Weitere NP-vollständige Probleme

Proseminar Theoretische Informatik

Marten Tilgner

December 10, 2014

Wir haben letzte Woche gesehen, dass $3SAT$ NP-vollständig ist. Heute werden wir für einige weitere Probleme in NP zeigen, dass sie NP-vollständig sind, indem wir $3SAT$ in polynomieller Zeit darauf reduzieren. Gegeben ist jedes mal eine 3-KNF-Formel $w = c_1 \wedge \ldots \wedge c_k$ mit Variablen $x_1, \ldots, x_l$. Daraus werden wir eine Instanz $f(w)$ des neuen NP-Problems konstruieren und zeigen, dass $f(w)$ genau dann wahr ist, wenn w erfüllbar ist. Schließlich müssen wir uns noch klar machen, dass die Konstruktion in Polynomialzeit läuft.

VERTEX-COVER (Knotenüberdeckung)

Sei $G = (V, E)$ ein ungerichteter Graph. Eine Knotenmenge $V' \subseteq V$ heißt Knotenüberdeckung, wenn jede Kante einen Knoten aus V' berührt, also wenn für alle $\{u, v\} \in E$ gilt: $u \in V'$ oder $v \in V'$. Damit definieren wir die Sprache

$$VERTEX - COVER = \{(G, t) \mid \text{G hat eine Knotenüberdeckung der Größe t}\}.$$

$VERTEX - COVER$ ist in NP, denn als Zertifikat können wir einfach eine Knotenüberdeckung der Größe t nehmen. Um zu beweisen, dass es auch NP-vollständig ist, zeigen wir $3SAT \leq_P VERTEX - COVER$, wir suchen also $f(w) = (G, t)$, so dass $w \in 3SAT \Leftrightarrow f(w) \in VERTEX - COVER$.

Konstruktion von G

- Für jede Variable x_j fügen wir zwei Knoten hinzu und verbinden sie durch eine Kante. Die beiden Knoten heißen x_j und $\overline{x_j}$.
- Für jede Klausel c_i fügen wir drei Knoten hinzu und verbinden sie zu einem Dreieck. Jeder der drei Knoten entspricht einem Literal in der Klausel.
- Schließlich wird jeder Klausel-Knoten mit dem entsprechenden Variablen-Knoten verbunden.

Wahl von t

Wir wählen $t = 2k + l$ (Ein Knoten pro Variable, zwei Knoten pro Klausel).

Die Reduktion funktioniert

Wir müssen zwei Richtungen zeigen. Zuerst nehmen wir an, dass w erfüllbar ist. Für jede Variable x_j nehmen wir einen Knoten, nämlich x_j, falls $x_j = 1$ oder $\overline{x_j}$, falls $x_j = 0$. Damit sind alle Kanten zwischen den Variablen-Knoten überdeckt. Für jede Klausel suchen wir einen Knoten aus, der einem wahren Literal entspricht und nehmen *die anderen beiden* in unsere Knotenüberdeckung. Das sind insgesamt t Knoten, die alle Kanten überdecken: Mit zwei Knoten ist jedes Klausel-Dreieck und zwei seiner Verbindungen überdeckt. Die letzte Verbindung ist auch überdeckt, denn sie geht zu einem wahren Literal, das wir schon im ersten Schritt aufgenommen haben.

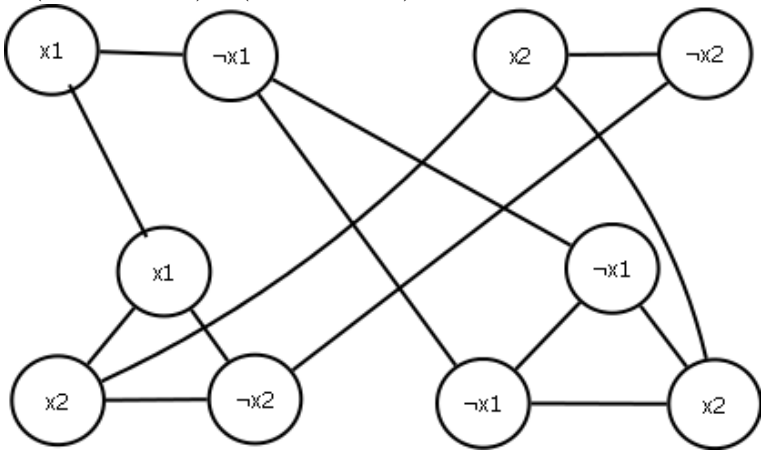
Nun nehmen wir an, dass G eine Knotenüberdeckung V' der Größe t hat. Um die Kanten zwischen den Variablen-Knoten zu überdecken, muss mindestens ein Knoten aus jedem Variablen-Knoten-Paar in V' sein. Aus jedem Klausel-Dreieck müssen mindestens zwei Knoten in V' sein. Da $|V'| = 2k + l$ haben wir also genau einen Knoten in jedem Variablen-Paar und genau zwei in jedem Klausel-Dreieck. Wir belegen jedes Literal, das wir in V'aufgenommen haben, mit 1. Die Belegung ist erfüllend, weil der fehlende Knoten in jedem Klausel-Dreieck mit einem wahren Literal verbunden sein muss, um alle Kanten zu überdecken: Jede Klausel enthält ein wahres Literal.

