Diskrete Optimierungsverfahren zur Lösung von Sudokus

Seminarvortrag von Daniel Scholz am 6. Dezember 2006

Am Beispiel der Lösung von Sudokurätseln mit Hilfe der linearen Optimierung werden verschiedenen Lösungsstrategien von ganzzahligen Programmen besprochen. Dazu muss zunächst ein geeignetes Modell zum Lösen von Sudokus mittels ganzzahliger linearer Optimierung gefunden werden.

Durch eine ausführliche *Mathematica*-Implementierung der vorgestellten Lösungsideen sollen die Verfahren verdeutlicht werden.

1 Einleitung

Bei einem ***Sudoku*** besteht die Aufgabe darin, 9×9 Felder mit einigen vorgegebenen Zahlen so zu vervollständigen, dass in jeder Zeile, in jeder Spalte und in jedem der neun 3×3 Quadrate jede der Zahlen $1, 2, \ldots, 9$ genau einmal vorkommt. Dabei haben die Sudokus in der Regel eine eindeutige Lösung.

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

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

Abbildung 1: Beispiel von zwei unterschiedlich schweren Sudokus.

Unser Ziel wird es sein zunächst Sudokus als mathematisches Modell zu betrachten und dieses Modell mit Hilfe der linearen Programmierung zu lösen.

Zunächst stellt sich aber die Frage, wie viele Sudokus es überhaupt geben kann. Es gibt nach [1] genau

$$6.670.903.752.021.072.936.960 \approx 6,67 \cdot 10^{21}$$

Möglichkeiten, um die Zahlen $1, 2, \ldots, 9$ nach den gegebenen Regeln auf ein 9×9 Feld zu verteilen. Betrachten wir Felder, bei denen die Zahlen nur

permutiert sind oder bei denen das Feld nur gedreht oder gespiegelt wurde als äquivalent, so erhalten wir immer noch

$$5.472.730.538 \approx 5,47 \cdot 10^9$$

Äquivalenzklassen, siehe [3]. Dennoch gibt es sehr viel mehr Sudokus, da hier ja zu einem Feld unterschiedliche Ziffern vorgegeben sein können und wir somit zu einem ausgefüllten Feld sehr viele Sudokus formulieren können.

2 Mathematisches Modell

Um ein Sudoku mit mathematischen Methoden lösen zu können, müssen wir zunächst ein Modell aufstellen.

Einfachstest Modell

Wir betrachten 81 Variablen

$$x_{i,j} \in \{1, 2, \ldots, 9\} \qquad \text{für} \quad 1 \leq i, j \leq 9$$

und es bedeute

$$x_{i,j} = k,$$

wenn im Feld (i, j) die Zahl $k \in \{1, 2, \ldots, 9\}$ steht. Die Sudokuregeln erhalten wir nun, indem wir Ungleichungen der Form

$$x_{1,i} \neq x_{1,j} \qquad \text{für} \quad i \neq j$$

erzeugen. Dies bedeutet gerade, dass in der ersten Reihe keine der 9 Variablen den gleichen Wert annehmen dürfen, dass also keine Zahl in der Zeile mehrmals vorkommt.

Dieses sehr einfache und anschauliche Modell ist allerdings ungeeignet, um es algorithmisch zu lösen. Daher betrachten wir das folgende verbesserte Modell:

Verbessertes Modell

Um von Ungleichungen auf Gleichungen zu kommen, müssen wir weitere Variablen einführen.

Wir betrachten die $9^3 = 729$ Variablen

$$x_{i,j,k} \in \{0, 1\} \qquad \text{für} \quad 1 \leq i, j, k \leq 9$$

und es bedeute

$$x_{i,j,k} = 1,$$

wenn im Feld (i, j) die Zahl k steht.

Die Variablen, die den Wert 1 haben, entsprechen in Abbildung 2 einer Kugel. Hier hat die Variable $x_{4,2,4}$ den Wert 1. Somit hat das gegebene Sudoku im Feld $(4, 2)$ den Eintrag 4.

