Algorithmen zur Visualisierung von Graphen

Aufwärtsplanare Zeichnungen

Vorlesung im Wintersemester 2010/2011

Robert Görke

08.12.2010

Test auf Aufwärtsplanarität

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

Test auf Aufwärtsplanarität

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, eingebettet

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

Beobachtungen

Beobachtungen

$>>$ Bimodalität notwendig (nicht hinreichend)

Beobachtungen

- Bimodalität notwendig (nicht hinreichend)
- betrachte Winkel zw. zwei ein-/ausgehenden Kanten

Beobachtungen

$>>$ Bimodalität notwendig (nicht hinreichend)

$>>$ betrachte Winkel zw. zwei ein-/ausgehenden Kanten

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

Beobachtungen

$>>$ Bimodalität notwendig (nicht hinreichend)

$>>$ betrachte Winkel zw. zwei ein-/ausgehenden Kanten

$>>$ Lemma: in aufw. 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) $\forall f \in \mathcal{F} : L(f) - S(f) = \begin{cases} -2 & \neq f_0 \\ 2 & f_0 \end{cases} \quad \Bigg| \quad \begin{matrix} L(f) = A(f) - 1 \\ L(f) = A(f) + 1 \end{matrix}$

Beobachtungen

Karlsruher Institut für Technologie

$>>$ Bimodalität notwendig (nicht hinreichend)

$>>$ betrachte Winkel zw. zwei ein-/ausgehenden Kanten

$>>$ Lemma: in aufw. 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) $\forall f \in \mathcal{F} : L(f) - S(f) = \begin{cases} -2 & \neq f_0 \\ 2 & f_0 \end{cases} \quad \left| \quad \begin{array}{l} L(f) = A(f) - 1 \\ L(f) = A(f) + 1 \end{array} \right.$

$\Phi : \{Q, S\} \to \mathcal{F}$

$\Phi : v \mapsto$ inzidente Facette

Beobachtungen

$>>$ Bimodalität notwendig (nicht hinreichend)

$>>$ betrachte Winkel zw. zwei ein-/ausgehenden Kanten

$>>$ Lemma: in aufw. Layout von D gilt:

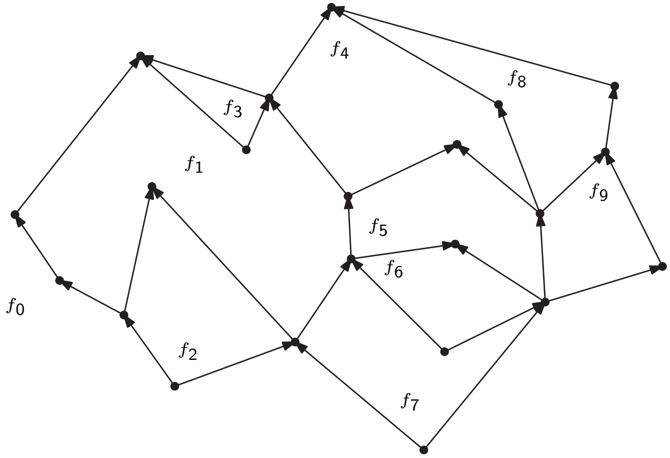
$$(1)\ \forall v \in V : L(v) = \begin{cases} 0 & \text{v innerer Knoten} \\ 1 & \text{v Quelle/Senke} \end{cases}$$

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

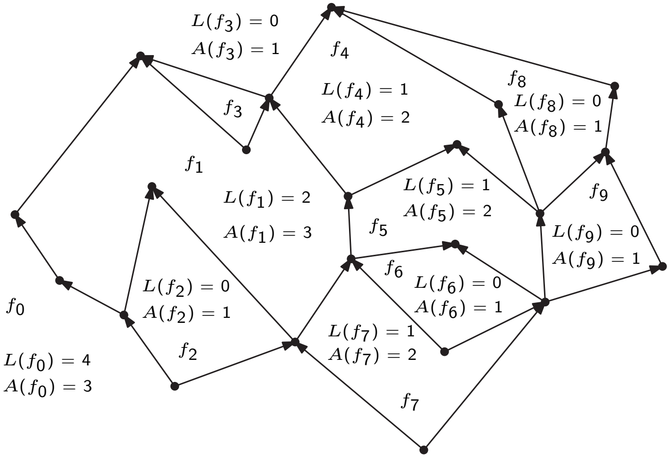
$\Phi : \{Q, S\} \to \mathcal{F}$
$\Phi : v \mapsto$ inzidente Facette
heißt *konsistent*, wenn gilt

