Karlsruher Institut für Technologie

Theoretische Grundlagen der Informatik Übung

9. Übungstermin · 5. Februar 2019

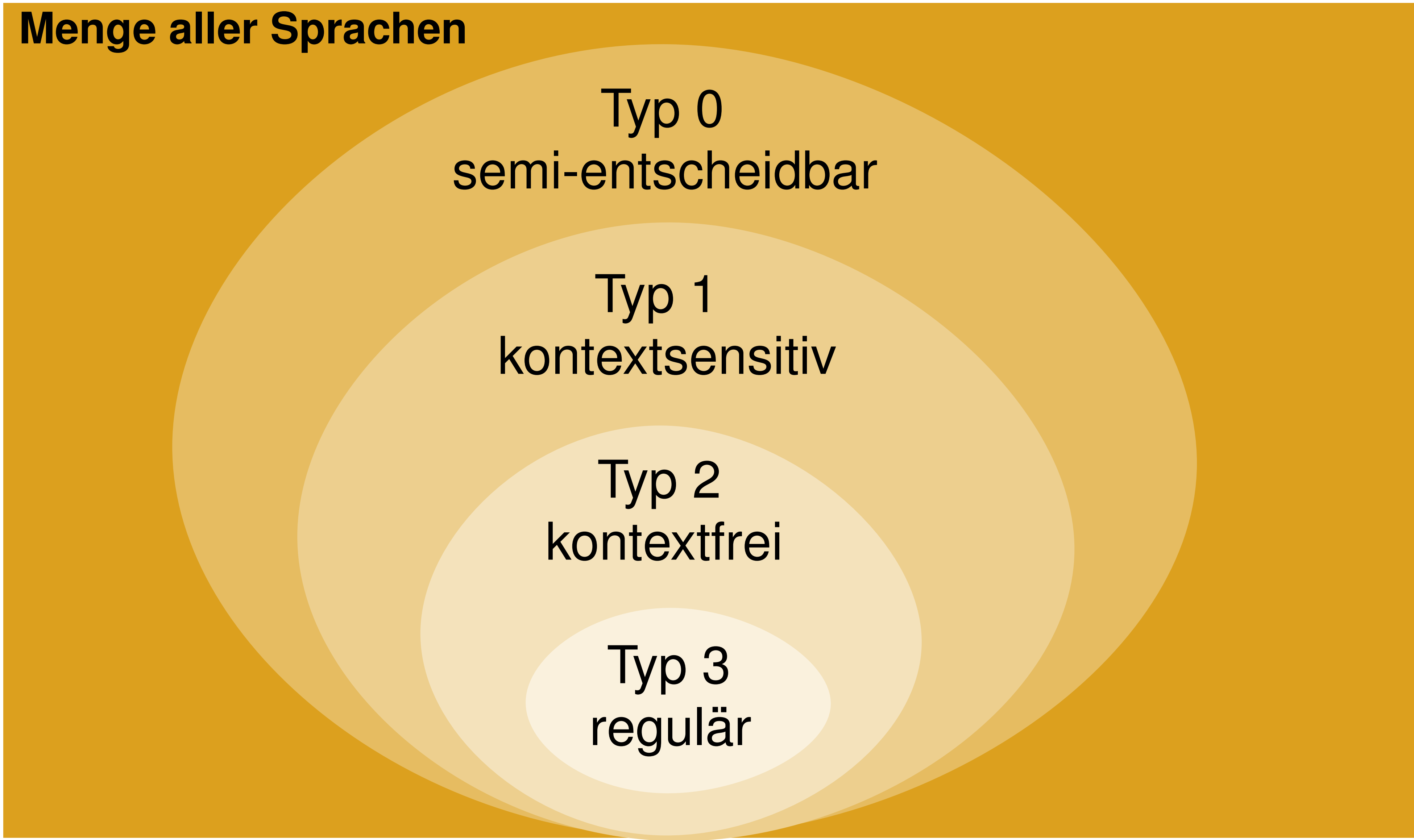
Jonas Sauer

Institut für Theoretische Informatik · Lehrstuhl Algorithmik

Übersicht

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Chomsky-Hierarchie



Einordnung

Typ	Erkannt durch	Erzeugt durch
0		
1		
2		
3		

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Einordnung

Typ	Erkannt durch	Erzeugt durch
0	■ NTM ■ DTM	■ 'Typ-0-Grammatik'
1		
2		
3		

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Einordnung

Typ	Erkannt durch	Erzeugt durch
0	■ NTM ■ DTM	■ 'Typ-0-Grammatik'
1	■ linear beschränkte NTM (LBA)	■ kontextsensitive Grammatik
2		
3		

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Einordnung

Typ	Erkannt durch	Erzeugt durch
0	■ NTM ■ DTM	■ 'Typ-0-Grammatik'
1	■ linear beschränkte NTM (LBA)	■ kontextsensitive Grammatik
2	■ NPDA	■ kontextfreie Grammatik
3		

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Einordnung

Typ	Erkannt durch	Erzeugt durch
0	NTM DTM	’Typ-0-Grammatik’
1	linear beschränkte NTM (LBA)	kontextsensitive Grammatik
2	NPDA	kontextfreie Grammatik
3	NEA DEA	rechtslineare Grammatik regulären Ausdruck

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Einordnung

Karlsruher Institut für Technologie

Typ	Erkannt durch	Erzeugt durch
0	NTM DTM	'Typ-0-Grammatik'
1	linear beschränkte NTM (LBA)	kontextsensitive Grammatik
2	NPDA	kontextfreie Grammatik
3	NEA DEA	rechtslineare Grammatik regulären Ausdruck

DPDA?

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

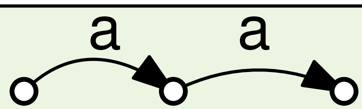
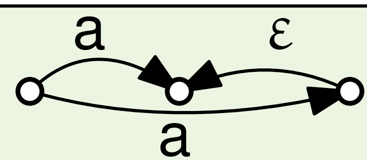
Einordnung

Typ	Erkannt durch	Erzeugt durch
0	■ NTM ■ DTM	■ 'Typ-0-Grammatik'
1	■ linear beschränkte NTM (LBA)	■ kontextsensitive Grammatik
2	■ NPDA	■ kontextfreie Grammatik
deterministisch kontextfreie Sprachen		
3	■ NEA ■ DEA	■ rechtslineare Grammatik ■ regulären Ausdruck

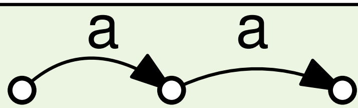
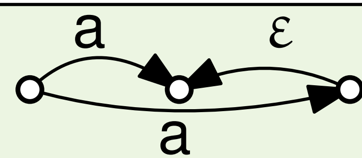
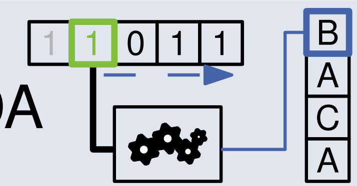
DPDA?

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

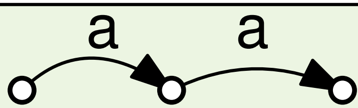
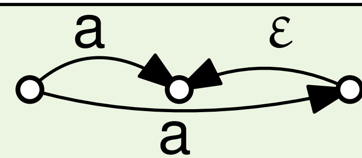
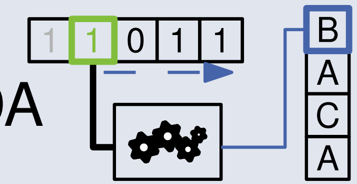
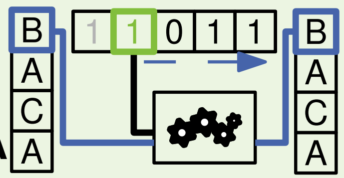
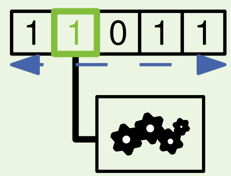
Maschinenmodelle

		gleich mächtig?	Poly. Trafo.?
DEA 	NEA 	✓	✗

Maschinenmodelle

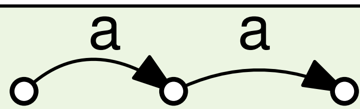
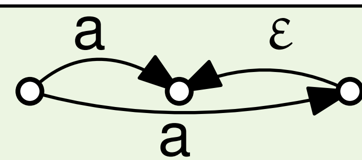
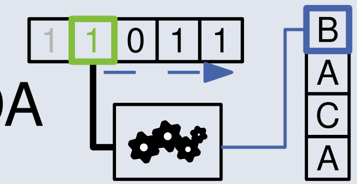
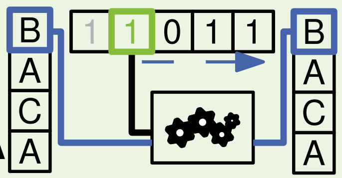
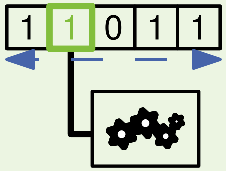
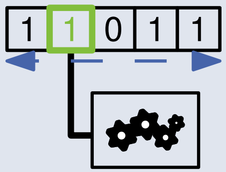
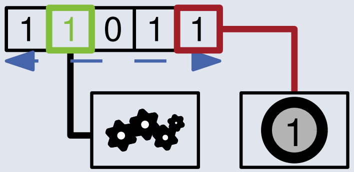
		gleich mächtig?	Poly. Trafo.?
DEA 	NEA 	✓	✗
DPDA	NPDA 	✗	

Maschinenmodelle

		gleich mächtig?	Poly. Trafo.?
DEA 	NEA 	✓	✗
DPDA	NPDA 	✗	
2-DPDA 	DTM 	✓	✓

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Maschinenmodelle

		gleich mächtig?	Poly. Trafo.?
DEA 	NEA 	✓	✗
DPDA	NPDA 	✗	
2-DPDA 	DTM 	✓	✓
DTM 	NTM 	✓	✓

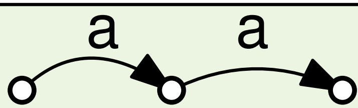
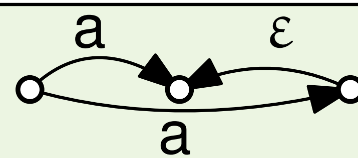
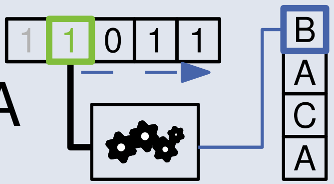
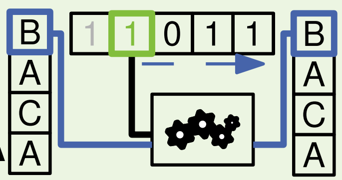
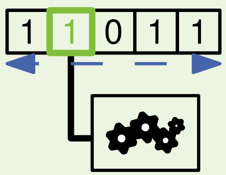
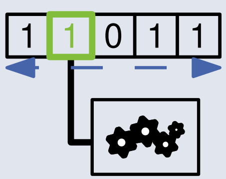
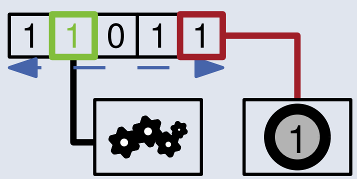
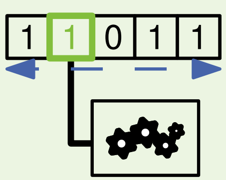
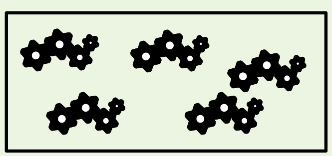
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Maschinenmodelle

		gleich mächtig?	Poly. Trafo.?
DEA	NEA	✓	✗
DPDA	NPDA	✗	
2-DPDA			✓
DTM	NTM	✓	✓

Achtung: Das hat nichts mit $P \stackrel{?}{=} NP$ zu tun!

Maschinenmodelle

		gleich mächtig?	Poly. Trafo.?
DEA 	NEA 	✓	✗
DPDA	NPDA 	✗	
2-DPDA 	DTM 	✓	✓
DTM 	NTM 	✓	✓
DTM 	DIE Maschine! 	Intuitiv: ✓	

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Maschinenmodelle

		gleich mächtig?	Poly. Trafo.?
DEA	NEA	✓	✗
DPDA	NPDA	✗	
2-DPDA	DTM	✓	✓
DTM	NTM	✓	✓
DTM	DIE Maschine!	Intuitiv: ✓	

Mehrband-TM, Mehrspur-TM, RAM,. . .

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Maschinenmodelle

		gleich mächtig?	Poly. Trafo.?
DEA	NEA	✓	✗
DPDA		✗	
2-DPDA			✓
DTM			✓
DTM	schine!	✓	

Mehrband-TM, Mehrspur-TM, RAM,. . .

Wie werden die Eigenschaften gezeigt?

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Entscheidbarkeit

Typ	$w \in L$	$L(\mathcal{M}) = \emptyset$	$L(\mathcal{M}_1) = L(\mathcal{M}_2)$	$L(\mathcal{M}_1) \cap L(\mathcal{M}_2) = \emptyset$
0	✗	✗	✗	✗
1	✓ NP-schwer	✗	✗	✗
2	✓ Ch-NF: $\mathcal{O}(n^3)$	✓	✗	✗
3	✓ $\mathcal{O}(n)$	✓	✓	✓

Karlsruher Institut für Technologie

Typ-1-Grammatik

Typ-1-Grammatik

Grammatik der Form $\mathbb{X}$: $\alpha A \beta \rightarrow \alpha\gamma\beta, \alpha, \beta, \gamma \in (V \cup \Sigma)^{\star}, |\gamma| > 0$

Grammatik der Form $\mathbb{Y}$: $\alpha \rightarrow \beta, |\alpha| \leq |\beta|, \alpha \in V^{+}, \beta \in ((V \cup \Sigma) \setminus S)^{+}$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Typ-1-Grammatik

Grammatik der Form $\mathbb{X}$: $\alpha A\beta \to \alpha\gamma\beta, \alpha, \beta, \gamma \in (V \cup \Sigma)^\star, |\gamma| > 0$

Grammatik der Form $\mathbb{Y}$: $\alpha \to \beta, |\alpha| \leq |\beta|, \alpha \in V^+, \beta \in ((V \cup \Sigma) \setminus S)^+$

Wird die gleiche Klasse von Sprachen generiert?