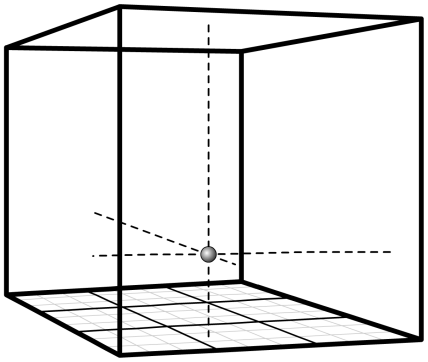

Abbildung 2: Die Variable $x_{4,2,4}$ hat den Wert 1.

Die *Höhe* der jeweiligen Kugel repräsentiert also den Eintrag im darunterliegenden Feld. Damit können wir die Sudokuregeln als Gleichungen formulieren.

Gleichungskette 1

Natürlich darf in jedem Feld (i, j) nur eine der 9 Variablen den Wert 1 annehmen. Wir erhalten die 81 Gleichungen

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

siehe Abbildung 3a.

Gleichungskette 2

Damit in jeder Spalte jede Zahl genau einmal vorkommt, erhalten wir die 81 Gleichungen

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

siehe Abbildung 3b.

Gleichungskette 3

Für jede Zeile ergeben sich analog die 81 Gleichungen

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

siehe Abbildung 3c.

Gleichungskette 4

Nun müssen wir noch fordern, dass in jedem der 3×3 Quadrate jede Zahl genau einmal vorkommt. Dazu bezeichne

$$Q(a,b) \;=\; \{(i,j) \mid 3(a-1) < i \leq 3a,\, 3(b-1) < j \leq 3b\} \qquad \text{für} \quad 1 \leq a, b \leq 3$$

die 9 Quadrate. Damit erhalten wir die 81 Gleichungen

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

siehe Abbildung 3d.

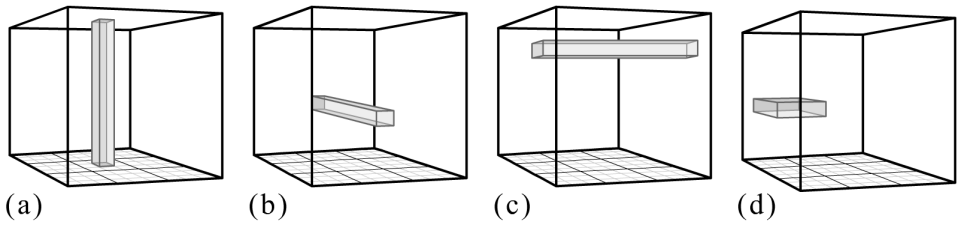

Abbildung 3: Veranschaulichung des mathematischen Modells.

Wir haben also insgesamt 729 Variablen und $4 \cdot 81 = 324$ Gleichungen. In jeder Gleichung betrachten wir 9 Variablen und in jeder der vier Gleichungsketten wird jede der 729 Variablen genau einmal betrachtet. Stellen wir die Matrix A zu diesem Gleichungssystem auf, so erhalten wir eine Matrix

$$A \;\in\; \mathbb{Z}^{324 \times 729},$$

die in jeder Zeile genau 9 und in jeder Spalte genau 4 Einträge hat, die 1 sind. Alle anderen Einträge sind 0.

Zu einem gegebenen Sudoku können wir nun die vorgegebenen Variablen auf 1 setzen, indem wir der Matrix A Zeilen hinzufügen. Ist zum Beispiel im Feld $(4,8)$ die Zahl 3 vorgegeben, so ergänzen wir eine Zeile, in der nur die Variable $x_{4,8,3}$ auf 1 gesetzt ist und alle anderen Variablen 0 sind. Mit

$$b \;=\; (1,1,1,\ldots,1) \;\in\; \mathbb{Z}^{324+v}$$

(wobei v die Anzahl der vorgegebenen Variablen bezeichne) ist die Lösung unseres Sudokus die eindeutige nicht negative ganzzahlige Lösung des Gleichungssystems

$$A \cdot x \;=\; b.$$

3 Wenige Grundlagen

Bevor wir zu den Lösungsideen übergehen können, müssen wir einfache Grundlagen klären.

