

Algorithmen zur Visualisierung von Graphen

Kombinatorische Optimierung mittels Flussmethoden II

Vorlesung im Wintersemester 2011/2012

Tamara Mchedlidze – Martin Nöllenburg – Ignaz Rutter

10.11.2011

Orthogonale Zeichnungen II

letztes Mal:

Satz

G Maxgrad-4-Graph mit fester planarer Einbettung
Orthogonale Zeichnung von G mit minimaler Anzahl an Knicken
kann effizient berechnet werden.

letztes Mal:

Satz

G Maxgrad-4-Graph mit fester planarer Einbettung

Orthogonale Zeichnung von G mit minimaler Anzahl an Knicken kann effizient berechnet werden.

Durch Erweiterung des Flußnetzwerks ebenfalls lösbar:

Gegeben Funktion $\text{flex} : E \rightarrow \mathbb{N}_0$, finde Zeichnung mit minimaler Knickzahl, sodass Kante e höchstens $\text{flex}(e)$ Knicke hat.

Orthogonale Zeichnungen II

letztes Mal:

Satz

G Maxgrad-4-Graph mit fester planarer Einbettung
Orthogonale Zeichnung von G mit minimaler Anzahl an Knicken
kann effizient berechnet werden.

Durch Erweiterung des Flußnetzwerks ebenfalls lösbar:
Gegeben Funktion $\text{flex} : E \rightarrow \mathbb{N}_0$, finde Zeichnung mit minimaler
Knickzahl, sodass Kante e höchstens $\text{flex}(e)$ Knicke hat.

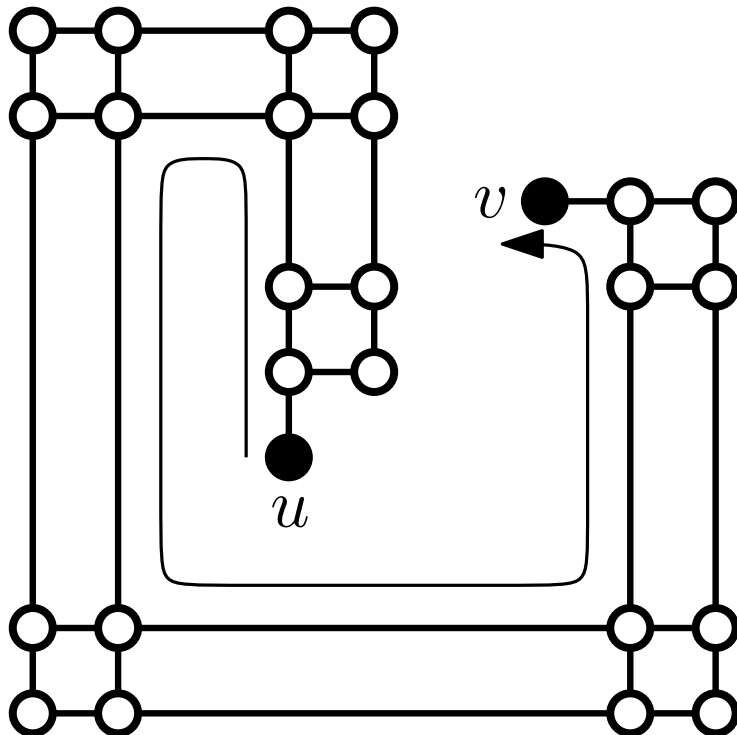
heute:

Satz

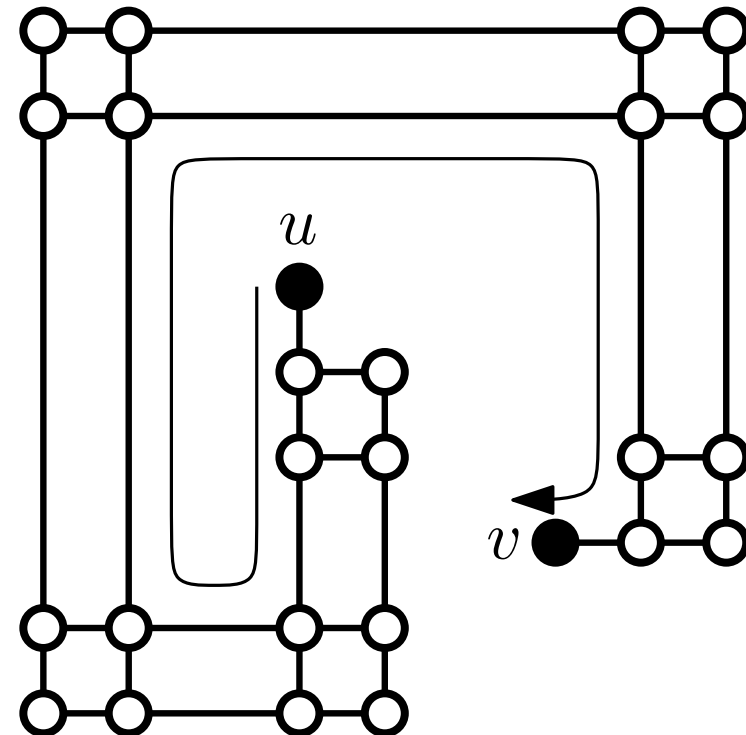
G Maxgrad-4-Graph mit variabler Einbettung
Entscheiden, ob G orthogonale Zeichnung ohne Knicke besitzt
ist NP-schwer.

Schwierigkeit bei 0-Knick-Zeichenbarkeit

Es gibt **starre** Konstruktionen



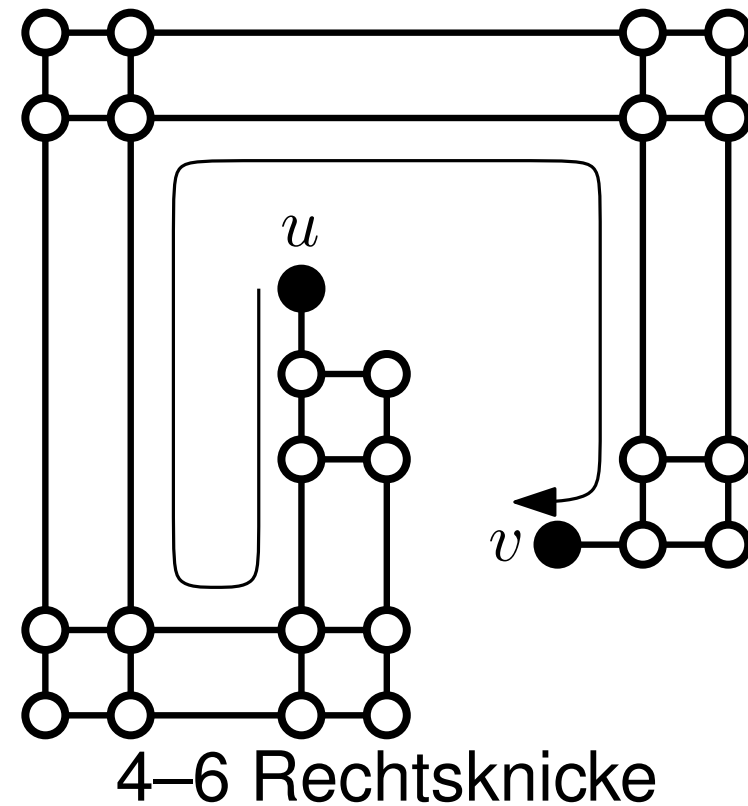
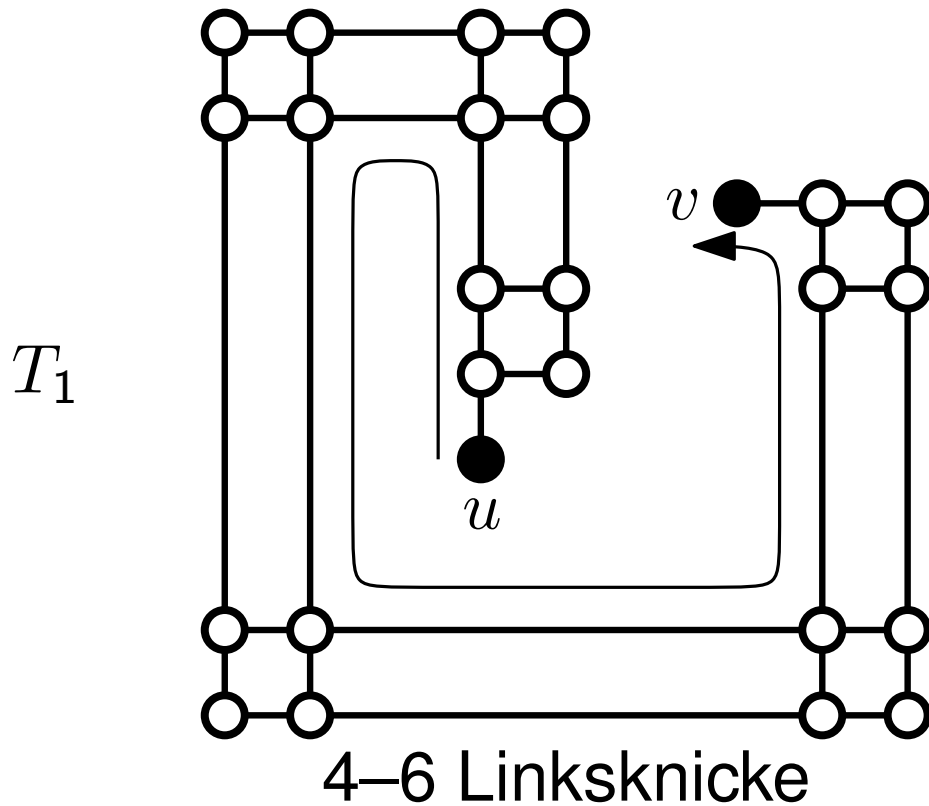
4–6 Linksknicke



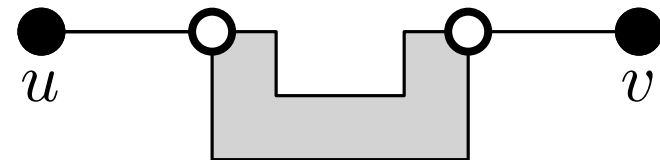
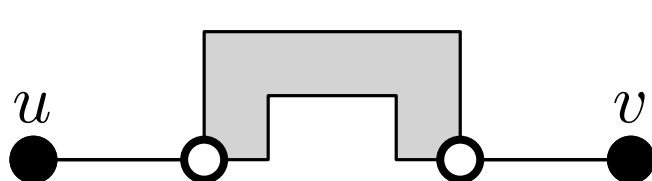
4–6 Rechtsknicke

Schwierigkeit bei 0-Knick-Zeichenbarkeit

Es gibt **starre** Konstruktionen



Tendril T_k : Konstruktion mit $4k$ bis $4k + 2$ Links- oder Rechtsknicken

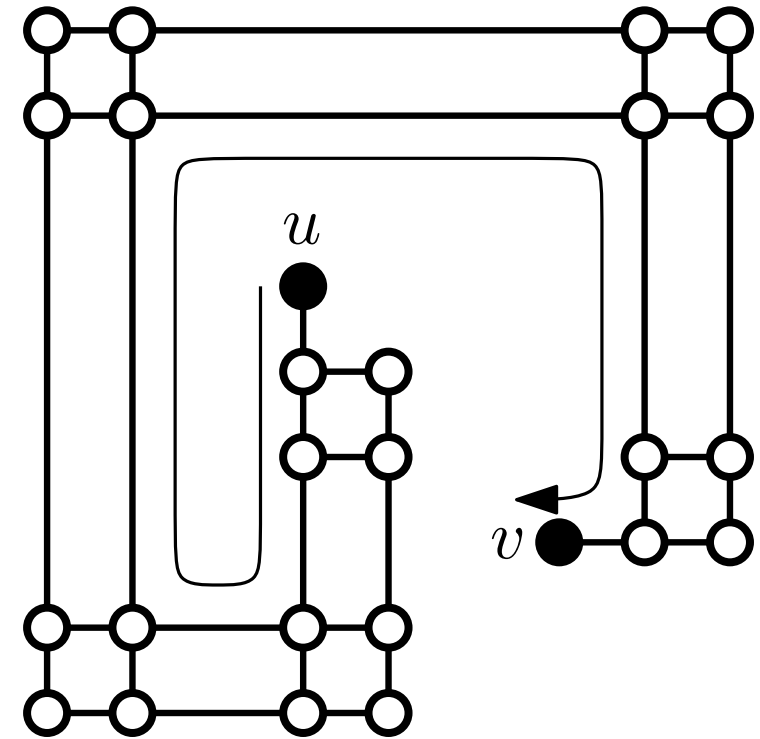
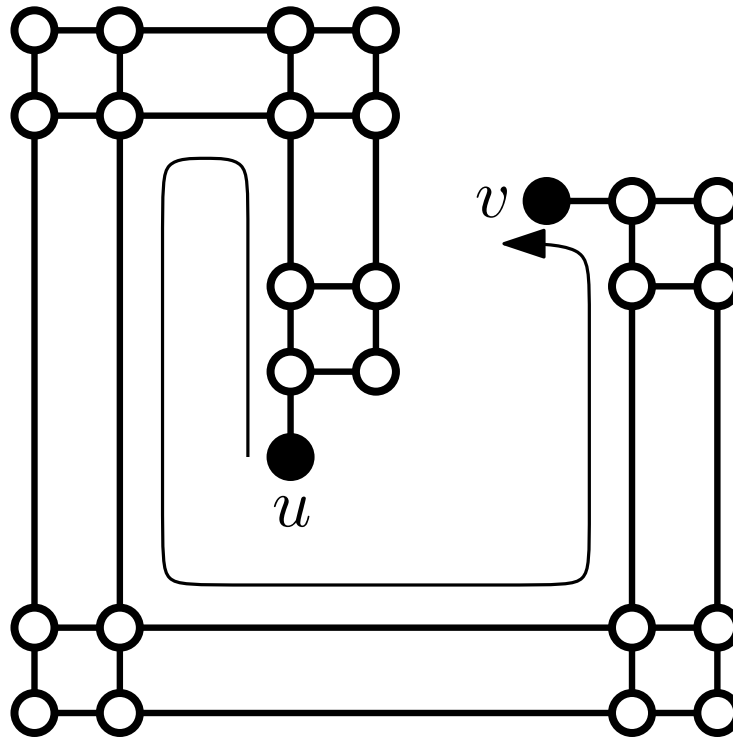


Zwei Gadgets

Tendrils

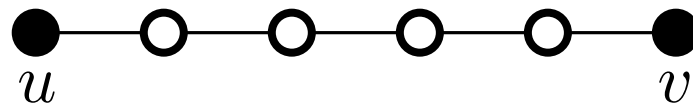
T_k

$4k + 1$ Knicke



Wiggles

W_k



0 bis $4k$ Knicke

Pfad der Länge $4k + 1$

Problem (Switch-Flow Network)

Eingabe:

- ungerichtetes Netzwerk $N = (V, E)$
- für jede Kante einen Kapazitätsbereichs $[c' \dots c'']$
- schreibe $[c]$ für $[c \dots c]$

Gültiger Fluß:

- Orientierung der Kanten
- Flußzuweisung an jede Kante gemäß Kapazitätsbereich
- Flußerhaltung gilt in jedem Knoten

Existiert ein gültiger Fluß?

Problem (Switch-Flow Network)

Eingabe:

- ungerichtetes Netzwerk $N = (V, E)$
- für jede Kante einen Kapazitätsbereichs $[c' \dots c'']$
- schreibe $[c]$ für $[c \dots c]$

Gültiger Fluß:

- Orientierung der Kanten
- Flußzuweisung an jede Kante gemäß Kapazitätsbereich
- Flußerhaltung gilt in jedem Knoten

Existiert ein gültiger Fluß?

Satz

Switch-Flow Network ist NP-schwer, sogar wenn N planar und 3-fach zusammenhängend ist.

Not-All-Equal 3SAT

Problem Not-All-Equal 3SAT:

Gegeben: 3SAT-Formel Φ

Gesucht: Variablenbelegung, sodass in keiner Klausel
alle Literale gleich belegt sind

Beispiel: $\Phi = (\neg x_1, x_2, \neg x_3), (\neg x_1, \neg x_2, x_3), (x_1, x_2, x_3)$

- $x_1 = \text{true}, x_2 = \text{false}, x_3 = \text{true}$ ist erfüllend
- $x_1 = \text{true}, x_2 = \text{true}, x_3 = \text{true}$ ist **nicht** erfüllend

Not-All-Equal 3SAT

Problem Not-All-Equal 3SAT:

Gegeben: 3SAT-Formel Φ

Gesucht: Variablenbelegung, sodass in keiner Klausel
alle Literale gleich belegt sind

Beispiel: $\Phi = (\neg x_1, x_2, \neg x_3), (\neg x_1, \neg x_2, x_3), (x_1, x_2, x_3)$

- $x_1 = \text{true}, x_2 = \text{false}, x_3 = \text{true}$ ist erfüllend
- $x_1 = \text{true}, x_2 = \text{true}, x_3 = \text{true}$ ist **nicht** erfüllend

Satz

Not-All-Equal 3SAT ist NP-schwer.