Typ-1-Grammatik

Grammatik der Form $\mathbb{X}$: $\alpha A \beta \to \alpha\gamma\beta, \alpha, \beta, \gamma \in (V \cup \Sigma)^\star, |\gamma| > 0$

Grammatik der Form $\mathbb{Y}$: $\alpha \to \beta, |\alpha| \leq |\beta|, \alpha \in V^+, \beta \in ((V \cup \Sigma) \setminus S)^+$

Wird die gleiche Klasse von Sprachen generiert?

$\mathcal{L}(\mathbb{X}) \subset \mathcal{L}(\mathbb{Y})$:

- $A \to \gamma$: Klar (da $|\gamma| \geq 1$)
- $A \to \ldots S \ldots$: Neues Startsymbol S'
- $a_1, \ldots, a_n \in \Sigma$ in α, β: Platzhalter $A_1, \ldots, A_n$

$\mathbb{X}$:

$X_1 a X_2 bX_3 \to X_1 a Y_1 c Y_2 bX_3$

$\mathbb{Y}$:

$X_1 A X_2 B X_3 \to X_1 A Y_1 C Y_2 B X_3$
$\to \cdots \to X_1 a Y_1 c Y_2 b X_3$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Typ-1-Grammatik

Grammatik der Form $\mathbb{X}$: $\alpha A \beta \rightarrow \alpha\gamma\beta, \alpha, \beta, \gamma \in (V \cup \Sigma)^\star, |\gamma| > 0$

Grammatik der Form $\mathbb{Y}$: $\alpha \rightarrow \beta, |\alpha| \leq |\beta|, \alpha \in V^+, \beta \in ((V \cup \Sigma) \setminus S)^+$

Wird die gleiche Klasse von Sprachen generiert?

$\mathcal{L}(\mathbb{Y}) \subset \mathcal{L}(\mathbb{X})$:

$$X_1 X_2 \ldots X_n \rightarrow Y_1 \ldots Y_{n+m}$$
$$X_i \in V, Y_i \in ((V \cup \Sigma) \setminus S)$$

- $X_1 X_2 \ldots X_n \rightarrow Z_1 X_2 \ldots X_n$
- $Z_1 X_2 \ldots X_n \rightarrow Z_1 Z_2 X_3 \ldots X_n$
- $\ldots$
- $Z_1 \ldots Z_{n-2} X_{n-1} X_n \rightarrow Z_1 \ldots Z_{n-2} Z_{n-1} X_n$

Typ-1-Grammatik

Grammatik der Form $\mathbb{X}$: $\alpha A \beta \to \alpha\gamma\beta, \alpha, \beta, \gamma \in (V \cup \Sigma)^\star, |\gamma| > 0$

Grammatik der Form $\mathbb{Y}$: $\alpha \to \beta, |\alpha| \leq |\beta|, \alpha \in V^+, \beta \in ((V \cup \Sigma) \setminus S)^+$

Wird die gleiche Klasse von Sprachen generiert?

$\mathcal{L}(\mathbb{Y}) \subset \mathcal{L}(\mathbb{X})$: $X_1 X_2 \dots X_n \to Y_1 \dots Y_{n+m}$

$X_i \in V, Y_i \in ((V \cup \Sigma) \setminus S)$

- $Z_1 \dots Z_{n-1} X_n \to Z_1 \dots Z_{n-1} Z_n Y_{n+1} \dots Y_{n+m}$
- $Z_1 \dots Z_{n-1} Z_n Y_{n+1} \dots Y_{n+m} \to Y_1 \dots Z_{n-1} Z_n Y_{n+1} \dots Y_{n+m}$
- $\dots$
- $Y_1 \dots Y_{n-1} Z_n Y_{n+1} \dots Y_{n+m} \to Y_1 \dots Y_{n-1} Y_n Y_{n+1} \dots Y_{n+m}$

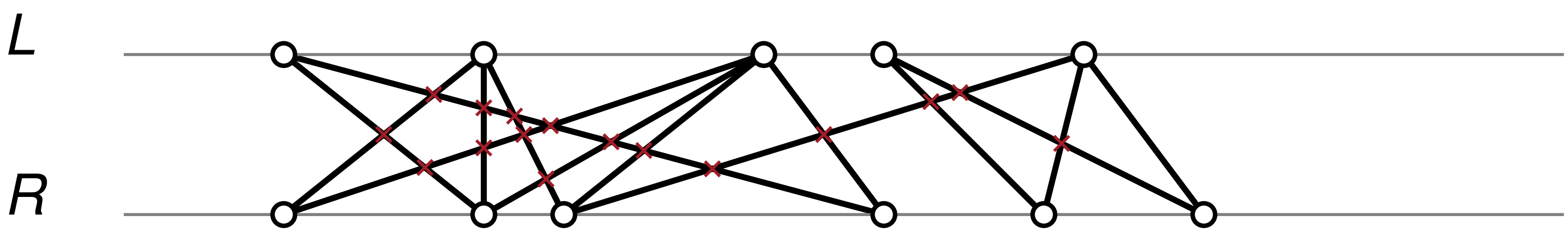
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Kreuzungsminimierung

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Einseitige Kreuzungsminimierung (OSCM)

Geg.: Bipartiter Graph $G = (L, R, E)$ und Knotenordnung r von R

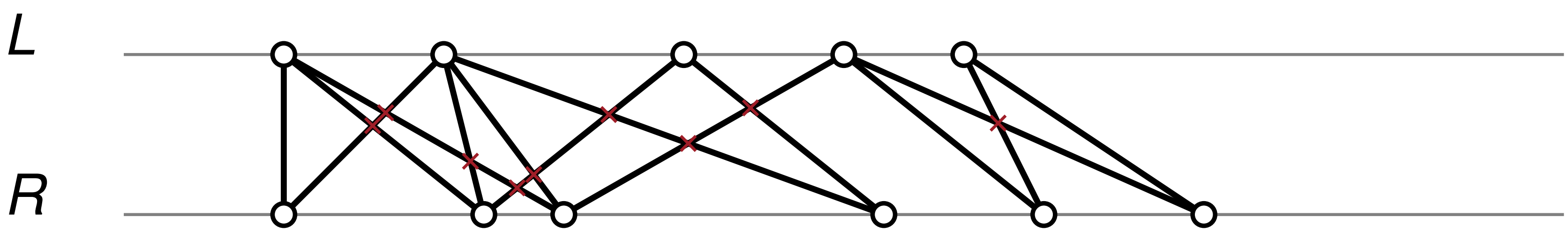


Ges.: Knotenordnung ℓ von L, sodass die Anzahl Kreuzungen von Kanten in E minimal ist

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Einseitige Kreuzungsminimierung (OSCM)

Geg.: Bipartiter Graph $G = (L, R, E)$ und Knotenordnung r von R



Ges.: Knotenordnung ℓ von L, sodass die Anzahl Kreuzungen von Kanten in E minimal ist

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Einseitige Kreuzungsminimierung (OSCM)

Geg.: Bipartiter Graph $G = (L, R, E)$ und Knotenordnung r von R

Ges.: Knotenordnung ℓ von L, so dass die Anzahl Kreuzungen von Kanten in E minimal ist

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Einseitige Kreuzungsminimierung (OSCM)

Geg.: Bipartiter Graph $G = (L, R, E)$ und Knotenordnung r von R

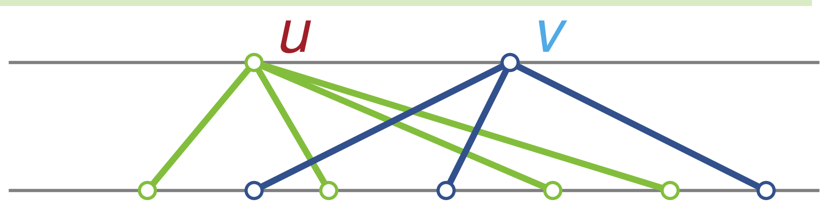
Ges.: Knotenordnung ℓ von L, so dass die Anzahl Kreuzungen von Kanten in E minimal ist

Beobachtung:

- Anzahl Kreuzungen einer 2-Lagen-Zeichnung von G hängt nur von ℓ und r ab, nicht von tatsächlichen Positionen
- Für $u, v \in L$ hängt Anzahl Kreuzungen inzidenter Kanten nur von $\ell(u) < \ell(v)$ oder $\ell(v) < \ell(u)$ ab

Def.: Kreuzungszahl für $\ell(u) < \ell(v)$

$$c_{uv} := \left|\{(uw, vz) \mid w \in N(u), z \in N(v), r(w) > r(z)\}\right|$$



$c_{uv} = 5$

Einseitige Kreuzungsminimierung (OSCM)

Geg.: Bipartiter Graph $G = (L, R, E)$ und Knotenordnung r von R

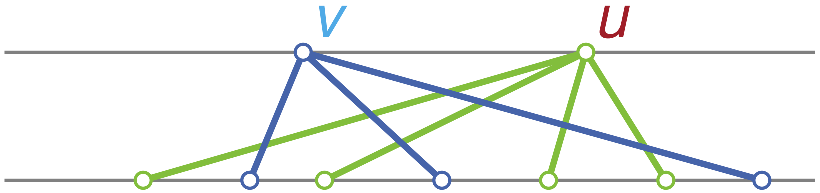
Ges.: Knotenordnung ℓ von L, so dass die Anzahl Kreuzungen von Kanten in E minimal ist

Beobachtung:

- Anzahl Kreuzungen einer 2-Lagen-Zeichnung von G hängt nur von ℓ und r ab, nicht von tatsächlichen Positionen
- Für $u, v \in L$ hängt Anzahl Kreuzungen inzidenter Kanten nur von $\ell(u) < \ell(v)$ oder $\ell(v) < \ell(u)$ ab

Def.: Kreuzungszahl für $\ell(u) < \ell(v)$

$$c_{uv} := \left|\{(uw, vz) \mid w \in N(u), z \in N(v), r(w) > r(z)\}\right|$$



$c_{uv} = 5$

$c_{vu} = 7$

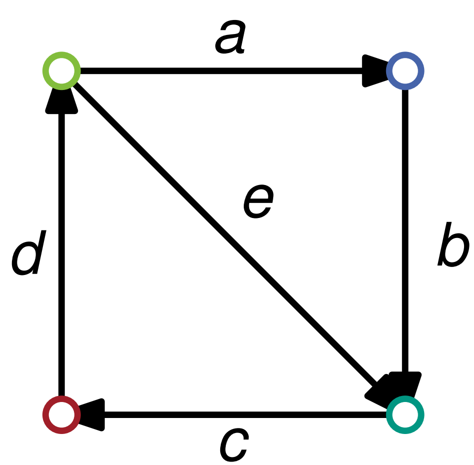
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Transformation

Satz 1: Das einseitige Kreuzungsminimierungsproblem (OSCM) ist NP-schwer.

Reduktion: FEEDBACK ARC SET $\propto$ OSCM

Problem FEEDBACKARCSET:
Gegeben: Gerichteter Graph $D = (V, A)$
Feedback Arc Set (FAS): Kantenmenge A', sodass $(V, A - A')$ azyklisch
Gesucht: FAS A' mit $|A'|$ minimal

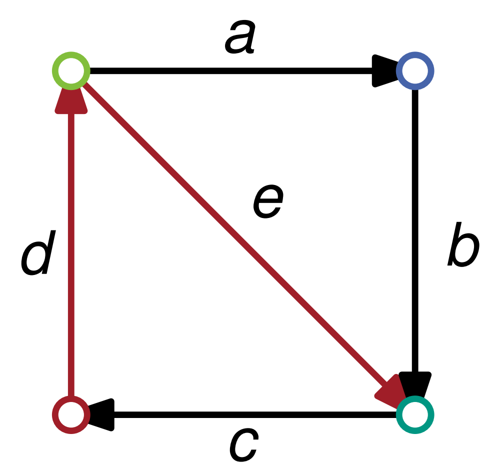


Transformation

Satz 1: Das einseitige Kreuzungsminimierungsproblem (OSCM) ist NP-schwer.