Definition 3.1

Wir betrachten das ***ganzzahlige lineare Programm***

$$\min c^T x \quad \text{so dass} \quad Ax = b, \quad x \in \mathbb{Z}^n, \quad x \geq 0. \tag{1}$$

Die ***Relaxierung*** von (1) ist

$$\min c^T x \quad \text{so dass} \quad Ax = b, \quad x \in \mathbb{R}^n, \quad x \geq 0. \tag{2}$$

Wir haben also nur den zulässigen Bereich vergrößert.

Lemma 3.2

Sei (P) die Relaxierung eines ganzzahligen linearen Programmes (IP), sei x^P optimal für (P) und x^{IP} optimal für (IP).

Dann gilt

$$c^T x^P \leq c^T x^{IP}.$$

Der optimale Zielfunktionswert der Relaxierung ist also immer kleiner oder gleich des optimalen Zielfunktionswertes des ganzzahligen Programmes.

Beweis

Es gilt sofort

$$c^T x^P = \min_{x \in \mathbb{R}^n} c^T x \leq \min_{x \in \mathbb{Z}^n} c^T x = c^T x^{IP},$$

da $\mathbb{Z}^n \subset \mathbb{R}^n$. □

Lemma 3.3

Sei (P) die Relaxierung eines ganzzahligen linearen Programmes (IP).

Ist x^* optimal für (P) und gilt $x^* \in \mathbb{Z}^n$, so ist x^* auch optimal für (IP).

Beweis

Es gilt

$$c^T x^* = \min_{x \in \mathbb{R}^n} c^T x \leq c^T x' \qquad \text{für alle} \quad x' \in \mathbb{Z}^n.$$

Somit ist $c^T x^*$ eine untere Schranke für (IP) und falls $x^* \in \mathbb{Z}^n$, so ist x^* auch optimal für (IP). □

Die einfachste Lösungsidee zur Minimierung von ganzzahligen Programmen wäre die folgende: Wir lösen das Problem über ganz $\mathbb{R}^n$ und erhoffen uns eine Lösung $x^* \in \mathbb{Z}^n$. Ist dies nicht der Fall, dann runden wir das Ergebniss auf eine $x' \in \mathbb{Z}^n$. Abbildung 4 verdeutlicht aber, dass wir damit sehr schlechte Ergebnisse erzielen können.

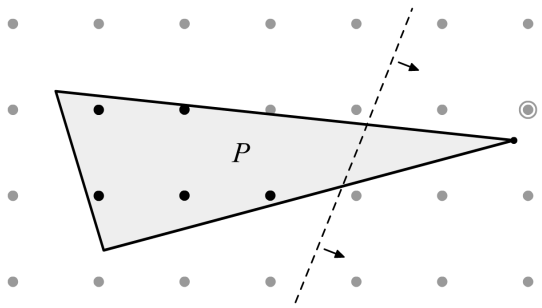

Abbildung 4: Die gerundete Lösung liegt weit weg von der ganzzahligen Lösung.

Die gerundete Lösung liegt sehr weit von der Lösung des ganzzahligen Problems entfernt. Bei ganzzahliger Programmierung brauchen wir also weitere Ideen.

4 Lösungsideen für das Sudokumodell

Wir betrachten wieder unser Modell mit den 729 Variablen und dem Gleichungssystem

$$A \in \mathbb{Z}^{324\times 729} \qquad \text{und} \qquad b \in \mathbb{Z}^{324}.$$

Wir wissen, dass jede Variable nur die Werte 0 und 1 annehmen kann, somit gibt es

$$2^{729} \approx 3 \cdot 10^{219}$$

Möglichkeiten für die Wahl der Variablen. Wenn die Probe zu jeder dieser Möglichkeiten nur $0,00000000001$ Sekunden dauern würde, bräuchte man 10^{200} Jahre, um alle Kombinationen auzuprobieren. Wir wissen zwar, dass immer genau 81 Variablen den Wert 1 haben und alle anderen Variablen 0 sind, trotzdem soll die Zahl 10^{200} verdeutlichen, dass ein einfaches Ausprobieren nicht zur Lösung des Problems dienen kann.

