Algorithmen zur Visualisierung von Graphen

Kombinatorische Optimierung mittels Flussmethoden II

Vorlesung im Wintersemester 2011/2012

Ignaz Rutter

10.11.2011

Institute for Theoretical Informatics
Algorithmics Group I

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.

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

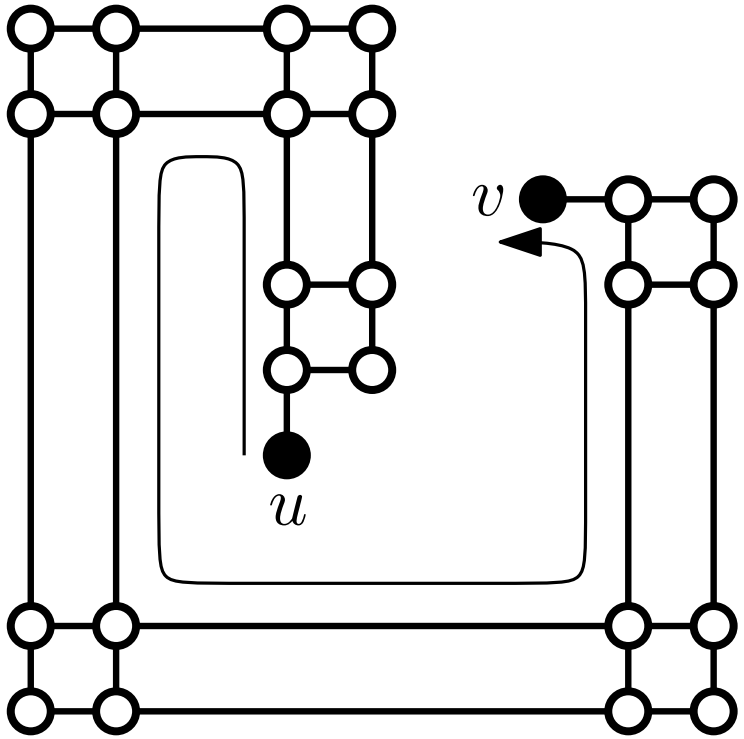
heute:

Satz

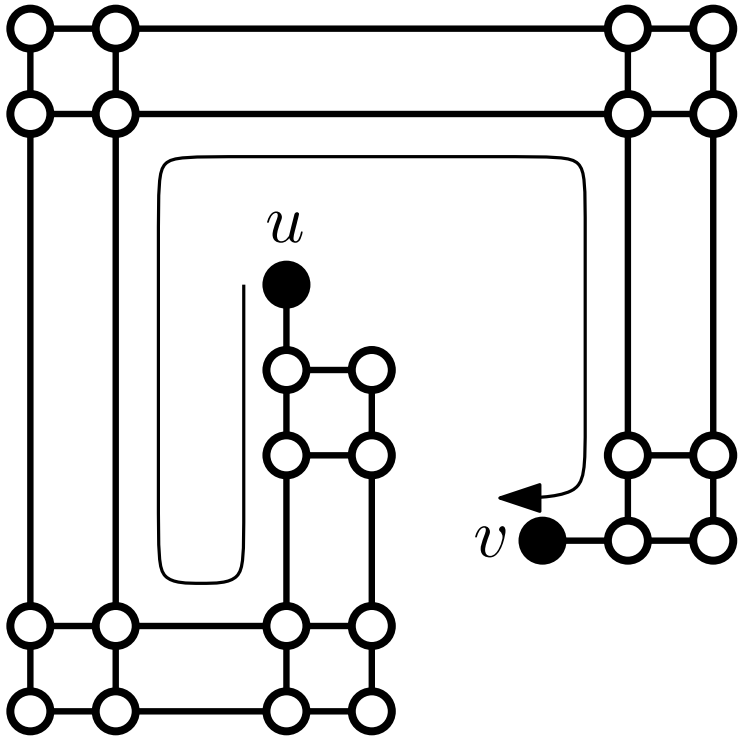
G Maxgrad-4-Graph mit variabler Einbettung
Entscheiden, ob G orthogonale Zeichnung ohne Knicke besitzt ist NP-schwer.

Schwierigkeit bei 0-Knick-Zeichenbarkeit

Es gibt starre Konstruktionen



4–6 Linksknicke

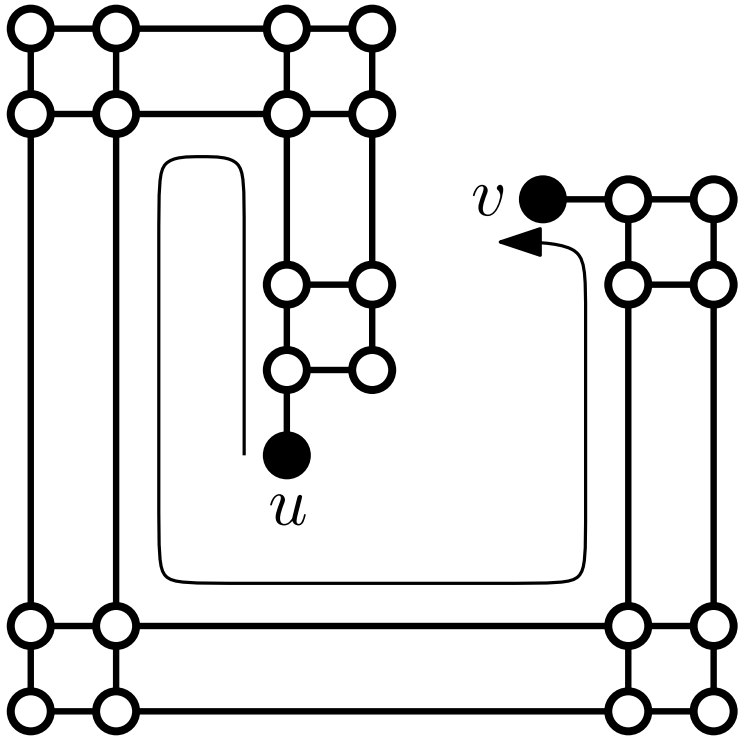


4–6 Rechtsknicke

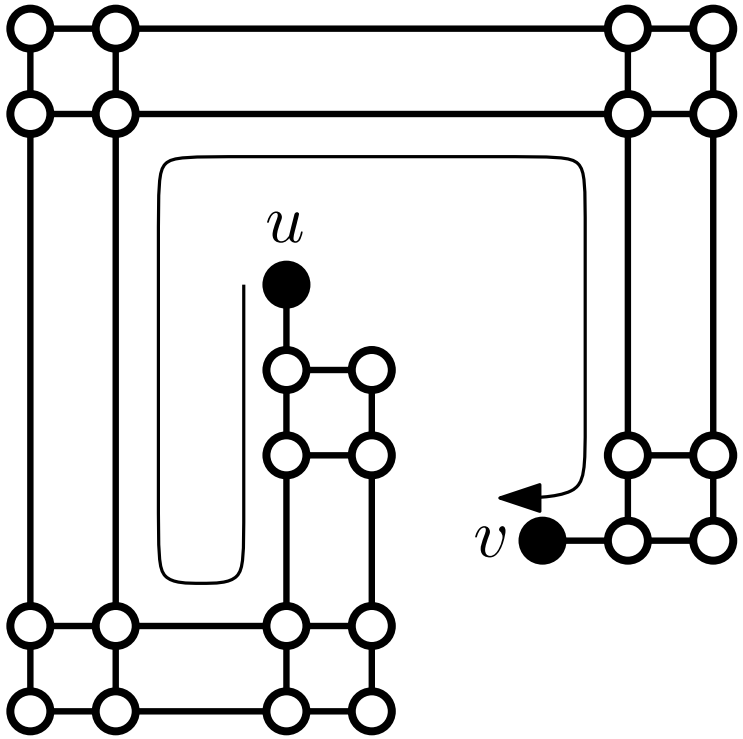
Schwierigkeit bei 0-Knick-Zeichenbarkeit

Es gibt starre Konstruktionen

T_1

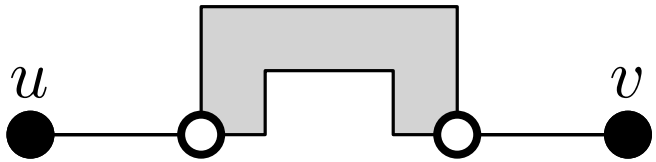


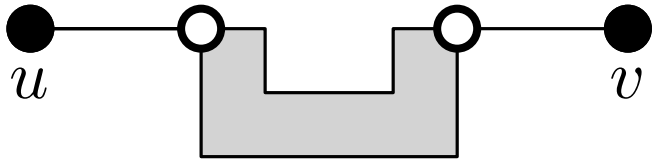
4–6 Linksknicke



4–6 Rechtsknicke

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



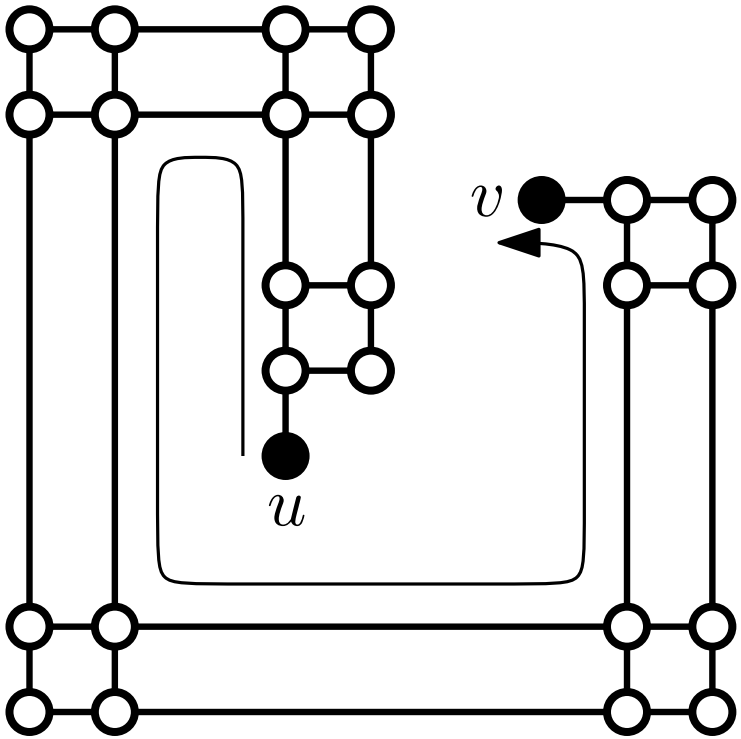


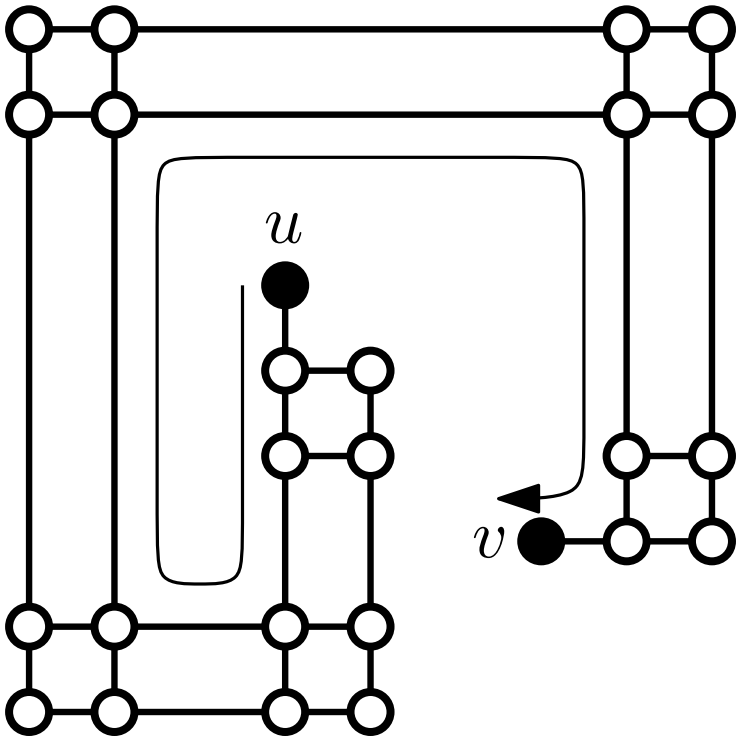
Zwei Gadgets

Tendrils

T_k

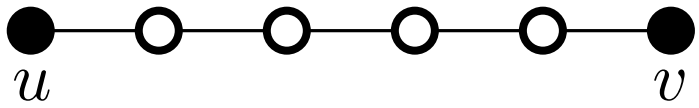
$4k + 1$ Knicke





Wiggles

W_k



Pfad der Länge $4k + 1$

0 bis $4k$ Knicke

Ein neues Flußproblem

Problem (Switch-Flow Network)

Eingabe:

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

Gültiger Fluß:

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

Existiert ein gültiger Fluß?

Ein neues Flußproblem

Problem (Switch-Flow Network)

Eingabe:

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

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

- $x_1 = \texttt{true}, x_2 = \texttt{false}, x_3 = \texttt{true}$ ist erfüllend
- $x_1 = \texttt{true}, x_2 = \texttt{true}, x_3 = \texttt{true}$ 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: $\Phi = (\neg x_1, x_2, \neg x_3), (\neg x_1, \neg x_2, x_3), (x_1, x_2, x_3)$

- $x_1 = \texttt{true}, x_2 = \texttt{false}, x_3 = \texttt{true}$ ist erfüllend
- $x_1 = \texttt{true}, x_2 = \texttt{true}, x_3 = \texttt{true}$ 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

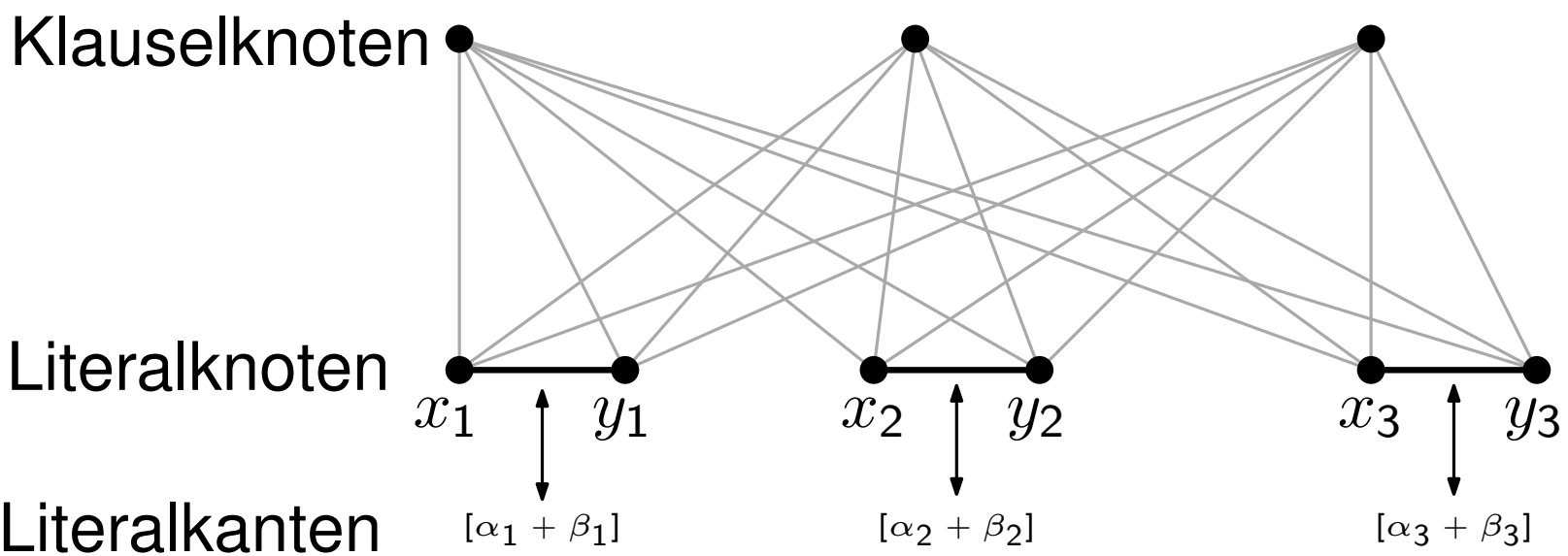
Φ Instanz von Not-All-Equal 3SAT
Literale: $x_1, y_1, \ldots, x_n, y_n$ mit $y_i = \neg x_i$
Klauseln: $c_1, \ldots, c_m$