Achtung: **Planares** Not-All-Equal 3SAT ist polynomiell lösbar.

(Äquivalent zu MAXCUT auf planaren Graphen,
vgl. Vorlesung „Algorithmen für planare Graphen“)

Reduktion von Not-All-Equal 3SAT

Φ Instanz von Not-All-Equal 3SAT

Literale: $x_1, y_1, \dots, x_n, y_n$ mit $y_i = \neg x_i$

Klauseln: c_1, \dots, c_m

Reduktion von Not-All-Equal 3SAT

Φ Instanz von Not-All-Equal 3SAT

Literale: $x_1, y_1, \dots, x_n, y_n$ mit $y_i = \neg x_i$

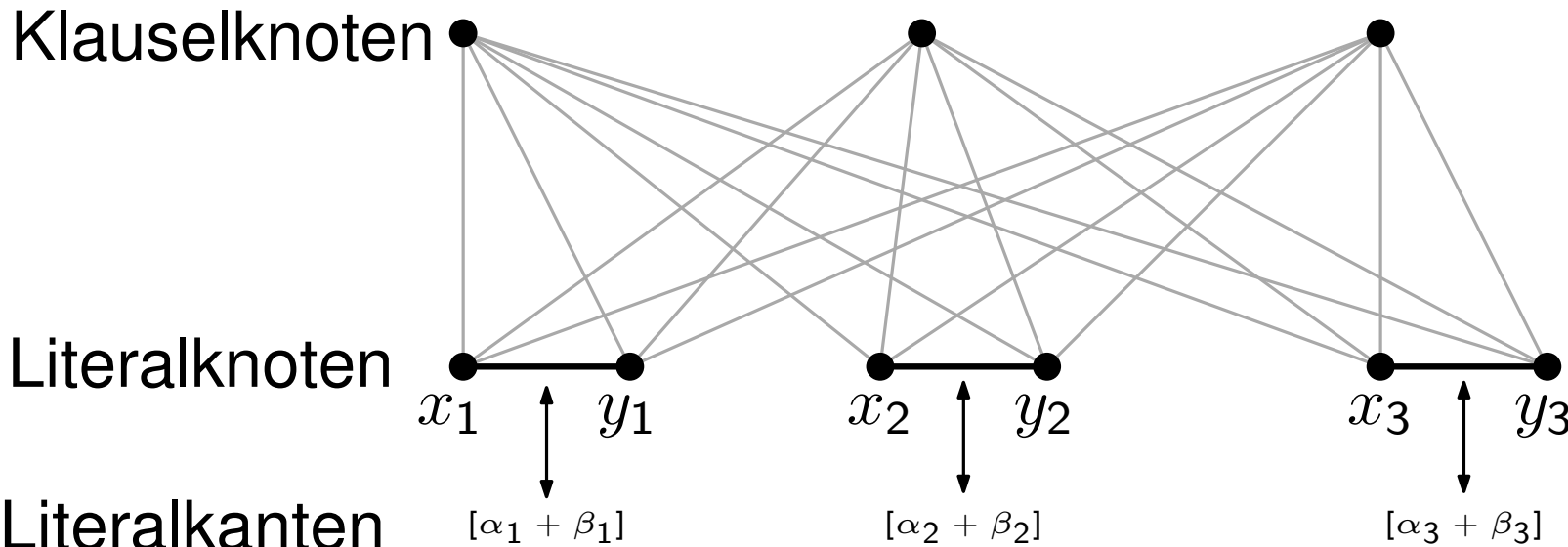
Klauseln: c_1, \dots, c_m

$\alpha_i := \#$ Vorkommen von x_i in Klauseln von Φ

$\beta_i := \#$ Vorkommen von y_i in Klauseln von Φ

Beachte: $\sum_{i=1}^n (\alpha_i + \beta_i) = 3m$

$(\neg x_1, x_2, \neg x_3), (\neg x_1, \neg x_2, x_3), (x_1, x_2, x_3)$



Reduktion von Not-All-Equal 3SAT

Φ Instanz von Not-All-Equal 3SAT

Literale: $x_1, y_1, \dots, x_n, y_n$ mit $y_i = \neg x_i$

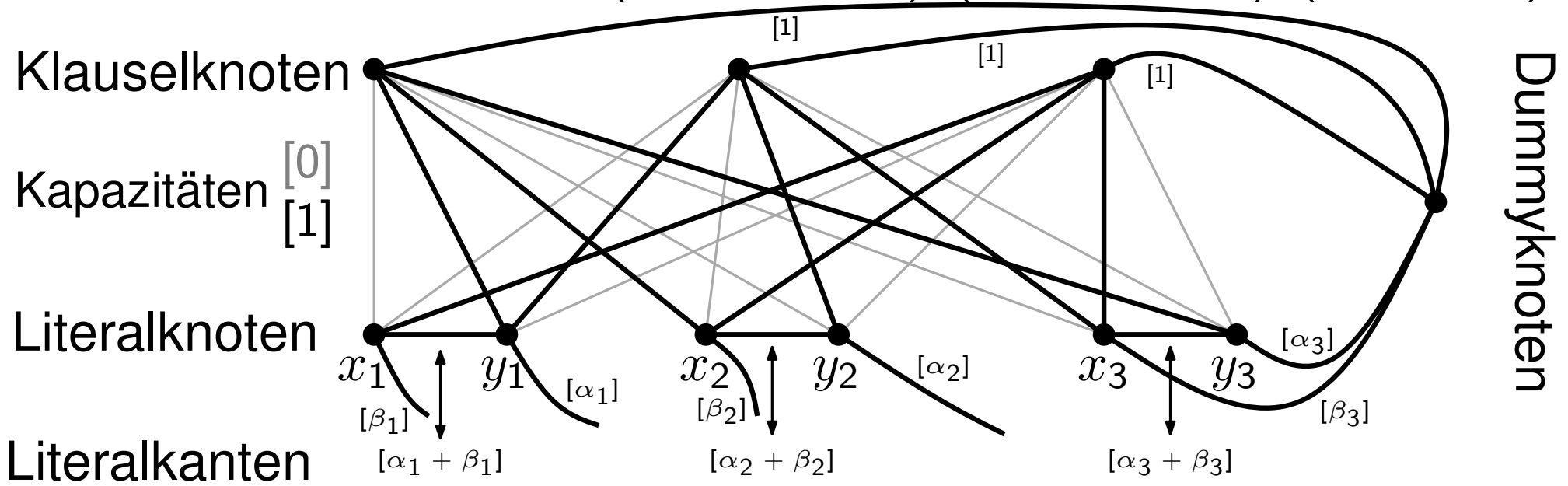
Klauseln: c_1, \dots, c_m

$\alpha_i := \#$ Vorkommen von x_i in Klauseln von Φ

$\beta_i := \#$ Vorkommen von y_i in Klauseln von Φ

Beachte: $\sum_{i=1}^n (\alpha_i + \beta_i) = 3m$

$(\neg x_1, x_2, \neg x_3), (\neg x_1, \neg x_2, x_3), (x_1, x_2, x_3)$



Reduktion von Not-All-Equal 3SAT

Φ Instanz von Not-All-Equal 3SAT

Literale: $x_1, y_1, \dots, x_n, y_n$ mit $y_i = \neg x_i$

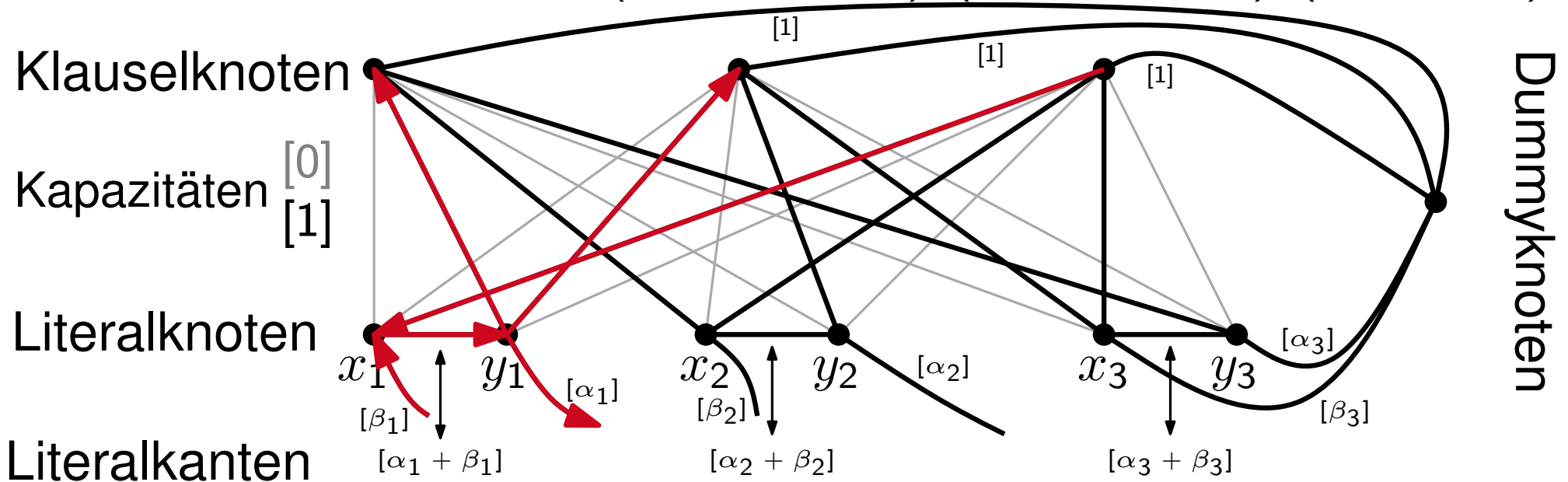
Klauseln: c_1, \dots, c_m

$\alpha_i := \#$ Vorkommen von x_i in Klauseln von Φ

$\beta_i := \#$ Vorkommen von y_i in Klauseln von Φ

Beachte: $\sum_{i=1}^n (\alpha_i + \beta_i) = 3m$

$(\neg x_1, x_2, \neg x_3), (\neg x_1, \neg x_2, x_3), (x_1, x_2, x_3)$



Reduktion von Not-All-Equal 3SAT

Φ Instanz von Not-All-Equal 3SAT

Literale: $x_1, y_1, \dots, x_n, y_n$ mit $y_i = \neg x_i$

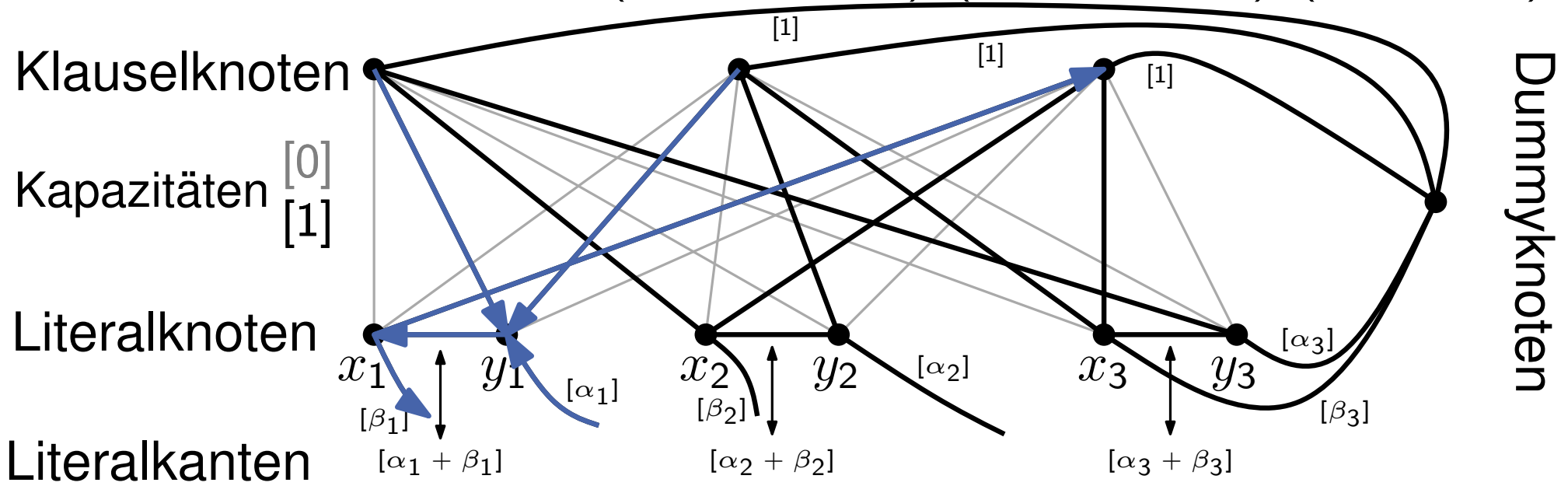
Klauseln: c_1, \dots, c_m

$\alpha_i := \#$ Vorkommen von x_i in Klauseln von Φ

