if $\Delta_i = t_i - a_i < 0$ make v_i' a sink and add (v_i, v_i') with capacity $-\Delta_i$

How do we set the remaining edge capacities?

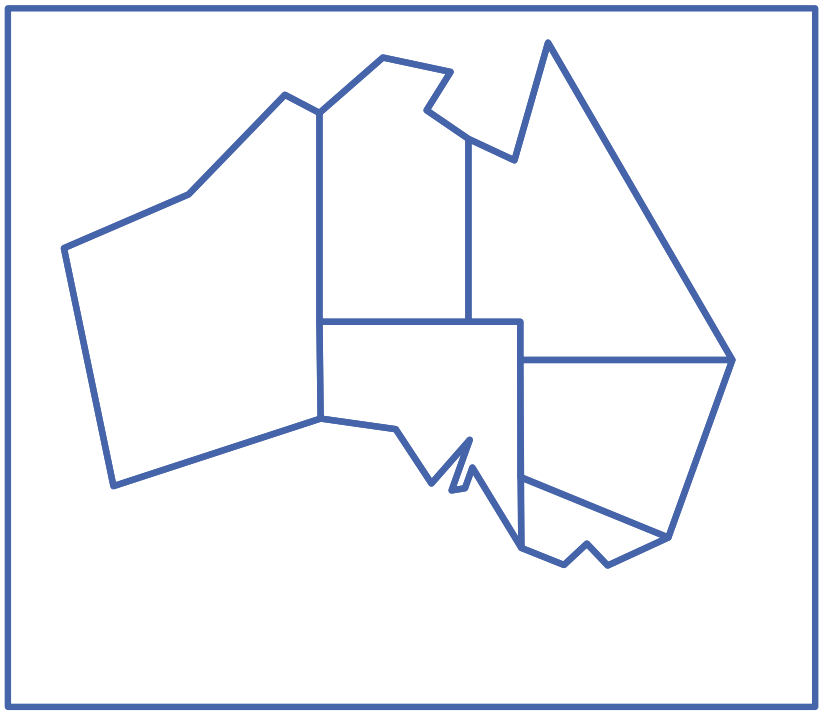
Defining Edge Capacities

Requirement: regardless of flow value, corresponding circular arc is not allowed to intersect any other arc

Defining Edge Capacities

Requirement: regardless of flow value, corresponding circular arc is not allowed to intersect any other arc

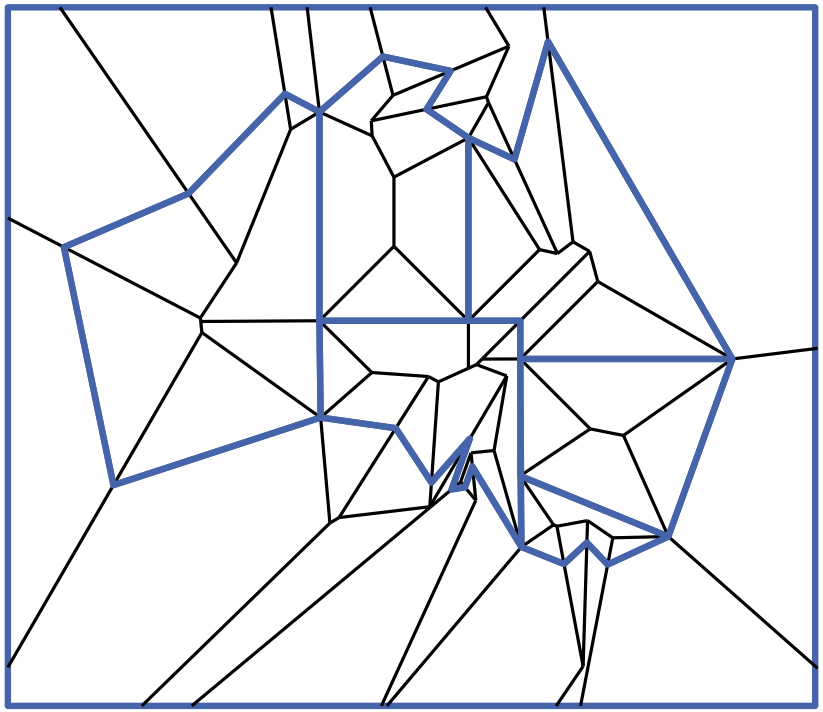
Solution: compute **straight skeleton** of the subdivision and confine every arc to stay within its skeleton cell



Defining Edge Capacities

Requirement: regardless of flow value, corresponding circular arc is not allowed to intersect any other arc

Solution: compute **straight skeleton** of the subdivision and confine every arc to stay within its skeleton cell

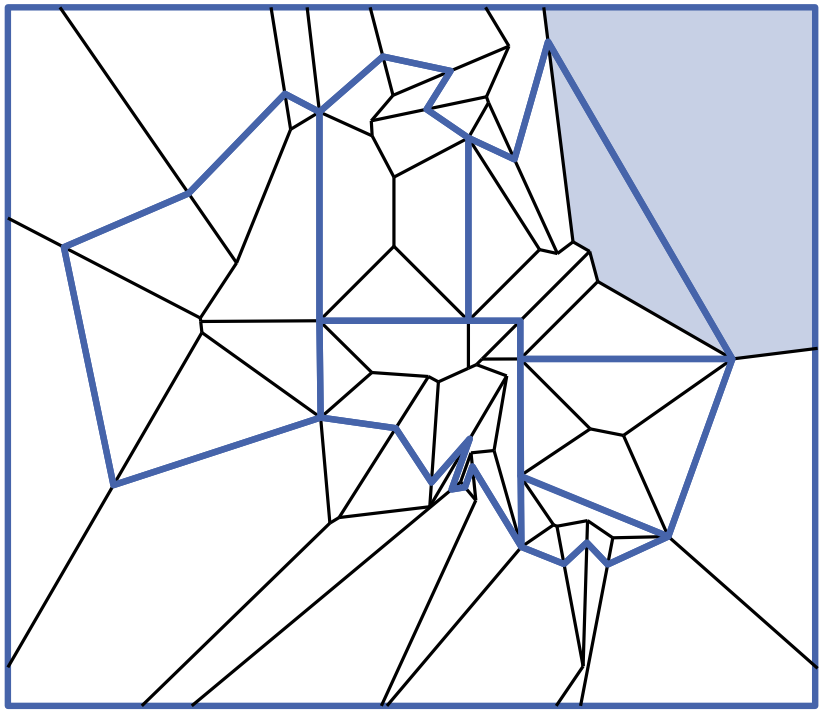


- straight skeleton can be computed in $O(m \log^2 m)$ time [Cheng, Vigneron '07]
- divides each m-edge polygon into m polygonal cells
- compute maximal arcs to both sides of each edge that remain inside their cells $\rightarrow$ no intersections
- max areas define edge capacities $c(e)$
- each flow $F(e) \leq c(e)$ can be realized as a valid circular arc with area transfer $F(e)$

Defining Edge Capacities

Requirement: regardless of flow value, corresponding circular arc is not allowed to intersect any other arc

Solution: compute **straight skeleton** of the subdivision and confine every arc to stay within its skeleton cell

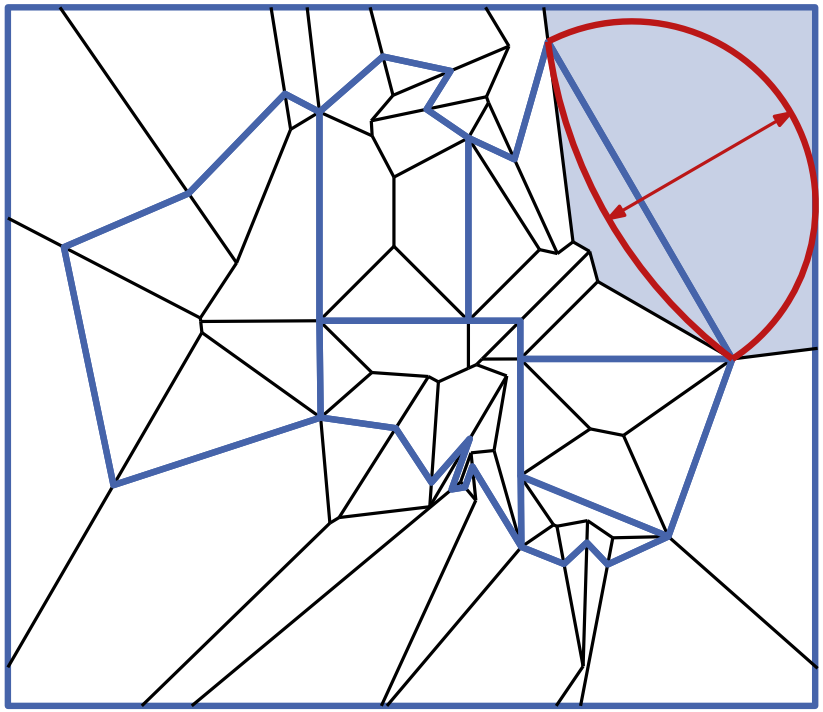


- straight skeleton can be computed in $O(m \log^2 m)$ time [Cheng, Vigneron '07]
- divides each m-edge polygon into m polygonal cells
- compute maximal arcs to both sides of each edge that remain inside their cells $\rightarrow$ no intersections
- max areas define edge capacities $c(e)$
- each flow $F(e) \leq c(e)$ can be realized as a valid circular arc with area transfer $F(e)$

Defining Edge Capacities

Requirement: regardless of flow value, corresponding circular arc is not allowed to intersect any other arc

Solution: compute **straight skeleton** of the subdivision and confine every arc to stay within its skeleton cell

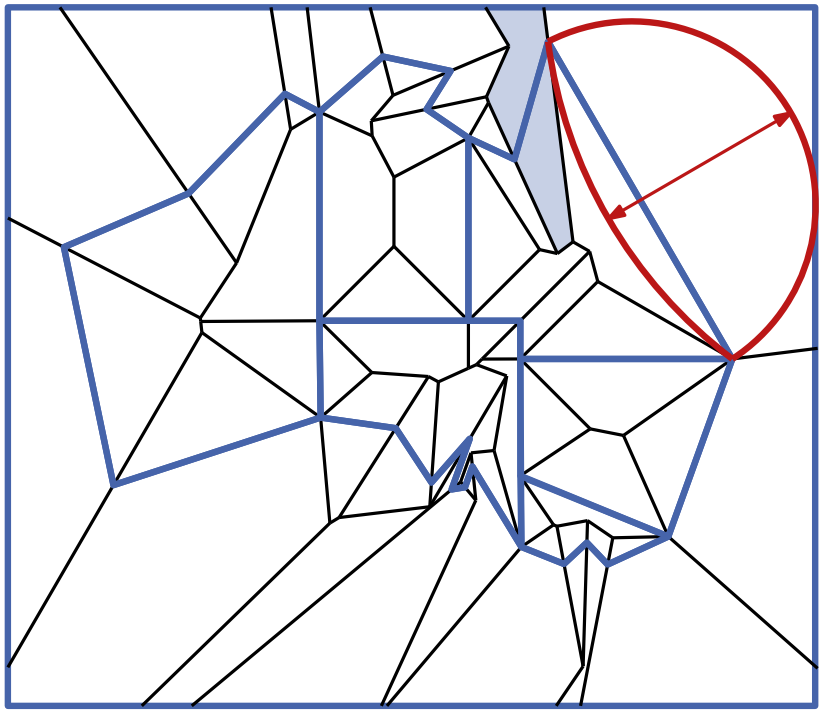


- straight skeleton can be computed in $O(m \log^2 m)$ time [Cheng, Vigneron '07]
- divides each m-edge polygon into m polygonal cells
- compute maximal arcs to both sides of each edge that remain inside their cells $\rightarrow$ no intersections
- max areas define edge capacities $c(e)$
- each flow $F(e) \leq c(e)$ can be realized as a valid circular arc with area transfer $F(e)$

Defining Edge Capacities

Requirement: regardless of flow value, corresponding circular arc is not allowed to intersect any other arc

Solution: compute **straight skeleton** of the subdivision and confine every arc to stay within its skeleton cell

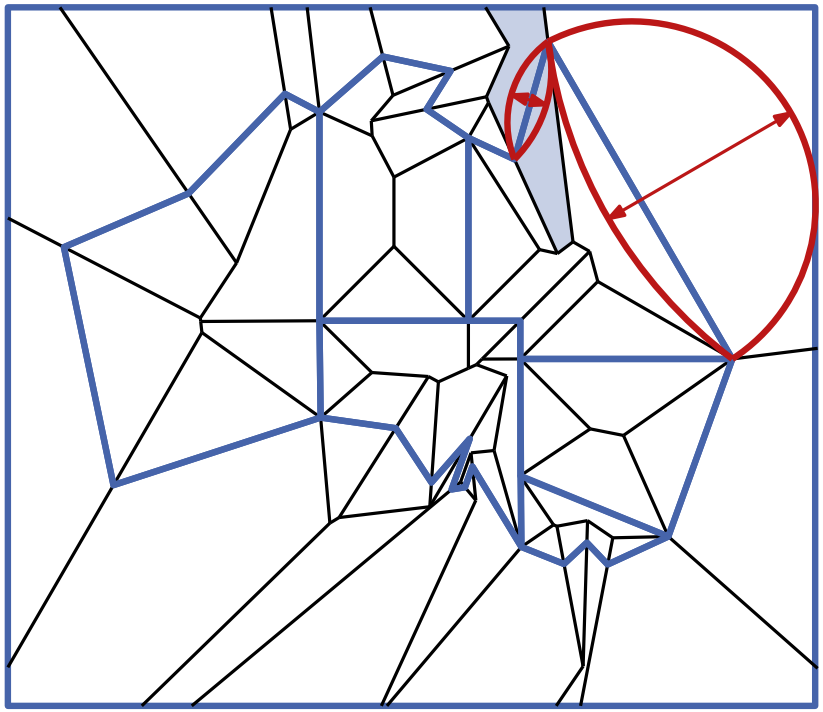


- straight skeleton can be computed in $O(m \log^2 m)$ time [Cheng, Vigneron '07]
- divides each m-edge polygon into m polygonal cells
- compute maximal arcs to both sides of each edge that remain inside their cells $\rightarrow$ no intersections
- max areas define edge capacities $c(e)$
- each flow $F(e) \leq c(e)$ can be realized as a valid circular arc with area transfer $F(e)$

Defining Edge Capacities

Requirement: regardless of flow value, corresponding circular arc is not allowed to intersect any other arc

Solution: compute **straight skeleton** of the subdivision and confine every arc to stay within its skeleton cell

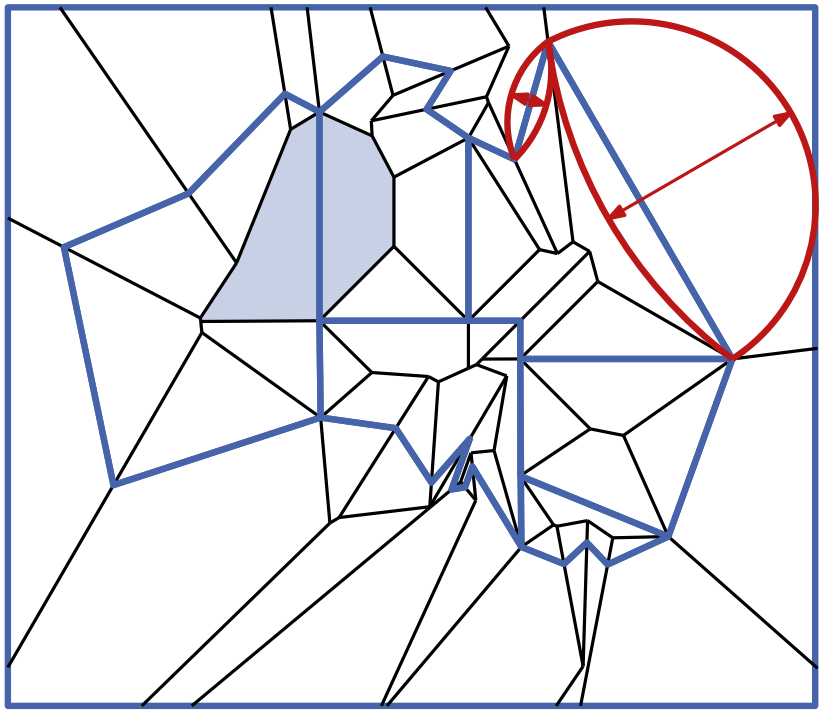


- straight skeleton can be computed in $O(m \log^2 m)$ time [Cheng, Vigneron '07]
- divides each m-edge polygon into m polygonal cells
- compute maximal arcs to both sides of each edge that remain inside their cells $\rightarrow$ no intersections
- max areas define edge capacities $c(e)$
- each flow $F(e) \leq c(e)$ can be realized as a valid circular arc with area transfer $F(e)$

Defining Edge Capacities

Requirement: regardless of flow value, corresponding circular arc is not allowed to intersect any other arc

Solution: compute **straight skeleton** of the subdivision and confine every arc to stay within its skeleton cell

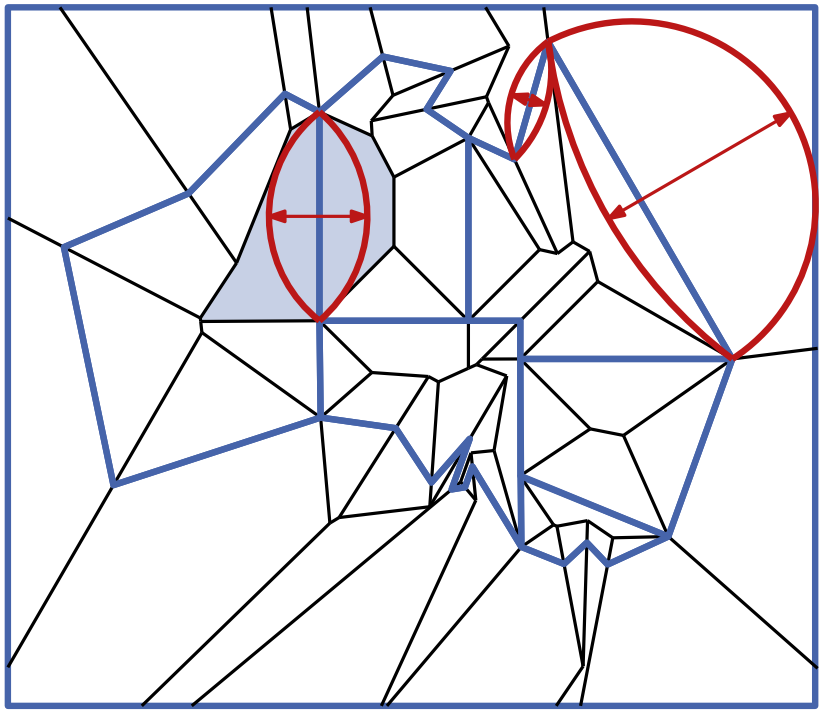


- straight skeleton can be computed in $O(m \log^2 m)$ time [Cheng, Vigneron '07]
- divides each m-edge polygon into m polygonal cells
- compute maximal arcs to both sides of each edge that remain inside their cells $\rightarrow$ no intersections
- max areas define edge capacities $c(e)$
- each flow $F(e) \leq c(e)$ can be realized as a valid circular arc with area transfer $F(e)$

Defining Edge Capacities

Requirement: regardless of flow value, corresponding circular arc is not allowed to intersect any other arc

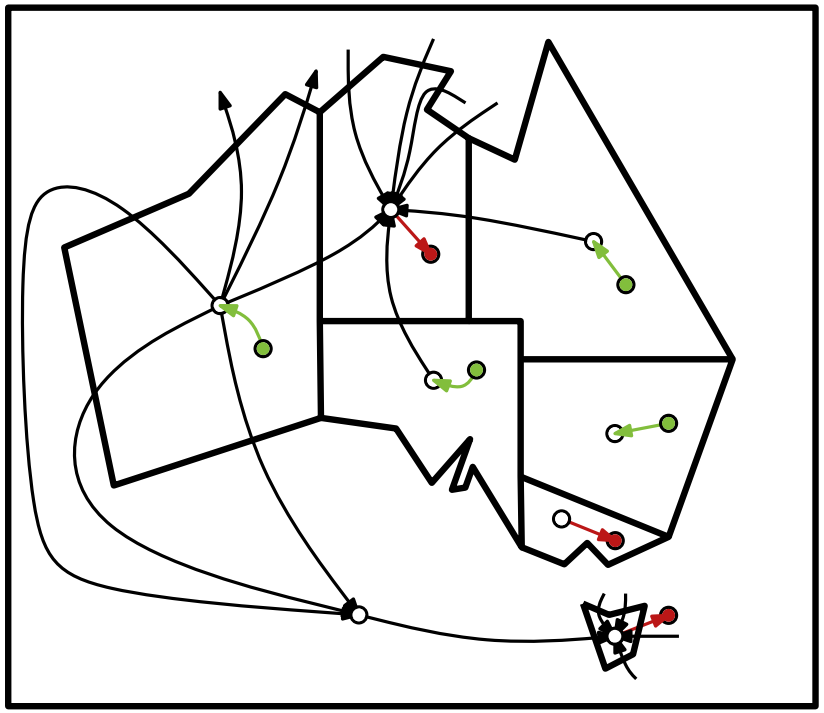
Solution: compute **straight skeleton** of the subdivision and confine every arc to stay within its skeleton cell



- straight skeleton can be computed in $O(m \log^2 m)$ time [Cheng, Vigneron '07]
- divides each m-edge polygon into m polygonal cells
- compute maximal arcs to both sides of each edge that remain inside their cells $\rightarrow$ no intersections
- max areas define edge capacities $c(e)$
- each flow $F(e) \leq c(e)$ can be realized as a valid circular arc with area transfer $F(e)$

Summary of the Algorithm

Via the dual graph G of the input subdivision and edge capacities c obtained from the straight skeleton we get a multiple-source multiple-sink flow network $\mathcal{N} = (G, c, S, T)$.



- G is planar as the dual graph of a planar graph
- maximum flow in a planar flow network $\mathcal{N}$ can be computed in $O(n \log^3 n)$ time [Borradaile et al. '11]
- since $\sum_i a_i = \sum_i t_i$ we get $\sum_{\Delta_i > 0} \Delta_i = -\sum_{\Delta_i < 0} \Delta_i$
- a maximum flow in $\mathcal{N}$ with value $D = \sum_{\Delta_i > 0} \Delta_i$ corresponds to an accurate circular-arc cartogram
- otherwise the cartographic error is minimized within the given capacity constraints

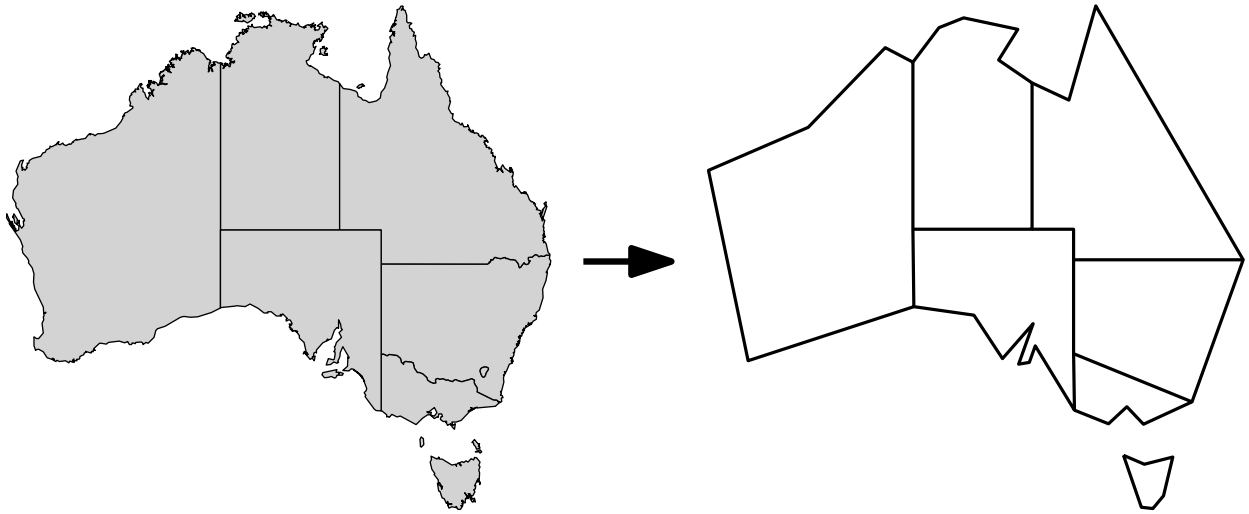
Aufgabe 2

An welchen Stellen beruht der Algorithmus auf heursitischen Überlegungen?

Aufgabe 2

An welchen Stellen beruht der Algorithmus auf heursitischen Überlegungen?

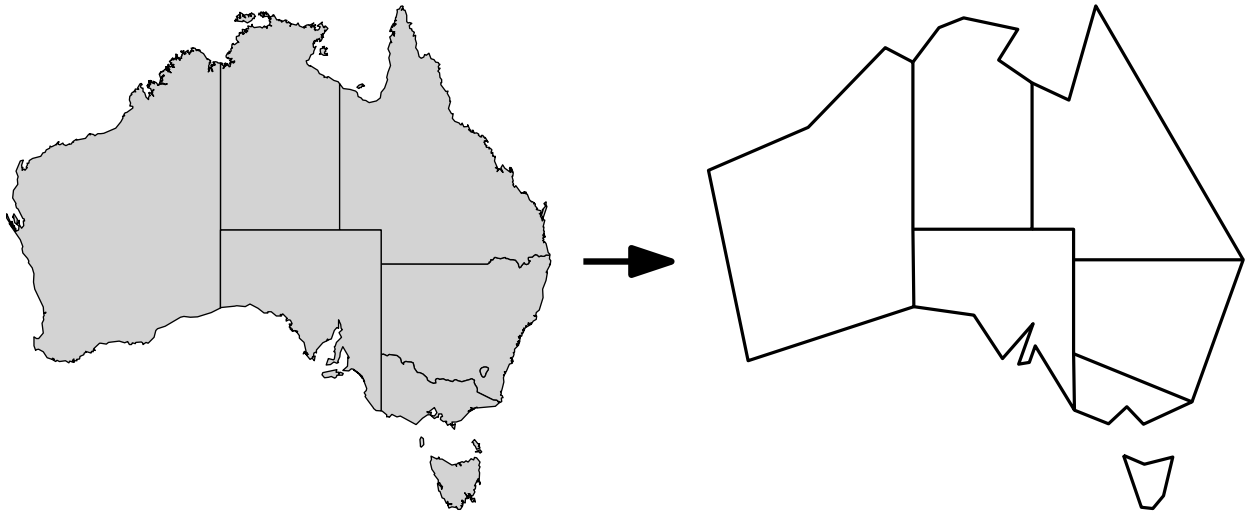
Vereinfachung des Polygons.



