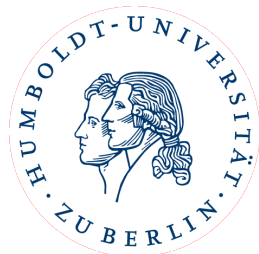
Theoretische Informatik 2

Johannes Köbler



Institut für Informatik
Humboldt-Universität zu Berlin

WS 2009/10

Das P-NP-Problem

- Wie wir gesehen haben, sind NTMs nicht mächtiger als DTMs, d.h. jede NTM kann von einer DTM simuliert werden.
- Die Frage, wieviel Zeit eine DTM zur Simulation einer NTM benötigt, ist eines der wichtigsten offenen Probleme der Informatik.
- Wegen $\mathrm{NTIME}(t) \subseteq \mathrm{DSPACE}(t) \subseteq \mathrm{NSPACE}(t) \subseteq \mathrm{DTIME}(2^{O(t)})$ erhöht sich die Laufzeit im schlimmsten Fall exponentiell.
- Insbesondere die Klasse NP enthält viele für die Praxis überaus wichtige Probleme, für die kein Polynomialzeitalgorithmus bekannt ist.
- Da jedoch nur Probleme in P als effizient lösbar angesehen werden, hat das so genannte P-NP-Problem, also die Frage, ob alle NP-Probleme effizient lösbar sind, eine immense praktische Bedeutung.
- Wie bereits erwähnt, ist diese Frage für die Platzkomplexität bereits gelöst (Satz von Savitch, 1970).

Die Polynomialzeitreduktion

Definition

- Eine Sprache $A \subseteq \Sigma^*$ ist auf $B \subseteq \Gamma^*$ in Polynomialzeit reduzierbar ($A \leq^p B$), falls eine Funktion $f : \Sigma^* \to \Gamma^*$ in FP existiert mit

$$\forall x \in \Sigma^* : x \in A \Leftrightarrow f(x) \in B.$$

- Eine Sprache A heißt $\leq^p$-hart für eine Sprachklasse $\mathcal{C}$ (kurz: $\mathcal{C}$-hart oder $\mathcal{C}$-schwer), falls gilt:

$$\forall L \in \mathcal{C} : L \leq^p A.$$

- Eine $\mathcal{C}$-harte Sprache A, die zu $\mathcal{C}$ gehört, heißt $\mathcal{C}$-vollständig.
- NPC bezeichnet die Klasse aller NP-vollständigen Sprachen.

Lemma

- Aus $A \leq^p B$ folgt $A \leq B$.
- Die Reduktionsrelation $\leq^p$ ist reflexiv und transitiv (s. Übungen).

Die Polynomialzeitreduktion

Satz

Die Klassen P und NP sind unter $\leq^p$ abgeschlossen.

Beweis

- Sei $B \in \mathrm{P}$ und gelte $A \leq^p B$ mittels einer Funktion $f \in \mathrm{FP}$.
- Seien M und T DTMs mit $L(M) = B$ und $T(x) = f(x)$.
- Weiter seien p und q polynomielle Zeitschranken für M und T.
- Betrachte die DTM M', die bei Eingabe x zuerst T simuliert, um $f(x)$ zu berechnen, und danach M bei Eingabe $f(x)$ simuliert.
- Dann gilt

$$x \in A \Leftrightarrow f(x) \in B \Leftrightarrow f(x) \in L(M) \Leftrightarrow x \in L(M').$$

- Also ist $L(M') = A$ und wegen

$$time_{M'}(x) \leq time_T(x) + time_M(f(x)) \leq q(|x|) + p(q(|x|))$$

ist M' polynomiell zeitbeschränkt und somit A in P. □

NP-Vollständigkeit

Satz

1. $A \leq^p B$ und A ist NP-hart $\Rightarrow$ B ist NP-hart.
2. $A \leq^p B$, A ist NP-hart und $B \in \text{NP} \Rightarrow B \in \text{NPC}$.
3. $\text{NPC} \cap \text{P} \neq \emptyset \Rightarrow \text{P} = \text{NP}$.

Beweis

1. Da A NP-hart ist, ist jede NP-Sprache L auf A reduzierbar.
   Da zudem $A \leq^p B$ gilt und $\leq^p$ transitiv ist, folgt $L \leq^p B$.
2. Klar, da B mit (1) NP-hart und nach Voraussetzung in NP ist.
3. Sei B eine NP-vollständige Sprache in P. Dann ist jede NP-Sprache A auf B reduzierbar und da P unter $\leq^p$ abgeschlossen ist, folgt $A \in \text{P}$. $\square$

NP-Vollständigkeit

Satz

Folgende Sprache ist NP-vollständig:

$$L = \left\{ w\#x\#0^m \;\middle|\; \begin{array}{l} w, x \in \{0,1\}^* \text{ und } M_w \text{ ist eine NTM,} \\ \text{die } x \text{ in } \leq m \text{ Schritten akzeptiert} \end{array} \right\}$$

Beweis

- Zunächst ist klar, dass $L \in$ NP mittels folgender NTM M ist:
  M simuliert bei Eingabe $w\#x\#0^m$ die NTM M_w bei Eingabe x für höchstens m Schritte. Falls M_w in dieser Zeit akzeptiert, akzeptiert M ebenfalls, andernfalls verwirft M.
- Sei nun A eine beliebige NP-Sprache. Dann ist A in Polynomialzeit auf eine Sprache $B \subseteq \{0,1\}^*$ in NP reduzierbar (siehe Übungen).
- Sei M_w eine durch ein Polynom p zeitbeschränkte NTM für B.
- Dann reduziert folgende FP-Funktion f die Sprache B auf L:

  $$f : x \mapsto w\#x\#0^{p(|x|)}.$$

□

Aussagenlogische Formeln

- Die Menge der booleschen (oder aussagenlogischen) Formeln über den Variablen $x_1, \ldots, x_n$, $n \geq 0$, ist induktiv wie folgt definiert:
  - Die Konstanten 0 und 1 sind boolesche Formeln.
  - Jede Variable x_i ist eine boolesche Formel.
  - Mit G und H sind auch die Negation $\neg G$, die Konjunktion $(G \wedge H)$ und die Disjunktion $(G \vee H)$ von G und H Formeln.
- Eine Belegung von $x_1, \ldots, x_n$ ist ein Wort $a = a_1 \ldots a_n \in \{0,1\}^n$.
- Der Wert $F(a)$ von F unter a ist induktiv wie folgt definiert:

F	0	1	x_i	$\neg G$	$(G \wedge H)$	$(G \vee H)$
$F(a)$	0	1	a_i	$1 - G(a)$	$G(a)H(a)$	$G(a) + H(a) - G(a)H(a)$

- Durch die Formel F wird also eine n-stellige boolesche Funktion $F : \{0,1\}^n \to \{0,1\}$ definiert, die wir ebenfalls mit F bezeichnen.
- F heißt erfüllbar, falls es eine Belegung a mit $F(a) = 1$ gibt.

Aussagenlogische Formeln

Notation

Wir benutzen die Implikation $G \to H$ als Abkürzung für die Formel $\neg G \vee H$.

Beispiel (Erfüllbarkeitstest mittels Wahrheitswerttabelle)

Die Formel $((\neg x_1 \vee \neg x_2) \to (x_2 \wedge x_3))$ ist erfüllbar:

a	$(\neg x_1 \vee \neg x_2)$	$(x_2 \wedge x_3)$	$((\neg x_1 \vee \neg x_2) \to (x_2 \wedge x_3))$
000	1	0	0
001	1	0	0
010	1	0	0
011	1	1	1
100	1	0	0
101	1	0	0
110	0	0	1
111	0	1	1

◁

Aussagenlogische Formeln

Präzedenzregeln zur Klammerersparnis

- Der Junktor $\wedge$ bindet stärker als der Junktor $\vee$ und dieser wiederum stärker als der Junktor $\rightarrow$.
- Formeln der Form $(x_1 \circ (x_2 \circ (x_3 \circ \cdots \circ x_n) \cdots))$, $\circ \in \{\wedge, \vee, \rightarrow\}$ kürzen wir durch $(x_1 \circ \cdots \circ x_n)$ ab.

Beispiel (Formel für die mehrstellige Entweder-Oder Funktion)

- Folgende Formel nimmt unter einer Belegung $a = a_1 \cdots a_n$ genau dann den Wert 1 an, wenn $\sum_{i=1}^{n} a_i = 1$ ist:

$$G(x_1, \ldots, x_n) = (x_1 \vee \cdots \vee x_n) \wedge \bigwedge_{1 \leq i < j \leq n} \neg(x_i \wedge x_j)$$

- D.h. es gilt genau dann $G(a) = 1$, wenn genau eine Variable x_i mit dem Wert $a_i = 1$ belegt ist.
- Diese Formel wird im Beweis des nächsten Satzes benötigt. $\triangleleft$

Das aussagenlogische Erfüllbarkeitsproblem

Erfüllbarkeitsproblem für boolesche Formeln (*satisfiability*, SAT):

Gegeben: Eine boolesche Formel F.
Gefragt: Ist F erfüllbar?

Satz (Cook, Karp, Levin)

SAT ist NP-vollständig.

Lässt sich also ein effizienter Algorithmus für SAT finden, so kann daraus leicht ein effizienter Algorithmus für jedes NP-Problem abgeleitet werden.

Korollar

$\text{SAT} \in \text{P} \Leftrightarrow \text{P} = \text{NP}$.

SAT ist NP-vollständig

Beweis von $\textsc{Sat} \in \text{NP}$

Es ist leicht zu sehen, dass $\textsc{Sat} \in \text{NP}$ ist:

Eine NTM kann bei Eingabe einer booleschen Formel F zunächst eine Belegung a nichtdeterministisch raten und dann in Polynomialzeit testen, ob $F(a) = 1$ ist (*guess and verify* Strategie). □

Beweis von $\textsc{Sat}$ ist NP-hart

- Sei L eine beliebige NP-Sprache und sei $M = (Z, \Sigma, \Gamma, \delta, q_0)$ eine durch ein Polynom p zeitbeschränkte k-NTM mit $L(M) = L$.
- Da sich eine $t(n)$-zeitbeschränkte k-NTM in Zeit $t^2(n)$ durch eine 1-NTM simulieren lässt, können wir $k = 1$ annehmen.
- Unsere Aufgabe besteht nun darin, in Polynomialzeit zu einer gegebenen Eingabe $w = w_1 \cdots w_n$ eine Formel F_w zu konstruieren, die genau dann erfüllbar ist, wenn $w \in L$ ist.
- O.B.d.A. sei $Z = \{q_0, \ldots, q_m\}$, $E = \{q_m\}$ und $\Gamma = \{a_1, \ldots, a_l\}$.
- Zudem sei $\delta(q_m, a) := \{(q_m, a, N)\}$ für alle $a \in \Gamma$.

Beweis von SAT ist NP-hart

Idee:

Konstruiere F_w so, dass F_w unter einer Belegung a genau dann wahr wird, wenn a eine akzeptierende Rechnung von $M(w)$ beschreibt.

- Wir bilden F_w über den Variablen

$$\begin{aligned} x_{t,q}, &\quad \text{für } 0 \leq t \leq p(n), q \in Z, \\ y_{t,i}, &\quad \text{für } 0 \leq t \leq p(n), -p(n) \leq i \leq p(n), \\ z_{t,i,a}, &\quad \text{für } 0 \leq t \leq p(n), -p(n) \leq i \leq p(n), a \in \Gamma. \end{aligned}$$

- Diese Variablen stehen für folgende Aussagen:

$x_{t,q}$: zum Zeitpunkt t befindet sich M im Zustand q,

$y_{t,i}$: zur Zeit t besucht M das Feld mit der Nummer i,

$z_{t,i,a}$: zur Zeit t steht das Zeichen a auf dem i-ten Feld.

- Konkret sei $F_w = R \wedge S_w \wedge \ddot{U}_1 \wedge \ddot{U}_2 \wedge E$.

Beweis von SAT ist NP-hart

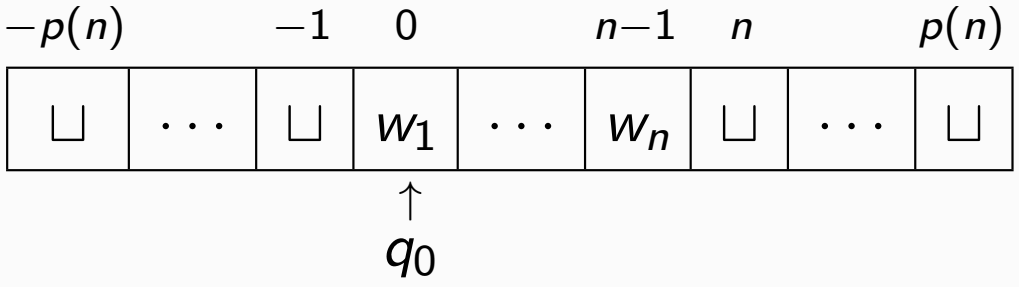
- Konkret sei $F_w = R \wedge S_w \wedge \ddot{U}_1 \wedge \ddot{U}_2 \wedge E$.
- Dabei stellt die Formel $R = \bigwedge_{t=0}^{p(n)} R_t$ (Randbedingungen) sicher, dass wir jeder erfüllenden Belegung eindeutig eine Folge von Konfigurationen $K_0, \ldots, K_{p(n)}$ zuordnen können:

$$R_t = G(x_{t,q_0}, \ldots, x_{t,q_m}) \wedge G(y_{t,-p(n)}, \ldots, y_{t,p(n)}) \wedge \bigwedge_{i=-p(n)}^{p(n)} G(z_{t,i,a_1}, \ldots, z_{t,i,a_l}).$$

- Die Teilformel R_t sorgt also dafür, dass zum Zeitpunkt t
  - genau ein Zustand $q \in \{q_0, \ldots, q_m\}$ eingenommen wird,
  - genau ein Bandfeld $i \in \{-p(n), \ldots, p(n)\}$ besucht wird und
  - auf jedem Feld i genau ein Zeichen $a \in \Gamma$ steht.

