Sudoku ist NP-vollständig

Seminar über Algorithmen und Komplexität
Freie Universität Berlin – Institut für Informatik – SS 2007
Sarah Will – 28.07.2007

Einführung

Sudoku ist ein japanisches Logikrätsel und hat folgende drei Regeln:

- gegeben ist ein n^2 x n^2 Gitter, welches in n x n Quadrate unterteilt ist. Der Wert n wird als Ordnung bezeichnet.
- Einige Felder sind mit den Zahlen zwischen 1 und n^2 gefüllt
- Das Ziel ist es alle leeren Felder so mit den Zahlen von 1 bis n^2 zu füllen, dass jede Zeile, jede Spalte und jedes n x n Quadrat jede Zahl nur einmal enthält.

		1			6			9
					4		3	
8			7		1	6		
	8	5						
	3	9				8	1	
						2	9	
		7	6		8			1
	2		4					
5			1			3		

4	7	1	3	2	6	5	8	9
9	6	2	5	8	4	1	3	7
8	5	3	7	9	1	6	4	2
2	8	5	9	1	3	4	7	6
6	3	9	2	4	7	8	1	5
7	1	4	8	6	5	2	9	3
3	4	7	6	5	8	9	2	1
1	2	6	4	3	9	7	5	8
5	9	8	1	7	2	3	6	4

Die NP-Vollständigkeit von Sudoku wird durch eine Reihe von Reduktionen gezeigt, von bestimmten logischen Problemen über graphentheoretische Probleme bzgl. Kantenzerlegung hin zum Problem der partiellen lateinischen Quadrate. Danach erfolgt eine Reduktion von lateinischen Quadraten auf Sudoku.

Another Solution Problem (ASP)

Funktionsprobleme

Sei Π ein Tripel (D,S,σ), das folgendes erfüllt

- D ist Menge der Instanzen eines Problems
- S ist Menge, die alles enthält was Lösung sein kann
- σ ist Abbildung von D nach 2^S. Für eine Instanz $x \in D$, ist $\sigma(x)$ ($\subseteq S$) Lösungsmenge von x und ein Element aus $\sigma(x)$ heißt Lösung von x.

Das Problem eine Lösung für ein $x \in D$ (bzw. Π) zu finden heißt Funktionsproblem.

FNP

FNP ist die Klasse, welche aus Funktionsproblemen $\Pi = (D,S,\sigma)$ besteht und für die gilt:

- es existiert ein polynomielles p, so dass $|s| \leq p(|x|)$ für alle $x \in D$ und für jedes $s \in \sigma(x)$ gilt

- für alle $x \in D$ und $y \in S$ kann in polynomieller Zeit entschieden werden, ob $y \in \sigma(x)$

Entscheidungsproblem für Π
Sei $\Pi = (D,S,\sigma)$ ein Funktionsproblem. $Y = \{x \in D \mid \sigma(x) \neq \emptyset\}$. $\Pi_d = (D,Y)$ (also das Paar bestehend aus Menge aller Instanzen und Menge aller Ja-Instanzen) nennt man Entscheidungsproblem induziert durch Π oder kurz Entscheidungsproblem für Π.

Beachte: Für jedes Entscheidungsproblem $\Pi' \in NP$ existiert ein Funktionsproblem $\Pi \in FNP$, so dass $\Pi_d = \Pi'$

Polynomielle ASP-Reduktion
Seien $\Pi_1 = (D_1,S_1,\sigma_1)$ und $\Pi_2 = (D_2,S_2,\sigma_2)$ Funktionsprobleme. Eine polynomielle ASP-Reduktion von Π_1 nach Π_2 wird erfüllt von $\varphi = (\varphi_D,\varphi_S)$:
- φ_D ist eine polynomiell berechenbare Abbildung von D_1 nach D_2
- für alle $x \in D_1$ ist φ_S eine polynomielle Bijektion von $\sigma 1(x)$ nach $\sigma_2(\varphi_D(x))$.

ASP-vollständig
Ein Funktionsproblem Π ist ASP-vollständig genau dann, wenn $\Pi \in FNP$ und $\Pi' \leq_{ASP} \Pi$ für alle $\Pi' \in FNP$.

ASP-Vollständigkeit impliziert NP-Vollständigkeit.