Reduktion von Not-All-Equal 3SAT

Φ Instanz von Not-All-Equal 3SAT
Literale: $x_1, y_1, \ldots, x_n, y_n$ mit $y_i = \neg x_i$
Klauseln: $c_1, \ldots, c_m$

$\alpha_i :=$ #Vorkommen von x_i in Klauseln von Φ
$\beta_i :=$ #Vorkommen von y_i in Klauseln von Φ
Beachte: $\sum_{i=1}^{n}(\alpha_i + \beta_i) = 3m$

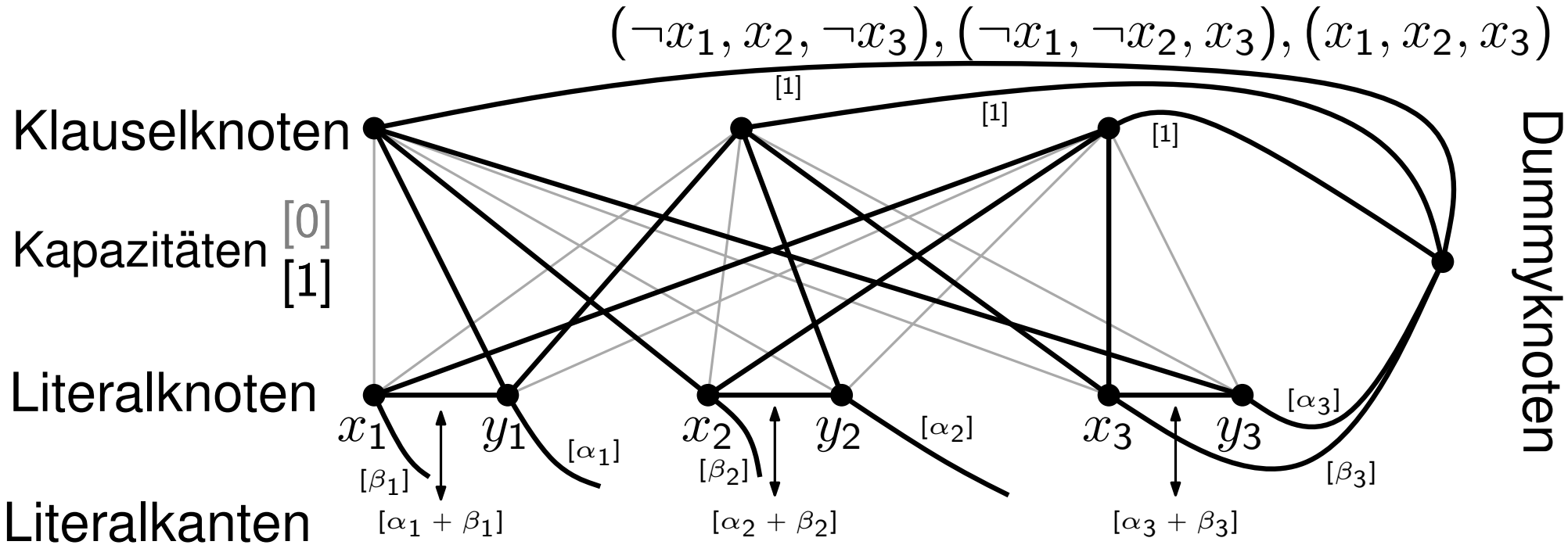
$$(\neg x_1, x_2, \neg x_3), (\neg x_1, \neg x_2, x_3), (x_1, x_2, x_3)$$



Reduktion von Not-All-Equal 3SAT

Φ Instanz von Not-All-Equal 3SAT
Literale: $x_1, y_1, \ldots, x_n, y_n$ mit $y_i = \neg x_i$
Klauseln: $c_1, \ldots, c_m$

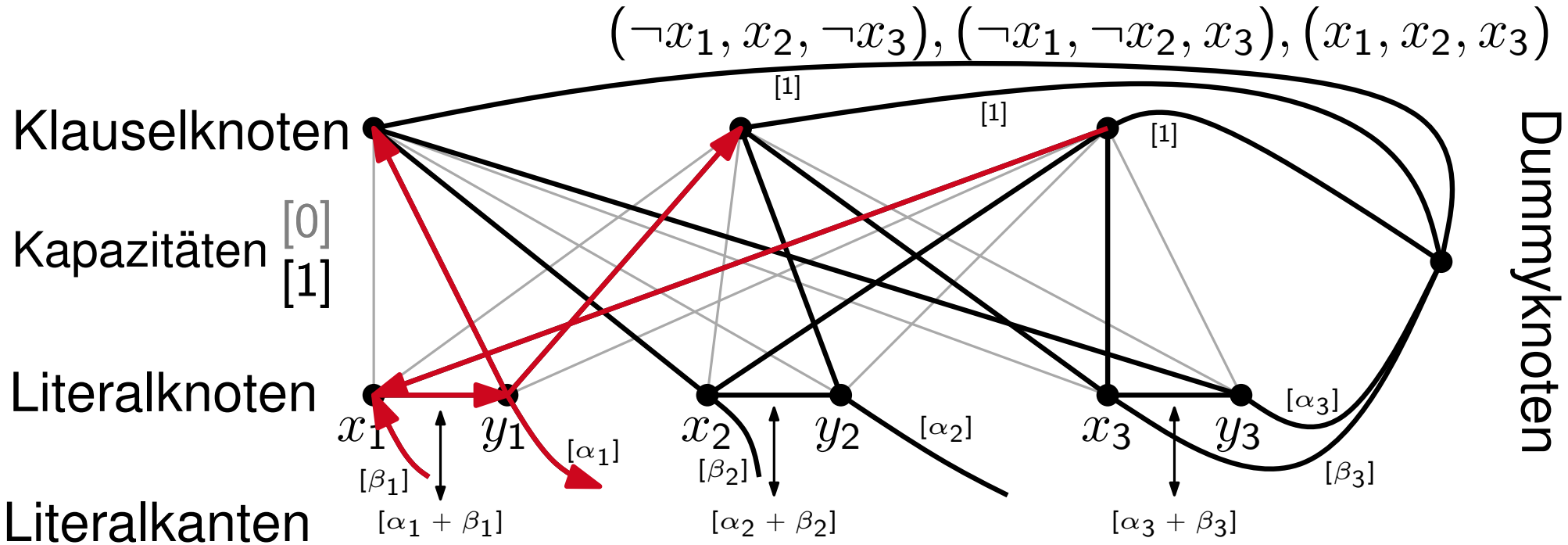
$\alpha_i :=$ #Vorkommen von x_i in Klauseln von Φ
$\beta_i :=$ #Vorkommen von y_i in Klauseln von Φ
Beachte: $\sum_{i=1}^{n}(\alpha_i + \beta_i) = 3m$



Reduktion von Not-All-Equal 3SAT

Φ Instanz von Not-All-Equal 3SAT
Literale: $x_1, y_1, \ldots, x_n, y_n$ mit $y_i = \neg x_i$
Klauseln: $c_1, \ldots, c_m$

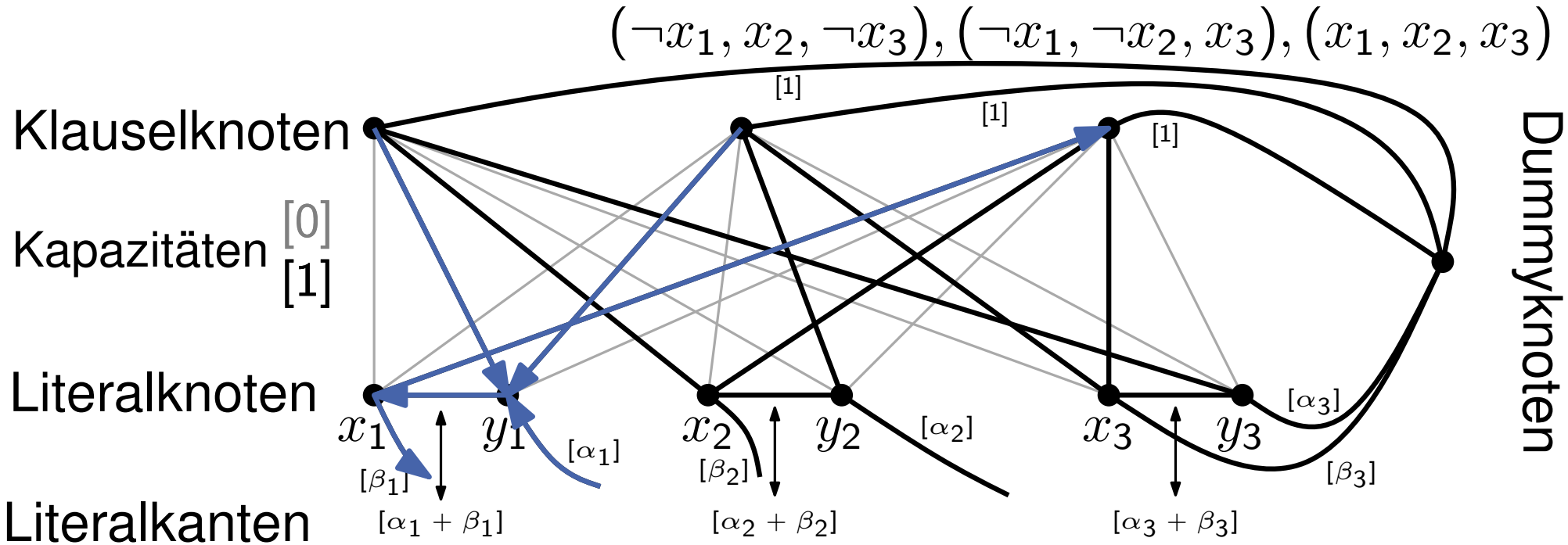
$\alpha_i :=$ #Vorkommen von x_i in Klauseln von Φ
$\beta_i :=$ #Vorkommen von y_i in Klauseln von Φ
Beachte: $\sum_{i=1}^{n}(\alpha_i + \beta_i) = 3m$



Reduktion von Not-All-Equal 3SAT

Φ Instanz von Not-All-Equal 3SAT
Literale: $x_1, y_1, \ldots, x_n, y_n$ mit $y_i = \neg x_i$
Klauseln: $c_1, \ldots, c_m$

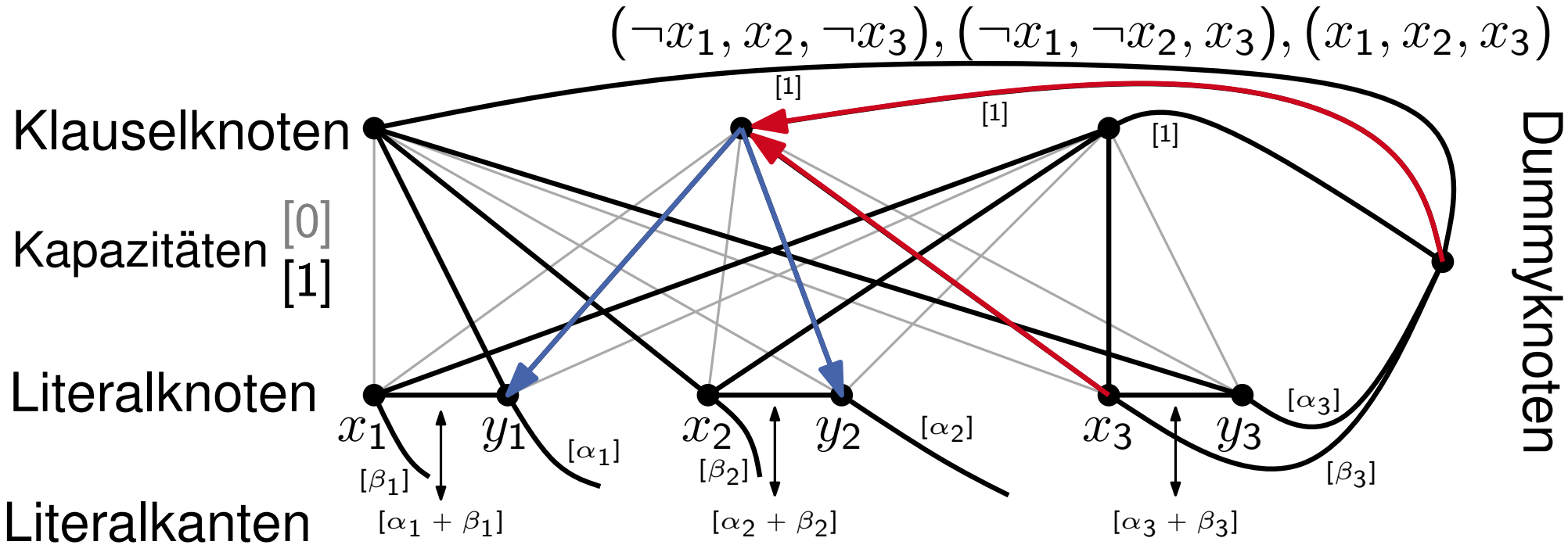
$\alpha_i :=$ #Vorkommen von x_i in Klauseln von Φ
$\beta_i :=$ #Vorkommen von y_i in Klauseln von Φ
Beachte: $\sum_{i=1}^{n}(\alpha_i + \beta_i) = 3m$



Reduktion von Not-All-Equal 3SAT

Φ Instanz von Not-All-Equal 3SAT
Literale: $x_1, y_1, \ldots, x_n, y_n$ mit $y_i = \neg x_i$
Klauseln: $c_1, \ldots, c_m$