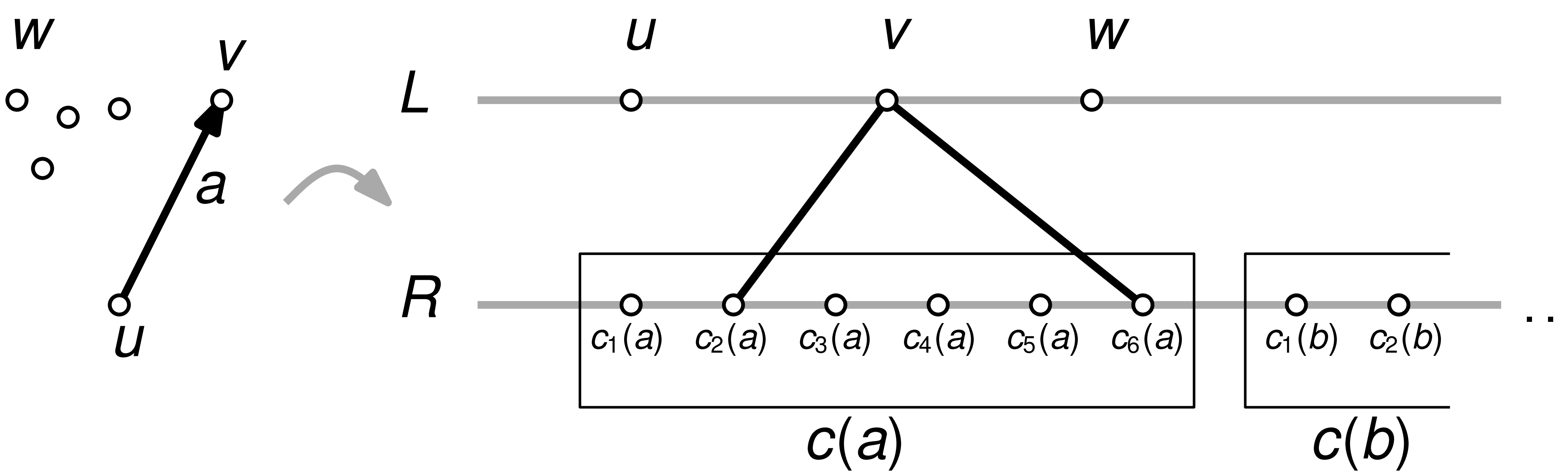
Reduktion: FEEDBACK ARC SET $\propto$ OSCM

Problem FEEDBACKARCSET:
Gegeben: Gerichteter Graph $D = (V, A)$
Feedback Arc Set (FAS): Kantenmenge A', sodass $(V, A - A')$ azyklisch
Gesucht: FAS A' mit $|A'|$ minimal



Feedback Arc Set $\propto$ OSCM

$D = (V, A)$ $\quad\quad$ $B = (L, R, E)$



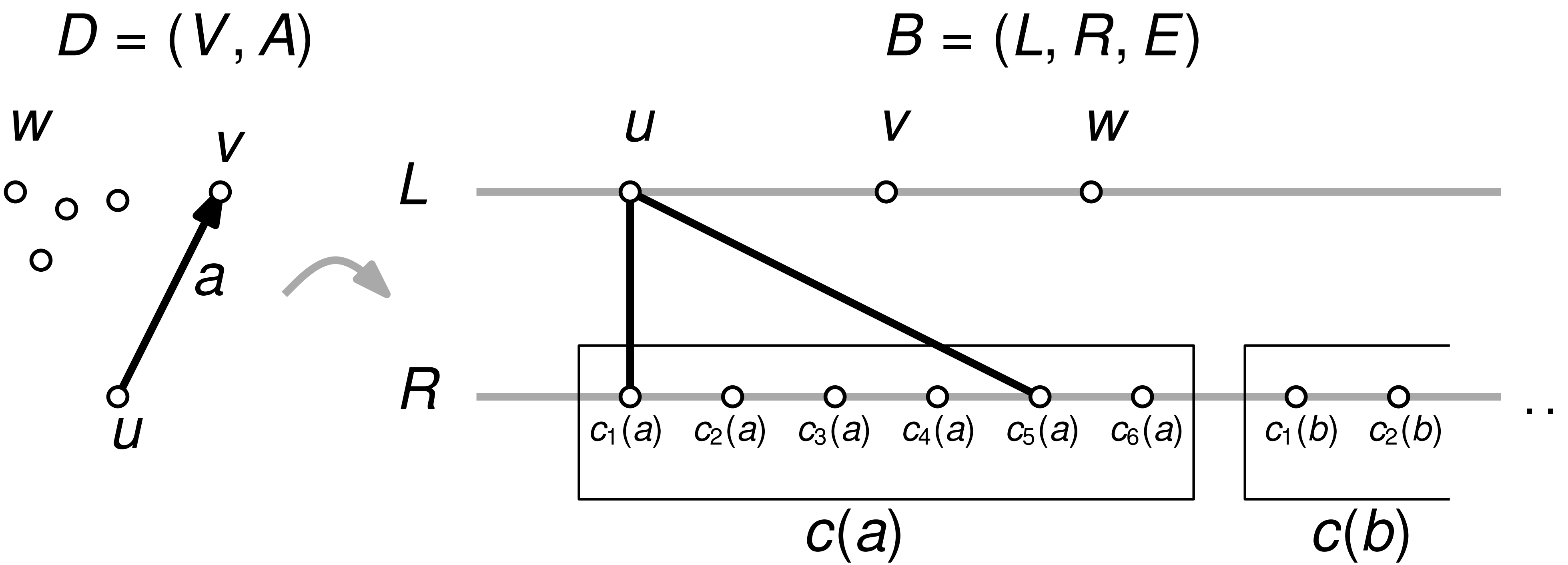
$L = V$

$$R = \bigcup_{a \in A} \{c_1(a), \ldots, c_6(a)\}$$

Für alle $v \in V$, $a \in A$:

- $a = (u, v)$: $\{v, c_2(a)\}, \{v, c_6(a)\} \in E$
- $a = (v, u)$: $\{v, c_1(a)\}, \{v, c_5(a)\} \in E$
- Sonst: $\{v, c_3(a)\}, \{v, c_4(a)\} \in E$

Feedback Arc Set $\propto$ OSCM



$L = V$

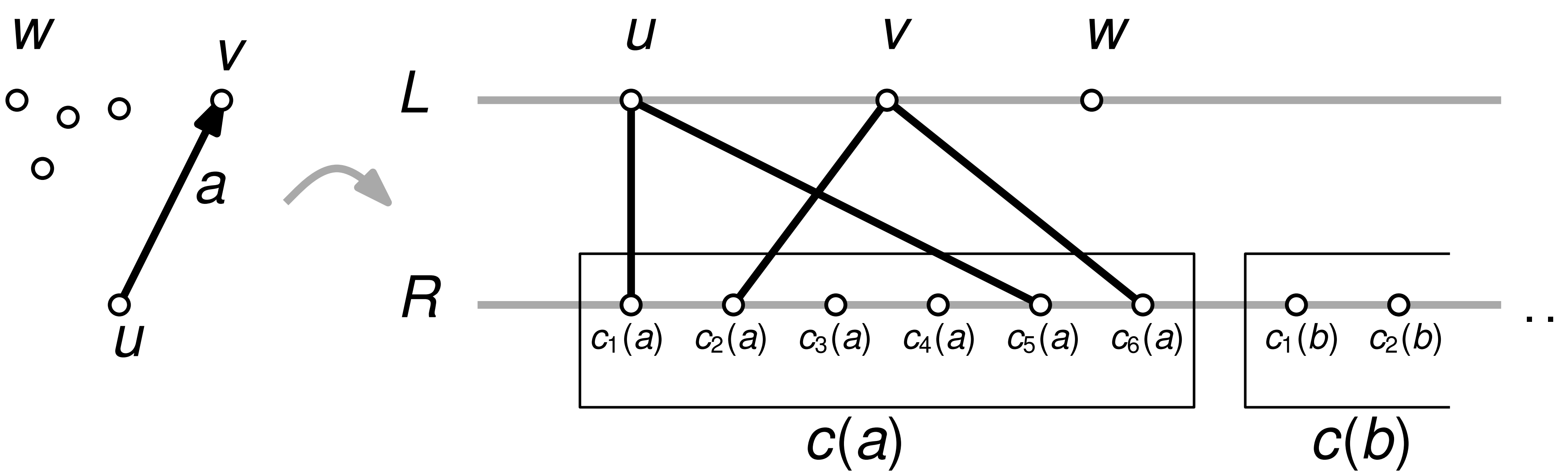
$$R = \bigcup_{a \in A} \{c_1(a), \ldots, c_6(a)\}$$

Für alle $v \in V$, $a \in A$:

- $a = (u, v)$: $\{v, c_2(a)\}, \{v, c_6(a)\} \in E$
- $a = (v, u)$: $\{v, c_1(a)\}, \{v, c_5(a)\} \in E$
- Sonst: $\{v, c_3(a)\}, \{v, c_4(a)\} \in E$

FEEDBACK ARC SET $\propto$ OSCM

$D = (V, A)$ $\qquad$ $B = (L, R, E)$



$L = V$

$$R = \bigcup_{a \in A} \{c_1(a), \ldots, c_6(a)\}$$

Für alle $v \in V$, $a \in A$:

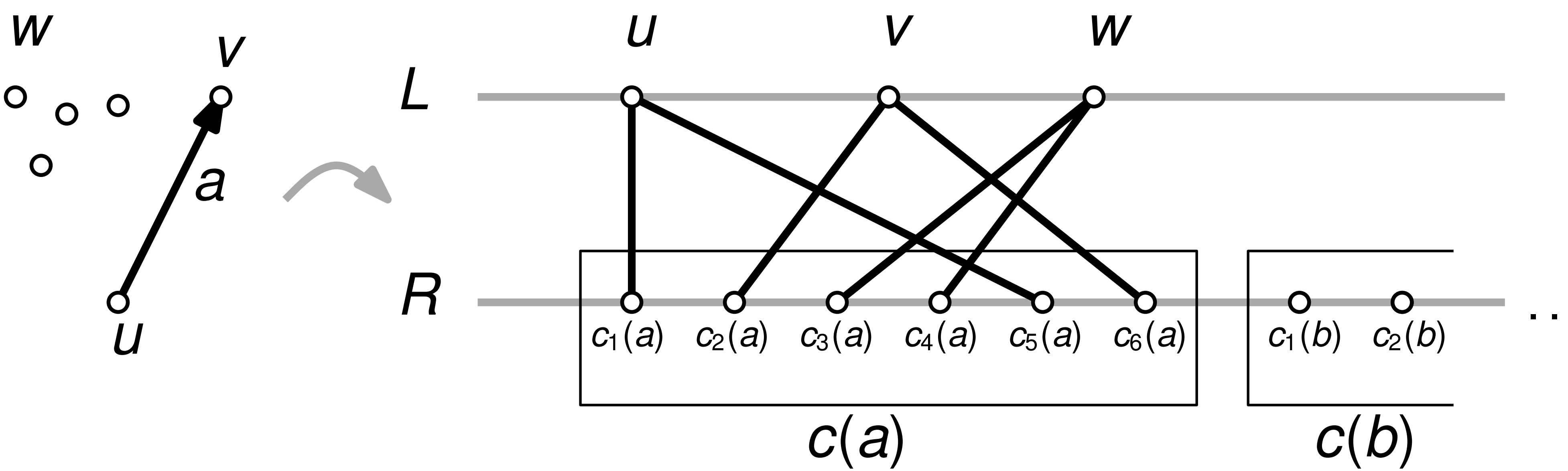
- $a = (u, v)$: $\{v, c_2(a)\}, \{v, c_6(a)\} \in E$
- $a = (v, u)$: $\{v, c_1(a)\}, \{v, c_5(a)\} \in E$
- Sonst: $\{v, c_3(a)\}, \{v, c_4(a)\} \in E$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

FEEDBACK ARC SET $\propto$ OSCM

Karlsruher Institut für Technologie

$D = (V, A)$ $\qquad$ $B = (L, R, E)$



$L = V$

$$R = \bigcup_{a \in A} \{c_1(a), \ldots, c_6(a)\}$$

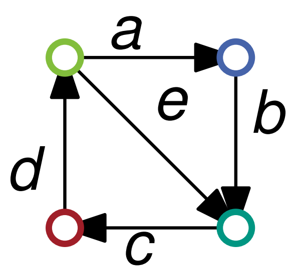
Für alle $v \in V$, $a \in A$:

- $a = (u, v)$: $\{v, c_2(a)\}, \{v, c_6(a)\} \in E$
- $a = (v, u)$: $\{v, c_1(a)\}, \{v, c_5(a)\} \in E$
- Sonst: $\{v, c_3(a)\}, \{v, c_4(a)\} \in E$

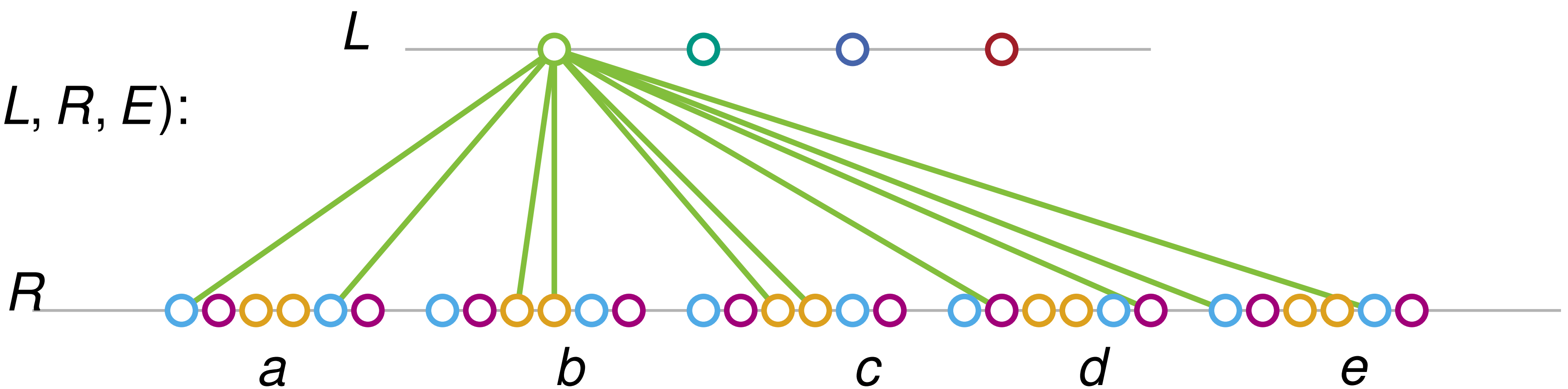
Beispiel

Karlsruher Institut für Technologie

$D = (V, A)$:

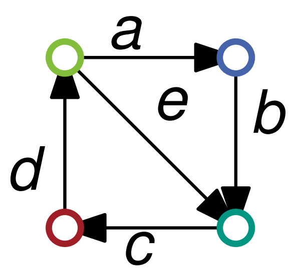


$B = (L, R, E)$:

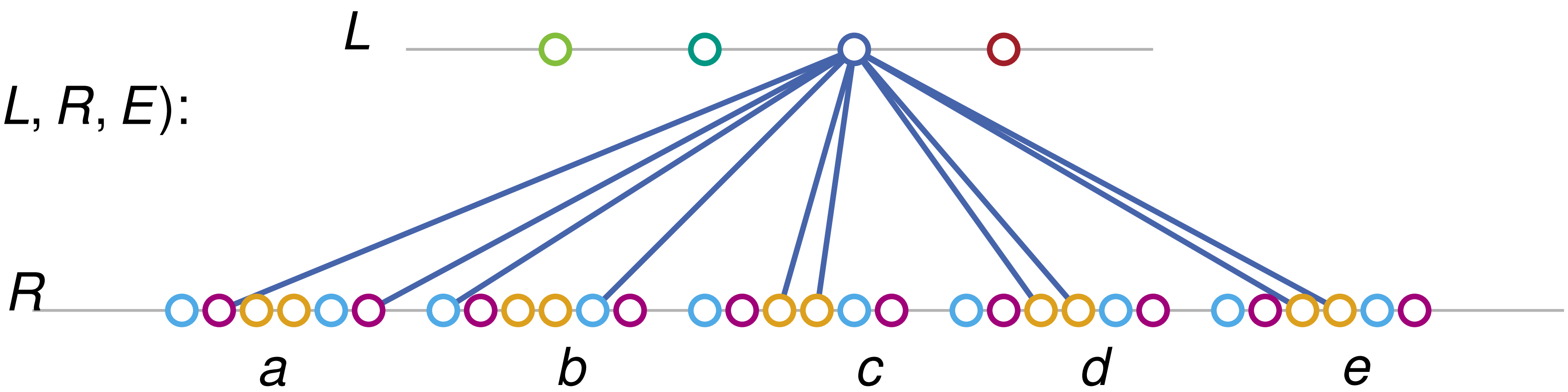


Beispiel

$D = (V, A)$:



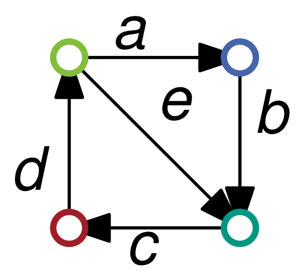
$B = (L, R, E)$:



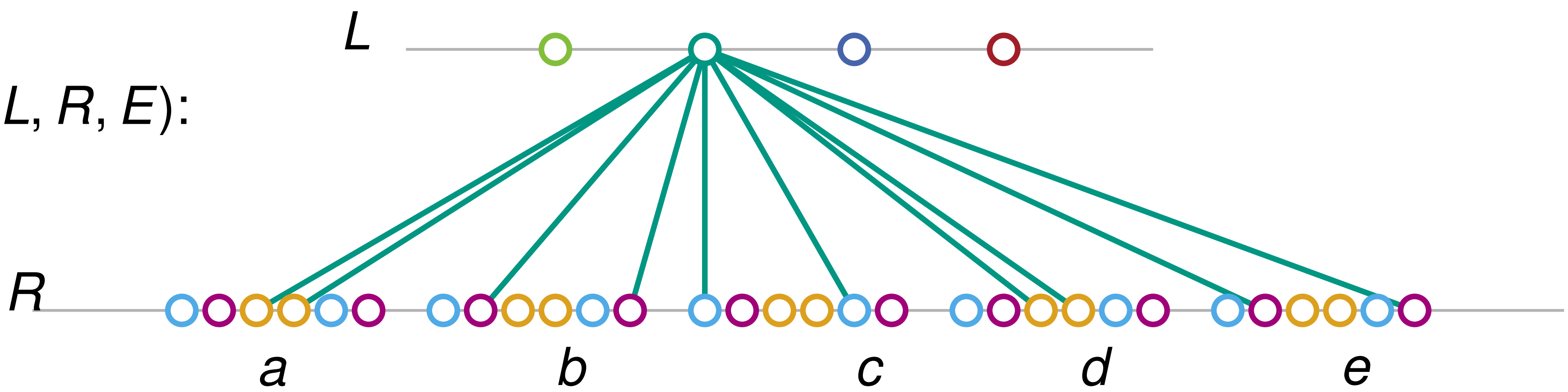
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Beispiel

$D = (V, A)$:

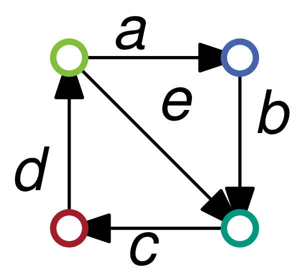


$B = (L, R, E)$:

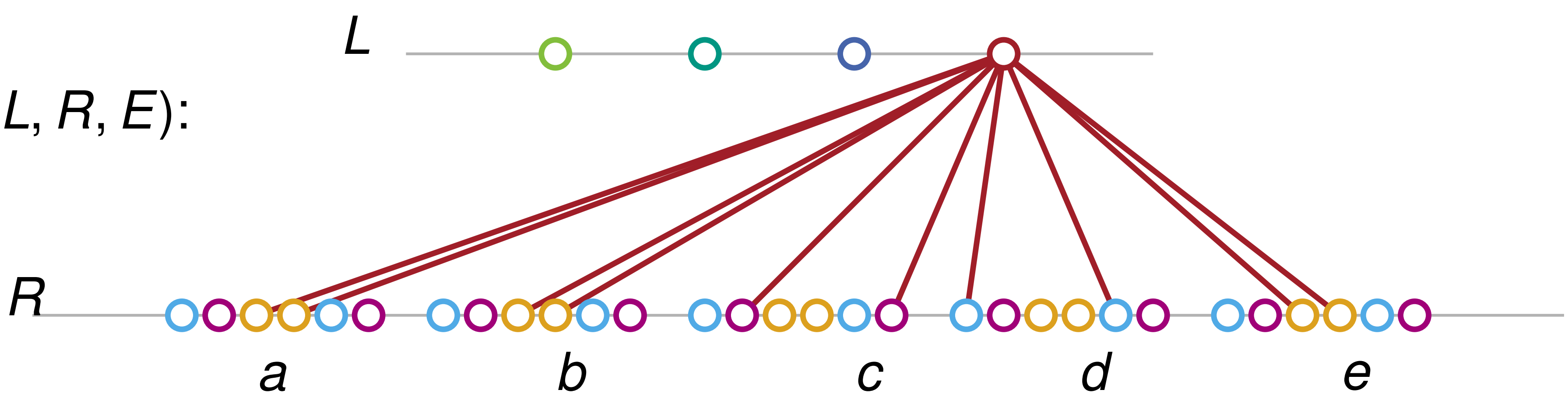


Beispiel

$D = (V, A)$:

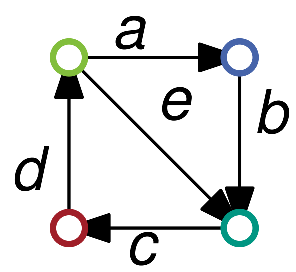


$B = (L, R, E)$:

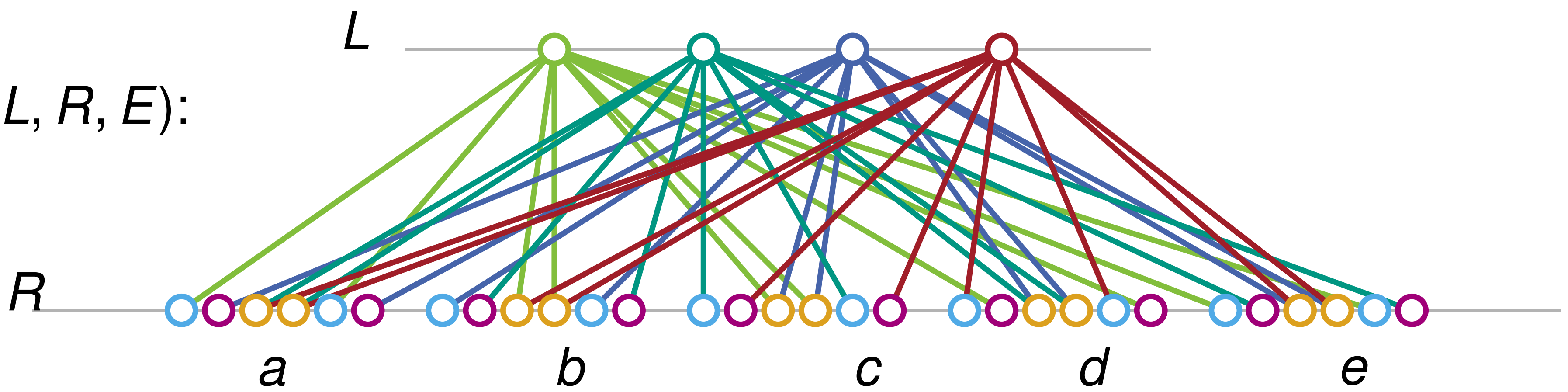


Beispiel

$D = (V, A)$:



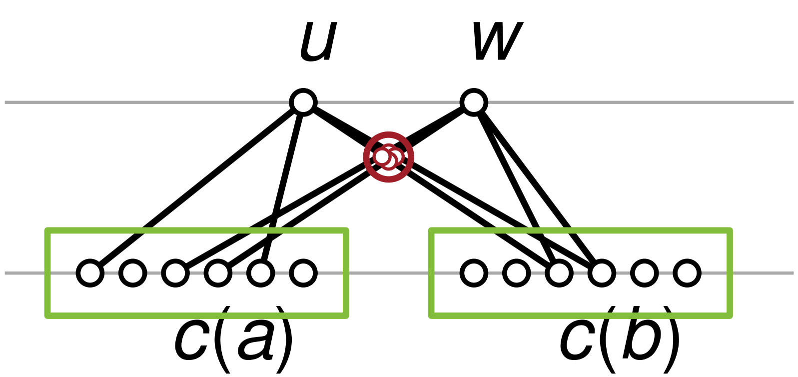
$B = (L, R, E)$:



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

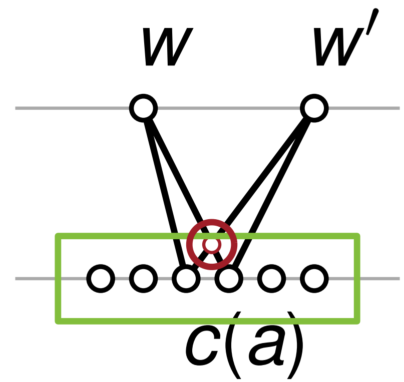
Kreuzungen zählen

$c(a)$ vs. $c(b)$:



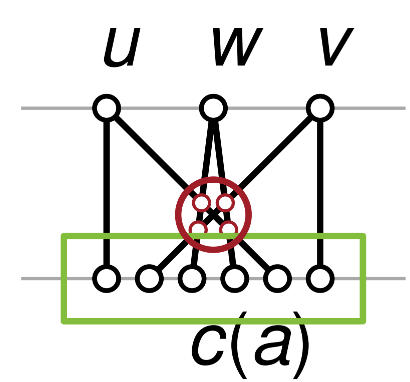
$$4 \cdot \binom{n}{2} \cdot \binom{m}{2}$$

w, w' nicht adjazent:



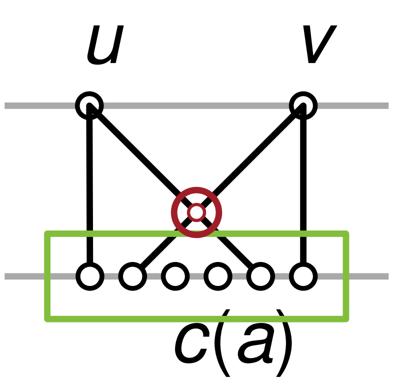
$$+m \cdot \binom{n-2}{2}$$

u, v adjazent, w nicht:



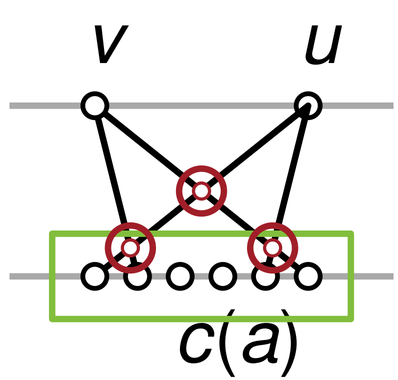
$$+4 \cdot m \cdot (n-2)$$

$a = (u, v), \ell(u) < \ell(v)$:



$$+(m-x)$$

$a = (u, v), \ell(v) < \ell(u)$:



$$+3x$$

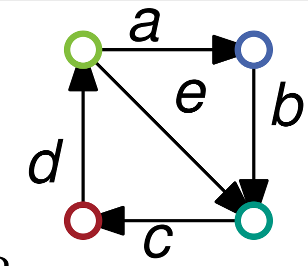
$n = |V| \quad m = |A|$

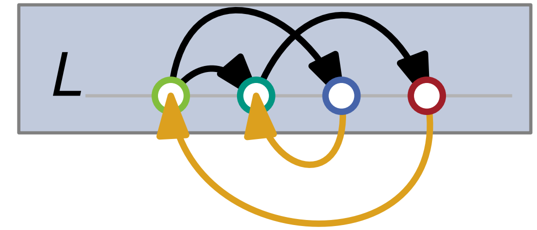
$x = |\{(u, v) \in A \mid \ell(u) > \ell(v)\}|$

$$M(x) = 4\binom{m}{2}\binom{n}{2} + m\binom{n-2}{2} + 4m(n-2) + m + 2x$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Korrektheit



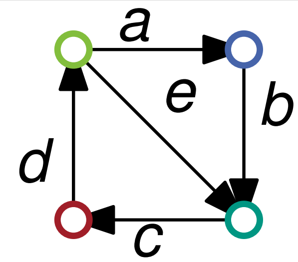


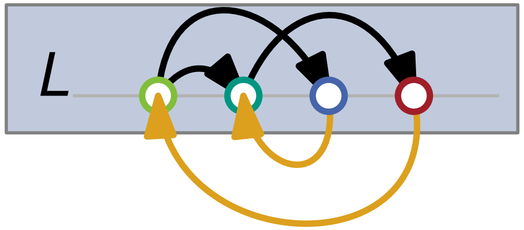
$$A_\ell := \{(u,v) \mid (u,v) \in A, \ell(u) > \ell(v)\} = \{b, d\}$$

$$M(x) = 4\binom{m}{2}\binom{n}{2} + m\binom{n-2}{2} + 4m(n-2) + m + 2x$$

Lemma: Sei ℓ eine Ordnung von L. Dann hat B genau $M(|A_\ell|)$ Kreuzungen.