Die Reduktion läuft in Polynomialzeit

Für jede Variable gibt es eine Kante, für jede Klausel gibt es 6 Kanten, insgesamt müssen also $6k + l = O(n)$ Kanten geschrieben werden.

Beispiel

Für $(x_2 \vee x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_1} \vee x_2)$



HAMPATH (Hamilton-Pfad)

Wir haben schon gesehen, dass $HAMPATH = \{(G, s, t) \mid$ G ist ein gerichteter Graph mit Hamilton-Pfad von s nach t$\}$ in NP ist. Nun zeigen wir $3SAT \leq_P HAMPATH$ und damit, dass es auch NP-vollständig ist.

Konstruktion von G

Der Graph G enthält Diamantförmige Untergraphen. Ein Diamant besteht aus einem oberen Knoten, zwei mittleren Knoten und einem unteren Knoten, so dass der obere Knoten eine ausgehende Kante zu jedem mittleren Knoten hat und der untere Knoten eine eingehende Kante von jedem mittleren Knoten. Außerdem enthält G Ketten von Knoten - das soll heißen das jeder Knoten zum nächsten Kettenglied eine eingehende und eine ausgehende Kante hat.

- Für jede Variable x_j fügen wir einen Diamanten hinzu, aber so, dass der untere Knoten von x_j der obere Knoten von x_{j+1} ist.
- Zwischen den mittleren Knoten jedes Diamanten fügen wir eine Kette der Länge $3k + 1$ hinzu. Dabei sollen immer zwei Knoten für eine Klausel stehen, und zwischen den Klausel-Paaren (sowie am Anfang und am Ende der Kette) ist je ein Trennungsknoten.
- Außerdem bekommt jede Klausel einen Knoten c_i. Dieser wird mit den Ketten verbunden: Wenn in der Klausel c_i das Literal x_j vorkommt, bekommt der Knoten c_i eine eingehende Kante vom ersten Knoten des i-ten Klausel-Paares im j-ten Diamanten und eine ausgehende Kante zum zweiten Knoten dieses Paares. Wenn stattdessen $\overline{x_j}$ vorkommt, werden die Richtungen der Kanten umgedreht.

Wahl von s und t

Der obere Knoten des ersten Diamanten ist s, der untere Knoten des l-ten Diamanten ist t.

Die Reduktion funktioniert

Wir gehen zuerst davon aus, dass es eine erfüllende Belegung gibt und suchen einen Hamilton-Pfad. Wir vernachlässigen die Klausel-Knoten aus dem dritten Schritt und gehen durch die Diamanten: Wenn in unserer Belegung die Variable $x_1 = 1$ ist, gehen wir im ersten Diamanten von s nach links, dann von links nach rechts durch die Kette und schließlich vom linken Knoten nach unten (im Zick-Zack). Wenn $x_1 = 0$ ist, gehen wir genau andersherum

(links und rechts vertauscht). Nun sind wir im oberen Knoten des zweiten Diamanten und machen das gleiche, bis wir bei t angekommen sind. Die fehlenden Klausel-Knoten erreichen wir, indem wir ein wahres Literal aus jeder Klausel auswählen und an der entsprechenden Stelle in der Klausel-Kette einen Umweg machen. Das funktioniert, weil die Richtungen der Verbindungen passend zum Zick-Zack-Kurs gewählt wurden.

