Theoretische Grundlagen der Informatik Übung

9. Übungstermin · 6. Februar 2018

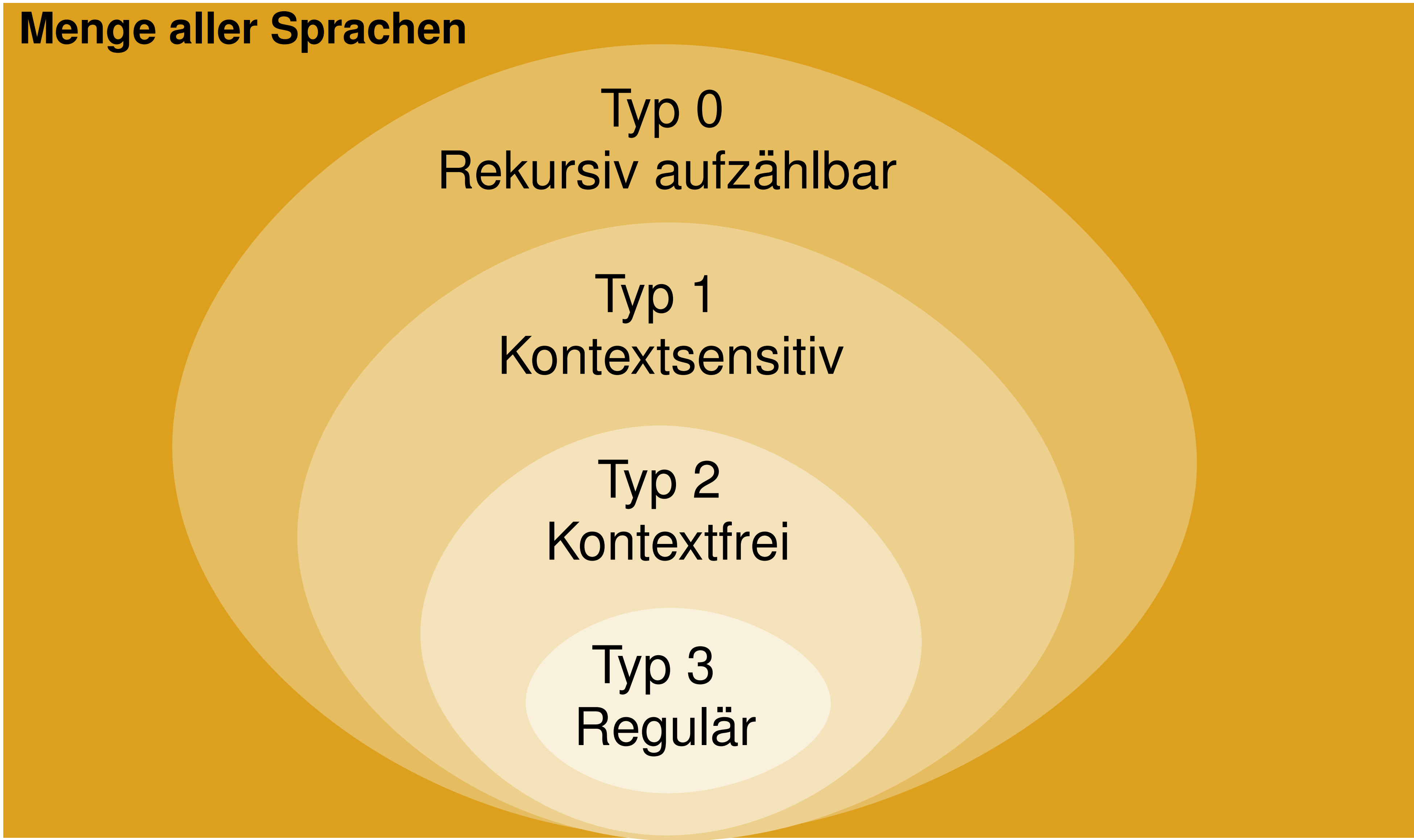
Guido Brückner

Institut für Theoretische Informatik · Lehrstuhl Algorithmik

Übersicht

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Chomsky-Hierarchien



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Einordnung

Typ	Beschreibungsmittel
0	
1	
2	
3	

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Einordnung

Typ	Beschreibungsmittel
0	■ TM ■ 'Typ-0 Grammatik'
1	
2	
3	

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Einordnung

Karlsruher Institut für Technologie

Typ	Beschreibungsmittel
0	■ TM ■ 'Typ-0 Grammatik'
1	■ linear beschränkte TM (LBA) ■ kontextsensitive Grammatik
2	
3	

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Einordnung

Typ	Beschreibungsmittel
0	■ TM ■ 'Typ-0 Grammatik'
1	■ linear beschränkte TM (LBA) ■ kontextsensitive Grammatik
2	■ PDA ■ kontextfreie Grammatik
3	

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Einordnung

Typ	Beschreibungsmittel
0	■ TM ■ ’Typ-0 Grammatik’
1	■ linear beschränkte TM (LBA) ■ kontextsensitive Grammatik
2	■ PDA ■ kontextfreie Grammatik
3	■ DEA ■ reguläre Grammatik ■ NEA ■ regulärer Ausdruck

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Einordnung

Typ	Beschreibungsmittel
0	■ TM ■ ’Typ-0 Grammatik’
1	■ linear beschränkte TM (LBA) ■ kontextsensitive Grammatik
2	■ PDA ■ kontextfreie Grammatik
3	■ DEA ■ NEA ■ reguläre Grammatik ■ regulärer Ausdruck

DPDA?

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Einordnung

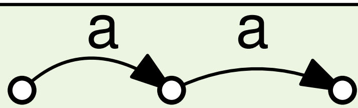
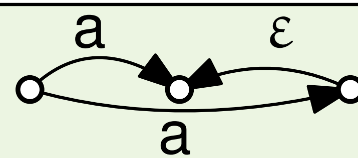
Karlsruher Institut für Technologie

Typ	Beschreibungsmittel
0	■ TM ■ ’Typ-0 Grammatik’
1	■ linear beschränkte TM (LBA) ■ kontextsensitive Grammatik
2	■ PDA ■ kontextfreie Grammatik
deterministisch Kontextfreie Sprachen	
3	■ DEA ■ NEA ■ reguläre Grammatik ■ regulärer Ausdruck

DPDA?

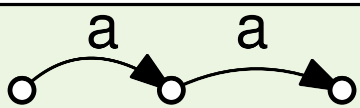
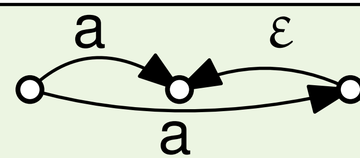
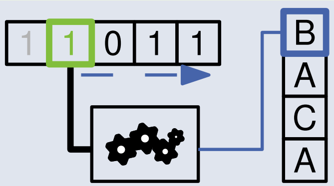
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Maschinenmodelle

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

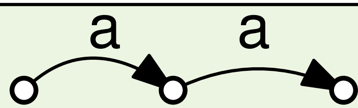
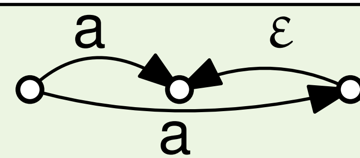
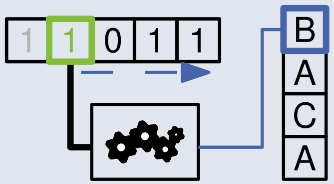
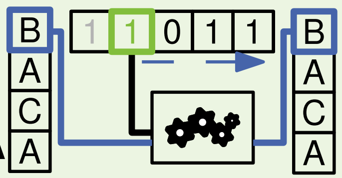
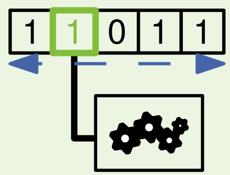
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Maschinenmodelle

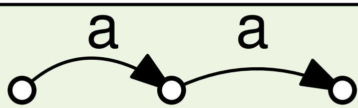
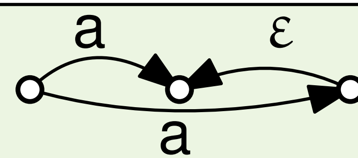
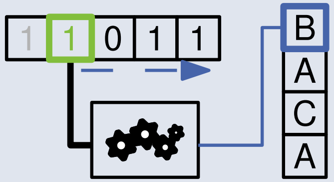
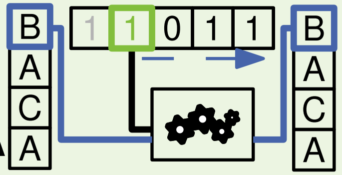
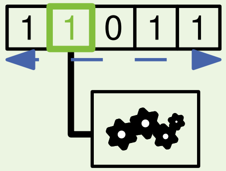
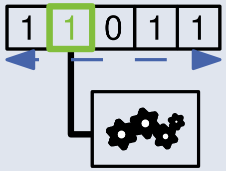
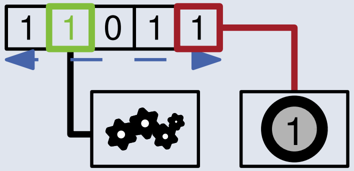
		gleich mächtig?	Poly. Trafo.?
DEA 	NEA 	✓	✗
DPDA	PDA 	✗	

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

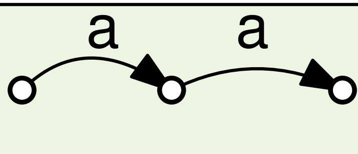
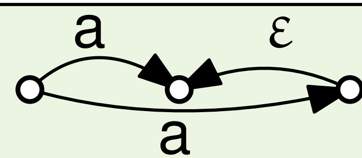
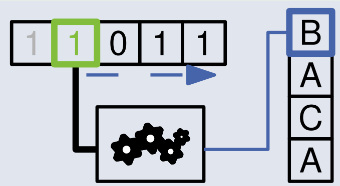
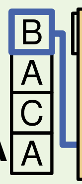
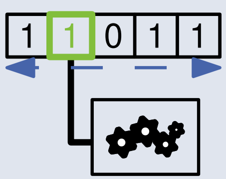
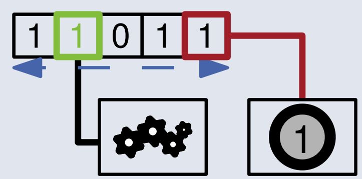
Maschinenmodelle

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

Maschinenmodelle

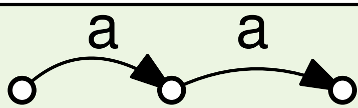
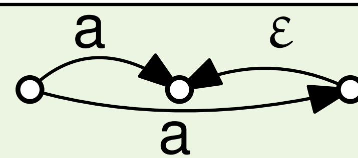
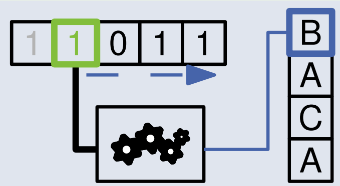
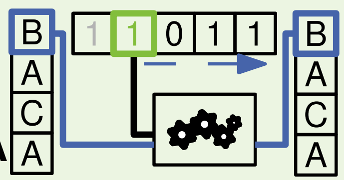
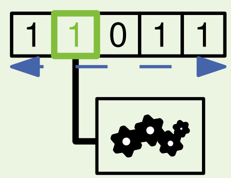
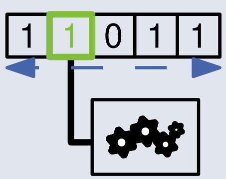
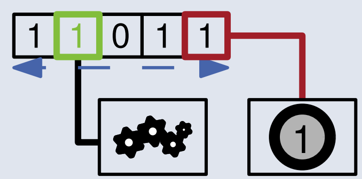
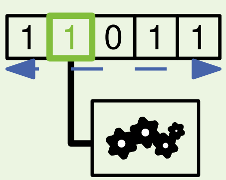
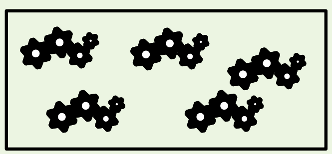
		gleich mächtig?	Poly. Trafo.?
DEA 	NEA 	✓	✗
DPDA	PDA 	✗	
2-DPDA 	DTM 	✓	✓
DTM 	NTM 	✓	✓