Korrektheit





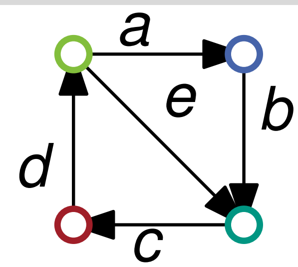
$A_\ell := \{(u,v) \mid (u,v) \in A, \ell(u) > \ell(v)\} = \{b, d\}$

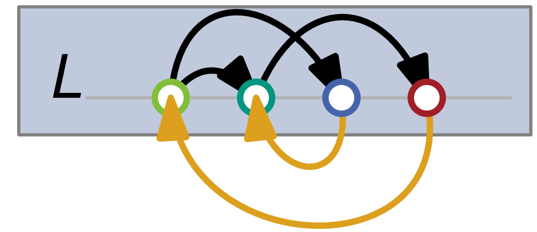
$M(x) = 4\binom{m}{2}\binom{n}{2} + m\binom{n-2}{2} + 4m(n-2) + m + 2x$

Lemma: Sei ℓ eine Ordnung von L. Dann hat B genau $M(|A_\ell|)$ Kreuzungen.

D hat FEEDBACK ARC SET der Größe $K \iff B$ hat $M(K)$ Kreuzungen

Korrektheit





$A_\ell := \{(u,v) \mid (u,v) \in A, \ell(u) > \ell(v)\} = \{b, d\}$

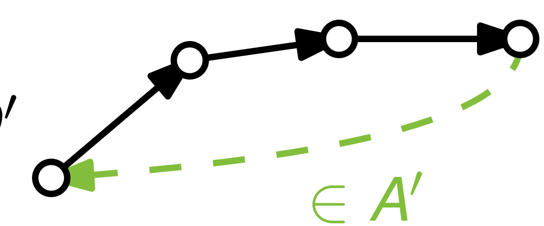
$M(x) = 4\binom{m}{2}\binom{n}{2} + m\binom{n-2}{2} + 4m(n-2) + m + 2x$

Lemma: Sei ℓ eine Ordnung von L. Dann hat B genau $M(|A_\ell|)$ Kreuzungen.

D hat Feedback Arc Set der Größe $K \iff B$ hat $M(K)$ Kreuzungen

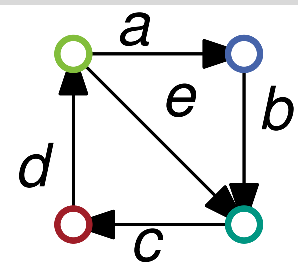
$\Rightarrow$:

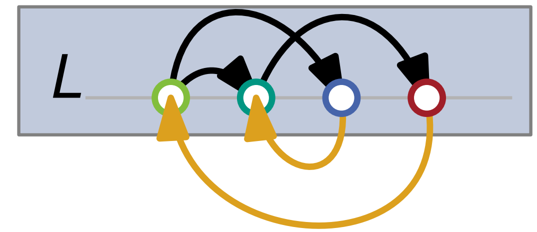
- Sei A' ein FAS von D der Größe K
- $D' = (V, A - A')$ azyklisch
- Sei ℓ eine topologische Sortierung von V bzgl. D'
- $A_\ell = A'$
- Nach Lemma hat B genau $M(K)$ Kreuzungen



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Korrektheit





$A_\ell := \{(u, v) \mid (u, v) \in A, \ell(u) > \ell(v)\} = \{b, d\}$

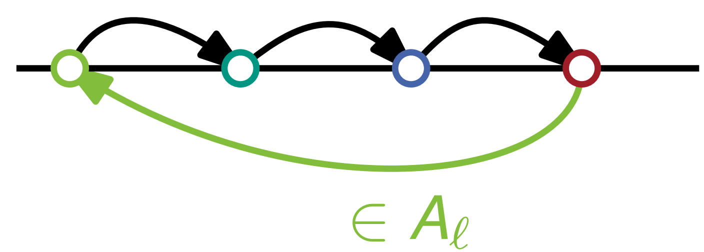
$M(x) = 4\binom{m}{2}\binom{n}{2} + m\binom{n-2}{2} + 4m(n-2) + m + 2x$

Lemma: Sei ℓ eine Ordnung von L. Dann hat B genau $M(|A_\ell|)$ Kreuzungen.

D hat FEEDBACK ARC SET der Größe $K \iff B$ hat $M(K)$ Kreuzungen

$\Leftarrow$:

- Sei ℓ eine Ordnung von L mit $M(K)$ Kreuzungen
- Dann hat A_ℓ Größe K (Lemma)
- $D' = (V, A - A_\ell)$ ist azyklisch
- A_ℓ ist FAS der Größe K



Median-Heuristik

Idee: Setze Koordinate auf Median der Nachbarn

- Für Knoten $v \in L$ mit Nachbarn $v_1, \ldots, v_k$ setze $\ell(v) = \text{med}(v) = r(v_{\lceil k/2 \rceil})$
- Falls $\ell(u) = \ell(v)$ mit ungleicher Gradparität, setze ungeraden Knoten nach links
- Falls $\ell(u) = \ell(v)$ mit gleicher Gradparität, setze beliebigen Knoten nach links
- Berechnung mit Linearzeit-Medianalgorithmus in $\mathcal{O}(|E|)$

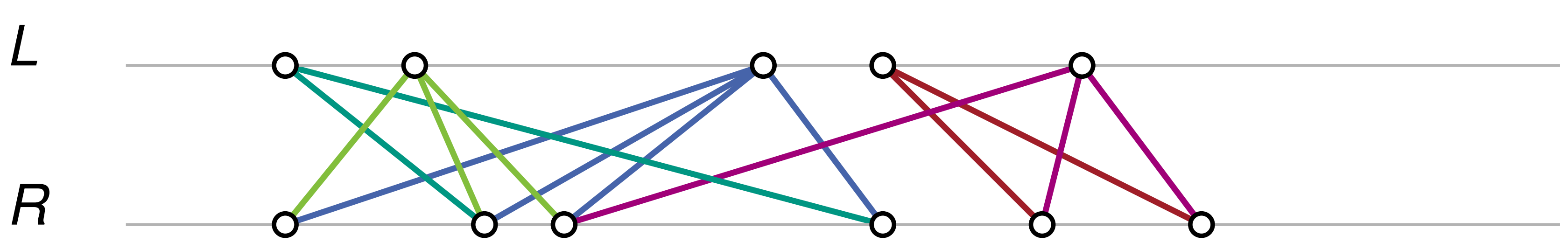
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Median-Heuristik

Karlsruher Institut für Technologie

Idee: Setze Koordinate auf Median der Nachbarn

- Für Knoten $v \in L$ mit Nachbarn $v_1, \ldots, v_k$ setze $\ell(v) = \mathrm{med}(v) = r(v_{\lceil k/2 \rceil})$
- Falls $\ell(u) = \ell(v)$ mit ungleicher Gradparität, setze ungeraden Knoten nach links
- Falls $\ell(u) = \ell(v)$ mit gleicher Gradparität, setze beliebigen Knoten nach links
- Berechnung mit Linearzeit-Medianalgorithmus in $\mathcal{O}(|E|)$

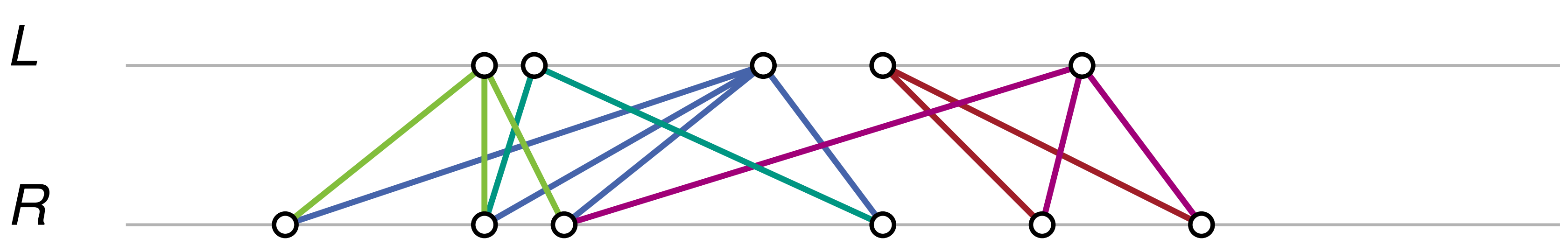


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Median-Heuristik

Idee: Setze Koordinate auf Median der Nachbarn

- Für Knoten $v \in L$ mit Nachbarn $v_1, \ldots, v_k$ setze $\ell(v) = \mathrm{med}(v) = r(v_{\lceil k/2 \rceil})$
- Falls $\ell(u) = \ell(v)$ mit ungleicher Gradparität, setze ungeraden Knoten nach links
- Falls $\ell(u) = \ell(v)$ mit gleicher Gradparität, setze beliebigen Knoten nach links
- Berechnung mit Linearzeit-Medianalgorithmus in $\mathcal{O}(|E|)$

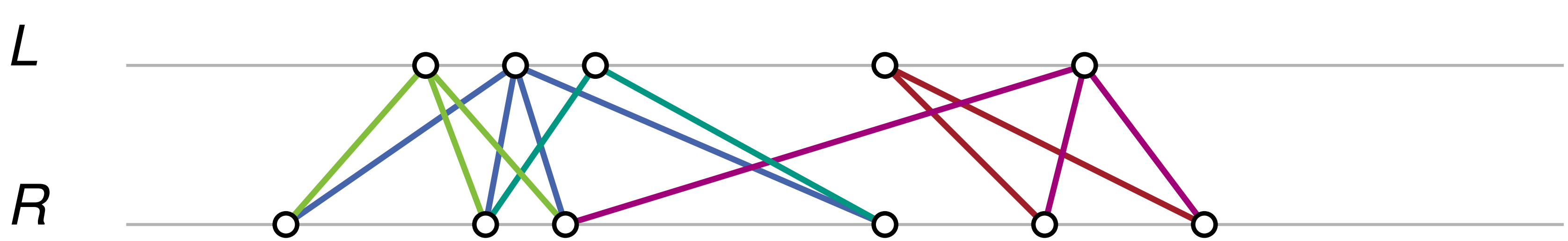


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Median-Heuristik

Idee: Setze Koordinate auf Median der Nachbarn

- Für Knoten $v \in L$ mit Nachbarn $v_1, \ldots, v_k$ setze $\ell(v) = \mathrm{med}(v) = r(v_{\lceil k/2 \rceil})$
- Falls $\ell(u) = \ell(v)$ mit ungleicher Gradparität, setze ungeraden Knoten nach links
- Falls $\ell(u) = \ell(v)$ mit gleicher Gradparität, setze beliebigen Knoten nach links
- Berechnung mit Linearzeit-Medianalgorithmus in $\mathcal{O}(|E|)$

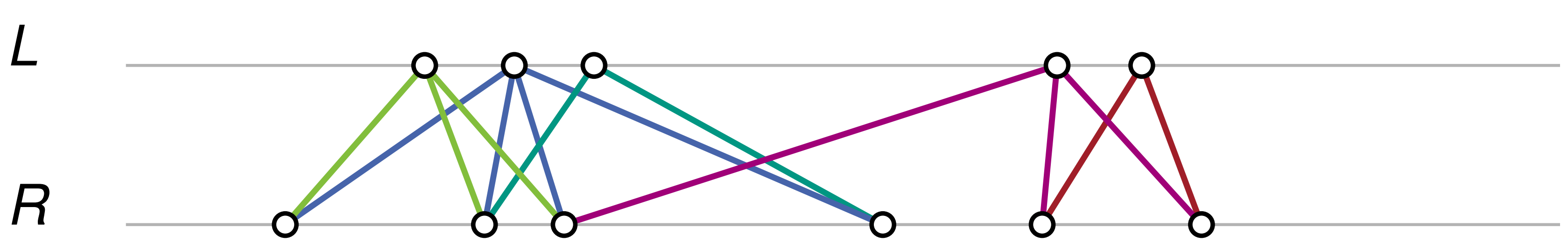


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Median-Heuristik

Idee: Setze Koordinate auf Median der Nachbarn

- Für Knoten $v \in L$ mit Nachbarn $v_1, \ldots, v_k$ setze $\ell(v) = \mathrm{med}(v) = r(v_{\lceil k/2 \rceil})$
- Falls $\ell(u) = \ell(v)$ mit ungleicher Gradparität, setze ungeraden Knoten nach links
- Falls $\ell(u) = \ell(v)$ mit gleicher Gradparität, setze beliebigen Knoten nach links
- Berechnung mit Linearzeit-Medianalgorithmus in $\mathcal{O}(|E|)$