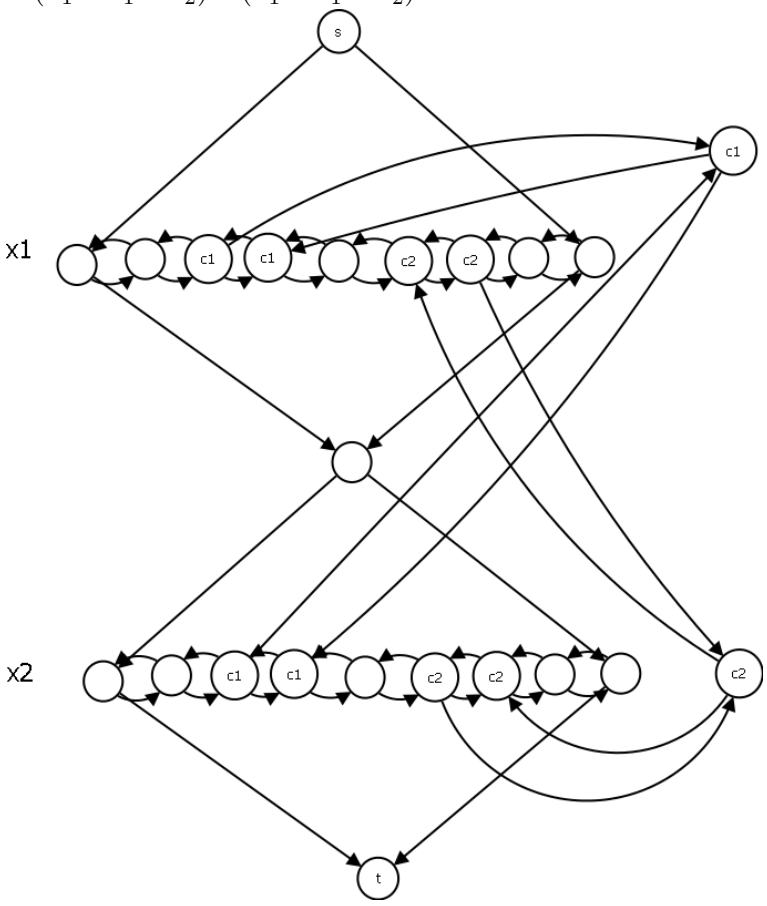
Für die andere Richtung nehmen wir an, dass es einen Hamilton-Pfad in G gibt. Wenn der Pfad die Form wie oben hat (Zick-Zack mit Umwegen), können wir eine erfüllende Belegung einfach ablesen: Die Variable x_j wird auf 1 gesetzt, wenn es im j-ten Diamanten zuerst nach links geht, und sie wird auf 0 gesetzt, wenn es zuerst nach rechts geht. Dass der Pfad genau diese Form hat, wird von den Trennungsknoten in den Ketten garantiert.

Die Reduktion läuft in Polynomialzeit

Für jede Variable brauchen wir ungefähr so viele Kanten, wie es Klauseln gibt, also k Kanten. Für jede Klausel ist die Anzahl der Kanten zwischen 2 und 6. Insgesamt müssen also $O(n^2)$ Kanten geschrieben werden.

Beispiel

Für $(x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_1} \vee x_2)$



UNHAMPATH (Hamilton-Pfad auf ungerichteten Graphen)

$UNHAMPATH$ ist in NP mit dem Hamilton-Pfad als Zertifikat. Um zu beweisen, dass es auch NP-vollständig ist, zeigen wir $HAMPATH \leq_P UNHAMPATH$. Sei $G = (V, E)$ ein gerichteter Graph. Wir konstruieren einen ungerichteten Graphen $G' = (V', E')$, indem wir für jeden Knoten $u \in V \setminus \{s, t\}$ drei Knoten in V' einfügen: u^{in}, u^{mid} und u^{out}. Für s und t kommen nur die Knoten s^{out} und t^{in} dazu. Außerdem bekommt G' für jedes u die Kanten $\left\{u^{in}, u^{mid}\right\}$ und $\left\{u^{mid}, u^{out}\right\}$. Schließlich fügen wir für jede Kante $(u, v) \in E$ die Kante $\{u^{out}, v^{in}\}$ in E' hinzu.

