Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

VOLKER KAIBEL THORSTEN KOCH

Mathematik für den Volkssport

ZIB-Report 06-28 (May 2006)

Mathematik für den Volkssport

Volker Kaibel Thorsten Koch

6. Juni 2006

„Volkssport Sudoku“ titelt der Stern in seiner Ausgabe vom 24. Mai 2006. In der Tat traut sich derzeit kaum noch eine Zeitung, ohne Sudoku zu erscheinen. Die Begeisterung am Lösen dieser Zahlenrätsel offenbart eine unvermutete Freude am algorithmischen Arbeiten. Mathematisch kann man Sudokus als lineare diophantische Gleichungssysteme mit Nichtnegativitätsbedingungen formulieren. Solche ganzzahligen linearen Programme sind die wichtigsten Modellierungswerkzeuge in zahlreichen Anwendungsgebieten wie z.B. der Optimierung von Telekommunikations- und Verkehrsnetzen. Moderne Verfahren zur Lösung dieser Optimierungsprobleme sind durch Sudokus allerdings deutlich weniger zu beeindrucken als Zeitungsleser.

Im Untertitel fährt der Stern fort: „Warum Millionen Deutsche dem Zahlenrätsel verfallen sind“. Kommt hier ein großes, bisher verborgenes Potential von Mathematik-Begeisterung zum Vorschein, oder ist es doch eher so, wie der *Independent* seine Sudokus bewirbt: „There is no math involved.“?
Beim Sudoku ist die Aufgabe, ein Feld wie in Abb. 1 so zu vervollständigen, dass in jeder Zeile, in jeder Spalte, und in jedem der angedeuteten 3×3-Quadrate jede der Zahlen $1, 2, \ldots, 9$ genau einmal vorkommt. Der Rätselsteller garantiert dabei, dass die Vervollständigung eindeutig ist.

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

Abbildung 1: Dieses Sudoku-Rätsel erfüllt zwar nicht die ästhetischen Ansprüche von Sudoku-Meistern (weil die Einträge nicht symmetrisch verteilt sind), kommt dafür aber mit der derzeit kleinsten bekannten Anzahl von 17 vorgegebenen Zahlen aus.

Natürlich ist hier Mathematik involviert, z.B., wenn man fragt, wie viele ausgefüllte Sudokus es denn gibt (es sind 6.670.903.752.021.072.936.960, also un-

gefähr $6,671 \cdot 10^{21}$, siehe [6], allerdings gibt es nur 5.472.730.538 Äquivalenzklassen, siehe [9]), oder bei der bisher ungelösten Frage nach der (viel größeren) Zahl der verschiedenen Sudokurätsel.
Ob das manuelle Lösen eines Sudokurätsels nun als Mathematik gewertet werden darf oder muss, soll nicht unser Thema sein. Wir möchten hier eine Modellierung diskutieren, mit der ein Sudokurätsel mit mathematischen Methoden und Software gelöst werden kann, wie sie z.B. im Berliner DFG-Forschungszentrum MATHEON („Mathematik für Schlüsseltechnologien“) zur Optimierung in Anwendungsbereichen wie der Telekommunikation, der Logistik oder dem Nahverkehr entwickelt und eingesetzt werden. Obwohl wir einen WWW-Service zur Lösung von Sudokus mit Hilfe der im folgenden vorgestellten Methoden eingerichtet haben, ist das Ziel dabei weniger, die dem Zahlenrätsel Verfallenen zu retten. Der Beitrag ist eher als eine Einladung in ein Gebiet gedacht, das sowohl mathematisch faszinierend als auch für Anwendungen in Schlüsseltechnologien sehr fruchtbar ist: die *ganzzahlige lineare Programmierung*.