$\beta_i := \#$ Vorkommen von y_i in Klauseln von Φ

Beachte: $\sum_{i=1}^n (\alpha_i + \beta_i) = 3m$

$(\neg x_1, x_2, \neg x_3), (\neg x_1, \neg x_2, x_3), (x_1, x_2, x_3)$



Reduktion von Not-All-Equal 3SAT

Φ Instanz von Not-All-Equal 3SAT

Literale: $x_1, y_1, \dots, x_n, y_n$ mit $y_i = \neg x_i$

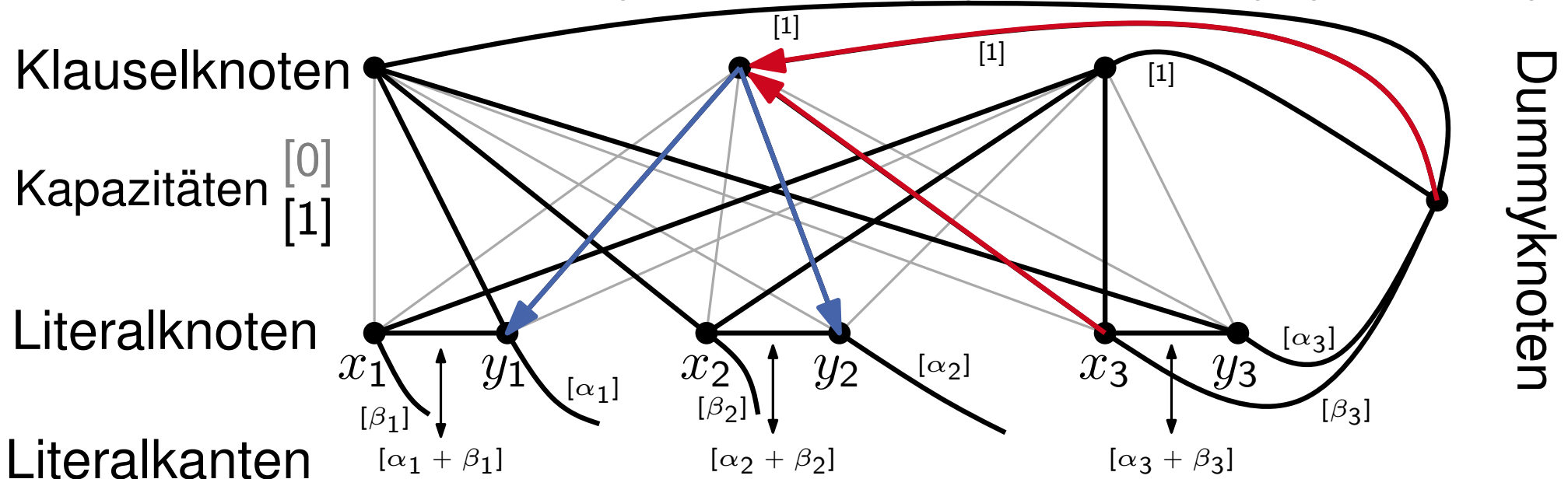
Klauseln: c_1, \dots, c_m

$\alpha_i := \#$ Vorkommen von x_i in Klauseln von Φ

$\beta_i := \#$ Vorkommen von y_i in Klauseln von Φ

Beachte: $\sum_{i=1}^n (\alpha_i + \beta_i) = 3m$

$(\neg x_1, x_2, \neg x_3), (\neg x_1, \neg x_2, x_3), (x_1, x_2, x_3)$



Reduktion von Not-All-Equal 3SAT

Φ Instanz von Not-All-Equal 3SAT

Literale: $x_1, y_1, \dots, x_n, y_n$ mit $y_i = \neg x_i$

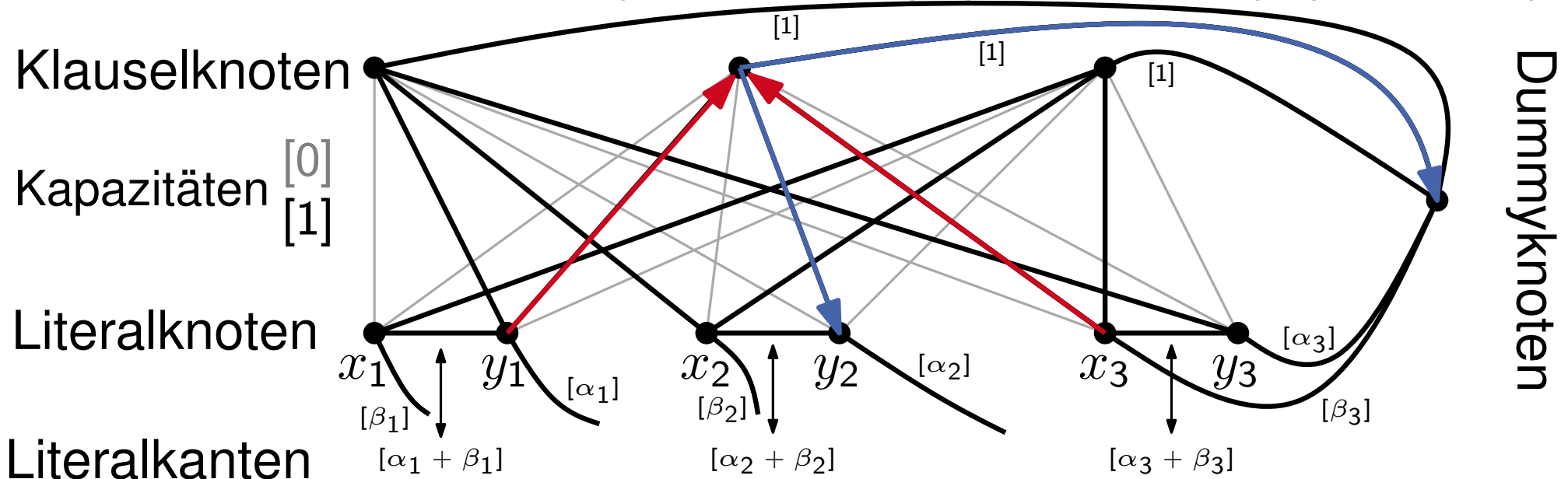
Klauseln: c_1, \dots, c_m

$\alpha_i := \#$ Vorkommen von x_i in Klauseln von Φ

$\beta_i := \#$ Vorkommen von y_i in Klauseln von Φ

Beachte: $\sum_{i=1}^n (\alpha_i + \beta_i) = 3m$

$(\neg x_1, x_2, \neg x_3), (\neg x_1, \neg x_2, x_3), (x_1, x_2, x_3)$



Reduktion von Not-All-Equal 3SAT

Φ Instanz von Not-All-Equal 3SAT

Literale: $x_1, y_1, \dots, x_n, y_n$ mit $y_i = \neg x_i$

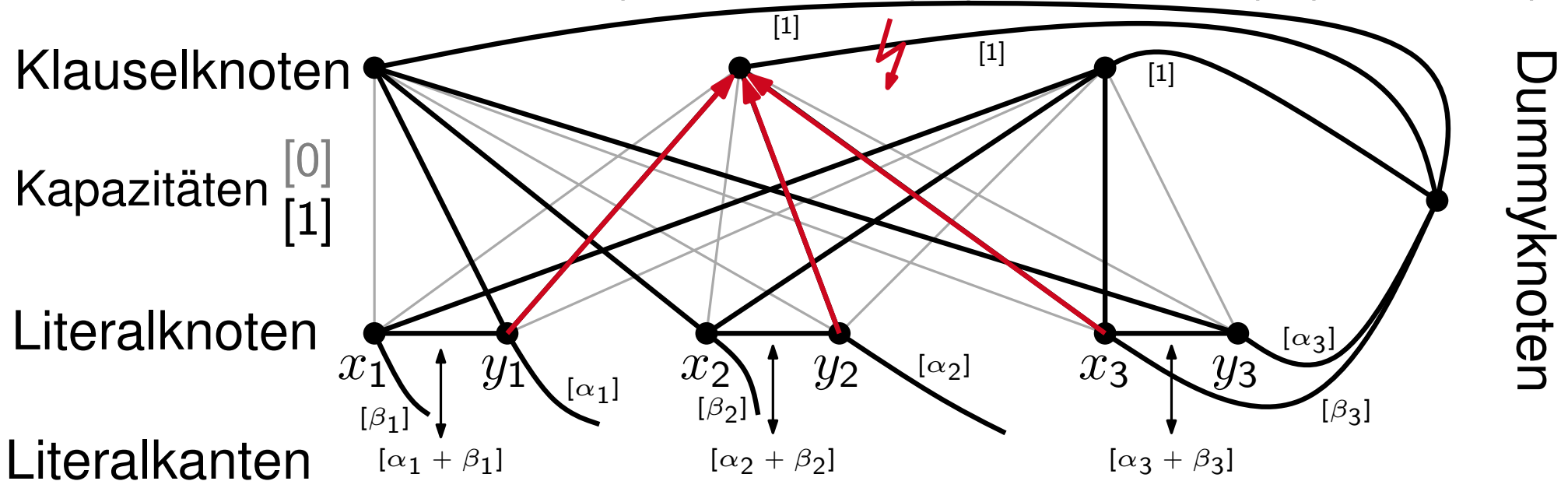
Klauseln: c_1, \dots, c_m

$\alpha_i := \#$ Vorkommen von x_i in Klauseln von Φ

$\beta_i := \#$ Vorkommen von y_i in Klauseln von Φ

Beachte: $\sum_{i=1}^n (\alpha_i + \beta_i) = 3m$

$(\neg x_1, x_2, \neg x_3), (\neg x_1, \neg x_2, x_3), (x_1, x_2, x_3)$



Reduktion von Not-All-Equal 3SAT

Φ Instanz von Not-All-Equal 3SAT

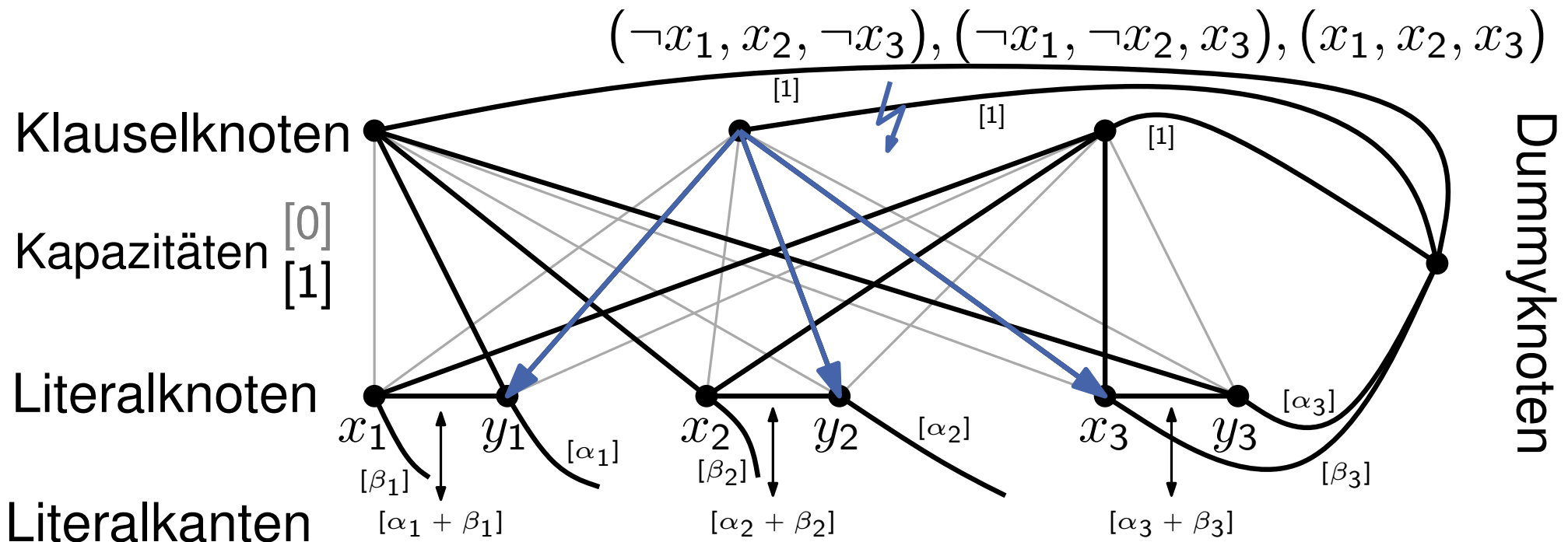
Literale: $x_1, y_1, \dots, x_n, y_n$ mit $y_i = \neg x_i$

Klauseln: c_1, \dots, c_m

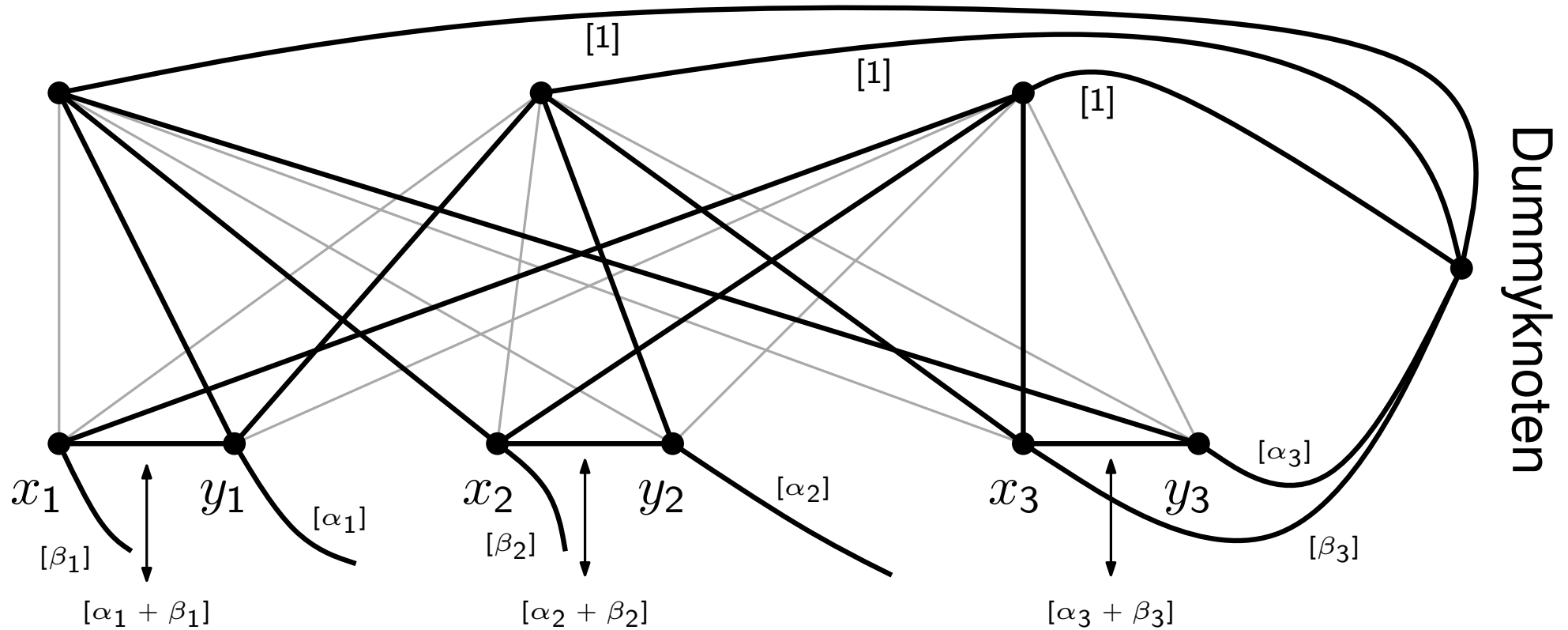
$\alpha_i := \#$ Vorkommen von x_i in Klauseln von Φ

$\beta_i := \#$ Vorkommen von y_i in Klauseln von Φ

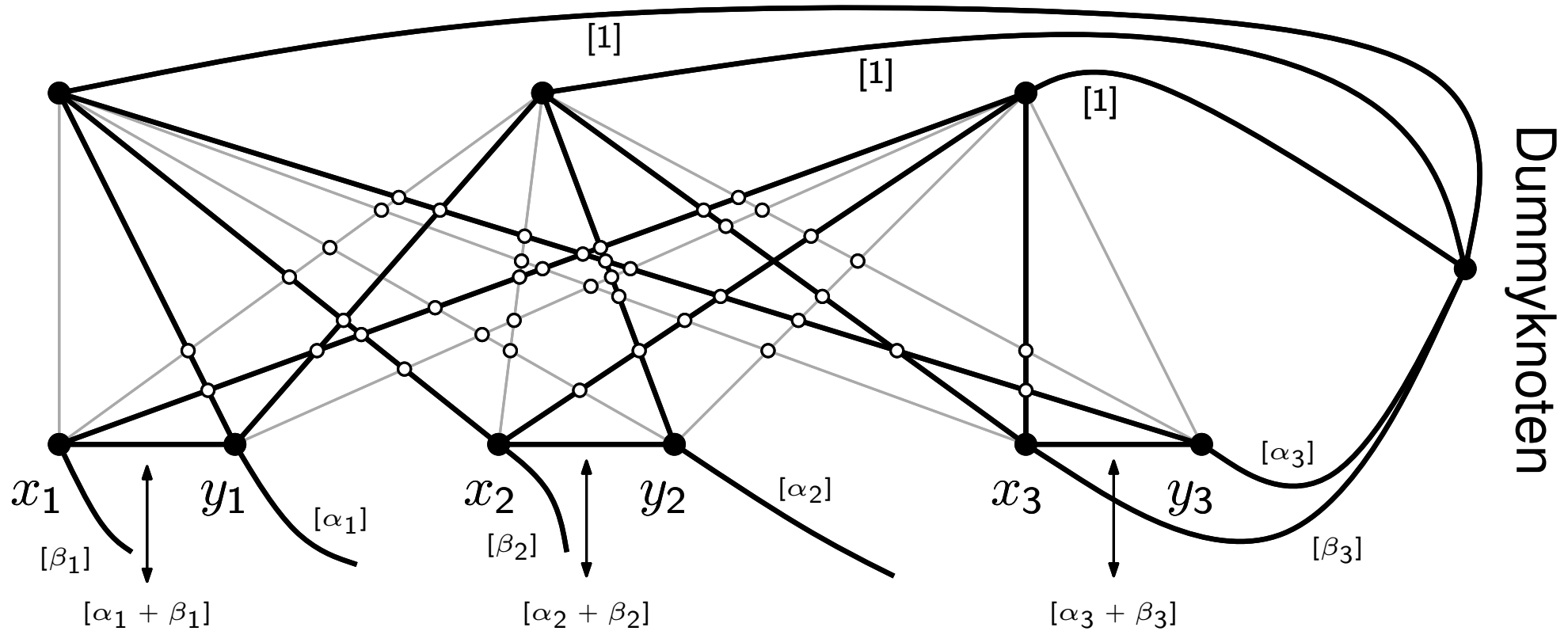
Beachte: $\sum_{i=1}^n (\alpha_i + \beta_i) = 3m$