Maschinenmodelle

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

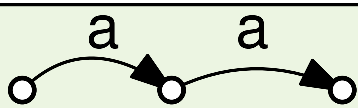
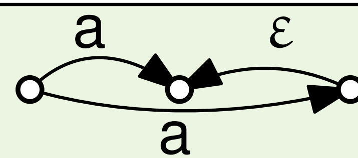
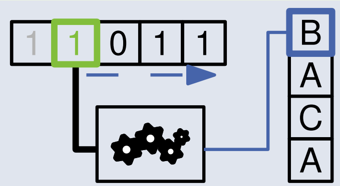
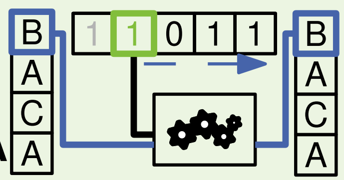
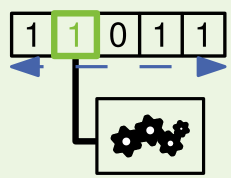
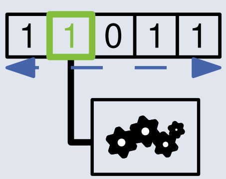
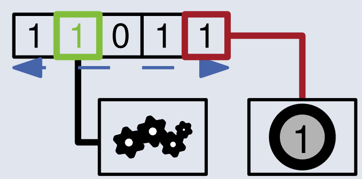
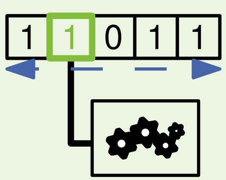
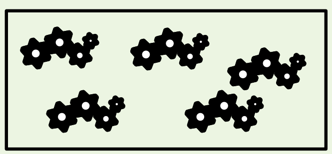
Achtung: Das hat nichts mit $P \stackrel{?}{=} NP$ zu tun!

Maschinenmodelle

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

Maschinenmodelle

Karlsruher Institut für Technologie

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

Mehrband TM, Mehrspur TM, RAM,. . .

Maschinenmodelle

Karlsruher Institut für Technologie

		gleich mächtig?	Poly. Trafo.?
DEA	NEA	✓	✗
DPDA		✗	
2-DPDA			✓
DTM			✓
DTM	DIE Ma-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	✓ $\mathcal{NP}$-schwer	✗	✗	✗
2	✓ CNF: $O(n^3)$	✓	✗	✗
3	✓ O(n)	✓	✓	✓

Typ-1 Grammatik

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

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

Werden die gleichen 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 $\mathbb{Y}$: $\alpha \to \beta, |\alpha| \leq |\beta|, \alpha \in V^+, \beta \in ((V \cup \Sigma) \setminus S)^+$

Werden die gleichen Sprachen generiert?

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

$\mathcal{L}(\mathbb{Y}) \subset \mathcal{L}(\mathbb{X})$: $\quad X_1 X_2 \dots X_n \to Y_1 \dots Y_m, m > 1$

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

Werden die gleichen Sprachen generiert?

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

$\mathcal{L}(\mathbb{Y}) \subset \mathcal{L}(\mathbb{X})$: $X_1 X_2 \dots X_n \to Y_1 \dots Y_m, m > 1$

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

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

Werden die gleichen Sprachen generiert?

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

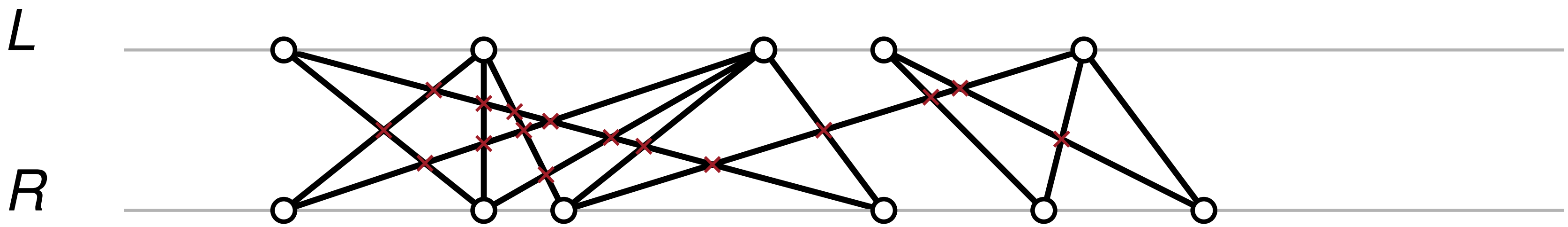
$\mathcal{L}(\mathbb{Y}) \subset \mathcal{L}(\mathbb{X})$: $X_1 X_2 \ldots X_n \to Y_1 \ldots Y_m, m > 1$

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

Kreuzungsminimierung

Einseitige Kreuzungsminimierung (OSCM)

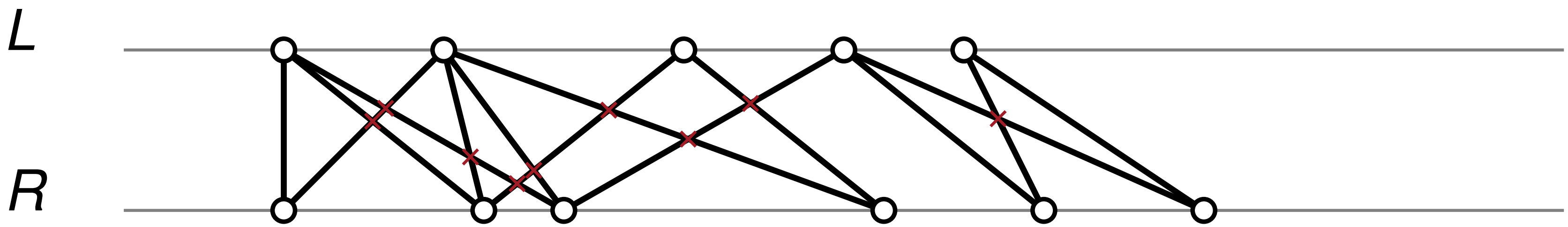
Geg.: Bipartiter Graph $G = (L, R, E)$ und Knotenordnung r von R



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

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

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

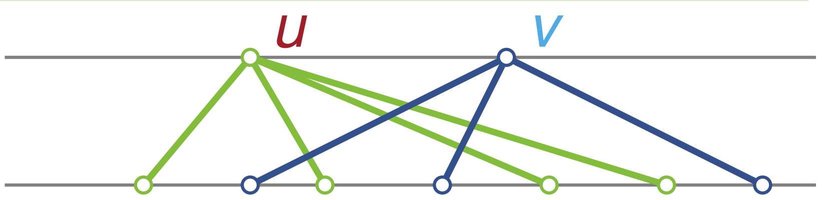
Ges.: Knotenordnung l von L, so dass die Anzahl Kreuzungen von Kanten in E minimal ist

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 $l(u) < l(v)$ oder $l(v) < l(u)$ ab

Def: Kreuzungszahl für $l(u) < l(v)$

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



$c_{uv} = 5$

$c_{vu} = 7$

Einseitige Kreuzungsminimierung (OSCM)

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

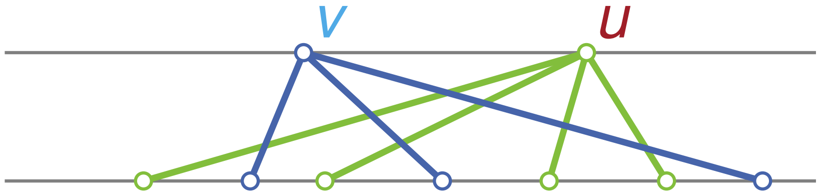
Ges.: Knotenordnung l von L, so dass die Anzahl Kreuzungen von Kanten in E minimal ist

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 $l(u) < l(v)$ oder $l(v) < l(u)$ ab

Def: Kreuzungszahl für $l(u) < l(v)$

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



$c_{uv} = 5$

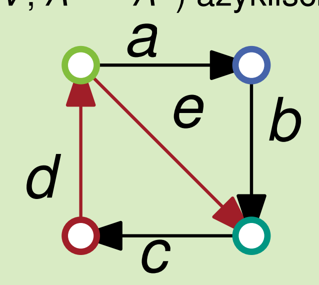
$c_{vu} = 7$

Transformation

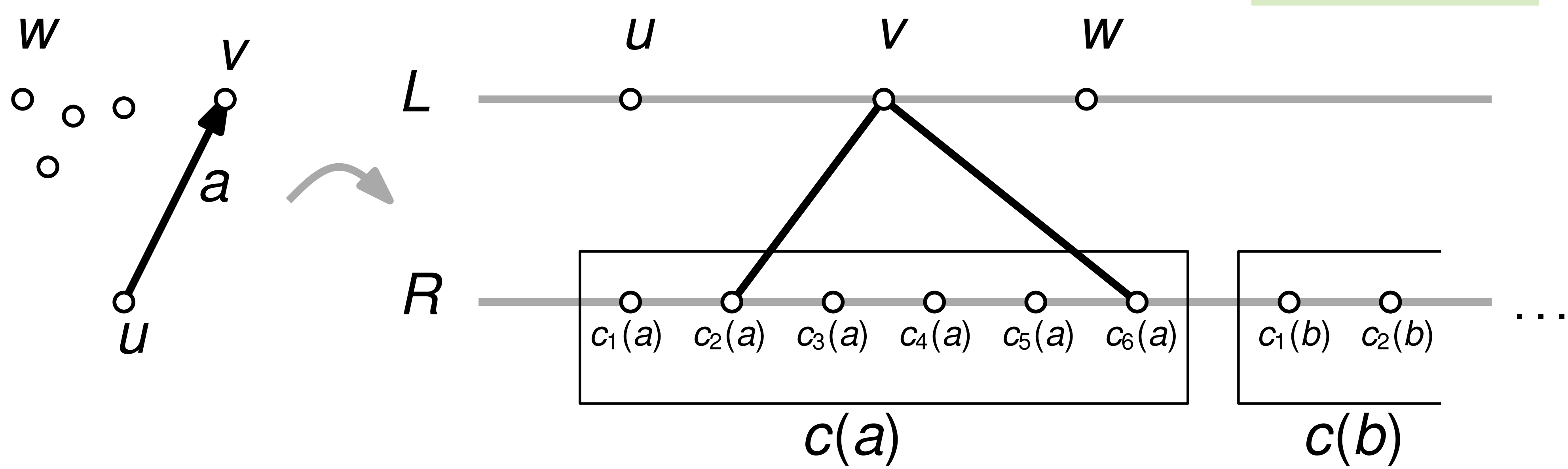
Satz: Das einseitige Kreuzungsminimierungsproblem (OSCM) ist $\mathcal{NP}$-schwer.