1 Ein mathematisches Modell

Sudokus kann man leicht mit Hilfe von 81 Variablen $x_{ij} \in \{1, 2, \ldots, 9\} =: [9]$ $(i, j \in [9])$ modellieren, wobei $x_{ij} = k$ bedeutet, dass im Feld (i, j) die Zahl k steht. Die Sudokuregeln übersetzen sich dann in Ungleichheitsbedingungen der Art $x_{2,3} \neq x_{2,7}$. Allerdings sind solche Ungleichheitsbedingungen in der Regel algorithmisch schwierig zu behandeln. Wir verwenden deswegen eine etwas andere Modellierung.

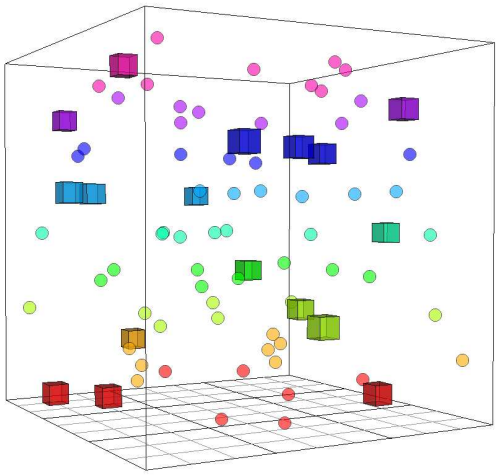


Abbildung 2: Lösung des Sudokus aus Abb. 1 mit 17 vorgegebenen Einträgen (Würfel), wobei sowohl die Höhe als auch die Farbe jeder Kugel/Würfel den Eintrag im darunter liegenden Feld repräsentiert.

Dazu führen wir 729 Variablen $x_{ijk} \in \{0, 1\}$ $(i, j, k \in [9])$ ein, wobei $x_{ijk} = 1$ sein soll, falls im Feld (i, j) die Zahl k steht. Die Variablen, die den Wert Eins ha-

ben, entsprechen den Positionen, an denen in Abbildung 2 Kugeln oder Würfel schweben. Die Sudokuregeln lassen sich jetzt als Gleichungen formulieren. Für jede Zahl $k \in [9]$ muss nun für jede Zeile $i \in [9]$

$$\sum_{j=1}^{9} x_{i,j,k} = 1 \tag{1}$$

(Abb. 3 (1)), für jede Spalte $j \in [9]$

$$\sum_{i=1}^{9} x_{i,j,k} = 1 \tag{2}$$

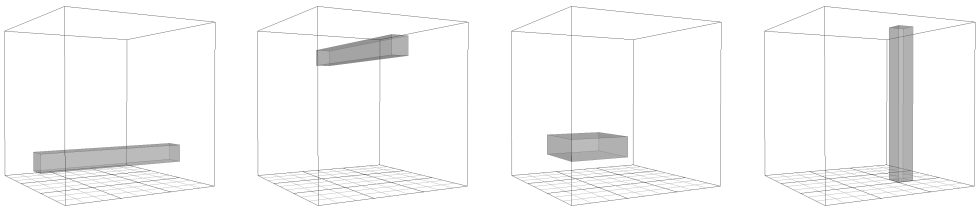
(Abb. 3 (2)), und für jedes 3×3-Quadrat $Q(a,b) = \{(i,j) : 3(a-1) < i \leq 3a, 3(b-1) < j \leq 3b\}$ $(a,b \in \{1,2,3\})$

$$\sum_{(i,j) \in Q(a,b)} x_{i,j,k} = 1 \tag{3}$$

(Abb. 3 (3)) gelten. Außerdem fordern wir für jedes Feld $(i,j) \in [9] \times [9]$

$$\sum_{k=1}^{9} x_{i,j,k} = 1 \tag{4}$$

(Abb. 3 (4)). Insgesamt erhalten wir $4 \cdot 81 = 324$ Gleichungen.



(1) (2) (3) (4)

Abbildung 3: Gleichungen im Sudoku-Modell. In jedem der grauen Bereiche muss genau eine Kugel (oder ein Würfel) schweben.