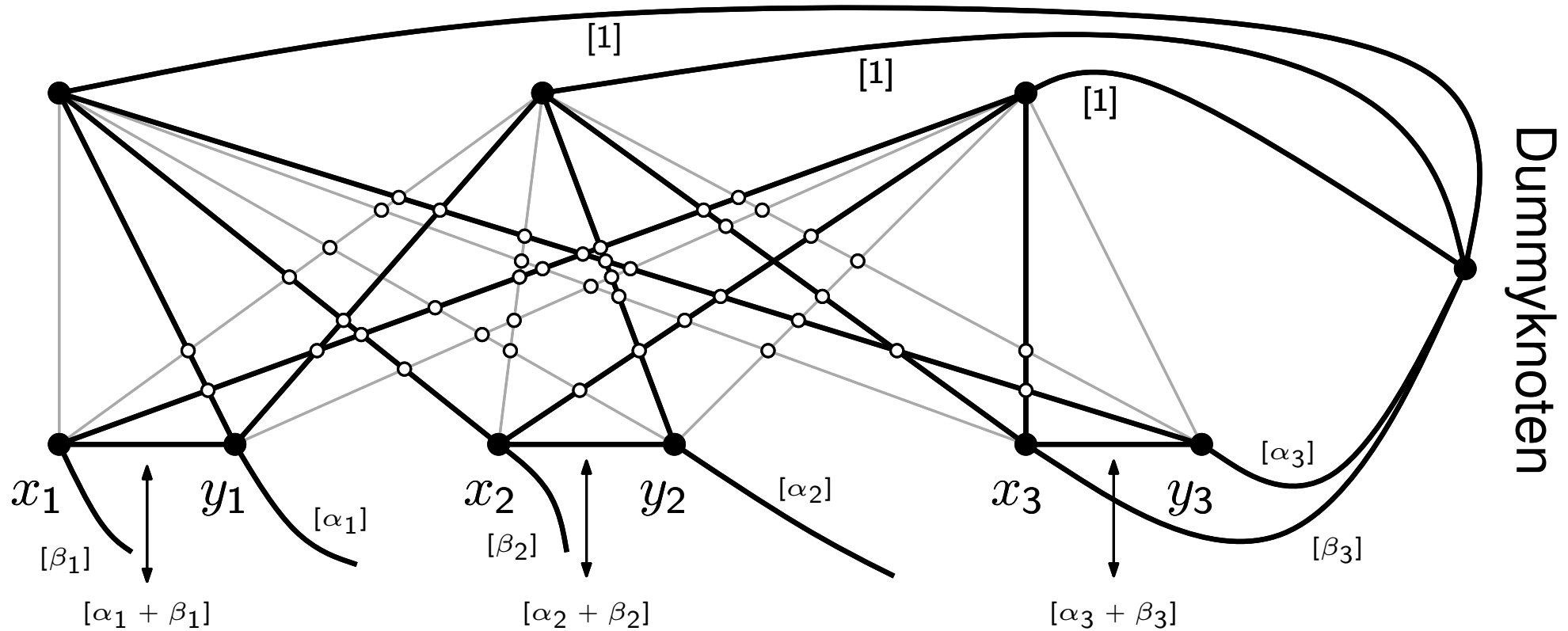
Herstellung von Planarität



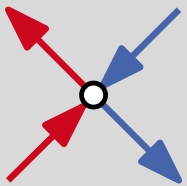
Herstellung von Planarität



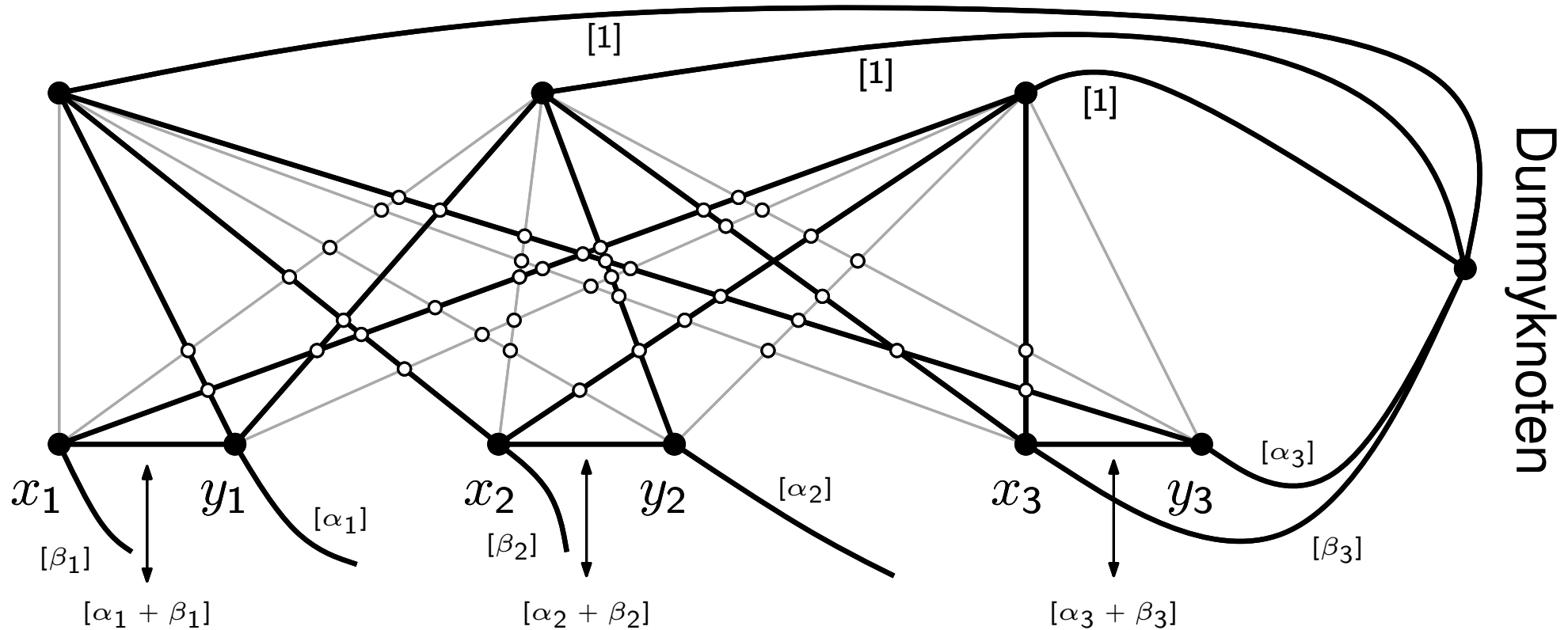
Herstellung von Planarität



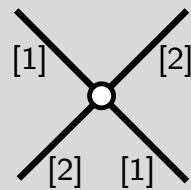
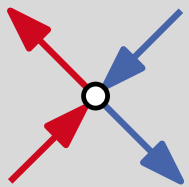
Problem:



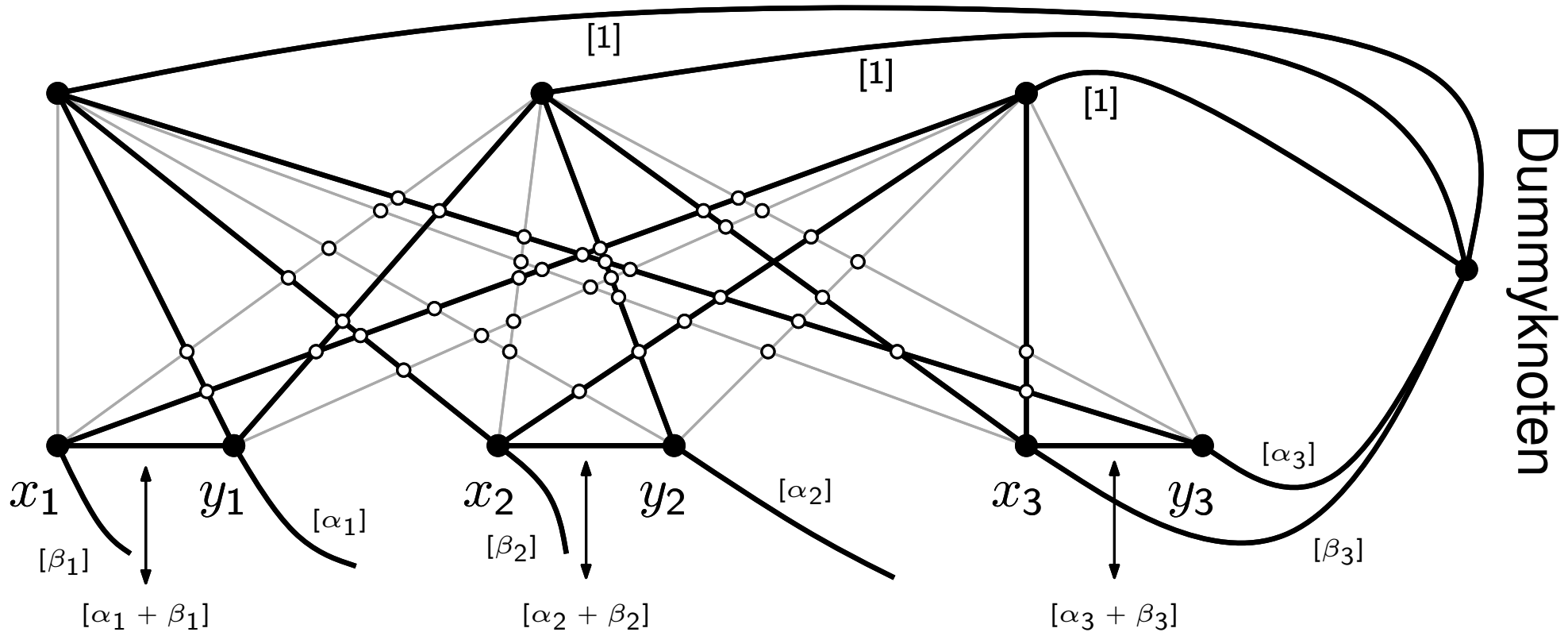
Herstellung von Planarität



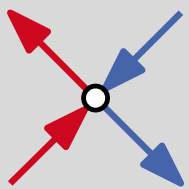
Problem: Lösungsidee:



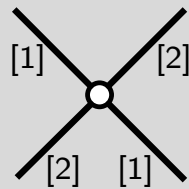
Herstellung von Planarität



Problem:



Lösungsidee:



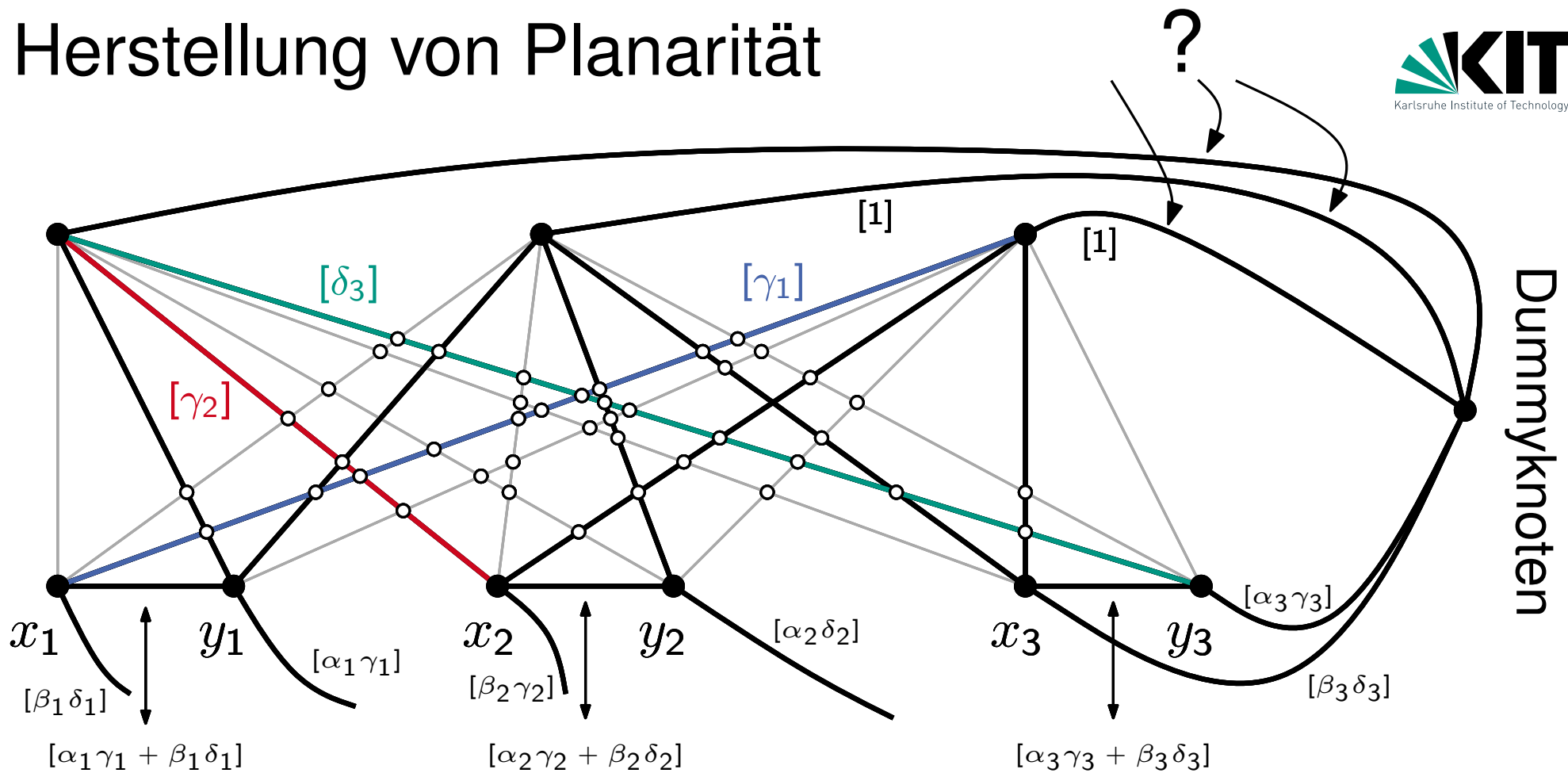
Allgemeiner:

Verschiedene Flußwerte für verschiedene Literale

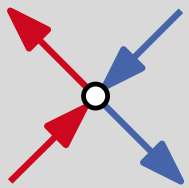
$$\text{für } x_i : \gamma_i := (2i - 1)\Theta$$

$$\text{für } y_i : \delta_i := 2i\Theta$$

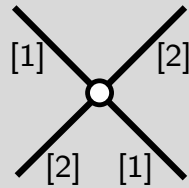
Herstellung von Planarität



Problem:



Lösungsidee:

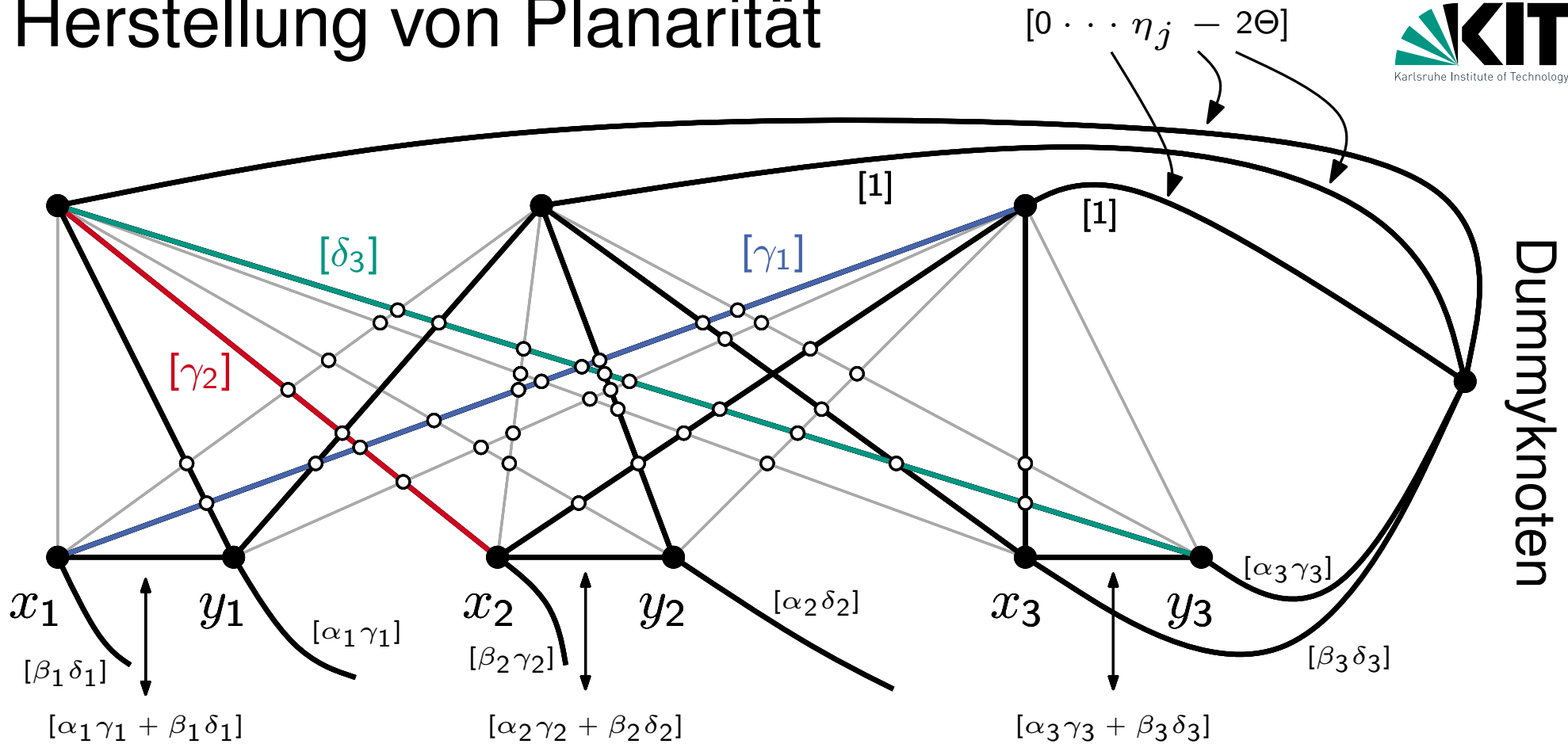


Allgemeiner: Verschiedene Flußwerte für verschiedene Literale

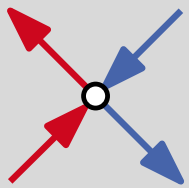
$$\text{für } x_i : \gamma_i := (2i - 1)\Theta$$

$$\text{für } y_i : \delta_i := 2i\Theta$$

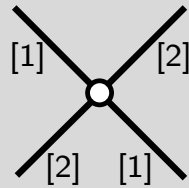
Herstellung von Planarität



Problem:



Lösungsidee:



Allgemeiner: Verschiedene Flußwerte für verschiedene Literale

für $x_i : \gamma_i := (2i - 1)\Theta$

für $y_i : \delta_i := 2i\Theta$

$\eta_j :=$ Summe der Kapazitäten inzidenter Literal-Klausel-Kanten für Klausel j

Konstruierter Graph ist 3-fach zusammenhängend

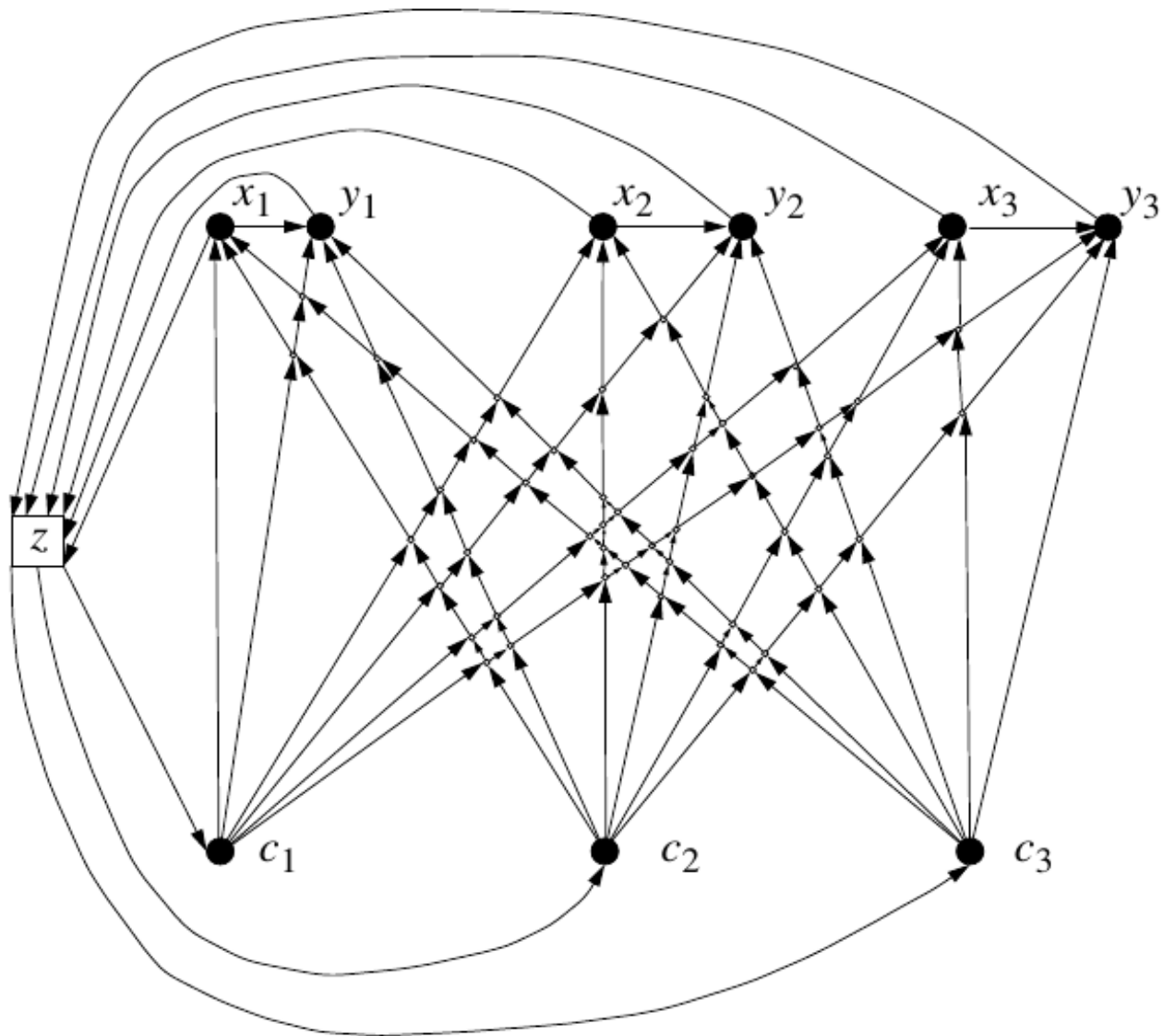
Satz

Switch-Flow Network ist NP-schwer, sogar wenn N planar und 3-fach zusammenhängend ist.

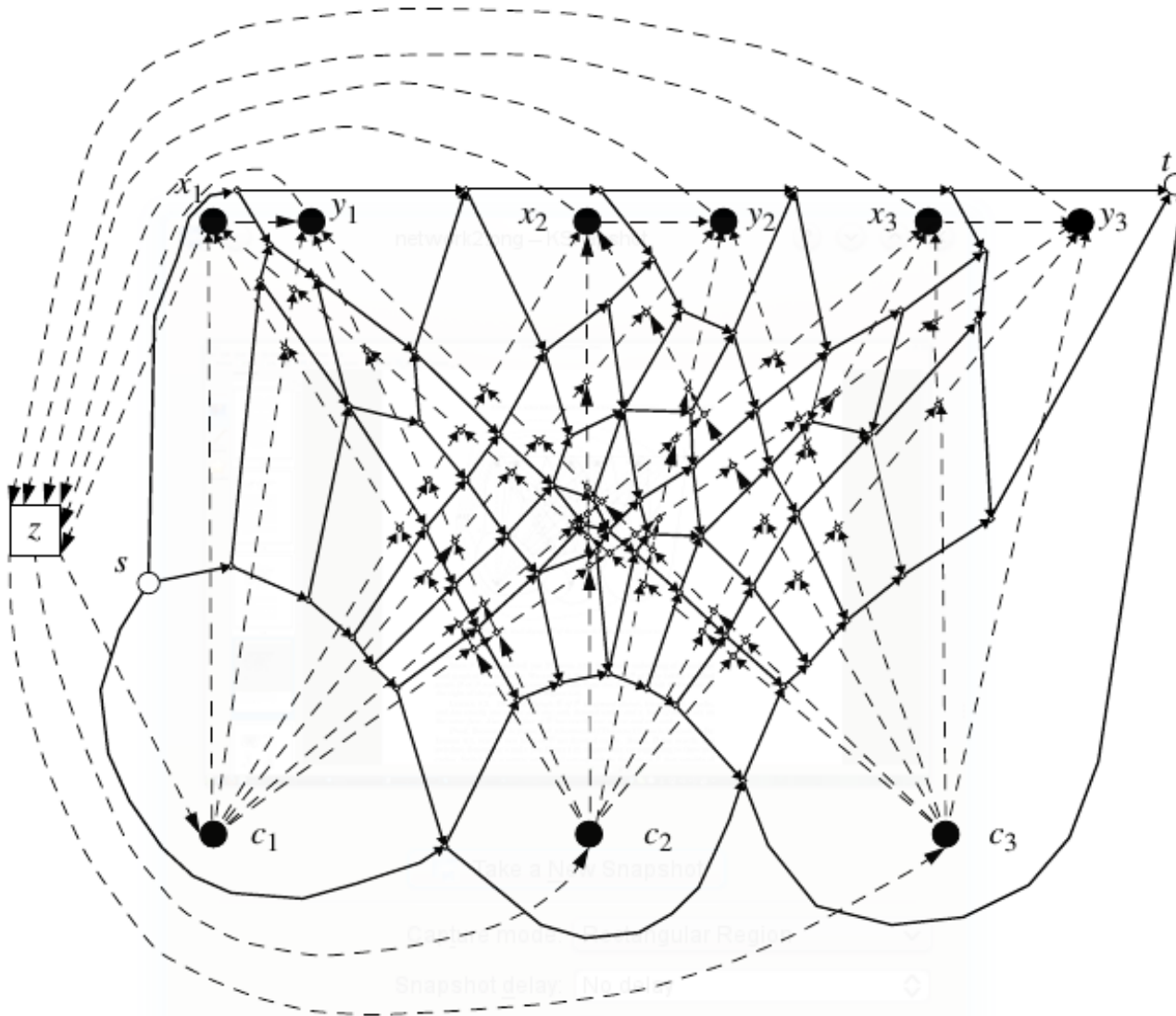
Idee: Konstruiere Graph G mit
 G besitzt Zeichnung ohne Knicke $\Leftrightarrow N$ lösbar.

- N hat eindeutige planare Einbettung
- Verwende Dualgraph von N als Basis
- Ersetze Kanten des Dualgraphs durch Konstruktionen, die den Fluß modellieren

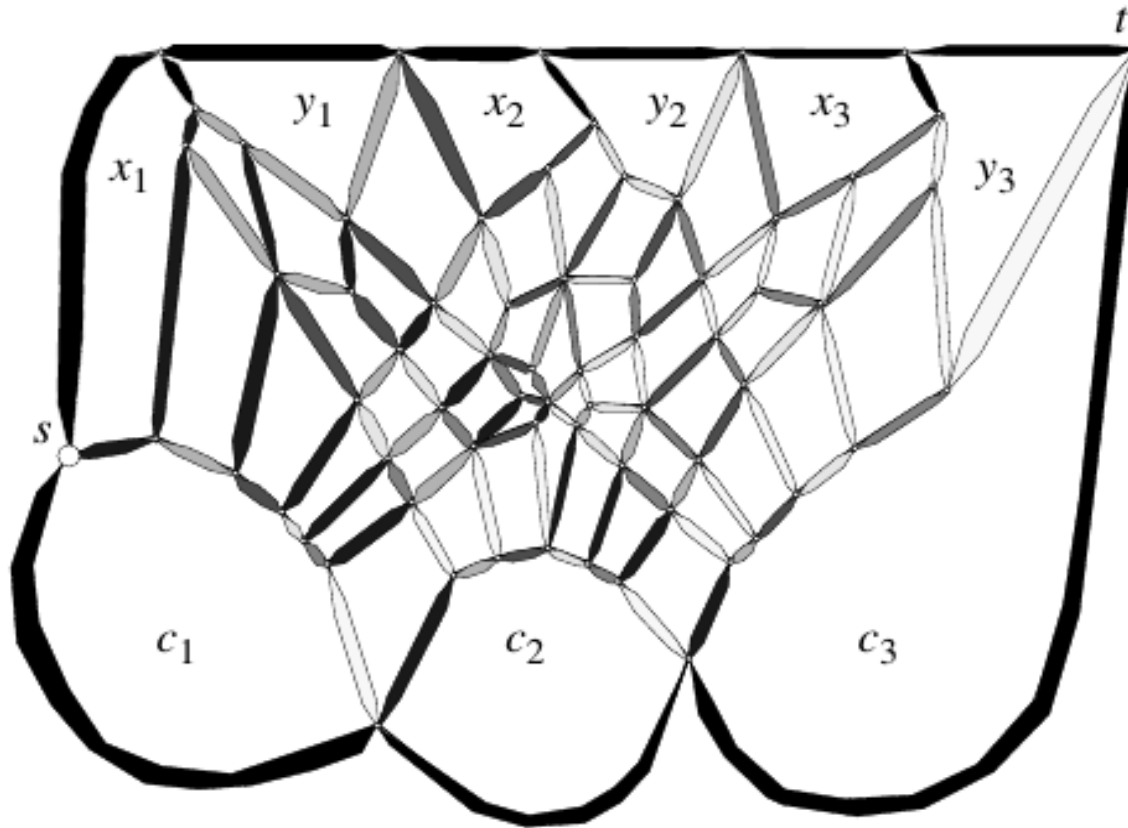
Konstruktion von G



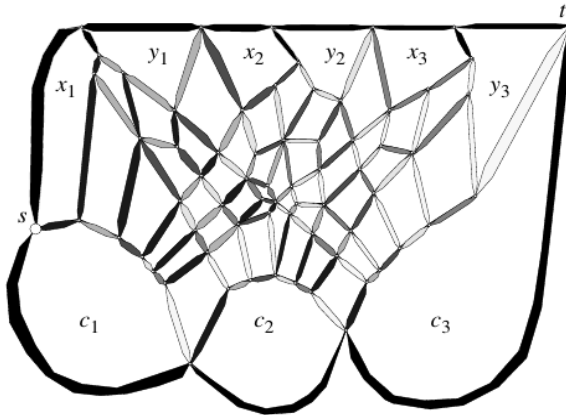
Konstruktion von G



Konstruktion von G



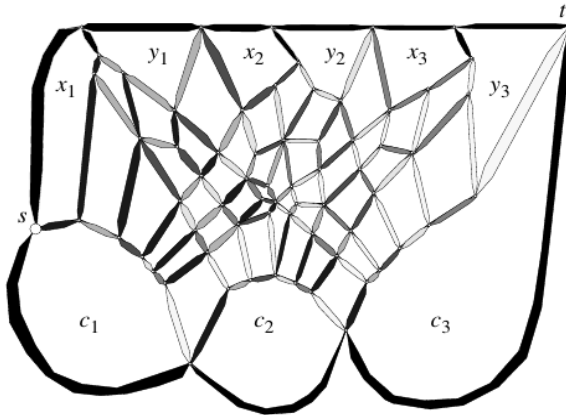
Konstruktion von G



Ersetze:

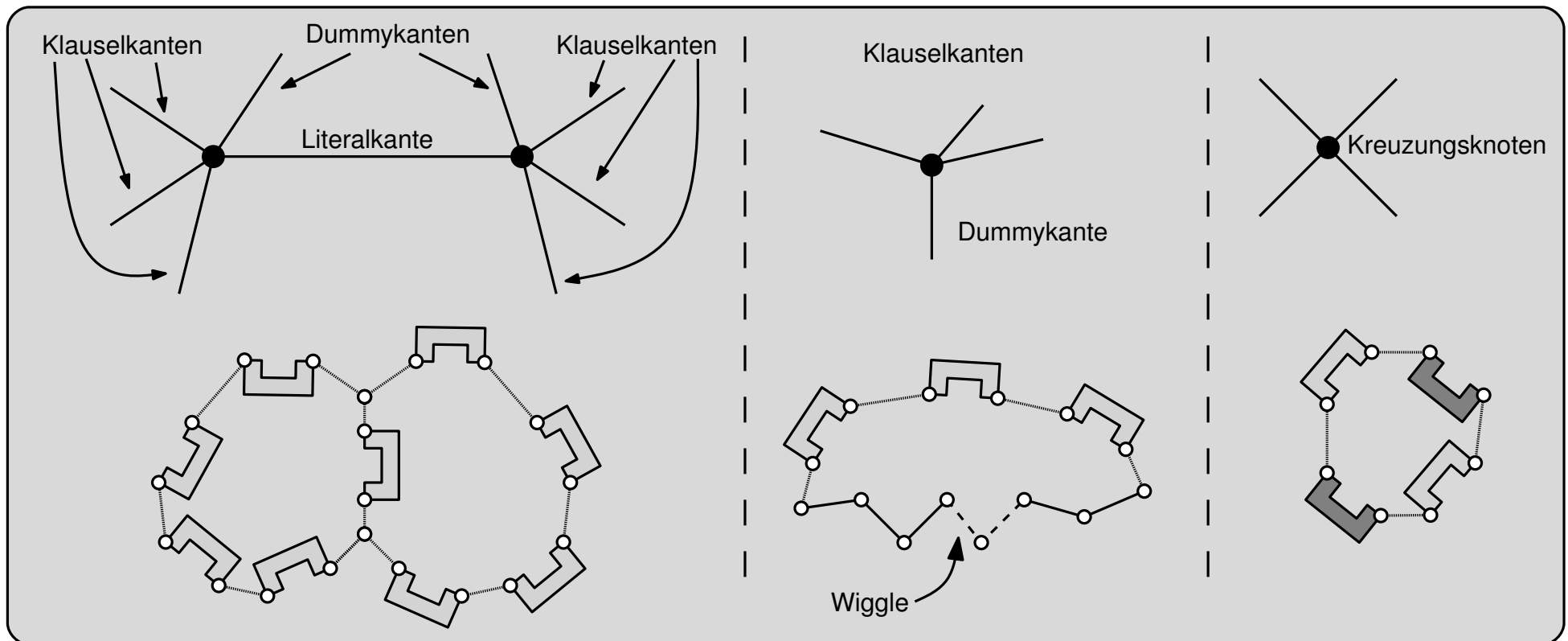
- Kante mit Kapazität $[c] \rightarrow$ Tendril T_c
- Kante mit Kapazität $[0 \dots c] \rightarrow$ Wiggle W_c