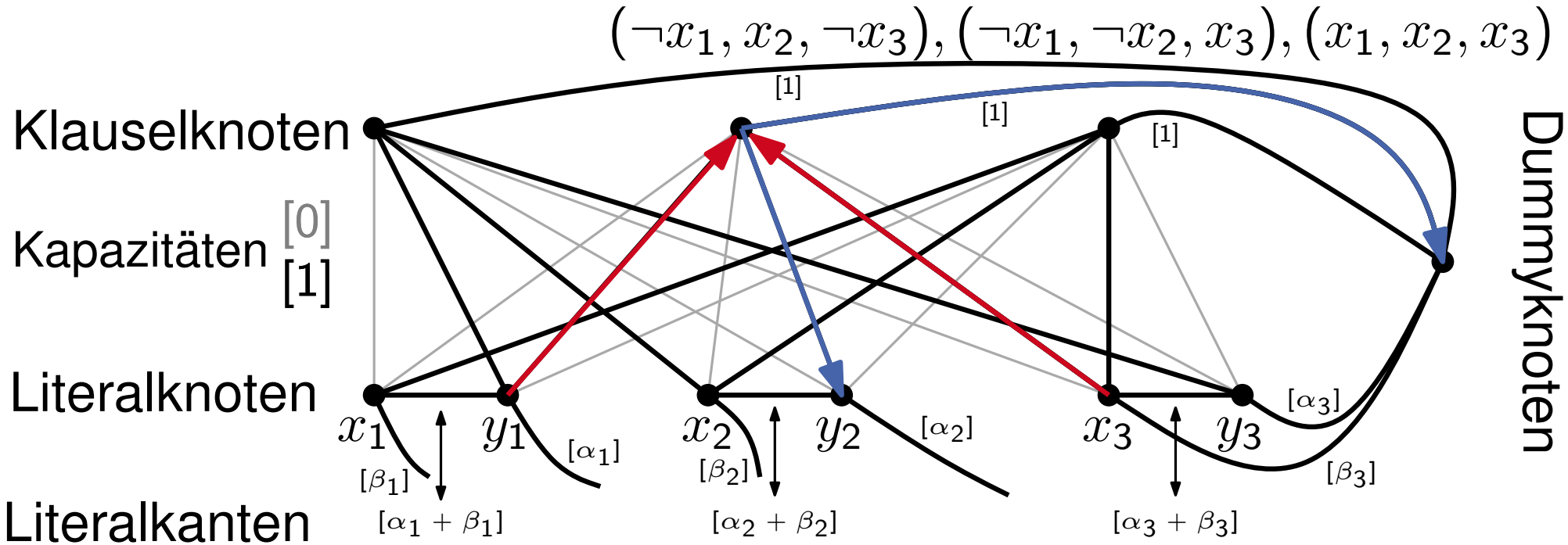
$\alpha_i :=$ #Vorkommen von x_i in Klauseln von Φ
$\beta_i :=$ #Vorkommen von y_i in Klauseln von Φ
Beachte: $\sum_{i=1}^{n}(\alpha_i + \beta_i) = 3m$



Reduktion von Not-All-Equal 3SAT

Φ Instanz von Not-All-Equal 3SAT
Literale: $x_1, y_1, \ldots, x_n, y_n$ mit $y_i = \neg x_i$
Klauseln: $c_1, \ldots, c_m$

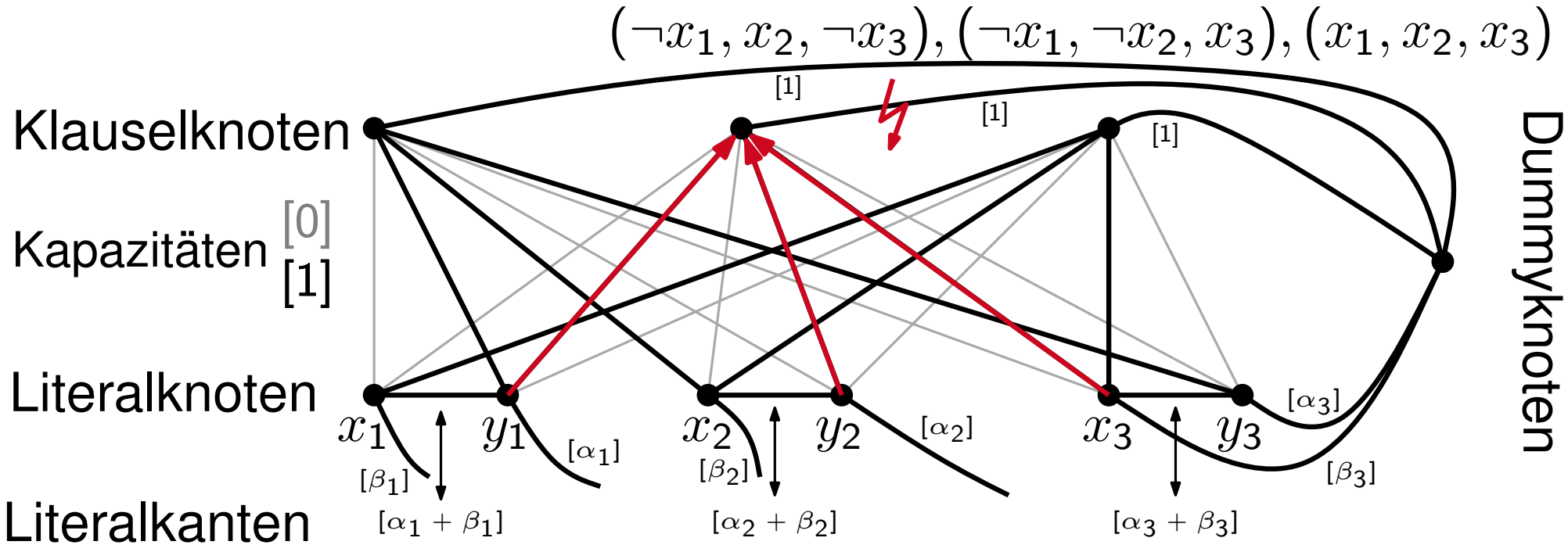
$\alpha_i :=$ #Vorkommen von x_i in Klauseln von Φ
$\beta_i :=$ #Vorkommen von y_i in Klauseln von Φ
Beachte: $\sum_{i=1}^{n}(\alpha_i + \beta_i) = 3m$



Reduktion von Not-All-Equal 3SAT

Φ Instanz von Not-All-Equal 3SAT
Literale: $x_1, y_1, \ldots, x_n, y_n$ mit $y_i = \neg x_i$
Klauseln: $c_1, \ldots, c_m$

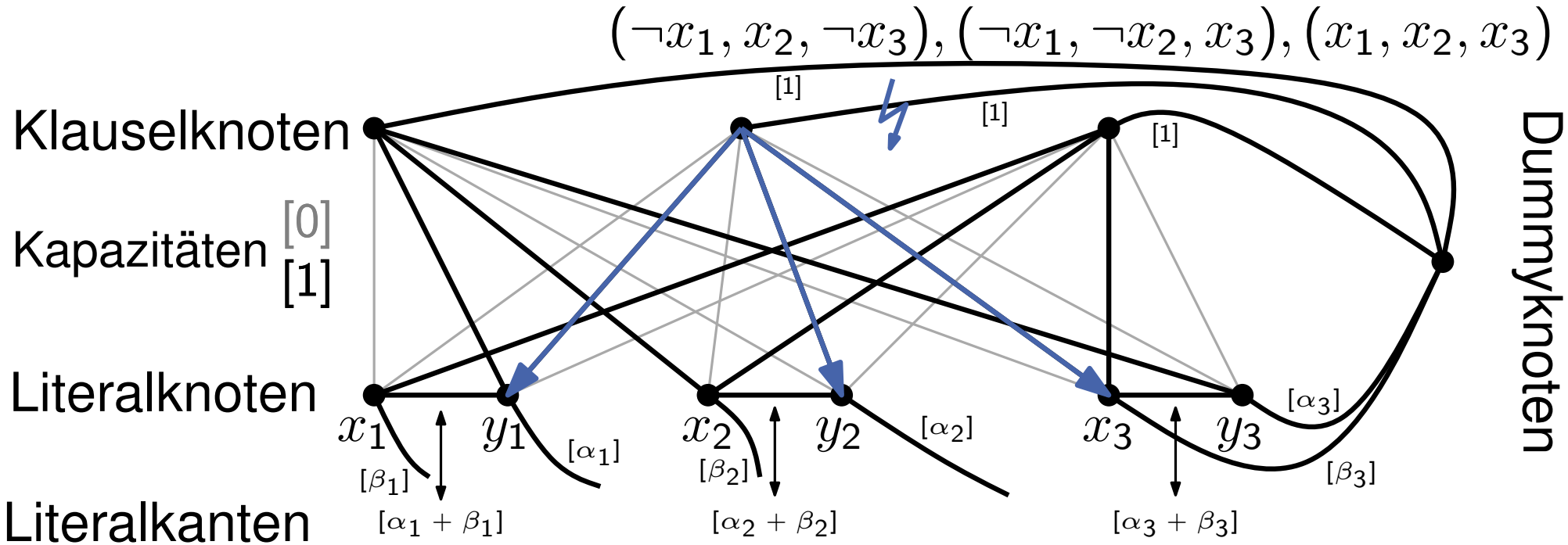
$\alpha_i :=$ #Vorkommen von x_i in Klauseln von Φ
$\beta_i :=$ #Vorkommen von y_i in Klauseln von Φ
Beachte: $\sum_{i=1}^{n}(\alpha_i + \beta_i) = 3m$



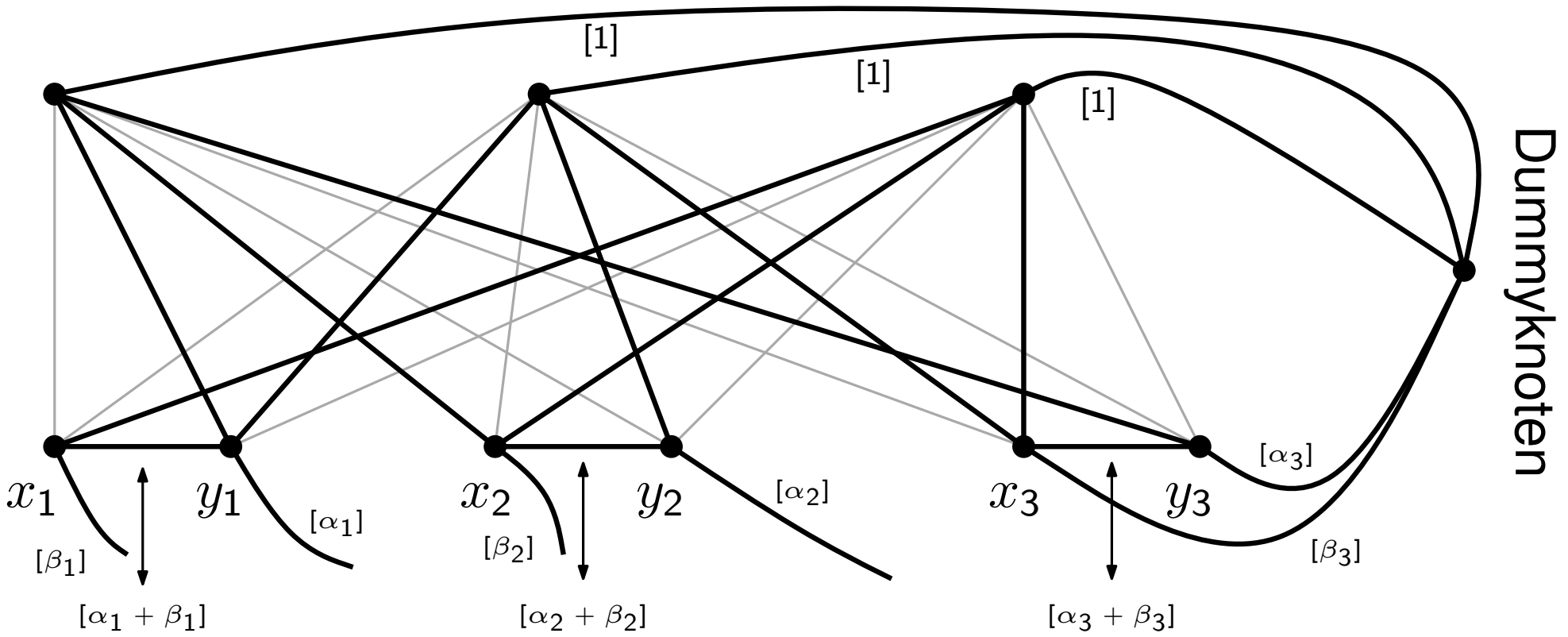
Reduktion von Not-All-Equal 3SAT

Φ Instanz von Not-All-Equal 3SAT
Literale: $x_1, y_1, \ldots, x_n, y_n$ mit $y_i = \neg x_i$
Klauseln: $c_1, \ldots, c_m$

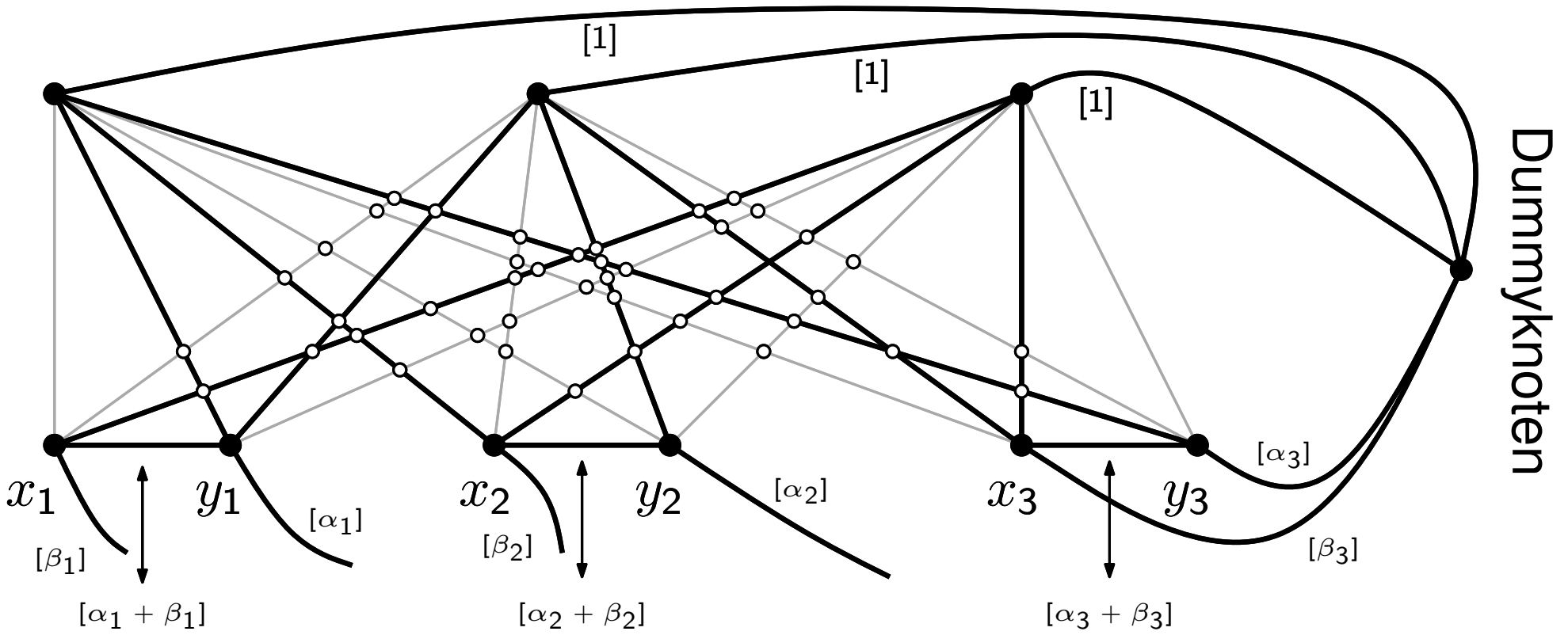
$\alpha_i :=$ #Vorkommen von x_i in Klauseln von Φ
$\beta_i :=$ #Vorkommen von y_i in Klauseln von Φ
Beachte: $\sum_{i=1}^{n}(\alpha_i + \beta_i) = 3m$