Ist nun $S \subseteq [9] \times [9]$ die Menge der Felder, deren Einträge vorgegeben sind, und ist $k(i,j) \in [9]$ (für $(i,j) \in S$) die im Feld (i,j) vorgegebene Zahl, so ist die Lösung des gegebenen Sudokurätsels die (eindeutige) nicht-negative ganzzahlige Lösung des um die Gleichungen $x_{i,j,k(i,j)} = 1$ (für alle $(i,j) \in S$) erweiterten Systems (1), (2), (3), (4).
Schaut man von oben auf den Würfel in Abbildung 2, so sieht man, dass das Sudokuproblem auch als ein Graphenfärbungsproblem aufgefasst werden kann (Abb. 4). Ist G der Graph mit Knotenmenge $[9] \times [9]$, in dem zwei Knoten $v, w \in [9] \times [9]$ genau dann mit einer Kante verbunden sind, wenn die Felder v und w in der gleichen Zeile oder Spalte oder im gleichen 3×3-Quadrat liegen,

so ist die Aufgabe, eine Färbung der Knoten mit neun Farben zu finden (wobei einige Knoten schon vorgefärbt sind), bei der keine adjazenten Knoten gleich gefärbt werden. Abbildung 5 zeigt den Graphen des 4×4-Sudokus, da der Graph des üblichen 9×9-Sudokus etwas unübersichtlich ist. Das kann man mit den eingeführten 0/1-Variablen mit Hilfe der Ungleichungen

$$x_{ijk} + x_{i'j'k} \leq 1$$

(für alle adjazenten Paare (i, j) und (i', j') und für alle $k \in [9]$) modellieren, die dann die Gleichungen (1), (2) und (3) ersetzen. Das resultierende Modell ist zusammengesetzt aus neun Modellen für das stabile Mengen Problem, die über die Gleichungen (4) gekoppelt sind.

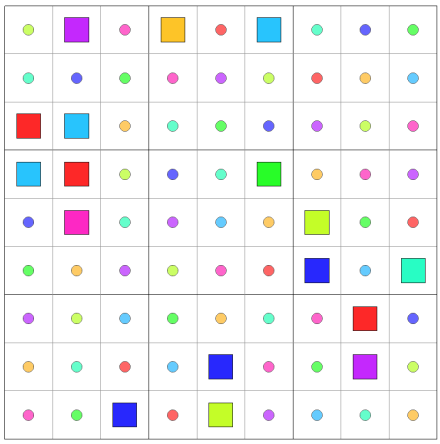


Abbildung 4: Sudoku als Färbungsproblem.

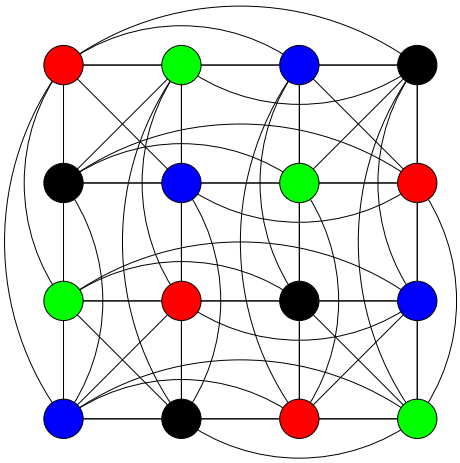


Abbildung 5: Gefärbter Graph zum 4×4-Sudoku.

