1 Einleitung

1.1 NP-Vollständigkeit

Eine Sprache B ist **NP-Vollständig** wenn gilt:

1. $B \in NP$
2. $\forall A \in NP : A \prec_p B$

Notiz:
$A \prec_p B$: A ist polynomialzeitreduzierbar auf B

1.2 Polynomialzeitreduktion

Eine Sprache A ist polynomialzeitreduzierbar auf Sprache B, $A \prec_p B$, wenn eine in polynomialer Zeit berechenbare Funktion $f : \Sigma^* \to \Sigma^*$ existiert, für die gilt:
$\forall w : w \in A \iff f(w) \in B$

Die Funktion f heißt dann Polynomialzeitreduktion von A nach B.

1.3 Definition 3SAT

Spezialform des Erfüllbarkeitsproblems

- Literal:
  x_i oder $\overline{x_i}$
- Variable:
  x_i (l: Anzahl der Variablen)
- Klausel:
  $(x_1 \vee \overline{x_2} \vee \overline{x_3} \vee x_4)$ (k: Anzahl der Klauseln)
- CNF-Formel(cnf-formula - conjunctive normal form):
  $(x_1 \vee \overline{x_2} \vee \overline{x_3} \vee x_4) \wedge (x_3 \vee \overline{x_5} \vee x_6) \wedge (x_3 \vee \overline{x_6})$
- ϕ: $(x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_2} \vee x_4 \vee ...) \wedge ... \wedge (\overline{x_3} \vee ... \vee ...)$

$3SAT = \{\ \langle\phi\rangle | \phi$ ist eine erfüllbare 3CNF-Formel$\}$

$\phi = 1 \iff \forall c_j$: mindestens ein Literal ist $true$

2 Hamilton-Pfad-Problem

2.1 Definition

- Geht durch jeden Knoten genau einmal
- Startet in s
- Endet in t

Notiz: Gesucht: Pfad von s nach t, der durch jeden knoten genau einmal geht.

2.2 Beweis Gerichtet

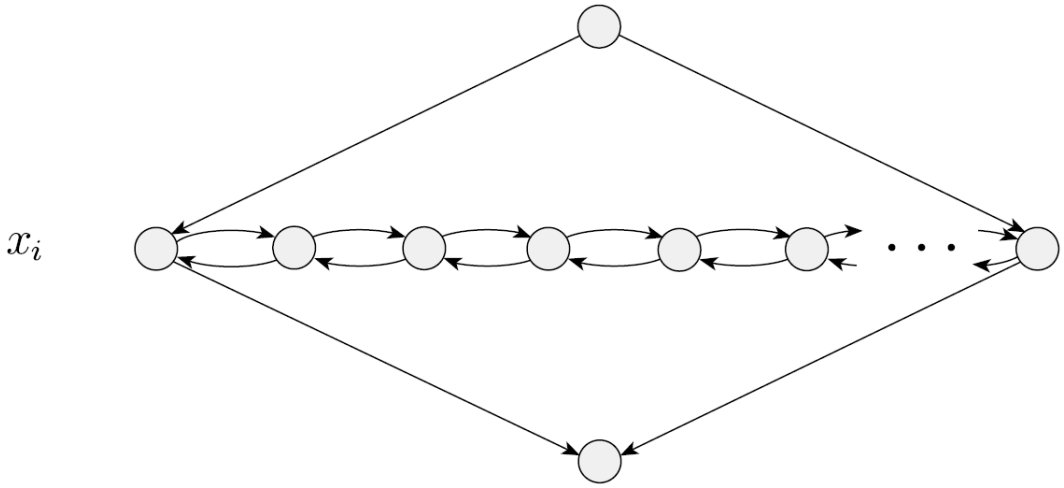
2.2.1 HAMPATH $\in$ NP

Notiz: Eine nichtdeterministische Turingmaschine rät das Ergebnis und Verwirft dann bzw. nimmt an.