Reduktion: FEEDBACK ARC SET

A' ist FAS gdw $D = (V, A - A')$ azyklisch



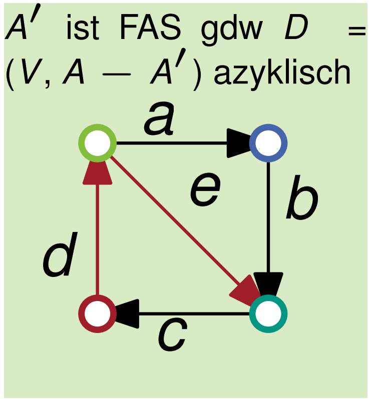
$D = (V, A)$ $\qquad$ $B = (W, E)$



Transformation

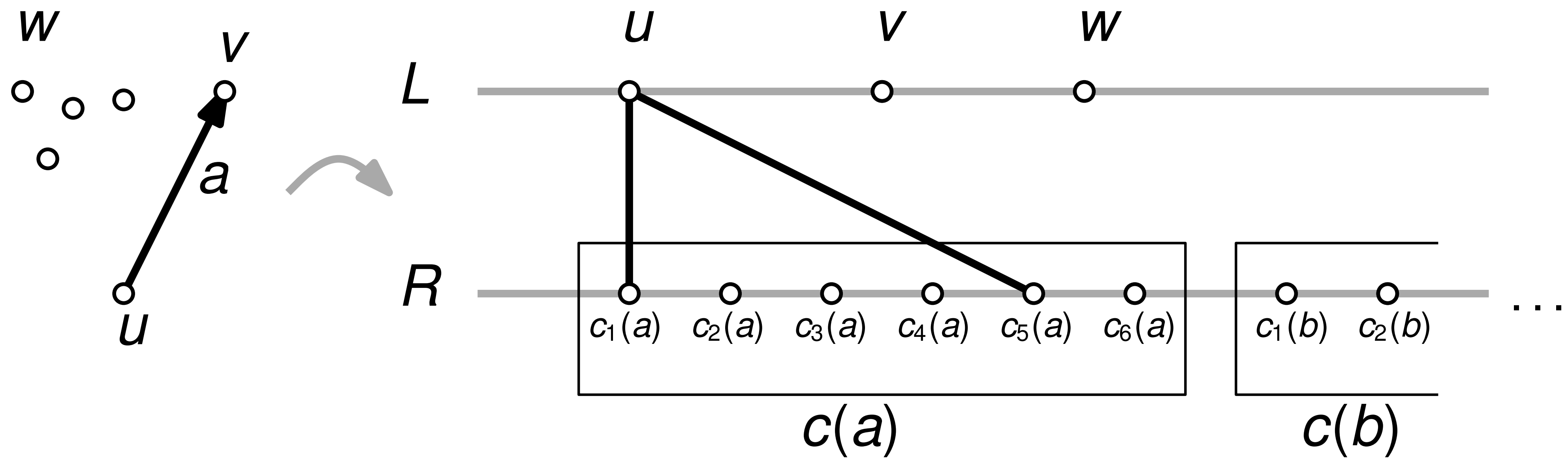
Satz: Das einseitige Kreuzungsminimierungsproblem (OSCM) ist $\mathcal{NP}$-schwer.

Reduktion: Feedback Arc Set



$D = (V, A)$

$B = (W, E)$

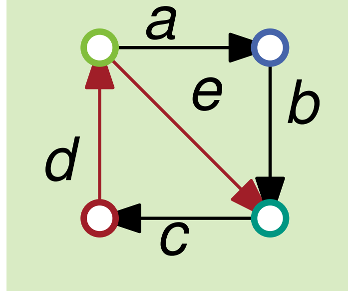


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Transformation

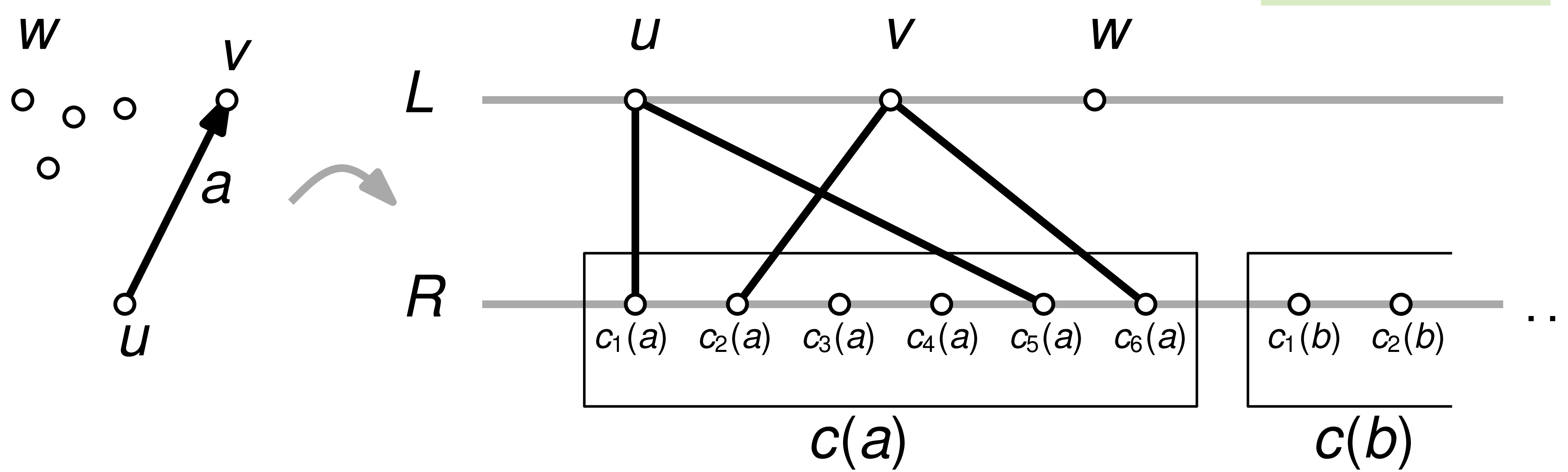
Satz: Das einseitige Kreuzungsminimierungsproblem (OSCM) ist $\mathcal{NP}$-schwer.

A' ist FAS gdw $D = (V, A - A')$ azyklisch



Reduktion: Feedback Arc Set

$D = (V, A)$ $B = (W, E)$

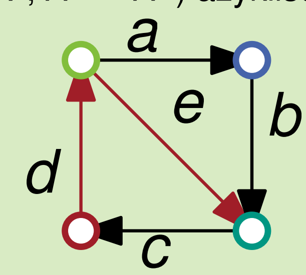


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Transformation

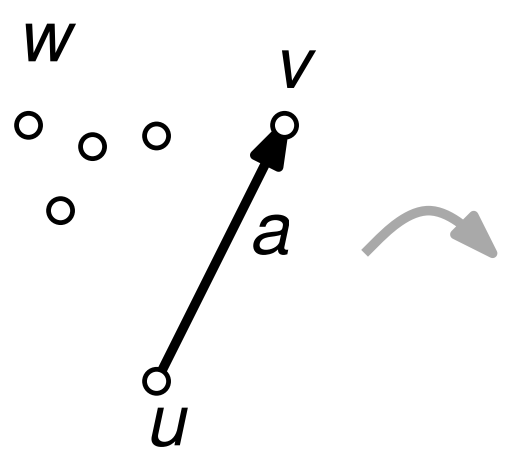
Satz: Das einseitige Kreuzungsminimierungsproblem (OSCM) ist $\mathcal{NP}$-schwer.