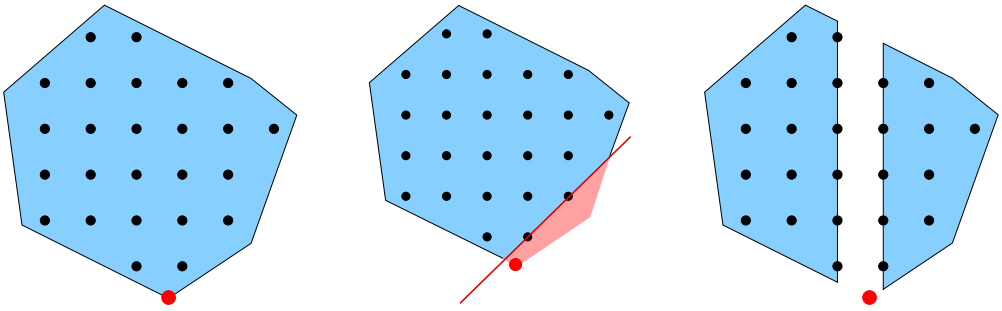
2 Ganzzahlige Lineare Programmierung

Das Sudokuproblem kann also als die Aufgabe modelliert werden, eine ganzzahlige, nicht negative Lösung eines linearen Gleichungssystems (oder Ungleichungssystems) zu bestimmen. Dieses algorithmische Problem, meistens in der Variante, dass unter allen Lösungen eine gefunden werden soll, die eine gegebene lineare Zielfunktion optimiert, ist als *ganzzahlige lineare Programmierung* (*integer linear programming*) bekannt.
Im Gegensatz zu den Varianten, bei denen man entweder auf die Nichtnegativitätsbedingungen verzichtet (*lineare diophantische Gleichungssysteme*) oder auf die Ganzzahligkeit (*lineare Programmierung*), zählt die ganzzahlige lineare Programmierung zu den im komplexitätstheoretischen Sinn schweren Problemen ($\mathcal{NP}$-schwer). Die Theorie der ganzzahligen linearen Programmierung hat sich in den vergangenen Jahrzehnten reich entfaltet. Sie weist Verbindungen zu Gebieten wie Minkowskis Geometrie der Zahlen (z.B. kurze Gittervektoren) oder der algorithmischen kommutativen Algebra (z.B. Gröbner-Basen) auf, vor allem aber ist sie stark mit der Konvexgeometrie (konvexe Polyeder) verbunden. Aus praktischer Sicht ist die ganzzahlige lineare Programmierung ein sehr mächtiges Modellierungs- und Lösungswerkzeug für Optimierungsprobleme in zahlreichen Anwendungsbereichen. Erweiterungen des erwähnten Modells für die Graphenfärbung werden etwa verwendet, um Frequenzzuweisungen in Mobilfunknetzen zu bestimmen. Die Ganzzahligkeitsbedingungen an die Variablen ermöglichen die Abbildung von vielfältigen Entscheidungsmöglichkeiten, unter anderem in der Planung von Telekommunikations- oder Verkehrsnetzen, in der Logistik oder auch im Bereich Chip-Design und -Verifikation.

3 Algorithmen und Software

Relaxiert man die Ganzzahligkeitsbedingungen eines ganzzahligen linearen Programms, so erhält man ein *kontinuierliches* lineares Programm. Um dieses zu lösen gibt es mehrere Verfahren: Das 1947 von Dantzig entwickelte, in der Praxis sehr erfolgreiche Simplexverfahren [5], die für die Theorie sehr wichtige, im Jahr 1979 von Khachiyan veröffentlichte Ellipsoid-Methode, und schließlich, beginnend mit der Arbeit von Karmarkar [7] im Jahr 1984, Innere-Punkte-Verfahren, die sowohl theoretisch effizient als auch in vielen Fällen praktisch erfolgreich sind.
Die Lösung der kontinuierlichen Relaxierung des Sudokumodells ist nicht notwendigerweise ganzzahlig. Wird aber zusätzlich eine Variable testweise auf Null oder Eins fixiert, kann es passieren, dass die kontinuierliche Relaxierung keine zulässige Lösung mehr hat, mit der Implikation, dass die Variable den anderen Wert annehmen muss. Wir haben dies bei 15.000 verschiedenen Sudokus nacheinander für beide möglichen Werte aller Variablen durchgeführt. Trägt man die so ermittelten fixen Werte in das Problem ein, war danach in allen Fällen die Lösung der kontinuierliche Relaxierung ganzzahlig und damit das Sudoku gelöst.
Derzeit basieren alle leistungsfähigen Löser für allgemeine ganzzahlige linea-