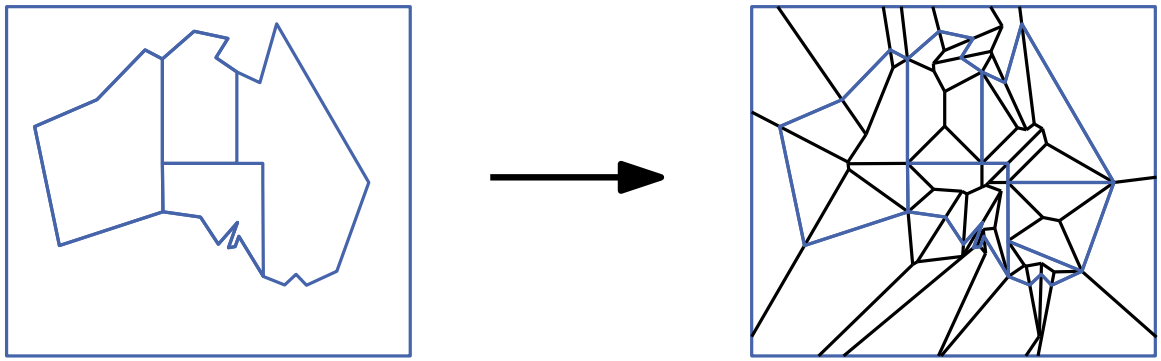
Aufgabe 2

An welchen Stellen beruht der Algorithmus auf heursitischen Überlegungen?

Vereinfachung des Polygons.



Wahl der maximalen *Ausbeulung* der Kreisbögen:



Aufgabe 2

Überlegen Sie sich Verbesserungsmöglichkeiten oder alternative Lösungsansätze zur Erstellung von Circular Arc Cartograms.

Aufgabe 2

Überlegen Sie sich Verbesserungsmöglichkeiten oder alternative Lösungsansätze zur Erstellung von Circular Arc Cartograms.

1. Wahl des Skelets/maximale Ausbeulung der Kreisbögen.

Aufgabe 2

Überlegen Sie sich Verbesserungsmöglichkeiten oder alternative Lösungsansätze zur Erstellung von Circular Arc Cartograms.

1. Wahl des Skelets/maximale Ausbeulung der Kreisbögen.
2. Vereinfachung der Polygons.

Aufgabe 2

Überlegen Sie sich Verbesserungsmöglichkeiten oder alternative Lösungsansätze zur Erstellung von Circular Arc Cartograms.

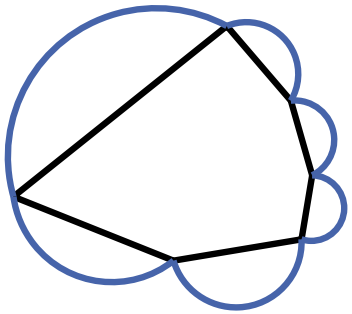
1. Wahl des Skelets/maximale Ausbeulung der Kreisbögen.
2. Vereinfachung der Polygons.
   - Bewege wenige Eckpunkte des Randes um bessere Ergebnisse zu erzielen.

Aufgabe 2

Überlegen Sie sich Verbesserungsmöglichkeiten oder alternative Lösungsansätze zur Erstellung von Circular Arc Cartograms.

1. Wahl des Skelets/maximale Ausbeulung der Kreisbögen.
2. Vereinfachung der Polygons.
   - ↳ Bewege wenige Eckpunkte des Randes um bessere Ergebnisse zu erzielen.

Beobachtung: Die anfängliche Kantenlänge ist entscheidend für den potentiellen Flächenzuwachs.

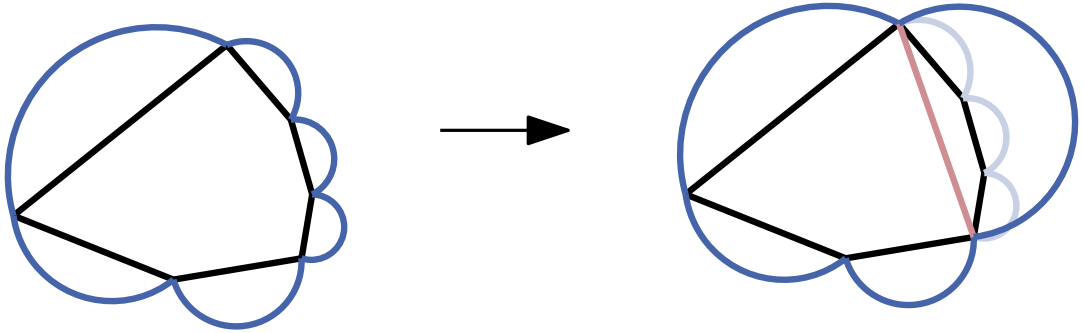


Aufgabe 2

Überlegen Sie sich Verbesserungsmöglichkeiten oder alternative Lösungsansätze zur Erstellung von Circular Arc Cartograms.

1. Wahl des Skelets/maximale Ausbeulung der Kreisbögen.
2. Vereinfachung der Polygons.
   - ↳ Bewege wenige Eckpunkte des Randes um bessere Ergebnisse zu erzielen.

Beobachtung: Die anfängliche Kantenlänge ist entscheidend für den potentiellen Flächenzuwachs.



Idee: Wenn Flächenzuwachs nicht ausreicht, vereinfache Polygon iterativ weiter.

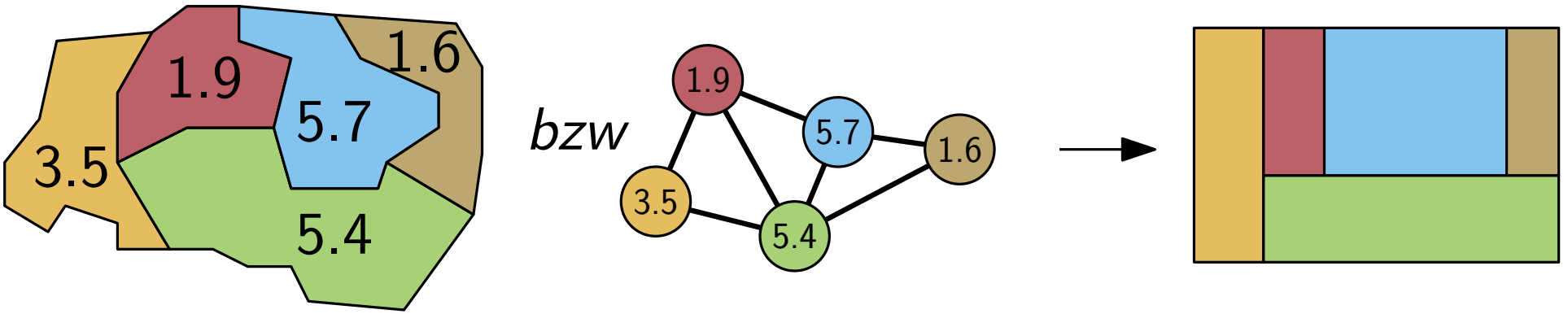
Problemstellung

Geg: politische Karte M (Rechtecksunterteilung), positives Gewicht w_i für jede Region R_i

bzw: knotengewichteter intern triangulierter planar eingeb. Graph G dual zu M, Knoten v_i entspricht Region R_i, Kanten zw. adjazenten Regionen, Knotengewichte w_i

Ges: verzerrte Karte M' äquivalent zu M mit $|R_i| = w_i$

bzw: flächenproportionale Kontaktrepräsentation von G, jeder Knoten v_i als geometrisches Objekt s_i mit Fläche w_i, so dass s_i und s_j sich berühren gdw. $v_i v_j \in E$



Allgemeines Flussmodell

Gegeben: Netzwerk $N = (D = (V, A), b, l, u)$

Allgemeines Flussmodell

Gegeben: Netzwerk $N = (D = (V, A), b, l, u)$

↳ gerichteter Graph $D = (V, A)$

Allgemeines Flussmodell

Gegeben: Netzwerk $N = (D = (V, A), b, l, u)$

↳ gerichteter Graph $D = (V, A)$

Knotenbewertung b: $V \to \mathbb{R}$ mit $\sum_{v \in V} b(i) = 0$

$b(v) > 0$: Quelle $\quad b(v) < 0$: Senke

Allgemeines Flussmodell

Gegeben: Netzwerk $N = (D = (V, A), b, l, u)$

↳ gerichteter Graph $D = (V, A)$
Knotenbewertung $b\colon V \to \mathbb{R}$ mit $\sum_{v \in V} b(i) = 0$
untere Kapazitäten $l\colon A \to \mathbb{R}_0^+$
obere Kapazitäten $u\colon A \to \mathbb{R}_0^+$

$b(v) > 0$: Quelle $b(v) < 0$: Senke

Allgemeines Flussmodell

Gegeben: Netzwerk $N = (D = (V, A), b, l, u)$

↳ gerichteter Graph $D = (V, A)$
Knotenbewertung $b\colon V \to \mathbb{R}$ mit $\sum_{v \in V} b(i) = 0$
untere Kapazitäten $l\colon A \to \mathbb{R}_0^+$
obere Kapazitäten $u\colon A \to \mathbb{R}_0^+$

$b(v) > 0$: Quelle $b(v) < 0$: Senke

Allgemeines Flussmodell

Gegeben: Netzwerk $N = (D = (V, A), b, l, u)$

↳ gerichteter Graph $D = (V, A)$
Knotenbewertung $b\colon V \to \mathbb{R}$ mit $\sum_{v \in V} b(i) = 0$
untere Kapazitäten $l\colon A \to \mathbb{R}_0^+$
obere Kapazitäten $u\colon A \to \mathbb{R}_0^+$

$b(v) > 0$: Quelle $b(v) < 0$: Senke

Abbildung $X\colon A \to R_0^+$ heißt *Fluss*, wenn

Allgemeines Flussmodell

Gegeben: Netzwerk $N = (D = (V, A), b, l, u)$

↳ gerichteter Graph $D = (V, A)$
Knotenbewertung $b\colon V \to \mathbb{R}$ mit $\sum_{v \in V} b(i) = 0$
untere Kapazitäten $l\colon A \to \mathbb{R}_0^+$
obere Kapazitäten $u\colon A \to \mathbb{R}_0^+$

$b(v) > 0$: Quelle $b(v) < 0$: Senke

Abbildung $X\colon A \to R_0^+$ heißt *Fluss*, wenn

1. Kapazitätsbedingung:

$$\forall (i,j) \in A : l(i,j) \leq X(i,j) \leq u(i,j)$$

2. Flusserhaltungsbedingung:

$$\forall i \in V \backslash \{s,t\} : \sum_{j:(i,j)\in A} X(i,j) - \sum_{j:(j,i)\in A} X(j,i) = b(i)$$

Allgemeines Flussmodell

Gegeben: Netzwerk $N = (D = (V, A), b, l, u)$

↳ gerichteter Graph $D = (V, A)$
Knotenbewertung $b\colon V \to \mathbb{R}$ mit $\sum_{v \in V} b(i) = 0$
untere Kapazitäten $l\colon A \to \mathbb{R}_0^+$
obere Kapazitäten $u\colon A \to \mathbb{R}_0^+$

$b(v) > 0$: Quelle $b(v) < 0$: Senke

Gesucht: Fluss X sodass

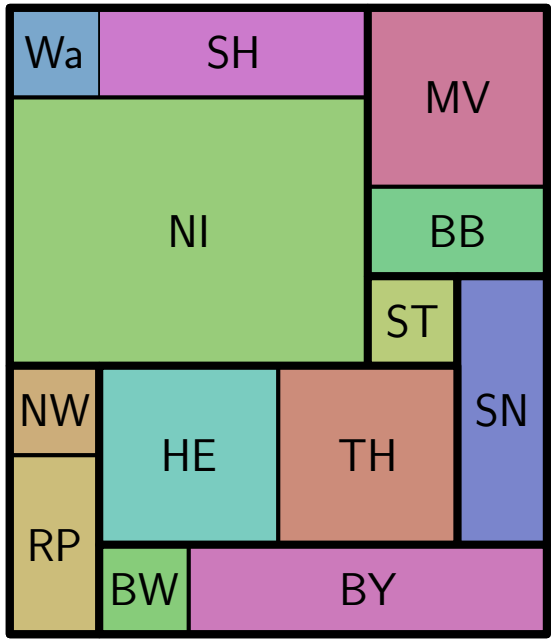
$$cost(X) = \sum_{(i,j) \in A} cost(i, j) \cdot X(i, j)$$

minimiert wird, wobei $cost\colon A \to \mathbb{R}_0^+$ gegebene Kostenfunktion ist.

Aufgabe 3

Gegeben:

- Ein Rechtecksdual $\mathcal{R}$ bestehend aus den Rechtecken $R_1, \ldots, R_n$.
- *Zielbreiten* $w_1, \ldots, w_n \in \mathbb{R}^+$
- *Zielhöhen* $h_1, \ldots, h_n \in \mathbb{R}^+$



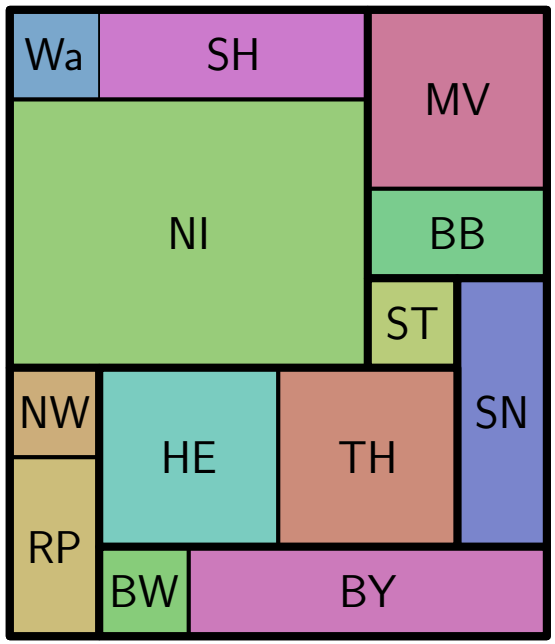
Aufgabe 3

Gegeben:

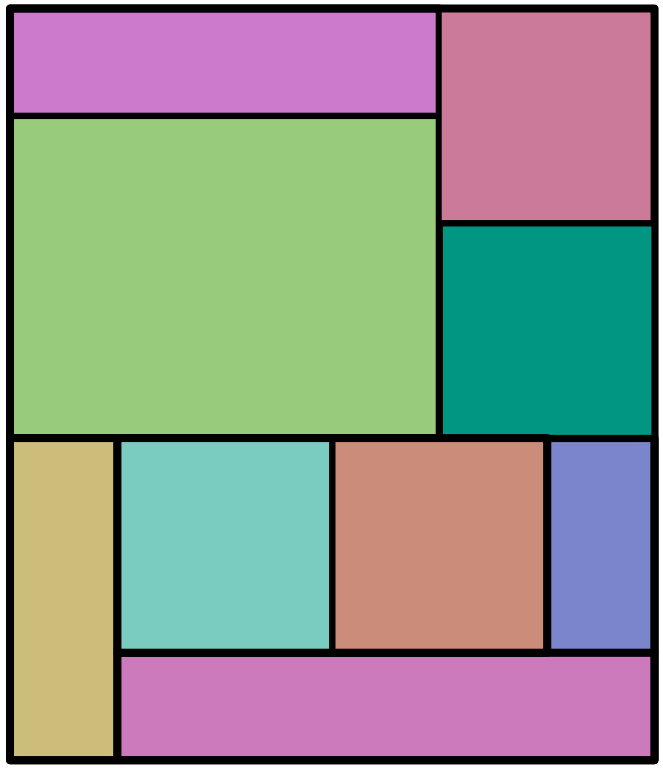
- Ein Rechtecksdual $\mathcal{R}$ bestehend aus den Rechtecken $R_1, \ldots, R_n$.
- *Zielbreiten* $w_1, \ldots, w_n \in \mathbb{R}^+$
- *Zielhöhen* $h_1, \ldots, h_n \in \mathbb{R}^+$

Gesucht: Für jedes Rechteck R_i Breite w'_i und Höhe h'_i sodass

a) die Adjazenzen aus $\mathcal{R}$ erhalten bleiben, und
b) $w'_i \geq w_i$ und $h'_i \geq h_i$ für alle $1 \leq i \leq n$, und
c) $\sum_{i=1}^{n} w'_i + \sum_{i=1}^{n} h'_i$ minimiert wird.



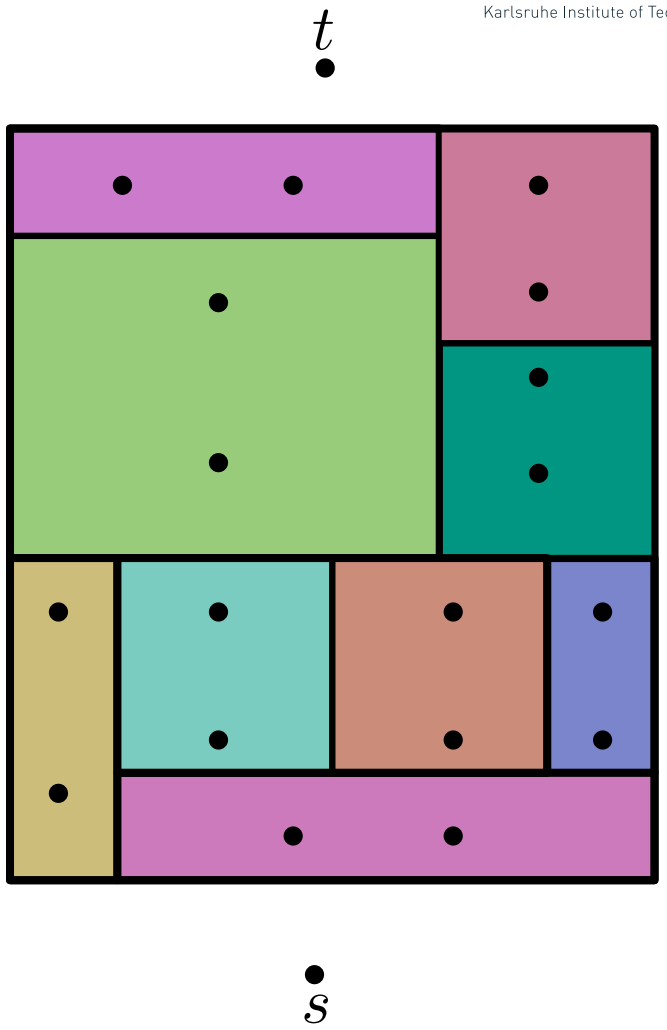
Lösung



Lösung

Idee: Flussnetzwerk N_H für horz. Kanten.

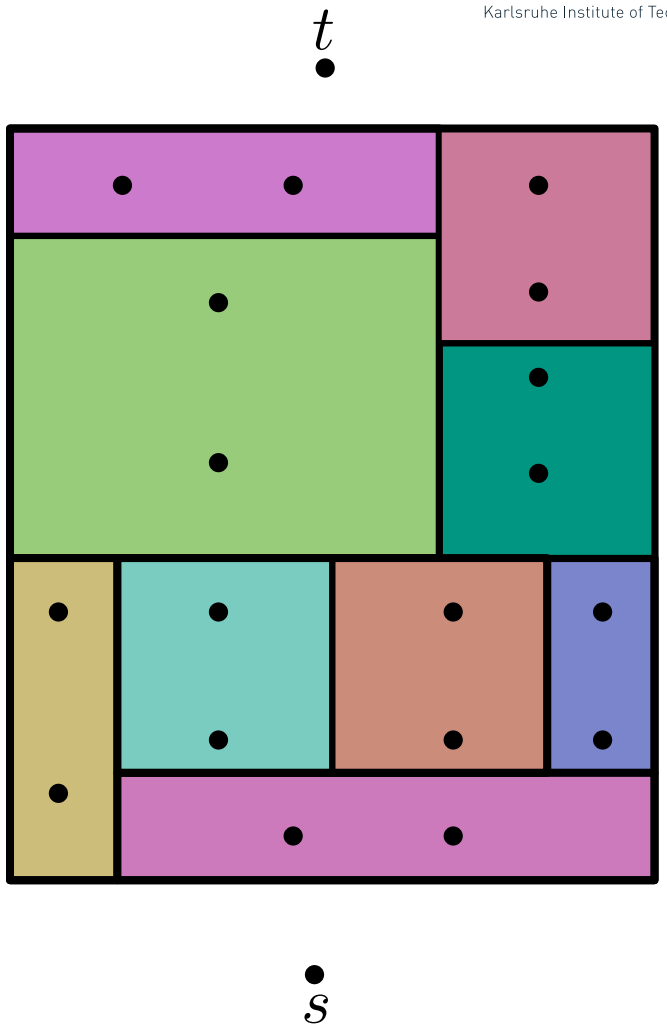
1. Für jedes R_i zwei Knoten u_i, v_i
2. Knoten s und t



Lösung

Idee: Flussnetzwerk N_H für horz. Kanten.

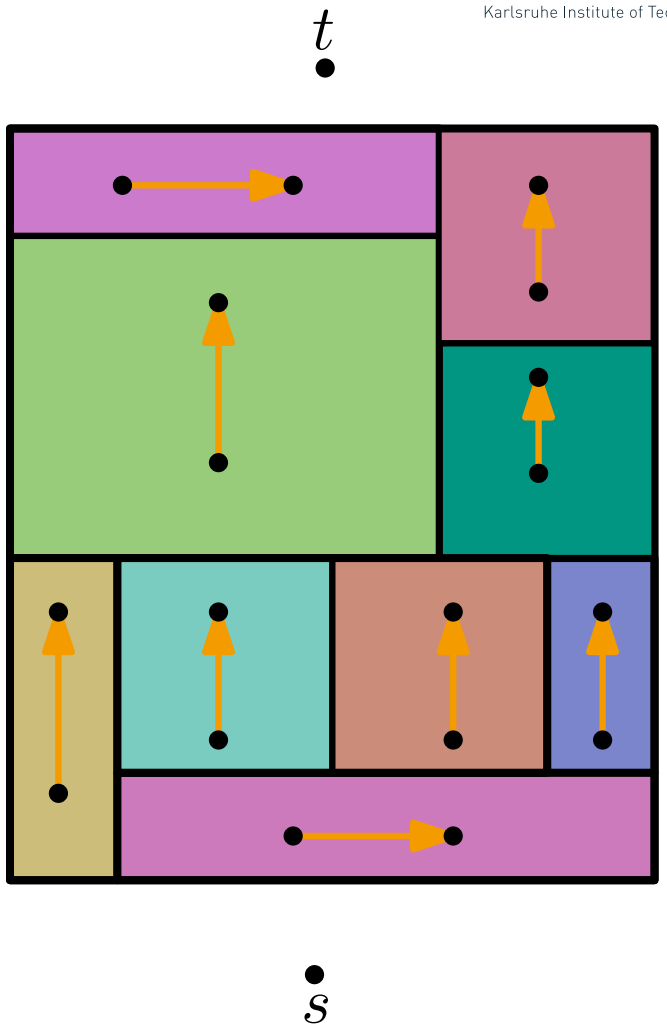
1. Für jedes R_i zwei Knoten u_i, v_i
2. Knoten s und t



Lösung

Idee: Flussnetzwerk N_H für horz. Kanten.

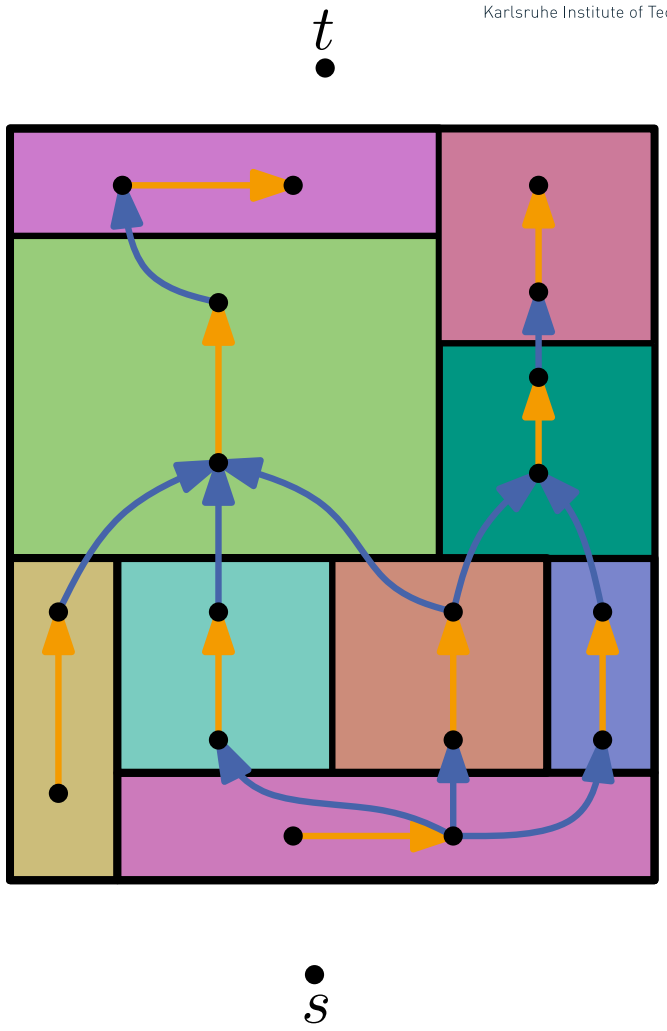
1. Für jedes R_i zwei Knoten u_i, v_i
2. Knoten s und t
3. Kante $e_i = (u_i, v_i)$
   mit $l(e_i) = w_i$, $u(e_i) = \infty$, $cost(e_i) = 1$



Lösung

Idee: Flussnetzwerk N_H für horz. Kanten.