A' ist FAS gdw $D = (V, A - A')$ azyklisch

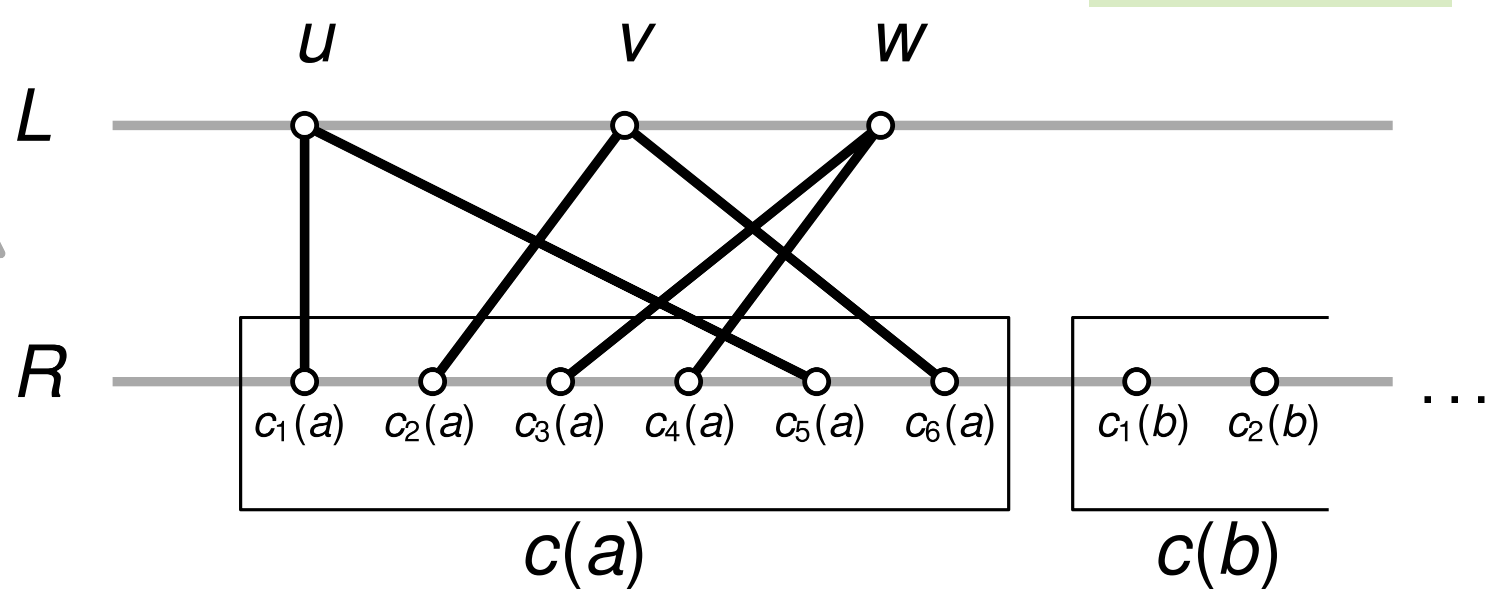


Reduktion: FEEDBACK ARC SET

$D = (V, A)$

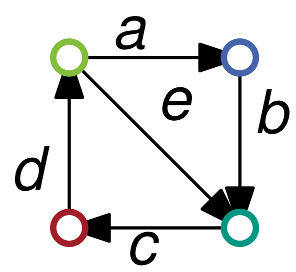


$B = (W, E)$

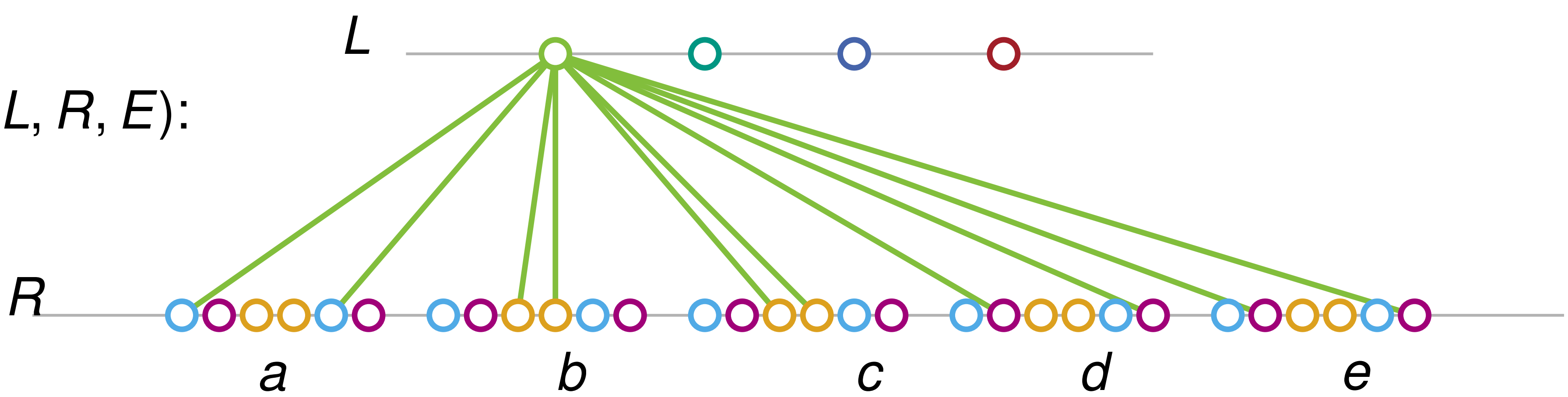


Beispiel

$D = (V, A)$:

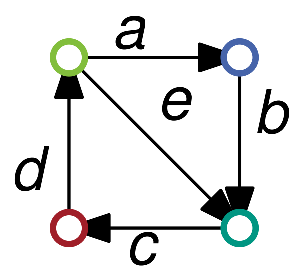


$B = (L, R, E)$:

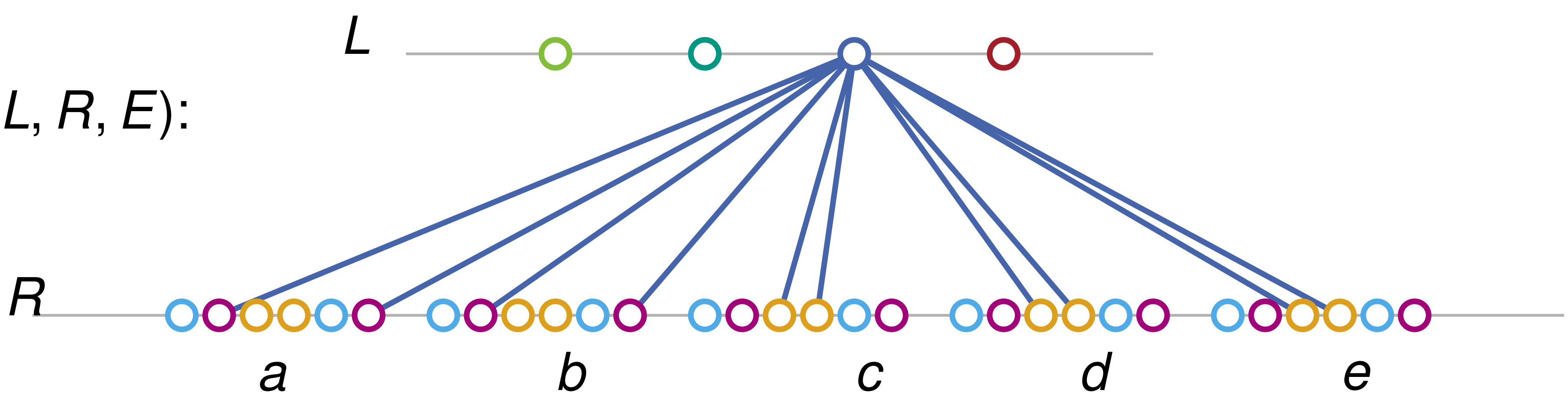


Beispiel

$D = (V, A)$:

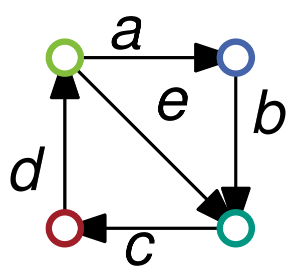


$B = (L, R, E)$:

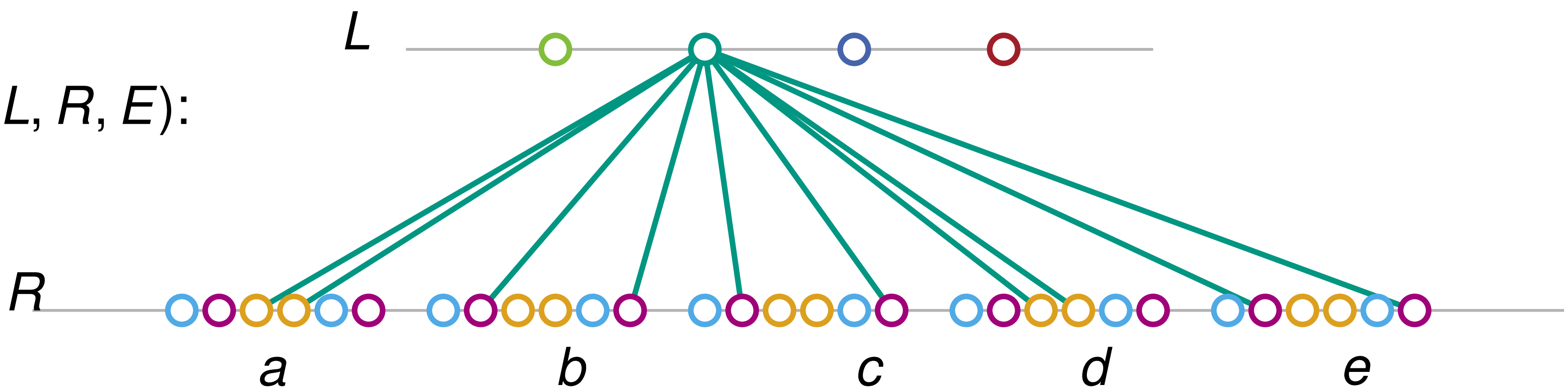


Beispiel

$D = (V, A)$:

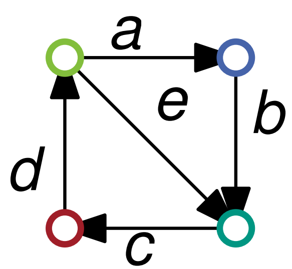


$B = (L, R, E)$:

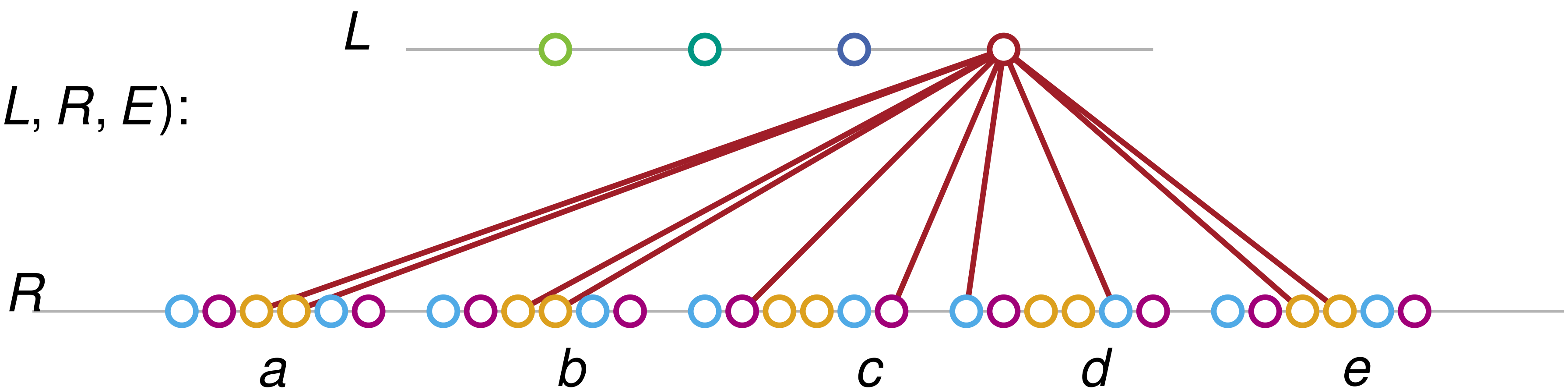


Beispiel

$D = (V, A)$:



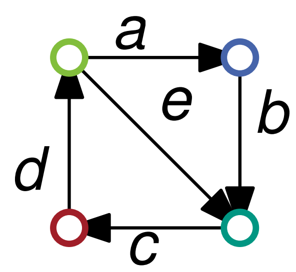
$B = (L, R, E)$:



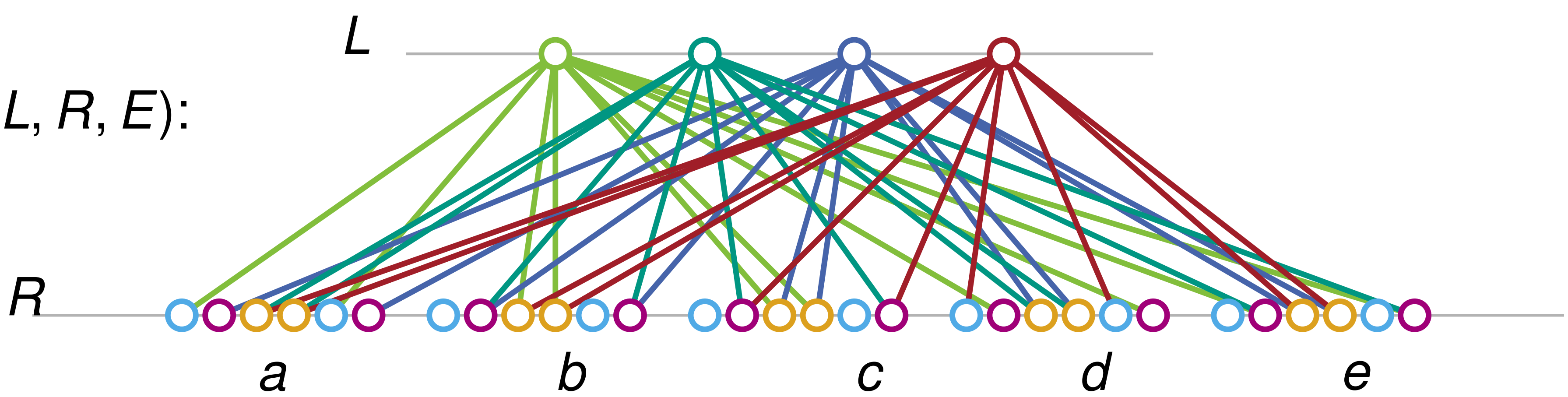
Beispiel

Karlsruher Institut für Technologie

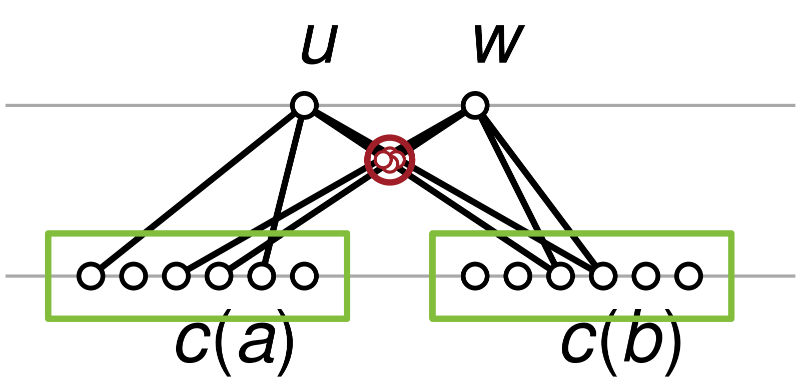
$D = (V, A)$:



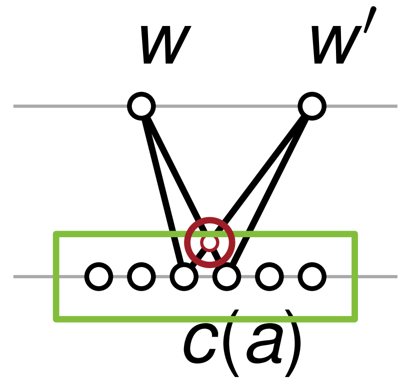
$B = (L, R, E)$:



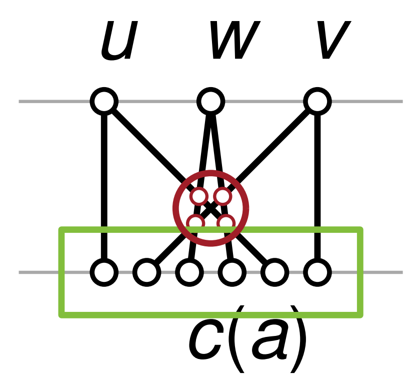
Kreuzungen Zählen



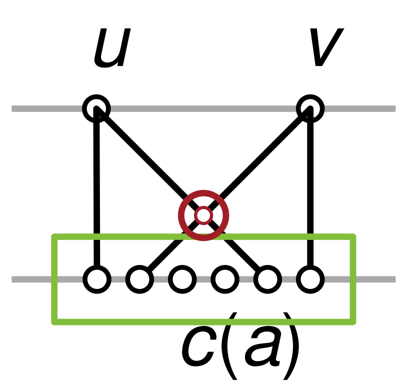
$4 \cdot \binom{n}{2} \cdot \binom{m}{2}$



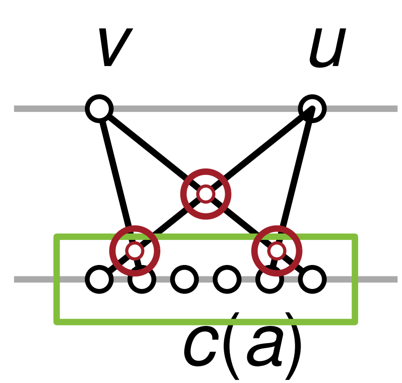
$+m \cdot \binom{n-2}{2}$



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

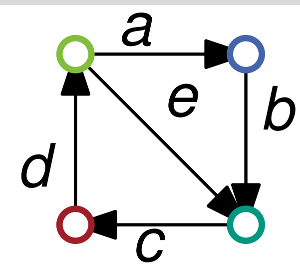


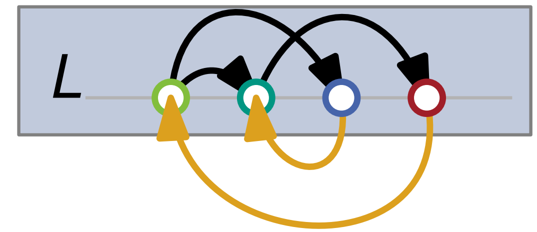
$+(m - m')$



$+3m'$

Korrektheit



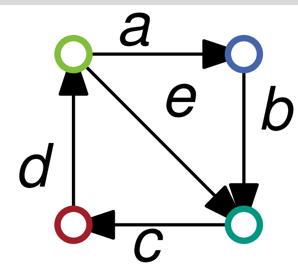


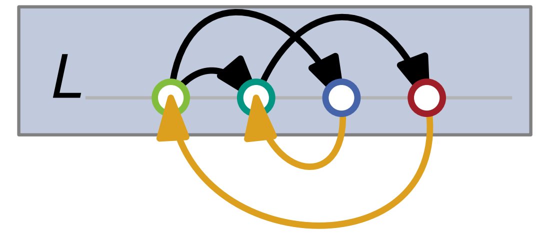
$A_l := \{(u,v) \mid (u,v) \in A, l(u) > l(v)\} = \{b, d\}$

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

Lemma: Sei l eine Ordnung von L, dann hat B genau $M(|A_l|)$ Kreuzungen

Korrektheit





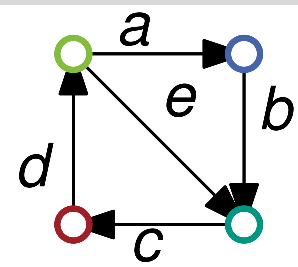
$A_l := \{(u,v) \mid (u,v) \in A, l(u) > l(v)\} = \{b, d\}$

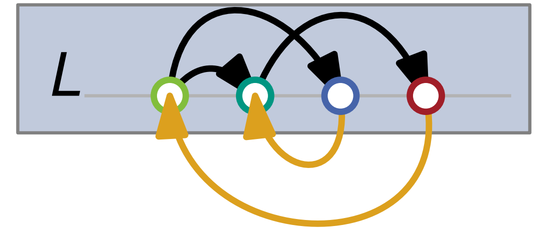
$M(x) = 4\binom{m}{2}\binom{n}{2} + m\binom{n-2}{2} + 4m(n-2) + m + 2x$

Lemma: Sei l eine Ordnung von L, dann hat B genau $M(|A_l|)$ Kreuzungen

D hat Feedback Arc Set der Größe K gdw. B hat $M(K)$ Kreuzungen

Korrektheit





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

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

Lemma: Sei l eine Ordnung von L, dann hat B genau $M(|A_l|)$ Kreuzungen

D hat FEEDBACK ARC SET der Größe K gdw. B hat $M(K)$ Kreuzungen

Sei A' ein FAS von D

$D' = (V, A - A')$ azyklisch

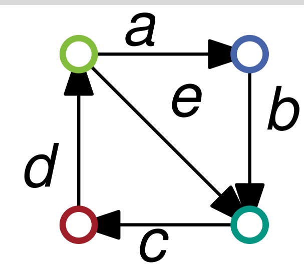
Sei l eine topologische Sortierung von V bzgl. D'

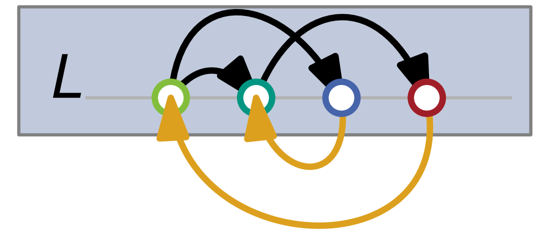
$A_l = A'$

$\Rightarrow$ nach Lemma hat B genau $M(K)$ Kreuzungen

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Korrektheit





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

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

Lemma: Sei l eine Ordnung von L, dann hat B genau $M(|A_l|)$ Kreuzungen

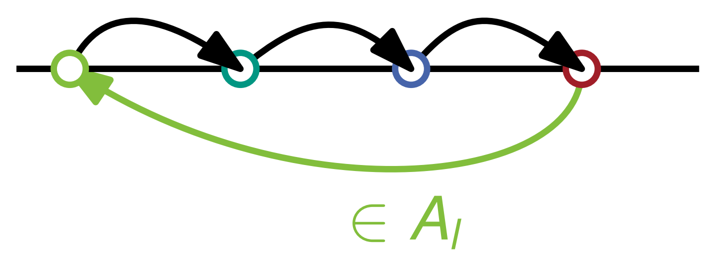
D hat FEEDBACK ARC SET der Größe K gdw. B hat $M(K)$ Kreuzungen

Sei l eine Ordnung von L mit $M(K)$ Kreuzungen

dann hat A_l Größe K (Lemma)

$D' = (V, A - A_l)$ ist azyklisch

$\Rightarrow A_l$ ist FAS der Größe K



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Median Heuristik

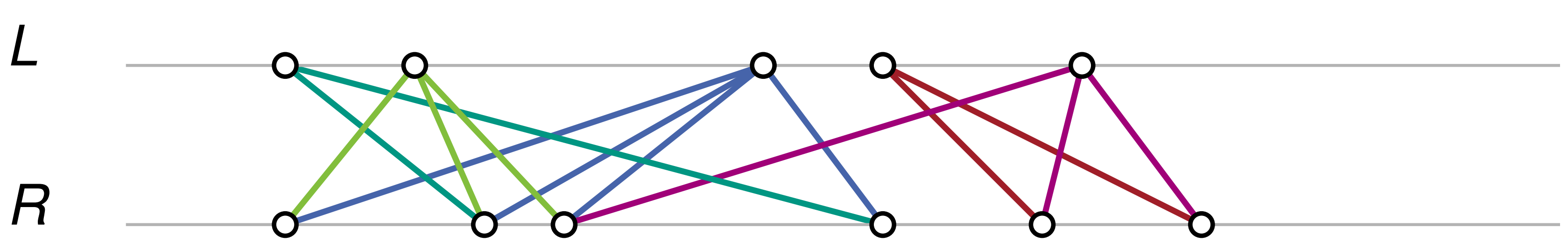
Idee: setze Koordinate auf Median der Nachbarn

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

Median Heuristik

Idee: setze Koordinate auf Median der Nachbarn

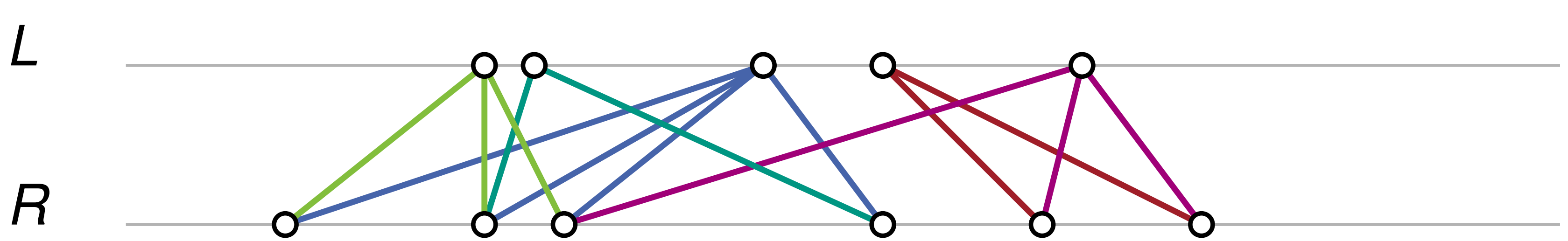
- für Knoten $v \in L$ mit Nachbarn $v_1, \ldots, v_k$ setze $l(v) = \text{med}(v) = r(v_{\lceil k/2 \rceil})$
- falls $l(u) = l(v)$ mit ungleicher Gradparität, setze ungeraden Knoten nach links
- falls $l(u) = l(v)$ mit gleicher Gradparität, setze beliebigen Knoten nach links
- Berechnung mit Linearzeit-Medianalgorithmus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