Einfaches Minimierungsproblem

Zunächst wollen wir das Problem als einfaches Minimierungsproblem behandeln. Wir betrachten also das Problem

$$\min c^T x \qquad \text{so dass} \qquad Ax = b, \quad x \in \mathbb{Z}^{729}, \quad x \geq 0.$$

Dabei ist A die bereits vorgestellt 324×729 Matrix und es gilt $b = (1, \ldots, 1)$. Damit die vorgegeben Variablen den Wert 1 annehmen, setzen wir einfach die entsprechenden Kosten im Kostenvektor c auf 0. Die Kosten aller anderen Variablen seien 1. Dies hat den gleichen Effekt wie das Hinzufügen von Zeilen an die Matrix A, allerdings sparen wir uns somit zusätzliche Nebenbedingungen.

Wenn wir wieder mit v die Anzahl der vorgegebenen Variablen bezeichnen, so kennen wir den optimalen Zielfunktionswert: Dieser ist nämlich

$$81 - v.$$

Zu einem gegebenen Sudoku mit einer eindeutigen Lösung erhalten wir also stets genau eine optimale ganzzahlige Lösung. Im Allgemeinen gibt es jedoch auch noch weitere optimale Lösungen, die nicht ganzzahlig sind.

Wir können aber zunächst die Relaxierung des Minimierungsproblems algorithmisch einfach lösen. Ist die Lösung der Relaxierung ganzzahlig, so haben wir nach Lemma 3.3 auch die Lösung des ganzzahligen Programmes berechnet und damit die Lösung des Sudokus erhalten. Leider ist die Lösung der Relaxierung aber nur in wenigen Fällen ganzzahlig.

Logische Variablen erkennen

Anschauliche Interpretation

Die folgende Idee soll an Hand von Abbildung 5 veranschaulicht werden.

							3	**1**
				3				
3								
						3		

Abbildung 5: Mit nur vier vorgegebenen Feldern kann sofort ein weiteres Feld gelöst werden.

Hier wurden nur die drei Felder mit den schwarzen Dreien und das eine Feld mit der Eins vorgegeben. Da aber zum Beispiel auch in der ersten Zeile

eine Drei stehen muss und da weder in einem 3×3 Quadrat noch in einer Spalte mehr als eine Drei vorkommen darf, bleibt nur ein mögliches Feld für die Drei aus der ersten Zeile übrig. Diese einfache Idee lässt sich nun auch algorithmisch lösen.

Übertragung auf das Modell

Bei jedem Sudoku sind einige Felde vorgegeben. In unserem Modell heißt das, dass wir einige Variablen sicher auf 1 setzen können. Nun wissen wir aber auch, dass gilt:

(1) In dem vorgegebenen Feld kann natürlich keine andere Zahl stehen.

(2) In der Spalte der vorgegebenen Zahl darf diese Zahl kein weiteres Mal vorkommen.

(3) In der Zeile der vorgegebenen Zahl darf diese Zahl kein weiteres Mal vorkommen.

(4) In dem 3×3 Quadrat der vorgegebenen Zahl darf diese Zahl kein weiteres Mal vorkommen.

Dies können wir algorithmisch folgendermaßen lösen: Wir wissen, dass es genau vier Zeilen in der Matrix A gibt, in denen die eine spezielle vorgegebene Variable $x_{i,j,k}$ betrachtet wird. Da diese Variable den Wert 1 annimmt, müssen alle anderen Variablen, die in diesen vier Zeilen vorkommen, den Wert 0 annehmen. Dadurch können wir mit jeder vorgegebenen Variable bis zu $4 \times 8 = 32$ weitere Variablen auf 0 setzen, also bis zu 32 weitere Variablen lösen. Wir können somit mehrere Spalten der Matrix A entweder entfernen oder alle Einträge dieser Spalten auf 0 setzen. Wiederholen wir dieses Vorgehen mit jeder vorgegebenen Variable, so kann es sein, dass in der Matrix A Zeilen entstehen, die nur einen von Null verschiedenen Eintrag haben, der 1 ist. Dies bedeutet aber, dass wir eine weitere Variable auf 1 setzen können und somit ein weiteres Feld gelöst haben.

Mit dieser Idee ist es teilweise möglich ein Sudoku komplett zu lösen oder zumindest bis zu 650 von den 729 Variablen zu erkennen. Nun können wir das deutlich vereinfachte System erneut als Minimierungsproblem betrachten, dessen Relaxierung lösen und auf eine ganzzahlige Lösung hoffen.