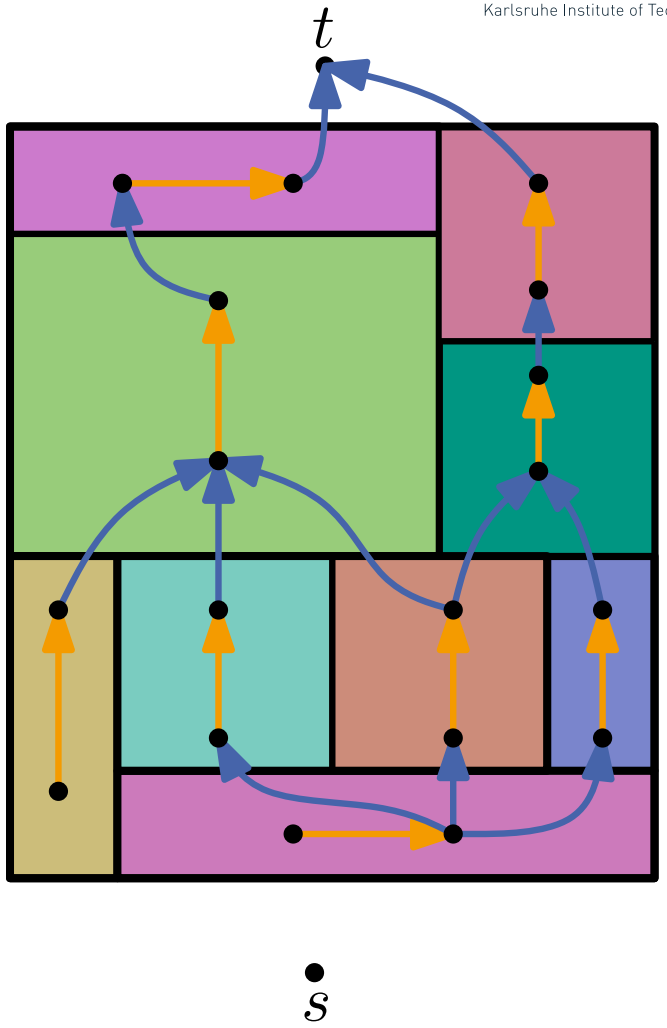
1. Für jedes R_i zwei Knoten u_i, v_i
2. Knoten s und t
3. Kante $e_i = (u_i, v_i)$
   mit $l(e_i) = w_i$, $u(e_i) = \infty$, $cost(e_i) = 1$
4. Kante e zwischen je zwei Knoten v_i und u_j
   getrennt durch horz. Kante (aufwärts).
   $l(e) = 0$, $u(e) = \infty$ und $cost(e) = 0$



Lösung

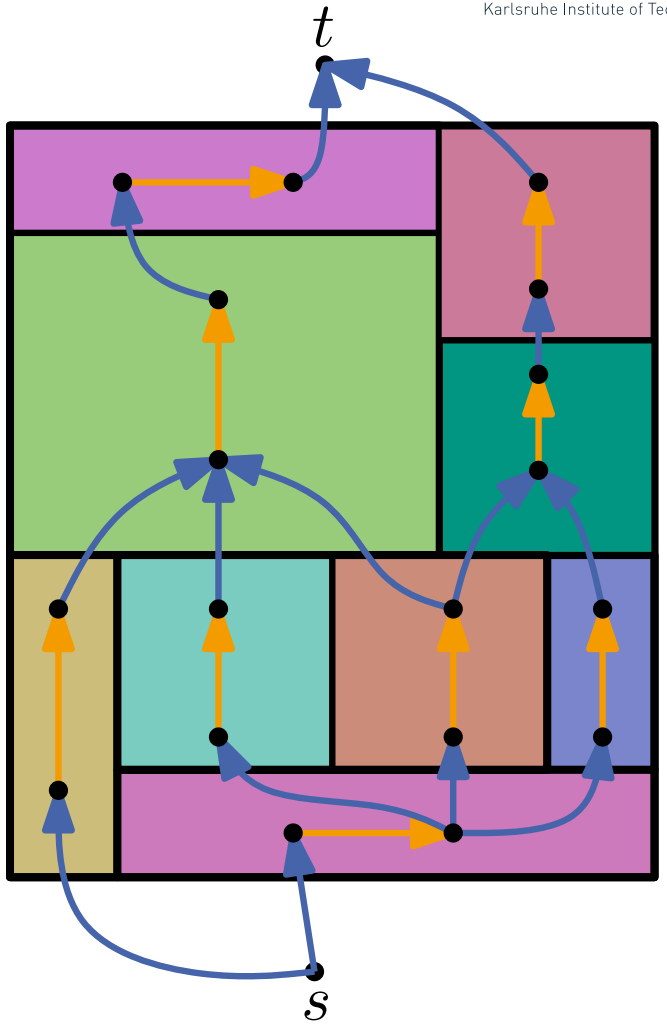
Idee: Flussnetzwerk N_H für horz. Kanten.

1. Für jedes R_i zwei Knoten u_i, v_i
2. Knoten s und t
3. Kante $e_i = (u_i, v_i)$
   mit $l(e_i) = w_i$, $u(e_i) = \infty$, $cost(e_i) = 1$
4. Kante e zwischen je zwei Knoten v_i und u_j
   getrennt durch horz. Kante (aufwärts).
   $l(e) = 0$, $u(e) = \infty$ und $cost(e) = 0$
5. Kante e zwischen Knoten der oberen Reihe
   Rechtecke und t.
   $l(e) = 0$, $u(e) = \infty$ und $cost(e) = 0$



Lösung

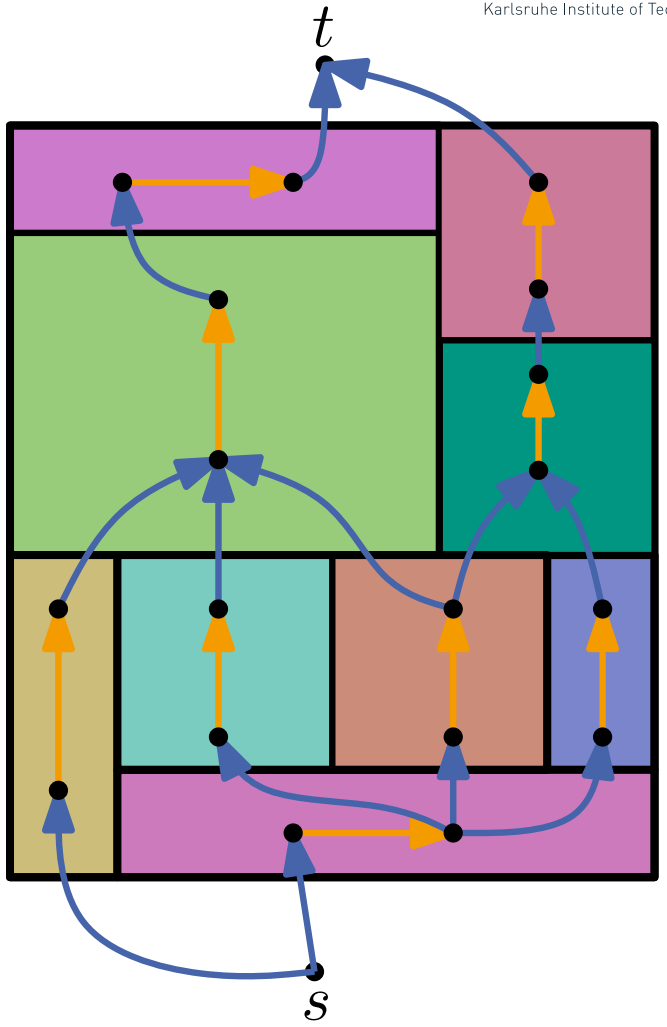
Idee: Flussnetzwerk N_H für horz. Kanten.



1. Für jedes R_i zwei Knoten u_i, v_i
2. Knoten s und t
3. Kante $e_i = (u_i, v_i)$
   mit $l(e_i) = w_i$, $u(e_i) = \infty$, $cost(e_i) = 1$
4. Kante e zwischen je zwei Knoten v_i und u_j
   getrennt durch horz. Kante (aufwärts).
   $l(e) = 0$, $u(e) = \infty$ und $cost(e) = 0$
5. Kante e zwischen Knoten der oberen Reihe
   Rechtecke und t.
   $l(e) = 0$, $u(e) = \infty$ und $cost(e) = 0$
6. Kante e zwischen s und Knoten der unteren Reihe Rechtecke.
   $l(e) = 0$, $u(e) = \infty$, $cost(e) = 0$

Lösung

Idee: Flussnetzwerk N_H für horz. Kanten.

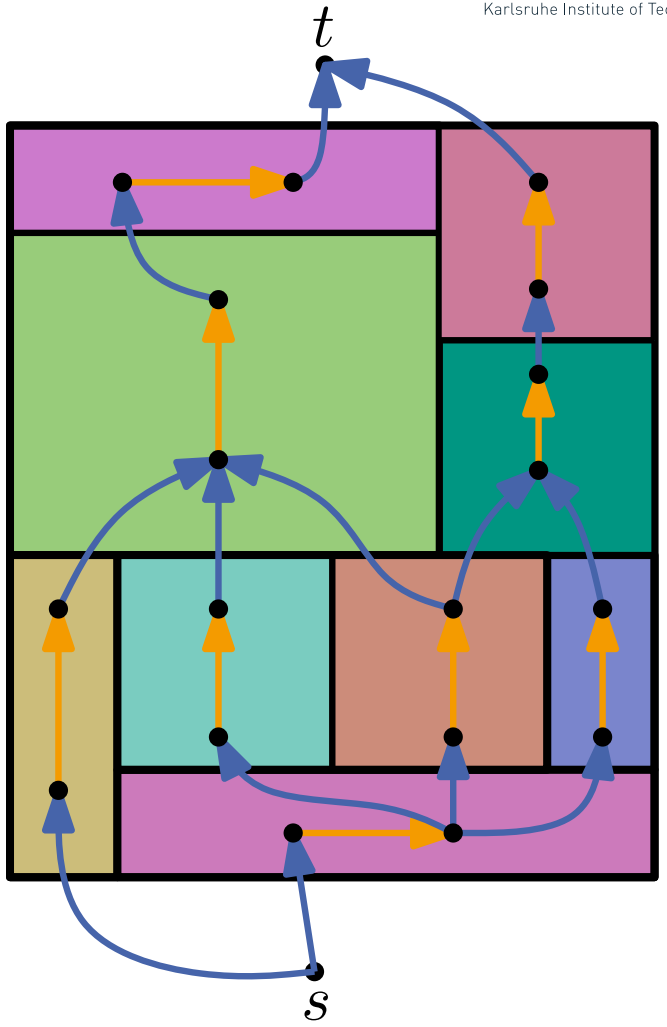


1. Für jedes R_i zwei Knoten u_i, v_i
2. Knoten s und t
3. Kante $e_i = (u_i, v_i)$
   mit $l(e_i) = w_i$, $u(e_i) = \infty$, $cost(e_i) = 1$
4. Kante e zwischen je zwei Knoten v_i und u_j
   getrennt durch horz. Kante (aufwärts).
   $l(e) = 0$, $u(e) = \infty$ und $cost(e) = 0$
5. Kante e zwischen Knoten der oberen Reihe Rechtecke und t.
   $l(e) = 0$, $u(e) = \infty$ und $cost(e) = 0$
6. Kante e zwischen s und Knoten der unteren Reihe Rechtecke.
   $l(e) = 0$, $u(e) = \infty$, $cost(e) = 0$

Wie Knotenbewertung $b\colon V \to \mathbb{R}_0^+$ festlegen?

Lösung

Idee: Flussnetzwerk N_H für horz. Kanten.



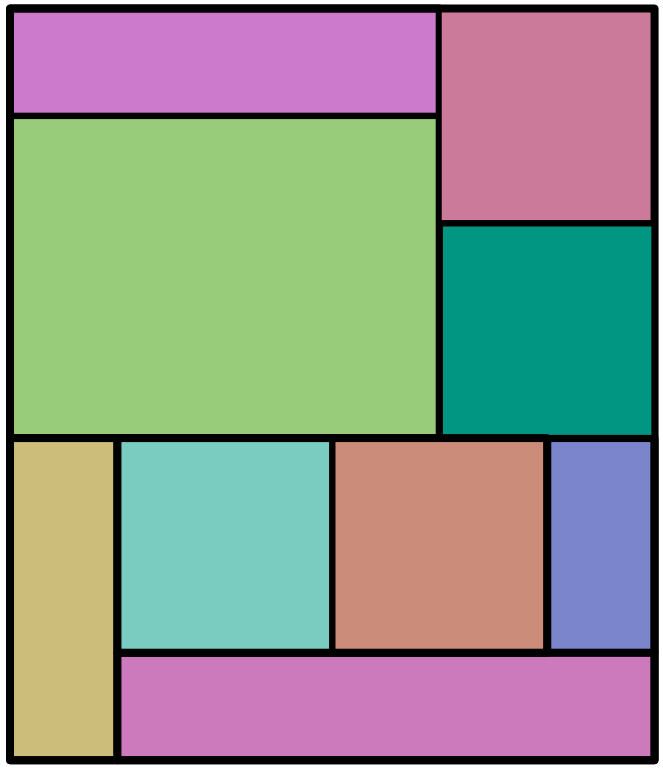
1. Für jedes R_i zwei Knoten u_i, v_i
2. Knoten s und t
3. Kante $e_i = (u_i, v_i)$
   mit $l(e_i) = w_i$, $u(e_i) = \infty$, $cost(e_i) = 1$
4. Kante e zwischen je zwei Knoten v_i und u_j
   getrennt durch horz. Kante (aufwärts).
   $l(e) = 0$, $u(e) = \infty$ und $cost(e) = 0$
5. Kante e zwischen Knoten der oberen Reihe
   Rechtecke und t.
   $l(e) = 0$, $u(e) = \infty$ und $cost(e) = 0$
6. Kante e zwischen s und Knoten der unteren Reihe Rechtecke.
   $l(e) = 0$, $u(e) = \infty$, $cost(e) = 0$

$b(s) =$ gewünschte Breite $\quad b(t) = -b(s) \quad b(v) = 0$ für restl. Knoten v

Lösung

Idee: Flussnetzwerk N_H für horz. Kanten.

Analog: Flussnetzwerk N_V für vert. Kanten.

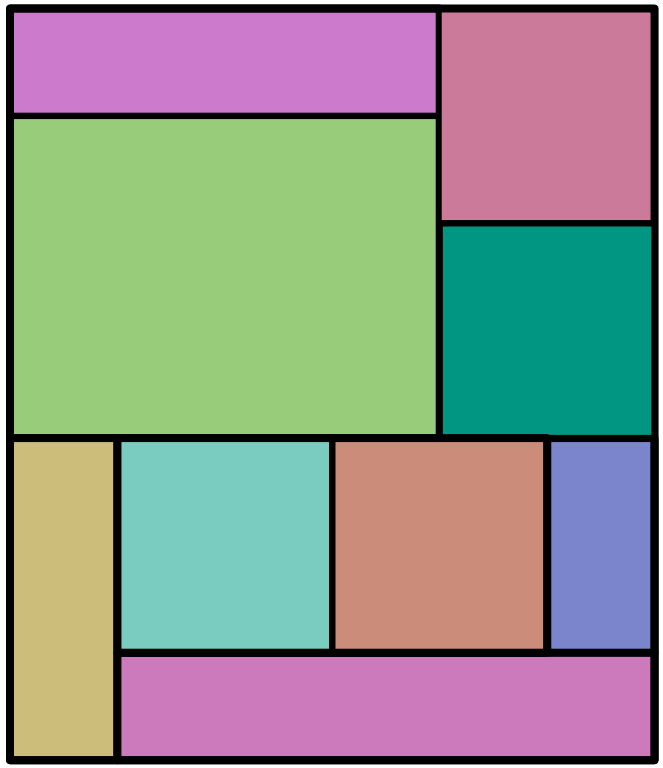


Lösung

Idee: Flussnetzwerk N_H für horz. Kanten.

Analog: Flussnetzwerk N_V für vert. Kanten.

Löse Flussproblem mit minimalen Kosten auf Netzwerk bestehend aus den Komponenten N_H und N_V.

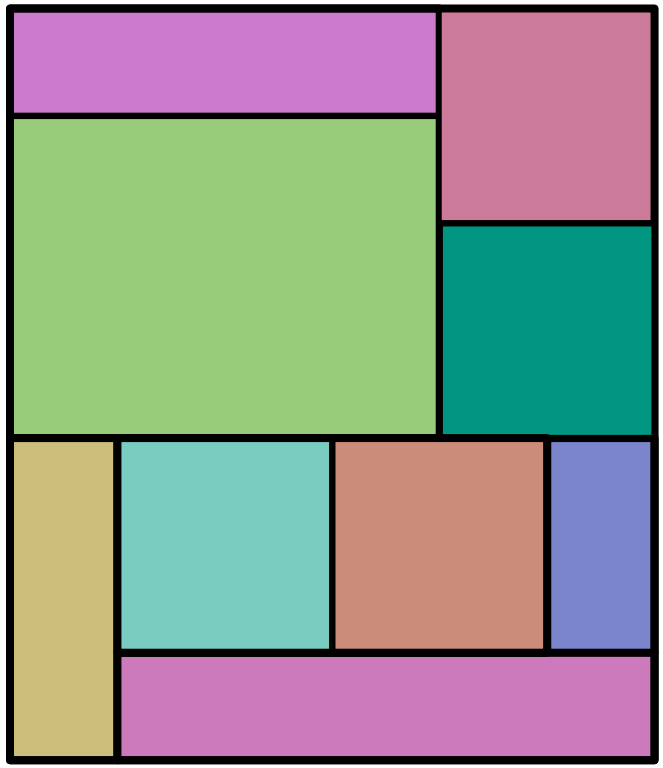


Lösung

Idee: Flussnetzwerk N_H für horz. Kanten.

Analog: Flussnetzwerk N_V für vert. Kanten.

Löse Flussproblem mit minimalen Kosten auf Netzwerk bestehend aus den Komponenten N_H und N_V.



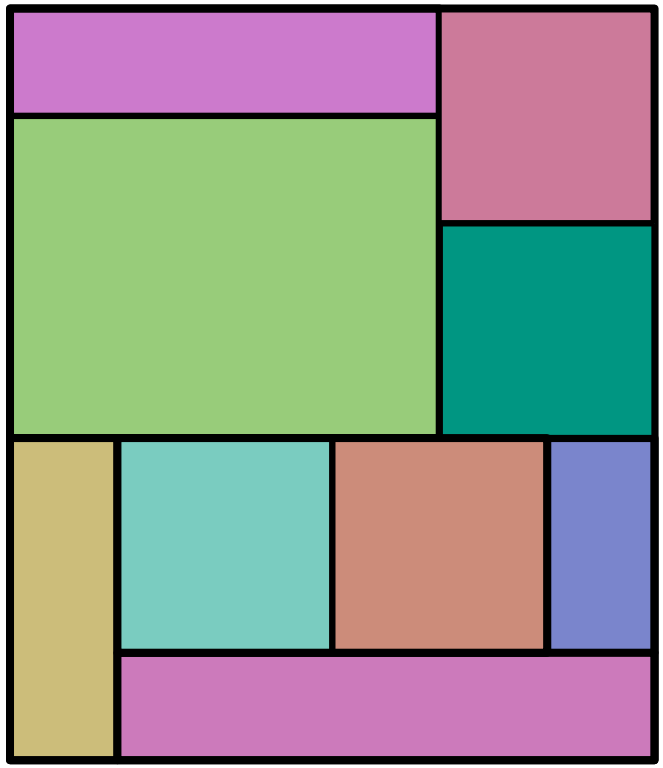
Kann ähnliches Netzwerk auch für Flächen konstruiert werden?

Lösung

Idee: Flussnetzwerk N_H für horz. Kanten.

Analog: Flussnetzwerk N_V für vert. Kanten.

Löse Flussproblem mit minimalen Kosten auf Netzwerk bestehend aus den Komponenten N_H und N_V.



Kann ähnliches Netzwerk auch für Flächen konstruiert werden?

Nein, denn eine reine Flächenzuweisung zu Rechtecken bestimmt die Form der Rechtecke nicht.

Rektilineare Kartogramme

Ein Ausflug in die Theorie: Welche Polygonkomplexität braucht man, um jede Flächenzuweisung als rektilineares Kartogramm realisieren zu können?

Rektilineare Kartogramme

Ein Ausflug in die Theorie: Welche Polygonkomplexität braucht man, um jede Flächenzuweisung als rektilineares Kartogramm realisieren zu können?

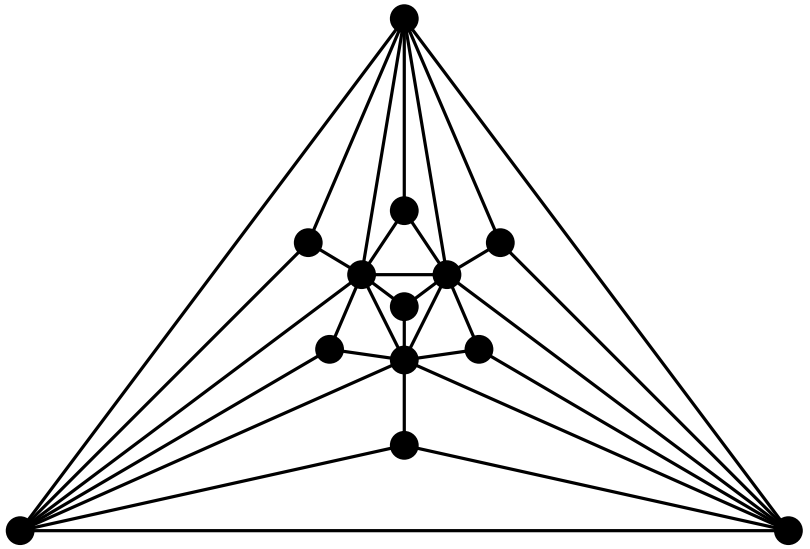
Satz: Es gibt planare triangulierte Graphen, die mindestens Komplexität 8 zur Kontaktrepräsentation mit rektilinearen Polygonen benötigen. [Yeap, Sarrafzadeh '93]

Rektilineare Kartogramme

Ein Ausflug in die Theorie: Welche Polygonkomplexität braucht man, um jede Flächenzuweisung als rektilineares Kartogramm realisieren zu können?

Satz: Es gibt planare triangulierte Graphen, die mindestens Komplexität 8 zur Kontaktrepräsentation mit rektilinearen Polygonen benötigen. [Yeap, Sarrafzadeh '93]

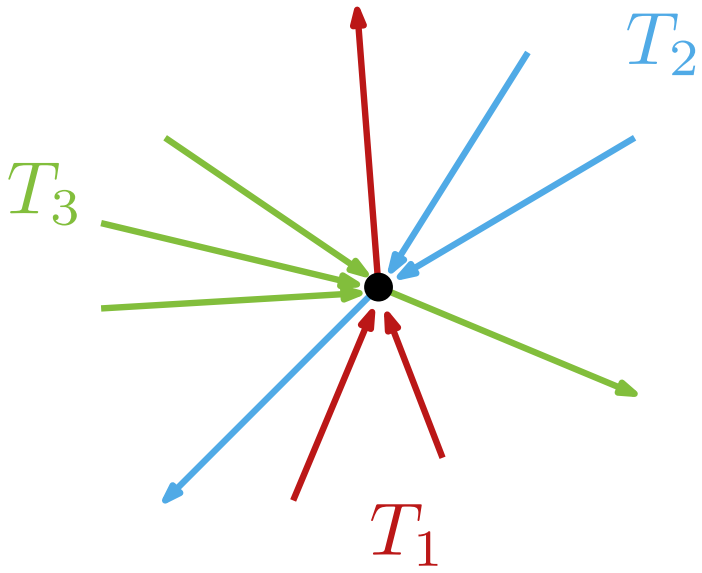
Beweis:



Schnyder Realizer

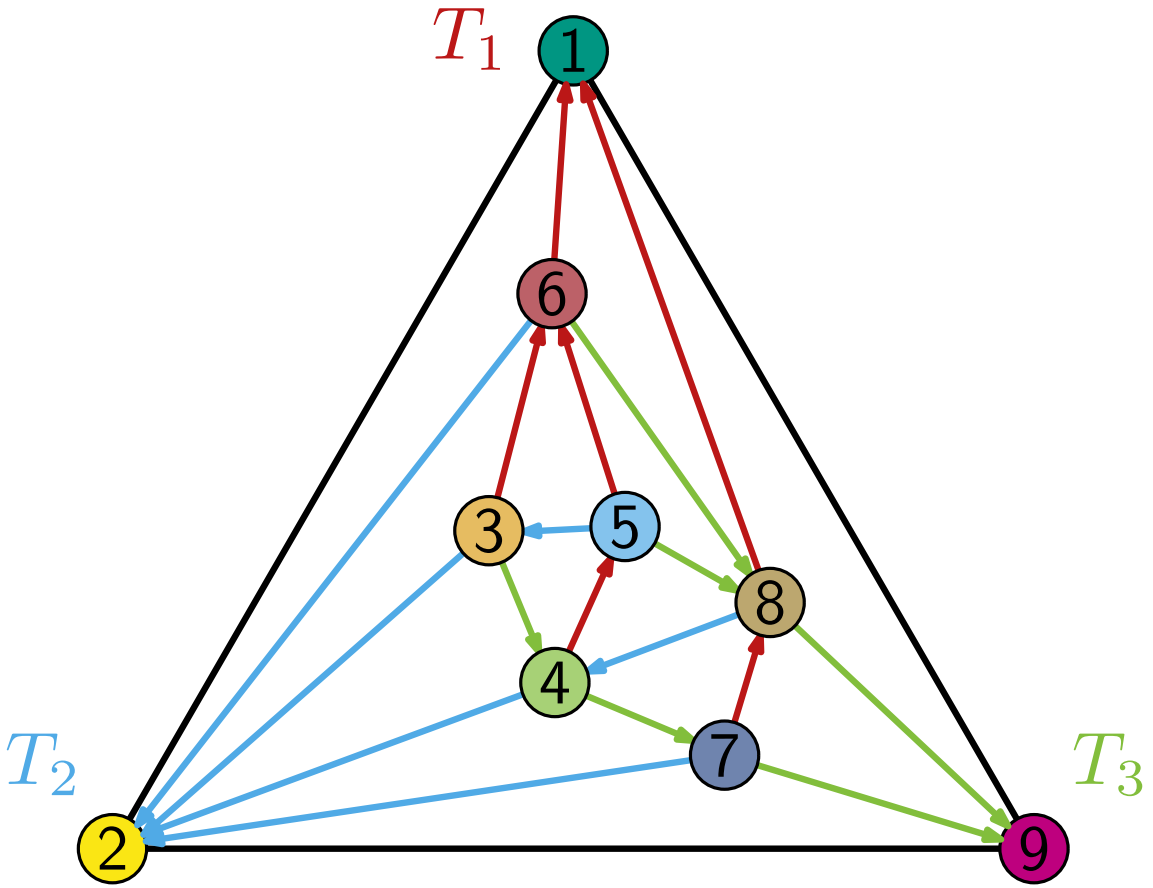
Sei G ein trianguliert planarer Graph. Ein **Schnyder Realizer** partitioniert die internen Kanten in drei Mengen T_1, T_2, T_3 von gerichteten Kanten, so dass

- jeder innere Knoten v hat genau eine Kante in jedem T_i^{out}
- die Ordnung der Kanten um jeden Knoten v im GUZS ist $T_1^{\text{in}}, T_3^{\text{out}}, T_2^{\text{in}}, T_1^{\text{out}}, T_3^{\text{in}}, T_2^{\text{out}}$



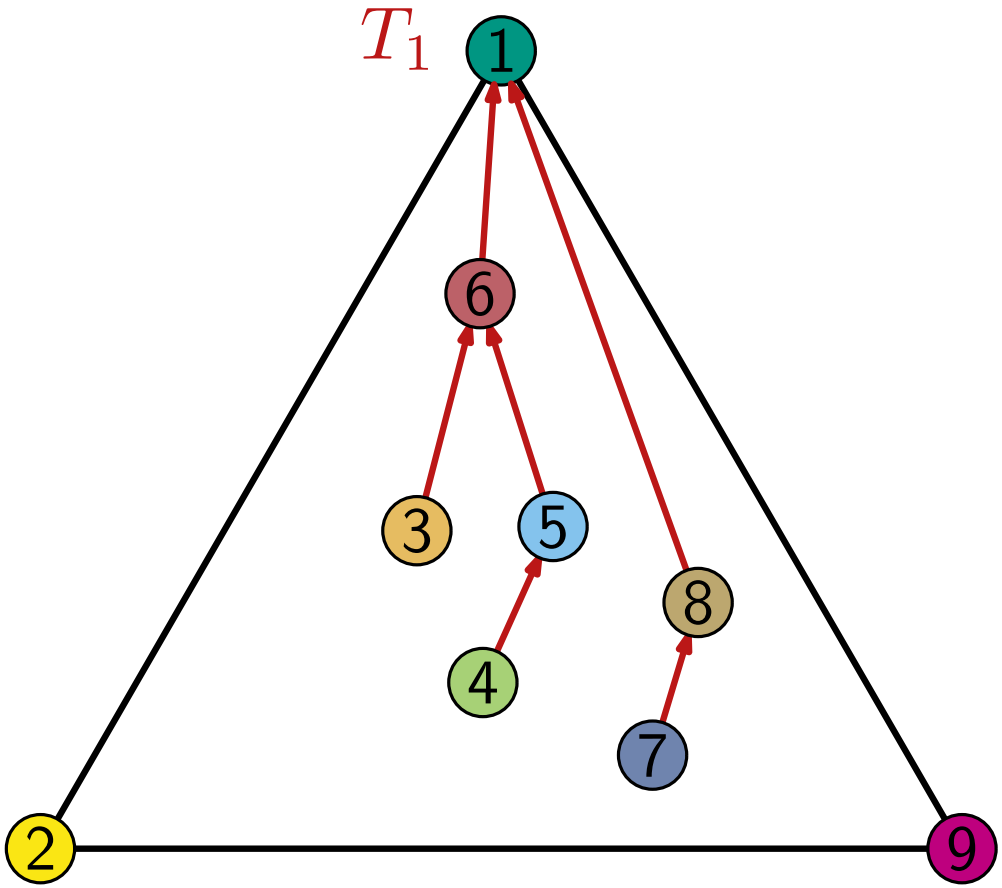
Schnyder Realizer

Satz: Jede Menge T_i $(i = 1, 2, 3)$ ist ein Spannbaum aller inneren Knoten und eines äußeren Knotens.
Jeder triangulierte Graph besitzt einen Schnyder Realizer und dieser kann in $O(n)$ Zeit berechnet werden.