Median Heuristik

Idee: setze Koordinate auf Median der Nachbarn

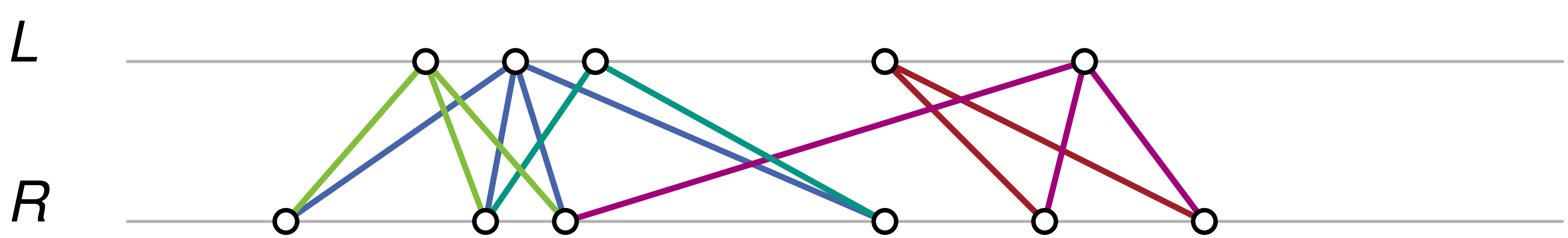
- für Knoten $v \in L$ mit Nachbarn $v_1, \ldots, v_k$ setze $l(v) = \text{med}(v) = r(v_{\lceil k/2 \rceil})$
- falls $l(u) = l(v)$ mit ungleicher Gradparität, setze ungeraden Knoten nach links
- falls $l(u) = l(v)$ mit gleicher Gradparität, setze beliebigen Knoten nach links
- Berechnung mit Linearzeit-Medianalgorithmus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



Median Heuristik

Idee: setze Koordinate auf Median der Nachbarn

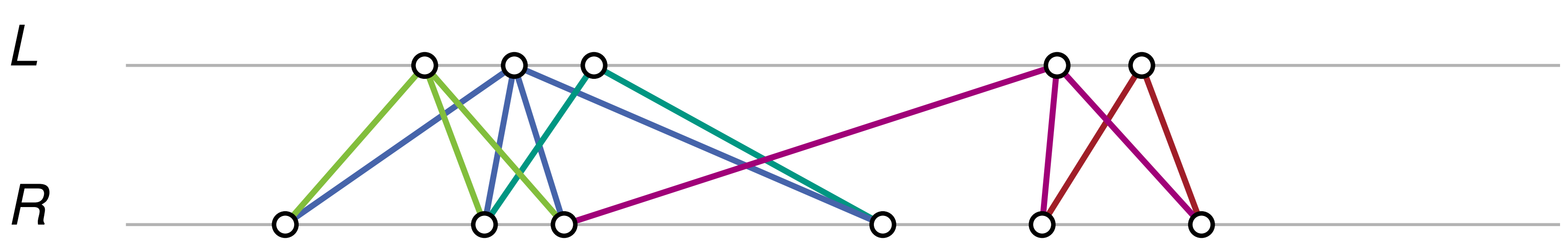
- für Knoten $v \in L$ mit Nachbarn $v_1, \ldots, v_k$ setze $l(v) = \mathrm{med}(v) = r(v_{\lceil k/2 \rceil})$
- falls $l(u) = l(v)$ mit ungleicher Gradparität, setze ungeraden Knoten nach links
- falls $l(u) = l(v)$ mit gleicher Gradparität, setze beliebigen Knoten nach links
- Berechnung mit Linearzeit-Medianalgorithmus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



Median Heuristik

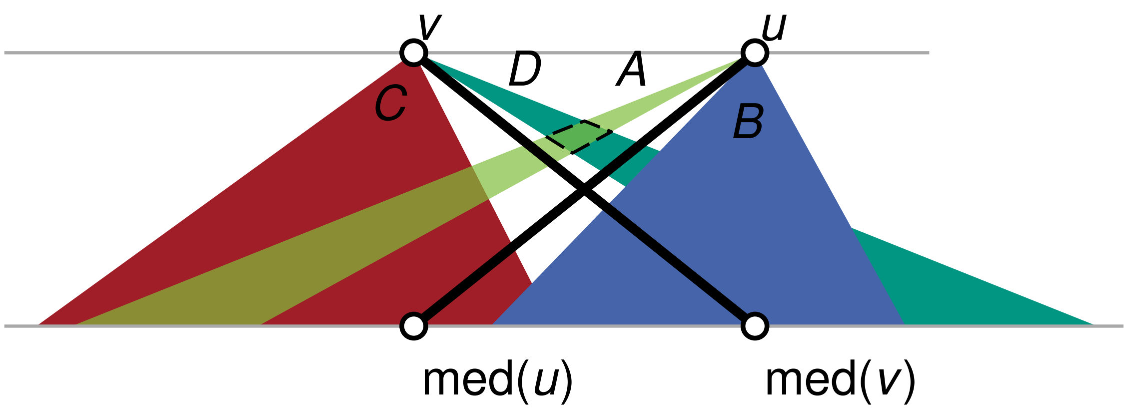
Idee: setze Koordinate auf Median der Nachbarn

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

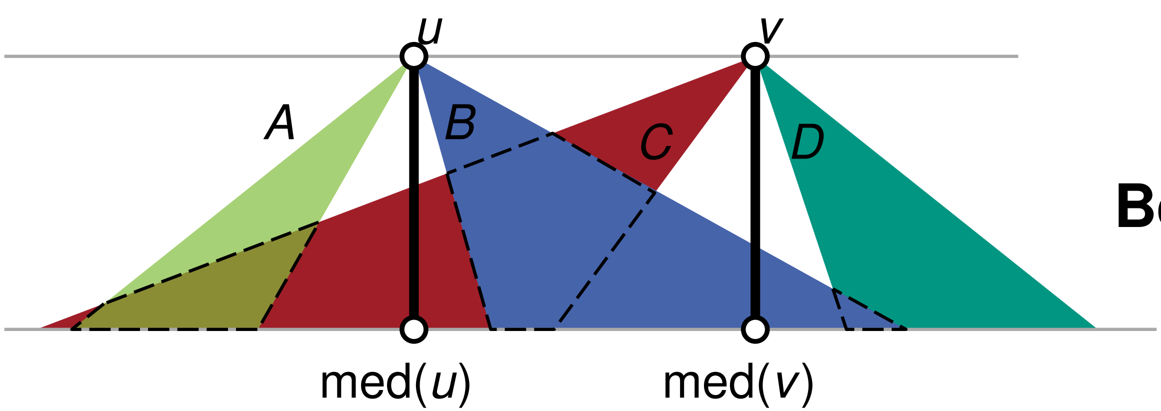


Approximation

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



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



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

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

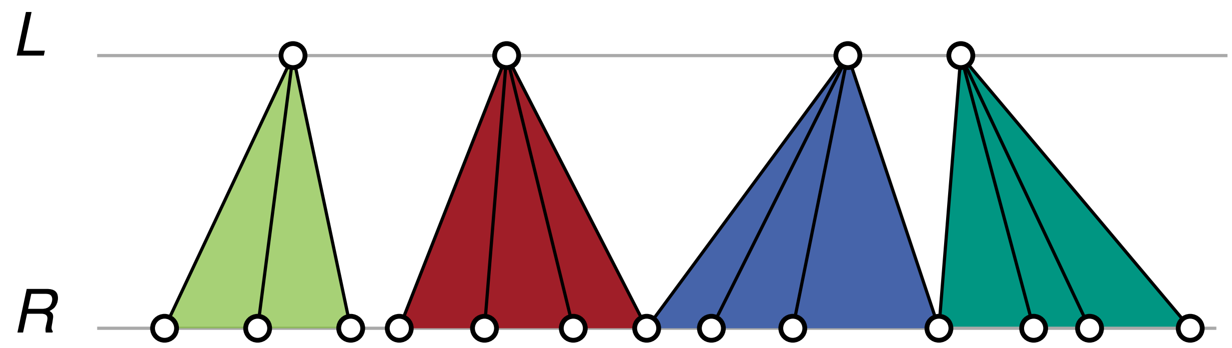
$$\Rightarrow c_{uv} \leq 3ad + 3d + a + 1$$

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 $\text{opt}(G, r) = 0$. Dann gilt $\text{med}(G, r) = 0$.

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
- weitesgehend unabhängige Veränderung
- 3SAT $\propto$ MAX2SAT

Komponenten Design

- Modulierung von Interaktion zwischen Komponenten
- Modeliere z.B. Literale und Klauseln
- 3SAT $\propto$ 3COLOR

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 Wahrheitbelegung 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 Wahrheitbelegung die alle Klauseln erfüllt?

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 Wahrheitbelegung 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 Wahrheitbelegung die alle Klauseln erfüllt?

Transformation 3SAT $\propto$ 4SAT

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 Wahrheitbelegung 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 Wahrheitbelegung die alle Klauseln erfüllt?

Transformation 3SAT $\propto$ 4SAT

Möglichkeit 1: C_3 zweimal kopieren und geschickt verbinden.

Sei (C_3, U) eine 3SAT Instanz

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

Beschränkung

Karlsruher Institut für Technologie

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

Transformation 3SAT $\propto$ 4SAT

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

Sei (C_3, U) eine 3SAT Instanz

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

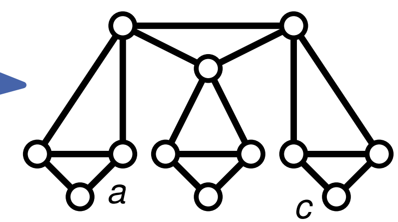
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Lokale Ersetzung

Problem EXACT COVER BY 3SAT (X3C):
Gegeben: Endlische 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, so dass jedes V_i ein Dreieck in G induziert?

Transformation X3C $\propto$ PIT



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

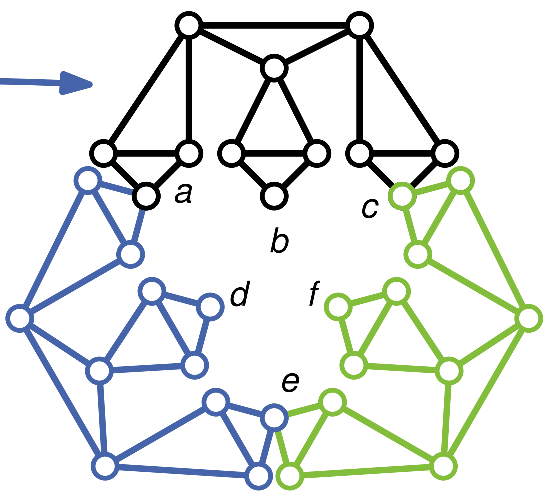
Lokale Ersetzung

Problem EXACT COVER BY 3SAT (X3C):
Gegeben: Endlische 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, so dass jedes V_i ein Dreieck in G induziert?