re Programme auf dem sogenannten *Branch-and-Cut* Verfahren (Abb 6). Dies ist ein erweitertes *Branch-and-Bound* Verfahren, bei dem untere Schranken durch Berechnung der kontinuierlichen Relaxierung des ganzzahligen linearen Programms bestimmt werden, Abb. 6 (1). Die Relaxierung wird dann durch Hinzufügen von Schnittebenen weiter verstärkt, Abb. 6 (2). Die strukturelle Untersuchung von Schnittebenen und die Entwicklung von *Separationsalgorithmen*, die zu einer gegebenen nicht ganzzahligen Lösung $x^\star$ der Relaxierung eine Schnittebene finden, welche $x^\star$ von der konvexen Hülle P der zulässigen ganzzahligen Punkte trennt, ist ein weites und aktives Forschungsgebiet [11]. Besonders interessiert ist man hier an Schnittebenen, die *Facetten* (also höchstdimensionale Seiten) von P erzeugen. Ein wesentlicher Grund für die enorme Leistungssteigerung der Verfahren zur Lösung von ganzzahligen linearen Programmen in den letzten Jahren sind Fortschritte bei den Separationsalgorithmen für eine Vielzahl von Schnittebenenklassen.



(1) Die Lösung der kontinuierlichen Relaxierung ist nicht ganzzahlig.

(2) Die unzulässige Lösung wird durch eine Schnittebene separiert.

(3) Zerlegen des Problems in Teilprobleme entfernt die unzulässige Lösung.

Abbildung 6: Bei Branch-and-Cut Verfahren wird erst die kontinuierliche Relaxierung des ganzzahligen Problems gelöst (1). Ist diese nicht ganzzahlig, so gibt es zwei Möglichkeiten fortzufahren: Hinzufügen von Schnittebenen (2) oder Zerlegen (3).

Ist keine weitere Verbesserung der Schranke mehr möglich, wird das Problem in zwei kleinere Teilprobleme zerlegt, typischerweise durch Aufteilung des Zulässigkeitsbereichs anhand einer Variablen mit gebrochenem Wert in der Relaxierungslösung, Abb. 6 (3). Ein wesentlicher Aspekt für die praktische Effizienz des Verfahrens ist die Auswahl dieser Variablen [2]. Der oben beschriebene Vorgang des testweise Fixierens einer Variablen ist Teil der unter dem Namen *Strong Branching* [4] bekannten Variablenauswahlregel. Dies ist die zur Zeit leistungsfähigste bekannte Auswahlregel. Da sie allerdings rechnerisch aufwendig ist, wird sie in der Regel nur approximativ eingesetzt.
Vor Beginn des eigentlichen *Branch-and-Cut* Verfahrens wird üblicherweise ein *Presolving* [3] durchgeführt. Dabei wird versucht, anhand von Implikationen, die aus dem ganzzahligen Programm abgeleitet werden, dieses zu vereinfachen. In verstärkter Form wird dieses Verfahren als *Probing* [10] bezeichnet. Nach unseren bisherigen Erfahrungen ist dieses ebenfalls ausreichend, um Sudokus