1-in-4 SAT ist NP-Vollständig

Cook hat gezeigt, dass das Entscheiden des Erfüllbarkeitsproblems von Formeln in KNF mit genau drei Literalen (3KNF SAT) NP-vollständig ist.
Schaefer hat gezeigt, dass die Entscheidung, ob eine 3KNF Formel eine erfüllende Belegung besitzt bei der jede Klausel genau ein wahres Literal besitzt (1-in-3 SAT) NP-vollständig ist. Dies gilt auch, wenn jede Klausel nur ein Literal und keine Negation besitzt.
Überführung von 1-in-3 SAT in „Eindeutiges 1-in-4 SAT“. Ein Beispiel dieses Problems ist eine Formel in KNF mit genau vier Literalen pro Klausel und einer 1-in-4 SAT erfüllenden Belegung (d.h. eine wahre Belegung, die genau ein Literal pro Klausel auf wahr setzt).

Theorem 1
Das eindeutige 1-in-4 SAT ist NP-vollständig.

Beweis
Zugehörigkeit zu NP ist klar. Vollständigkeit wird erreicht durch eine Reduktion auf 1-in-3 SAT. Gegeben sei eine Instanz von 1-in-3 SAT, eine Menge S von Klauseln, in der keine Klausel ein negiertes Literal enthält, konstruiere eine Menge von Klauseln T durch eine neue Variable N und füge N in jede Klausel von S ein. Jede Klausel in T besteht aus vier unterschiedlichen Literalen. Drei zusätzliche Klauseln werden zu T hinzugefügt mit vier Variablen {A,B,C,D}; diese drei sind disjunkt: {A,B,C,N'}, {A,C,D,N'} und {A,B,D,N'} wobei N' die Negation von N ist. Die Konjunktion der Klauselmenge in T ist 1-in-4 erfüllend durch Setzen von N gleich wahr, alle ursprünglichen Variablen gleich falsch, A wahr und B, C und D falsch. Man kann

leicht sehen, dass keine andere 1-in-4 erfüllende Belegung N gleich wahr setzt. Ist N falsch, dann gibt es eine 1-in-4 erfüllende Belegung genau dann, wenn die Klauselmenge S 1-in-3 erfüllend ist und A, B, C und D falsch sind.

Kantenzerlegung von Graphen in Dreiecke ist NP-vollständig

Das Problem heißt „eindeutige Zerlegung in Dreiecke“, ein Beispiel ist ein Graph G und eine Kantenzerlegung von G in Dreiecke. Das Problem ist nun zu entscheiden, ob es mehr als eine solche Zerlegung gibt.

Theorem 2
Eine eindeutige Zerlegung in Dreiecke ist NP-vollständig für tripartite Graphen.

Beweisskizze
Zugehörigkeit zu NP ist klar. Um Vollständigkeit zu zeigen wird „Eindeutiges 1-in-4 SAT“ reduziert auf dieses Problem. Für den Beweis brauchen wir einen Graphen H(3,p), der zwei unterschiedliche Zerlegungen in Dreiecke besitzt. H(3,p) hat die Knotenmenge $\{(x,y,z) \mid x+y+z \equiv 0 \pmod p\}$. Zwei Konten sind adjazent, wenn sie in einer Position übereinstimmen und sich in den anderen zwei um eins unterscheiden. H(3,p) ist dreifärbbar (tripartite) genau dann, wenn $p = 0 \pmod 3$; es wird fortan angenommen, dass $p = 0 \pmod 3$. H(3,p) besitzt zwei unterschiedliche Kantenzerlegungen in Dreiecke; bezeichnet im Folgenden als T-Zerlegung und F-Zerlegung. T- und F-Patches sind wie bei Holyer[3] definiert.
Durch die Reduktion vom eindeutigen 1-in-4 SAT wird jede Variable durch eine Kopie von H(3,p) dargestellt, die korrekt mit drei Farben gefärbt wurde. Tauchen vier Variablen zusammen in einer Klausel auf, so werden diese identifiziert durch die vier H(3,p)’s. Ist die Variable nicht negiert, so wird ein F-Patch verwendet, ist sie negiert dann ein T-Patch. Nach Identifikation der vier Patches, wird das zentrale Dreieck entfernt. Diese Identifikation wird für alle Klauseln ausgeführt, so dass Konten mit derselben Farbe immer identifiziert werden; dies garantiert, dass der so konstruierte Graph tripartite bleibt. Jede Identifikation von vier Patches verlangt, dass eine Variable in der Identifikation einen T-Zerlegung erhält („wahr“), wenn sie nicht negiert ist oder eine F-Zerlegung, wenn sie negiert ist. Die verbleiben drei erhalten alle eine F-Zerlegung, wenn sie nicht negiert sind und eine T-Zerlegung, wenn sie negiert sind. Die so erhaltene Kantenzerlegung entspricht genau einer 1-in-4 Belegung.