2.2.2 Konstruktion von G

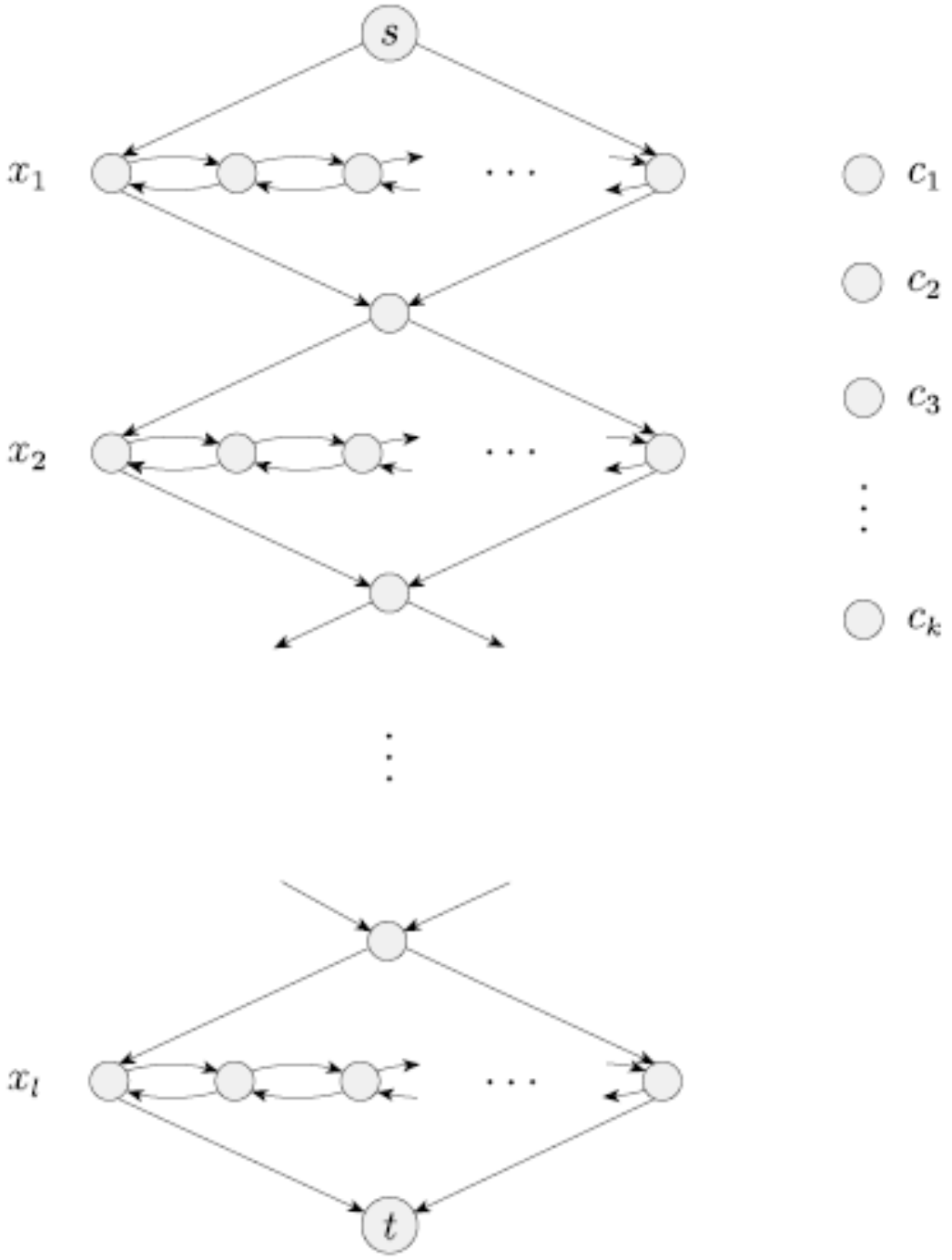
$\phi \rightarrow G$

Darstellung Variable x_i

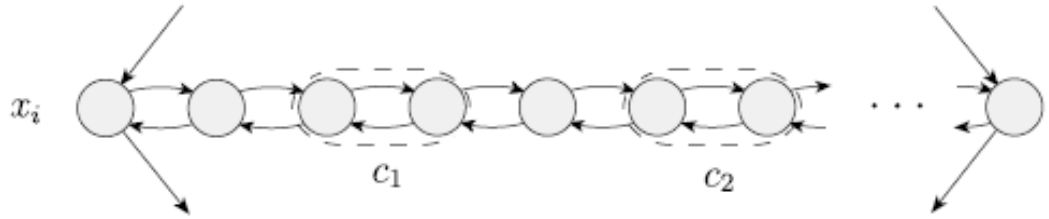


Darstellung Klausel c_j

High-level structure of G