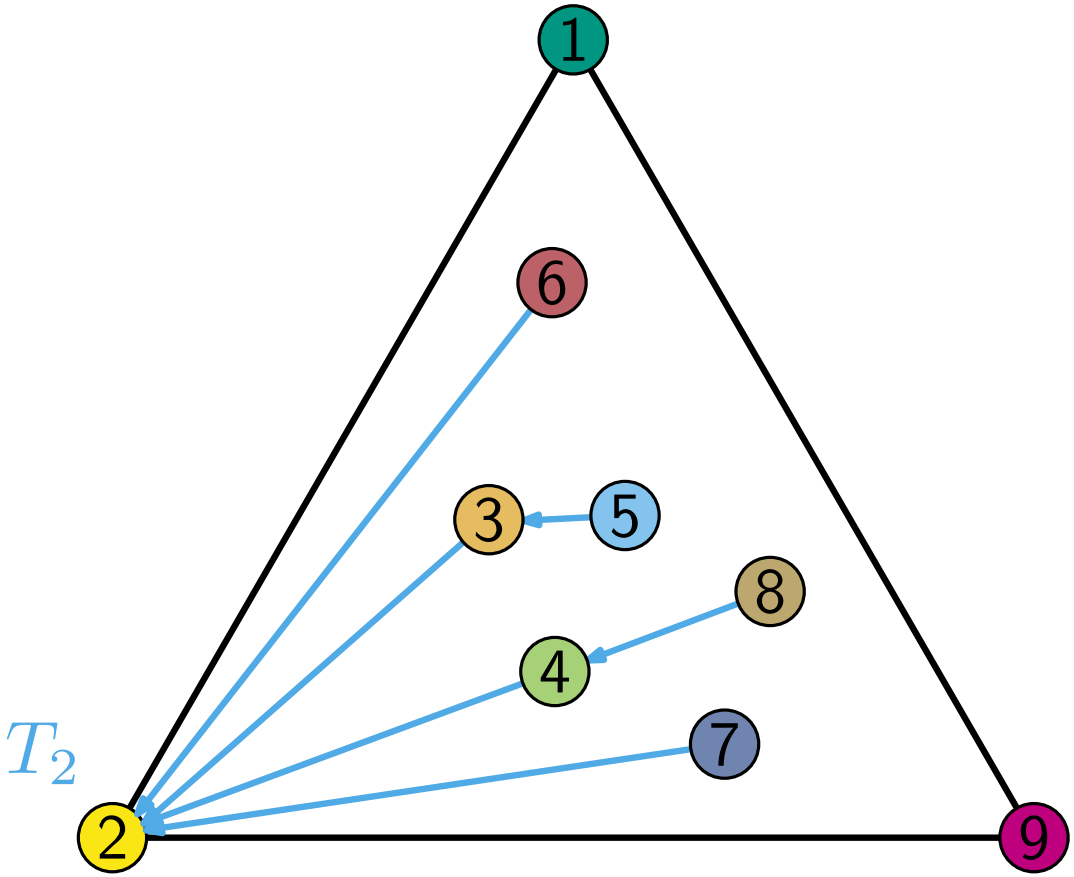
Schnyder Realizer

Satz: Jede Menge T_i $(i = 1, 2, 3)$ ist ein Spannbaum aller inneren Knoten und eines äußeren Knotens.
Jeder triangulierte Graph besitzt einen Schnyder Realizer und dieser kann in $O(n)$ Zeit berechnet werden.



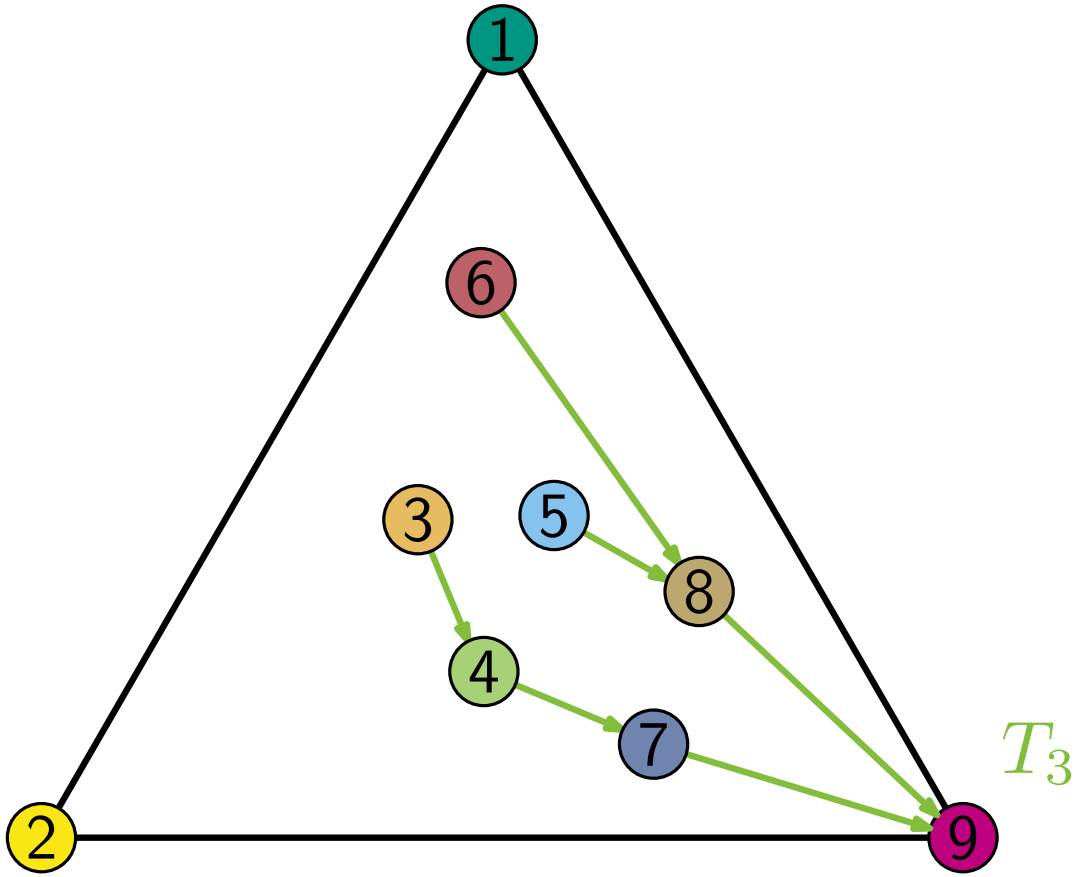
Schnyder Realizer

Satz: Jede Menge T_i $(i = 1, 2, 3)$ ist ein Spannbaum aller inneren Knoten und eines äußeren Knotens.
Jeder triangulierte Graph besitzt einen Schnyder Realizer und dieser kann in $O(n)$ Zeit berechnet werden.



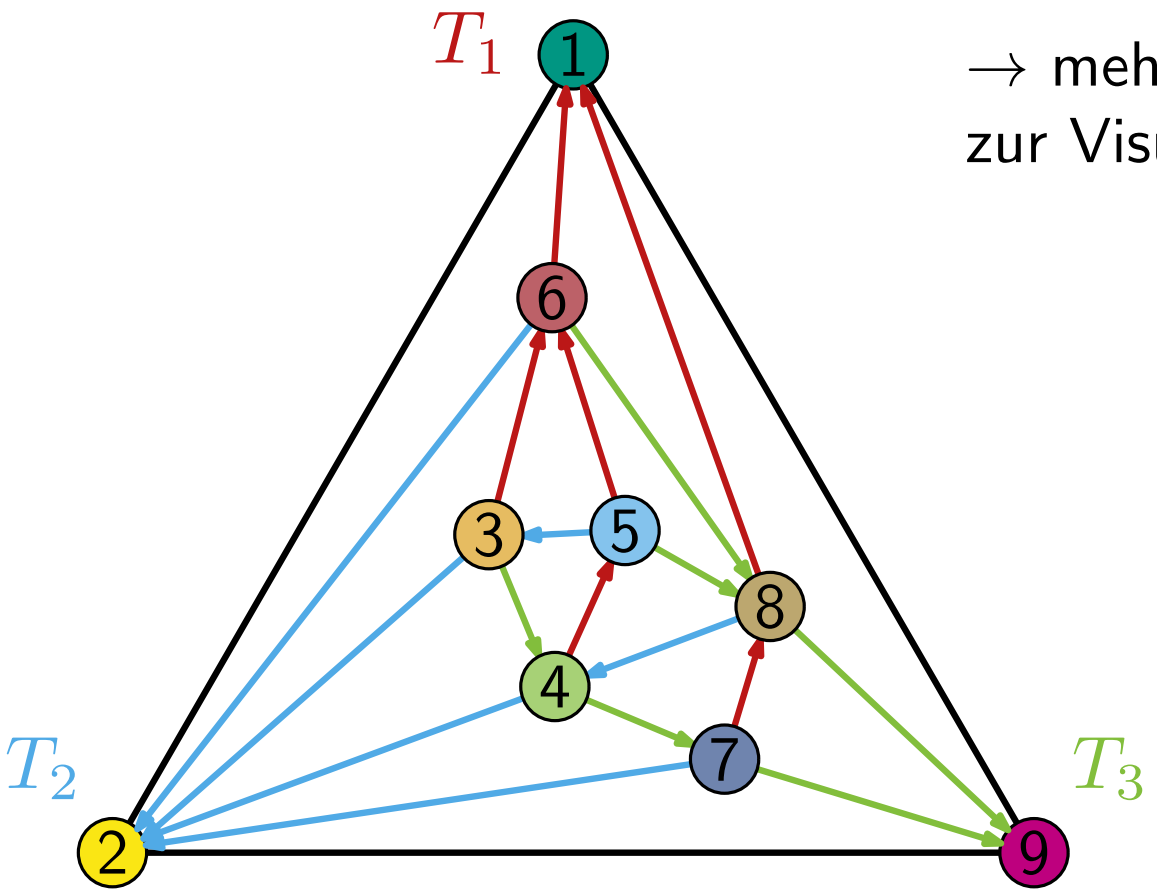
Schnyder Realizer

Satz: Jede Menge T_i $(i = 1, 2, 3)$ ist ein Spannbaum aller inneren Knoten und eines äußeren Knotens.
Jeder triangulierte Graph besitzt einen Schnyder Realizer und dieser kann in $O(n)$ Zeit berechnet werden.



Schnyder Realizer

Satz: Jede Menge T_i $(i = 1, 2, 3)$ ist ein Spannbaum aller inneren Knoten und eines äußeren Knotens.
Jeder triangulierte Graph besitzt einen Schnyder Realizer und dieser kann in $O(n)$ Zeit berechnet werden.



$\rightarrow$ mehr dazu in der VL Algorithmen zur Visualisierung von Graphen

Kanonische Ordnung

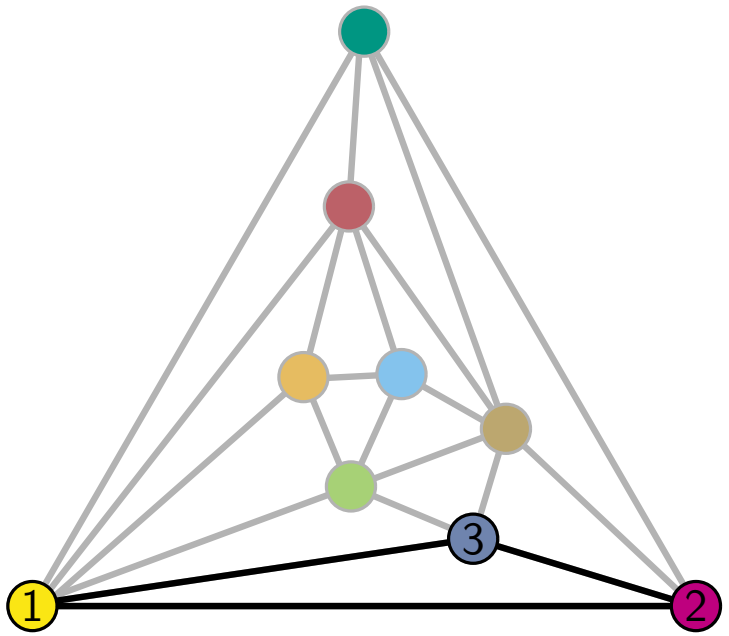
Die **kanonische Ordnung** der Knoten eines triangulierten planaren Graphen G mit den drei äußeren Knoten u, v, w im GUZS ist eine Ordnung $v_1 = u, v_2 = v, v_3, \ldots, v_n = w$, so dass für jedes i $(4 \leq i \leq n)$ gilt:

- Graph $G_{i-1} \subseteq G$ induziert durch $v_1, \ldots, v_{i-1}$ ist 2-fach zshgd. und äußere Facette ist begrenzt durch Kreis C_{i-1}, der $v_1 v_2$ enthält
- Knoten v_i liegt in der äußeren Facette von G_{i-1} und seine Nachbarn in G_{i-1} bilden zusammenhängendes Intervall im Pfad $C_{i-1} \setminus v_1 v_2$

Kanonische Ordnung

Die **kanonische Ordnung** der Knoten eines triangulierten planaren Graphen G mit den drei äußeren Knoten u, v, w im GUZS ist eine Ordnung $v_1 = u, v_2 = v, v_3, \ldots, v_n = w$, so dass für jedes i $(4 \leq i \leq n)$ gilt:

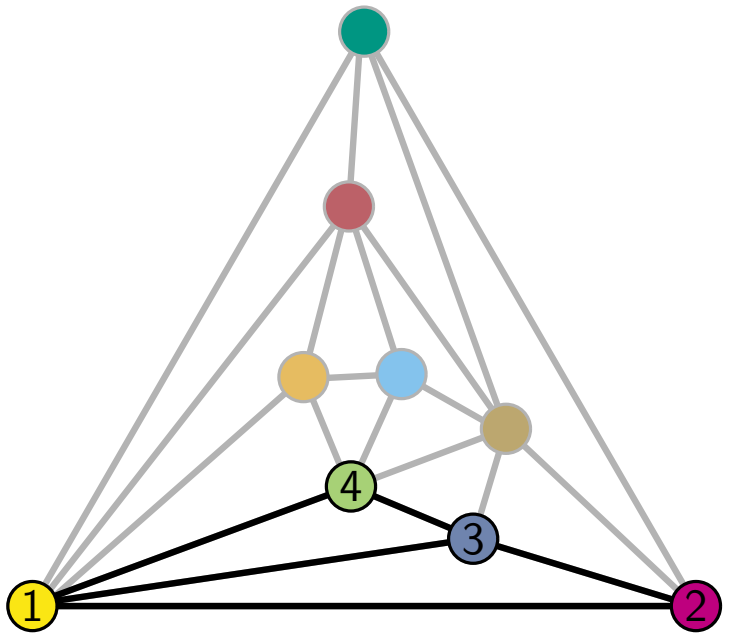
- Graph $G_{i-1} \subseteq G$ induziert durch $v_1, \ldots, v_{i-1}$ ist 2-fach zshgd. und äußere Facette ist begrenzt durch Kreis C_{i-1}, der $v_1 v_2$ enthält
- Knoten v_i liegt in der äußeren Facette von G_{i-1} und seine Nachbarn in G_{i-1} bilden zusammenhängendes Intervall im Pfad $C_{i-1} \setminus v_1 v_2$



Kanonische Ordnung

Die **kanonische Ordnung** der Knoten eines triangulierten planaren Graphen G mit den drei äußeren Knoten u, v, w im GUZS ist eine Ordnung $v_1 = u, v_2 = v, v_3, \ldots, v_n = w$, so dass für jedes i $(4 \leq i \leq n)$ gilt:

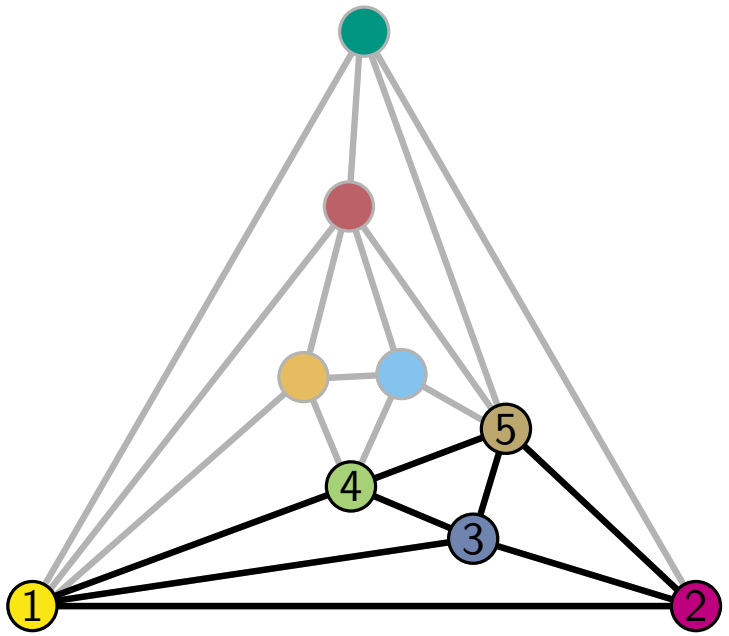
- Graph $G_{i-1} \subseteq G$ induziert durch $v_1, \ldots, v_{i-1}$ ist 2-fach zshgd. und äußere Facette ist begrenzt durch Kreis C_{i-1}, der $v_1 v_2$ enthält
- Knoten v_i liegt in der äußeren Facette von G_{i-1} und seine Nachbarn in G_{i-1} bilden zusammenhängendes Intervall im Pfad $C_{i-1} \setminus v_1 v_2$



Kanonische Ordnung

Die **kanonische Ordnung** der Knoten eines triangulierten planaren Graphen G mit den drei äußeren Knoten u, v, w im GUZS ist eine Ordnung $v_1 = u, v_2 = v, v_3, \ldots, v_n = w$, so dass für jedes i $(4 \leq i \leq n)$ gilt:

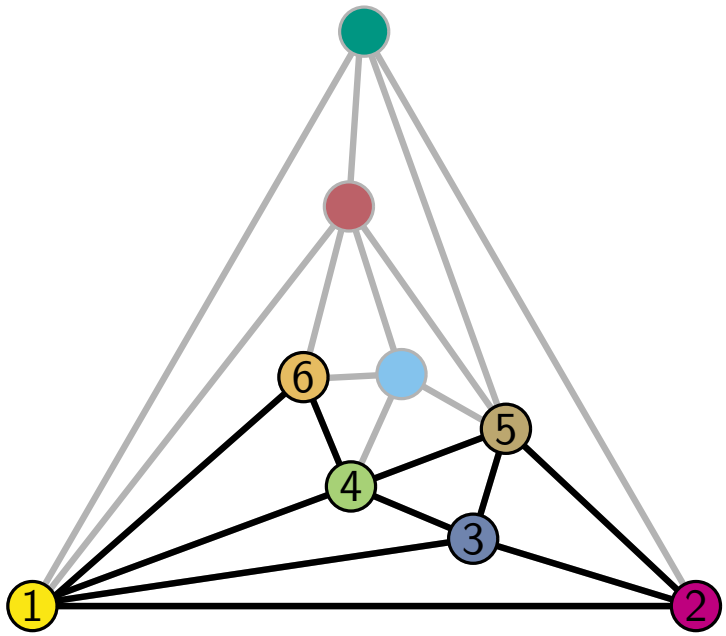
- Graph $G_{i-1} \subseteq G$ induziert durch $v_1, \ldots, v_{i-1}$ ist 2-fach zshgd. und äußere Facette ist begrenzt durch Kreis C_{i-1}, der $v_1 v_2$ enthält
- Knoten v_i liegt in der äußeren Facette von G_{i-1} und seine Nachbarn in G_{i-1} bilden zusammenhängendes Intervall im Pfad $C_{i-1} \setminus v_1 v_2$



Kanonische Ordnung

Die **kanonische Ordnung** der Knoten eines triangulierten planaren Graphen G mit den drei äußeren Knoten u, v, w im GUZS ist eine Ordnung $v_1 = u, v_2 = v, v_3, \ldots, v_n = w$, so dass für jedes i $(4 \leq i \leq n)$ gilt:

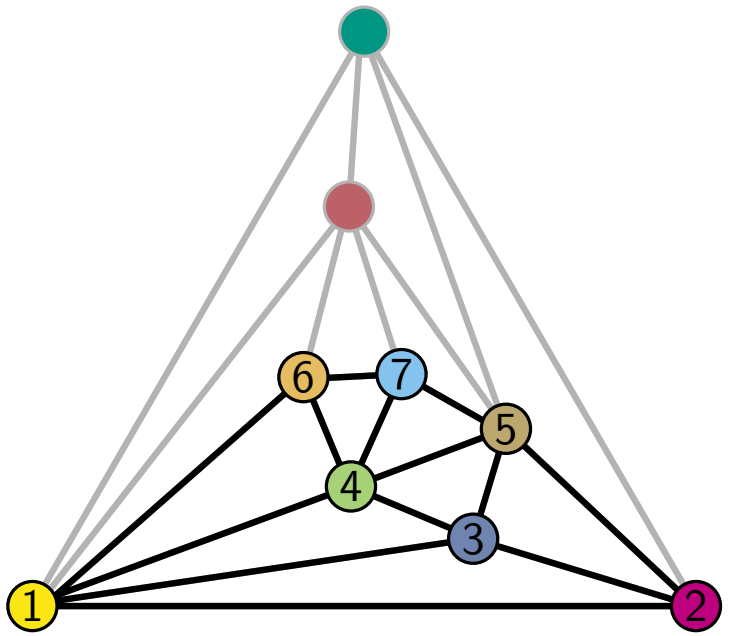
- Graph $G_{i-1} \subseteq G$ induziert durch $v_1, \ldots, v_{i-1}$ ist 2-fach zshgd. und äußere Facette ist begrenzt durch Kreis C_{i-1}, der $v_1 v_2$ enthält
- Knoten v_i liegt in der äußeren Facette von G_{i-1} und seine Nachbarn in G_{i-1} bilden zusammenhängendes Intervall im Pfad $C_{i-1} \setminus v_1 v_2$



Kanonische Ordnung

Die **kanonische Ordnung** der Knoten eines triangulierten planaren Graphen G mit den drei äußeren Knoten u, v, w im GUZS ist eine Ordnung $v_1 = u, v_2 = v, v_3, \ldots, v_n = w$, so dass für jedes i $(4 \leq i \leq n)$ gilt:

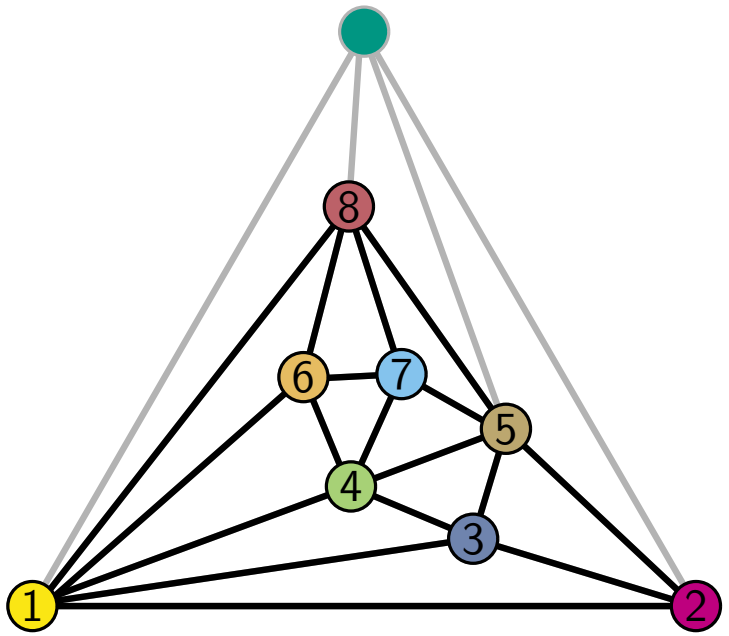
- Graph $G_{i-1} \subseteq G$ induziert durch $v_1, \ldots, v_{i-1}$ ist 2-fach zshgd. und äußere Facette ist begrenzt durch Kreis C_{i-1}, der $v_1 v_2$ enthält
- Knoten v_i liegt in der äußeren Facette von G_{i-1} und seine Nachbarn in G_{i-1} bilden zusammenhängendes Intervall im Pfad $C_{i-1} \setminus v_1 v_2$



Kanonische Ordnung

Die **kanonische Ordnung** der Knoten eines triangulierten planaren Graphen G mit den drei äußeren Knoten u, v, w im GUZS ist eine Ordnung $v_1 = u, v_2 = v, v_3, \ldots, v_n = w$, so dass für jedes i $(4 \leq i \leq n)$ gilt:

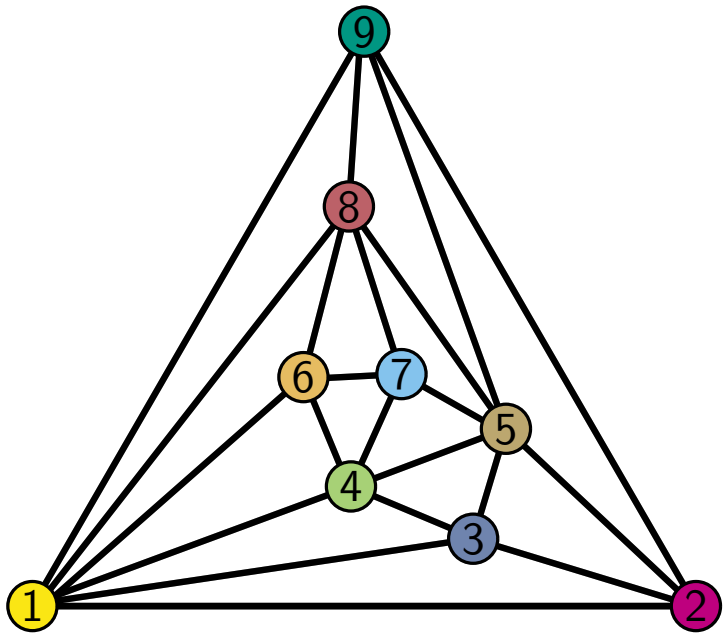
- Graph $G_{i-1} \subseteq G$ induziert durch $v_1, \ldots, v_{i-1}$ ist 2-fach zshgd. und äußere Facette ist begrenzt durch Kreis C_{i-1}, der $v_1 v_2$ enthält
- Knoten v_i liegt in der äußeren Facette von G_{i-1} und seine Nachbarn in G_{i-1} bilden zusammenhängendes Intervall im Pfad $C_{i-1} \setminus v_1 v_2$



Kanonische Ordnung

Die **kanonische Ordnung** der Knoten eines triangulierten planaren Graphen G mit den drei äußeren Knoten u, v, w im GUZS ist eine Ordnung $v_1 = u, v_2 = v, v_3, \ldots, v_n = w$, so dass für jedes i $(4 \leq i \leq n)$ gilt:

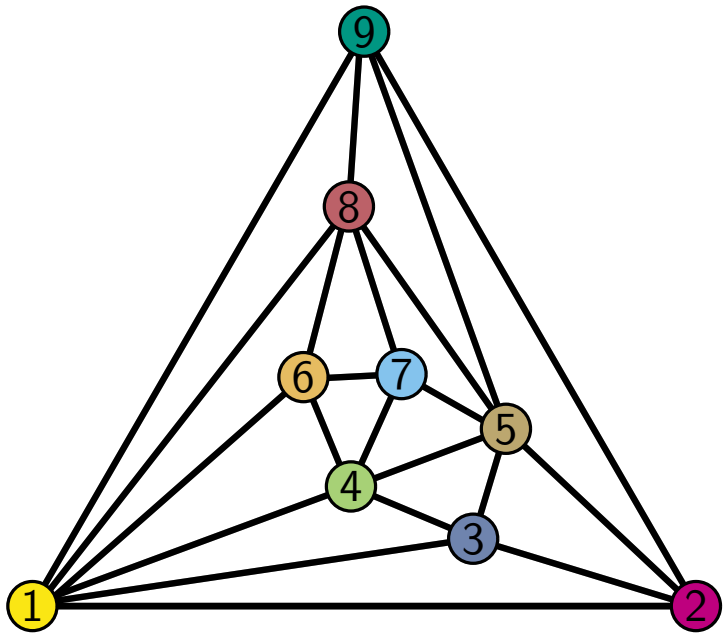
- Graph $G_{i-1} \subseteq G$ induziert durch $v_1, \ldots, v_{i-1}$ ist 2-fach zshgd. und äußere Facette ist begrenzt durch Kreis C_{i-1}, der $v_1 v_2$ enthält
- Knoten v_i liegt in der äußeren Facette von G_{i-1} und seine Nachbarn in G_{i-1} bilden zusammenhängendes Intervall im Pfad $C_{i-1} \setminus v_1 v_2$



Kanonische Ordnung

Die **kanonische Ordnung** der Knoten eines triangulierten planaren Graphen G mit den drei äußeren Knoten u, v, w im GUZS ist eine Ordnung $v_1 = u, v_2 = v, v_3, \ldots, v_n = w$, so dass für jedes i $(4 \leq i \leq n)$ gilt:

- Graph $G_{i-1} \subseteq G$ induziert durch $v_1, \ldots, v_{i-1}$ ist 2-fach zshgd. und äußere Facette ist begrenzt durch Kreis C_{i-1}, der $v_1 v_2$ enthält
- Knoten v_i liegt in der äußeren Facette von G_{i-1} und seine Nachbarn in G_{i-1} bilden zusammenhängendes Intervall im Pfad $C_{i-1} \setminus v_1 v_2$



Satz: Jeder triangulierte Graph G besitzt eine kanonische Ordnung; sie kann in $O(n)$ Zeit berechnet werden.

$\rightarrow$ mehr dazu in der VL Algorithmen zur Visualisierung von Graphen

Kanonische Ordnung & Schnyder Realizer

Karlsruhe Institute of Technology

Aus der Vorlesung: Eine kanonische Ordnung der Knoten von G definiert einen Schnyder Realizer von G, indem jedem Knoten v_i drei ausgehende Kanten zum ersten und letzten Knoten in C_{i-1} und zum Nachfolger mit höchstem Index zugeordnet werden.

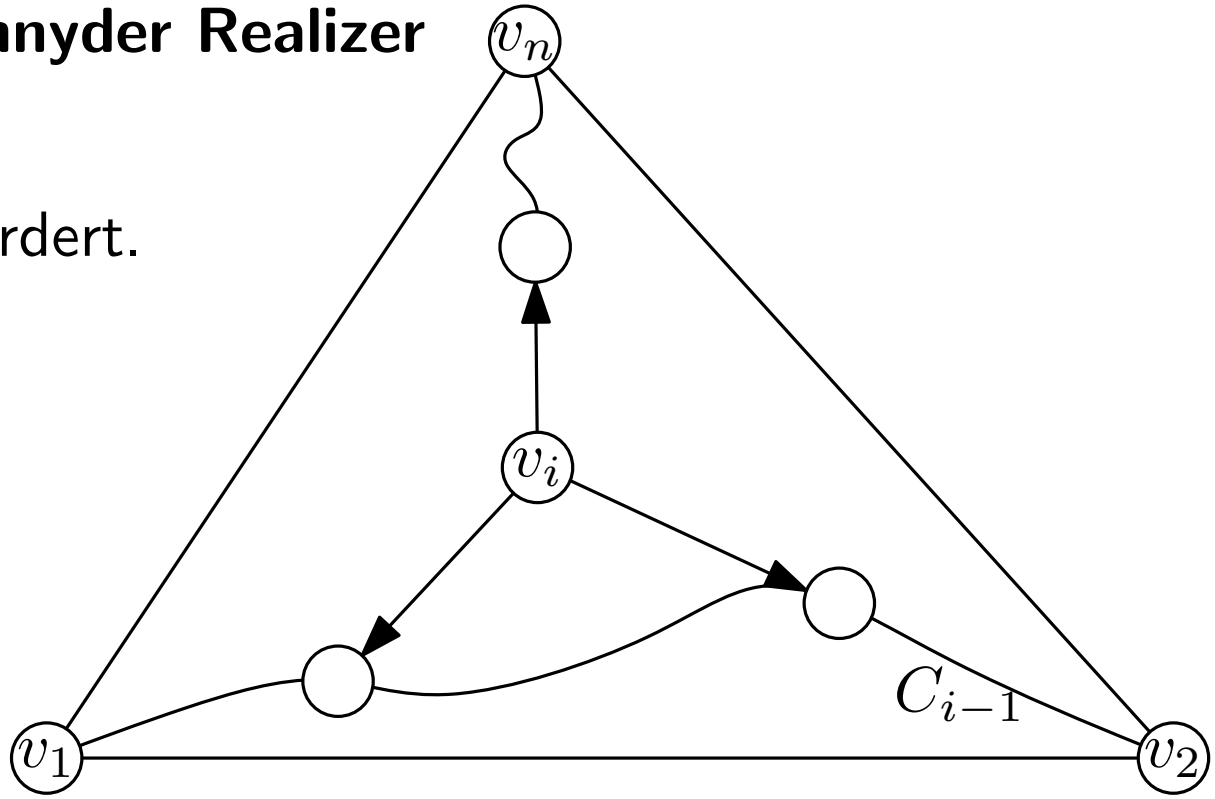
Wirklich Konstruktionsvorschrift für Schnyder Realizer?

Beweis

Kanonische Ordnung $\rightarrow$ Schnyder Realizer

Behauptung:

Kantenordnung an v_i wie gefordert.



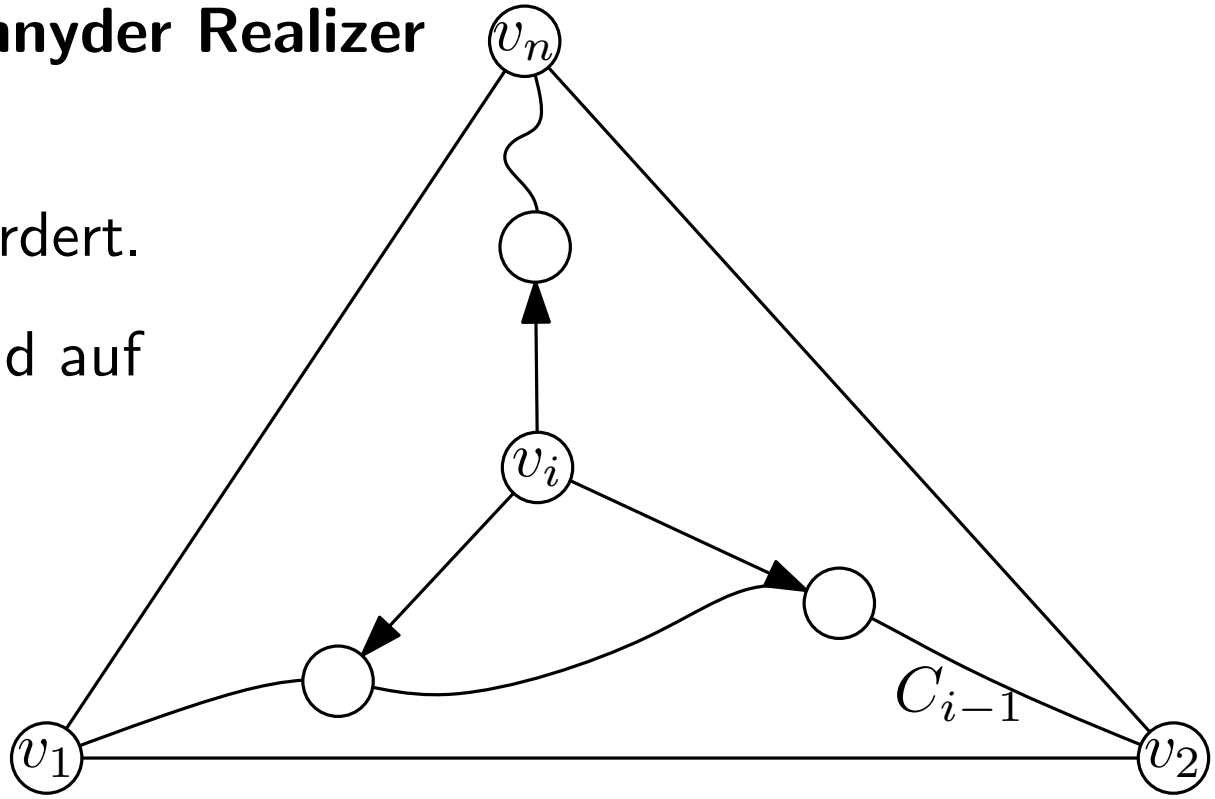
Beweis

Kanonische Ordnung $\rightarrow$ Schnyder Realizer

Behauptung:

Kantenordnung an v_i wie gefordert.

Betrachte Einbettung basierend auf kanonischer Ordnung.



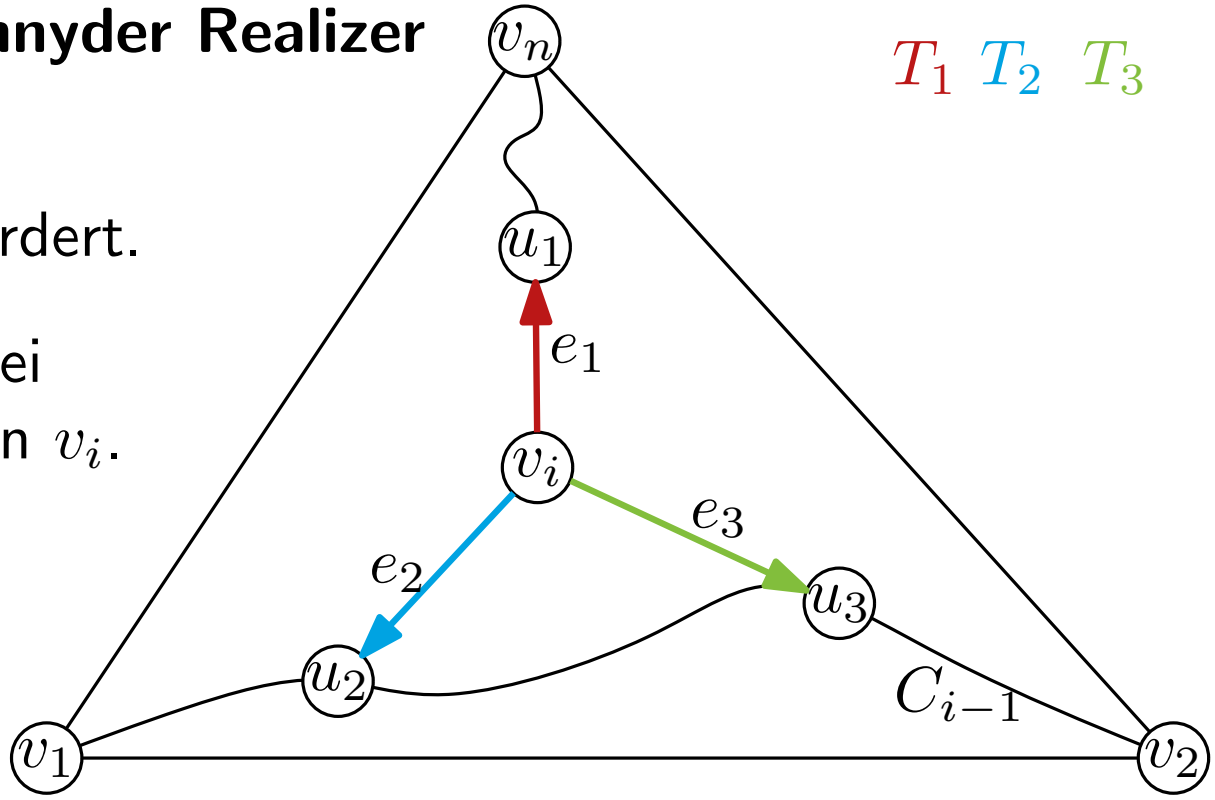
Beweis

Kanonische Ordnung $\rightarrow$ Schnyder Realizer

Behauptung:

Kantenordnung an v_i wie gefordert.

Nach Konstruktion: Es gibt drei ausgehende Kanten pro Knoten v_i. Färbe diese wie dargestellt.



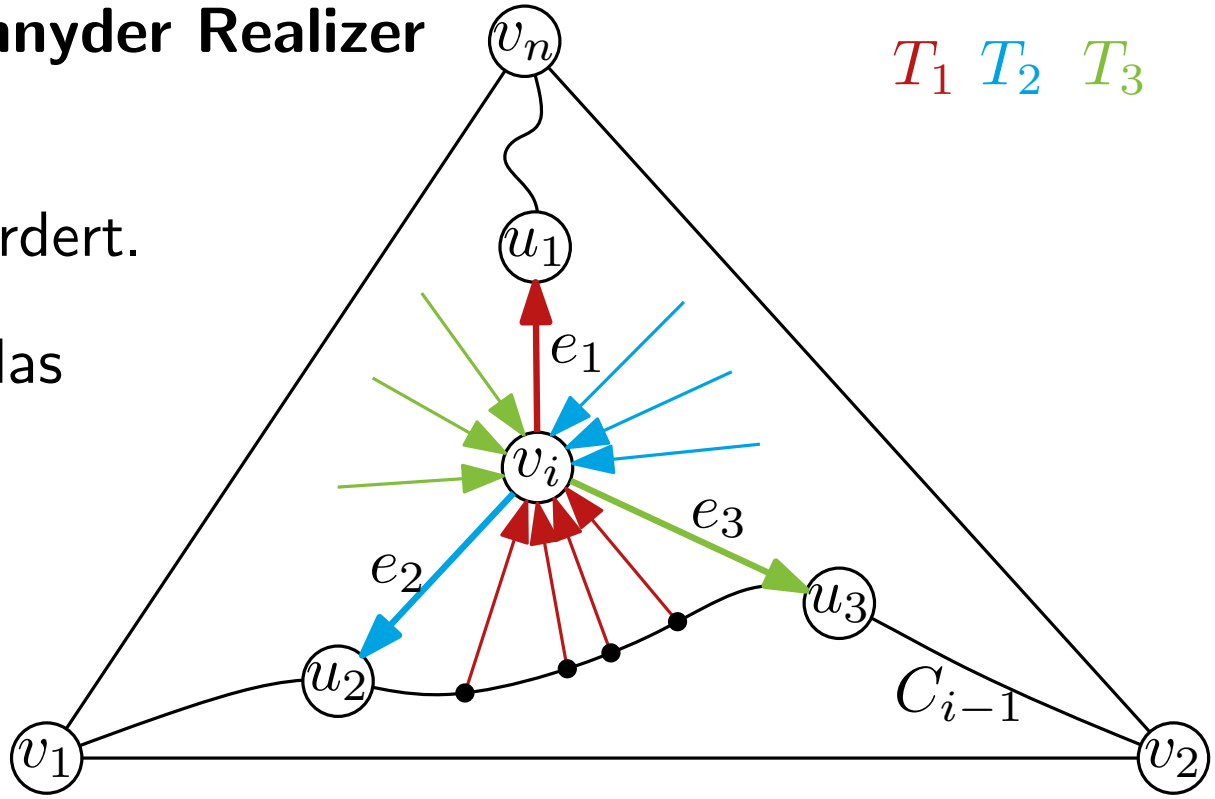
Beweis

Kanonische Ordnung $\rightarrow$ Schnyder Realizer

Behauptung:

Kantenordnung an v_i wie gefordert.

Zeige für eingehende Kanten das geforderte Ordnung gilt.



Beweis

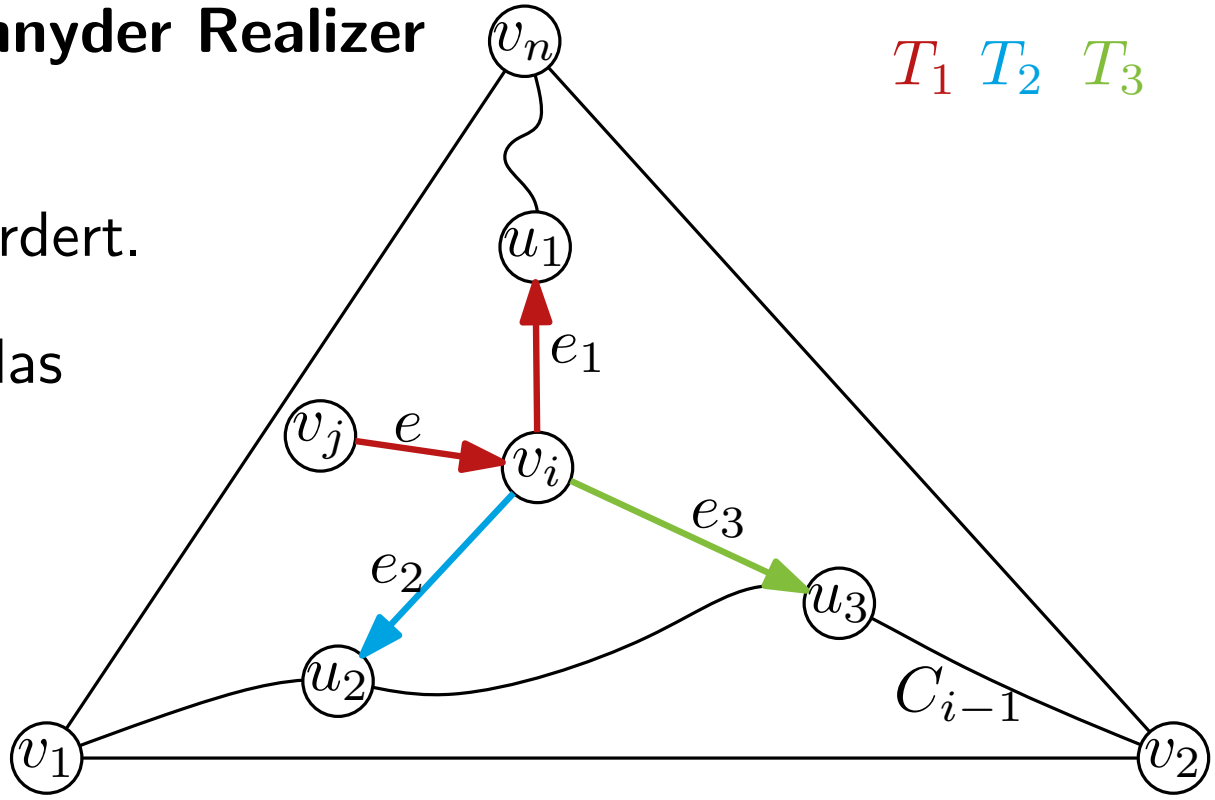
Kanonische Ordnung → Schnyder Realizer

T_1 T_2 T_3

Behauptung:

Kantenordnung an v_i wie gefordert.

Zeige für eingehende Kanten das geforderte Ordnung gilt.



Annahme: $\exists$ Kante $e = (v_j, v_i) \in T_1$ rechts von e_2 und links von e_1

Beweis

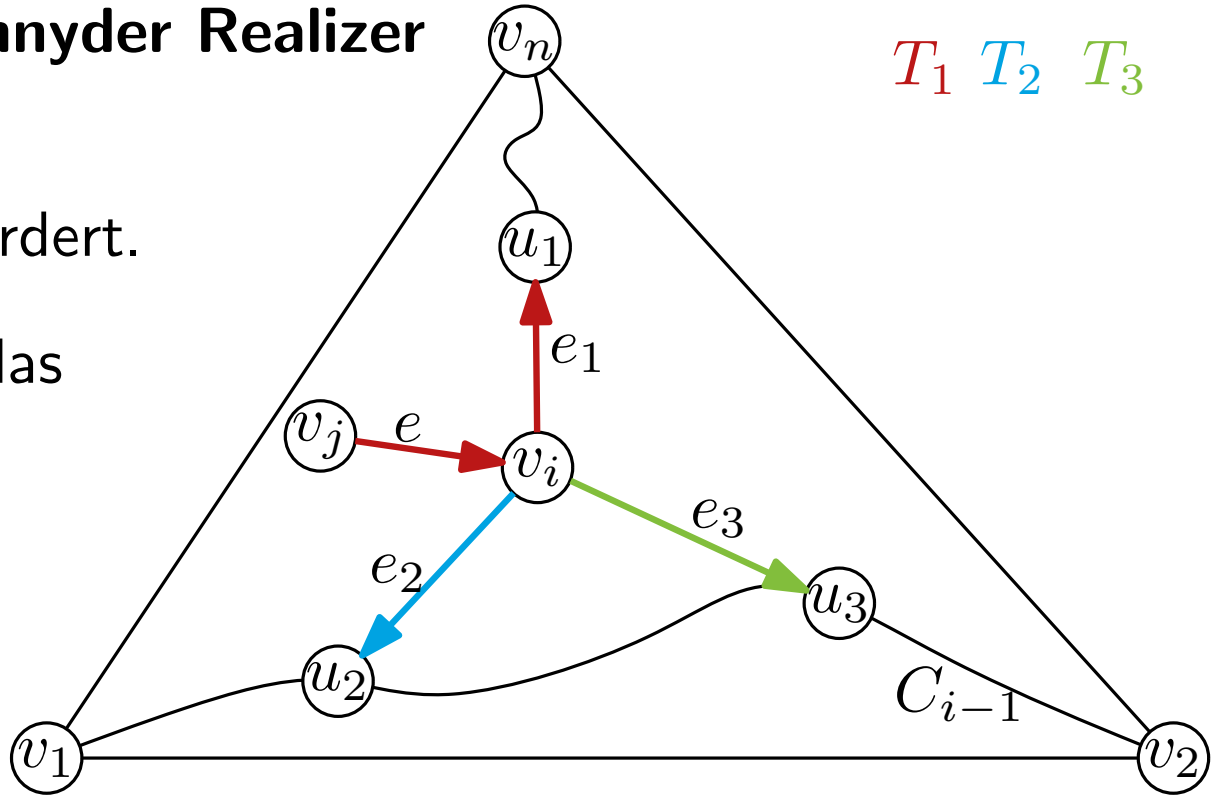
Kanonische Ordnung $\rightarrow$ Schnyder Realizer

T_1 T_2 T_3

Behauptung:

Kantenordnung an v_i wie gefordert.