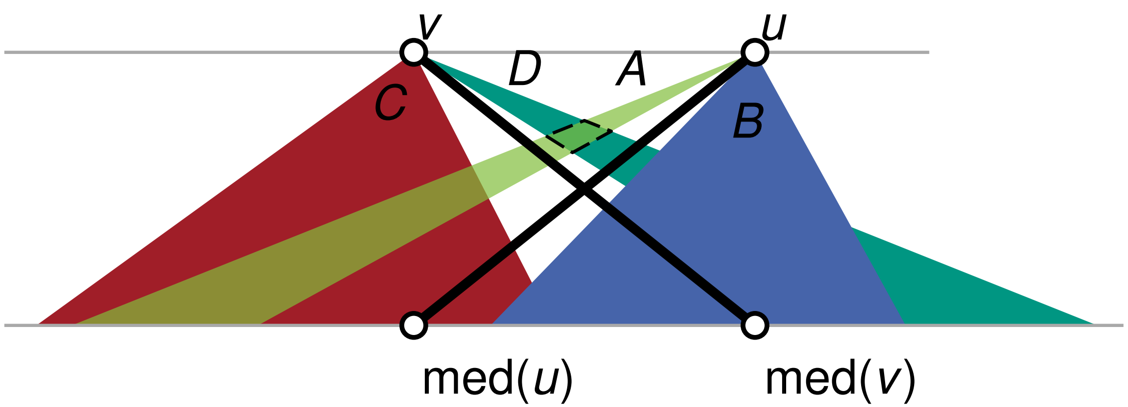


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

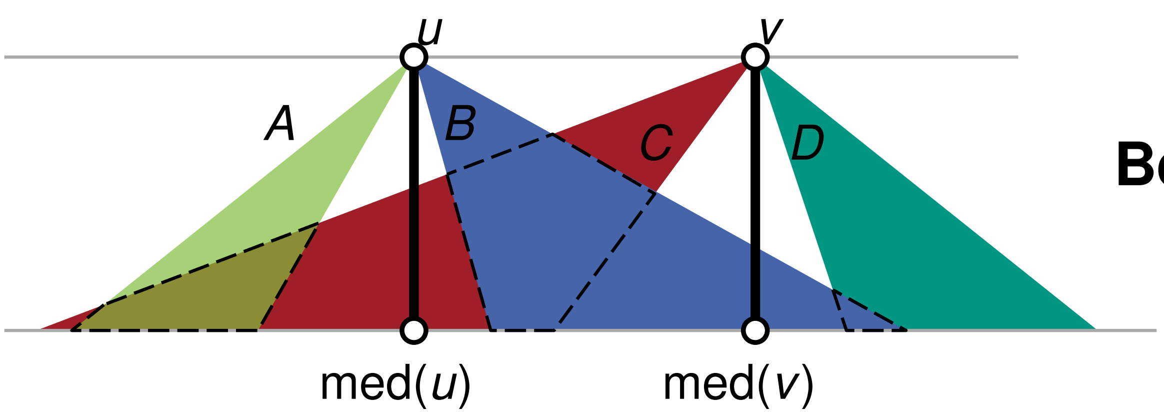
Approximation

Satz 2: Sei $G = (L, R, E)$ ein bipartiter Graph und r eine beliebige Ordnung von R. Dann gilt $\mathrm{med}(G, r) \leq 3\,\mathrm{opt}(G, r)$.

$a = |A|, b = |B|, c = |C|, d = |D|$



$$c_{vu} \geq ad + a + d$$



$$c_{uv} \leq ac + bc + bd + c + b$$

Beobachtung: $b \leq a + 1, c \leq d$

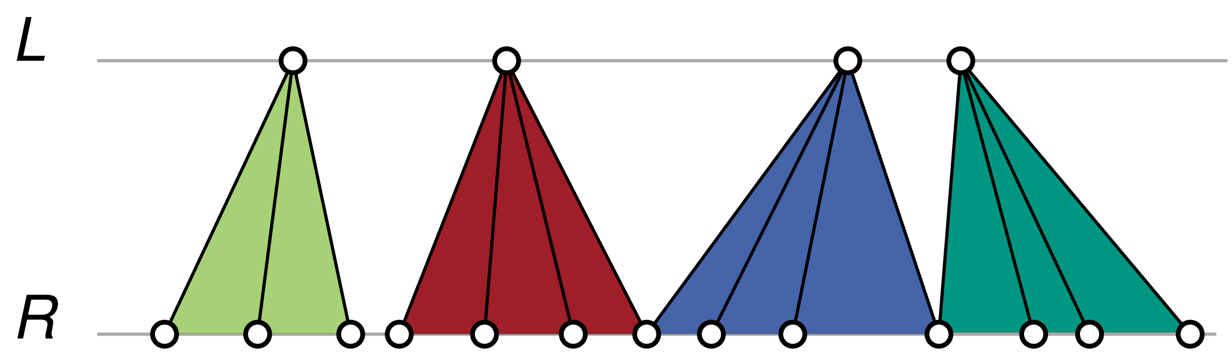
$$\Rightarrow c_{uv} \leq 3ad + 3d + a + 1$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Kreuzungsfrei

Satz 3: Sei $G = (L, R, E)$ ein bipartiter Graph und r eine beliebige Ordnung von R mit $\mathrm{opt}(G, r) = 0$. Dann gilt $\mathrm{med}(G, r) = 0$.

Beobachtung: Nachbarschaften der Knoten aus L bilden (fast) disjunkte Intervalle bzgl. r



Reduktionsschemas

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Reduktionsschemas

Beschränkung eines Problems

- offensichtliche 1-zu-1-Beziehung
- 3SAT $\propto$ 4SAT

Lokales Ersetzen

- Veränderung der lokalen Struktur
- weitestgehend unabhängige Veränderung
- X3C $\propto$ PIT

Komponenten-Design

- Modellierung von Interaktion zwischen Komponenten
- Modelliere z.B. Literale und Klauseln
- SAT $\propto$ ORTHOGONALEZEICHNUNG

Beschränkung

Problem 3SAT:
Gegeben: Menge U von Variablen, Menge C_3 von Klauseln über U, wobei jede Klausel genau **drei** Literale enthält
Frage: Existiert eine Wahrheitsbelegung, die alle Klauseln erfüllt?

Problem 4SAT:
Gegeben: Menge U von Variablen, Menge C_4 von Klauseln über U, wobei jede Klausel genau **vier** Literale enthält
Frage: Existiert eine Wahrheitsbelegung, die alle Klauseln erfüllt?

Beschränkung

Problem 3SAT:
Gegeben: Menge U von Variablen, Menge C_3 von Klauseln über U, wobei jede Klausel genau **drei** Literale enthält
Frage: Existiert eine Wahrheitsbelegung, die alle Klauseln erfüllt?

Problem 4SAT:
Gegeben: Menge U von Variablen, Menge C_4 von Klauseln über U, wobei jede Klausel genau **vier** Literale enthält
Frage: Existiert eine Wahrheitsbelegung, die alle Klauseln erfüllt?

Transformation 3SAT $\propto$ 4SAT

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Beschränkung

Problem 3SAT:
Gegeben: Menge U von Variablen, Menge C_3 von Klauseln über U, wobei jede Klausel genau **drei** Literale enthält
Frage: Existiert eine Wahrheitsbelegung, die alle Klauseln erfüllt?

Problem 4SAT:
Gegeben: Menge U von Variablen, Menge C_4 von Klauseln über U, wobei jede Klausel genau **vier** Literale enthält
Frage: Existiert eine Wahrheitsbelegung, die alle Klauseln erfüllt?

Transformation 3SAT $\propto$ 4SAT

Möglichkeit 1: C_3 zweimal kopieren und geschickt verbinden.

Sei (C_3, U) eine 3SAT-Instanz

$$(C_4 = \{(a \vee b \vee c \vee d) \wedge (a \vee b \vee c \vee \neg d) \mid a \vee b \vee c \in C_3, U \cup \{d\})$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Beschränkung

Problem 3SAT:
Gegeben: Menge U von Variablen, Menge C_3 von Klauseln über U, wobei jede Klausel genau **drei** Literale enthält
Frage: Existiert eine Wahrheitsbelegung, die alle Klauseln erfüllt?

Problem 4SAT:
Gegeben: Menge U von Variablen, Menge C_4 von Klauseln über U, wobei jede Klausel genau **vier** Literale enthält
Frage: Existiert eine Wahrheitsbelegung, die alle Klauseln erfüllt?

Transformation 3SAT $\propto$ 4SAT

Möglichkeit 2: Gadgetkonstruktion: erzwinge $\varphi(x_1) = \texttt{falsch}$.

Sei (C_3, U) eine 3SAT-Instanz

$$(C_4 = \{(a \vee b \vee c \vee x_1) \mid a \vee b \vee c \in C_3, U \cup \{x_1, x_2, x_3, x_4\})$$
$$\bigcup \{(\neg x_1, x_2, x_3, x_4) \wedge (\neg x_1, \neg x_2, x_3, x_4) \wedge \ldots \wedge (\neg x_1, \neg x_2, \neg x_3, \neg x_4)\}$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

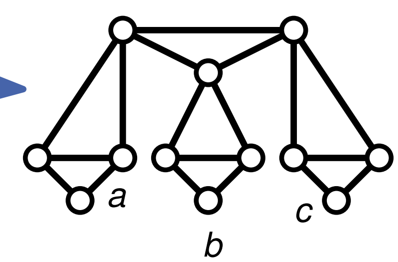
Lokale Ersetzung

Problem **EXACT COVER BY 3-SETS (X3C):**
Gegeben: Endliche Menge X mit $|X| = 3q$ und Menge C von 3-elementigen Teilmengen von X.
Frage: Existiert eine exakte Überdeckung von X mit Mengen aus C?

Problem **PARTITION INTO TRIANGLES (PIT):**
Gegeben: Ungerichteter Graph $G = (V, E)$ mit $|V| = 3q$.
Frage: Gibt es eine Partitionierung $V_1 \cup V_2 \cup \cdots \cup V_k$ von V, sodass jedes V_i ein Dreieck in G induziert?

Transformation X3C $\propto$ PIT

$$C = \{\{a, b, c\} \qquad\qquad\qquad\qquad \}$$



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

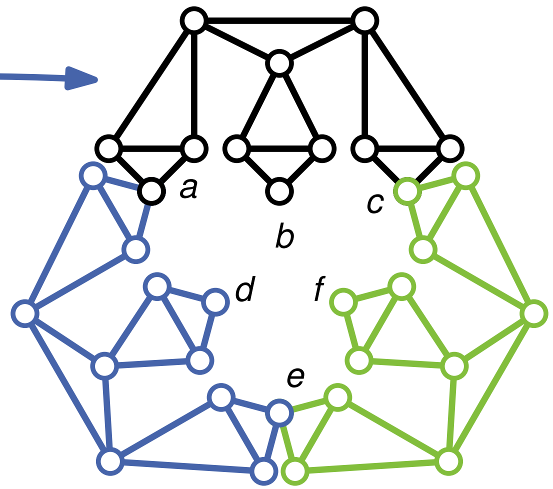
Lokale Ersetzung

Problem Exact Cover by 3-Sets (X3C):
Gegeben: Endliche Menge X mit $|X| = 3q$ und Menge C von 3-elementigen Teilmengen von X.
Frage: Existiert eine exakte Überdeckung von X mit Mengen aus C?

Problem Partition into Triangles (PIT):
Gegeben: Ungerichteter Graph $G = (V, E)$ mit $|V| = 3q$.
Frage: Gibt es eine Partitionierung $V_1 \cup V_2 \cup \cdots \cup V_k$ von V, sodass jedes V_i ein Dreieck in G induziert?

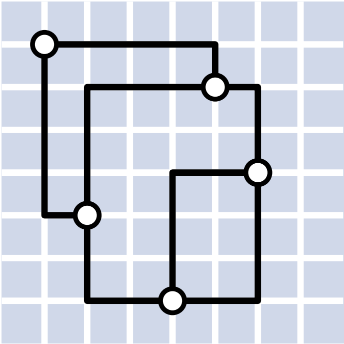
Transformation X3C $\propto$ PIT

$C = \{\{a, b, c\}, \{a, d, e\}, \{c, f, e\}\}$



Komponenten-Design

Problem ORTHOGONALEZEICHNUNG:
Gegeben: Ungerichteter planarer Graph $G = (V, E)$, $K \in \mathbb{N}$.
Frage: Existiert eine orthogonale Zeichnung von G auf einem Gitter der Größe K?

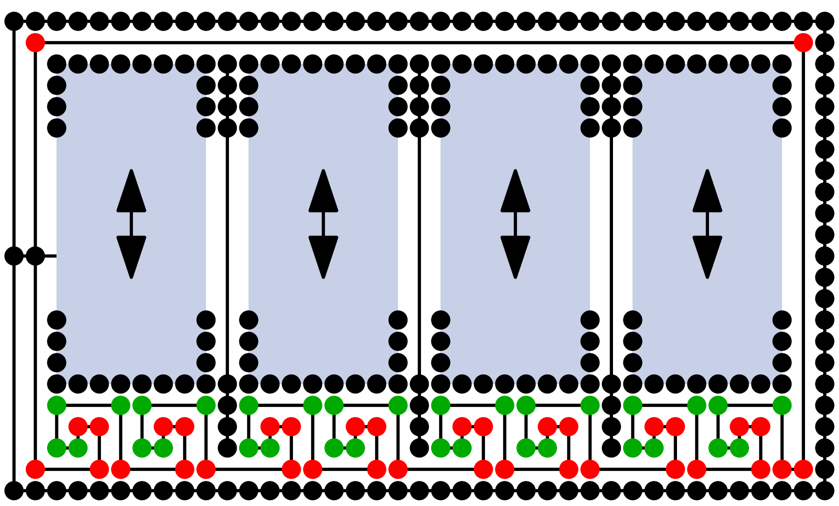


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Komponenten-Design

Problem **ORTHOGONALEZEICHNUNG:**
Gegeben: Ungerichteter planarer Graph $G = (V, E)$, $K \in \mathbb{N}$.
Frage: Existiert eine orthogonale Zeichnung von G auf einem Gitter der Größe K?