Konstruktion von G



Ersetze:

- Kante mit Kapazität $[c] \rightarrow$ Tendril T_c
- Kante mit Kapazität $[0 \dots c] \rightarrow$ Wiggle W_c

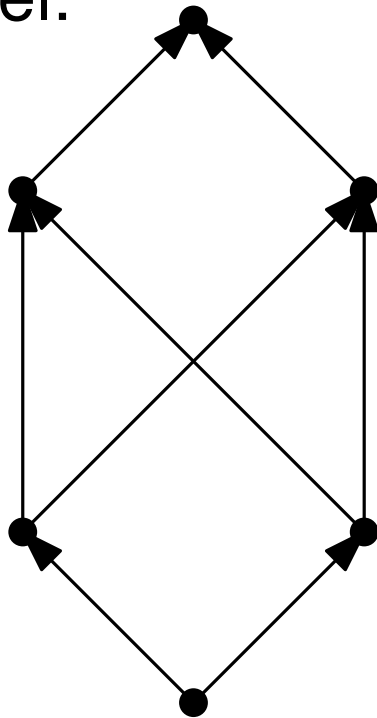


Aufwärtsplanare Zeichnungen

Definition

Ein gerichteter azyklischer Graph $D = (V, A)$ heißt **aufwärtsplanar**, wenn es eine planare Einbettung von D in die Ebene gibt, bei der alle Kanten aufwärtsgerichtet sind.

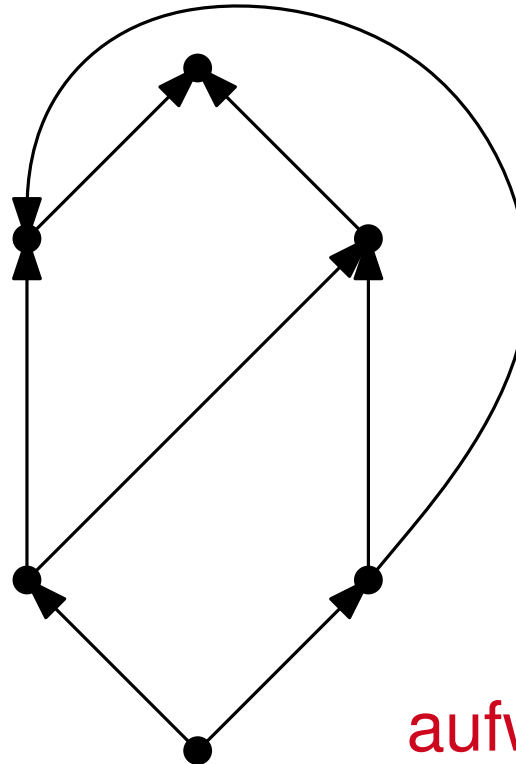
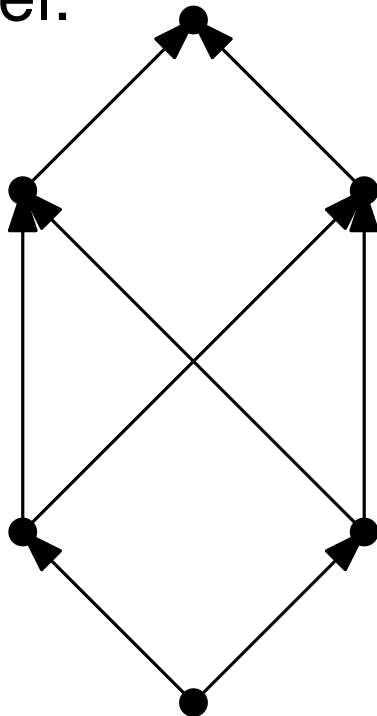
Beispiel:



Definition

Ein gerichteter azyklischer Graph $D = (V, A)$ heißt **aufwärtsplanar**, wenn es eine planare Einbettung von D in die Ebene gibt, bei der alle Kanten aufwärtsgerichtet sind.

Beispiel:



planar!

aufwärtsplanar? – Nein!

Problem: Test auf Aufwärtsplanarität

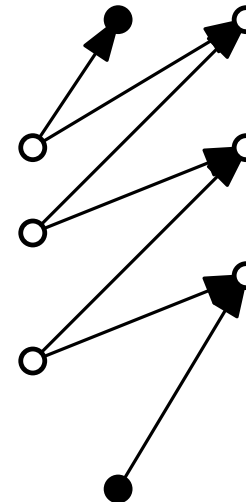
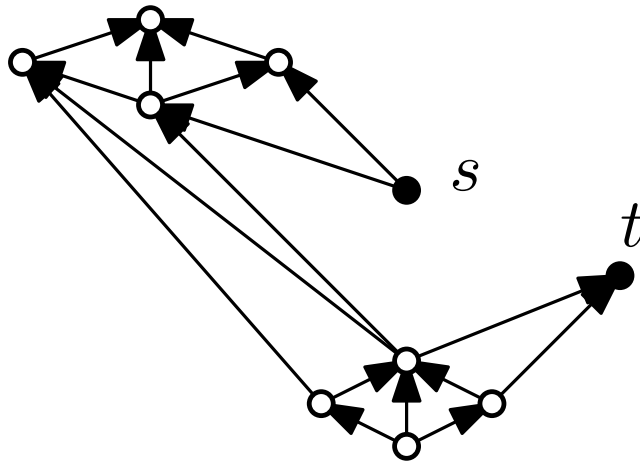
Gegeben ein gerichteter azyklischer Graph $D = (V, A)$. Teste, ob D aufwärtsplanar ist. Falls D aufwärtsplanar ist, so konstruiere ein entsprechendes Layout

Problem: Test auf Aufwärtsplanarität

Gegeben ein gerichteter azyklischer Graph $D = (V, A)$. Teste, ob D aufwärtsplanar ist. Falls D aufwärtsplanar ist, so konstruiere ein entsprechendes Layout

Garg & Tamassia: Das Problem ist NP-schwer.

- ähnliche Konstruktion wie zuvor
- andere Tendrils und Wiggles



Trotzdem Charakterisierung

Definition:

DAG $D = (V, A)$ heißt st-Graph, wenn

- es ex. eindeutige Quelle s in V
- es ex. eindeutige Senke t in V
- Kante st ist in A enthalten

Trotzdem Charakterisierung

Definition:

DAG $D = (V, A)$ heißt st-Graph, wenn

- es ex. eindeutige Quelle s in V
- es ex. eindeutige Senke t in V
- Kante st ist in A enthalten

Satz (Charakterisierung aufwärtsplanarer Graphen)

Für einen gerichteten Graphen $D = (V, A)$ sind folgende Aussagen äquivalent:

1. D ist aufwärtsplanar
2. D hat ein geradliniges aufwärtsplanares Layout
3. D ist aufspannender Subgraph eines planaren st-Graphen

Trotzdem Charakterisierung

Definition:

DAG $D = (V, A)$ heißt st-Graph, wenn

- es ex. eindeutige Quelle s in V
- es ex. eindeutige Senke t in V
- Kante st ist in A enthalten

Satz (Charakterisierung aufwärtsplanarer Graphen)

Für einen gerichteten Graphen $D = (V, A)$ sind folgende Aussagen äquivalent:

1. D ist aufwärtsplanar
2. D hat ein geradliniges aufwärtsplanares Layout
3. D ist aufspannender Subgraph eines planaren st-Graphen

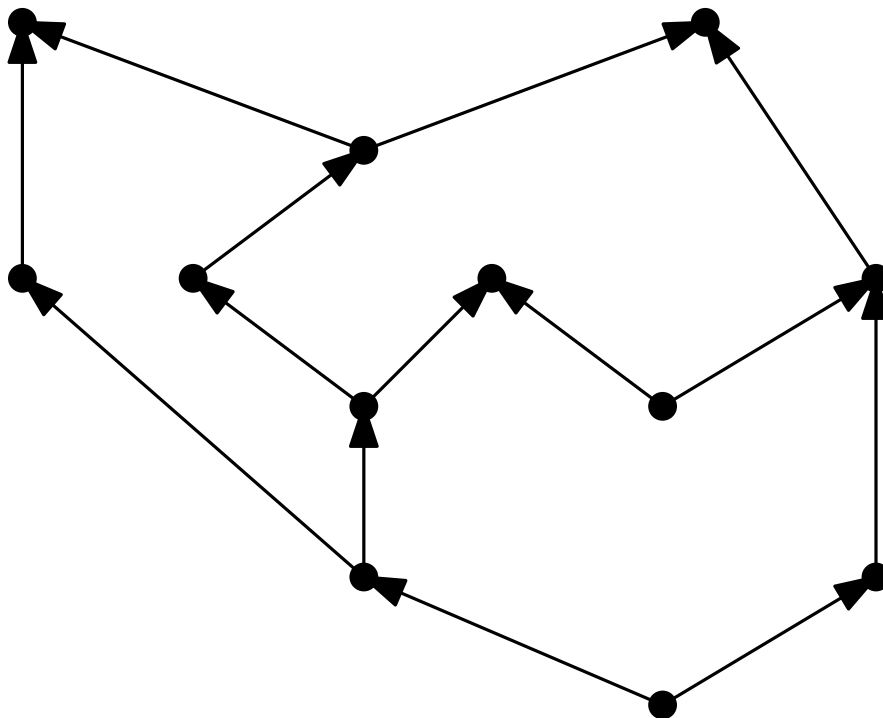
Beweis: (2) \Rightarrow (1) ist klar

(3) \Leftrightarrow (1) einfach

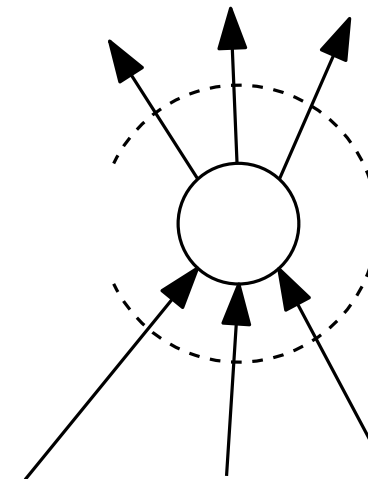
(3) \Rightarrow (2) braucht etwas mehr Arbeit

Problem: Test auf Aufwärtsplanarität mit fester Einbettung

Gegeben ein gerichteter azyklischer Graph $D = (V, A)$ mit Einbettung \mathcal{F}, f_0 . Teste, ob D, \mathcal{F}, f_0 aufwärtsplanar ist und konstruiere ggf. ein entsprechendes Layout



Einbettung ist **bimodal**



ausgehend

eingehend

Beobachtungen

Beobachtungen

- Bimodalität notwendig (nicht hinreichend)

Beobachtungen

- Bimodalität notwendig (nicht hinreichend)
- betrachte Winkel zw. zwei ein-/ausgehenden Kanten
Winkel α ist **groß** wenn $\alpha > \pi$, **klein** sonst.

$L(v) := \#$ große Winkel an Knoten v

$L(f) := \#$ große Winkel in Facette f .

$S(v)$ bzw. $S(f)$: Anzahl **kleiner** Winkel

nur zw. ein- bzw. aus-
gehenden Kanten!

Beobachtungen

- Bimodalität notwendig (nicht hinreichend)
- betrachte Winkel zw. zwei ein-/ausgehenden Kanten
Winkel α ist **groß** wenn $\alpha > \pi$, **klein** sonst.

$L(v) := \#$ große Winkel an Knoten v

$L(f) := \#$ große Winkel in Facette f .

$S(v)$ bzw. $S(f)$: Anzahl **kleiner** Winkel

nur zw. ein- bzw. ausgehenden Kanten!

Lemma:

- in Aufwärts-Layout von D gilt:

$$(1) \forall v \in V : L(v) = \begin{cases} 0 & v \text{ innerer Knoten} \\ 1 & v \text{ Quelle/Senke} \end{cases}$$

$$(2) \forall f \in \mathcal{F} : L(f) - S(f) = \begin{cases} -2 & \neq f_0 \\ 2 & f_0 \end{cases}$$

Folgerung

- $A(f) := \#$ Winkel zwischen zwei eingehenden Kanten an f
Es gilt stets: $L(f) + S(f) = 2A(f)$ für alle Facetten

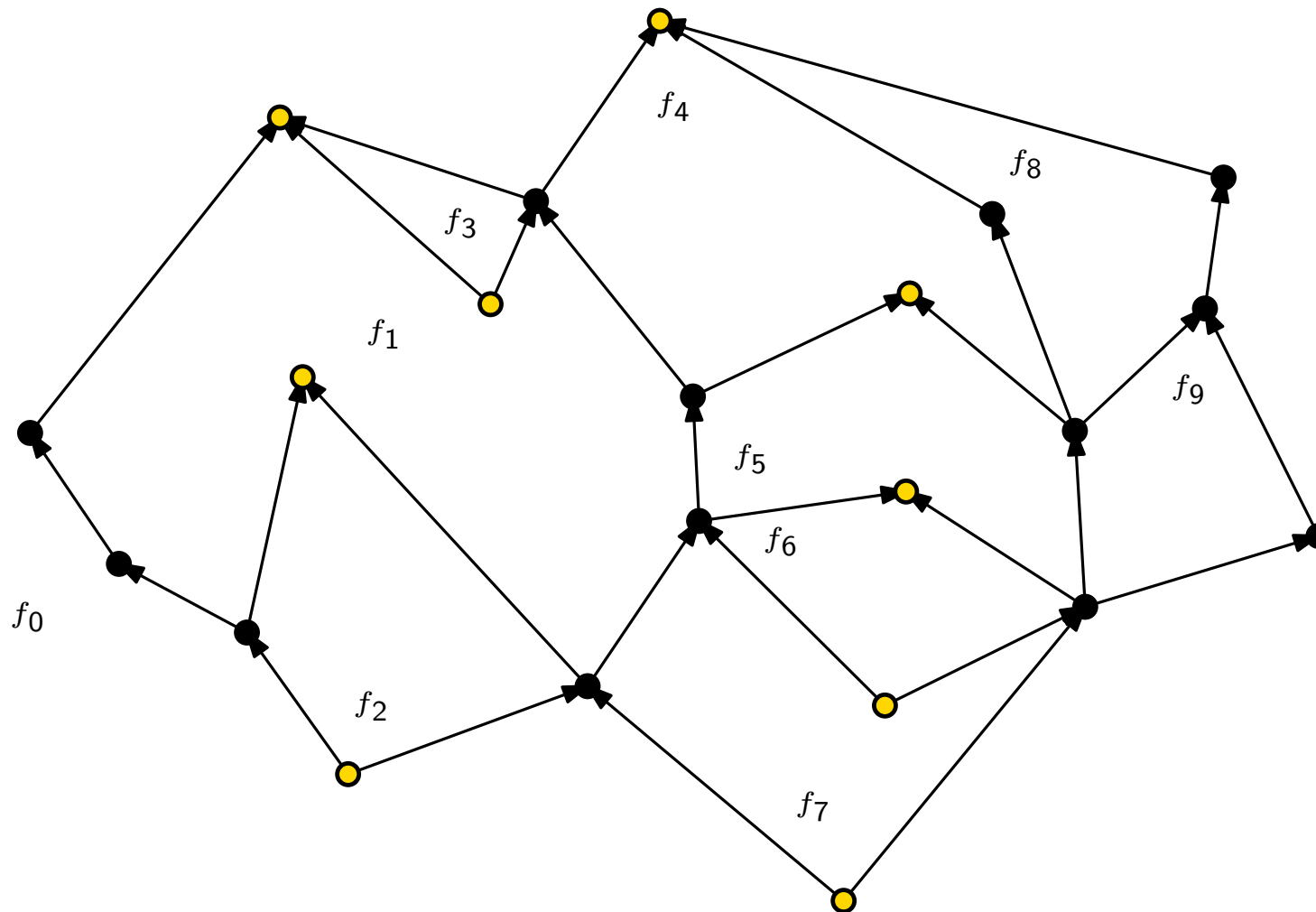
- in Aufwärts-Layout von D gilt:

$$(2) \forall f \in \mathcal{F} : L(f) = \begin{cases} A(f) - 1 & \neq f_0 \\ A(f) + 1 & f_0 \end{cases}$$