Zeige für eingehende Kanten das geforderte Ordnung gilt.



Annahme: $\exists$ Kante $e = (v_j, v_i) \in T_1$ rechts von e_2 und links von e_1

Nach Def. von ausgehenden Kanten von v_j gilt $j < i$

Beweis

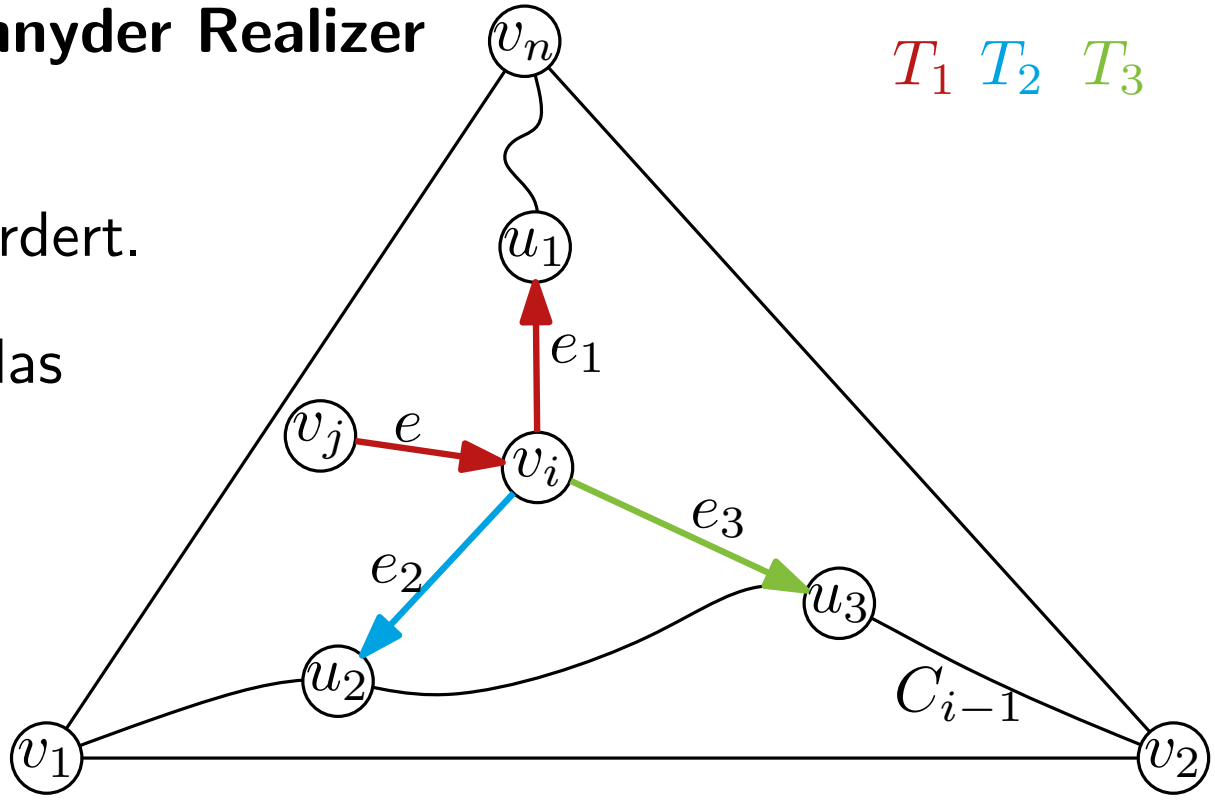
Kanonische Ordnung → Schnyder Realizer

T_1 T_2 T_3

Behauptung:

Kantenordnung an v_i wie gefordert.

Zeige für eingehende Kanten das geforderte Ordnung gilt.



Annahme: $\exists$ Kante $e = (v_j, v_i) \in T_1$ rechts von e_2 und links von e_1

Nach Def. von ausgehenden Kanten von v_j gilt $j < i$

→ v_j liegt auf C_{i-1} ↯ u_2 erster Knoten auf C_{i-1}

Beweis

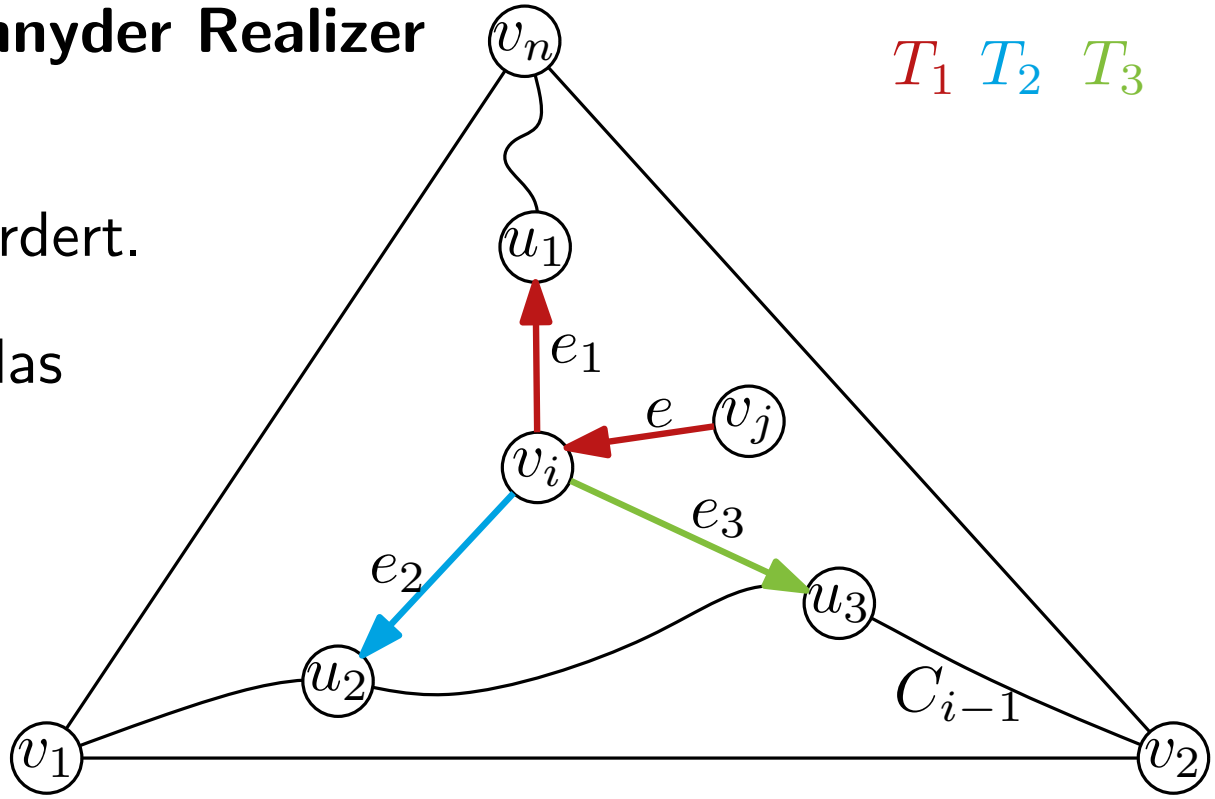
Kanonische Ordnung → Schnyder Realizer

T_1 T_2 T_3

Behauptung:

Kantenordnung an v_i wie gefordert.

Zeige für eingehende Kanten das geforderte Ordnung gilt.



Analog: $\exists$ Kante $e = (v_j, v_i) \in T_1$ rechts von e_1 und links von e_3

Beweis

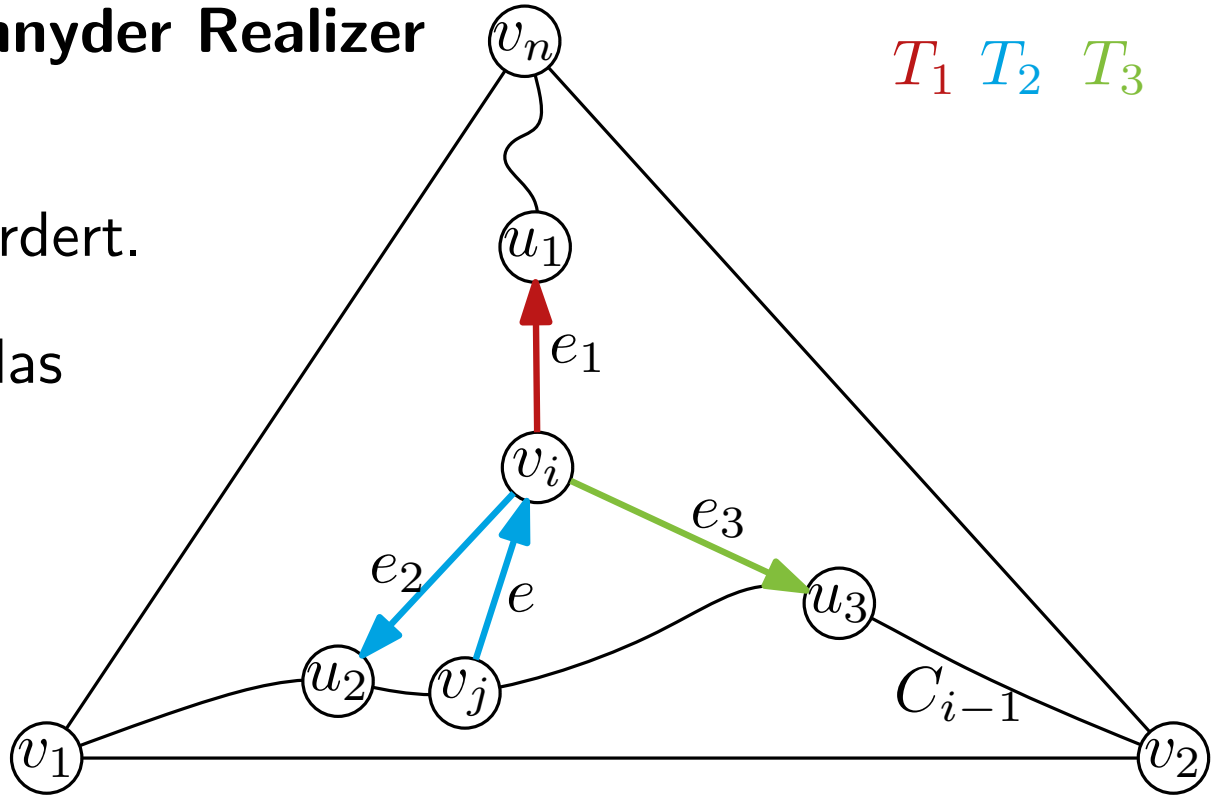
Kanonische Ordnung $\rightarrow$ Schnyder Realizer

T_1 T_2 T_3

Behauptung:

Kantenordnung an v_i wie gefordert.

Zeige für eingehende Kanten das geforderte Ordnung gilt.



Annahme: $\exists$ Kante $e = (v_j, v_i) \in T_2$ links von e_2 und rechts von e_3

Beweis

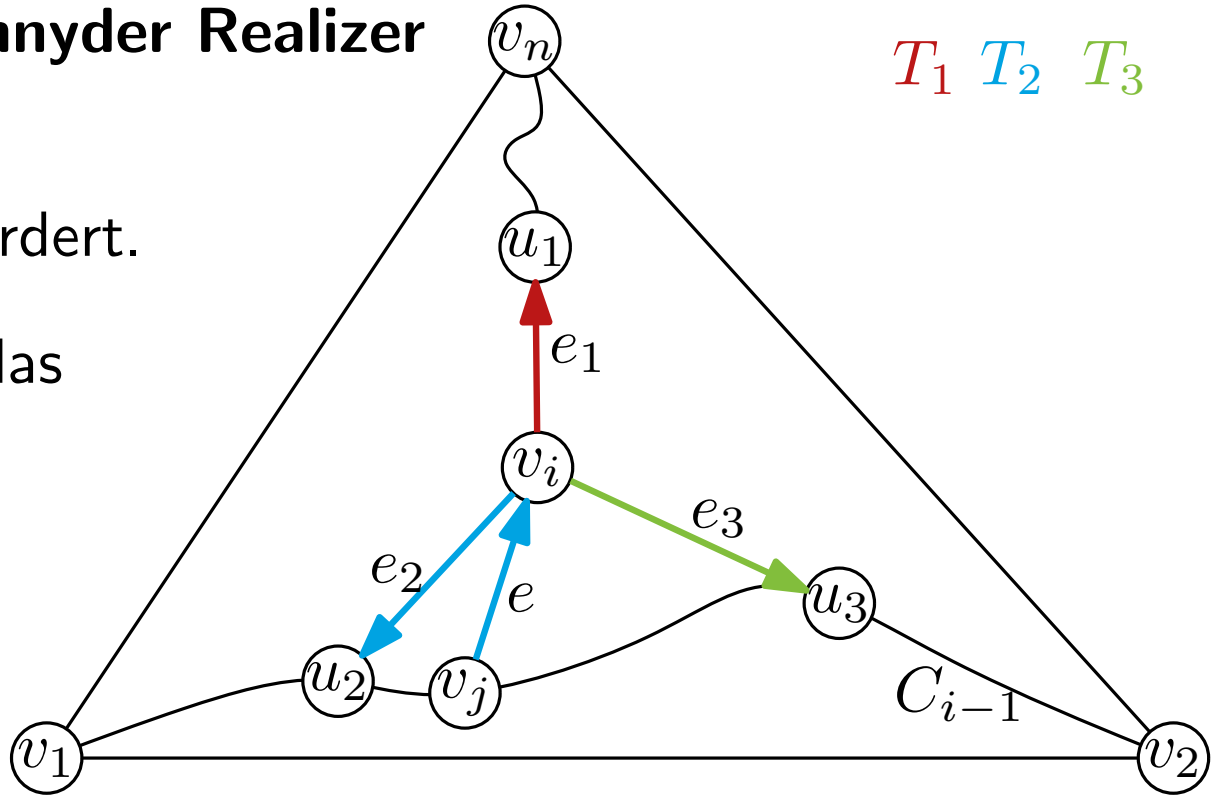
Kanonische Ordnung → Schnyder Realizer

T_1 T_2 T_3

Behauptung:

Kantenordnung an v_i wie gefordert.

Zeige für eingehende Kanten das geforderte Ordnung gilt.



Annahme: $\exists$ Kante $e = (v_j, v_i) \in T_2$ links von e_2 und rechts von e_3

v_j liegt auf C_{i-1} nach def von kanon. Ordnung und Wahl von u_2 und u_3.

→ $i > j$

↯ Nach Konstruktion muss $j > i$ gelten.

Beweis

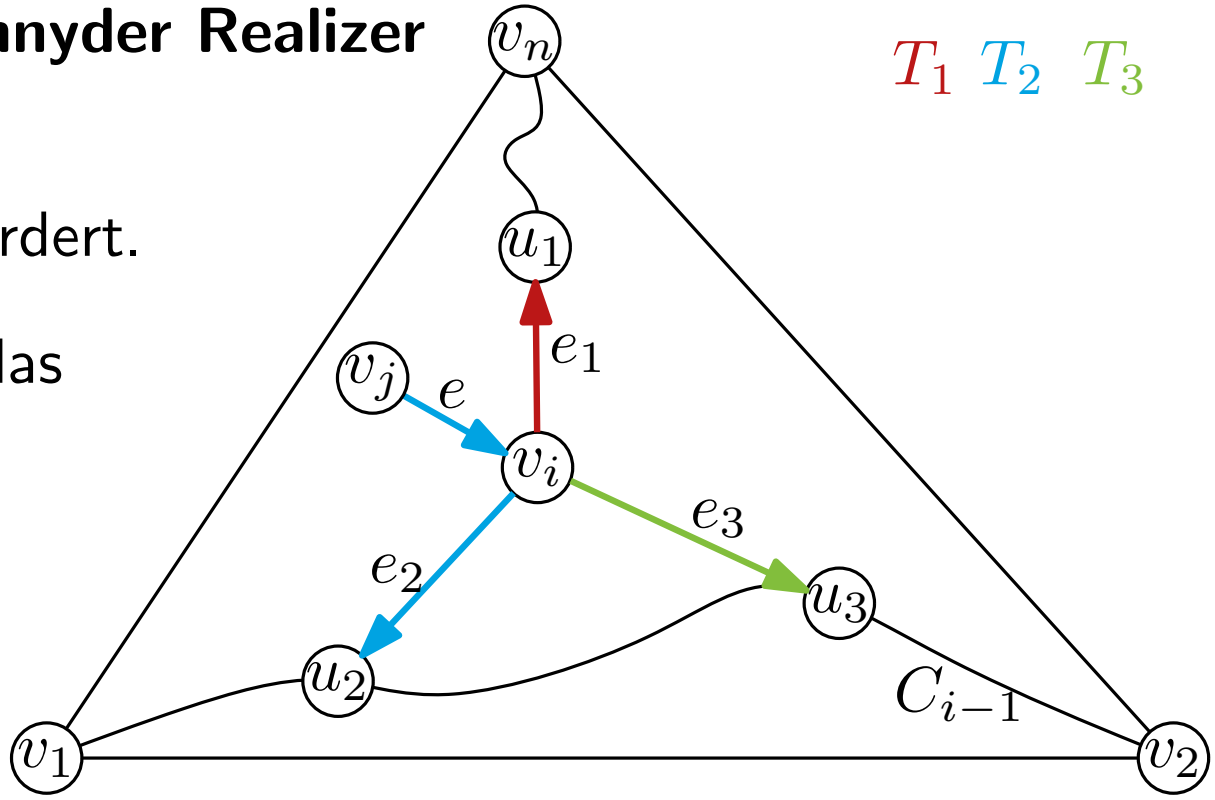
Kanonische Ordnung → Schnyder Realizer

T_1 T_2 T_3

Behauptung:

Kantenordnung an v_i wie gefordert.

Zeige für eingehende Kanten das geforderte Ordnung gilt.



Annahme: $\exists$ Kante $e = (v_j, v_i) \in T_2$ links von e_1 und rechts von e_2

Beweis

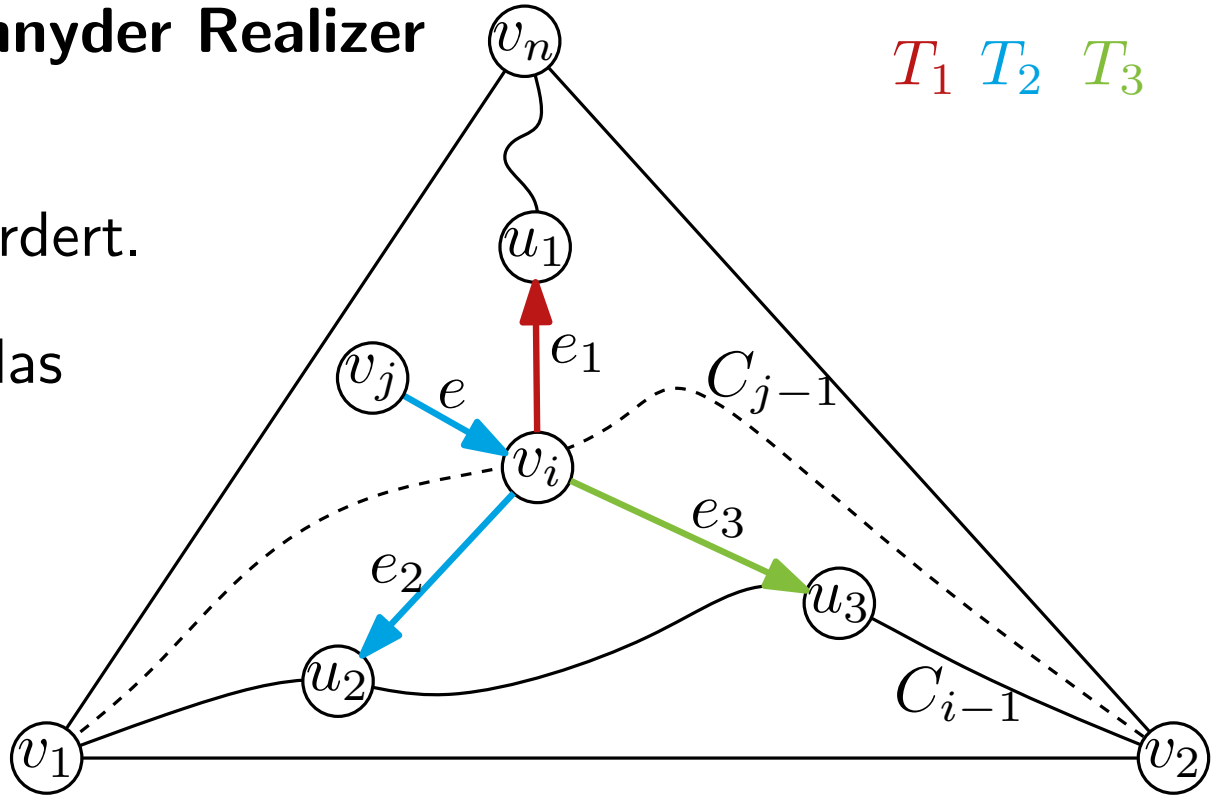
Kanonische Ordnung → Schnyder Realizer

T_1 T_2 T_3

Behauptung:

Kantenordnung an v_i wie gefordert.

Zeige für eingehende Kanten das geforderte Ordnung gilt.



Annahme: $\exists$ Kante $e = (v_j, v_i) \in T_2$ links von e_1 und rechts von e_2

v_j ist erster Knoten auf C_{j-1}

Beweis

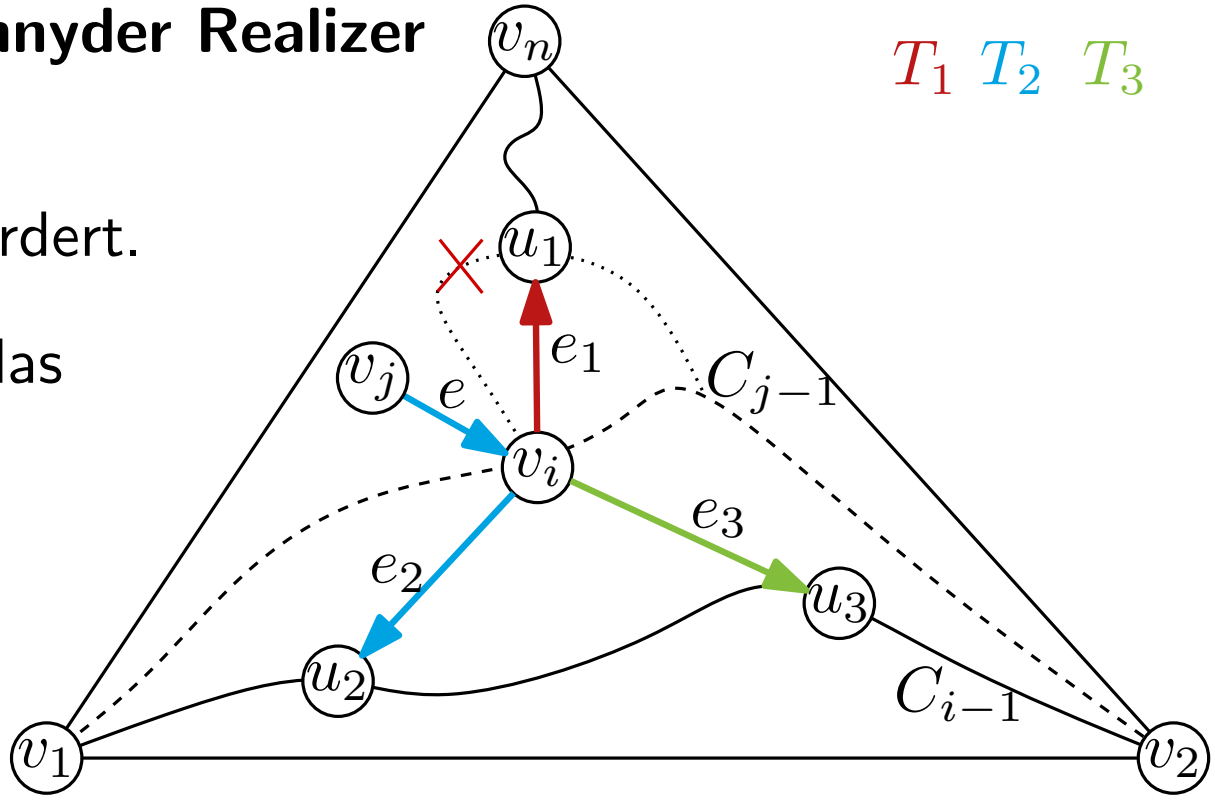
Kanonische Ordnung → Schnyder Realizer

T_1 T_2 T_3

Behauptung:

Kantenordnung an v_i wie gefordert.

Zeige für eingehende Kanten das geforderte Ordnung gilt.