ASP für Lateinische Quadrate ist NP-vollständig

Ein lateinisches Quadrat der Ordnung n ist eine n x n Matrix, in der in jeder Spalte und jeder Zeile die Zahlen von 1 bis n genau einmal auftauchen. (siehe Beispiel 1)

Beispiel 1:

1	**3**	**2**
3	**2**	**1**
2	**1**	**3**

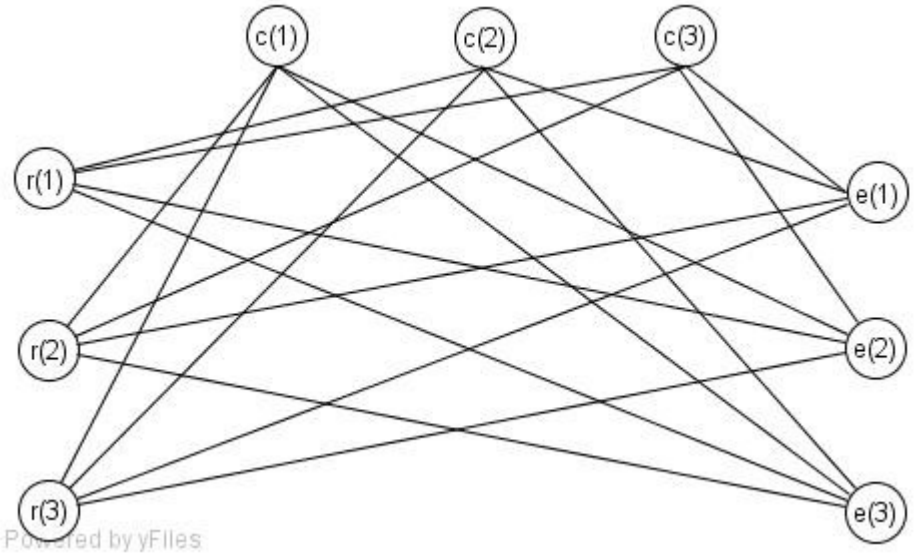
Beispiel 2:

1		
	2	
		3

Ein partielles lateinisches Quadrat ist ein lateinisches Quadrat, welches teilweise leere Felder aufweist (siehe Beispiel 2).

Gegeben sei ein partielles lateinisches Quadrat der Ordnung n, der zugehörige Lückengraph ist ein Graph mit der Knotenmenge {r(1),...,r(n),c(1),...,c(n), e(1),...,e(n)}, also 3n Knoten, wobei r(i) die i-te Zeile, c(i) die i-te Spalte und e(i) das Element i markiert. Die Kante {r(i),c(j)} ist enthalten, wenn der Eintrag (i,j) leer ist. Die Kante {r(i),e(k)} ist enthalten, wenn Reihe i das Element k nicht enthält. Die Kante {c(j),e(k)} ist enthalten, wenn Spalte j das Element k nicht enthält. Dieser Inzidenzgraph hat ein Kantenzerlegung in Dreiecke genau dann, wenn das partielle lateinische Quadrat eine Lösung besitzt. Und zwar, die Anzahl der Lösungen für das partielle lateinische Quadrat und die der Kantenzerlegungen sind gleichgroß. Das folgende Theorem zeigt, dass alle gleichmäßigen tripartiten Graphen aus den Lückengraphen von partiellen lateinischen Quadraten entstehen.

1		
	2	
		3



Theorem 3
Gegeben sei ein gleichmäßiger tripartiter Graph G mit 3n Knoten, dann gibt es ein partielles lateinisches Quadrat der Ordnung 2n, dessen Lückengraph aus G zusammen mit 3n vereinzelten Knoten besteht. Außerdem kann ein solches partielles lateinisches Quadrat in Polynomialzeit von n konstruiert werden.

Durch diese Konstruktion erhalten wir

Theorem 4
Sei ein partielles lateinisches Quadrat P gegeben und eine Lösung L, dann ist die Entscheidung, ob P eine weitere Lösung besitzt NP-vollständig.