Beweis von SAT ist NP-hart

- Die Formel S_w (wie Startbedingung) stellt sicher, dass zum Zeitpunkt 0 tatsächlich die Startkonfiguration vorliegt:



$$S_w = x_{0,q_0} \wedge y_{0,0} \wedge \bigwedge_{i=-p(n)}^{-1} z_{0,i,\sqcup} \wedge \bigwedge_{i=0}^{n-1} z_{0,i,w_{i+1}} \wedge \bigwedge_{i=n}^{p(n)} z_{0,i,\sqcup}$$

- Die Formel $\ddot{U}_1$ sorgt dafür, dass der Inhalt von nicht besuchten Feldern beim Übergang von K_t zu K_{t+1} unverändert bleibt:

$$\ddot{U}_1 = \bigwedge_{t=0}^{p(n)-1} \bigwedge_{i=-p(n)}^{p(n)} \bigwedge_{a \in \Gamma} (\neg y_{t,i} \wedge z_{t,i,a} \rightarrow z_{t+1,i,a})$$

Beweis von SAT ist NP-hart

- $\ddot{U}_2$ achtet darauf, dass sich bei jedem Übergang der Zustand, die Kopfposition und das gerade gelesene Zeichen gemäß einer Anweisung in δ verändern:

$$\ddot{U}_2 = \bigwedge_{t=0}^{p(n)-1} \bigwedge_{i=-p(n)}^{p(n)} \bigwedge_{a\in\Gamma} \bigwedge_{p\in Z} \big(x_{t,p} \wedge y_{t,i} \wedge z_{t,i,a} \rightarrow \bigvee_{(q,b,D)\in\delta(p,a)} x_{t+1,q} \wedge y_{t+1,i+D} \wedge z_{t+1,i,b}\big),$$

  wobei

$$i + D = \begin{cases} i-1, & D = L \\ i, & D = N \\ i+1, & D = R \end{cases}$$

- Schließlich überprüft E, ob M zur Zeit $p(n)$ den Endzustand q_m erreicht hat:

$$E = x_{p(n),q_m}$$

Beweis von SAT ist NP-hart

- Da der Aufbau der Formel $f(w) = F_w$ einem einfachen Bildungsgesetz folgt und ihre Länge polynomiell in n ist, folgt $f \in \mathrm{FP}$.
- Es ist klar, dass F_w im Fall $w \in L(M)$ erfüllbar ist, indem wir die Variablen von F_w gemäß einer akz. Rechnung von $M(w)$ belegen.
- Umgekehrt führt eine Belegung a mit $F_w(a) = 1$ wegen $R(a) = 1$ eindeutig auf eine Konfigurationenfolge $K_0, \ldots, K_{p(n)}$, so dass gilt:
  - K_0 ist Startkonfiguration von $M(w)$ (wegen $S_w(a) = 1$),
  - $K_i \vdash K_{i+1}$ für $i = 0, \ldots, p(n) - 1$ (wegen $\ddot{U}_1(a) = \ddot{U}_2(a) = 1$),
  - M nimmt spätestens bei Erreichen der Konfiguration $K_{p(n)}$ den Endzustand q_m an (wegen $E(a) = 1$).
- Also gilt für alle $w \in \Sigma^*$ die Äquivalenz

$$w \in L(M) \Leftrightarrow F_w \in \mathrm{SAT},$$

d.h. die FP-Funktion $f : w \mapsto F_w$ reduziert $L(M)$ auf SAT. □

Boolesche Schaltkreise

Als nächstes betrachten wir das Erfüllbarkeitsproblem für boolesche Schaltkreise.

Definition

- Ein boolescher Schaltkreis über den Variablen $x_1, \ldots, x_n$ ist eine Folge $s = (g_1, \ldots, g_m)$ von Gattern

  $$g_l \in \{0, 1, x_1, \ldots, x_n, (\neg, j), (\wedge, j, k), (\vee, j, k)\} \text{ mit } 1 \leq j, k < l.$$

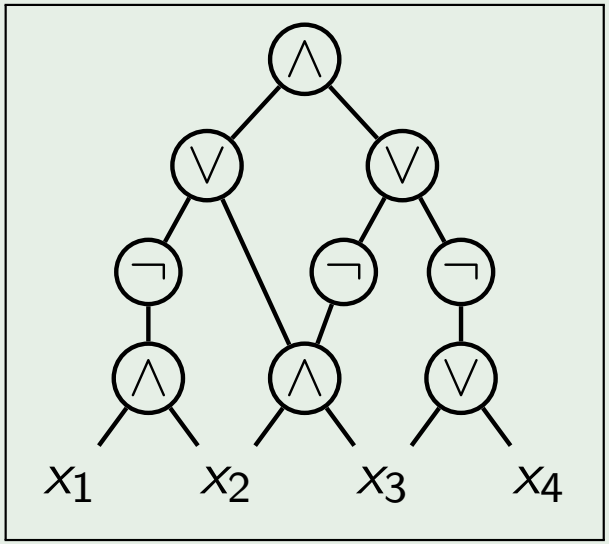
- Die am Gatter g_l berechnete n-stellige boolesche Funktion ist induktiv wie folgt definiert:

g_l	0	1	x_i	$(\neg, j)$	$(\wedge, j, k)$	$(\vee, j, k)$
$g_l(a)$	0	1	a_i	$1 - g_j(a)$	$g_j(a)g_k(a)$	$g_j(a) + g_k(a) - g_j(a)g_k(a)$

- s berechnet die boolesche Funktion $s(a) = g_m(a)$.
- s heißt erfüllbar, wenn eine Eingabe $a \in \{0, 1\}^n$ mit $s(a) = 1$ ex.

Boolesche Schaltkreise

Beispiel



Graphische Darstellung des Schaltkreises $c = (x_1, x_2, x_3, x_4, (\wedge, 1, 2), (\wedge, 2, 3), (\vee, 3, 4), (\neg, 5), (\neg, 6), (\neg, 7), (\vee, 6, 8), (\vee, 9, 10), (\wedge, 11, 12))$.

Bemerkung

- Die Anzahl der Eingänge eines Gatters g wird als Fanin von g bezeichnet,
- die Anzahl der Ausgänge von g (also die Anzahl der Gatter, die g als Eingabe benutzen) als Fanout.
- Boolesche Formeln entsprechen also den booleschen Schaltkreisen mit (maximalem) Fanout 1 und umgekehrt.
- Eine boolesche Formel F kann somit leicht in einen äquivalenten Schaltkreis s mit $s(a) = F(a)$ für alle Belegungen a transformiert werden.

Das Erfüllbarkeitsproblem für boolesche Schaltkreise

Erfüllbarkeitsproblem für boolesche Schaltkreise (CirSat):

Gegeben: Ein boolescher Schaltkreis s.
Gefragt: Ist s erfüllbar?

Satz

CirSat ist NP-vollständig.

Beweis

Klar, da Sat $\leq^p$ CirSat und CirSat $\in$ NP gilt. □

Bemerkung

- Da Sat NP-vollständig ist, ist CirSat auf Sat reduzierbar.
- Dies bedeuted jedoch nicht, dass sich jeder Schaltkreis in Polynomialzeit in eine logisch äquivalente Formel überführen lässt, sondern nur in eine erfüllbarkeitsäquivalente Formel.
- CirSat ist sogar auf eine spezielle Sat-Variante reduzierbar.

Formeln in konjunktiver Normalform (KNF)

Definition

- Ein Literal ist eine Variable x_i oder eine negierte Variable $\neg x_i$, die wir auch kurz mit $\bar{x}_i$ bezeichnen.
- Eine Klausel ist eine Disjunktion $C = \bigvee_{j=1}^{k} l_j$ von Literalen.
- Eine boolesche Formel F ist in konjunktiver Normalform (kurz KNF), falls F eine Konjunktion von Klauseln $C_1, \ldots, C_m$ ist,

$$F = \bigwedge_{j=1}^{m} C_j.$$

- Enthält jede Klausel höchstens k Literale, so heißt F in k-KNF.

Notation

Klauseln werden oft als Menge $C = \{l_1, \ldots, l_k\}$ ihrer Literale und KNF-Formeln als Menge $F = \{C_1, \ldots, C_m\}$ ihrer Klauseln dargestellt.

Erfüllbarkeitsprobleme für Formeln in KNF

Erfüllbarkeitsproblem für k-KNF Formeln (k-SAT):

Gegeben: Eine boolesche Formel F in k-KNF.

Gefragt: Ist F erfüllbar?

Not-All-Equal-SAT (NAESAT):

Gegeben: Eine Formel F in 3-KNF.

Gefragt: Hat F eine (erfüllende) Belegung, unter der in keiner Klausel alle Literale denselben Wahrheitswert haben?

Beispiel

- Die 3-KNF Formel $F = (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_3) \wedge (x_2 \vee \bar{x}_3 \vee x_4)$ ist alternativ durch folgende Klauselmenge darstellbar:

$$F = \{\{x_1, \bar{x}_2\}, \{\bar{x}_1, x_3\}, \{x_2, \bar{x}_3, x_4\}\}$$

- Offenbar ist $F(1111) = 1$, d.h. $F \in$ 3-SAT. Zudem gilt $F \in$ NAESAT.

Komplexität von Erfüllbarkeitsproblemen

Satz

- 1-SAT und 2-SAT sind in P entscheidbar (siehe Übungen).
- 3-SAT und NAESAT sind NP-vollständig.

3-SAT ist NP-vollständig

Reduktion von CIRSAT auf 3-SAT

- Wir transformieren einen Schaltkreis $s = (g_1, \ldots, g_m)$ mit n Eingängen in eine 3-KNF Formel F_s über den Variablen $x_1, \ldots, x_n, y_1, \ldots, y_m$, die die Klausel $\{y_m\}$ und für jedes Gatter g_i die folgenden Klauseln enthält:

Gatter g_i	zugeh. Klauseln in F_S	Semantik
0	$\{\bar{y}_i\}$	$y_i = 0$
1	$\{y_i\}$	$y_i = 1$
x_j	$\{\bar{y}_i, x_j\}, \{\bar{x}_j, y_i\}$	$y_i \leftrightarrow x_j$
$(\neg, j)$	$\{\bar{y}_i, \bar{y}_j\}, \{y_j, y_i\}$	$y_i \leftrightarrow \bar{y}_j$
$(\wedge, j, k)$	$\{\bar{y}_i, y_j\}, \{\bar{y}_i, y_k\}, \{\bar{y}_j, \bar{y}_k, y_i\}$	$y_i \leftrightarrow y_j \wedge y_k$
$(\vee, j, k)$	$\{\bar{y}_j, y_i\}, \{\bar{y}_k, y_i\}, \{\bar{y}_i, y_j, y_k\}$	$y_i \leftrightarrow y_j \vee y_k$

3-SAT ist NP-vollständig

Reduktion von CIRSAT auf 3-SAT

- Nun ist leicht zu sehen, dass für alle $a \in \{0,1\}^n$ folgende Äquivalenz gilt:

$$s(a) = 1 \Leftrightarrow \exists b \in \{0,1\}^m : F_s(ab) = 1.$$

- Ist nämlich $a \in \{0,1\}^n$ eine Eingabe mit $s(a) = 1$. Dann erhalten wir mit

$$b_i = g_i(a) \text{ für } i = 1, \ldots, m$$

eine erfüllende Belegung $ab_1 \cdots b_m$ für F_s.

- Ist umgekehrt $ab_1 \cdots b_m$ eine erfüllende Belegung für F_s, so muss
  - $b_m = 1$ sein, da $\{y_m\}$ eine Klausel in F_s ist, und
  - durch Induktion über $i = 1, \ldots, m$ folgt

$$g_i(a) = b_i,$$

d.h. insbesondere folgt $s(a) = g_m(a) = b_m = 1$.

3-SAT ist NP-vollständig

Reduktion von CIRSAT auf 3-SAT

- Nun ist leicht zu sehen, dass für alle $a \in \{0,1\}^n$ folgende Äquivalenz gilt:

$$s(a) = 1 \Leftrightarrow \exists b \in \{0,1\}^m : F_s(ab) = 1.$$

- Dies bedeutet, dass der Schaltkreis s und die 3-KNF-Formel F_s erfüllbarkeitsäquivalent sind, d.h.

$$s \in \text{CIRSAT} \Leftrightarrow F_s \in \text{3-SAT}.$$

- Zudem ist leicht zu sehen, dass die Reduktionsfunktion $s \mapsto F_s$ in FP berechenbar ist, womit $\text{CIRSAT} \leq^p \text{3-SAT}$ folgt. □

Eine Variante von 3-SAT

Not-All-Equal-SAT (NAESAT):

Gegeben: Eine Formel F in 3-KNF.

Gefragt: Hat F eine (erfüllende) Belegung, unter der in keiner Klausel alle Literale denselben Wahrheitswert haben?

Satz

NAESAT ist NP-vollständig.

Beweis

- NAESAT $\in$ NP ist klar.
- Wir reduzieren CIRSAT auf NAESAT, indem wir die Reduktion von CIRSAT auf 3-SAT geeignet anpassen.