Annahme: $\exists$ Kante $e = (v_j, v_i) \in T_2$ links von e_1 und rechts von e_2

v_j ist erster Knoten auf C_{j-1}

Da u_1 Nachfolger mit höchstem Index, kann C_{j-1} weder u_1 noch Nachfolger von u_1 enthalten. ($j <$ Index von u_1 in kanon. Ordnung)

Beweis

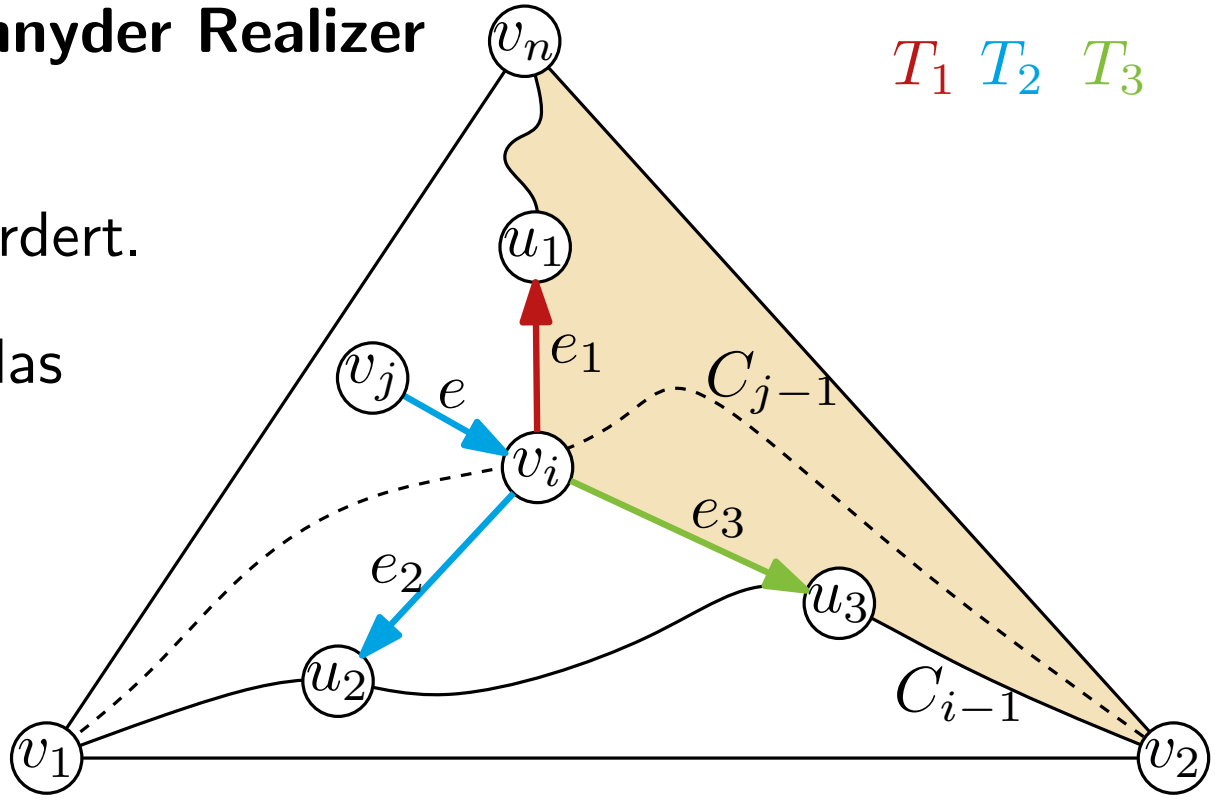
Kanonische Ordnung → Schnyder Realizer

T_1 T_2 T_3

Behauptung:

Kantenordnung an v_i wie gefordert.

Zeige für eingehende Kanten das geforderte Ordnung gilt.



Annahme: $\exists$ Kante $e = (v_j, v_i) \in T_2$ links von e_1 und rechts von e_2

v_j ist erster Knoten auf C_{j-1}

Teilstück C' zwischen v_i und v_2 von C_{j-1} liegt damit im orangen Bereich.

Beweis

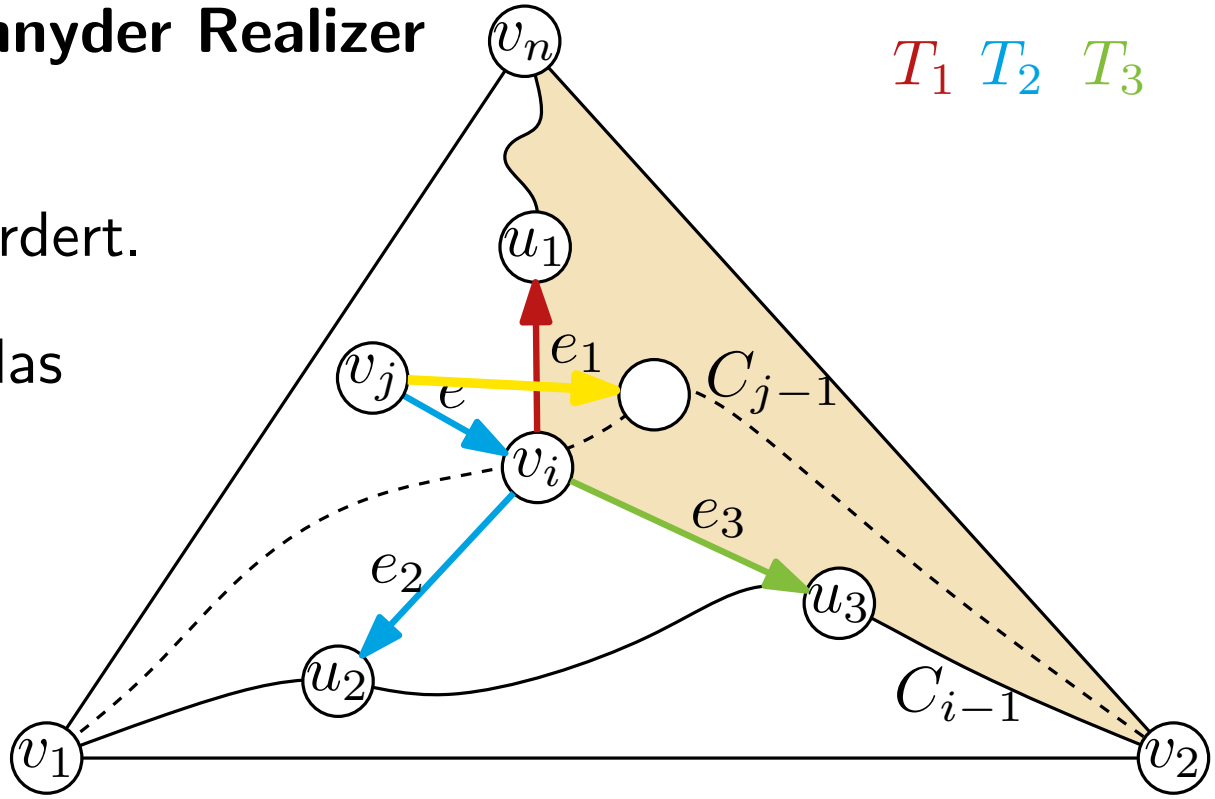
Kanonische Ordnung → Schnyder Realizer

T_1 T_2 T_3

Behauptung:

Kantenordnung an v_i wie gefordert.

Zeige für eingehende Kanten das geforderte Ordnung gilt.



Annahme: $\exists$ Kante $e = (v_j, v_i) \in T_2$ links von e_1 und rechts von e_2

v_j ist erster Knoten auf C_{j-1}

Teilstück C' zwischen v_i und v_2 von C_{j-1} liegt damit im orangen Bereich.

Damit G_i 2-fach zusammenhängend: v_j ist verbunden mit Knoten auf C'.

↯ Planarität

Kanonischer Ordnung & Schnyder Realizer

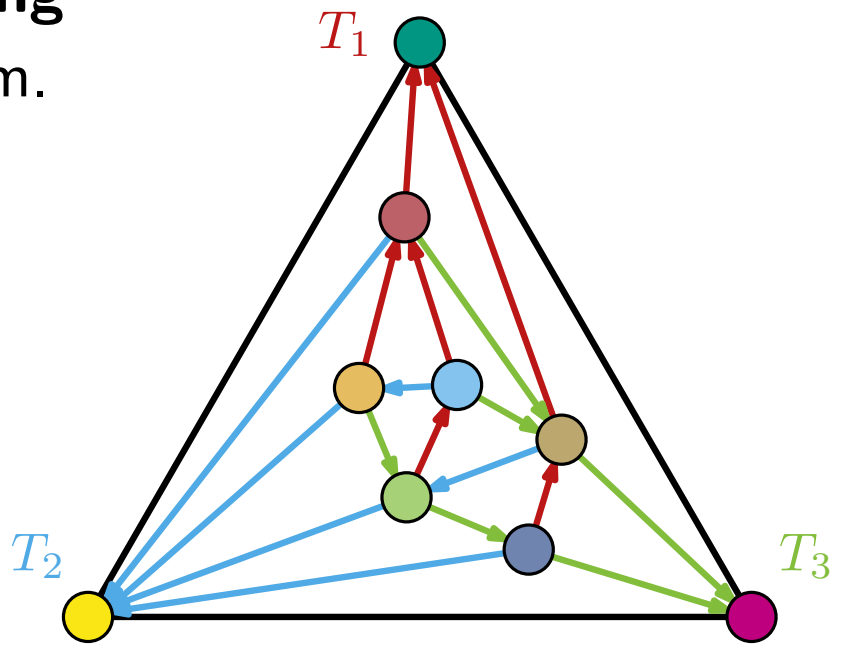
Aus der Vorlesung: Ein Schnyder Realizer mit Bäumen T_1, T_2, T_3 definiert eine kanonische Ordnung als topologische Ordnung des azyklischen Graphen $T_1 \cup T_2^{-1} \cup T_3^{-1}$.
(T^{-1}: invertiere Kantenrichtungen von T)

Wirklich Konstruktionsvorschrift für kanonische Ordnung?