Beweis
Das Problem der eindeutigen Kantenzerlegung in Dreiecke für tripartite Graphen, dessen NP-Vollständigkeit aus Theorem 2 folgt, wird reduziert. Wir gehen daher von einen gleichmäßigen tripartiten Graph G aus und einer Kantenzerlegung von G in Dreiecke. Unter Verwendung von Theorem 3 wird ein partielles lateinisches Quadrat P konstruiert, dessen Lückengraph G entspricht; die Kantenzerlegung von G liefert eine Lösung für P. Die Existenz einer zweiten Lösung für P ist gleichbedeutend mit einer zweiten Kantenzerlegung für G und somit ist die NP-Vollständigkeit gezeigt.

Sudoku ist NP-vollständig

Lemma 1
Für $i, j \in \{0,...,n^2 - 1\}$ gilt

$$S(i,j) = \begin{cases} \perp & (i,j) \in B \\ ((i \bmod n)n + \lfloor i/n \rfloor + j) \bmod n^2 & sonst \end{cases}$$

wobei $B = \{(i,j) \mid \lfloor i/n \rfloor = 0 \text{ und } (j \bmod n) = 0\}$
Sudoku S' erhält man durch Ausfüllen von S und S' ist eine Lösung, wenn

- $\forall (i,j) \in B, S'(i,j) \bmod n = 0$
- Quadrat L mit L(i,j/n) = S(i,j)/n für alle (i,j) $\in$ B ist lateinisches Quadrat

Das Quadrat S_0 definiert durch $S_0 = ((i \bmod n)n + \lfloor i/n \rfloor + j) \bmod n^2$ bildet eine Lösung für Sudoku, d.h. jede Zeile, Spalte und jedes kleine Quadrat enthalten die Zahlen von 0 bis n^2-1 genau einmal.

Theorem 5
Eine Lösung für ein gegebenes Sudokuproblem zu finden ist ASP-vollständig.

Beweis
Zugehörigkeit zu FNP ist klar.
Polynomialzeitliche ASP-Reduktion vom Problem lateinischer Quadrate auf Sudoku.
Für ein gegebenes partielles lateinisches Quadrat L der Ordnung n, konstruiere Sudoku S wie folgt:

$$S(i,j) = \begin{cases} L(i, j/n)n & (i,j) \in B, L(i, j/n) \neq \perp \\ \perp & (i,j) \in B, L(i, j/n) = \perp \\ ((i \bmod n)n + \lfloor i/n \rfloor + j) \bmod n^2 & sonst \end{cases}$$

Aus Lemma 1 folgt, jede Lösung für L hat eine zugehörige Lösung für S und diese ist in Polynomialzeit berechenbar. Somit ist die ASP-Reduktion in Polynomialzeit erfüllt.

0		
	1	
		2

➔

0	1	2		4	5		7	8
	4	5	3	7	8		1	2
	7	8		1	2	6	4	5
1	2	3	4	5	6	7	8	0
4	5	6	7	8	0	1	2	3
7	8	0	1	2	3	4	5	6
2	3	4	5	6	7	8	0	1
5	6	7	8	0	1	2	3	4
8	0	1	2	3	4	5	6	7

Literatur

[1] Takayuki Yato. Complexity And Completeness Of Finding Another Solution And Its Application To Puzzles, Master Thesis, Graduate School of Science The University of Tokyo, January 2003
http://www-imai.is.s.u-tokyo.ac.jp/%7Eyato/data2/MasterThesis.pdf

[2] C. J. Colbourn, M. J. Colbourn, and D. R. Stinson. The computational complexity of recognizing critical sets. In Graph theory, Singapore 1983, Lecture Notes in Mathematics 1073, Seite 248–253.

[3] Ian Holyer. The NP-Completeness of Some Edge-Partition Problems. In SIAM J. COMPUT, Vol. 10, No. 4, November 1981, Seite 713-717.
http://www.cs.bris.ac.uk/~ian/graphs/part.pdf

[4] Volker Kaibel, Thorsten Koch. Mathematik für den Volkssport. In Mitteilungen der DMV, 14:2, 2006, Seite 93-96.
http://www.math.uni-magdeburg.de/~kaibel/Downloads/mdmv-sudoku.pdf

[5] Thomas J. Schaefer. The complexity of satisfiability problems. In Proceedings of the 10th Annual ACM Symposium on Theory of Computing, 1978, Seite 216–226.