Transformation SAT $\propto$ ORTHOGONALEZEICHNUNG

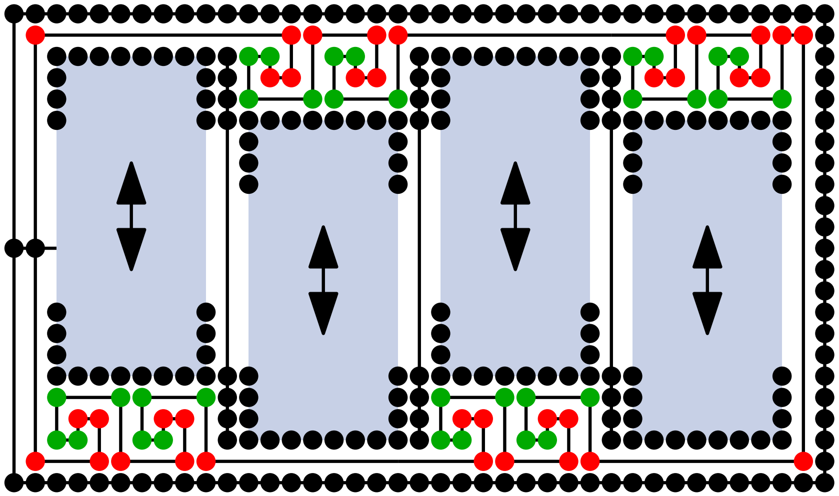


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Komponenten-Design

Problem ORTHOGONALEZEICHNUNG:
Gegeben: Ungerichteter planarer Graph $G = (V, E)$, $K \in \mathbb{N}$.
Frage: Existiert eine orthogonale Zeichnung von G auf einem Gitter der Größe K?

Transformation SAT $\propto$ ORTHOGONALEZEICHNUNG

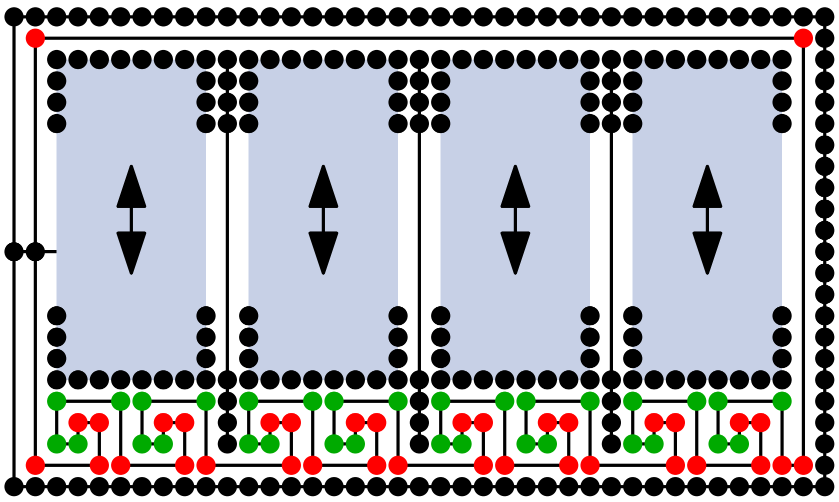


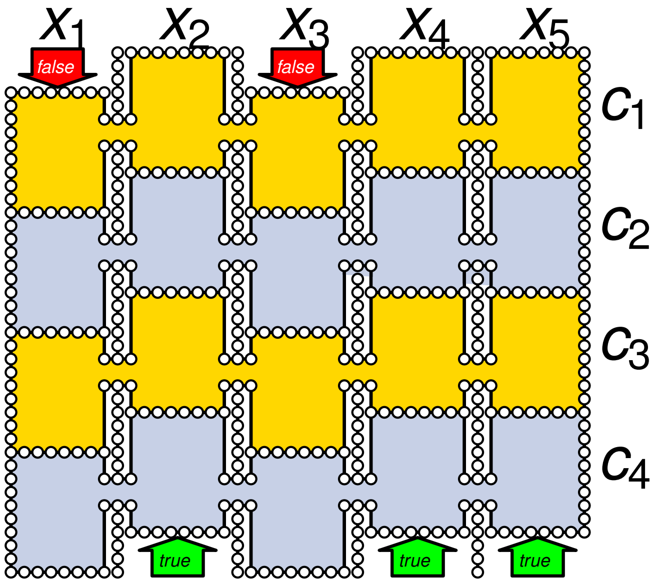
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Komponenten-Design

Problem ORTHOGONALEZEICHNUNG:
Gegeben: Ungerichteter planarer Graph $G = (V, E)$, $K \in \mathbb{N}$.
Frage: Existiert eine orthogonale Zeichnung von G auf einem Gitter der Größe K?

Transformation SAT $\propto$ ORTHOGONALEZEICHNUNG



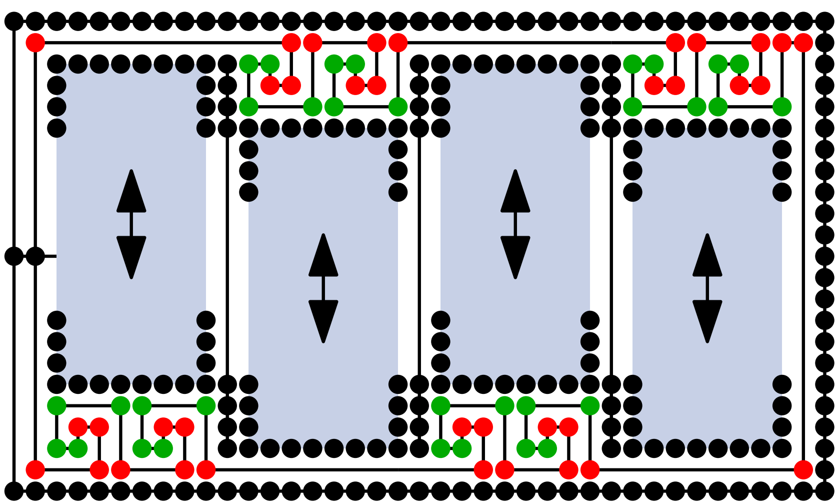


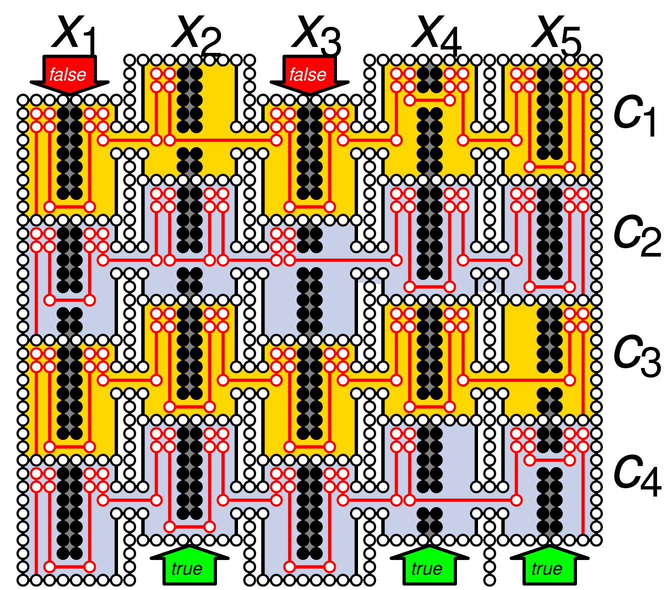
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Komponenten-Design

Problem ORTHOGONALEZEICHNUNG:
Gegeben: Ungerichteter planarer Graph $G = (V, E)$, $K \in \mathbb{N}$.
Frage: Existiert eine orthogonale Zeichnung von G auf einem Gitter der Größe K?

Transformation SAT $\propto$ ORTHOGONALEZEICHNUNG



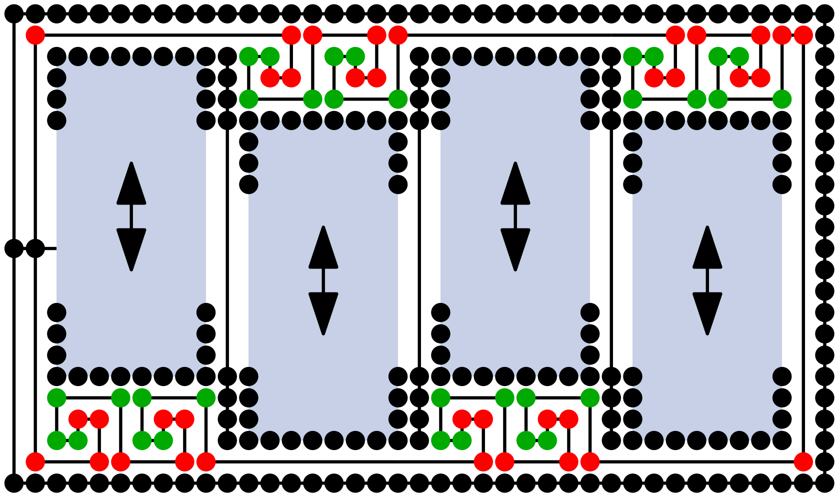


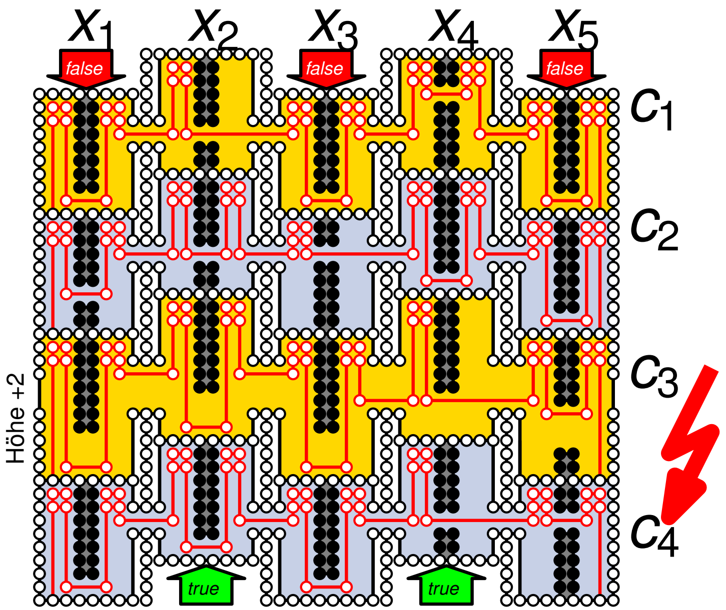
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Komponenten-Design

Problem ORTHOGONALEZEICHNUNG:
Gegeben: Ungerichteter planarer Graph $G = (V, E)$, $K \in \mathbb{N}$.
Frage: Existiert eine orthogonale Zeichnung von G auf einem Gitter der Größe K?

Transformation SAT $\propto$ ORTHOGONALEZEICHNUNG





Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Starke NP-Vollständigkeit

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Starke NP-Vollständigkeit

- Wir sind immer davon ausgegangen, dass Zahlen binär kodiert werden
- Zahlen können exponentiell groß in der Größe der Eingabe sein
- **Intuition:** Dadurch kodiert man eventuell Komplexität in die Zahlen

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Starke NP-Vollständigkeit

- Wir sind immer davon ausgegangen, dass Zahlen binär kodiert werden
- Zahlen können exponentiell groß in der Größe der Eingabe sein
- **Intuition:** Dadurch kodiert man eventuell Komplexität in die Zahlen

Definition: Ein NP-vollständiges Problem ist *schwach* NP-*vollständig*, wenn es bei *unärer* Kodierung aller Zahlen in Polynomialzeit lösbar ist.

Beispiele: KNAPSACK, SUBSETSUM.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Starke NP-Vollständigkeit

- Wir sind immer davon ausgegangen, dass Zahlen binär kodiert werden
- Zahlen können exponentiell groß in der Größe der Eingabe sein
- **Intuition:** Dadurch kodiert man eventuell Komplexität in die Zahlen

Definition: Ein NP-vollständiges Problem ist *schwach NP-vollständig*, wenn es bei *unärer* Kodierung aller Zahlen in Polynomialzeit lösbar ist.

Beispiele: KNAPSACK, SUBSETSUM.

Ein solcher Algorithmus heißt *pseudopolynomiell*.

Starke NP-Vollständigkeit

- Wir sind immer davon ausgegangen, dass Zahlen binär kodiert werden
- Zahlen können exponentiell groß in der Größe der Eingabe sein
- **Intuition:** Dadurch kodiert man eventuell Komplexität in die Zahlen

Definition: Ein Problem ist *stark* NP*-vollständig*, wenn es bei *unärer* Kodierung aller Zahlen NP-vollständig bleibt.

Beispiele: SAT, k-COLORING, TRAVELINGSALESMAN, BINPACKING.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Starke NP-Vollständigkeit

- Wir sind immer davon ausgegangen, dass Zahlen binär kodiert werden
- Zahlen können exponentiell groß in der Größe der Eingabe sein
- **Intuition:** Dadurch kodiert man eventuell Komplexität in die Zahlen

Definition: Ein Problem ist *stark* NP*-vollständig*, wenn es bei *unärer* Kodierung aller Zahlen NP-vollständig bleibt.

Beispiele: SAT, k-COLORING, TRAVELINGSALESMAN, BINPACKING.