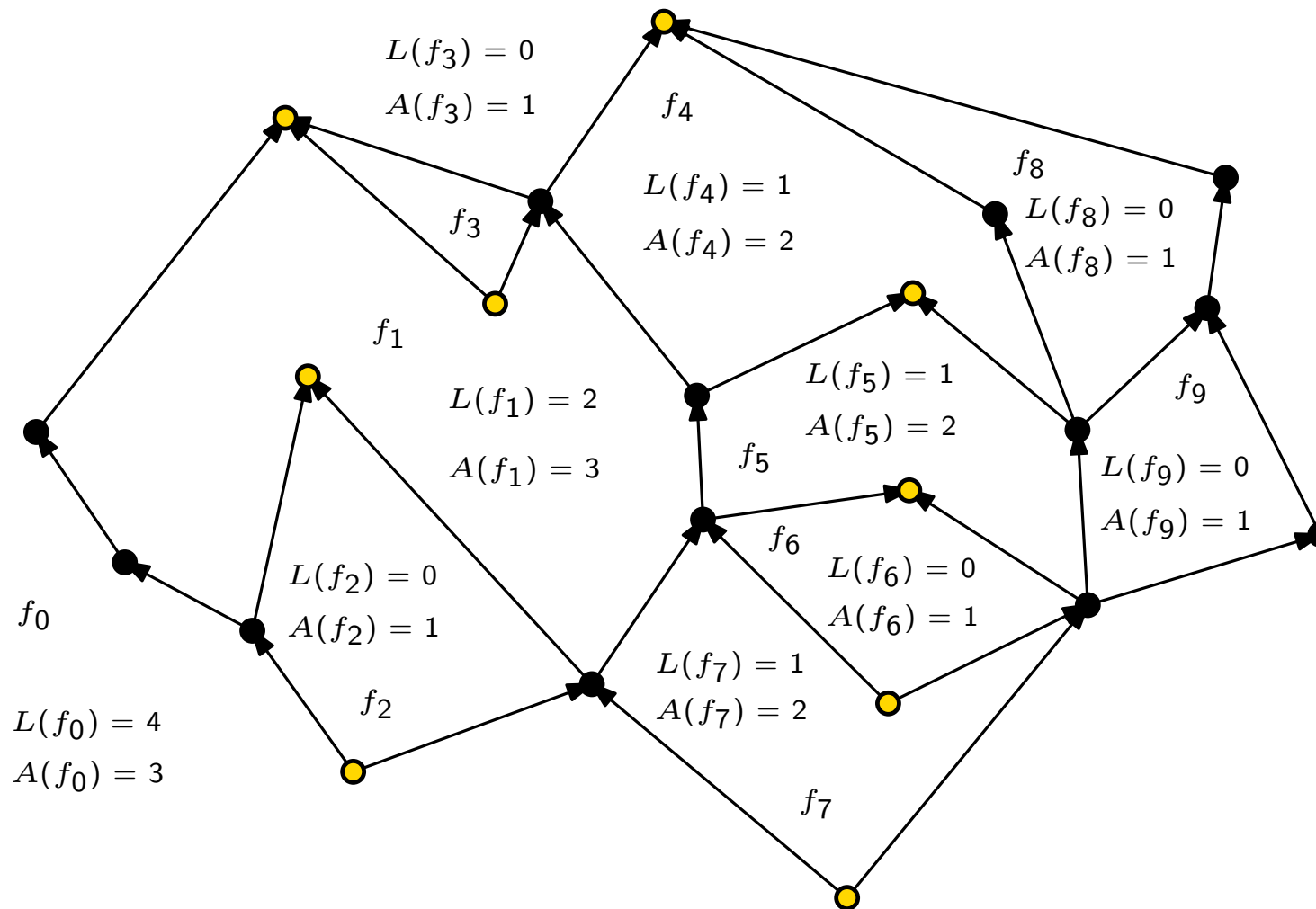
Definiert Zuordnung von Quellen/Senken zu inzidenten Facetten

$$\left. \begin{array}{l} \Phi : \{Q, S\} \rightarrow \mathcal{F} \\ \Phi : v \mapsto \text{inzidente Facette} \\ \text{heißt } \textit{konsistent}, \text{ wenn gilt} \end{array} \right\} |\Phi^{-1}(f)| = \begin{cases} A(f) - 1 & f \neq f_0 \\ A(f) + 1 & f_0 \end{cases}$$

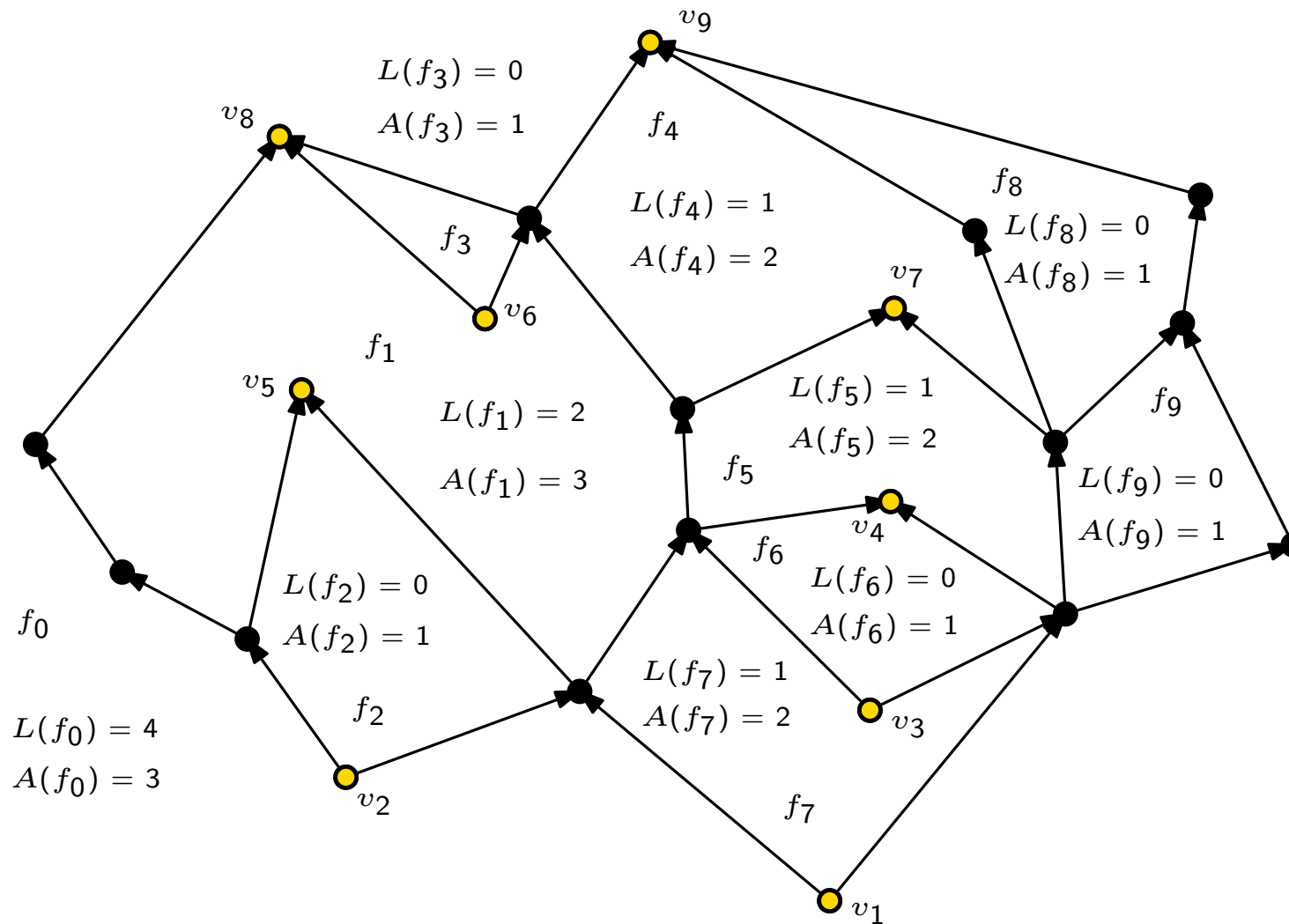
Beispiel Facettenzuordnung



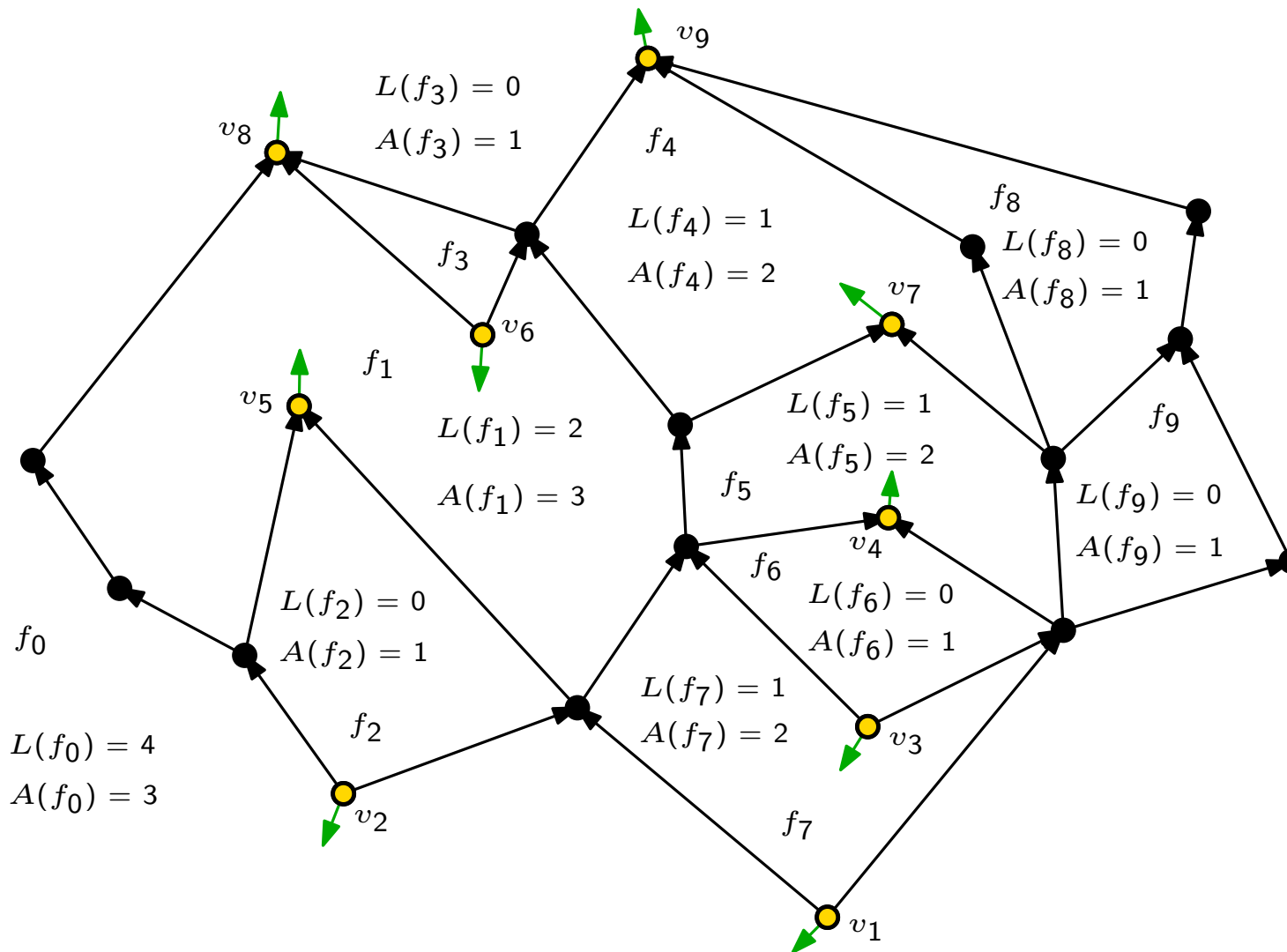
Beispiel Facettenzuordnung



Beispiel Facettenzuordnung



Beispiel Facettenzuordnung



- $\Phi(v_1) = f_0$
- $\Phi(v_2) = f_0$
- $\Phi(v_3) = f_7$
- $\Phi(v_4) = f_5$
- $\Phi(v_5) = f_1$
- $\Phi(v_6) = f_1$
- $\Phi(v_7) = f_4$
- $\Phi(v_8) = f_0$
- $\Phi(v_9) = f_0$

Satz

Für einen gerichteten azyklischen Graphen $D = (V, A)$ mit kombinatorischer Einbettung \mathcal{F}, f_0 gilt:

aufwärtsplanar \iff bimodal und \exists konsistentes Φ

Satz

Für einen gerichteten azyklischen Graphen $D = (V, A)$ mit kombinatorischer Einbettung \mathcal{F}, f_0 gilt:

aufwärtsplanar \iff bimodal und \exists konsistentes Φ

\Rightarrow : soeben hergeleitet

Satz

Für einen gerichteten azyklischen Graphen $D = (V, A)$ mit kombinatorischer Einbettung \mathcal{F}, f_0 gilt:

aufwärtsplanar \iff bimodal und \exists konsistentes Φ

\Rightarrow : soeben hergeleitet

\Leftarrow : Umkehrung gilt, ist sogar konstruktiv

Zunächst: $D, \mathcal{F}, f_0 \overset{?}{\rightsquigarrow} \Phi$ konsistent

Satz

Für einen gerichteten azyklischen Graphen $D = (V, A)$ mit kombinatorischer Einbettung \mathcal{F}, f_0 gilt:

aufwärtsplanar \iff bimodal und \exists konsistentes Φ

\Rightarrow : soeben hergeleitet

\Leftarrow : Umkehrung gilt, ist sogar konstruktiv

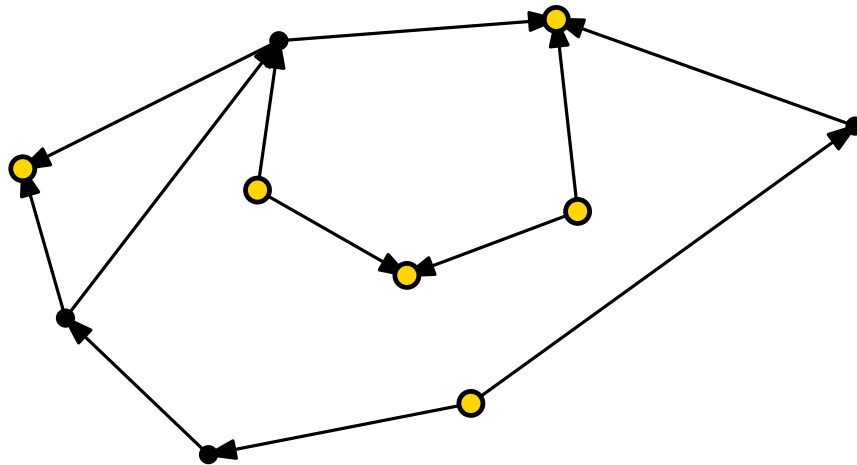
Zunächst: $D, \mathcal{F}, f_0 \stackrel{?}{\rightsquigarrow} \Phi$ konsistent

Flußnetzwerk!