Eine derartige Idee zur Vereinfachung des Problems, bevor wir versuchen das ganzzahlige Programm zu lösen, wird als ***Presolving*** bezeichnet.

Leider ist auch hier festzustellen, dass wir nur zu den Sudokus eine ganzzahlige Lösung erhalten, bei denen wir auch zuvor schon eine ganzzahlige Lösung hatten.

Zulässigkeit prüfen

Wir brauchen also eine weitere *Vorstufe*, bevor wir zum Sudokumodell eine ganzzahlige Lösung des Minimierungsproblems erhalten. Dazu verwenden wir eine Idee, die sich nur noch schwer anschaulich verstehen lässt:

Anschauliche Interpretation

Dieses etwas trickreiche Verfahren soll an Hand von Abbildung 6 veranschaulicht werden.

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

Abbildung 6: Durch trickreiche Überlegungen lässt sich ein weiteres Feld lösen.

Hier kann die graue Zwei in der ersten Spalte gelöst werden, ohne dafür raten zu müssen.

Übertragung auf das Modell

Zu den im Durchschnitt 30 bis 120 Variablen, die nach dem Presolving noch nicht gelöst sind, führen wir einen zusätzlichen Test durch. Wir setzen diese Variablen alle nacheinander testweise auf 1 und untersuchen, ob unser Minimierungsproblem mit dieser zusätzlich auf 1 gesetzten Variable überhaupt noch zulässig ist. Ist dies nicht der Fall, so impliziert dies, dass die testweise auf 1 gesetzte Variable den Wert 0 annehmen muss.

Ein derartiges Testverfahren wird als ***Probing*** bezeichnet. Nach dem Probing sind in der Regel nur noch 20 bis 60 Variablen ungelöst. Wenn wir nun das stark vereinfachte Sudokurätsel als Minimierungsproblem betrachten, so ist die Lösung bei allen bekannten Sudokus mit einer eindeutigen Lösung ganzzahlig. Die Autoren aus [2] behaupten, dass sie mit diesem Probing-Verfahren alle ihnen bekannten 15.000 Sudokus gelöst haben. Trotzdem ist

dies natürlich kein Beweis dafür, dass sich alle Sudokus mit einer eindeutigen Lösung auf diese Art und Weise lösen lassen.

5 Diskussion

Zunächst einmal haben wir am Beispiel von Sudokus festgestellt, wie schwierig es sein kann ganzzahlige Programme zu lösen. Wir wollen nun noch kurz diskutieren, wie im Allgemeinen ganzzahlige Probleme gelöst werden können.

Backtracking

Eines sollte auch noch klar sein: Natürlich können Sudokus auch ohne ganzzahliger Programmierung gelöst werden. Wir können nach dem Presolving ein ***Backtracking***-Verfahren starten. Wir verwenden dazu das gleiche Modell und wenn keine Variablen mehr durch einfache Implikationen gelöst werden können, erraten wir einfach eine der ungelösten Variablen. Wenn wir mit dieser geratenen Lösung das Sudoku komplett lösen können, – eventuell müssen dabei natürlich weitere Variablen erraten werden – wurde richtig geraten. Lässt sich das Sudoku nicht komplett lösen haben wir offenbar falsch geraten und müssen zum ersten erratenen Feld zurückkehren und anders raten.

Allgemein ist Backtracking ein Verfahren zur Lösung von ***Constraint satisfaction*** Problemen. Dabei handelt es sich um Probleme, bei denen ganz allgemein Zustände oder Objekte gefunden werden müssen, die eine Anzahl von Bedingungen oder Kriterien erfüllen.

Total unimodular

Eine $m \times n$ Matrix A heißt ***total unimodular***, wenn jede quadratische Submatrix von A eine Determinante hat, die -1, 1 oder 0 ist. Daraus folgt natürlich auch, dass die Matrix A selber nur Einträge haben darf, die -1, 1 oder 0 sind. Dies ist auch bei unserer Matrix für das Sudokumodell der Fall.

Für Minimierungsprobleme mit derartigen Matrizen A lässt sich zeigen, dass für jeden Vektor $b \in \mathbb{Z}^m$ und für jeden Kostenvektor $c \in \mathbb{R}^n$ die Lösung der entsprechenden Relaxierung ganzzahlig ist.