so weit zu vereinfachen, dass die kontinuierliche Relaxierung ganzzahlig wird, so dass auf Strong Branching verzichtet werden kann.
Es ist uns bisher kein Sudoku bekannt, bei dem das Problem tatsächlich in Teilprobleme zerlegt werden musste. Dabei war auch die Verwendung von Schnittebenen nicht notwendig. Offenbar reicht es dem Algorithmus aus, die Implikationen, die sich aus dem Setzen jeweils eines Wertes ergeben, zu verfolgen um ein Sudoku lösen zu können. Die Frage ob es ein Sudoku gibt bei dem dies nicht der Fall ist, bleibt dabei offen.
Löser für ganzzahlige lineare Programme sind heute komplexe Programme mit einer Vielzahl von Komponenten. Die wichtigste ist sicher der Löser für lineare Programme, aber dazu kommen Presolve- und Probing-Verfahren, eine große Zahl von Heuristiken die versuchen, zulässige ganzzahlige Lösungen aus den gebrochenen Relaxierungen zu berechnen, eine Vielzahl von Separierungsalgorithmen, sowie komplexe Variablen- und Knotenauswahlregeln.
Wer selbst ein wenig probieren möchte kann die von uns verwendete, am Zuse-Institut und im MATHEON entwickelte Software aus dem Internet laden. Die Verwendung für die Forschung ist frei. Als Löser für die ganzzahligen Programme haben wir SCIP [1] verwendet. Modelliert wurde das Problem in der Modellierungssprache Zimpl [8]. Die Sudokusammlung, das Modell als Zimpl-Datei und einiges mehr kann man unter `http://www.zib.de/koch/sudoku` finden. Auf `http://www.matheon.de` haben wir einen WWW-Service eingerichtet, bei dem man ein Sudokurätsel eingeben und es (natürlich mit Hilfe ganzzahliger Programmierung) lösen lassen kann.

4 Fazit

Moderne Optimierungssoftware löst auch Sudokurätsel im Vorübergehen. Dabei dauert es weniger als 15 Minuten vom Verstehen des Rätsels bis ein verwendbares Modell zur Verfügung steht, das nicht nur 9×9-, sondern auch 16×16-Sodukus usw. abbildet, und selbst vor „Killer-Sudokus“ (eine schwierigere Variante) nicht zurück schreckt. Und das Lösen selbst dauert mit moderner mathematischer Software nur noch Sekundenbruchteile. Das soll den Spaß der Rätselfreunde allerdings nicht bremsen. Denn immerhin wissen wir aus dem Stern-Artikel:

„Sudoku schützt vor Alzheimer und macht nicht dick.“

Literatur

[1] Tobias Achterberg, *SCIP - a framework to integrate constraint and mixed integer programming*, Tech. Report 04-19, Zuse-Institut Berlin, 2004, `http://scip.zib.de`.

[2] Tobias Achterberg, Thorsten Koch und Alexander Martin, *Branching rules revisited*, Operations Research Letters **33** (2005), 42–54.

[3] Erling D. Andersen und Knud D. Andersen, *Presolve in linear programming*, Mathematical Programming **71** (1995), 221–245.

[4] D. Applegate, R. E. Bixby, V. Chvátal und W. Cook, *Finding cuts in the TSP*, Tech. Report 95-05, DIMACS, März 1995.

[5] George B. Dantzig, *Linear programming and extensions*, Princeton University Press, 1963.

[6] Bertram Felgenhauer und Frazer Jarvis, *Mathematics of Sudoku I*, Manuskript (siehe `http://www.afjarvis.staff.shef.ac.uk/sudoku)`, Januar 2006.

[7] N. Karmarkar, *A new polynomial-time algorithm for linear programming*, Combinatorica **4** (1984), 373–395.

[8] Thorsten Koch, *Rapid mathematical programming*, Dissertation, Technische Universität Berlin, 2004, `http://www.zib.de/koch/zimpl`.

[9] Ed Russell und Frazer Jarvis, *Mathematics of Sudoku II*, Manuskript (siehe `http://www.afjarvis.staff.shef.ac.uk/sudoku)`.

[10] M.W.P. Savelsbergh, *Preprocessing and probing for mixed integer programming problems*, ORSA Journal on Computing (1994), 445–454.

[11] Laurence A. Wolsey, *Integer programming*, John Wiley & Sohns, 1998.