Wenn es einen Hamilton-Pfad $P = (s, u_1, \ldots, u_n, t)$ in G gibt, ist offensichtlich $P' = (s^{out}, u_1^{in}, u_1^{mid}, u_1^{out}, u_2^{in}, \ldots, u_n^{out}, t^{i}$ ein Hamilton-Pfad in G'. Umgekehrt muss ein Hamilton-Pfad von s^{out}nach t^{in}in G' aus solchen $(u_i^{in}, u_i^{mid}, u_i^{out})$-Tripeln bestehen, denn von s^{out} geht es nur zu einem u_i^{in}-Knoten. Der nächste Knoten muss u_i^{mid} sein, denn wenn

wir u_i^{mid} von der anderen Seite besuchen, sind wir in einer Sackgasse. Von u_i^{mid} geht es nur nach u_i^{out} und so weiter. Diesem Pfad aus Tripeln entspricht ein Hamilton-Pfad in G.

Die Reduktion läuft in Polynomialzeit.

SUBSET-SUM (Teilsummenproblem)

Wir wissen schon, dass $SUBSET-SUM = \{(S,t) \mid \exists T \subseteq S : \sum_{x \in T} x = t\}$in NP ist. Wir zeigen, dass es NP-vollständig ist, indem wir $3SAT \leq_P SUBSET-SUM$ beweisen.

Konstruktion von S

Wir beschreiben die Zahlen von S anhand ihrer Dezimaldarstellung und gehen davon aus, dass jede Zahl $l+k$ Stellen hat und auch mit Nullen anfangen kann. Meistens betrachten wir die ersten l Stellen und die restlichen k Stellen getrennt.

- Für jede Variable x_j sind zwei Zahlen y_j und z_j in S. In den ersten l Stellen steht an der j-ten Stelle eine Eins. In den verbleibenden k Stellen steht an der i-ten Stelle von y_j eine Eins, wenn in der Klausel c_i das Literal x_j vorkommt und an der i-ten Stelle von z_j steht eine Eins, wenn in der Klausel c_i das Literal $\overline{x_j}$ vorkommt. Sonst enthalten y_j und z_j nur Nullen.
- Für jede Klausel bekommt S zwei Zahlen g_i und h_i. Dabei ist $g_i = h_i$. Die ersten l Stellen enthalten nur Nullen. In den übrigen k Stellen ist die i-te Stelle eine Eins, der Rest sind Nullen.

Wahl von t

Als t nehmen wir die $l+k$-stellige Zahl, die aus l Einsen gefolgt von k Dreien besteht.

Die Reduktion funktioniert

Wir nehmen zuerst an, dass es eine erfüllende Belegung gibt und suchen Zahlen $T \subseteq S$, die sich zu t summieren. Für jede Variable x_j nehmen wir y_j in T auf, wenn in unserer Belegung $x_j = 1$ ist. Wenn $x_j = 0$ ist, nehmen wir stattdessen z_j. Damit haben wir eine $l+k$-stellige Zahl, die an der ersten l Stellen Einsen hat. An den folgenden k Stellen stehen Werte zwischen 1 und 3, denn jede Klausel enthält mindestens ein wahres Literal und höchstens drei. Wenn von den k Stellen die i-te Stelle einen kleineren Wert als 3 hat, nehmen wir g_i oder h_i oder beide dazu. So kommen wir auf t.

Jetzt nehmen wir an, dass es eine Teilmenge $T \subseteq S$ gibt mit $\sum_{x \in T} x = t$. Da die ersten l Stellen 1 sind, wissen wir, dass für jedes $j \in \{1, \ldots, l\}$ ein y_j oder ein z_j in T sein muss, aber nicht beides. Wir setzen $x_j = 1$, wenn $y_j \in T$ und $x_j = 0$, wenn $z_j \in T$. Diese Belegung ist erfüllend, denn die letzten k Stellen haben alle den Wert 3. Durch die g_i und h_i kann jede Stelle höchstens auf 2 gebracht werden, also muss mindestens eine Eins von einem x_j oder y_j kommen, das heißt jede Klausel enthält ein wahres Literal.

Die Reduktion läuft in Polynomialzeit

Die Tabelle hat ungefähr $(k+l)^2$ einfach zu berechnende Einträge, wir brauchen also $O(n^2)$ einfache (polynomielle) Schritte.

Beispiel

Für $(x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_3})$

	x_1	x_2	x_3	c_1	c_2
y_1	1	0	0	1	0
z_1	1	0	0	0	1
y_2		1	0	1	1
z_2		1	0	0	0
y_3			1	0	0
z_3			1	1	1
g_1				1	0
h_1				1	0
g_2					1
h_2					1
t	1	1	1	3	3