Solche ganzzahligen Probleme lassen sich also noch vergleichsweise einfach lösen. Leider ist die Matrix A zu unserem Modell nicht total unimodular, obwohl sie nur sehr wenige von Null verschiedene Einträge hat, die 1.

Presolving und Probing

Wie bereits beschrieben handelt es sich beim ***Presolving*** um Verfahren, mit denen anhand von Implikationen, die aus dem ganzzahligen Programm abgeleitet werden können, dieses zu vereinfachen. Beim ***Probing*** liegt eine verstärkte Form des Presolvings vor. Hier werden keine einfachen Implikationen sondern unterschiedliche Argumente verwendet, um das Problem zu vereinfachen.

Man hofft sich hierbei das Problem soweit vereinfachen zu können, dass die Lösung der Relaxierung ganzzahlig ist. Ist dies trotzdem nicht der Fall, muss auf Schnittebenenverfahren zurückgegreifen werden oder das Problem muss in Teilprobleme zerlegt werden. Diese beiden Verfahren sind jedoch sehr aufwendig (siehe nächsten Abschnitt).

Moderen Software für ganzzahlige Programme sind sehr komplex. Sie behandeln eine große Anzahl von Heuristiken, führen danach wenn möglich Presolving sowie Probing durch, versuchen eine ganzzahlige Lösung aus der gebrochenen Relaxierung zu berechnen und greifen nur zur Not auf Schnittebenenverfahren zurück.

Schnittebenenverfahren

Erhält man bei einem ganzzahligen linearen Programm auch mit Presolving und Probing keine Lösung, kann zum Beispiel das ***Schnittebenenverfahren*** angewandt werden.

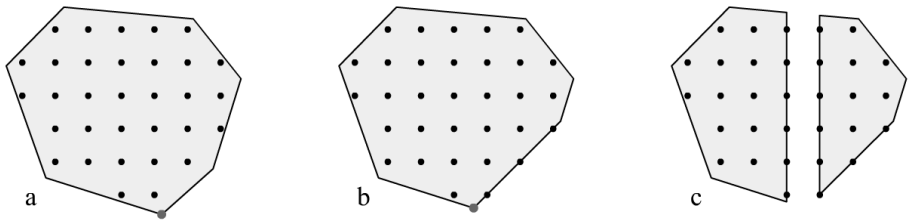

Abbildung 7: Lösen der kontinuierlichen Relaxierung, Hinzufügen von Schnittebenen und Zerlegung in Teilprobleme.

Dazu bestimmt man eine untere Schranke des Problems durch Berechnung der kontinuierlichen Relaxierung des ganzzahligen linearen Programms (Abbildung 7a). Die Relaxierung wird dann durch Hinzufügen von Schnittebenen weiter verstärkt (Abbildung 7b). Ist keine weitere Verbesserung der Schranke mehr möglich, wird das Problem in zwei kleinere Teilprobleme zerlegt (Abbildung 7c).

Fazit

Zumindest eines sollte nun klar geworden sein:

Ganzzahlige Optimierung ist schwer*!*

Literatur

[1] Felgenhauer, B. ; Jarvis, F.: Mathematics of Sudoku I. In: *Mathematical Spectrum* 15 (2006), S. 15–23

[2] Kaibel, V. ; Koch, T.: Mathematik für den Volkssport. In: *DMV-Mitteilungen* 14 (2006), S. 93–96

[3] Russell, E. ; Jarvis, F.: Mathematics of Sudoku II. In: *Mathematical Spectrum*. – forthcoming

[4] Schöbel, A.: *Optimierung*. 2006. – Skript zur Vorlesung im Sommersemester 2005 an der Universität Göttingen

[5] Wikipedia: *Backtracking*. 17. Oktober 2006. – Aufgerufen unter der Adresse `http://en.wikipedia.org/wiki/Backtracking`

[6] Wikipedia: *Constraint satisfaction problem*. 17. Oktober 2006. – `http://en.wikipedia.org/wiki/Constraint_satisfaction_problem`

[7] Wikipedia: *Sudoku*. 17. Oktober 2006. – Aufgerufen unter der Adresse `http://en.wikipedia.org/wiki/Sudoku`