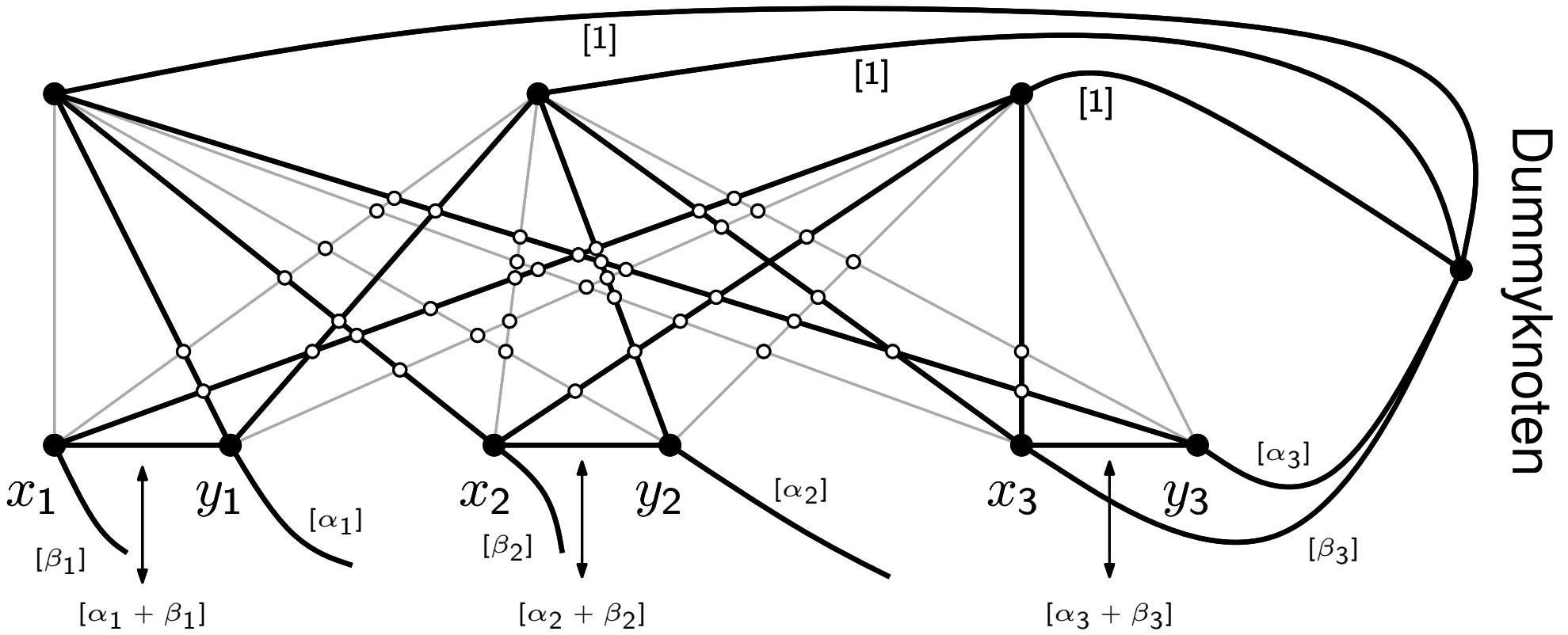
Herstellung von Planarität



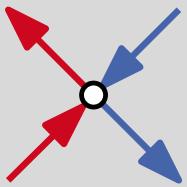
Herstellung von Planarität



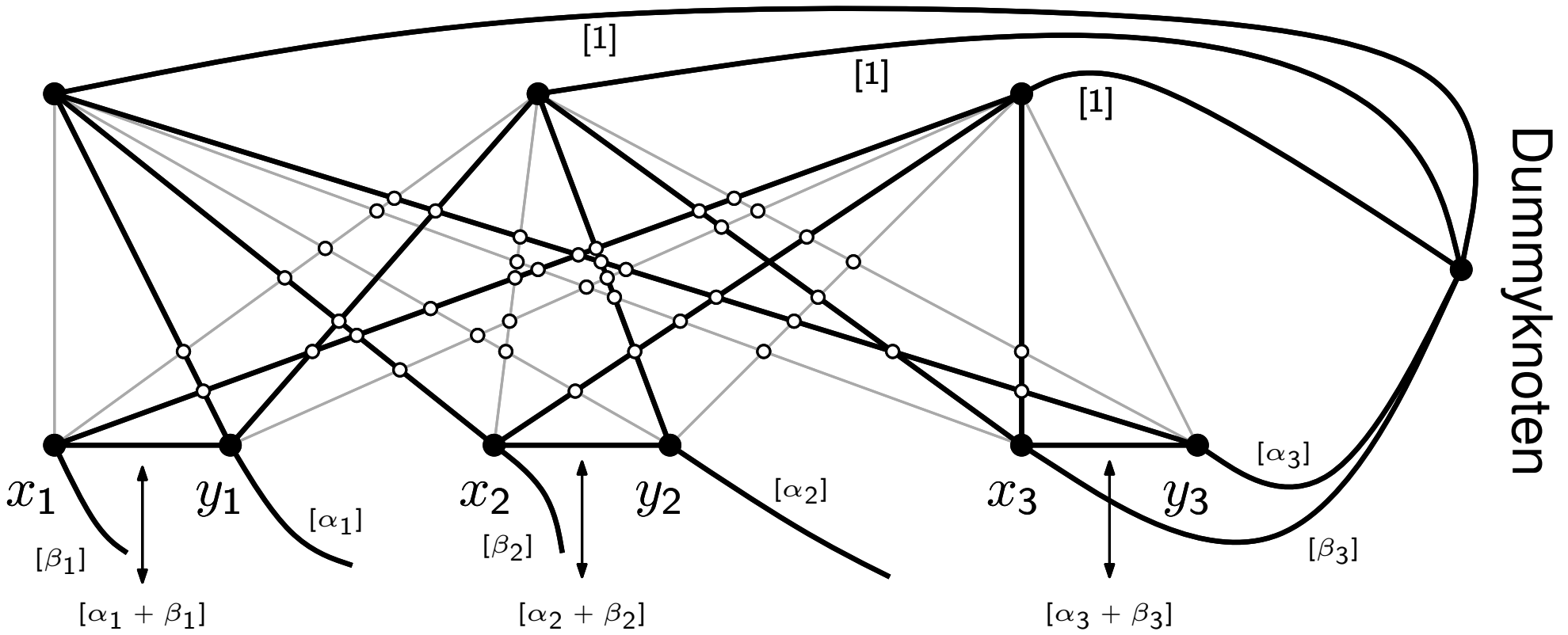
Herstellung von Planarität



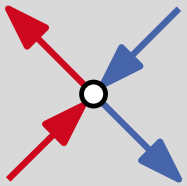
Problem:



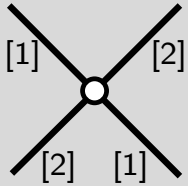
Herstellung von Planarität



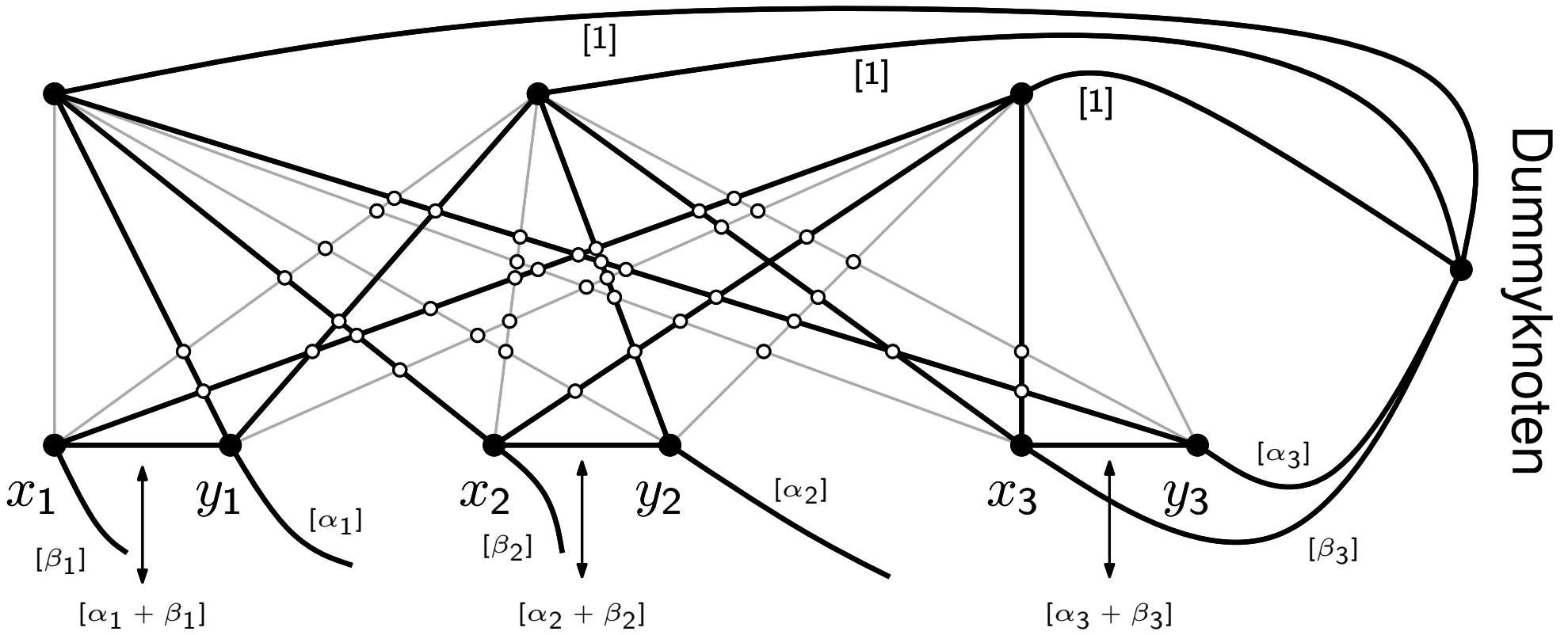
Problem:



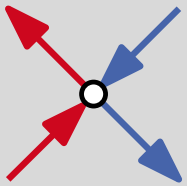
Lösungsidee:



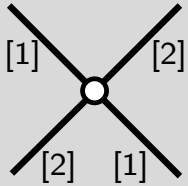
Herstellung von Planarität



Problem:



Lösungsidee:

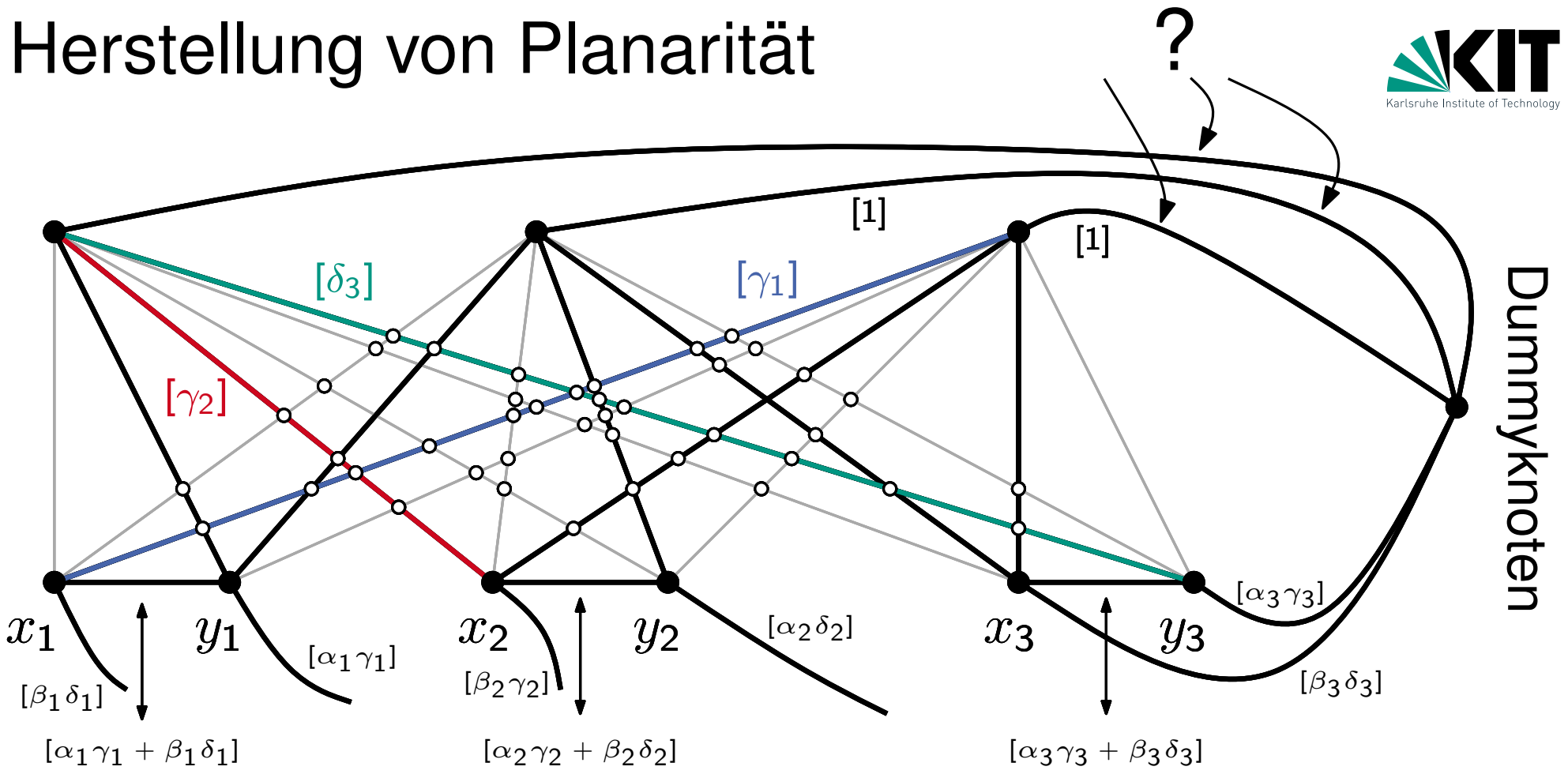


Allgemeiner: Verschiedene Flußwerte für verschiedene Literale

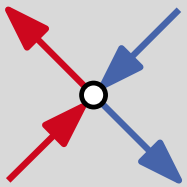
für $x_i : \gamma_i := (2i - 1)\Theta$

für $y_i : \delta_i := 2i\Theta$

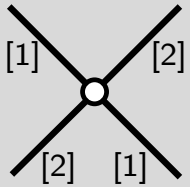
Herstellung von Planarität



Problem:



Lösungsidee:

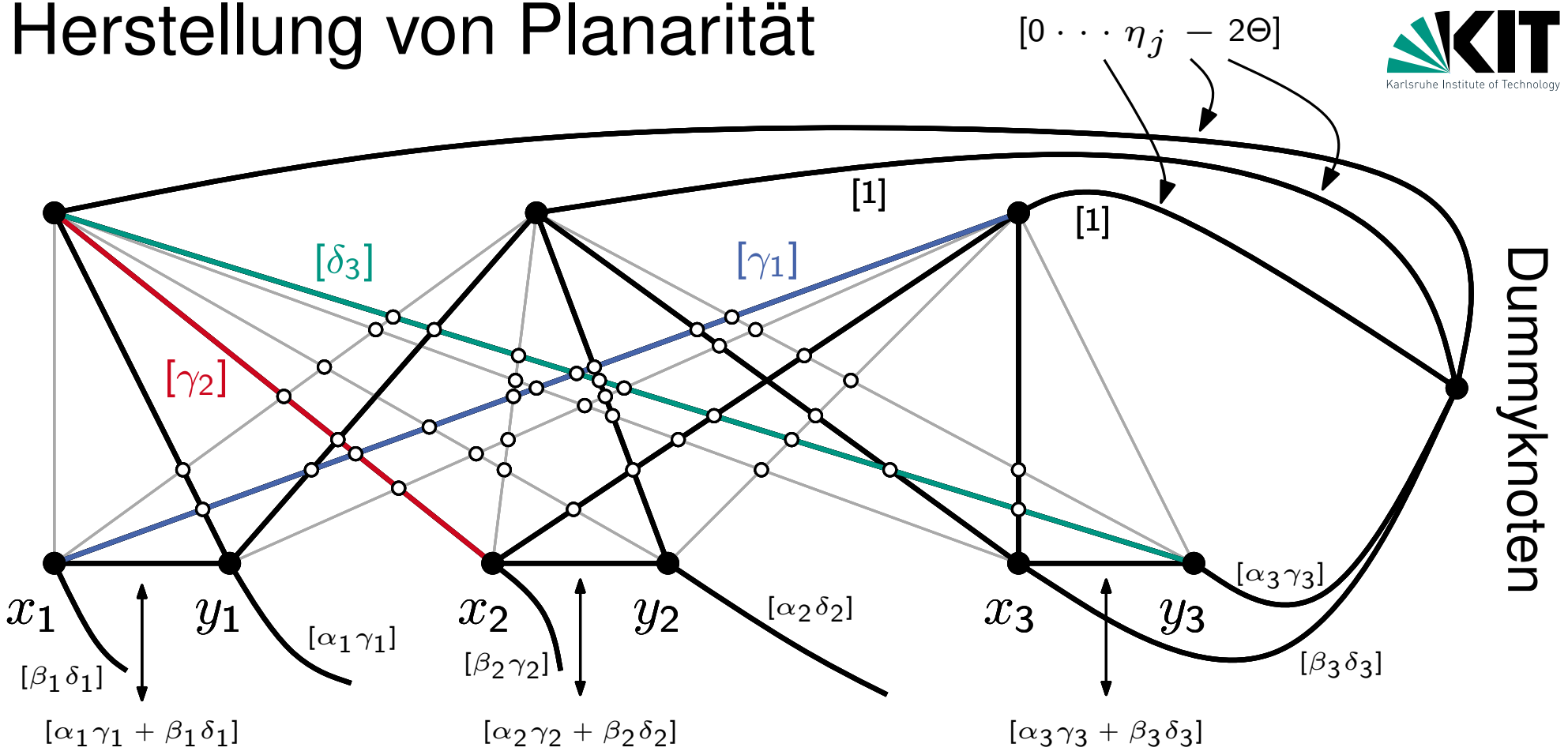


Allgemeiner: Verschiedene Flußwerte für verschiedene Literale

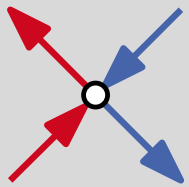
für $x_i : \gamma_i := (2i-1)\Theta$

für $y_i : \delta_i := 2i\Theta$

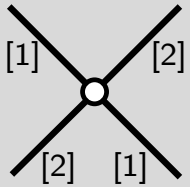
Herstellung von Planarität



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

Zwischenstand

Konstruierter Graph ist 3-fach zusammenhängend

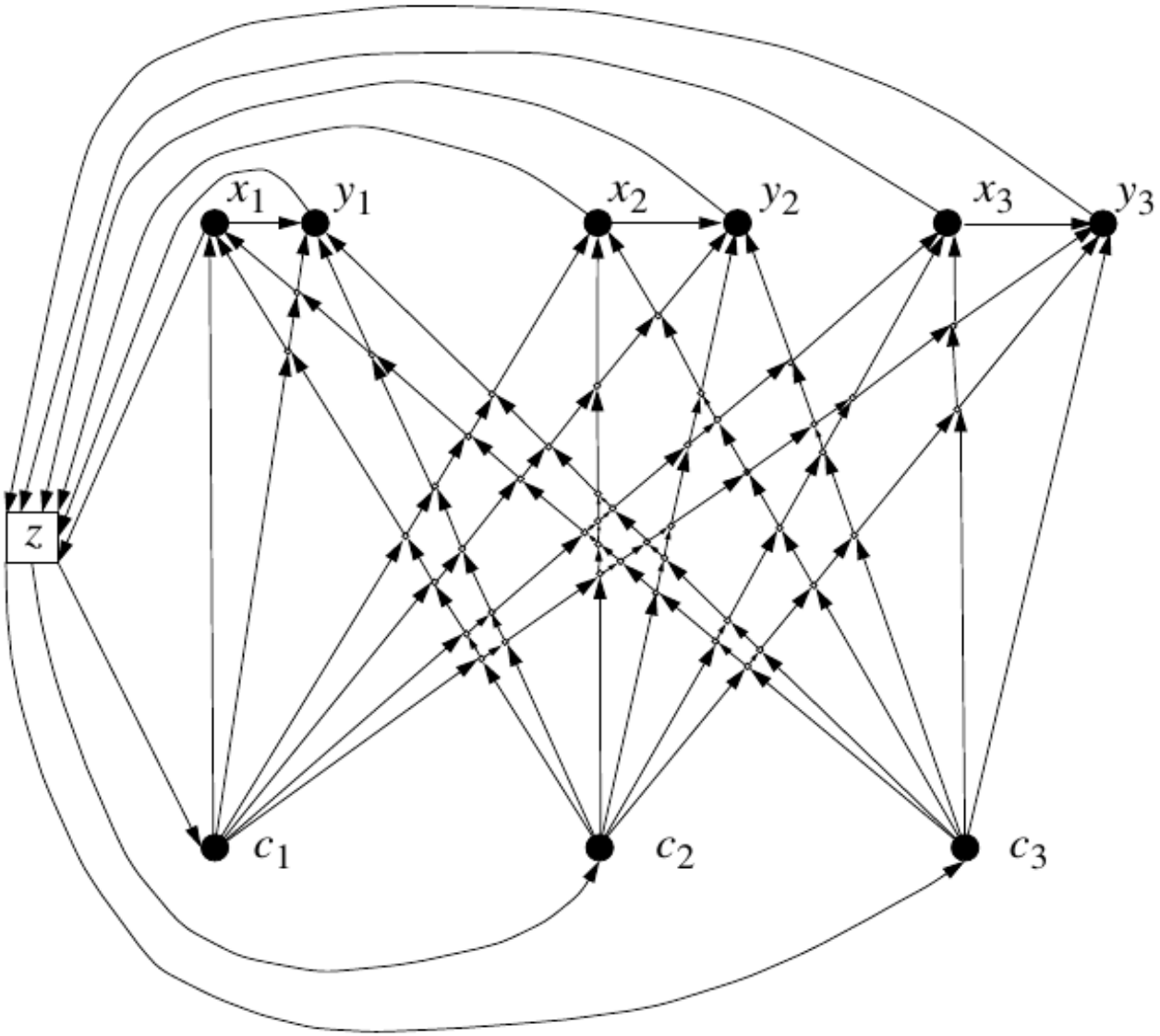
Satz

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

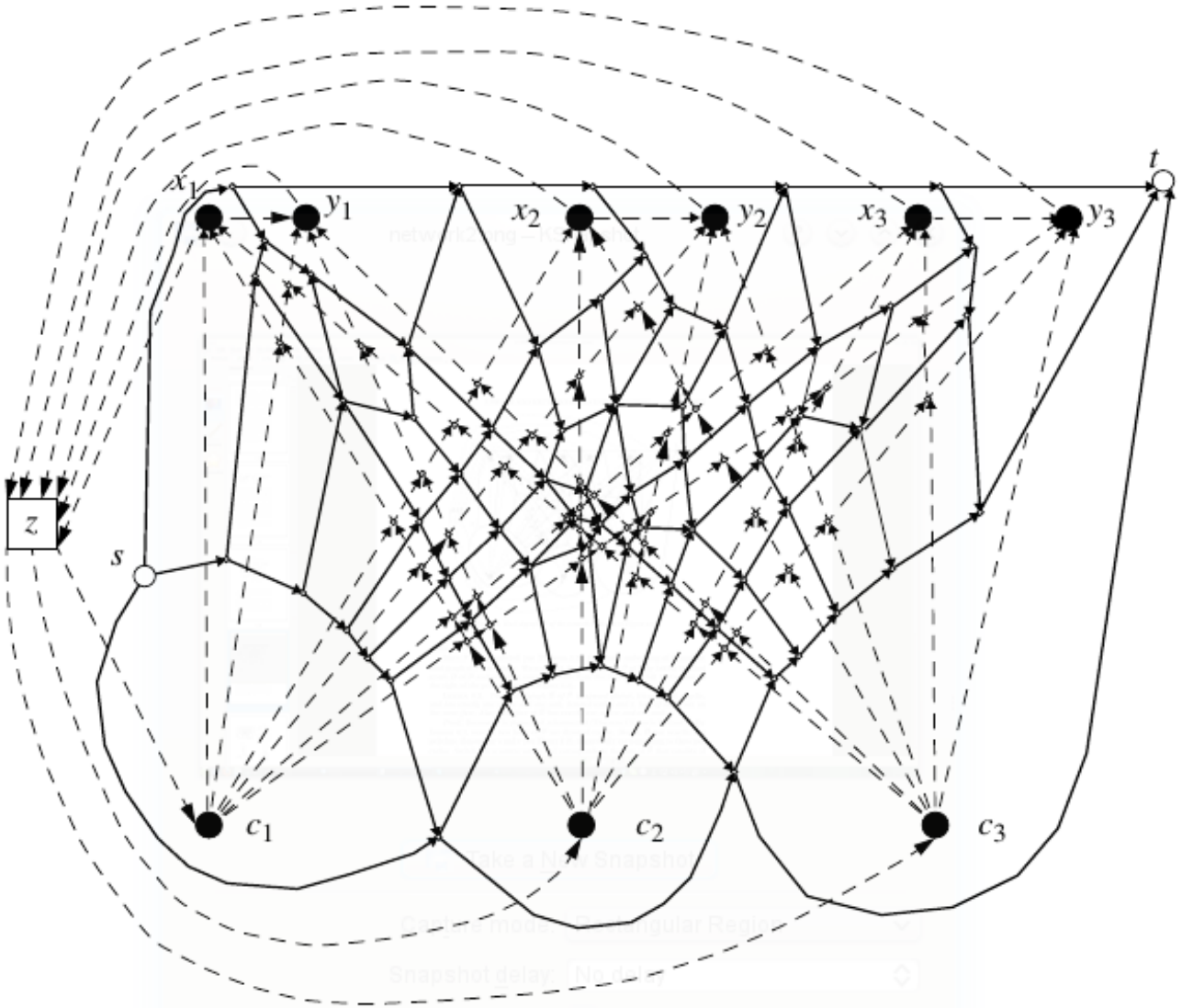
Idee: Konstruiere Graph G mit
G besitzt Zeichnung ohne Knicke $\Leftrightarrow$ N lösbar.

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

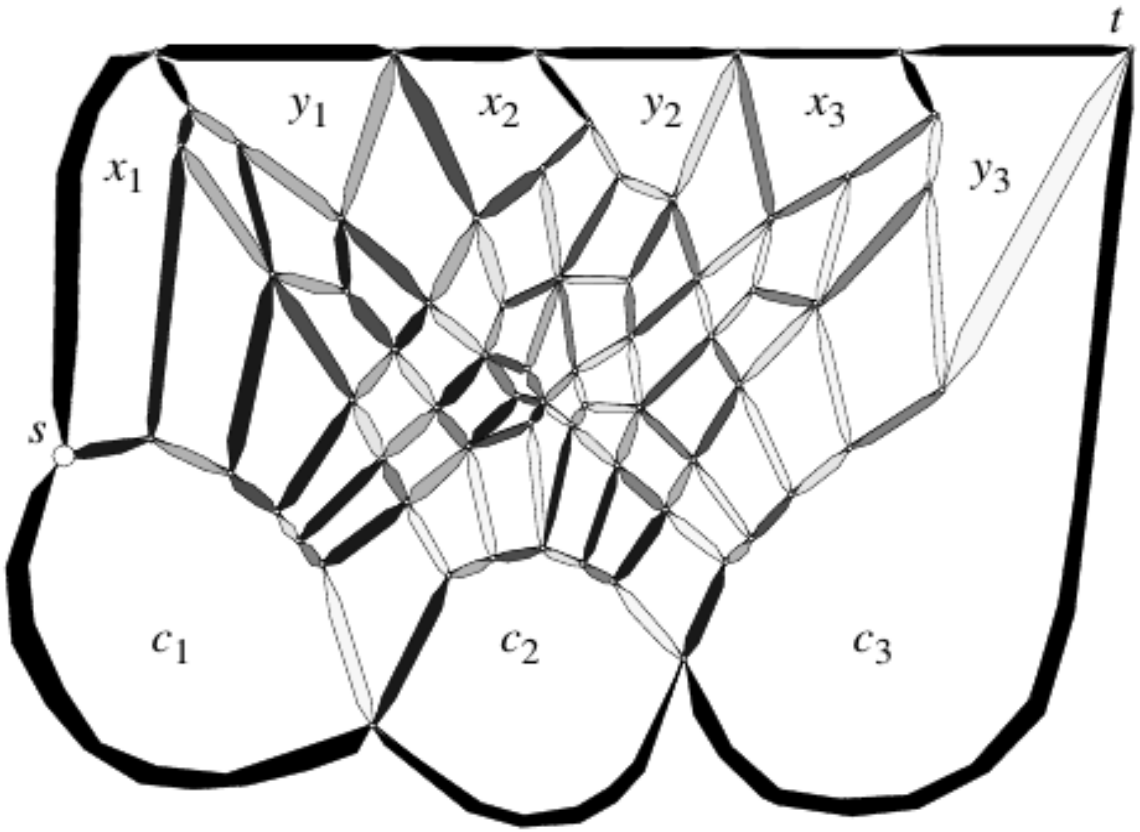
Konstruktion von G



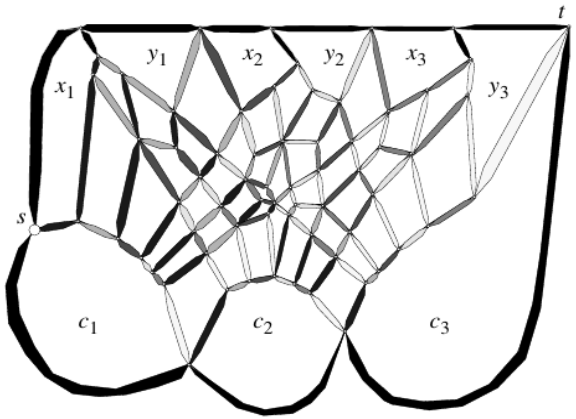
Konstruktion von G



Konstruktion von G



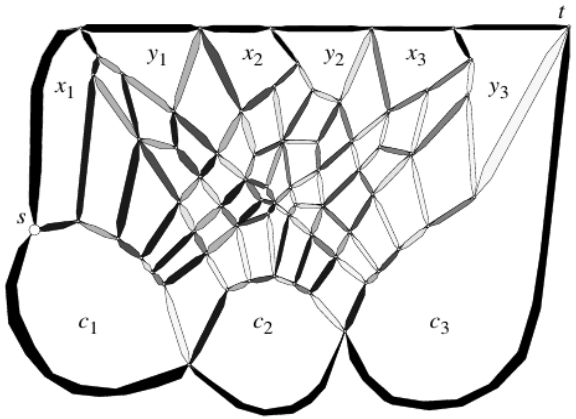
Konstruktion von G



Ersetze:

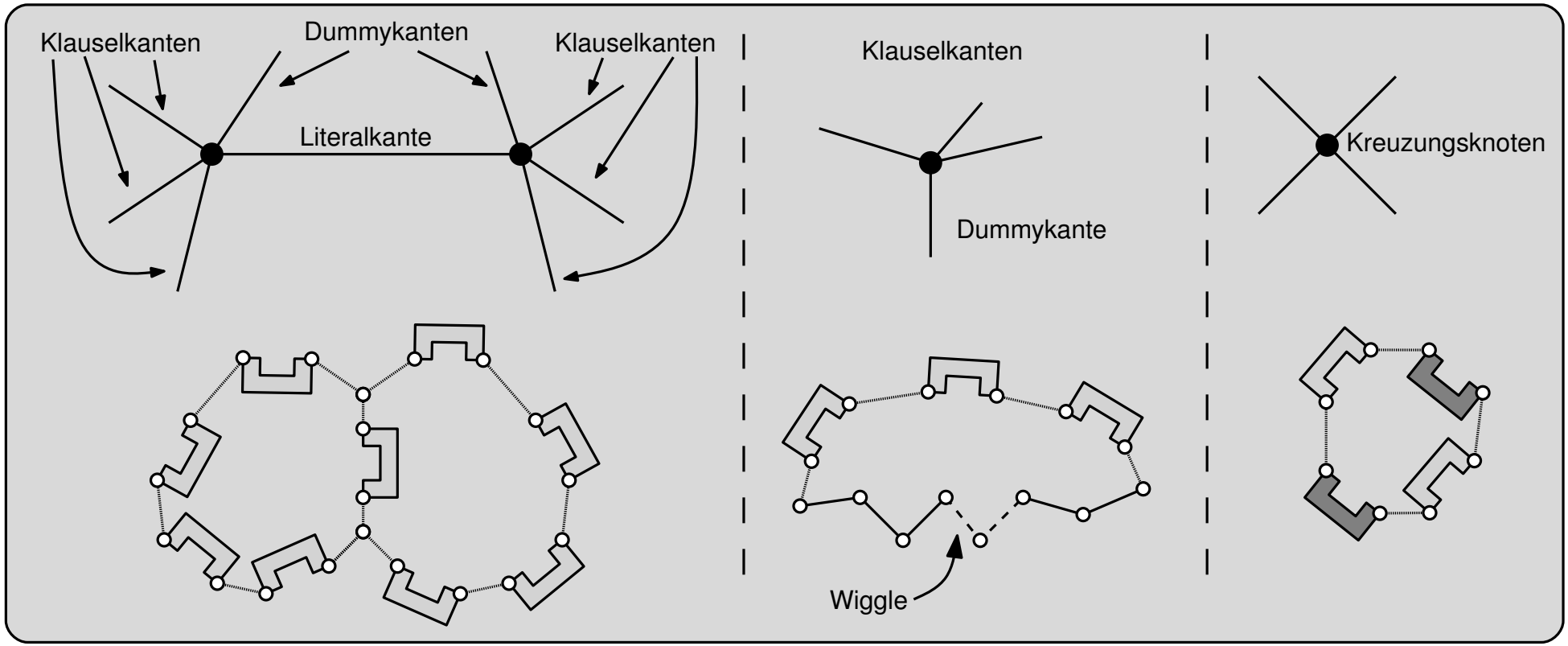
- Kante mit Kapazität $[c] \rightarrow$ Tendril T_c
- Kante mit Kapazität $[0 \cdots c] \rightarrow$ Wiggle W_c

Konstruktion von G



Ersetze:

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



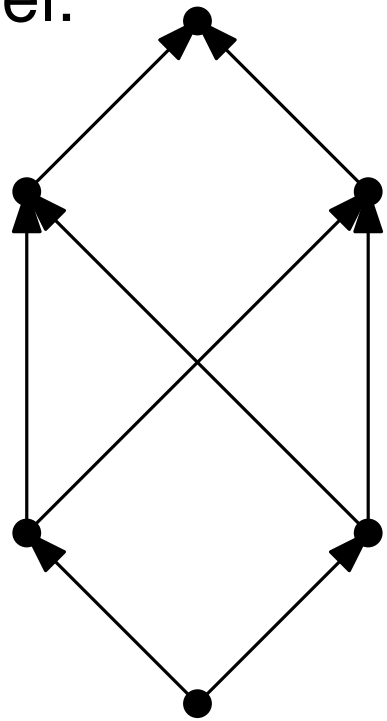
Aufwärtsplanare Zeichnungen

Aufwärtsplanarität

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:

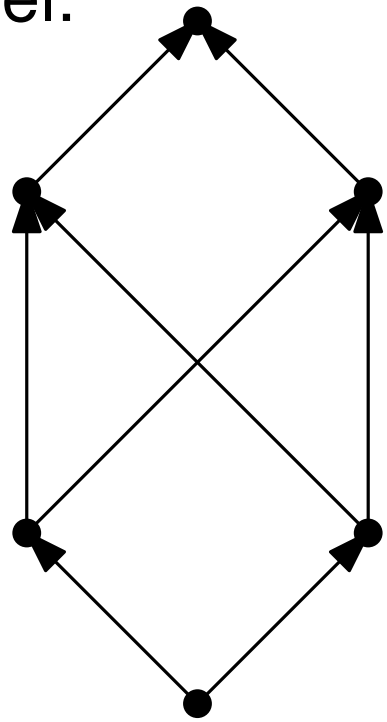


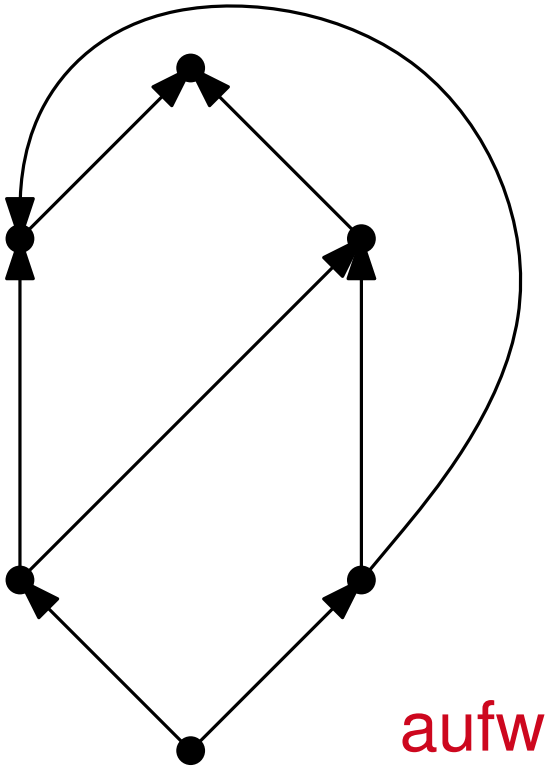
Aufwärtsplanarität

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!

Problemstellung

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

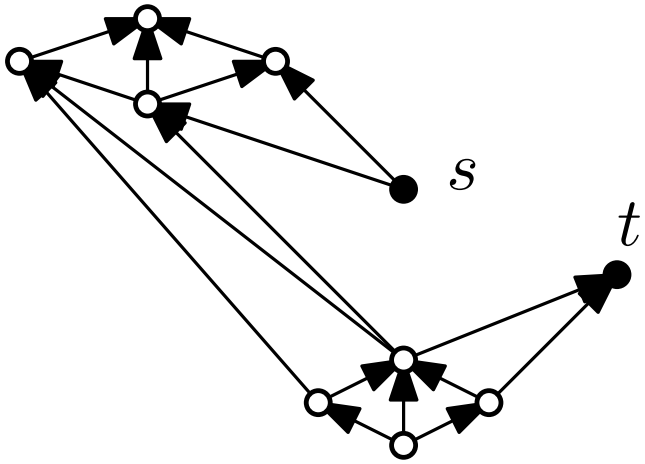
Problemstellung

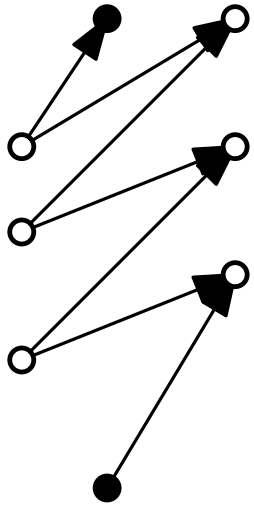
Problem: Test auf Aufwärtsplanarität

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 Layouts
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:

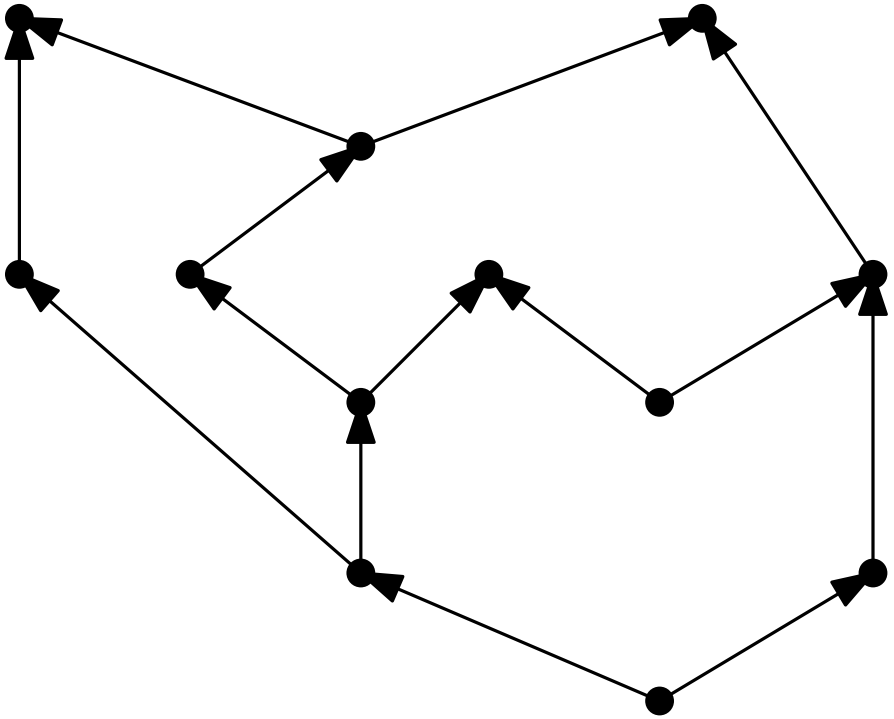
1. D ist aufwärtsplanar
2. D hat ein geradliniges aufwärtsplanares Layouts
3. D ist aufspannender Subgraph eines planaren st-Graphen

Beweis: (2) $\Rightarrow$ (1) ist klar
(3) $\Leftrightarrow$ (1) einfach
(3) $\Rightarrow$ (2) braucht etwas mehr Arbeit

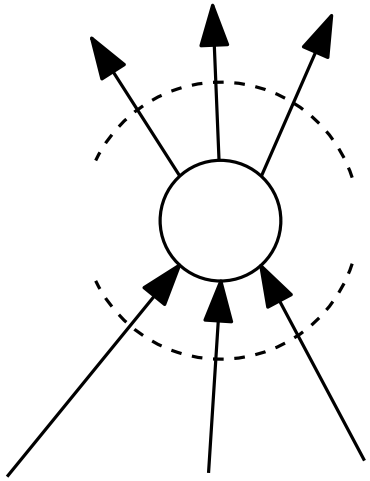
Neue Problemstellung: Feste Einbettung

Problem: Test auf Aufwärtsplanarität mit fester Einbettung

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



Einbettung ist bimodal



ausgehend

eingehend

Beboachtungen

Beboachtungen

- Bimodalität notwendig (nicht hinreichend)

Beboachtungen

- 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!

Beboachtungen

- 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!

Lemma:

- in Aufwärs-Layout von D gilt:

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

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

Folgerung

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

- in Aufwärs-Layout von D gilt:

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

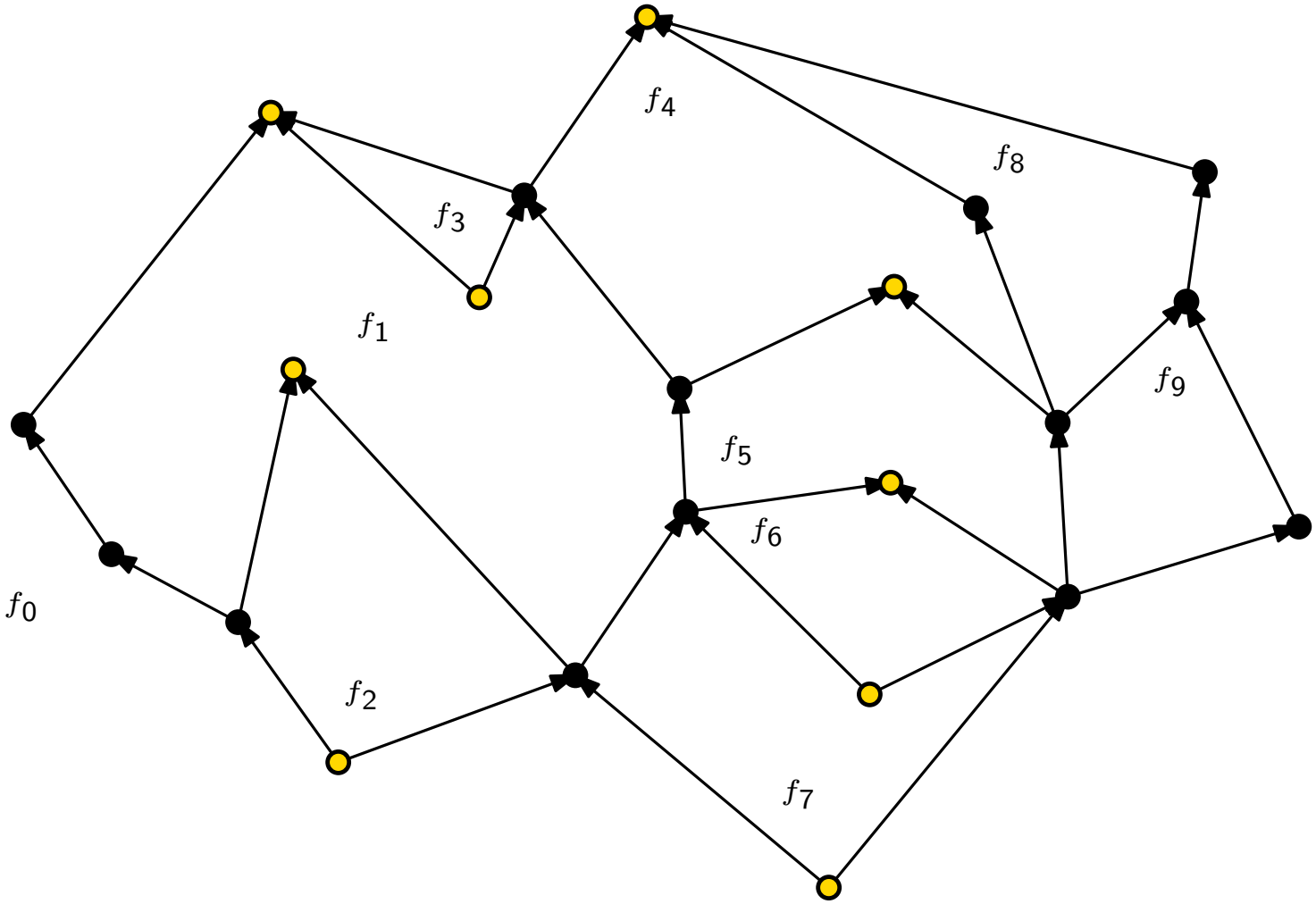
Definiert Zuordnung von Quellen/Senken zu inzidenten Facetten