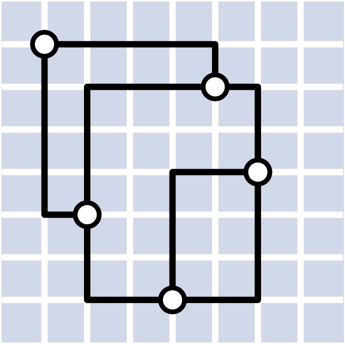
Transformation X3C $\propto$ PIT

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



Komponenten Design

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

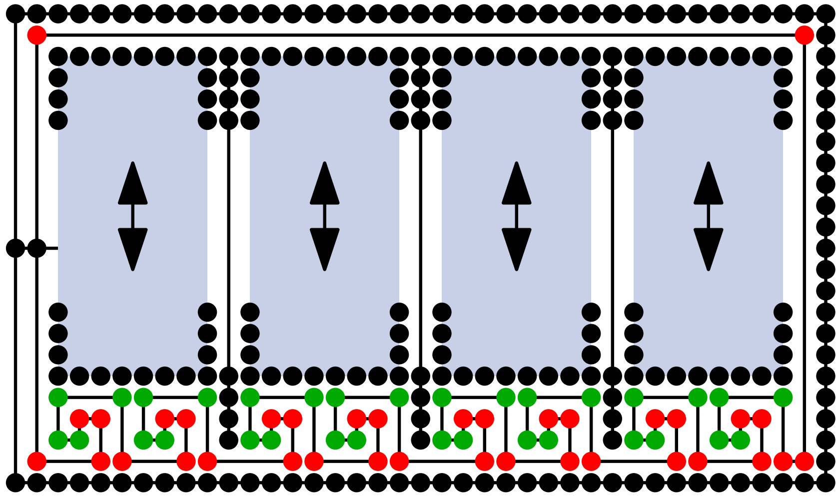


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Komponenten Design

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

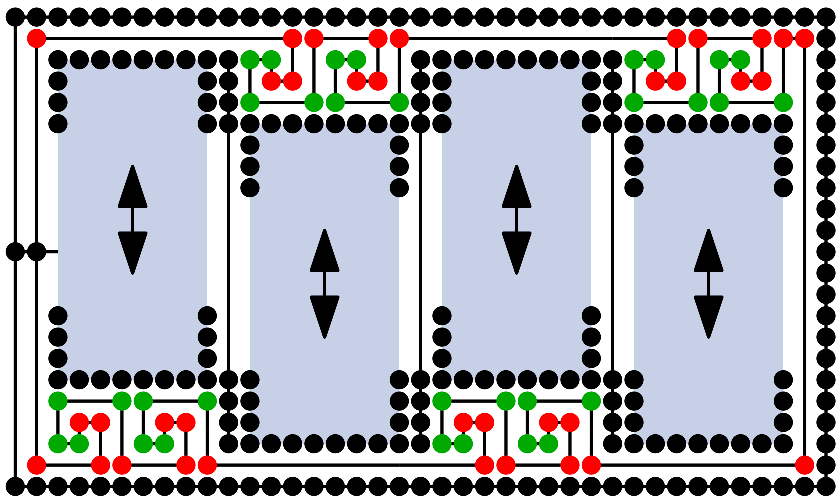
Transformation SAT $\propto$ Orthogonale Zeichnung



Komponenten Design

Problem Orthogonale Zeichnung
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 $\propto$ Orthogonale Zeichnung

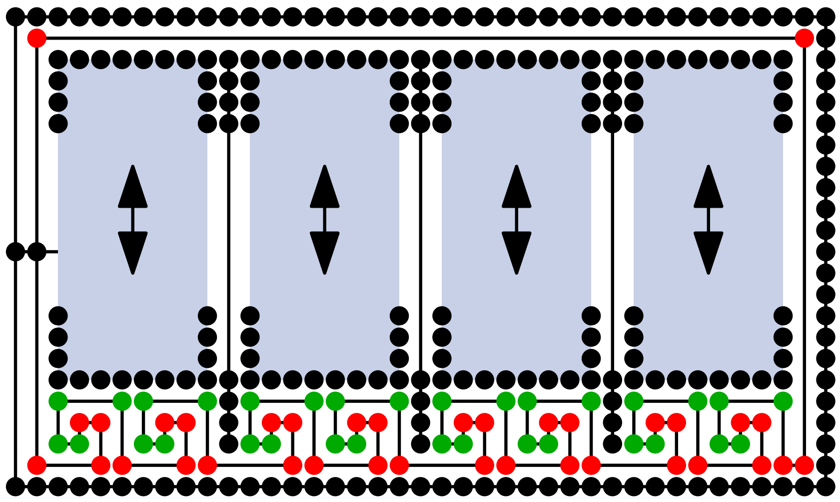


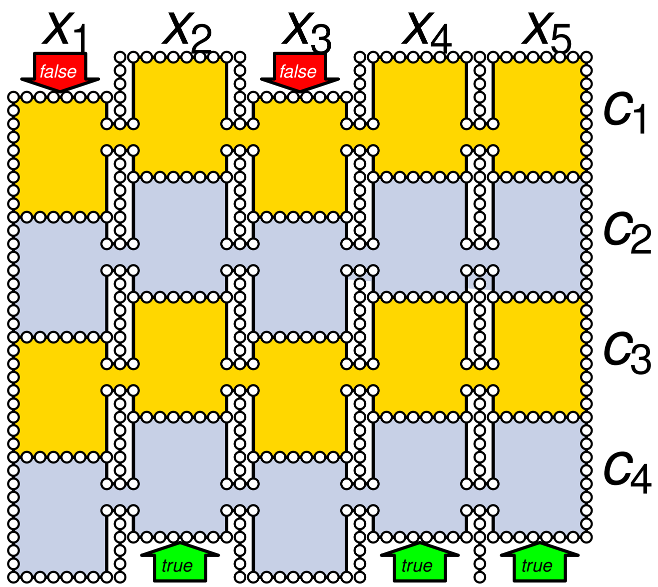
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Komponenten Design

Problem Orthogonale Zeichnung
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 $\propto$ Orthogonale Zeichnung



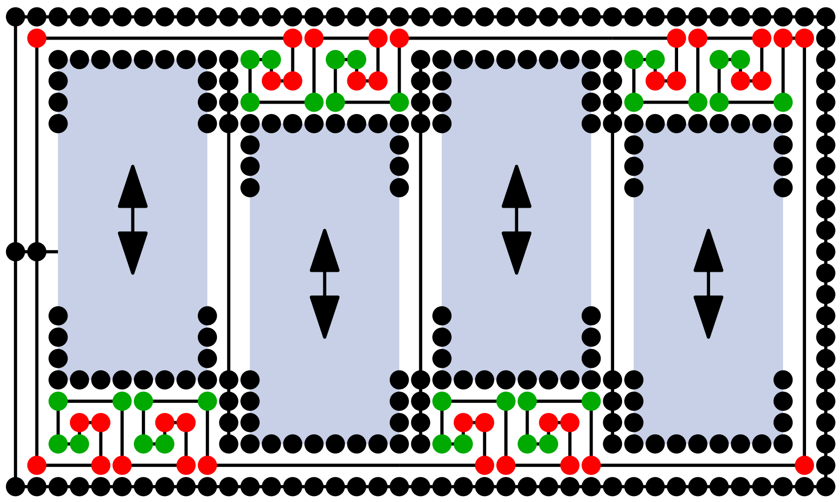


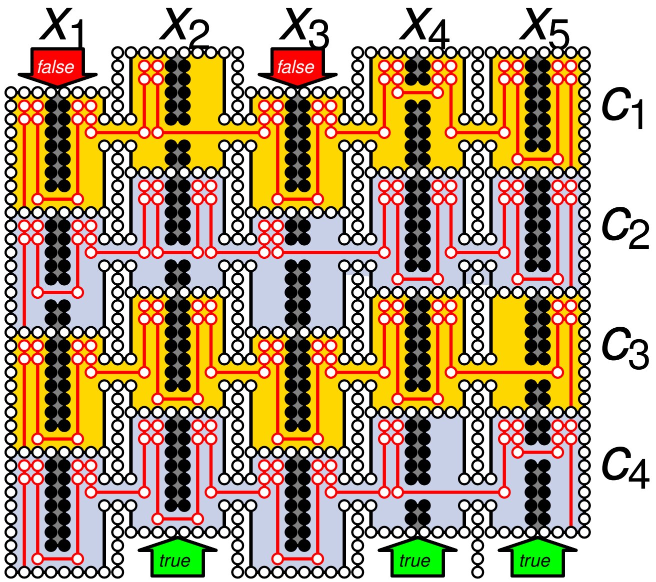
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Komponenten Design

Problem ORTHOGONALE ZEICHNUNG
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 $\propto$ ORTHOGONALE ZEICHNUNG

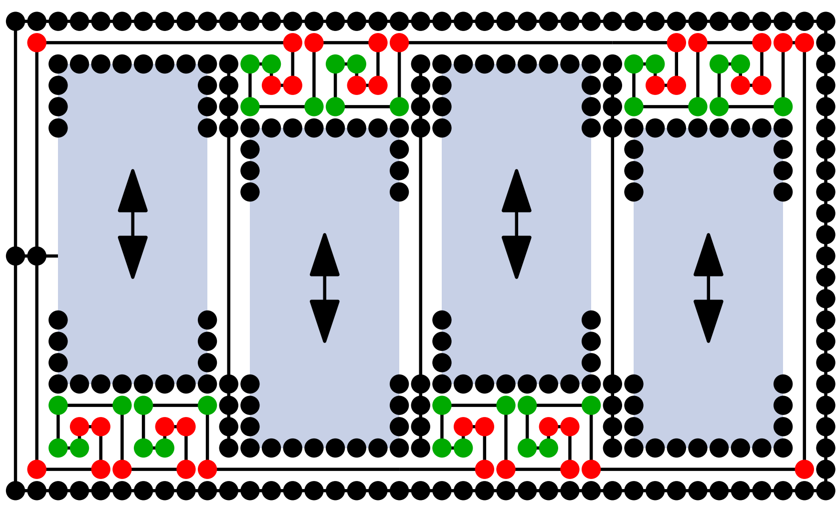


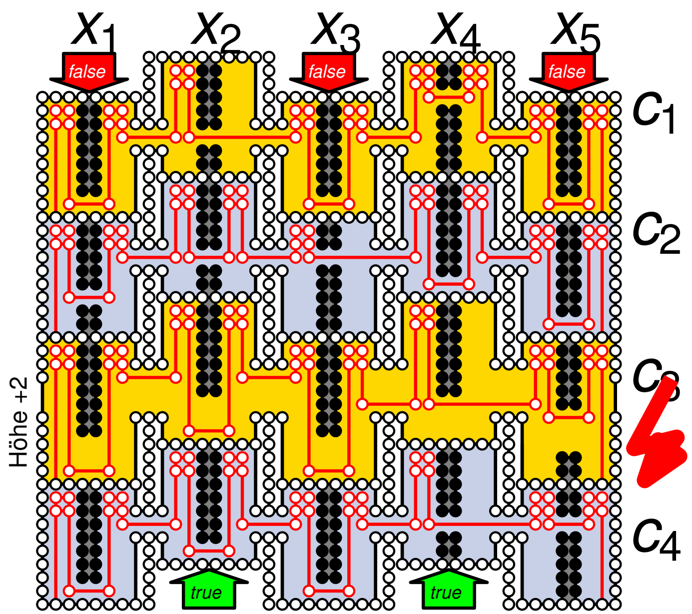


Komponenten Design

Problem ORTHOGONALE ZEICHNUNG
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 $\propto$ ORTHOGONALE ZEICHNUNG





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.