NaeSat ist NP-vollständig

Reduktion von CirSat auf NaeSat

- Wir reduzieren CirSat auf NaeSat, indem wir die Reduktion von CirSat auf 3-Sat geeignet anpassen:

Gatter g_i	zugeh. Klauseln in F'_S	Semantik
0	$\{\bar{y}_i, z\}$	$y_i = 0$
1	$\{y_i, z\}$	$y_i = 1$
x_j	$\{\bar{y}_i, x_j, z\}, \{\bar{x}_j, y_i, z\}$	$y_i \leftrightarrow x_j$
$(\neg, j)$	$\{\bar{y}_i, \bar{y}_j, z\}, \{y_j, y_i, z\}$	$y_i \leftrightarrow \bar{y}_j$
$(\wedge, j, k)$	$\{\bar{y}_i, y_j, z\}, \{\bar{y}_i, y_k, z\}, \{\bar{y}_j, \bar{y}_k, y_i\}$	$y_i \leftrightarrow y_j \wedge y_k$
$(\vee, j, k)$	$\{\bar{y}_j, y_i, z\}, \{\bar{y}_k, y_i, z\}, \{\bar{y}_i, y_j, y_k\}$	$y_i \leftrightarrow y_j \vee y_k$

- Es ist leicht zu sehen, dass die Dreierklauseln unter jeder erfüllenden Belegung von F_S bereits beide Wahrheitswerte annehmen.
- Zu den übrigen Klauseln können wir eine neue Variable z hinzufügen.

NaeSat ist NP-vollständig

Reduktion von CirSat auf NaeSat

- Sei $F'_s(x_1, \ldots, x_n, y_1, \ldots, y_m, z)$ die 3-KNF Formel, die die Klausel $\{y_m, z\}$ und für jedes Gatter g_i die folgenden Klauseln enthält:

Gatter g_i	zugeh. Klauseln F'_s
0	$\{\bar{y}_i, z\}$
1	$\{y_i, z\}$
x_j	$\{\bar{y}_i, x_j, z\}, \{\bar{x}_j, y_i, z\}$
$(\neg, j)$	$\{\bar{y}_i, \bar{y}_j, z\}, \{y_j, y_i, z\}$
$(\wedge, j, k)$	$\{\bar{y}_i, y_j, z\}, \{\bar{y}_i, y_k, z\}, \{\bar{y}_j, \bar{y}_k, y_i\}$
$(\vee, j, k)$	$\{\bar{y}_j, y_i, z\}, \{\bar{y}_k, y_i, z\}, \{\bar{y}_i, y_j, y_k\}$

- F'_s entsteht also aus F_s, indem wir zu jeder Klausel mit ≤ 2 Literalen die neue Variable z hinzufügen.

NaeSat ist NP-vollständig

Reduktion von CirSat auf NaeSat

- Da die Funktion $f : s \mapsto F'_s$ offenbar in FP berechenbar ist, bleibt nur noch die Korrektheit von f zu zeigen:

$$s \in \text{CirSat} \Leftrightarrow F'_s \in \text{NaeSat}.$$

- Ist $s \in \text{CirSat}$, so existiert eine Belegung ab mit $F_s(ab) = 1$.
- Folglich enthalten unter dieser Belegung alle Klauseln von F_s (und damit auch alle Klauseln von F'_s) ein wahres Literal.
- Tatsächlich wird unter ab in jeder Dreierklausel von F_s (und damit in jeder Klausel von F'_s, die nicht z enthält) auch ein Literal falsch, da ab neben jeder Dreierklausel der Form
  - $\{\bar{y}_i, y_j, y_k\}$ die Klauseln $\{\bar{y}_j, y_i\}$ und $\{\bar{y}_k, y_i\}$ und neben
  - $\{y_i, \bar{y}_j, \bar{y}_k\}$ die Klauseln $\{y_j, \bar{y}_i\}$ und $\{y_k, \bar{y}_j\}$ erfüllt.
- Setzen wir also $z = 0$, so wird in jeder Klausel von F'_s mindestens ein Literal wahr und mindestens ein Literal falsch.

NaeSat ist NP-vollständig

Reduktion von CirSat auf NaeSat

- die umgekehrte Implikation sei nun $F'_s \in \text{NaeSat}$ angenommen.
- Dann existiert eine Belegung $abc \in \{0,1\}^{n+m+1}$ für

  $$F'_s(x_1, \ldots, x_n, y_1, \ldots, y_m, z),$$

  unter der in jeder Klausel ein wahres und ein falsches Literal vorkommen.
- Da dies auch unter der komplementären Belegung $\overline{abc}$ der Fall ist, können wir $c = 0$ annehmen.
- Dann erfüllt aber die Belegung ab die Formel F_s.
- Also ist $s(a) = 1$ und damit $s \in \text{CirSat}$. □

SAT als Optimierungsproblem

- Ist eine KNF-Formel F nicht erfüllbar, so kann es nützlich sein, herauszufinden, wieviele Klauseln in F maximal erfüllbar sind.
- Die Schwierigkeit dieses Optimierungsproblems lässt sich durch folgendes Entscheidungsproblem charakterisieren.

MAX-k-SAT:

Gegeben: Eine Formel F in k-KNF und eine Zahl l.

Gefragt: Existiert eine Belegung a, die mindestens l Klauseln in F erfüllt?

Hierbei können Klauseln in F auch mehrfach vorkommen.

Satz

1. MAX-1-SAT $\in$ P.
2. MAX-2-SAT $\in$ NPC.

SAT als Optimierungsproblem

Reduktion von 3-SAT auf MAX-2-SAT

- Für eine Dreierklausel $C = \{l_1, l_2, l_3\}$, in der die Variable v nicht vorkommt, sei $G(l_1, l_2, l_3, v)$ die 2-KNF Formel, die aus folgenden 10 Klauseln besteht:

$$\{l_1\}, \{l_2\}, \{l_3\}, \{v\}, \{\overline{l_1}, \overline{l_2}\}, \{\overline{l_2}, \overline{l_3}\}, \{\overline{l_1}, \overline{l_3}\}, \{l_1, \overline{v}\}, \{l_2, \overline{v}\}, \{l_3, \overline{v}\}$$

- Die folgenden 3 Eigenschaften von G sind leicht zu verifizieren:
  - Keine Belegung von G erfüllt mehr als 7 Klauseln von G.
  - Jede Belegung a von C mit $C(a) = 1$ ist zu einer Belegung a' von G erweiterbar, die 7 Klauseln von G erfüllt.
  - Keine Belegung a von C mit $C(a) = 0$ ist zu einer Belegung a' von G erweiterbar, die 7 Klauseln von G erfüllt.
- Sei F eine 3-KNF-Formel über $x_1, \ldots, x_n$ mit m Klauseln.
- Wir nehmen an, dass $C_j = \{l_{j1}, l_{j2}, l_{j3}\}$, $j = 1, \ldots, k$, die Dreier- und $C_{k+1}, \ldots, C_m$ die Einer- und Zweierklauseln von F sind.

SAT als Optimierungsproblem

Reduktion von 3-SAT auf MAX-2-SAT

- Wir nehmen an, dass $C_j = \{l_{j1}, l_{j2}, l_{j3}\}$, $j = 1, \ldots, k$, die Dreier- und $C_{k+1}, \ldots, C_m$ die Einer- und Zweierklauseln von F sind.
- Betrachte die 2-KNF Formel F' mit $10k + (m - k)$ Klauseln ist, die wie folgt aus F entsteht:
  - Ersetze jede Dreierklausel $C_j = \{l_{j1}, l_{j2}, l_{j3}\}$ in F durch die 10 Klauseln der 2-KNF-Formel $G_j = G(l_{j1}, l_{j2}, l_{j3}, v_j)$.
- Dann gilt

$$F \in \text{3-SAT} \Leftrightarrow (F', m + 6k) \in \text{MAX-2-SAT}.$$

- Die Vorwärtsrichtung ergibt sich unmittelbar aus der Tatsache, dass jede erfüllende Belegung a für F zu einer Belegung a' für F' erweiterbar ist, die $7k + m - k = m + 6k$ Klauseln von F' erfüllt.

SAT als Optimierungsproblem

Reduktion von 3-SAT auf MAX-2-SAT

- Dann gilt

$$F \in \text{3-SAT} \Leftrightarrow (F', m + 6k) \in \text{MAX-2-SAT}.$$

- Die Vorwärtsrichtung ergibt sich unmittelbar aus der Tatsache, dass jede erfüllende Belegung a für F zu einer Belegung a' für F' erweiterbar ist, die $7k + m - k = m + 6k$ Klauseln von F' erfüllt.
- Für die Rückwärtsrichtung sei a eine Belegung, die mindestens $m + 6k$ Klauseln von F' erfüllt.
- Da in jeder 10er-Gruppe G_j, $j = 1, \ldots, k$, maximal 7 Klauseln erfüllbar sind, muss a in jeder 10er-Gruppe genau 7 Klauseln und zudem alle Klauseln C_j für $j = k + 1, \ldots, m$ erfüllen.
- Dies ist aber nur möglich, wenn a alle Klauseln C_j von F erfüllt.
- Zudem ist leicht zu sehen, dass $g : F \mapsto (F', m + 6k)$ in FP berechenbar ist, woraus 3-SAT $\leq^p$ MAX-2-SAT folgt. □

Das Wortproblem für NFAs

- Als nächstes zeigen wir, dass das Wortproblem für NFAs effizient entscheidbar ist.
- Dagegen wird sich das Äquivalenzproblem für NFAs als schwer lösbar (genauer: co-NP-hart) herausstellen.

Satz

Das Wortproblem für NFAs,

$$\mathrm{WP}_{\mathrm{NFA}} = \{N\#x \mid N \text{ ist ein NFA und } x \in L(N)\},$$

ist in P entscheidbar.

P-Algorithmus für $\mathrm{WP}_{\mathrm{NFA}}$

```
1  Input: Ein NFA N = (Z, Σ, δ, Q_0, E) und ein Wort x = x_1 ... x_n
2     Q := Q_0
3     for i := 1 to n do
4        Q := ⋃_{q∈Q} δ(q, x_i)
5     if Q ∩ E ≠ ∅ then accept else reject
```

Das Wortproblem für NFAs ist in P entscheidbar

Satz

Das Wortproblem für NFAs,

$$\mathrm{WP}_{\mathrm{NFA}} = \{N\#x \mid N \text{ ist ein NFA und } x \in L(N)\},$$

ist in P entscheidbar.

P-Algorithmus für $\mathrm{WP}_{\mathrm{NFA}}$

1 **Input:** Ein NFA $N = (Z, \Sigma, \delta, Q_0, E)$ und ein Wort $x = x_1 \dots x_n$
2 $Q := Q_0$
3 **for** $i := 1$ **to** n **do**
4 $Q := \bigcup_{q \in Q} \delta(q, x_i)$
5 **if** $Q \cap E \neq \emptyset$ **then accept else reject**

Es ist klar, dass dieser Algorithmus korrekt arbeitet und auch in eine polynomiell zeitbeschränkte DTM für die Sprache $\mathrm{WP}_{\mathrm{NFA}}$ transformiert werden kann.

Das Leerheitsproblem und das Schnittproblem für NFAs

Ganz ähnlich folgt auch, dass das Leerheits- und das Schnittproblem für NFAs in Polynomialzeit lösbar sind (siehe Übungen).

Korollar

Das Leerheitsproblem für NFAs,

$$\mathrm{LP}_{\mathrm{NFA}} = \{N \mid N \text{ ist ein NFA und } L(N) = \emptyset\}$$

und das Schnittproblem für NFAs,

$$\mathrm{SP}_{\mathrm{NFA}} = \{N_1 \# N_2 \mid N_1 \text{ und } N_2 \text{ sind NFAs mit } L(N_1) \cap L(N_2) \neq \emptyset\}$$

sind in P lösbar.

Entscheidungsprobleme für reguläre Ausdrücke

Korollar

Das Wort-, das Leerheits- und das Schnittproblem für reguläre Ausdrücke sind in P entscheidbar.

Beweis

- Ein regulärer Ausdruck α lässt sich in Polynomialzeit in einen äquivalenten NFA N_α transformieren (siehe Übungen).
- Daher gilt $\mathrm{WP}_{\mathrm{RA}} \leq^p \mathrm{WP}_{\mathrm{NFA}}$ mittels $f : (\alpha \# x) \mapsto (N_\alpha \# x)$.
- Da nach vorigem Satz $\mathrm{WP}_{\mathrm{NFA}} \in \mathrm{P}$ ist, und da P unter $\leq^p$ abgeschlossen ist, folgt $\mathrm{WP}_{\mathrm{RA}} \in \mathrm{P}$.
- Ähnlich zeigt man $\mathrm{LP}_{\mathrm{RA}} \leq^p \mathrm{LP}_{\mathrm{NFA}}$ und $\mathrm{SP}_{\mathrm{RA}} \leq^p \mathrm{SP}_{\mathrm{NFA}}$, was die Zugehörigkeit von $\mathrm{LP}_{\mathrm{RA}}$ und $\mathrm{SP}_{\mathrm{RA}}$ zu P impliziert. □

Dagegen sind das Äquivalenz- und das Inklusionsproblem für reguläre Ausdrücke co-NP-hart. Dies gilt sogar für sternfreie reguläre Ausdrücke (kurz SFRAs), also für reguläre Ausdrücke, die keinen Stern enthalten.

Entscheidungsprobleme für reguläre Sprachen

Die Tabelle gibt die Komplexitäten der wichtigsten Entscheidungsprobleme für durch DFAs, NFAs oder (sternfreie) reguläre Ausdrücke gegebene reguläre Sprachen an.

	Wort-problem $x \in L$?	Leerheits-problem $L = \emptyset$?	Schnitt-problem $L_1 \cap L_2 \neq \emptyset$?	Äquivalenz-problem $L_1 = L_2$?	Inklusions-problem $L_1 \subseteq L_2$
DFA	P	P	P	P	P
SFRA	P	P	P	co-NP-vollständig	
RA	P	P	P	PSPACE-vollständig	
NFA	P	P	P	PSPACE-vollständig	

Als nächstes zeigen wir, dass die Probleme $\ddot{\mathrm{A}}\mathrm{P}_{\mathrm{SFRA}}$ und $\mathrm{IP}_{\mathrm{SFRA}}$ co-NP-vollständig sind.