$\Phi : \{Q, S\} \to \mathcal{F}$
$\Phi : v \mapsto$ inzidente Facette

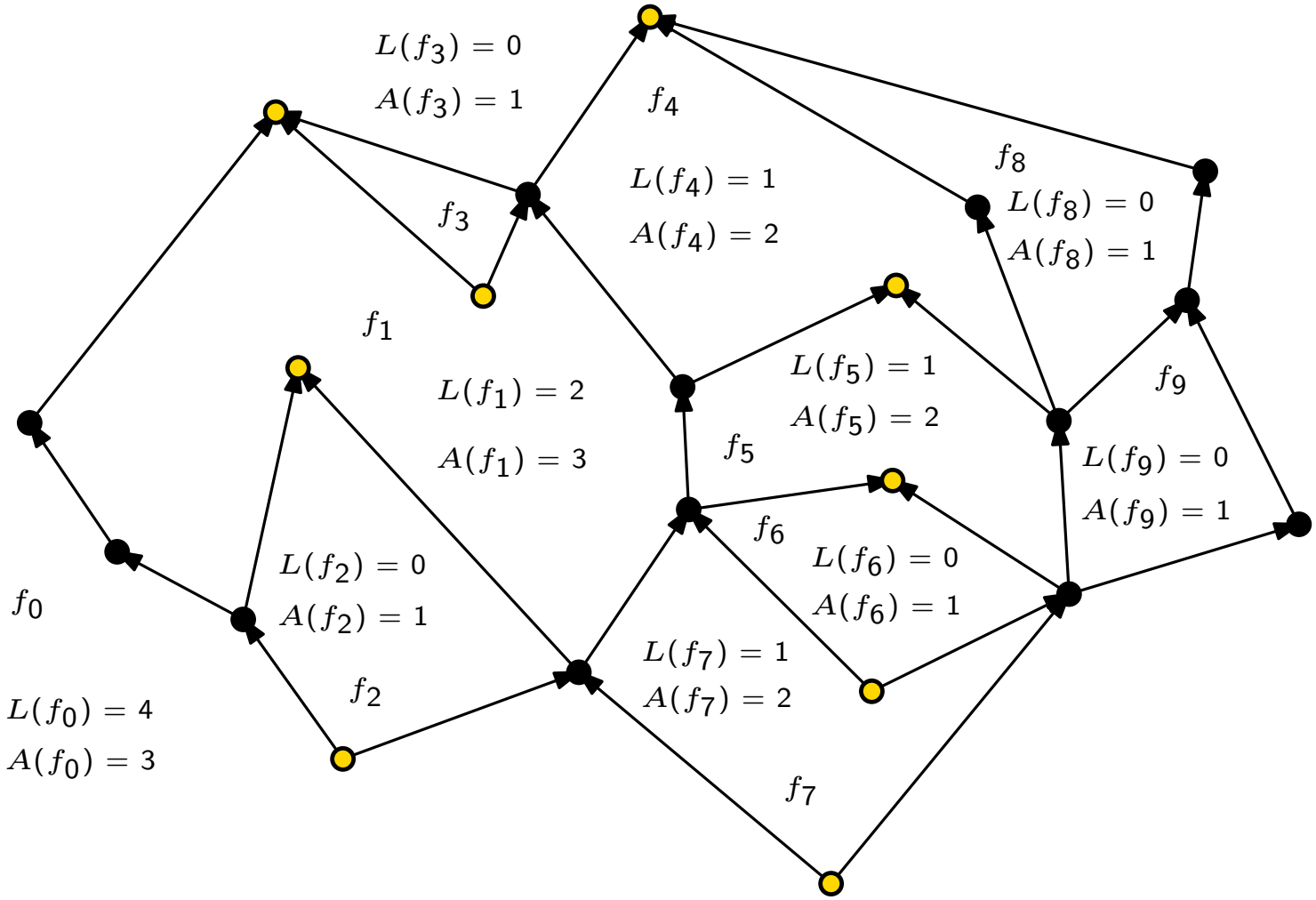
heißt *konsistent*, wenn gilt

$$|\Phi^{-1}(f)| = \begin{cases} A(f) - 1 & f \neq f_0 \\ A(f) + 1 & f_0 \end{cases}$$

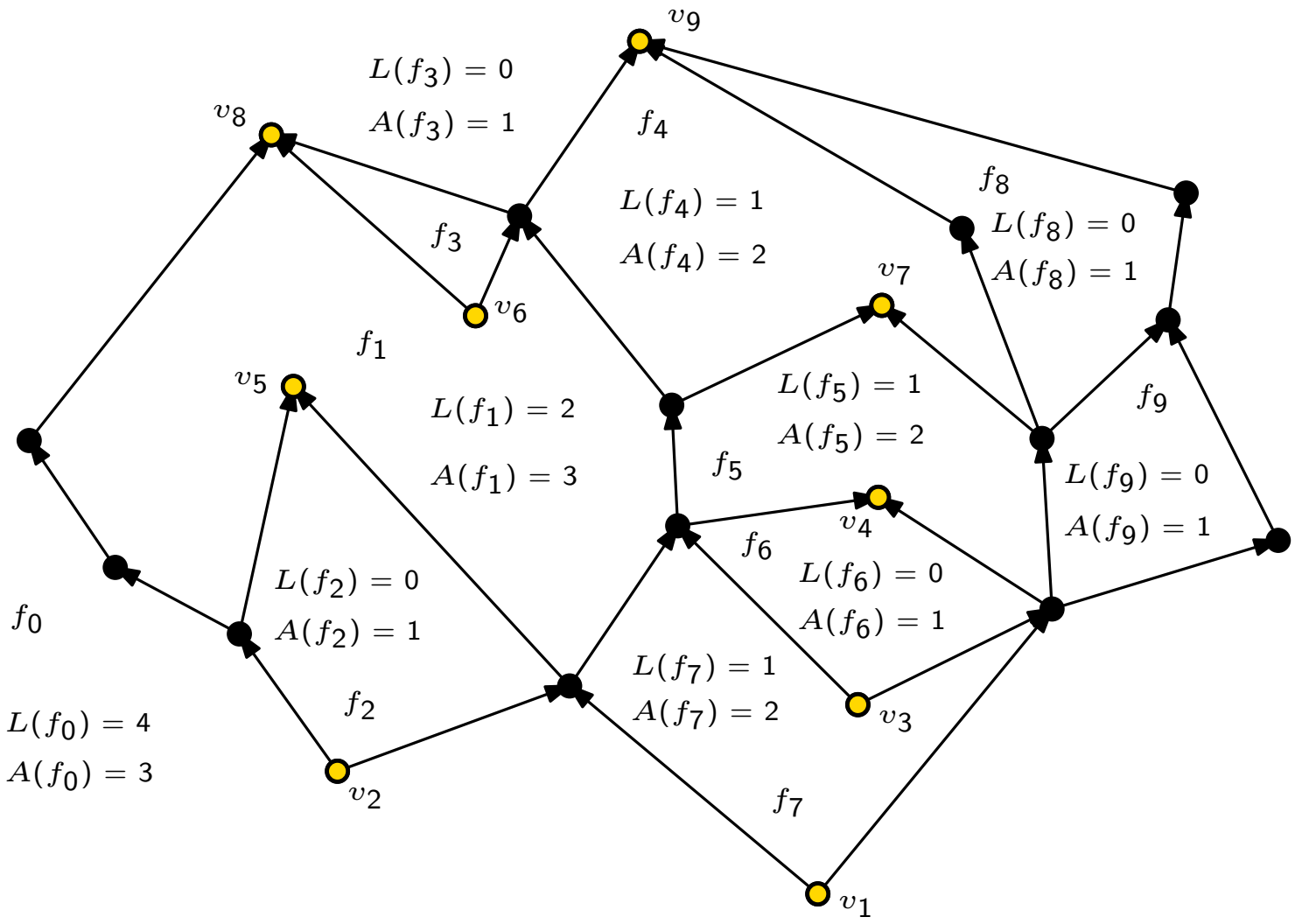
Beispiel Facettenzuordnung



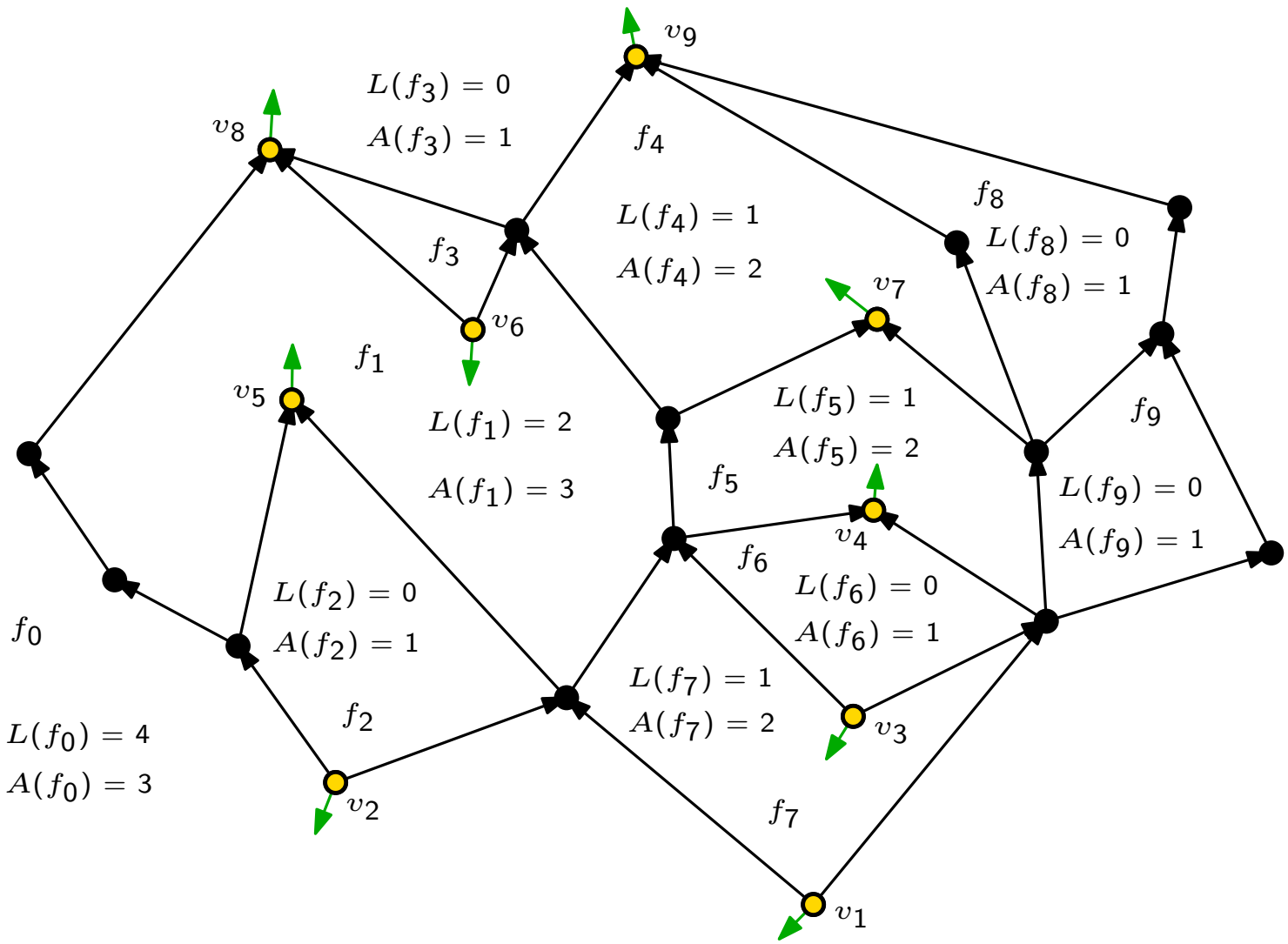
Beispiel Facettenzuordnung



Beispiel Facettenzuordnung



Beispiel Facettenzuordnung



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

Kernaussage

Satz

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

aufwärtsplanar $\iff$ bimodal und $\exists$ konsistentes Φ

Kernaussage

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

Kernaussage

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

Kernaussage

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

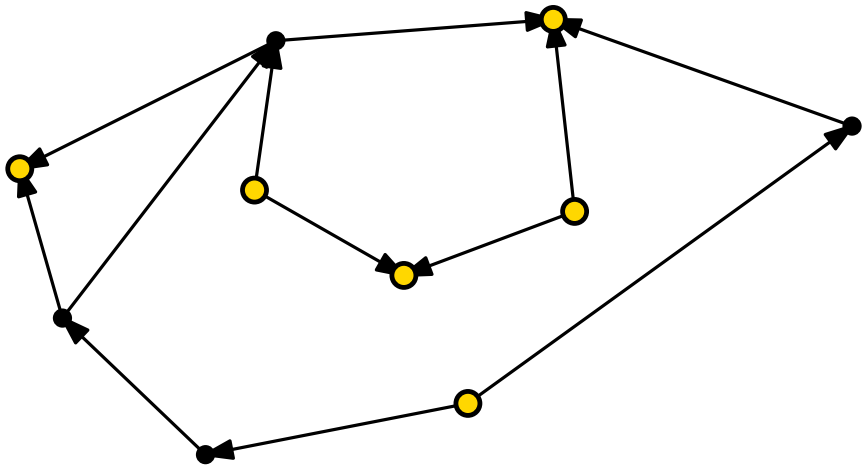
Flußnetzwerk!