Definition Flussnetzwerk $N_{\mathcal{F}, f_0}(D) = ((W, A_N); \ell; u; b)$

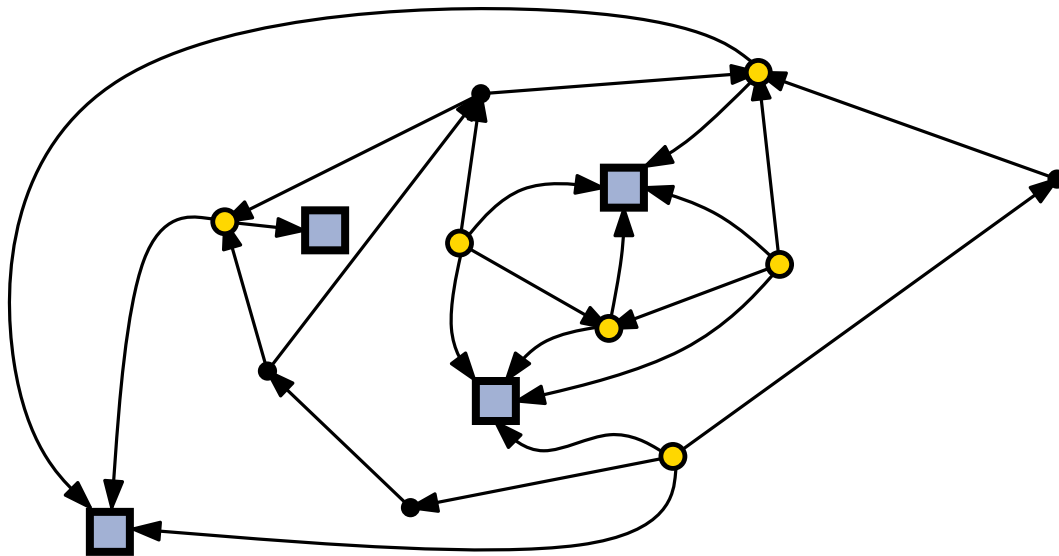
- $W = \{v \in V \mid v \text{ ist Quelle oder Senke}\} \cup \mathcal{F}$
- $A_N = \{(v, f) \mid v \text{ inzident zu } f\}$
- $l(a) = 0 \quad \forall a \in A_N$
- $u(a) = 1 \quad \forall a \in A_N$
- $b(q) = \begin{cases} 1 & \forall q \in W \cap V \\ -(A(q) - 1) & \forall q \in \mathcal{F} \setminus \{f_0\} \\ -(A(q) + 1) & q = f_0 \end{cases}$

Beispielnetzwerk



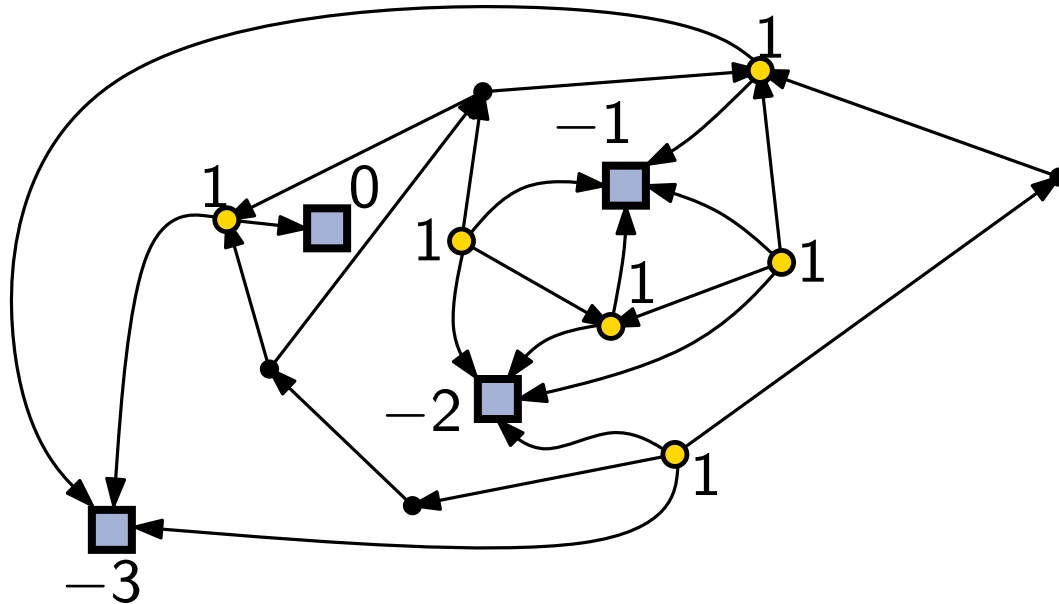
- normaler Knoten
- Quelle / Senke

Beispielnetzwerk



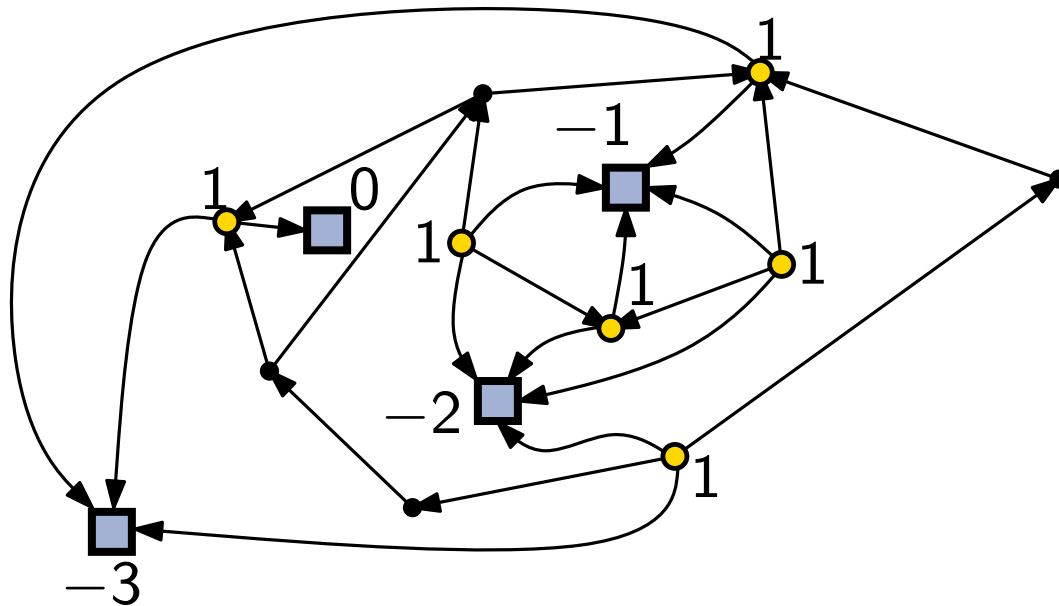
- normaler Knoten
- Quelle / Senke
- Facettenknoten

Beispielnetzwerk



- normaler Knoten
- Quelle / Senke
- Facettenknoten

Beispielnetzwerk

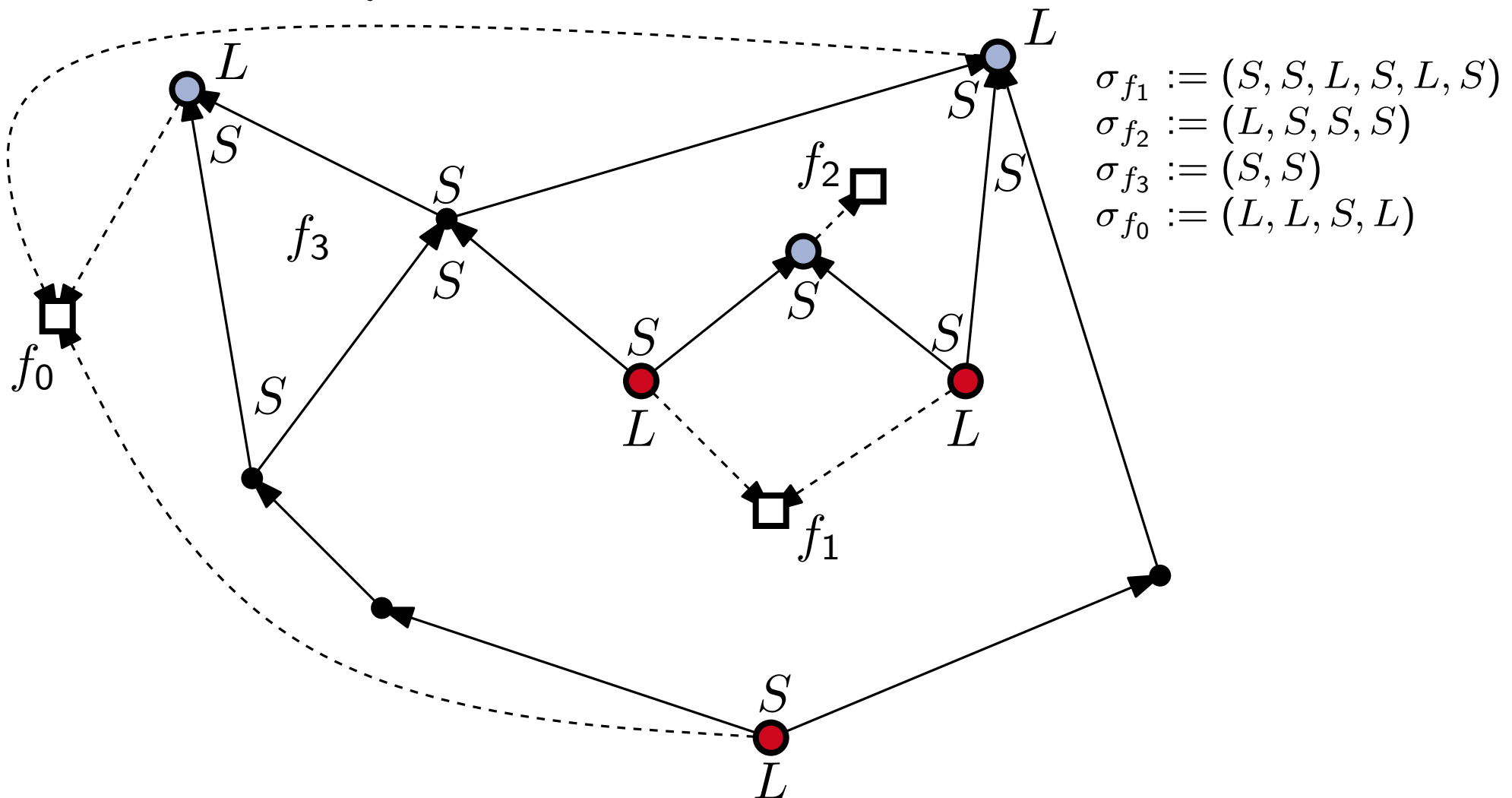


- normaler Knoten
- Quelle / Senke
- Facettenknoten

- Starte mit Nullfluss
- Suche erhöhende Wege
- Geht auch ohne festgelegtes f_0

Winkelfolgen an Facetten

- Betrachte Folge σ_f von Winkeln L, S an lokalen Quellen und Senken von f

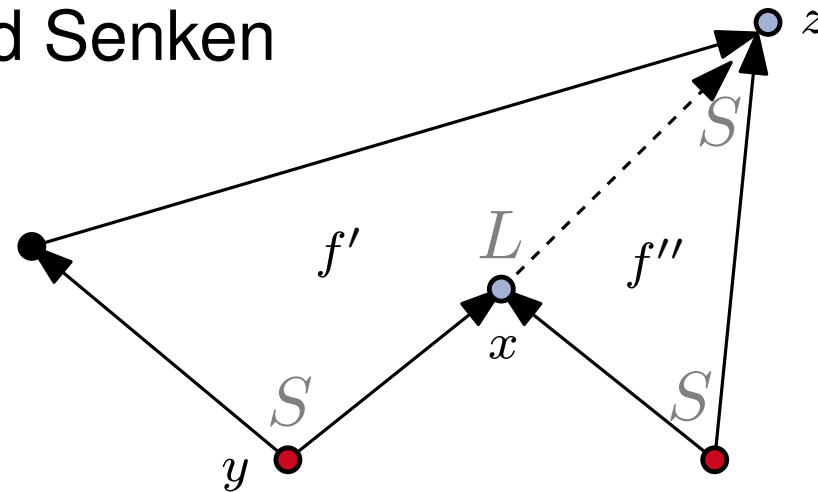
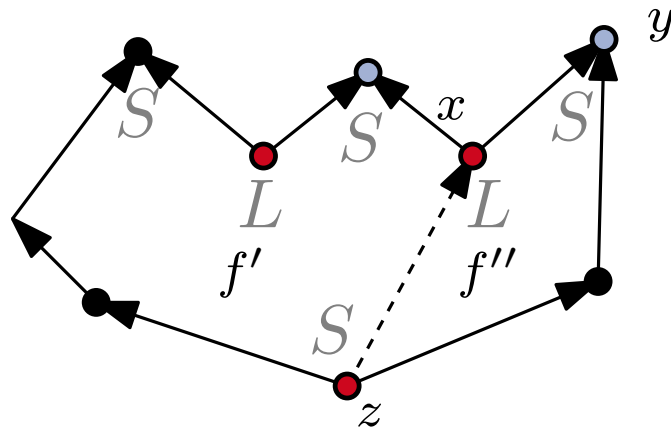


Algorithmus: $\Phi, \mathcal{F}, f_0 \rightsquigarrow$ s-t-Graph $\supseteq D$

- $f \neq f_0$ mit $|\sigma_f| \geq 2$ enthält L, S, S an x, y, z

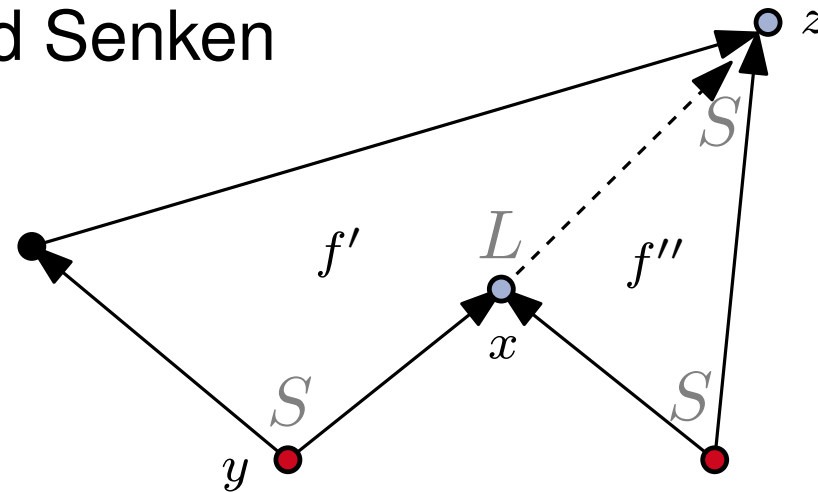
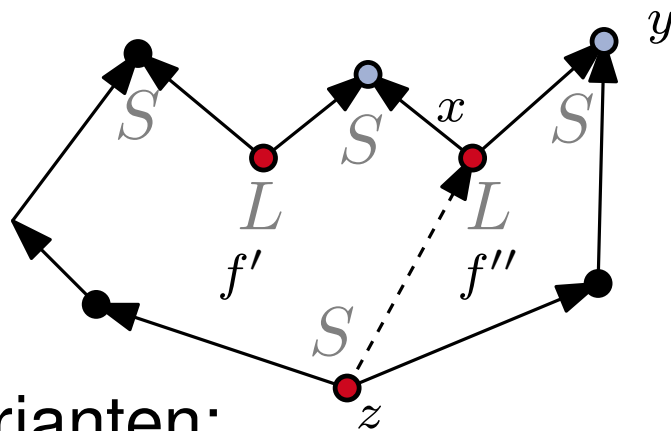
Algorithmus: $\Phi, \mathcal{F}, f_0 \rightsquigarrow$ s-t-Graph $\cong D$

- $f \neq f_0$ mit $|\sigma_f| \geq 2$ enthält L, S, S an x, y, z
- x Quelle \Rightarrow verfeinere mit (z, x)
- x Senke \Rightarrow verfeinere mit (x, z)
- Ziel: Entferne alle Quellen und Senken



Algorithmus: $\Phi, \mathcal{F}, f_0 \rightsquigarrow s$ - t -Graph $\cong D$

- $f \neq f_0$ mit $|\sigma_f| \geq 2$ enthält L, S, S an x, y, z
- x Quelle \Rightarrow verfeinere mit (z, x)
- x Senke \Rightarrow verfeinere mit (x, z)
- Ziel: Entferne alle Quellen und Senken



Invarianten:

- Planarität, Azyklizität, Bimodalität
- Quellen/Senken von f werden Quellen/Senken von f'
- Füge zwischen irgendeiner Quelle s und Senke t auf f_0 die Kante (s, t) ein.
 \Rightarrow planarer s - t -Graph, der D enthält