Um zu zeigen, dass ein Problem NP-vollständig ist, ist es egal, ob das Problem, von dem man reduziert, schwach oder stark NP-vollständig ist.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Klausurhinweise

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Klausur

19.03., Dienstag

Letzte Möglichkeit zur Anmeldung

22.03., Freitag

Bekanntgabe der Hörsaalverteilung:

- **Rechtzeitig** überprüfen, ob man auf der Liste steht
- Hörsaal herausfinden, in dem man schreibt

25.03., Montag
Klausur

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Klausur

19.03., Dienstag — Letzte Möglichkeit zur Anmeldung

22.03., Freitag — Bekanntgabe der Hörsaalverteilung:

- **Rechtzeitig** überprüfen, ob man auf der Liste steht
- Hörsaal herausfinden, in dem man schreibt

25.03., Montag
Klausur

Ablauf der Klausur:

- Klausur beginnt **pünktlich** um 12:00 Uhr
- Bitte **frühzeitig** im **zugewiesenen** Hörsaal einfinden
- Bearbeitungszeit: 2 Stunden

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Klausur

19.03., Dienstag — Letzte Möglichkeit zur Anmeldung

22.03., Freitag — Bekanntgabe der Hörsaalverteilung:

- **Rechtzeitig** überprüfen, ob man auf der Liste steht
- Hörsaal herausfinden, in dem man schreibt

25.03., Montag
Klausur

Nicht vergessen:

- Studentenausweis
- Stifte
- Personalausweis!

Ablauf der Klausur:

- Klausur beginnt **pünktlich** um 12:00 Uhr
- Bitte **frühzeitig** im **zugewiesenen** Hörsaal einfinden
- Bearbeitungszeit: 2 Stunden

Klausur

19.03., Dienstag — Letzte Möglichkeit zur Anmeldung

22.03., Freitag — Bekanntgabe der Hörsaalverteilung:
- **Rechtzeitig** überprüfen, ob man auf der Liste steht
- Hörsaal herausfinden, in dem man schreibt

25.03., Montag
Klausur

Nicht vergessen:
- Studentenausweis
- Stifte
- Personalausweis!

Ablauf der Klausur:
- Klausur beginnt **pünktlich** um 12:00 Uhr
- Bitte **frühzeitig** im **zugewiesenen** Hörsaal ei
- Bearbeitungszeit: 2 Stunden

Keine Hilfsmittel!

Klausur

19.03., Dienstag — Letzte Möglichkeit zur Anmeldung

22.03., Freitag — Bekanntgabe der Hörsaalverteilung:

- **Rechtzeitig** überprüfen, ob man auf der Liste steht
- Hörsaal herausfinden, in dem man schreibt

Abmeldung:

- Bis 24.03.19: über Studierendenportal
- Am 25.03.19: Nur noch **direkt vor** der Klausur im entsprechenden Hörsaal (Studentenausweis mitbringen!)

Ablauf der Klausur:

- Klausur beginnt **pünktlich** um 12:00 Uhr
- Bitte **frühzeitig** im **zugewiesenen** Hörsaal einfinden
- Bearbeitungszeit: 2 Stunden

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Klausur

Inhalt der Klausur: Stoff der Vorlesung und Übung (ohne Einschränkung)

Vorsicht: In den Tutorien werden teilweise andere Verfahren vorgestellt.

Allgemeine Tipps zur Klausur:

- Hinweise beachten!
- Mit Aufgaben anfangen, die man beherrscht
- Falls Ankreuzaufgaben gestellt werden:
  - Ausreichend Zeit nehmen
  - Auf Formulierungen achten
  - Hilfreich: Gegenbeispiele, Proberechnungen und Argumente

Leicht vermeidbare Fehler

Aufgabenstellung:
Geben Sie einen (Keller-)Automaten,
eine Turingmaschine, eine Grammatik an . . .

Definition der einzelnen Bestandteile nicht vergessen!!!

- Angabe als explizite Liste oder als Tupel
- Wichtig ist, dass klar wird, wie zum Beispiel bei Grammatiken die Terminal- und Variablenmenge aussieht: Die Angabe der Produktionen alleine genügt nicht!!!
- Wenn **vollständiger** Automat verlangt: Fehlerzustand nicht vergessen

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Leicht vermeidbare Fehler

Karlsruher Institut für Technologie

Aufgabenstellung:
Geben Sie einen (Keller-)Automaten,
eine Turingmaschine, eine Grammatik an . . .

Definition der einzelnen Bestandteile nicht vergessen!!!

- Angabe als explizite Liste oder als Tupel
- Wichtig ist, dass klar wird, wie zum Beispiel bei Grammatiken die Terminal- und Variablenmenge aussieht: Die Angabe der Produktionen alleine genügt nicht!!!
- Wenn **vollständiger** Automat verlangt: Fehlerzustand nicht vergessen

Aufgabenstellung:
- Zeigen Sie, dass . . .
- Zeigen oder widerlegen Sie, dass . . .

Schreiben Sie, was Sie zeigen möchten: ”Die Aussage gilt / gilt nicht”.

Antwort ausreichend begründen!

Leicht vermeidbare Fehler

Aufgabenstellung:
Geben Sie einen (Keller-)Automaten,
eine Turingmaschine, eine Grammatik an . . .

Definition der einzelnen Bestandteile nicht vergessen!!!

- Angabe als explizite Liste oder als Tupel
- Wichtig ist, dass klar wird, wie zum Beispiel bei Grammatiken die Terminal- und Variablenmenge aussieht: Die Angabe der Produktionen alleine genügt nicht!!!
- Wenn **vollständiger** Automat verlangt: Fehlerzustand nicht vergessen

Aufgabenstellung:
- Zeigen Sie, dass . . .
- Zeigen oder widerlegen Sie, dass . . .

Schreiben Sie, was Sie zeigen möchten: ”Die Aussage gilt / gilt nicht”.

Antwort ausreichend begründen!

Aufgabenstellung:
- Berechnen Sie mithilfe des Verfahrens aus der Vorlesung . . .

Schritte der Berechnung dokumentieren!

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Leicht vermeidbare Fehler

Aufgabenstellung:

Beweisen Sie, dass . . .

Wenn nicht anders verlangt, alles Notwendige beweisen!

Beispiel: Beweise, dass Problem NP-vollständig ist.

Nicht nur zeigen, dass Problem NP-schwer ist, sondern auch, dass es in NP liegt.

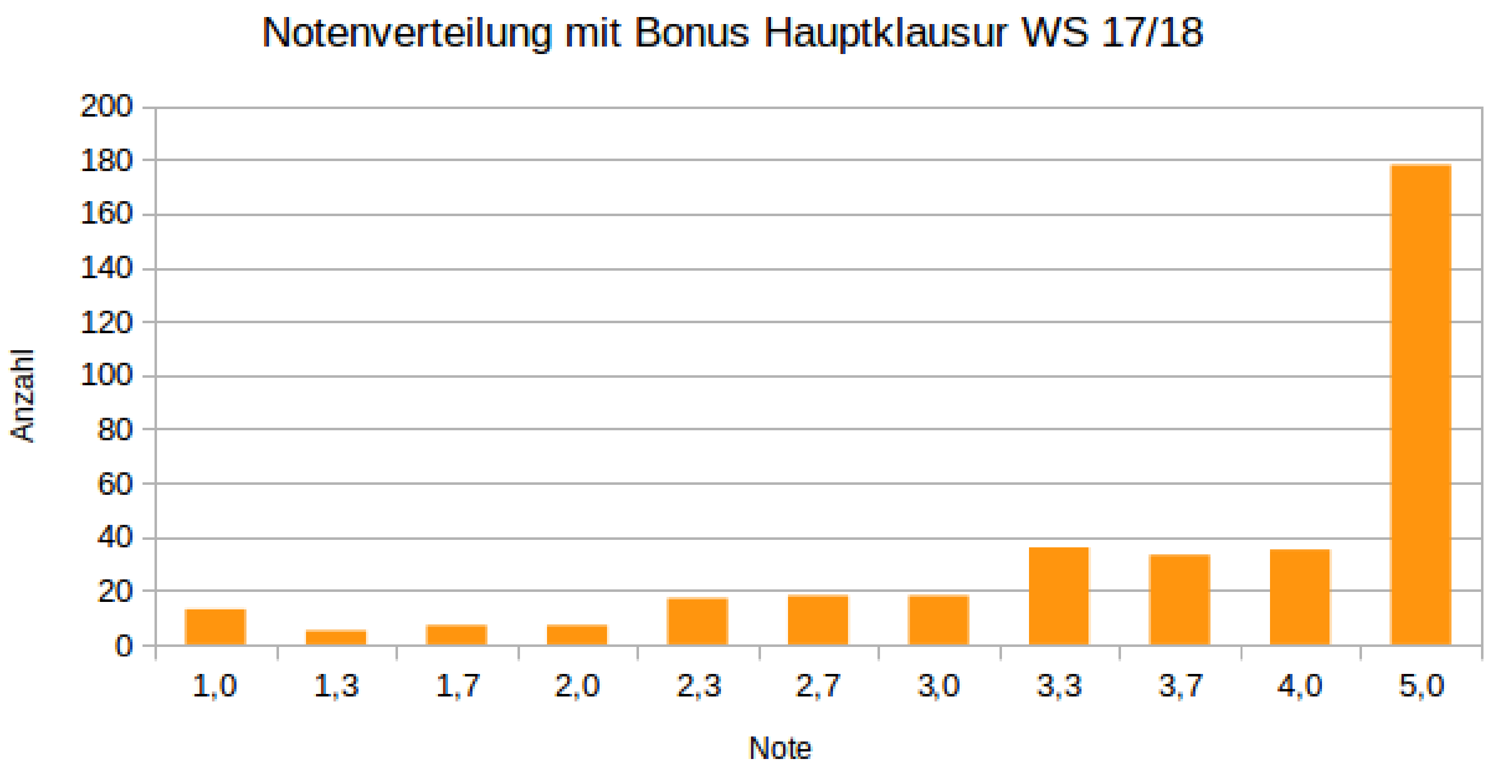
- Ist beim Automaten ein Fehlerzustand implizit gegeben?
- Akzeptiert der Kellerautomat durch leeren Stack oder durch akzeptierenden Zustand?
- Wird gerade $L \in$ NP gezeigt, oder dass L NP-schwer ist?
- . . .

Antwort ausreichend erklären!

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

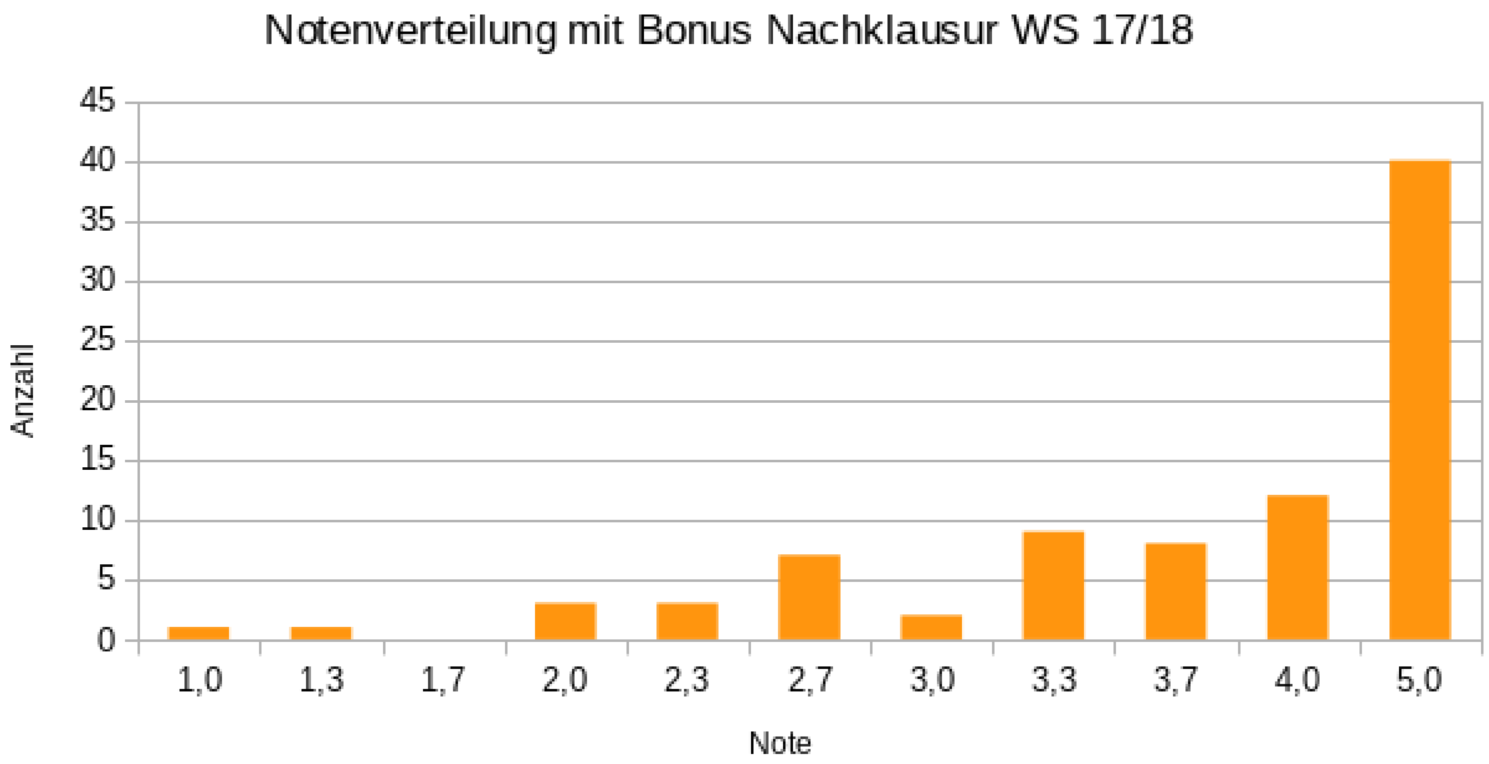
Leicht vermeidbare Fehler

- zu wenig lernen



Leicht vermeidbare Fehler

- zu wenig lernen



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Viel Erfolg bei der Klausur!

Institut für Theoretische Informatik
Lehrstuhl Algorithmik