Flussnetzwerk zur Konstruktion von Φ

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

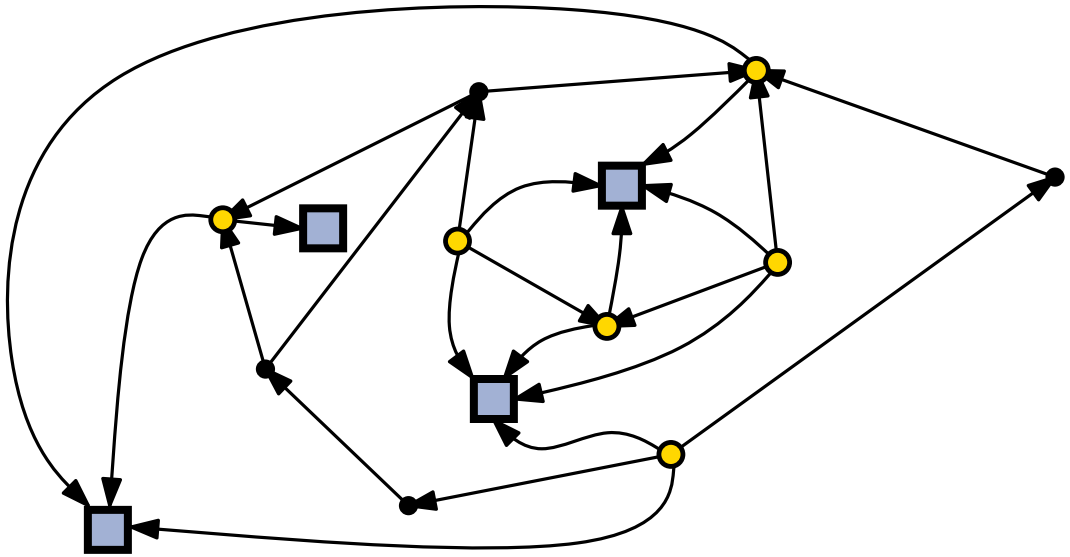
- $W = \{v \in V \mid v \text{ ist Quelle oder Senke}\} \cup \mathcal{F}$
- $A_N = \{(v, f) \mid v \text{ inzident zu } f$
- $l(a) = 0 \quad \forall a \in A_N$
- $u(a) = 1 \quad \forall a \in A_N$
- $$b(q) = \begin{cases} 1 & \forall q \in W \cap V \\ -(A(q) - 1) & \forall q \in \mathcal{F} \setminus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

Beispielnetzwerk



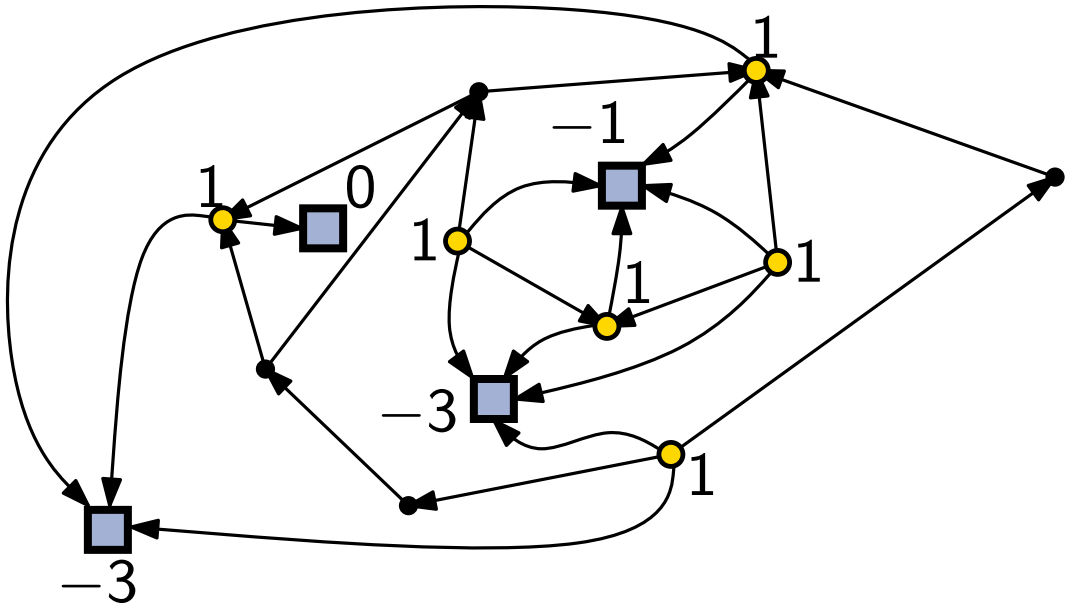
- normaler Knoten
- Quelle / Senke

Beispielnetzwerk



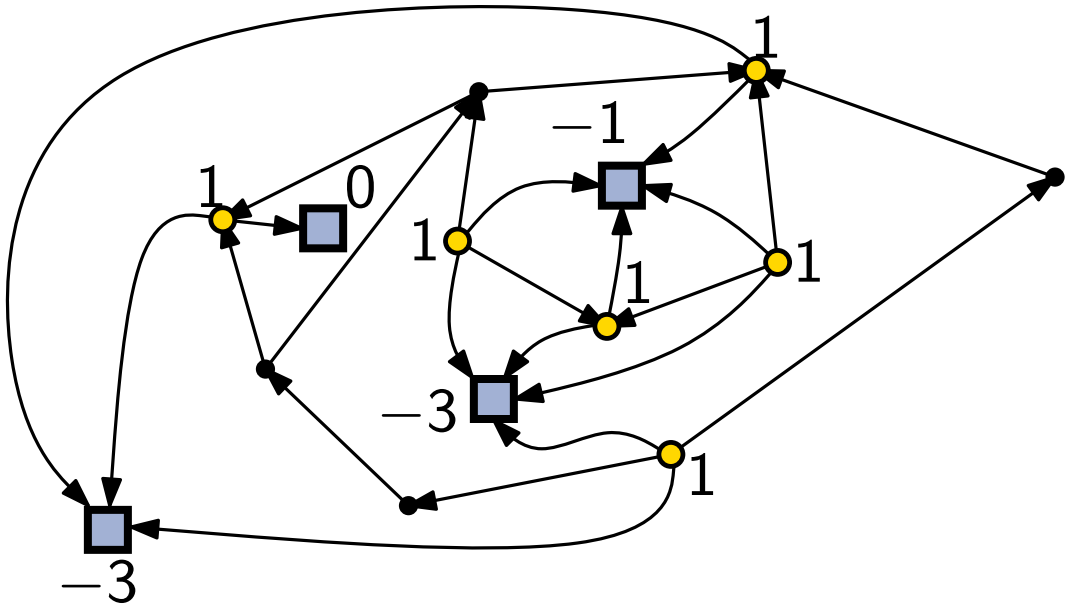
- normaler Knoten
- Quelle / Senke
- Facettenknoten

Beispielnetzwerk



- normaler Knoten
- Quelle / Senke
- Facettenknoten

Beispielnetzwerk

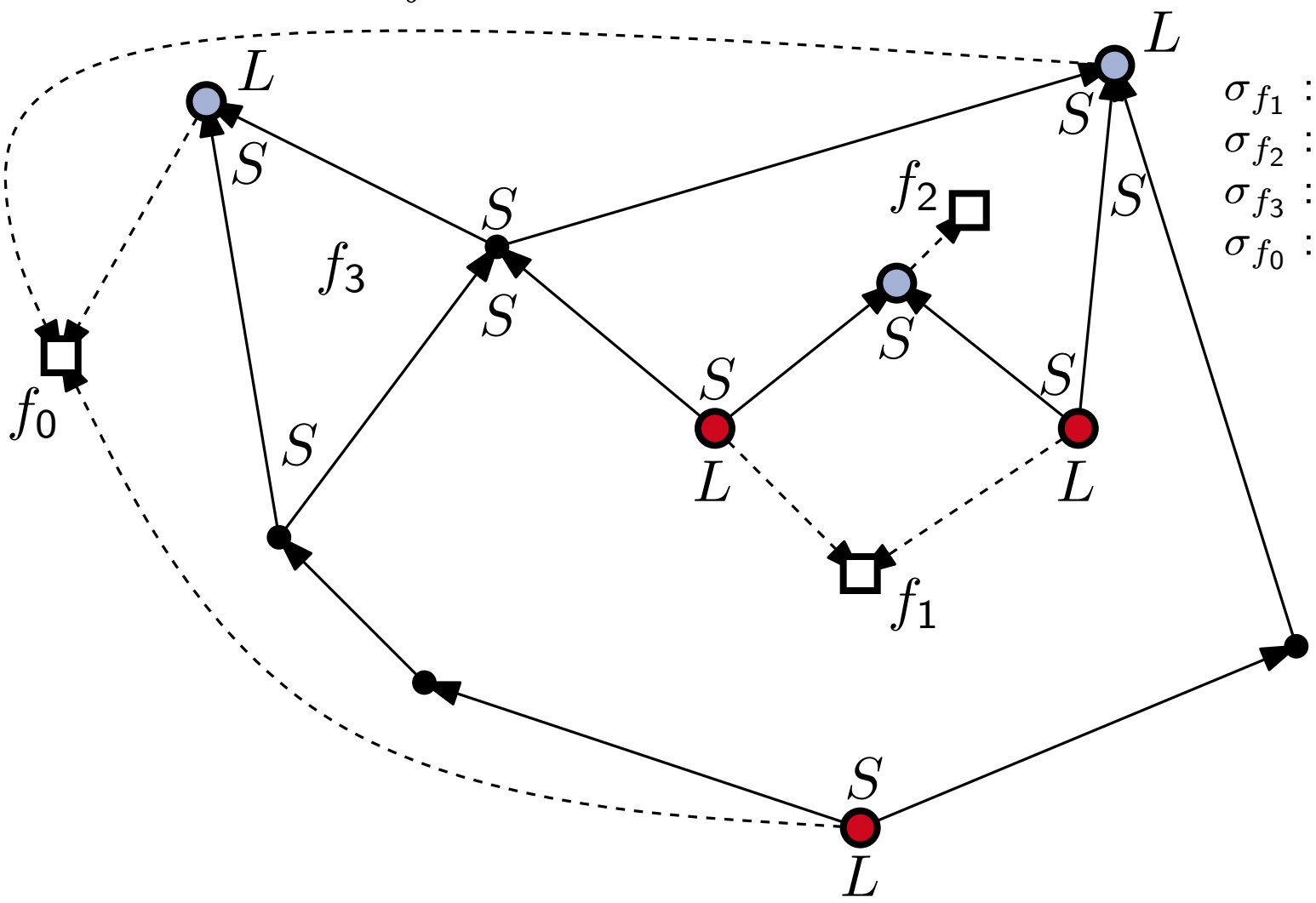


- normaler Knoten
- Quelle / Senke
- Facettenknoten

- 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



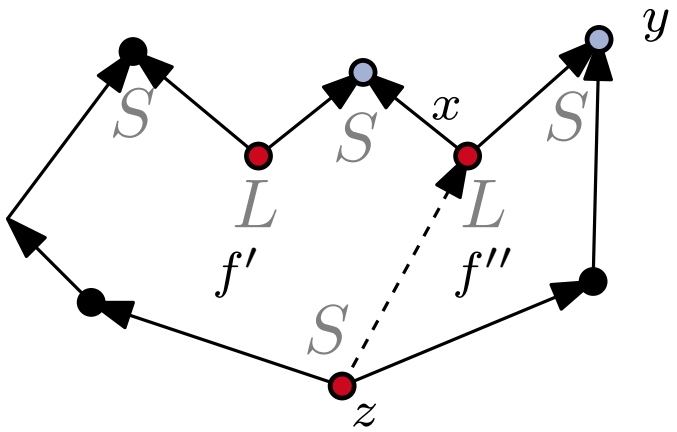
$$\sigma_{f_1} := (S, S, L, S, L, S)$$
$$\sigma_{f_2} := (L, S, S, S)$$
$$\sigma_{f_3} := (S, S)$$
$$\sigma_{f_0} := (L, L, S, L)$$

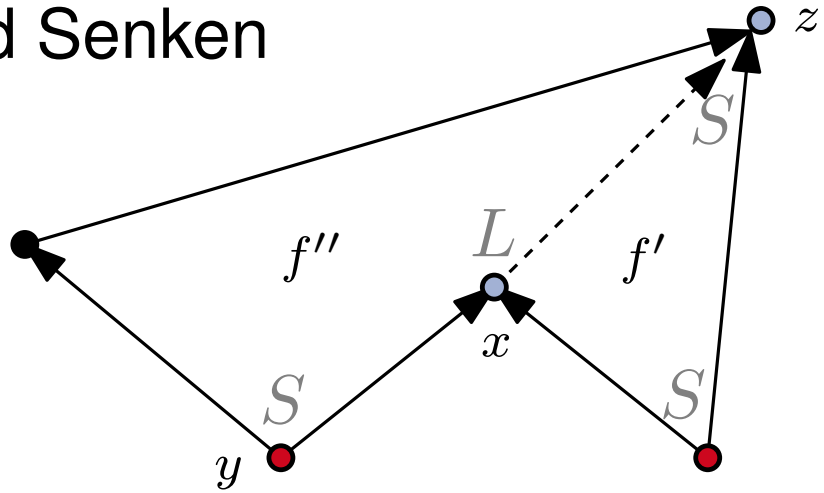
Algorithmus: $\Phi, \mathcal{F}, f_0 \rightsquigarrow$ s-t-Graph $\supseteq D$

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

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

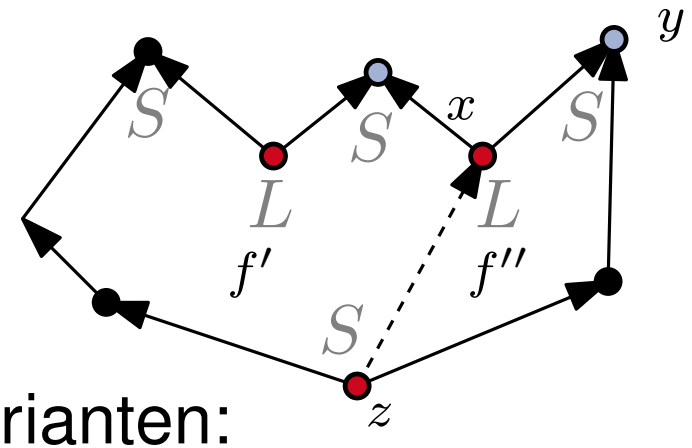
- $f \neq f_0$ mit $|\sigma_f| \geq 2$ enthält L, S, S an x, y, z
- x Quelle $\Rightarrow$ verfeinere mit (z, x)
- x Senke $\Rightarrow$ verfeinere mit (x, z)
- Ziel: Entferne alle Quellen und Senken

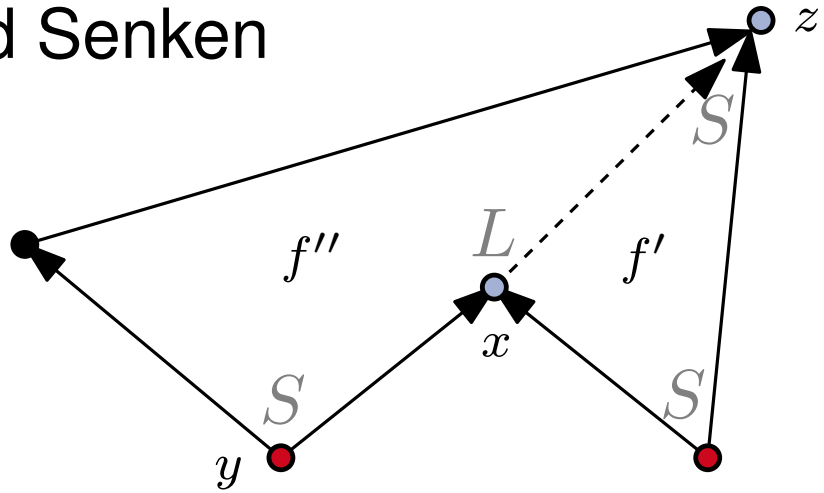




Algorithmus: $\Phi, \mathcal{F}, f_0 \rightsquigarrow$ s-t-Graph $\supseteq D$

- $f \neq f_0$ 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