Das Äquivalenzproblem für sternfreie reguläre Ausdrücke

Lemma

Folgende Sprache ist NP-vollständig:

$$\mathrm{SF} = \{\alpha\#0^n \mid \alpha \text{ ist ein SFRA über } \{0,1\} \text{ mit } L(\alpha) \neq \{0,1\}^n\}.$$

Beweis

- Wir zeigen zuerst $\mathrm{SF} \in$ NP.
- Anschließend reduzieren wir 3-$\textsc{Sat}$ auf SF.

Das Äquivalenzproblem für sternfreie reguläre Ausdrücke

- Sei α ein sternfreier regulärer Ausdruck der Länge m.
- Es ist leicht zu zeigen (durch Induktion über den Aufbau von α), dass $L(\alpha)$ nur Wörter der Länge $\leq m$ enthält.
- D.h. es gilt $L(\alpha) \subseteq \{0,1\}^{\leq m}$, wobei $\Sigma^{\leq m} = \bigcup_{i=0}^{m} \Sigma^i$ ist.
- Daher akzeptiert folgender NP-Algorithmus die Sprache $\mathrm{SF} = \{\alpha \# 0^n \mid \alpha \text{ ist SFRA über } \{0,1\} \text{ mit } L(\alpha) \neq \{0,1\}^n\}$.

NP-Algorithmus für SF

```
1  Input: SFRA α über {0,1} und eine unär kodierte Zahl n
2     guess x ∈ {0,1}^n
3     if x ∉ L(α) then accept
4     m := |α|
5     guess y ∈ {0,1}^{≤m}
6     if y ∈ L(α) − {0,1}^n then accept else reject
```

Reduktion von 3-SAT auf SF

- Sei $F = \{C_1, \ldots, C_m\}$ eine 3-KNF Formel.
- Betrachte den SFRA $\alpha_F = (\alpha_1 | \cdots | \alpha_m)$ mit $\alpha_j = \beta_{j1} \cdots \beta_{jn}$ und

$$\beta_{ji} = \begin{cases} 0, & x_i \in C_j, \\ 1, & \bar{x}_i \in C_j, \\ (0|1), & \text{sonst.} \end{cases}$$

- Dann ist $L(\alpha_j) = \{a \in \{0,1\}^n \mid C_j(a) = 0\}$ und daher folgt

$$\begin{aligned} F \in \text{3-SAT} &\Leftrightarrow \exists a \in \{0,1\}^n : F(a) = 1 \\ &\Leftrightarrow \exists a \in \{0,1\}^n \, \forall j = 1, \ldots, m : C_j(a) = 1 \\ &\Leftrightarrow \exists a \in \{0,1\}^n \, \forall j = 1, \ldots, m : a \notin L(\alpha_j) \\ &\Leftrightarrow \exists a \in \{0,1\}^n : a \notin L(\alpha_F) \\ &\Leftrightarrow L(\alpha_F) \neq \{0,1\}^n. \end{aligned}$$

□

Das Äquivalenzproblem für sternfreie reguläre Ausdrücke

Korollar

Das Äquivalenzproblem für sternfreie reguläre Ausdrücke,

$$\ddot{\mathrm{A}}\mathrm{P}_{\mathrm{SFRA}} = \{\alpha\#\beta \mid \alpha, \beta \text{ sind SFRAs mit } L(\alpha) = L(\beta)\},$$

ist co-NP-vollständig.

Beweis

- Wir zeigen, dass das Inäquivalenzproblem für SFRAs NP-vollständig ist.
- Die Zugehörigkeit zu NP liefert folgender Algorithmus:

```
1  Input: SFRAs α und β
2     m := max{|α|, |β|}
3     guess x ∈ {0,1}^{≤m}
4     if x ∈ L(α) Δ L(β) then accept else reject
```

- Zudem lässt sich SF leicht auf dieses Problem reduzieren:

$$\alpha\#0^n \mapsto \alpha\#\underbrace{(0|1)\cdots(0|1)}_{n\text{-mal}}.$$

□

Das Äquivalenz- und das Inklusionsproblem für NFAs

- Das Komplement von SF lässt sich auch auf das Inklusionsproblem für sternfreie reguläre Ausdrücke reduzieren (siehe Übungen).
- Folglich ist $\mathrm{IP}_{\mathrm{SFRA}}$ ebenfalls co-NP-vollständig.
- Das Äquivalenz- und Inklusionsproblem für reguläre Ausdrücke sind demnach co-NP-hart.
- Diese Probleme sind sogar PSPACE-vollständig (auch für NFAs).

	Wort-problem $x \in L$?	Leerheits-problem $L = \emptyset$?	Schnitt-problem $L_1 \cap L_2 \neq \emptyset$?	Äquivalenz-problem $L_1 = L_2$?	Inklusions-problem $L_1 \subseteq L_2$
DFA	P	P	P	P	P
SFRA	P	P	P	co-NP-vollständig	
RA	P	P	P	PSPACE-vollständig	
NFA	P	P	P	PSPACE-vollständig	

Notation – ungerichtete Graphen

- Ein (ungerichteter) Graph ist ein Paar $G = (V, E)$, wobei
  V - eine endliche Menge von Knoten/Ecken und
  E - die Menge der Kanten ist.

  Hierbei gilt

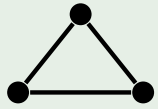
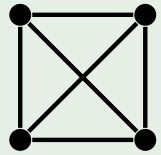
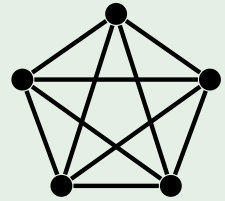
  $$E \subseteq \binom{V}{2} := \{\{u, v\} \subseteq V \mid u \neq v\}.$$

- Die Knotenzahl von G ist $n(G) = \|V\|$.
- Die Kantenzahl von G ist $m(G) = \|E\|$.
- Die Nachbarschaft von $v \in V$ ist $N_G(v) = \{u \in V \mid \{u, v\} \in E\}$ und die Nachbarschaft von $U \subseteq V$ ist $N_G(U) = \bigcup_{u \in U} N_G(u)$.
- Der Grad von $v \in V$ ist $\deg_G(v) = \|N_G(v)\|$.
- Der Minimalgrad von G ist $\delta(G) := \min_{v \in V} \deg_G(v)$ und
- der Maximalgrad von G ist $\Delta(G) := \max_{v \in V} \deg_G(v)$.

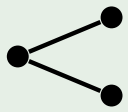
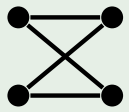
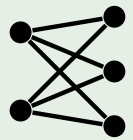
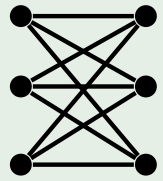
Falls G aus dem Kontext ersichtlich ist, schreiben wir auch einfach n, m, $N(v)$, $\deg(v)$, δ usw.

Beispiel

- Der vollständige Graph (V, E) mit $\|V\| = n$ und $E = \binom{V}{2}$ wird mit K_n und der leere Graph $(V, \emptyset)$ wird mit E_n bezeichnet.

K_1:  K_2:  K_3:  K_4:  K_5: 

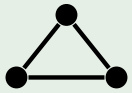
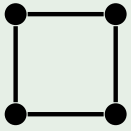
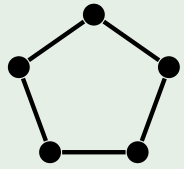
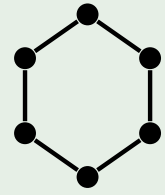
- Der vollständige bipartite Graph (A, B, E) auf $a + b$ Knoten, d.h. $A \cap B = \emptyset$, $\|A\| = a$, $\|B\| = b$ und $E = \{\{u, v\} \mid u \in A, v \in B\}$ wird mit $K_{a,b}$ bezeichnet.

$K_{1,1}$:  $K_{1,2}$:  $K_{2,2}$:  $K_{2,3}$:  $K_{3,3}$: 

- Der Pfad der Länge n wird mit P_n bezeichnet.

P_1:  P_2:  P_3:  P_4: 

- Der Kreis der Länge n wird mit C_n bezeichnet.

C_3:  C_4:  C_5:  C_6: 

Notation – ungerichtete Graphen (Fortsetzung)

- Ein Graph $H = (V', E')$ heißt Sub-/Teil-/Untergraph von $G = (V, E)$, falls $V' \subseteq V$ und $E' \subseteq E$ ist.

  Der Subgraph H heißt (durch V') induziert, falls $E' = E \cap \binom{V'}{2}$ ist. Hierfür schreiben wir auch $H = G[V']$.
- Ein Weg ist eine Folge von Knoten $v_0, \ldots, v_j$ mit $\{v_i, v_{i+1}\} \in E$ für $i = 0, \ldots, j-1$.

  Ein Weg heißt einfach oder Pfad, falls alle durchlaufenen Knoten verschieden sind.

  Die Länge des Weges ist die Anzahl der Kanten, also j.

  Ein Weg $v_0, \ldots, v_j$ heißt auch v_0-v_j-Weg.
- Ein Zyklus ist ein u-v-Weg der Länge $j \geq 2$ mit $u = v$.
- Ein Kreis ist ein Zyklus $v_0, v_1 \ldots, v_{j-1}, v_0$ der Länge $j \geq 3$, für den $v_0, v_1, \ldots, v_{j-1}$ paarweise verschieden sind.

Cliquen, Stabilität und Kantenüberdeckungen

- Eine Knotenmenge $U \subseteq V$ heißt Clique, wenn jede Kante mit beiden Endpunkten in U in E ist, d.h. es gilt $\binom{U}{2} \subseteq E$.

  Die Cliquenzahl ist

  $$\omega(G) := \max\{\|U\| \mid U \text{ ist Clique in } G\}.$$

- Eine Knotenmenge $U \subseteq V$ heißt stabil, wenn keine Kante in G beide Endpunkte in U hat, d.h. es gilt $E \cap \binom{U}{2} = \emptyset$.

  Die Stabilitätszahl ist

  $$\alpha(G) := \max\{\|U\| \mid U \text{ ist stabile Menge in } G\}.$$

- Eine Knotenmenge $U \subseteq V$ heißt Kantenüberdeckung (engl. *vertex cover*), wenn jede Kante $e \in E$ mindestens einen Endpunkt in U hat, d.h. es gilt $e \cap U \neq \emptyset$ für alle Kanten $e \in E$.

  Die Überdeckungszahl ist

  $$\beta(G) = \min\{\|U\| \mid U \text{ ist eine Kantenüberdeckung in } G\}.$$

Fundamentale Graphprobleme

Für einen gegebenen Graphen G und eine gegebene Zahl $k \geq 1$ betrachten wir die folgenden Fragestellungen:

Clique:

Gefragt: Hat G eine Clique der Größe k?

Independent Set (IS):

Gefragt: Hat G eine stabile Menge der Größe k?

Vertex Cover (VC):

Gefragt: Hat G eine Kantenüberdeckung der Größe k?

Satz

Clique, IS und VC sind NP-vollständig.

IS ist NP-vollständig

Reduktion von 3-SAT auf IS

- Sei $F = \{C_1, \ldots, C_m\}$ mit $C_j = \{l_{j,1}, \ldots, l_{j,k_j}\}$ für $j = 1, \ldots, m$ eine 3-KNF-Formel über den Variablen $x_1, \ldots, x_n$.
- Betrachte den Graphen $G = (V, E)$ mit

  $$V = \{v_{ji} \,|\, 1 \leq j \leq m, 1 \leq i \leq k_j\} \text{ und}$$
  $$E = \{\{v_{ji}, v_{j'i'}\} \in \tbinom{V}{2} \,|\, j = j' \text{ oder } l_{ji} = \bar{l}_{j'i'} \text{ oder } l_{j'i'} = \bar{l}_{ji}\}.$$

- Nun gilt

  $F \in$ 3-SAT $\Leftrightarrow$ es gibt eine Belegung, die in jeder Klausel C_j (mindestens) ein Literal l_{j,i_j} wahr macht

  $\Leftrightarrow$ es gibt m Literale $l_{1,i_1}, \ldots, l_{m,i_m}$, die paarweise nicht komplementär sind (d.h. $l_{ji_j} \neq \bar{l}_{j'i_{j'}}$ für $j \neq j'$)

  $\Leftrightarrow$ es gibt m Knoten $v_{1,i_1}, \ldots, v_{m,i_m}$, die nicht durch Kanten verbunden sind

  $\Leftrightarrow$ G besitzt eine stabile Menge von m Knoten. $\square$

Clique ist NP-vollständig

Korollar

Clique ist NP-vollständig.

Beweis

- Es ist leicht zu sehen, dass jede Clique in einem Graphen $G = (V, E)$ eine stabile Menge in dem zu G komplementären Graphen $\bar{G} = (V, \bar{E})$ mit $\bar{E} = \binom{V}{2} \setminus E$ ist und umgekehrt.
- Daher lässt sich IS mittels

  $$f : (G, k) \mapsto (\bar{G}, k)$$

  auf Clique reduzieren. □

VC ist NP-vollständig

Korollar

VC ist NP-vollständig.

Beweis

- Offensichtlich ist eine Menge I genau dann stabil, wenn ihr Komplement $V \setminus I$ eine Kantenüberdeckung ist.
- Daher lässt sich IS mittels

$$f : (G, k) \mapsto (G, n(G) - k)$$

auf VC reduzieren. □

Färbbarkeit von Graphen

Definition

- Eine Abbildung $f\colon V \to \mathbb{N}$ heißt Färbung von G, wenn $f(u) \neq f(v)$ für alle $\{u, v\} \in E$ gilt.
- G heißt k-färbbar, falls eine Färbung $f\colon V \to \{1, \ldots, k\}$ existiert.
- Die chromatische Zahl ist

$$\chi(G) = \min\{k \in \mathbb{N} \mid G \text{ ist } k\text{-färbbar}\}.$$

k-Färbbarkeit (k-Coloring):

Gegeben: Ein Graph G.

Gefragt: Ist G k-färbbar?