Kanonische Ordnung & Schnyder Realizer

Schnyder Realizer $\rightarrow$ Kanonische Ordnung

1. Schritt: Drehe Kanten von T_2 und T_3 um.

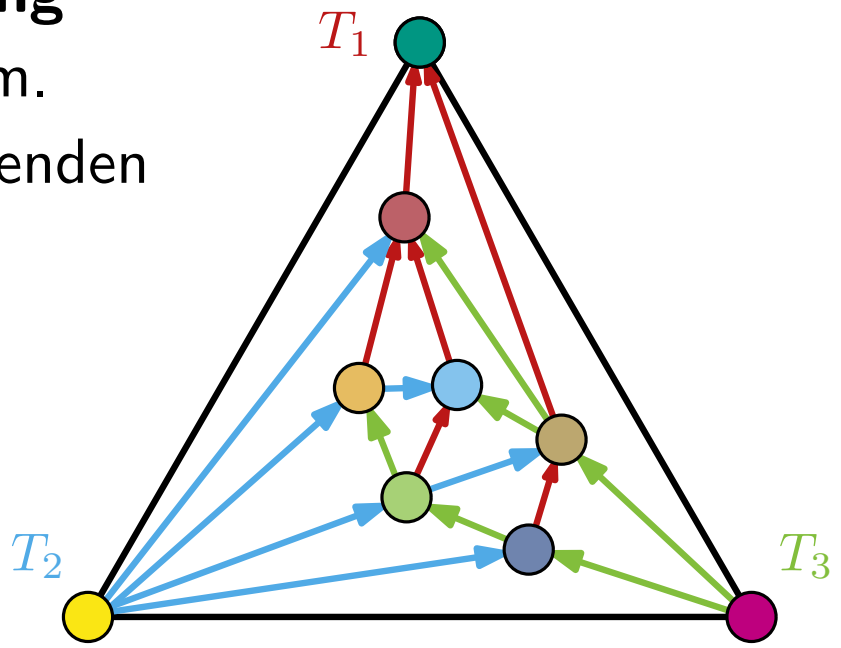


Kanonische Ordnung & Schnyder Realizer

Schnyder Realizer $\rightarrow$ Kanonische Ordnung

1. Schritt: Drehe Kanten von T_2 und T_3 um.

2. Schritt: Entferne Knoten mit nur eingehenden Kanten und indiziere ihn entsprechend.

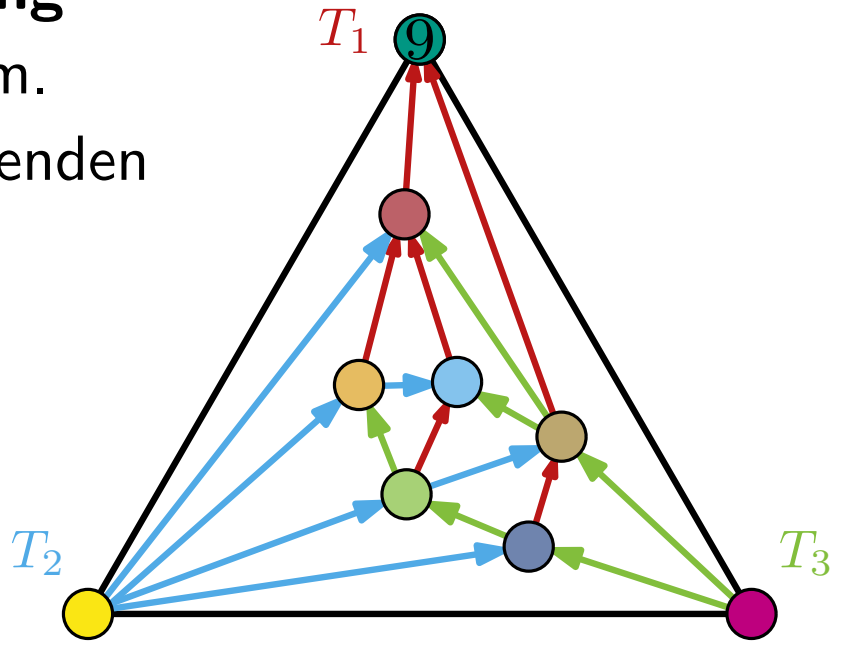


Kanonische Ordnung & Schnyder Realizer

Schnyder Realizer $\rightarrow$ Kanonische Ordnung

1. Schritt: Drehe Kanten von T_2 und T_3 um.

2. Schritt: Entferne Knoten mit nur eingehenden Kanten und indiziere ihn entsprechend.

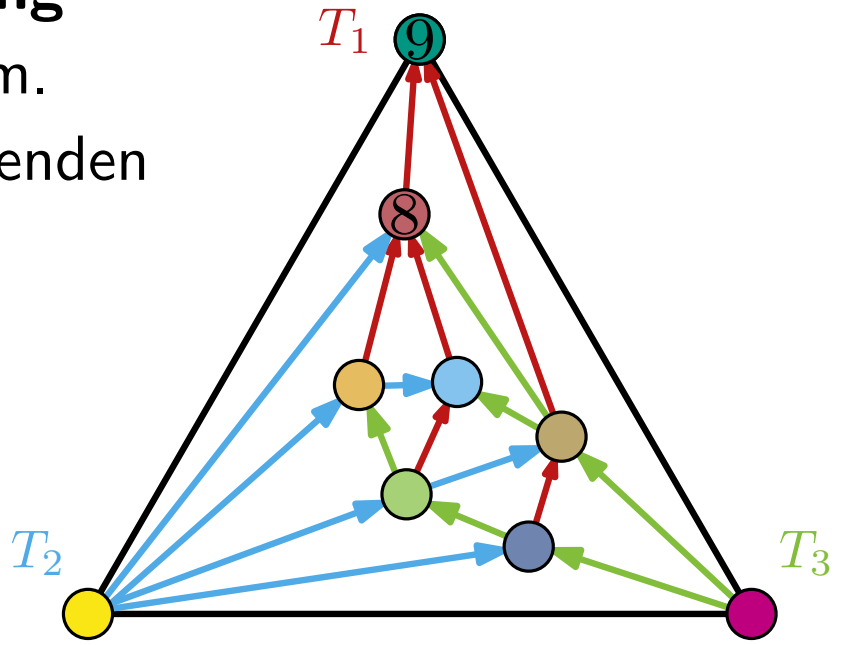


Kanonische Ordnung & Schnyder Realizer

Schnyder Realizer $\rightarrow$ Kanonische Ordnung

1. Schritt: Drehe Kanten von T_2 und T_3 um.

2. Schritt: Entferne Knoten mit nur eingehenden Kanten und indiziere ihn entsprechend.

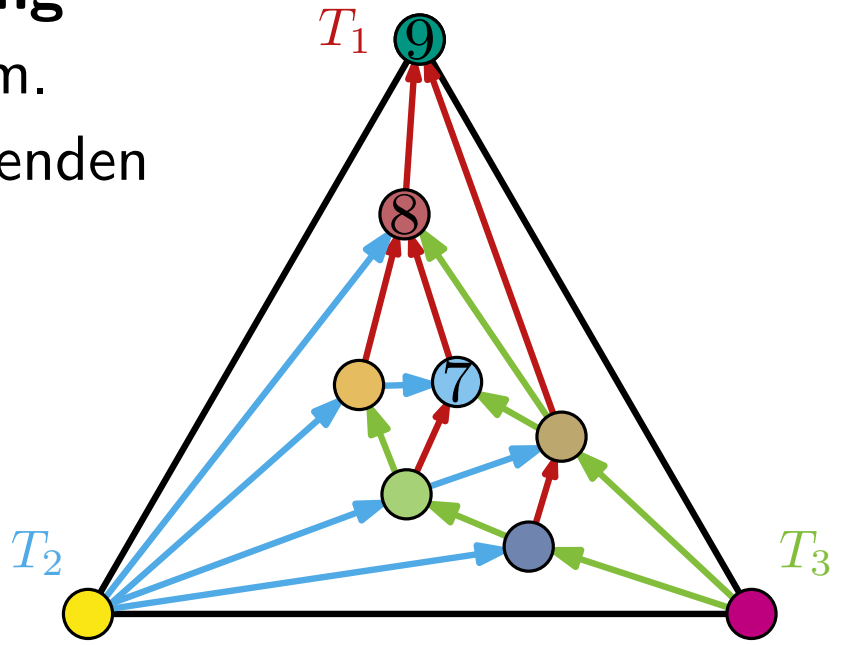


Kanonische Ordnung & Schnyder Realizer

Schnyder Realizer $\rightarrow$ Kanonische Ordnung

1. Schritt: Drehe Kanten von T_2 und T_3 um.

2. Schritt: Entferne Knoten mit nur eingehenden Kanten und indiziere ihn entsprechend.

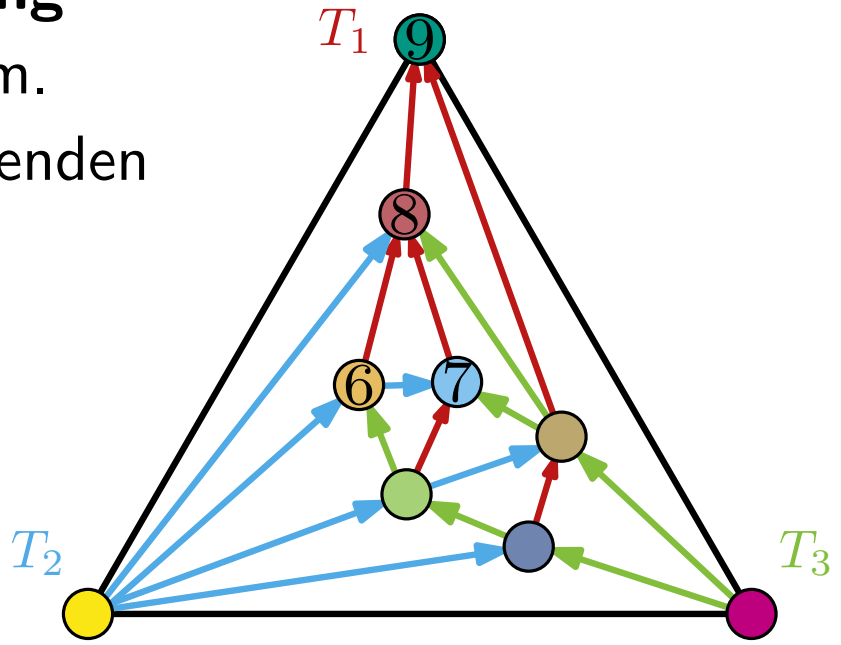


Kanonische Ordnung & Schnyder Realizer

Schnyder Realizer $\rightarrow$ Kanonische Ordnung

1. Schritt: Drehe Kanten von T_2 und T_3 um.

2. Schritt: Entferne Knoten mit nur eingehenden Kanten und indiziere ihn entsprechend.

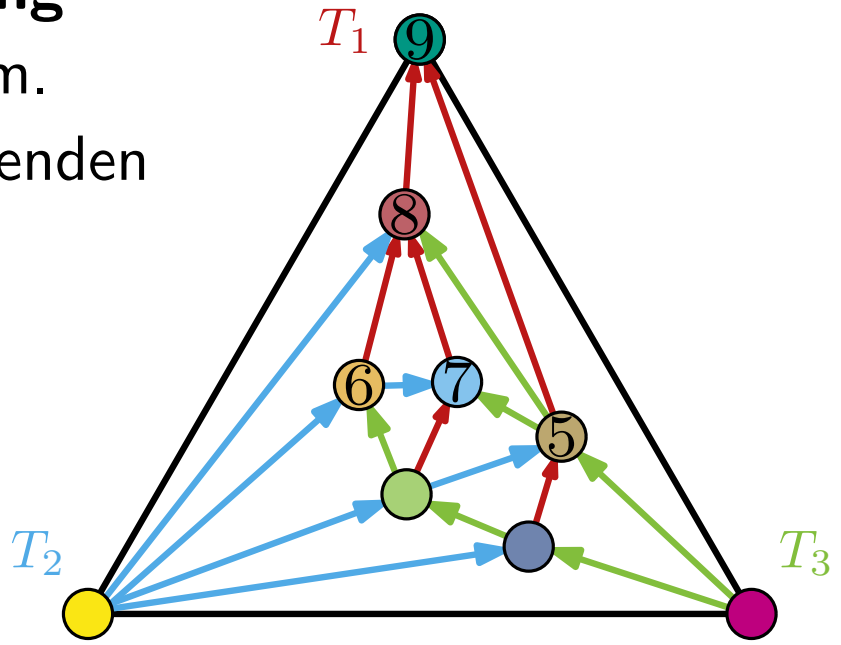


Kanonische Ordnung & Schnyder Realizer

Schnyder Realizer $\rightarrow$ Kanonische Ordnung

1. Schritt: Drehe Kanten von T_2 und T_3 um.

2. Schritt: Entferne Knoten mit nur eingehenden Kanten und indiziere ihn entsprechend.

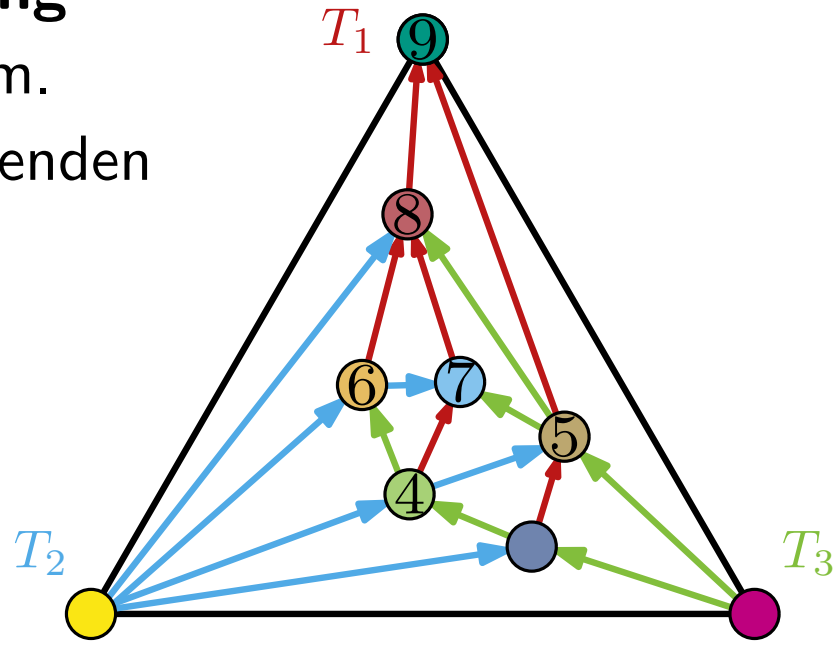


Kanonische Ordnung & Schnyder Realizer

Schnyder Realizer $\rightarrow$ Kanonische Ordnung

1. Schritt: Drehe Kanten von T_2 und T_3 um.

2. Schritt: Entferne Knoten mit nur eingehenden Kanten und indiziere ihn entsprechend.

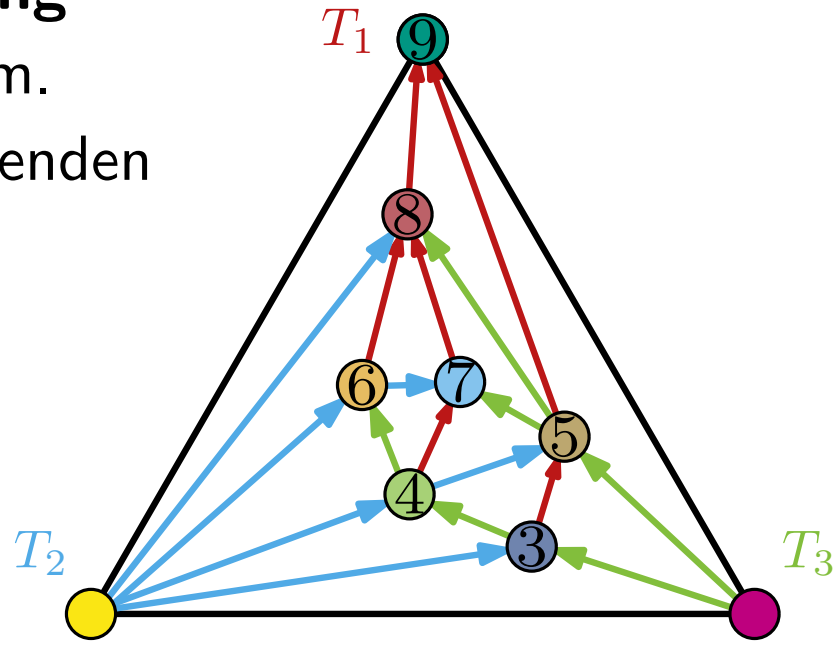


Kanonische Ordnung & Schnyder Realizer

Schnyder Realizer $\rightarrow$ Kanonische Ordnung

1. Schritt: Drehe Kanten von T_2 und T_3 um.

2. Schritt: Entferne Knoten mit nur eingehenden Kanten und indiziere ihn entsprechend.

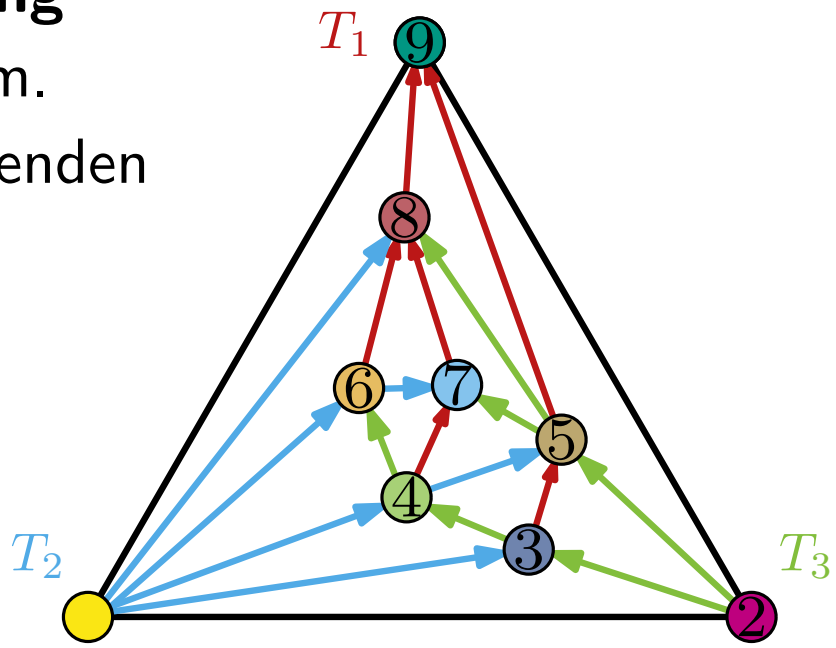


Kanonische Ordnung & Schnyder Realizer

Schnyder Realizer $\rightarrow$ Kanonische Ordnung

1. Schritt: Drehe Kanten von T_2 und T_3 um.

2. Schritt: Entferne Knoten mit nur eingehenden Kanten und indiziere ihn entsprechend.

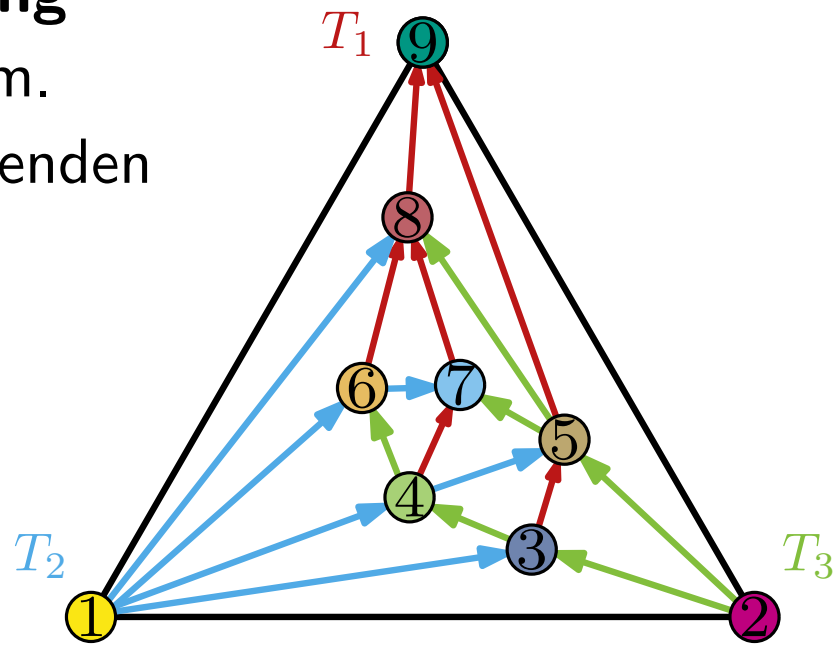


Kanonische Ordnung & Schnyder Realizer

Schnyder Realizer $\rightarrow$ Kanonische Ordnung

1. Schritt: Drehe Kanten von T_2 und T_3 um.

2. Schritt: Entferne Knoten mit nur eingehenden Kanten und indiziere ihn entsprechend.



Kanonische Ordnung & Schnyder Realizer

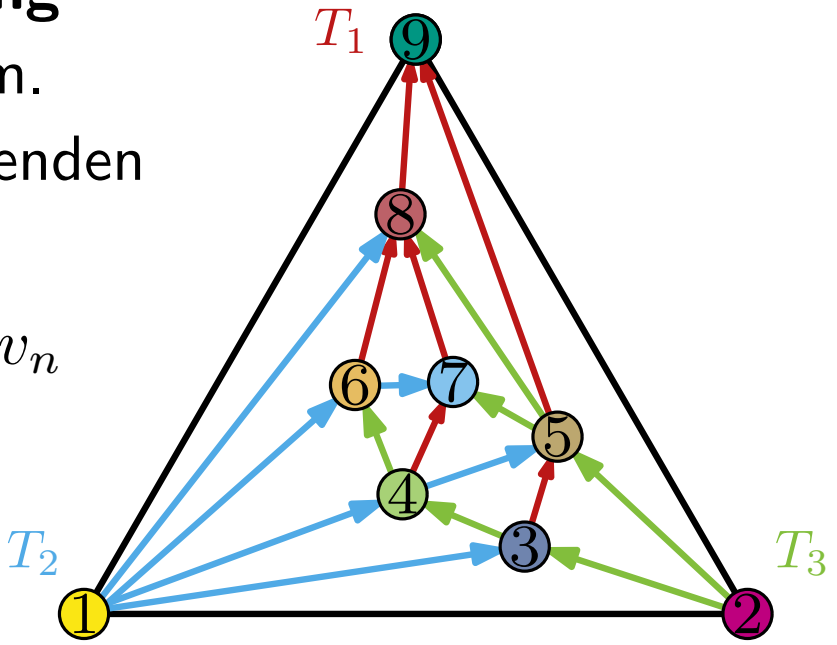
Schnyder Realizer $\rightarrow$ Kanonische Ordnung

1. Schritt: Drehe Kanten von T_2 und T_3 um.

2. Schritt: Entferne Knoten mit nur eingehenden Kanten und indiziere ihn entsprechend.

Baue G in umgekehrte Reihenfolge $v_1, \ldots, v_n$ wieder auf. (Induktion)

- Wurzeln von T_2 und T_3 zuletzt gelöscht.
- Beginne mit Dreieck $v_1 v_2 v_3$



Kanonische Ordnung & Schnyder Realizer

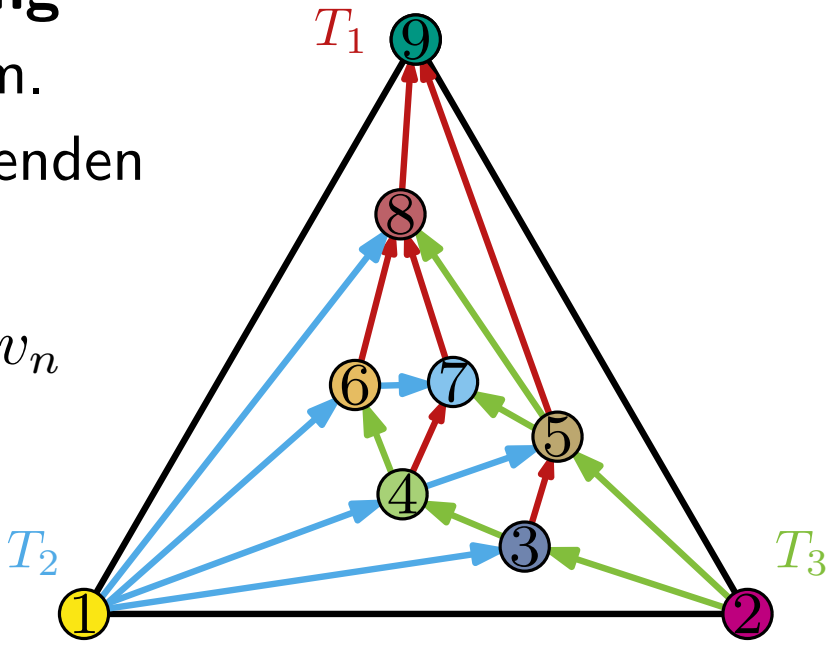
Schnyder Realizer $\rightarrow$ Kanonische Ordnung

1. Schritt: Drehe Kanten von T_2 und T_3 um.

2. Schritt: Entferne Knoten mit nur eingehenden Kanten und indiziere ihn entsprechend.

Baue G in umgekehrte Reihenfolge $v_1, \ldots, v_n$ wieder auf. (Induktion)

- Wurzeln von T_2 und T_3 zuletzt gelöscht.
- Beginne mit Dreieck $v_1 v_2 v_3$

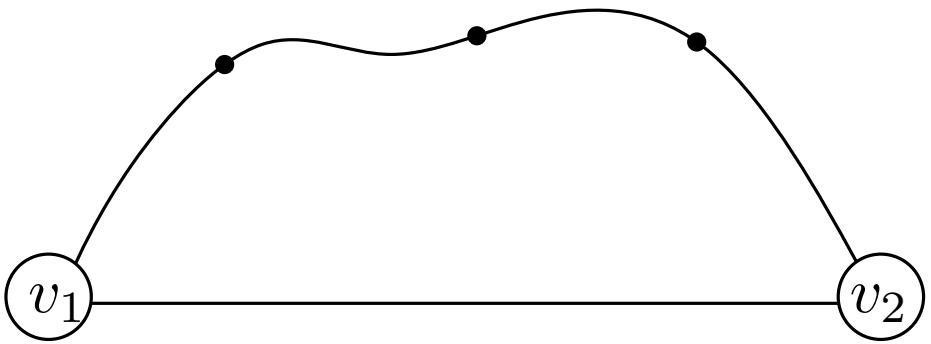


Hinweis: Da Graph G trianguliert und äußere Facette durch v_1, v_2 und v_n vorgeben ist, ist die Einbettung eindeutig $\rightarrow$ betrachte diese.

Beweis

Betrachte Einfügen von v_i:

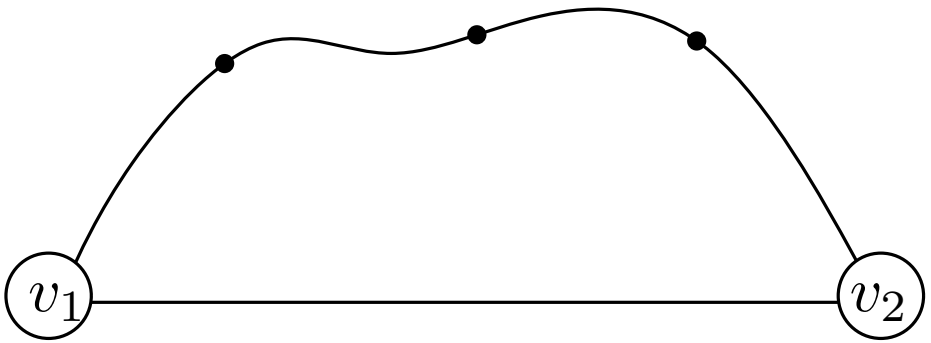
IV: $v_1, \ldots, v_{i-1}$ induzieren 2-fach zshgd. Graph G_{i-1}, sodass



Beweis

Betrachte Einfügen von v_i:

IV: $v_1, \ldots, v_{i-1}$ induzieren 2-fach zshgd. Graph G_{i-1}, sodass alle Knoten die Kanten zu Knoten $v_i, \ldots, v_n$ haben, auf der äußeren Facette liegen.

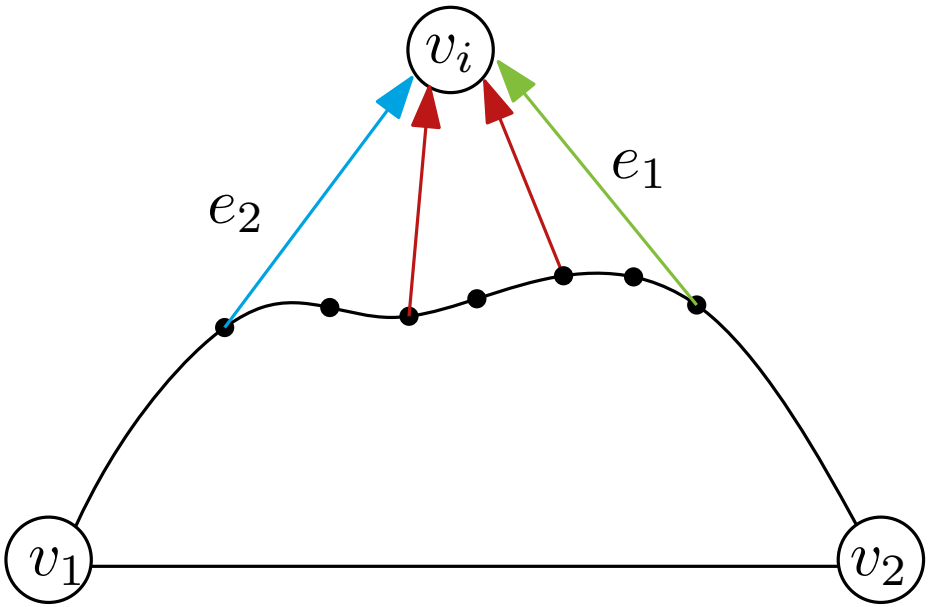


Beweis

Betrachte Einfügen von v_i:

IV: $v_1, \ldots, v_{i-1}$ induzieren 2-fach zshgd. Graph G_{i-1}, sodass alle Knoten die Kanten zu Knoten $v_i, \ldots, v_n$ haben, auf der äußeren Facette liegen.

Setze v_i in äußere Facette von G_{i-1}. Nach IV entstehen keine Kreuzungen.



Beweis

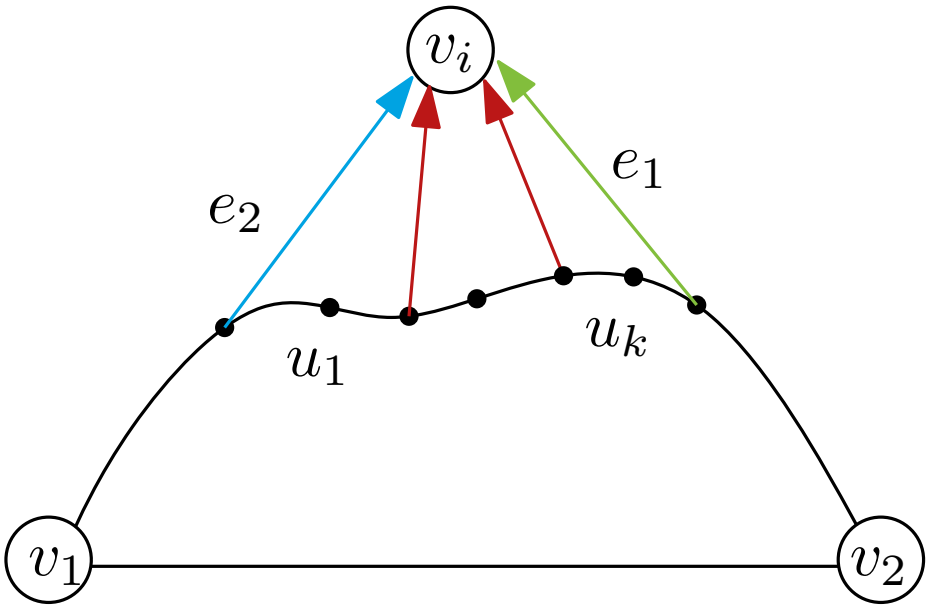
Betrachte Einfügen von v_i:

IV: $v_1, \ldots, v_{i-1}$ induzieren 2-fach zshgd. Graph G_{i-1}, sodass alle Knoten die Kanten zu Knoten $v_i, \ldots, v_n$ haben, auf der äußeren Facette liegen.

Setze v_i in äußere Facette von G_{i-1}. Nach IV entstehen keine Kreuzungen.

Seien $u_1, \ldots, u_k$ Knoten auf äußerer Facette zwischen e_2 und e_1.

Zeige: $u_1, \ldots, u_k$ haben keine Kanten zu $v_{i+1}, \ldots, v_n$ in G



Beweis

Betrachte Einfügen von v_i:

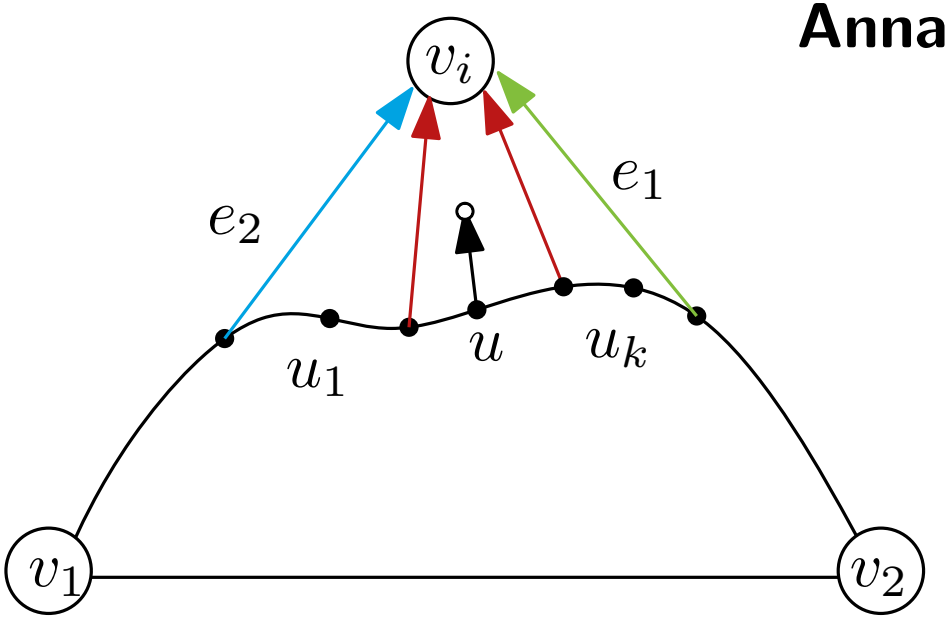
IV: $v_1, \ldots, v_{i-1}$ induzieren 2-fach zshgd. Graph G_{i-1}, sodass alle Knoten die Kanten zu Knoten $v_i, \ldots, v_n$ haben, auf der äußeren Facette liegen.

Setze v_i in äußere Facette von G_{i-1}. Nach IV entstehen keine Kreuzungen.

Seien $u_1, \ldots, u_k$ Knoten auf äußerer Facette zwischen e_2 und e_1.

Zeige: $u_1, \ldots, u_k$ haben keine Kanten zu $v_{i+1}, \ldots, v_n$ in G

Annahme: Es gibt solchen Knoten u in G



Beweis

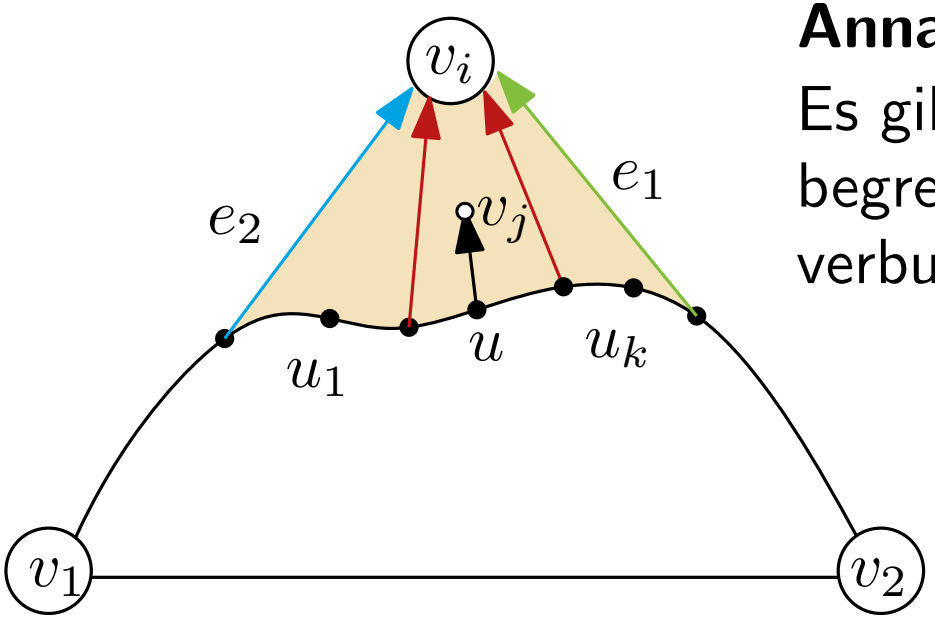
Betrachte Einfügen von v_i:

IV: $v_1, \ldots, v_{i-1}$ induzieren 2-fach zshgd. Graph G_{i-1}, sodass alle Knoten die Kanten zu Knoten $v_i, \ldots, v_n$ haben, auf der äußeren Facette liegen.

Setze v_i in äußere Facette von G_{i-1}. Nach IV entstehen keine Kreuzungen.

Seien $u_1, \ldots, u_k$ Knoten auf äußerer Facette zwischen e_2 und e_1.

Zeige: $u_1, \ldots, u_k$ haben keine Kanten zu $v_{i+1}, \ldots, v_n$ in G



Annahme: Es gibt solchen Knoten u in G
Es gibt Knoten v_j mit $j \geq i+1$ in Region begrenzt durch e_1, e_2 und C_{i-1}, der mit u verbunden ist.

Beweis

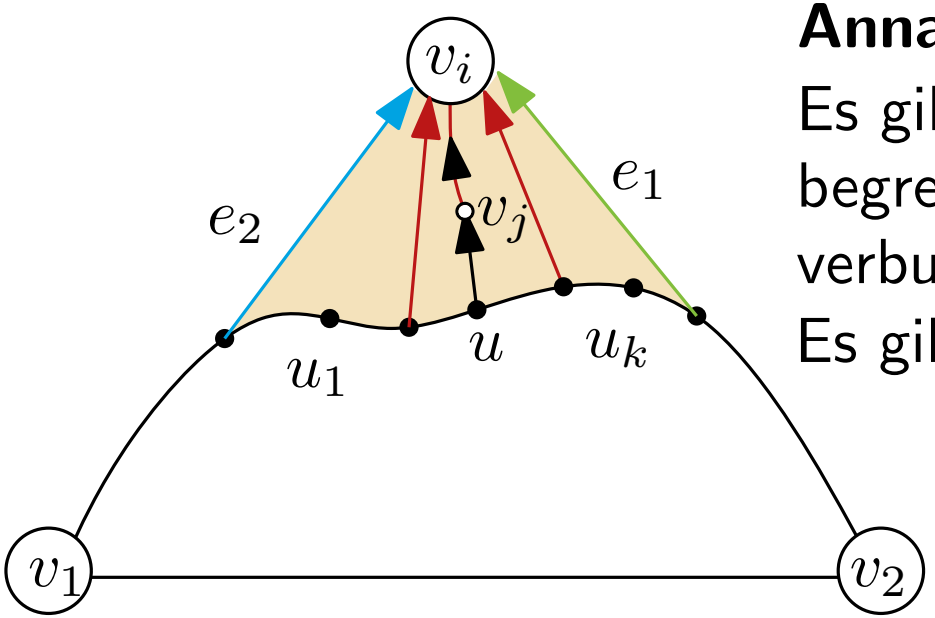
Betrachte Einfügen von v_i:

IV: $v_1, \ldots, v_{i-1}$ induzieren 2-fach zshgd. Graph G_{i-1}, sodass alle Knoten die Kanten zu Knoten $v_i, \ldots, v_n$ haben, auf der äußeren Facette liegen.

Setze v_i in äußere Facette von G_{i-1}. Nach IV entstehen keine Kreuzungen.

Seien $u_1, \ldots, u_k$ Knoten auf äußerer Facette zwischen e_2 und e_1.

Zeige: $u_1, \ldots, u_k$ haben keine Kanten zu $v_{i+1}, \ldots, v_n$ in G



Annahme: Es gibt solchen Knoten u in G
Es gibt Knoten v_j mit $j \geq i+1$ in Region begrenzt durch e_1, e_2 und C_{i-1}, der mit u verbunden ist.
Es gibt gerichteten Pfad in T_1 von v_j nach v_i.

Beweis

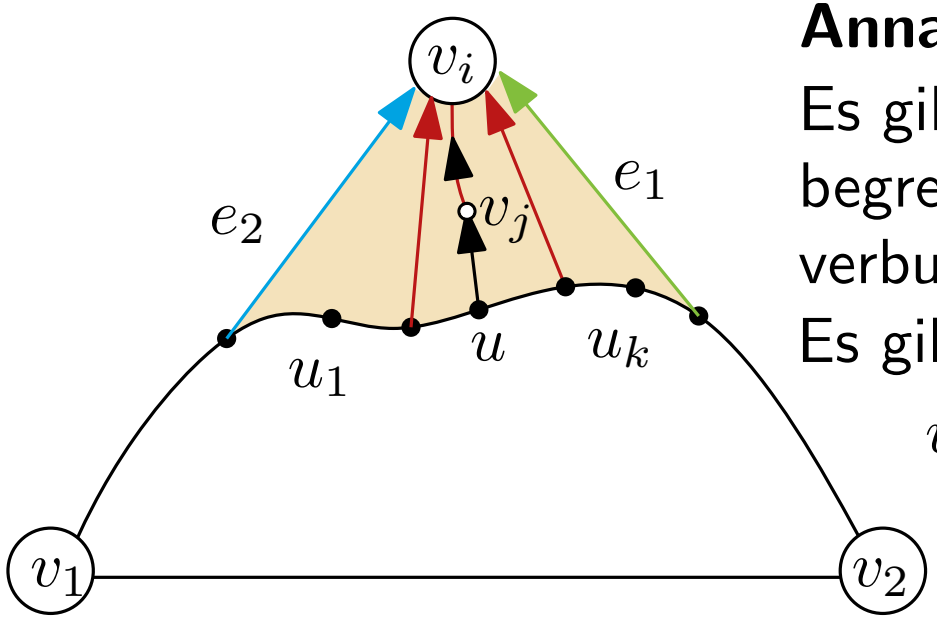
Betrachte Einfügen von v_i:

IV: $v_1, \ldots, v_{i-1}$ induzieren 2-fach zshgd. Graph G_{i-1}, sodass alle Knoten die Kanten zu Knoten $v_i, \ldots, v_n$ haben, auf der äußeren Facette liegen.

Setze v_i in äußere Facette von G_{i-1}. Nach IV entstehen keine Kreuzungen.

Seien $u_1, \ldots, u_k$ Knoten auf äußerer Facette zwischen e_2 und e_1.

Zeige: $u_1, \ldots, u_k$ haben keine Kanten zu $v_{i+1}, \ldots, v_n$ in G



Annahme: Es gibt solchen Knoten u in G
Es gibt Knoten v_j mit $j \geq i+1$ in Region begrenzt durch e_1, e_2 und C_{i-1}, der mit u verbunden ist.
Es gibt gerichteten Pfad in T_1 von v_j nach v_i.

v_j wird vor v_i eingefügt $\rightarrow j < i$