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

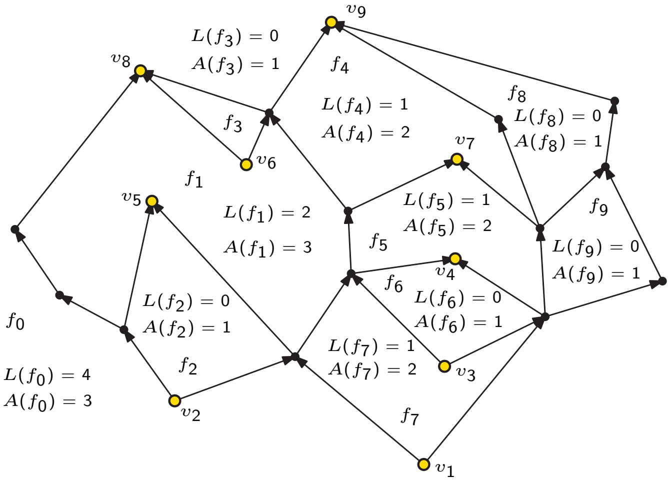
Beispiel Facettenzuordnung



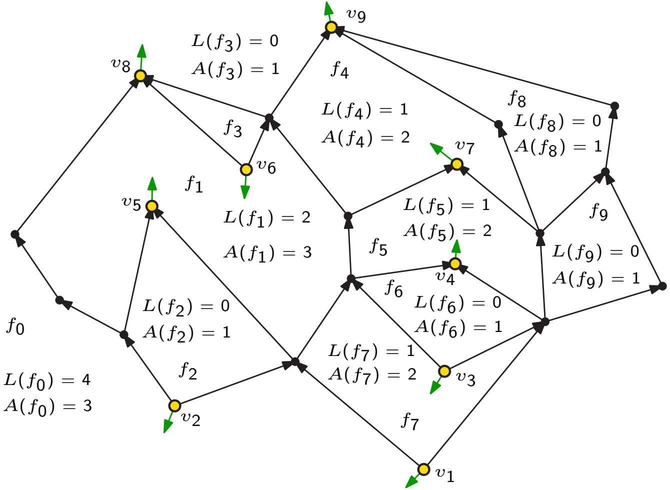
Beispiel Facettenzuordnung



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

Karlsruher Institut für Technologie

Satz

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

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

Kernaussage

Satz

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

$$\text{aufwärtsplanar} \iff \text{bimodal und } \exists \text{ kosistentes } \Phi$$

$\Rightarrow$: soeben hergeleitet

Kernaussage

Satz

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

$$\text{aufwärtsplanar} \iff \text{bimodal und } \exists \text{ kosistentes } \Phi$$

$\Rightarrow$: soeben hergeleitet

$\Leftarrow$: Umkehrung gilt, ist sogar konstruktiv

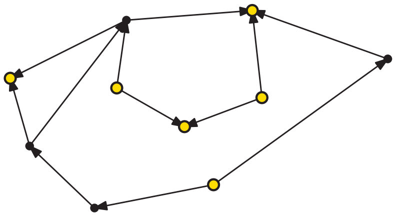
Zunächst: $D, \mathcal{F}, f_0 \quad \overset{?}{\rightsquigarrow} \quad \Phi$ kosistent

Flussnetzwerk zur Konstruktion von Φ

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

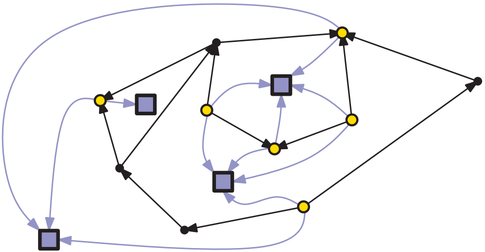
- $W = \{v \in V \mid v \text{ ist Quelle oder Senke}\} \cup \mathcal{F}$
- $A_N = \{(v, f) \mid v \text{ inzident zu } f$
- $l(a) = 0 \quad \forall a \in A_N$
- $u(a) = 1 \quad \forall a \in A_N$
- $b(q) = \begin{cases} 1 & \forall q \in W \cap V \\ -(A(q) - 1) & \forall q \in \mathcal{F} \setminus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

Beispielnetzwerk



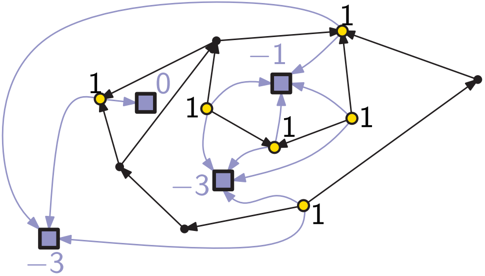
- normaler Knoten
- Quelle / Senke

Beispielnetzwerk



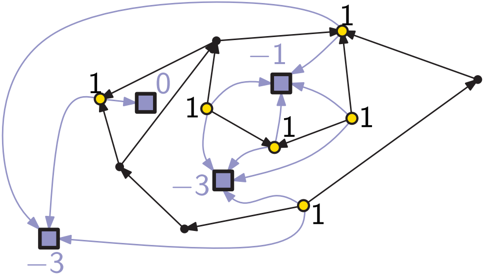
- normaler Knoten
- Quelle / Senke
- Facettenknoten

Beispielnetzwerk



- normaler Knoten
- Quelle / Senke
- Facettenknoten

Beispielnetzwerk



- normaler Knoten
- Quelle / Senke
- Facettenknoten

≫ Starte mit Nullfluss

≫ Suche erhöhende Wege

≫ Geht auch ohne festgelegtes f_0

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

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

- 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 $\supseteq D$

- Betrachte Folge σ_f von Winkeln L, S an lokalen Quellen und Senken von f
- $f \neq f_0$ mit $|\sigma_f| \geq 2$ enthält L, S, S an x, y, z

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

- Betrachte Folge σ_f von Winkeln L, S an lokalen Quellen und Senken von f
- $f \neq f_0$ 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)

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

Karlsruher Institut für Technologie

- Betrachte Folge σ_f von Winkeln L, S an lokalen Quellen und Senken von f
- $f \neq f_0$ 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 $\supseteq D$

Karlsruher Institut für Technologie

- Betrachte Folge σ_f von Winkeln L, S an lokalen Quellen und Senken von f
- $f \neq f_0$ 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
- Füge zwischen irgendeiner Quelle s und Senke t auf f_0 die Kante (s, t) ein.
  $\Rightarrow$ planarer s-t-Graph, der D enthält