Satz

1. 1-Coloring, 2-Coloring $\in$ P (siehe Übungen).
2. 3-Coloring ist NP-vollständig.

Färbbarkeit von Graphen

Reduktion von NaeSat auf 3-Coloring

- Sei eine 3-KNF-Formel $F = \{C_1, \ldots, C_m\}$ über den Variablen $x_1, \ldots, x_n$ mit Klauseln

  $$C_j = \{l_{j,1}, \ldots, l_{j,k_j}\}, \;\; k_j \leq 3$$

  gegeben.
- Wir können annehmen, dass F keine Einerklauseln enthält.
- Wir konstruieren einen Graphen $G_F = (V, E)$, der genau dann 3-färbbar ist, wenn $F \in \text{NaeSat}$ ist.
- Wir setzen

  $$V = \{s, x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n\} \cup \{v_{jk} \,|\, 1 \leq j \leq m, 1 \leq k \leq k_j\}$$

  und

  $$E = \Big\{\{s, x_i\}, \{s, \bar{x}_i\}, \{x_i, \bar{x}_i\} \;\Big|\; 1 \leq i \leq n\Big\} \cup \Big\{\{s, v_{jk}\} \;\Big|\; k_j = 2\Big\} \cup$$
  $$\Big\{\{v_{jk}, v_{jl}\} \;\Big|\; k \neq l\Big\} \cup \Big\{\{v_{jk}, x_i\} \;\Big|\; l_{jk} = \bar{x}_i\Big\} \cup \Big\{\{v_{jk}, \bar{x}_i\} \;\Big|\; l_{jk} = x_i\Big\}.$$

Färbbarkeit von Graphen

Reduktion von NaeSat auf 3-Coloring

- Wir setzen

$$V = \{s\} \cup \{x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n\} \cup \{v_{jk} \,|\, 1 \leq j \leq m, 1 \leq k \leq k_j\},$$

und

$$E = \Big\{\{s, x_i\}, \{s, \bar{x}_i\}, \{x_i, \bar{x}_i\} \,\Big|\, 1 \leq i \leq n\Big\} \cup \Big\{\{s, v_{jk}\} \,\Big|\, k_j = 2\Big\} \cup \Big\{\{v_{jk}, v_{jl}\} \,\Big|\, k \neq l\Big\} \cup \Big\{\{v_{jk}, x_i\} \,\Big|\, l_{jk} = \bar{x}_i\Big\} \cup \Big\{\{v_{jk}, \bar{x}_i\} \,\Big|\, l_{jk} = x_i\Big\}.$$

- Sei $a = a_1 \cdots a_n$ eine Belegung für F, unter der in jeder Klausel $C_j = \{l_{j1}, \ldots, l_{jk_j}\}$ ein Literal wahr und eines falsch wird.
- Wir können annehmen, dass $l_{j1}(a) = 0$ und $l_{j2}(a) = 1$ ist.
- Dann lässt sich G_F wie folgt mit den 3 Farben $0, 1, 2$ färben:

Knoten v	s	x_i	$\bar{x}_i$	v_{j1}	v_{j2}	v_{j3} (falls $k_j = 3$)
Farbe $c(v)$	2	a_i	$\bar{a}_i$	0	1	2

Färbbarkeit von Graphen

Reduktion von NaeSat auf 3-Coloring

- Sei nun umgekehrt $c : V \to \{0, 1, 2\}$ eine 3-Färbung von G_F.
- Dann können wir annehmen, dass $c(v) = 2$ ist.
- Also gilt für $i = 1, \ldots, n$, dass $\{c(x_i), c(\bar{x}_i)\} = \{0, 1\}$ ist.
- Zudem müssen die Knoten $v_{j1}, \ldots, v_{jk_j}$ im Fall $k_j = 2$ mit 0 und 1 und im Fall $k_j = 3$ mit allen drei Farben 0, 1 und 2 gefärbt sein.
- Wir können annehmen, dass $c(v_{j1}) = 0$ und $c(v_{j2}) = 1$ ist.
- Wegen $\{v_{jk}, \bar{l}_{jk}\} \in E$ muss $c(v_{jk}) \neq c(\bar{l}_{jk})$ für $k = 1, \ldots, k_j$ und daher $c(v_{jk}) = c(l_{jk})$ für $k = 1, 2$ gelten.
- Also macht die Belegung $a = c(x_1) \cdots c(x_n)$ die Literale l_{j1} falsch und l_{j2} wahr.
- Insgesamt gilt also

$$F \in \text{NaeSat} \Leftrightarrow G_F \in \text{3-Coloring}.$$

□

Matchings und der Heiratssatz

Das Heiratsproblem

- Sei V eine Gruppe von heiratswilligen Personen.
- Wir verbinden $u, w \in V$ durch eine Kante, falls aus Sicht von u und w die Möglichkeit einer Heirat zwischen u und w besteht.
- Sind Vielehen ausgeschlossen, so lässt sich jedes mögliche Heiratsarrangement durch ein Matching beschreiben.

Definition

Sei $G = (V, E)$ ein Graph.

- Zwei Kanten $e, e' \in E$ heißen unabhängig, falls $e \cap e' = \emptyset$ ist.
- Eine Kantenmenge $M \subseteq E$ heißt Matching in G, falls die Kanten in M paarweise unabhängig sind.

Matchings und der Heiratssatz

Definition

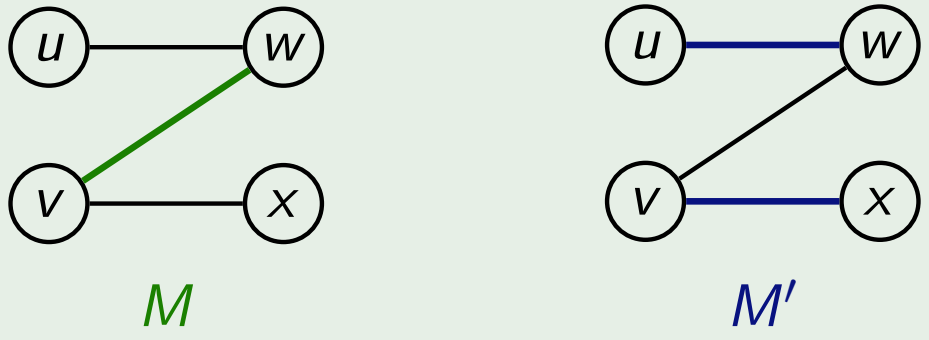
Sei $G = (V, E)$ ein Graph.

- Zwei Kanten $e, e' \in E$ heißen unabhängig, falls $e \cap e' = \emptyset$ ist.
- Eine Kantenmenge $M \subseteq E$ heißt Matching in G, falls die Kanten in M paarweise unabhängig sind.
- Die Matchingzahl von G ist
$$\mu(G) = \max\{\|M\| \mid \ M \text{ ist ein Matching in } G\}.$$
- Ein Matching M der Größe $\|M\| = \mu(G)$ heißt optimal.
- M heißt gesättigt, falls $M \cup \{e\}$ für keine Kante $e \in E \setminus M$ ein Matching ist.
- Analog lassen sich Matchings $M \subseteq E$ in einem d-uniformen Hypergraphen $G = (V, E)$ mit $E \subseteq \binom{V}{d}$ definieren.
- Ein Matching M heißt perfekt, falls $\|M\| \geq \lfloor \|V\|/d \rfloor$ ist.

Matchings und der Heiratssatz

Beispiel

Ein gesättigtes Matching muss nicht optimal sein:



- Das Matching $M = \big\{\{v, w\}\big\}$ ist gesättigt, da es sich nicht zu einem größeren Matching erweitern lässt.
- M ist jedoch nicht optimal, da $M' = \big\{\{u, w\}, \{v, x\}\big\}$ größer ist.
- Die Greedy-Methode, ausgehend von $M = \emptyset$ solange Kanten zu M hinzuzufügen, bis sich M nicht mehr zu einem größeren Matching erweitern lässt, funktioniert also i.a. nicht.

Matchings und der Heiratssatz

Falls beim Heiratsproblem Homoehen ausgeschlossen sind, ist der resultierende Graph $G = (V, E)$ bipartit.

Definition

- Sei $G = (V, E)$ ein Graph und seien $U, W \subseteq V$. Dann bezeichne

$$E(U, W) = \left\{ \{u, w\} \in E \;\middle|\; u \in U, w \in W \right\}$$

  die Menge aller Kanten zwischen U und W.
- G heißt bipartit, falls sich V in zwei Mengen U und W mit $E(U, W) = E$ zerlegen lässt.
- In diesem Fall notieren wir G auch in der Form $G = (U, W, E)$.

Matchingprobleme

d-dimensionales Matching (d-MATCH)

Gegeben: Ein d-uniformer Hypergraph G und eine Zahl $k \geq 1$.

Gefragt: Hat G ein Matching M der Größe $\|M\| \geq k$?

d-dimensionales perfektes Matching (d-PERFECTMATCH)

Gegeben: Ein d-uniformer Hypergraph G.

Gefragt: Hat G ein perfektes Matching?

Im Fall $d = 2$ schreiben wir auch einfach MATCH bzw. PERFECTMATCH.

Bipartites Matching (BIMATCH)

Gegeben: Ein bipartiter Graph G und eine Zahl $k \geq 1$.

Gefragt: Hat G ein Matching M der Größe $\|M\| \geq k$?

Matchingprobleme

Satz

- d-MATCH, d-PERFECTMATCH und BIMATCH sind für $d \leq 2$ in P entscheidbar.
- Für $d \geq 3$ sind d-MATCH und d-PERFECTMATCH NP-vollständig.

3-dimensionales perfektes Matching ist NP-vollständig

Satz

3-PerfectMatch $\in$ NPC.

Beweis

- 3-PerfectMatch $\in$ NP ist klar. Wir reduzieren von 3-Sat.
- Sei $F = \{C_1, \ldots, C_m\}$ eine 3-KNF Formel über $x_1, \ldots, x_n$.
- Betrachte den Hypergraphen $G_F = (V, E)$ mit der Knotenmenge

$$V = \{x_i^j, \bar{x}_i^j, a_i^j, b_i^j, c_j, c_j', g_k, g_k' \mid i \in [n], j \in [m], k \in [m(n-1)]\},$$

wobei $[l] = \{1, \ldots, l\}$ ist, und der Hyperkantenmenge

$$\begin{aligned} E \;= & \{\{x_i^j, a_i^j, b_i^j\}, \{\bar{x}_i^j, b_i^j, a_i^{j+1 \bmod m}\} \mid i \in [n], j \in [m]\} \\ & \cup \{\{x_i^j, c_j, c_j'\} \mid x_i \in C_j\} \cup \{\{\bar{x}_i^j, c_j, c_j'\} \mid \bar{x}_i \in C_j\} \\ & \cup \{\{x_i^j, g_k, g_k'\}, \{\bar{x}_i^j, g_k, g_k'\} \mid i \in [n], j \in [m], k \in [m(n-1)]\}. \end{aligned}$$

Reduktion von 3-SAT auf 3-PERFECTMATCH

- Falls F erfüllbar ist, hat G_F ein (perfektes) Matching M der Größe $2nm$, wovon
  - nm Hyperkanten alle $2nm$ a- und b-Knoten sowie nm der x-Knoten (und zwar alle Knoten x_i^j, falls $x_i = 0$ und alle Knoten $\bar{x}_i^j$, falls $x_i = 1$ ist),
  - m Hyperkanten alle $2m$ c-Knoten sowie weitere m x-Knoten und
  - $m(n-1)$ Hyperkanten alle $2m(n-1)$ g-Knoten sowie die restlichen $m(n-1)$ x-Knoten überdecken.
- Umgekehrt ist auch leicht zu sehen, dass sich aus jedem perfekten Matching M von G eine Belegung a mit $F(a) = 1$ ableiten lässt:
  - Setze die Variable x_i genau dann gleich 1, wenn M Kanten der Form $\{\bar{x}_i^j, b_i^j, a_i^{j+1 \bmod m}\}$ enthält.
  - Da diese Kanten alle falschen Literalknoten überdecken, müssen die Klauselknoten c_j, c_j' durch wahre Literalknoten überdeckt werden.
- Zudem ist leicht zu sehen, dass die Funktion $F \mapsto G_F$ in FP ist. □

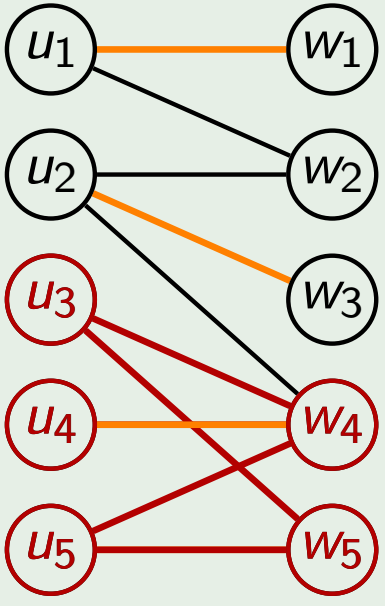
Bipartites Matching und der Heiratssatz

Beispiel

- Der bipartite Graph G enthält ein Matching M der Größe 4.
- Die Nachbarschaft von $A = \{u_3, u_4, u_5\}$ ist

$$N(A) = \{w_4, w_5\}.$$

- Daher kann jedes Matching höchstens 2 der 3 Knoten in A und somit höchstens 4 der 5 Knoten in U überdecken.
- Dies zeigt, dass M optimal ist.



$G = (U, W, E)$

◁

Bipartites Matching und der Heiratssatz

- Sei $A \subseteq U$ beliebig und sei M ein beliebiges Matching.
- Da M höchstens $\|N(A)\|$ Knoten in A überdecken kann, gilt

$$\|M\| \leq \|U - A\| + \|N(A)\| = \|U\| - (\|A\| - \|N(A)\|).$$

- Folglich ist $\mu(G) \leq \|U\| - \max_{A \subseteq U} (\|A\| - \|N(A)\|)$.