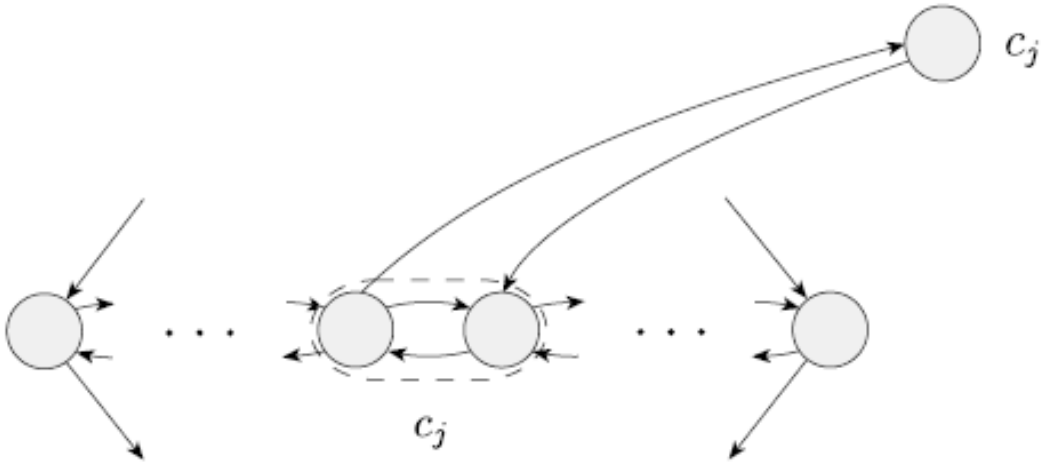


Horizontale Struktur im Diamant



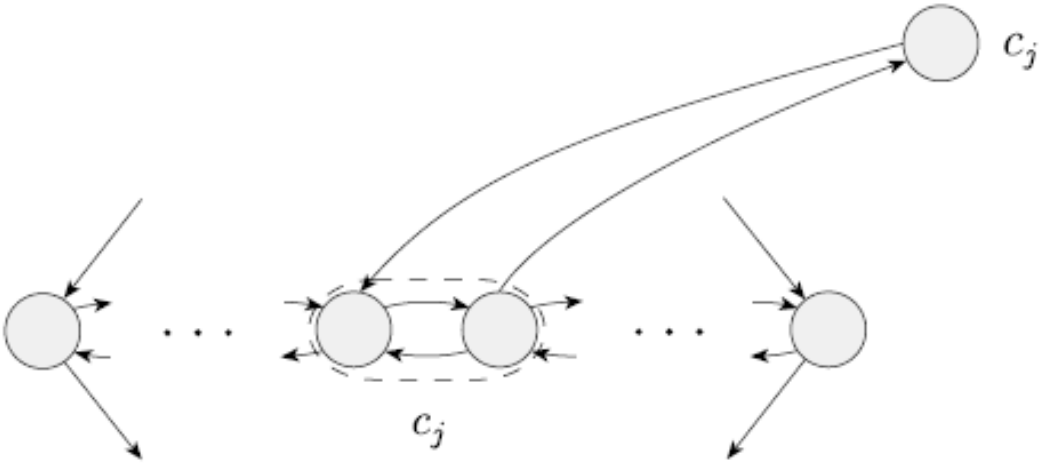
$\langle 3k + 1 \rangle$ Knoten

Zusätzliche Kanten wenn x_i in c_j ist



Notiz: Am Beispiel von ϕ zeigen!