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.

Klausurhinweise

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Klausur

06.03., Dienstag — Letzte Möglichkeit zur Anmeldung.

12.03., Montag — Bekanntgabe der Hörsaalverteilung:

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

13.03., Dienstag
Klausur

Klausur

06.03., Dienstag — Letzte Möglichkeit zur Anmeldung.

12.03., Montag — Bekanntgabe der Hörsaalverteilung:

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

13.03., Dienstag
Klausur

Ablauf der Klausur:

- Klausur beginnt **pünktlich** um 14:00 Uhr
- Bitte **frühzeitig** im **zugewiesenen** Hörsaal einfinden.
- Bearbeitungszeit: 2 Stunden.

Klausur

06.03., Dienstag — Letzte Möglichkeit zur Anmeldung.

12.03., Montag — Bekanntgabe der Hörsaalverteilung:

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

13.03., Dienstag
Klausur

Nicht vergessen:

- Studentenausweis
- Stifte
- Personalausweis!

Ablauf der Klausur:

- Klausur beginnt **pünktlich** um 14:00 Uhr
- Bitte **frühzeitig** im **zugewiesenen** Hörsaal einfinden.
- Bearbeitungszeit: 2 Stunden.

Klausur

06.03., Dienstag — Letzte Möglichkeit zur Anmeldung.

12.03., Montag — Bekanntgabe der Hörsaalverteilung:

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

13.03., Dienstag
Klausur

Nicht vergessen:

- Studentenausweis
- Stifte
- Personalausweis!

Ablauf der Klausur:

- Klausur beginnt **pünktlich** um 14:00 Uhr
- Bitte **frühzeitig** im **zugewiesenen** Hörsaal ei
- Bearbeitungszeit: 2 Stunden.

Keine Hilfsmittel!

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Klausur

06.03., Dienstag | Letzte Möglichkeit zur Anmeldung.

12.03., Montag | Bekanntgabe der Hörsaalverteilung:

- **Rechtzeitig** überprüfen, ob man auf der Liste steht

Abmeldung:

- Bis 06.03.18: über Studierendenportal
- Bis 12.03.18: formloses Schreiben bei Übungsleiter oder Sekretariat abgeben (Studentenausweis mit bringen!).
- Am 13.03.18: Nur noch **direkt vor** der Klausur im entsprechend Hörsaal (Studentenausweis mit bringen!).

- Klausur beginnt **pünktlich** um 14:00 Uhr
- Bitte **frühzeitig** im **zugewiesenen** Hörsaal einfinden.
- Bearbeitungszeit: 2 Stunden.

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.

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.

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!

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!

Aufgabenstellung:

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

Schritte der Berechnung dokumentieren!

Leicht vermeidbare Fehler

Aufgabenstellung:

Beweisen Sie, dass . . .

Wenn nicht anders verlangt, alles notwendige beweisen!

Beispiel: Beweise, dass Problem NP-vollständig ist.

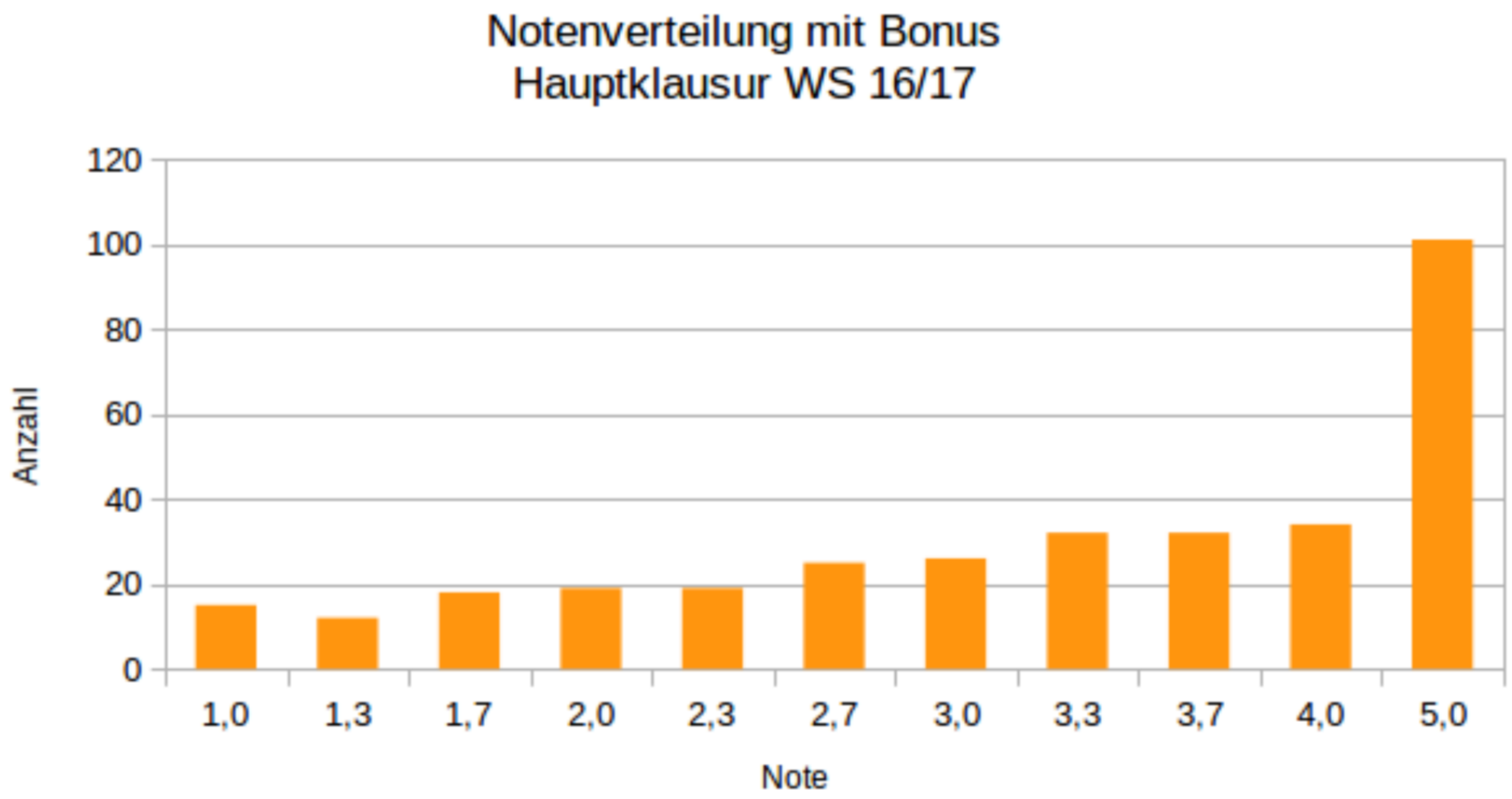
Nicht nur zeigen, dass Problem NP-schwer ist, sondern auch, dass es in NP liegt.

- ist beim Automaten ein Fehlerzustand implizit gegeben?
- akzeptiert der Kellerautomat durch leeren Stack oder durch akzeptierenden Zustand?
- wird gerade $L \in \mathrm{NP}$ gezeigt, oder dass L NP-schwer ist?
- . . .

Antwort ausreichend erklären!

Leicht vermeidbare Fehler

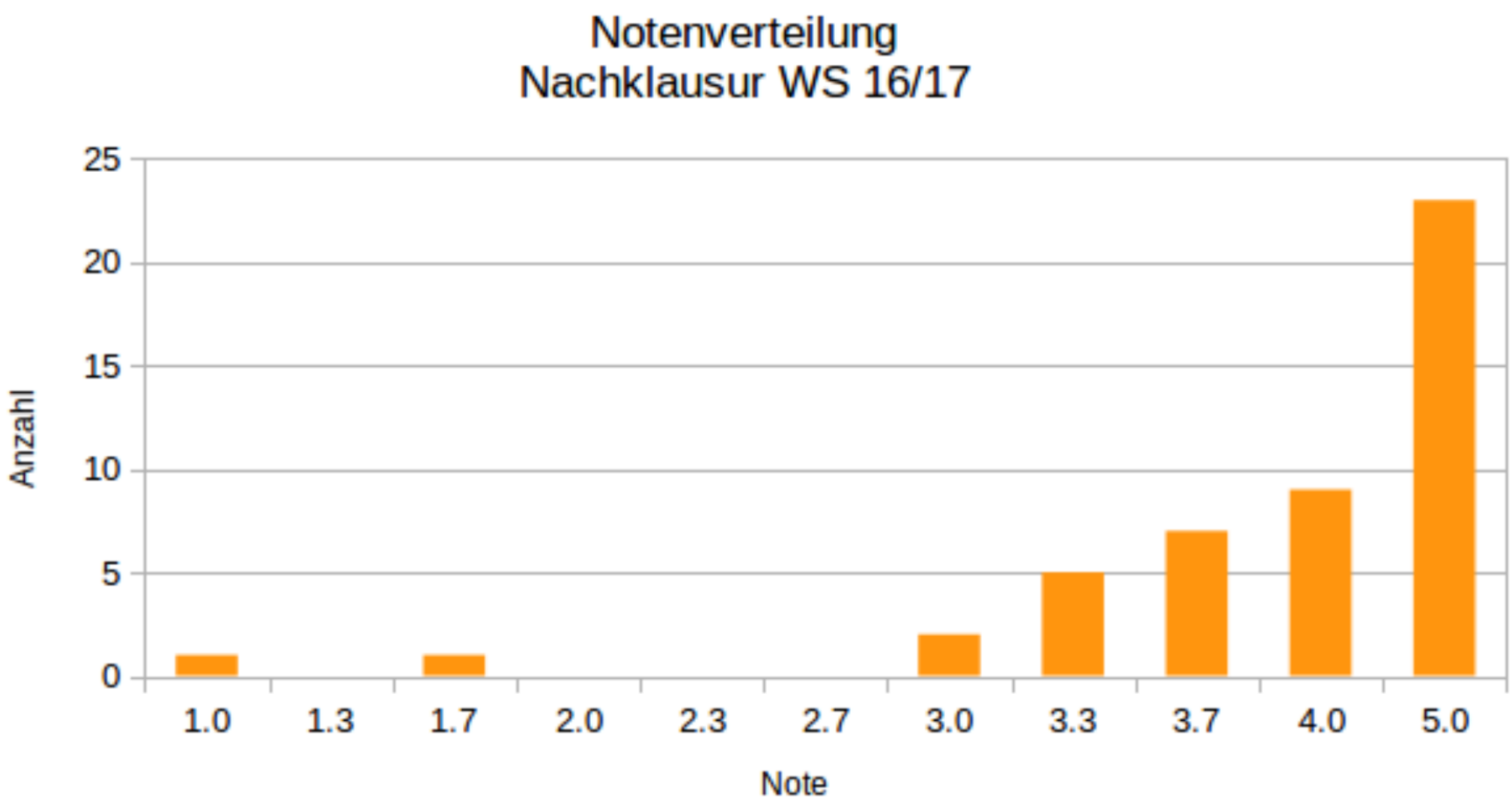
- zu wenig lernen



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Leicht vermeidbare Fehler

- zu wenig lernen



Karlsruher Institut für Technologie

Viel Erfolg bei der Klausur!