Frage: Ist diese Schranke in jedem bipartiten Graphen scharf?

Anders ausgedrückt: Lässt sich die Optimalität von M immer durch Angabe einer Menge $A \subseteq U$ mit $\|U\| - \|M\| = \|A\| - \|N(A)\|$ beweisen?

Satz (Heiratssatz von Hall)

Für einen bipartiten Graphen $G = (U, W, E)$ ist

$$\mu(G) = \|U\| - \max_{A \subseteq U} (\|A\| - \|N(A)\|).$$

Insbesondere hat G genau dann ein Matching der Größe $\|M\| = \|U\|$, wenn für alle Teilmengen $A \subseteq U$ gilt: $\|N(A)\| \geq \|A\|$.

Bipartites Matching und der Heiratssatz

Im Beweis des Heiratssatzes benutzen wir folgende Notation:

Für eine Knotenmenge $A \subseteq V$ und ein Matching M bezeichne

$$A_M^+ = \left\{ u \in A \,\middle|\, \exists v \in V : \{u, v\} \in M \right\}$$

die Menge aller von M in A überdeckten Knoten und

$$A_M^- = A - A_M^+$$

bezeichne die Menge aller von M in A nicht überdeckten Knoten.

Beweis des Heiratssatzes

- Wir konstruieren zu jedem Matching M der Größe

  $$\|M\| < \|U\| - \max_{A \subseteq U}(\|A\| - \|N(A)\|)$$

  ein Matching M' mit $\|M'\| > \|M\|$.
- Hierzu betrachten wir den gerichteten Graphen G_M, der aus G dadurch entsteht, dass wir alle Matchingkanten in M von W nach U und alle übrigen Kanten von U nach W orientieren.
- Angenommen, es ex. in G_M ein Pfad von U_M^- nach W_M^-:

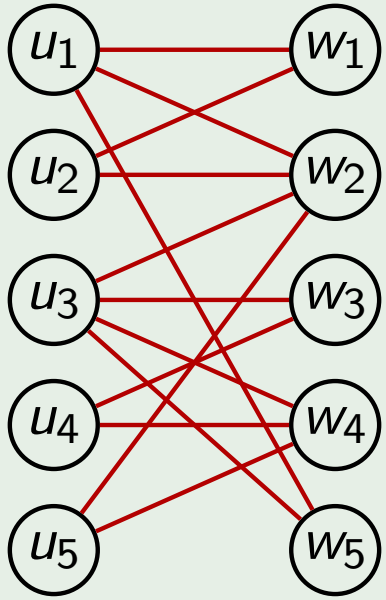
  $$P: \; u_0 \to w_0 \to u_1 \to w_1 \to \cdots u_{k-1} \to w_{k-1} \to u_k \to w_k$$

- Dann hat P ungerade Länge $2k+1$, $k \geq 0$, und wir können M zu M' vergrößern, indem wir die k zu P gehörigen Matchingkanten aus M entfernen und dafür die $k+1$ übrigen Kanten von P zu M hinzufügen:

  $$M' = \left(M \setminus \left\{\{w_i, u_{i+1}\} \,\middle|\, 0 \leq i < k\right\}\right) \cup \left\{\{u_i, w_i\} \,\middle|\, 0 \leq i \leq k\right\}.$$