Zusätzliche Kanten wenn $\overline{x_i}$ in c_j ist



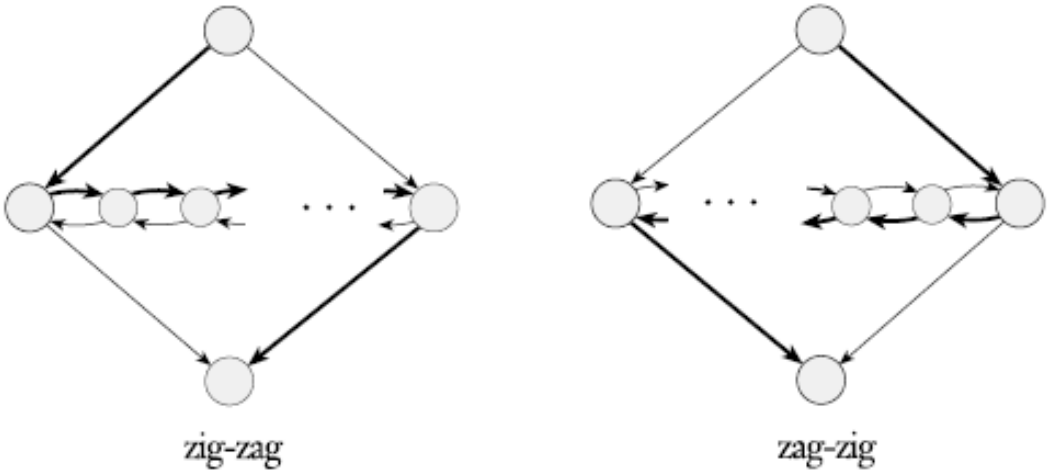
2.2.3 ϕ ist erfüllbar

Notiz: Zunächst werden die Knoten c_j ignoriert.

Notiz: Der Pfad geht von s nach t durch jeden Diamanten nach einander.

Notiz: Um alle Knoten zu treffen muss der Pfad entweder zig-zaggen oder zag-ziggen.

Zig-zagging and Zag-zigging



Notiz: Zig-zagging wenn Variable $x_i = true$, Zag-zigging wenn Variable $x_i = false$

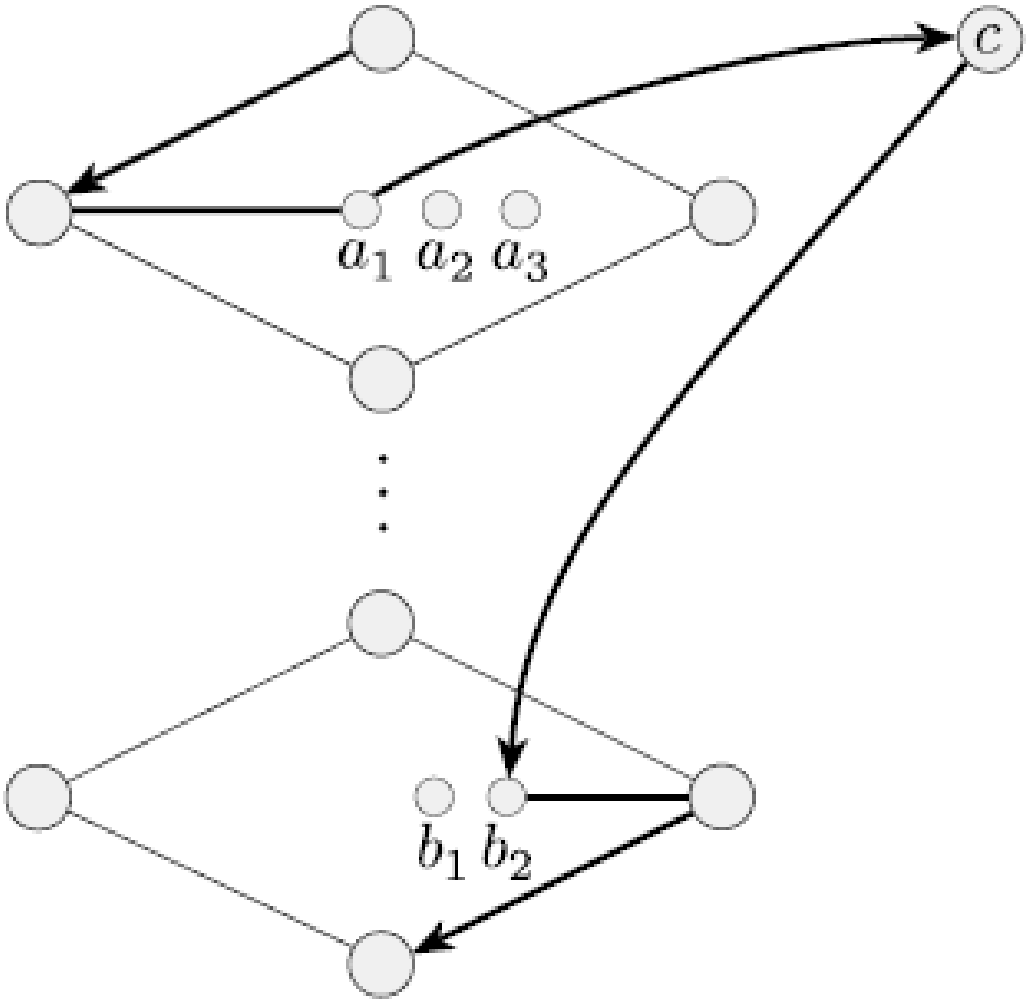
Notiz: Jetzt fehlen nurnoch die Knoten c_j.

Notiz: In jeder Klausel c_j wählen wir einen Literal aus, dem wir $true$ zuweisen.

Notiz: Jeder wahre Literal in einer Klausel ist nur eine Option für einen Umweg über einen Klauselnoten. $\rightarrow$ Es wird immer nur ein Umweg zu jedem Klauselknoten genommen. $\rightarrow$ Konstruktion von G ist beendet.

2.2.4 G hat einen HAMPATH

- HAMPATH muss normal sein, geht durch Diamanten von oben nach unten(ausgenommen von den Umwegen über die Klauselknoten
- zig-zag $\rightarrow x_i = true$, zag-zig $\rightarrow x_i = false$
- Anhand dessen, wie der Umweg über einen Klauselknoten genommen wird kann man erkennen welcher Literal der Klausel wahr ist
- Hampath ist normal zeigen



Laufzeit

Notiz: Anhand des Graphen erläutern(Anzahl Knoten)

2.2.5 Ungerichtet

TODO: muss der auch rein?

3 SUBSET-SUM-Problem

3.1 Definition

- Integer-Arithmetik (dezimal)
- Menge von Zahlen S: $x_1, ..., x_k$
- Target t
- Kann t durch ein Subset von S erreicht werden?

3.2 Beweis

3SAT $\prec_p$ SUBSET-SUM

ϕ: $(x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_2} \vee x_4 \vee ...) \wedge ... \wedge (\overline{x_3} \vee ... \vee ...)$

3.2.1 SUBSET-SUM $\in$ NP

Notiz: Eine nichtdeterministische Turingmaschine rät das Ergebnis und Verwirft dann bzw. nimmt an.

3.2.2 Konstruktion der Tabelle

- Elemente aus S und die Zahl t sind die Zeilen der Tabelle (in Dezimaldarstellung)
- Die Zeilen oberhalb des Doppellinie sind $y_1, z_1, y_2, z_2, ..., y_l, z_l$ und $g_1, h_1, g_2, h_2, ..., g_k, h_k$
- Unter der Doppellinie ist t, l 1 gefolgt von k 3
- S hat ein Paar y_i, z_i für jede Variable x_i in ϕ
- linke Seite: 1 gefolgt von $l - i$ 0
- rechte Seite: $1 \iff c_j\ x_i$ beinhaltet, $0 \iff c_j\ \overline{x_i}$ beinhaltet
- nicht gefüllte Stellen sind 0

- g_j, h_j für jede Klausel c_j
- $g_j = h_j$, 1 gefolgt von $k - j$ 0
- Kein Carry möglich

	1	2	3	4	...	1	c_1	c_2	...	c_k
y_1	1	0	0	0	...	0	1	0	...	0
z_1	1	0	0	0	...	0	0	0	...	0
y_2		1	0	0	...	0	0	0	...	0
z_2		1	0	0	...	0	1	1	...	0
y_3			1	0	...	0	0	0	...	0
z_3			1	0	...	0	1	0	...	1
...					...				...	
y_l						1	0	0	...	0
z_l						1	0	0	...	0
g_1						1	0	0	...	0
h_1						1	0	0	...	0
g_2							1	0	...	0
h_2							1	0	...	0
...					...				...	
g_k										1
h_k										1
t	1	1	1	1	...	1	3	3	...	3

3.2.3 Annahme

Es existiert eine Konfiguration für die ϕ erfüllt ist.

- Wähle y_i wenn Variable x_i wahr ist, ansonsten z_i
- Summe ergibt nur 1 auf der linken Seite, auf der rechten Seite nur 1 bis 3
- g und h wählen bis t erreicht

3.2.4 Annahme

Ein Subset von S ergibt t.

- y_i oder z_i um 1 für linke Seite zu bekommen
- Wenn y_i im Subset dann $x_i = true$, ansonsten $false$
- Summe der rechten Seite ist immer 3, maximal 2 können von g und h kommen, daher alle c_j wahr

- Wenn y_i dann ist x_i in c_j und wird $true$ gesetzt, wenn z_i dann $\overline{x_i}$ in c_j und x_i wird aus $false$ gesetzt
- Daher ist ϕ wahr

3.2.5 Laufzeit

Tabelle hat etwa eine Größe von $(l + k)^2$
Jeder Eintrag ist leicht zu berechnen $\Rightarrow O(n^2)$