Beweis des Heiratssatzes

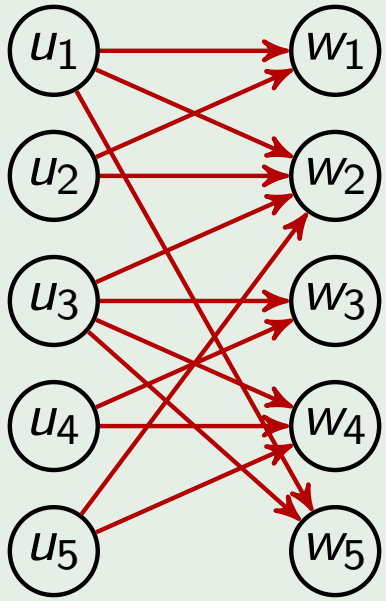
Beispiel



$$G = (U, W, E)$$

Beweis des Heiratssatzes

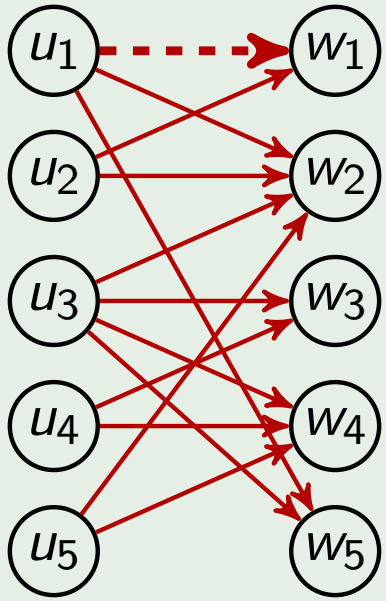
Beispiel



G_M für $M = \emptyset$

Beweis des Heiratssatzes

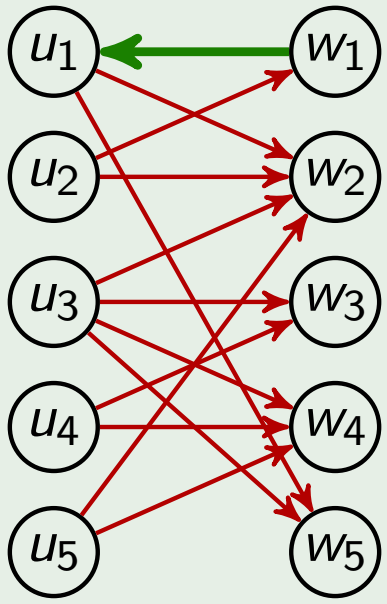
Beispiel



G_M für $M = \emptyset$

Beweis des Heiratssatzes

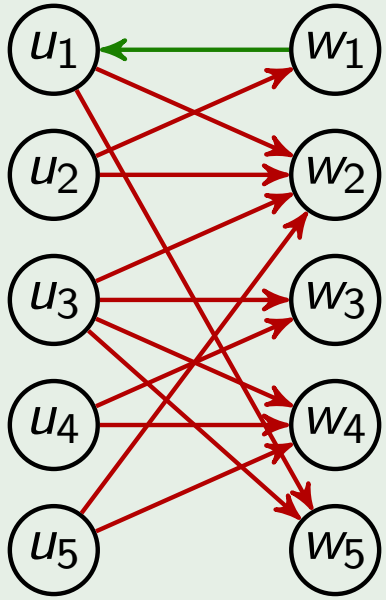
Beispiel



G_M für $M = \{\{u_1, w_1\}\}$

Beweis des Heiratssatzes

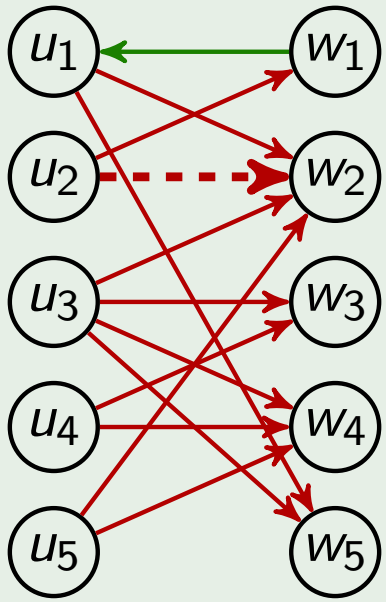
Beispiel



G_M für $M = \{\{u_1, w_1\}\}$

Beweis des Heiratssatzes

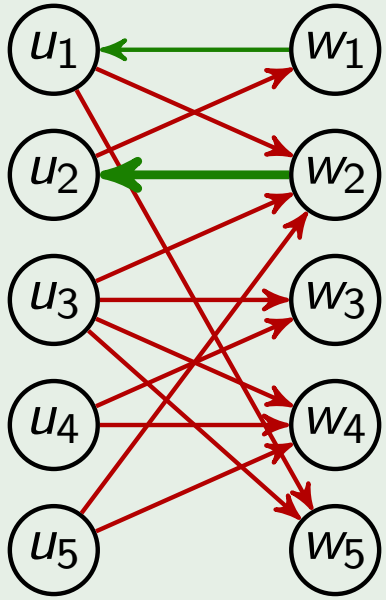
Beispiel



G_M für $M = \{\{u_1, w_1\}\}$

Beweis des Heiratssatzes

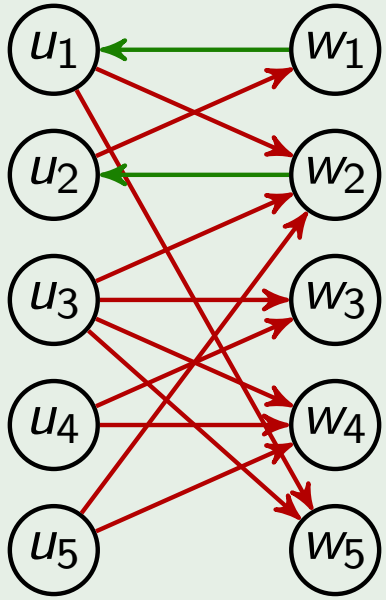
Beispiel



G_M für $M = \{\{u_1, w_1\}, \{u_2, w_2\}\}$

Beweis des Heiratssatzes

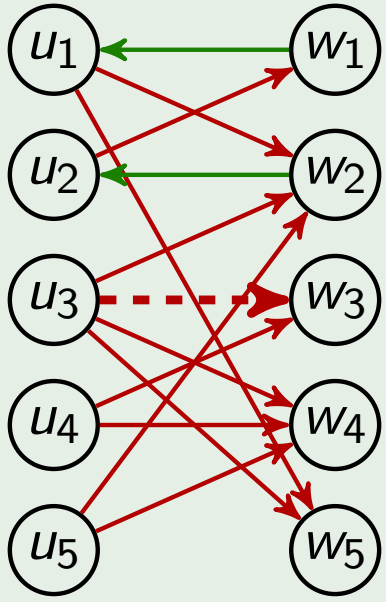
Beispiel



G_M für $M = \{\{u_1, w_1\}, \{u_2, w_2\}\}$

Beweis des Heiratssatzes

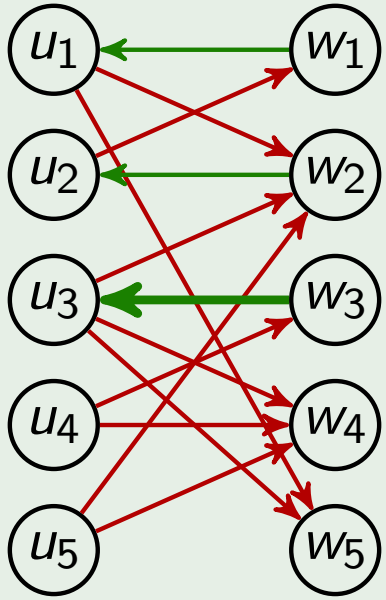
Beispiel



G_M für $M = \{\{u_1, w_1\}, \{u_2, w_2\}\}$

Beweis des Heiratssatzes

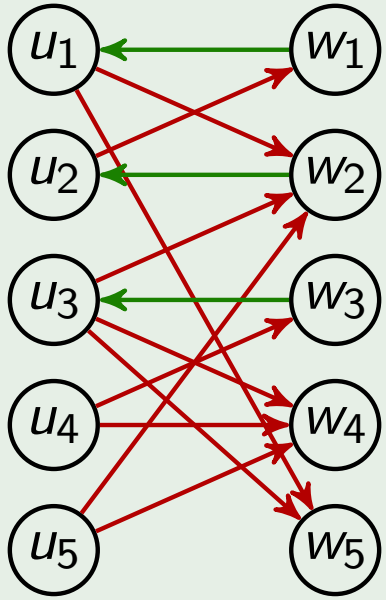
Beispiel



G_M für $M = \{\{u_1, w_1\}, \{u_2, w_2\}, \{u_3, w_3\}\}$

Beweis des Heiratssatzes

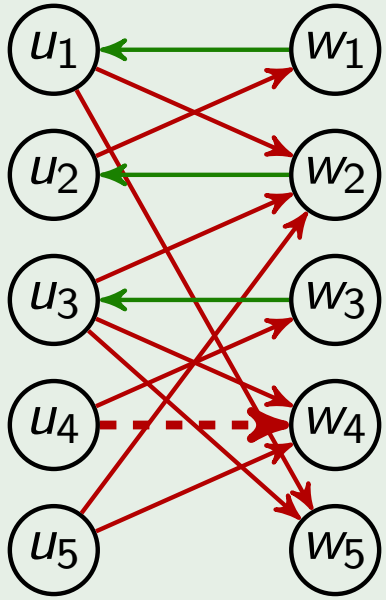
Beispiel



G_M für $M = \{\{u_1, w_1\}, \{u_2, w_2\}, \{u_3, w_3\}\}$

Beweis des Heiratssatzes

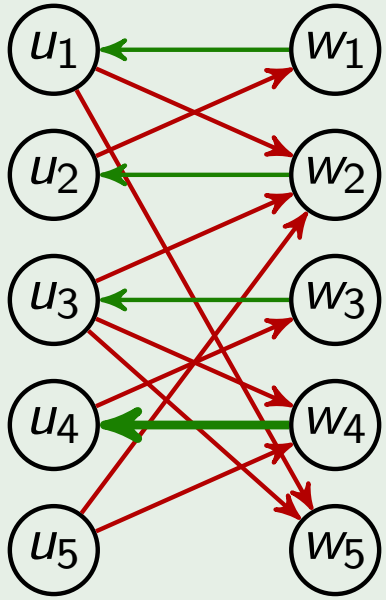
Beispiel



G_M für $M = \{\{u_1, w_1\}, \{u_2, w_2\}, \{u_3, w_3\}\}$

Beweis des Heiratssatzes

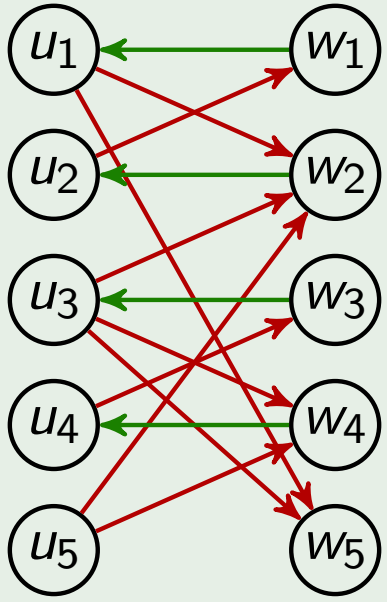
Beispiel



G_M für $M = \{\{u_1, w_1\}, \{u_2, w_2\}, \{u_3, w_3\}, \{u_4, w_4\}\}$

Beweis des Heiratssatzes

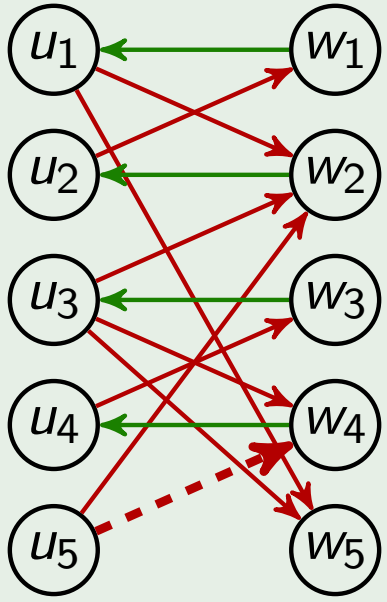
Beispiel



G_M für $M = \{\{u_1, w_1\}, \{u_2, w_2\}, \{u_3, w_3\}, \{u_4, w_4\}\}$

Beweis des Heiratssatzes

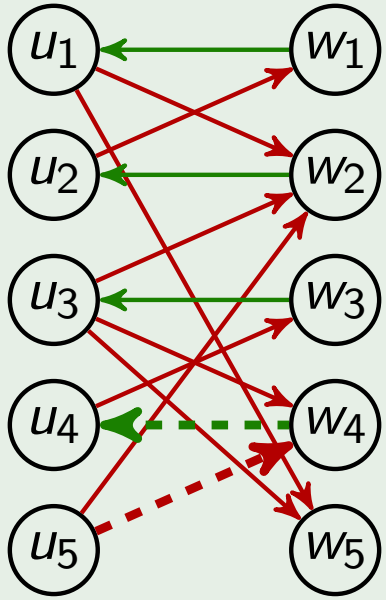
Beispiel



G_M für $M = \{\{u_1, w_1\}, \{u_2, w_2\}, \{u_3, w_3\}, \{u_4, w_4\}\}$

Beweis des Heiratssatzes

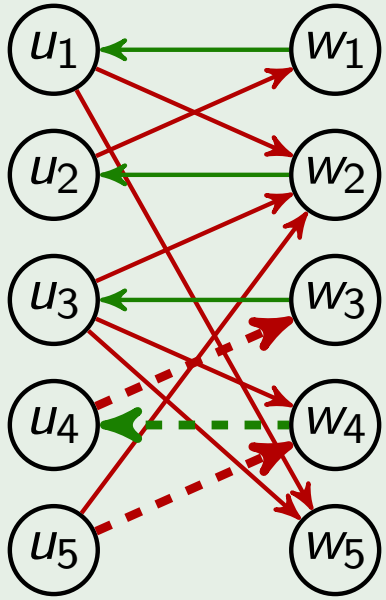
Beispiel



G_M für $M = \{\{u_1, w_1\}, \{u_2, w_2\}, \{u_3, w_3\}, \{u_4, w_4\}\}$

Beweis des Heiratssatzes

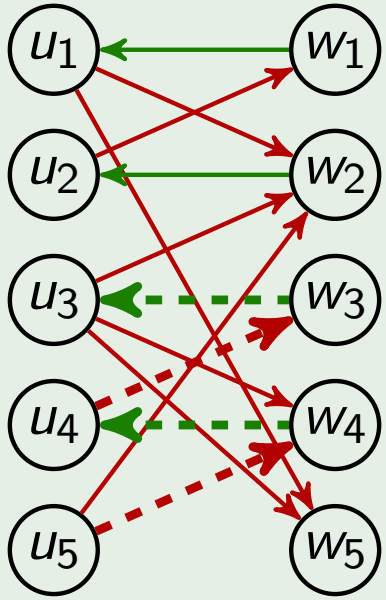
Beispiel



G_M für $M = \{\{u_1, w_1\}, \{u_2, w_2\}, \{u_3, w_3\}, \{u_4, w_4\}\}$

Beweis des Heiratssatzes

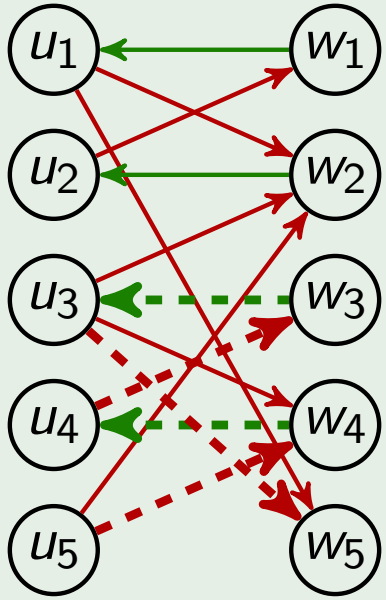
Beispiel



G_M für $M = \{\{u_1, w_1\}, \{u_2, w_2\}, \{u_3, w_3\}, \{u_4, w_4\}\}$

Beweis des Heiratssatzes

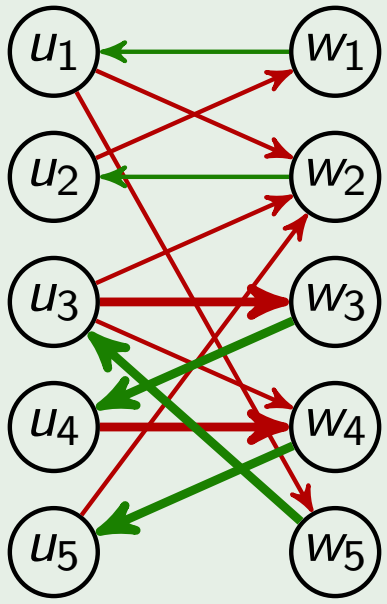
Beispiel



G_M für $M = \{\{u_1, w_1\}, \{u_2, w_2\}, \{u_3, w_3\}, \{u_4, w_4\}\}$

Beweis des Heiratssatzes

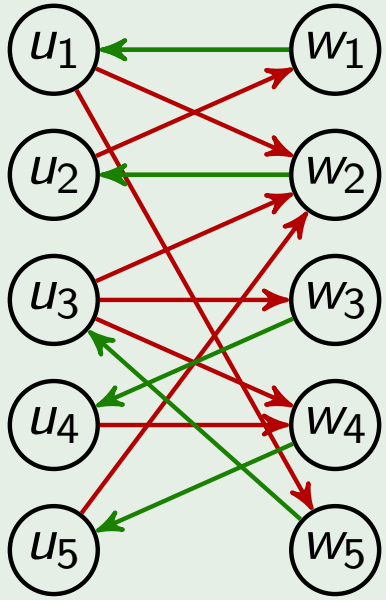
Beispiel



G_M für $M = \{\{u_1, w_1\}, \{u_2, w_2\}, \{u_3, w_5\}, \{u_4, w_3\}, \{u_5, w_4\}\}$

Beweis des Heiratssatzes

Beispiel



G_M für $M = \{\{u_1, w_1\}, \{u_2, w_2\}, \{u_3, w_5\}, \{u_4, w_3\}, \{u_5, w_4\}\}$

◁

Beweis des Heiratssatzes

- Noch zu zeigen: Falls es in G_M keinen Pfad von U_M^- nach W_M^- gibt, dann lässt sich die Optimalität von M durch Angabe einer Menge $A \subseteq U$ mit $\|U\| - \|M\| = \|A\| - \|N(A)\|$ beweisen.
- Sei X die Menge aller in G_M von U_M^- aus erreichbaren Knoten.
- Da in G_M kein Pfad von U_M^- nach W_M^- ex., ist $X \cap W \subseteq W_M^+$.
- Wir zeigen, dass $A = X \cap U$ die Größe $\|U\| - \|M\| + \|N(A)\|$ hat.
- Da A alle Knoten in U_M^- enthält, ist

$$\|A\| = \|U_M^-\| + \|A_M^+\| = \|U\| - \|M\| + \|A_M^+\|.$$

- Es reicht also zu zeigen, dass $\|A_M^+\| = \|N(A)\|$ ist.
- Es ist klar, dass jeder Knoten $w \in N(A)$, der mit einem Knoten $u \in A$ über eine Kante $e = \{u, w\} \in E \setminus M$ verbunden ist, zu X gehört.
- Aber auch im Fall $e \in M$ muss w zu X gehören, da u in G_M nur über diese Matchingkante e erreichbar ist.
- Also ist $N(A) \subseteq X \cap W \subseteq W_M^+$, d.h jeder Knoten $w \in N(A)$ hat einen Matching-Partner u in A, woraus $\|N(A)\| = \|A_M^+\|$ folgt. $\square$

Bipartites Matching und der Heiratssatz

Korollar (zum Beweis des Heiratssatzes)

In bipartiten Graphen ist ein optimales Matching in Polynomialzeit berechenbar, d.h. $\textsc{BiMatch} \in \mathrm{P}$.

Zur Erinnerung: Für ein Matching M in G ist G_M der Digraph, der aus G dadurch entsteht, dass wir alle Matchingkanten in M von W nach U und die übrigen Kanten von U nach W orientieren.

Algorithmus zur Bestimmung eines optimalen Matchings

Input: Ein bipartiter Graph $G = (U, W, E)$
  $M := \emptyset$
  while $\exists$ Pfad P von U_M^- nach W_M^- in G_M **do**
    sei $P : u_0 \to w_0 \to u_1 \to \cdots \to w_{k-1} \to u_k \to w_k$
    $M := \left(M \setminus \left\{\{w_i, u_{i+1}\} \,\middle|\, 0 \leq i < k\right\}\right) \cup \left\{\{u_i, w_i\} \,\middle|\, 0 \leq i \leq k\right\}$
Output: M

Frage: Wie lässt sich P effizient berechnen?

Notation – gerichtete Graphen

Definition

- Ein gerichteter Graph oder Digraph ist ein Paar $G = (V, E)$, wobei

  V - eine endliche Menge von Knoten/Ecken und

  E - die Menge der Kanten ist.

  Hierbei gilt

  $$E \subseteq V \times V = \{(u, v) \mid u, v \in V\},$$

  wobei E auch Schlingen (u, u) enthalten kann.
- Die Nachfolgermenge von v ist $N^+(v) = \{u \in V \mid (v, u) \in E\}$.
- Die Vorgängermenge von v ist $N^-(v) = \{u \in V \mid (u, v) \in E\}$.
- Die Nachbarmenge von v ist $N(v) = N^+(v) \cup N^-(v)$.
- Der Ausgangsgrad von v ist $\deg^+(v) = \|N^+(v)\|$.
- Der Eingangsgrad von v ist $\deg^-(v) = \|N^-(v)\|$.
- Der Grad von v ist $\deg(v) = \deg^+(v) + \deg^-(v)$.

Gerichtete Zyklen und Kreise

Definition

Sei $G = (V, E)$ ein Digraph.

- Ein (gerichteter) v_0-v_j-Weg in G ist eine Folge von Knoten $(v_0, \ldots, v_j)$ mit $(v_i, v_{i+1}) \in E$ für $i = 0, \ldots, j-1$.
- Ein (gerichteter) Zyklus in G ist ein gerichteter u-v-Weg der Länge $j \geq 1$ mit $u = v$.
- Ein gerichteter Weg heißt einfach oder (gerichteter) Pfad, falls alle durchlaufenen Knoten paarweise verschieden sind.
- Ein (gerichteter) Kreis in G ist ein gerichteter Zyklus $(v_0, \ldots, v_{j-1}, v_0)$ der Länge $j \geq 1$, für den $v_0, \ldots, v_{j-1}$ paarweise verschieden sind.
- G heißt azyklisch, wenn es in G keinen gerichteten Kreis gibt.

Eulerlinien

Definition

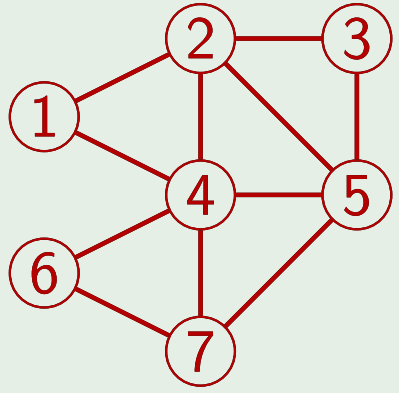
Sei $G = (V, E)$ ein Graph (oder Digraph) und sei $s = (v_0, v_1, \ldots, v_l)$ ein (gerichteter) Weg in G, d.h. $\{v_i, v_{i+1}\}$ bzw. $(v_i, v_{i+1}) \in E$ für $i = 0, \ldots, l-1$.

- $s = (v_0, v_1, \ldots, v_l)$ heißt Eulerlinie (auch Eulerzug oder Eulerweg) in G, falls s jede Kante in E genau einmal durchläuft, d.h. es gilt $l = \|E\|$ und

$$\left\{ \{v_i, v_{i+1}\} \text{ bzw. } (v_i, v_{i+1}) \;\middle|\; i = 0, \ldots, l-1 \right\} = E.$$

Beispiel (Eulerlinie in einem ungerichteten Graphen)

$$s = (4, 1, 2, 3, 5, 7, 6, 4, 5, 2, 4, 7)$$



Eulerkreise

Definition

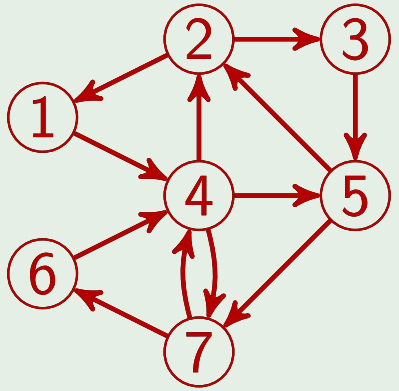
- $s = (v_0, v_1, \ldots, v_l)$ in G heißt Eulerlinie (auch Eulerzug oder Eulerweg), falls s jede Kante in E genau einmal durchläuft, d.h. es gilt $l = \|E\|$ und

$$\left\{ \{v_i, v_{i+1}\} \text{ bzw. } (v_i, v_{i+1}) \;\middle|\; i = 0, \ldots, l-1 \right\} = E.$$

- Ist s zudem ein Zyklus, d.h. es gilt $v_l = v_0$, so heißt s Eulerkreis (auch Eulerzyklus oder Eulertour).

Beispiel (Eulerkreis in einem Digraphen)

$$s = (1, 4, 5, 2, 3, 5, 7, 4, 7, 6, 4, 2, 1)$$



Hamiltonpfade

Definition

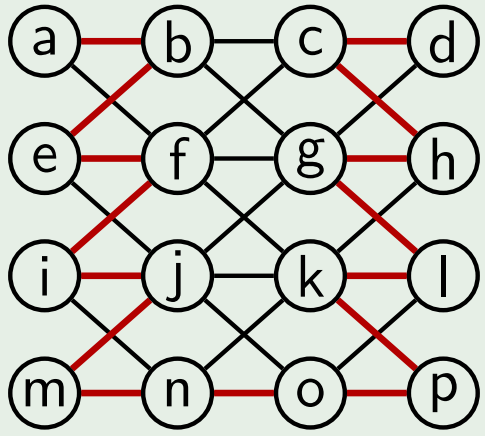
Sei $G = (V, E)$ ein Graph (oder Digraph) und sei $s = (v_0, v_1, \ldots, v_l)$ ein (gerichteter) Pfad in G, d.h. $\{v_i, v_{i+1}\}$ bzw. $(v_i, v_{i+1}) \in E$ für $i = 0, \ldots, l-1$ und die Knoten $v_0, \ldots, v_l$ sind paarweise verschieden.

- s heißt Hamiltonpfad in G, falls s jeden Knoten in V genau einmal durchläuft, d.h. es gilt

$$V = \{v_0, \ldots, v_l\} \text{ und } l = \|V\| - 1.$$

Beispiel (Hamiltonpfad in einem ungerichteten Graphen)

$$s = (a, b, e, f, i, j, m, n, o, p, k, l, g, h, c, d)$$



Hamiltonpfade

Definition

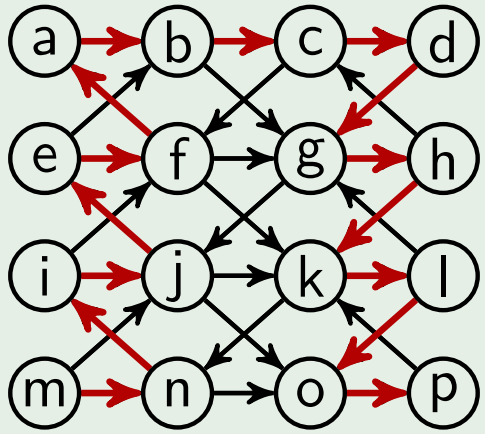
Sei $G = (V, E)$ ein Graph (oder Digraph) und sei $s = (v_0, v_1, \ldots, v_l)$ ein (gerichteter) Pfad in G, d.h. $\{v_i, v_{i+1}\}$ bzw. $(v_i, v_{i+1}) \in E$ für $i = 0, \ldots, l-1$ und die Knoten $v_0, \ldots, v_l$ sind paarweise verschieden.

- s heißt Hamiltonpfad in G, falls s jeden Knoten in V genau einmal durchläuft, d.h. es gilt

$$V = \{v_0, \ldots, v_l\} \text{ und } l = \|V\| - 1.$$

Beispiel (Hamiltonpfad in einem Digraphen)

$$s = (m, n, i, j, e, f, a, b, c, d, g, h, k, l, o, p)$$



Hamiltonkreise

Definition

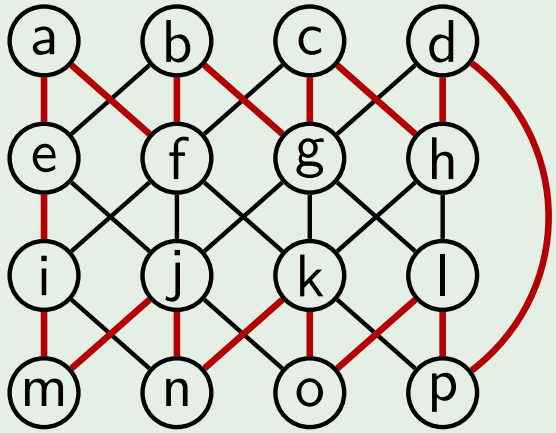
- $s = (v_0, v_1, \ldots, v_l)$ in G heißt Hamiltonpfad, falls s jeden Knoten in V genau einmal durchläuft, d.h. es gilt

$$V = \{v_0, \ldots, v_l\} \text{ und } l = \|V\| - 1.$$

- Ist zudem $\{v_0, v_l\} \in E$, d.h. $s' = (v_0, v_1, \ldots, v_l, v_0)$ ist ein (gerichteter) Kreis, so heißt s' Hamiltonkreis.

Beispiel (Hamiltonkreis in einem ungerichteten Graphen)

$s = (a, f, b, g, c, h, d, p, l, o, k, n, j, m, i, e, a)$



Euler- und Hamiltonkreise

Definition

Wir betrachten für einen gegebenen Graphen (bzw. Digraphen) G und zwei Knoten s und t folgende Entscheidungsprobleme:

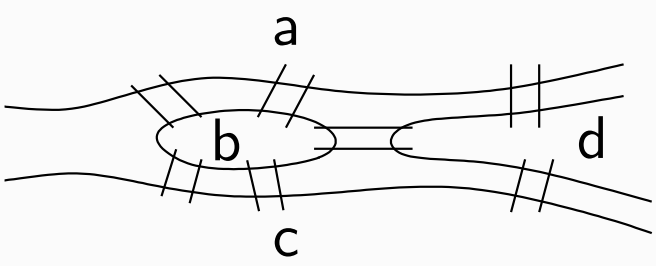
- Das Eulerlinienproblem (EulerPath bzw. DiEulerPath):
  Hat G eine Eulerlinie von s nach t?
- Das Hamiltonpfadproblem (HamPath bzw. DiHamPath):
  Hat G einen Hamiltonpfad von s nach t?

Zudem betrachten wir für einen gegebenen Graphen (bzw. Digraphen) G folgende Probleme:

- Das Eulerkreisproblem (EulerCycle bzw. DiEulerCycle):
  Hat G einen Eulerkreis?
- Das Hamiltonkreisproblem (HamCycle bzw. DiHamCycle):
  Hat G einen Hamiltonkreis?

Das Königsberger Brückenproblem

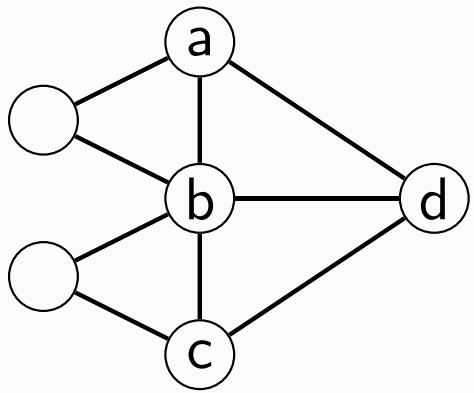
Die 7 Königsberger Brücken



Frage

Gibt es einen Spaziergang über alle 7 Brücken, bei dem keine Brücke mehrmals überquert wird und der zum Ausgangspunkt zurückführt?

Gelöst von Euler (1707 – 1783) durch Betrachtung des folgenden Graphen, der offenbar genau dann einen Eulerkreis hat, wenn die Antwort auf obige Frage „ja“ ist.



Satz (Euler, 1736)

Ein zusammenhängender Graph $G = (V, E)$ besitzt genau dann einen Eulerkreis, wenn all seine Knoten geraden Grad haben.

Eulerkreise und -linien

Satz (Euler, 1736)

Ein zusammenhängender Graph $G = (V, E)$ besitzt genau dann einen Eulerkreis, wenn all seine Knoten geraden Grad haben.

Beweis

- Falls G einen Eulerkreis s besitzt, existiert zu jeder Kante, auf der s zu einem Knoten gelangt, eine weitere Kante, auf der s den Knoten wieder verlässt.
- Daher muss jeder Knoten geraden Grad haben.
- Ist umgekehrt G zusammenhängend und hat jeder Knoten geraden Grad, so können wir wie folgt einen Eulerkreis s konstruieren.

Eulerkreise und -linien

- Ist umgekehrt G zusammenhängend und hat jeder Knoten geraden Grad, so können wir wie folgt einen Eulerkreis s konstruieren.

Algorithmus zur Berechnung eines Eulerkreises

1 **Input:** Graph $G = (V, E)$
2 Wähle $u \in V$ beliebig und initialisiere s zu $s = (u)$
3 Wähle einen beliebigen Knoten u auf dem Weg s, der mit einer unmarkierten Kante verbunden ist.
4 Folge ausgehend von u den unmarkierten Kanten auf einem beliebigen Weg z solange wie möglich und markiere dabei jede durchlaufene Kante.
(Da von jedem erreichten Knoten $v \neq u$ ungerade viele markierte Kanten ausgehen, muss der Weg z zum Ausgangspunkt u zurückführen.)
5 Füge den Zyklus z an der Stelle u in s ein.
6 Falls noch nicht alle Kanten markiert sind, gehe zu 3.
7 **Output:** s

□

Eulerkreise und -linien

Satz (Euler, 1736)

i) Ein zusammenhängender Graph $G = (V, E)$ besitzt genau dann eine Eulerlinie von s nach t, wenn s und t ungeraden Grad und alle übrigen Knoten geraden Grad haben.

ii) Ein stark zusammenhängender Digraph besitzt genau dann einen Eulerkreis, wenn für jeden Knoten u der Eingangs- und der Ausgangsgrad übereinstimmen.

Beweis

i) Da G genau dann eine Eulerlinie von s nach t hat, wenn der Graph $G' = (V', E')$ mit $V' = V \cup \{u_{neu}\}$ und $E' = E \cup \{\{t, u_{neu}\}, \{u_{neu}, s\}\}$ einen Eulerkreis hat, folgt dies aus dem vorigen Satz.

ii) Dies folgt vollkommen analog zum ungerichteten Fall.

Korollar

$\textsc{EulerCycle}, \textsc{EulerPath}, \textsc{DiEulerCycle}, \textsc{DiEulerPath} \in \mathsf{P}$.

Hamiltonkreise

Satz

HamPath, HamCycle, DiHamPath und DiHamCycle sind NP-vollständig.

Bemerkung

Bevor wir den Satz beweisen, betrachten wir ein weiteres Problem, das mit dem Hamiltonkreisproblem eng verwandt ist und große praktische Bedeutung hat.

Das Problem des Handlungsreisenden

- Gegeben sind die Entfernungen d_{ij} zwischen n Städten $i, j \in \{1, \ldots, n\}$.
- Gesucht ist eine Rundreise $(i_1, \ldots, i_n)$ mit minimaler Länge $d_{i_1,i_2} + \cdots + d_{i_{n-1},i_n} + d_{i_n,i_1}$, die jede Stadt genau einmal besucht.
- Die Entscheidungsvariante dieses Optimierungsproblems ist wie folgt definiert.

Problem des Handlungsreisenden (TSP; *traveling salesman problem*)

Gegeben: Eine $n \times n$ Matrix $D = (d_{i,j}) \in \mathbb{N}^{n \times n}$ und eine Zahl k.

Gefragt: Existiert eine Permutation $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$, so dass die Rundreise $(\pi(1), \ldots, \pi(n))$ die Länge $\leq k$ hat?

Satz

$$\text{3-Sat} \leq^p \text{DiHamPath} \leq^p \begin{matrix}\text{HamPath,}\\ \text{DiHamCycle}\end{matrix} \leq^p \text{HamCycle} \leq^p \text{TSP}.$$

Hamiltonkreise und -pfade

Satz

$$\text{3-SAT} \leq^p \text{DIHAMPATH} \leq^p \begin{matrix}\text{HAMPATH,}\\ \text{DIHAMCYCLE}\end{matrix} \leq^p \text{HAMCYCLE} \leq^p \text{TSP}.$$

Reduktion von HAMCYCLE auf TSP

- Sei $G = (V, E)$ ein Graph, wobei wir $V = \{1, \ldots, n\}$ annehmen.
- Dann lässt sich G in Polynomialzeit auf die TSP Instanz (D, n) mit $D = (d_{i,j})$ und

$$d_{i,j} = \begin{cases} 1, & \text{falls } \{i,j\} \in E, \\ 2, & \text{sonst,} \end{cases}$$

  transformieren.
- Diese Reduktion ist korrekt, da G genau dann einen Hamiltonkreis hat, wenn es in dem Distanzgraphen D eine Rundreise $(\pi(1), \ldots, \pi(n))$ der Länge $L(\pi) \leq n$ gibt. □