

Sudokus lösen mit MATLAB

Von Cleve Moler

Menschliche Rätselfreunde und Computerprogramme lösen Sudoku mithilfe sehr verschiedener Methoden. Die Faszination, ein Sudoku per Hand zu lösen, entspringt der Freude an der Entdeckung und Beherrschung unzähliger subtiler Kombinationen und Muster, die Hinweise auf die abschließende Lösung geben. Ein Computer dagegen lässt sich nur schwer auf eine Weise programmieren, die diese menschliche Fähigkeit zur Mustererkennung eins zu eins kopiert. Die meisten Programme zur Lösung von Sudokus schlagen darum einen völlig anderen Weg ein: Er beruht auf der nahezu unbegrenzten Leistungsfähigkeit des Computers, mit schierer Rechenleistung Lösungen systematisch nach Versuch und Irrtum auszuprobieren. Diese Methode habe ich auch für mein MATLAB®-Programm gewählt.

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

ABB. 1: Ein Beispielrätsel, die Hinweise sind blau dargestellt. Dieses Beispiel hat eine besonders schöne Symmetrie.

Herausforderung Sudoku

Wie sie wahrscheinlich wissen, muss man zum Lösen eines Sudoku ein 9-mal-9-Gitter so füllen, dass alle Zeilen und Spalten sowie alle 3-mal-3-Unterquadrate jeweils alle Ziffern von 1 bis 9 enthalten. Anfangs ist das Gitter nur mit wenigen Ziffern besetzt, den Hinweisen. Im Gegensatz zu Magischen Quadraten und anderen Zahlenrätseln muss man beim Sudoku nicht rechnen; die Elemente in einem Sudoku-Gitter könnten ebenso gut Buchstaben oder beliebige Symbole sein.

Abbildung 1 zeigt ein Anfangsgitter. Mir gefällt an diesem von Gordon Royle von der

University of Western Australia stammenden Beispiel besonders seine Symmetrie. In Abbildung 2 ist die Lösung dargestellt.

Lösen von Sudokus durch Rekursives Backtracking

Unser MATLAB-Programm arbeitet mit nur einem Muster, den einelementigen Mengen oder kurz Singletons, sowie einer ganz grundlegenden Methode der Informatik, dem Rekursiven Backtracking.

Um zu verstehen, wie das Programm funktioniert, betrachten wir zunächst ein

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

ABB. 2: Das gelöste Rätsel. Die fehlenden Ziffern sind eingetragen und jede Zeile, Spalte sowie jedes 3-mal-3-Unterquadrat enthält die Ziffern von 1 bis 9.

einfacheres 4-mal-4-Gitter mit 2-mal-2-Unterquadraten. Solche Rätsel nennt man nicht Sudoku, sondern Shidoku, weil „Shi“ auf Japanisch „vier“ bedeutet.

Abbildung 3 zeigt unser erstes Shidoku. Der Lösungsweg ist in den Abbildungen 4 bis 6 aufgeführt. In Abbildung 4 sind die erlaubten Einträge, die Kandidaten, als kleine Ziffern dargestellt. In Zeile zwei finden wir beispielsweise eine „3“ und in Spalte eins eine „1“, so dass für die Position (2,1) nur noch „2“ und „4“ in Frage kommen. Vier der Zellen enthalten nur einen einzigen Kandidaten. Diese Zellen sind

die Singletons. In Abbildung 5 wurde eines der Singletons eingetragen und danach die Kandidaten neu berechnet. In Abbildung 6 sind alle noch verbleibenden Singletons eingetragen, die Lösung ist bereits eindeutig abzulesen.

Man könnte als leichtes Rätsel eines definieren, das sich schon durch Einsetzen von Singletons lösen lässt. Innerhalb dieser Definition war das erste Beispiel leicht – ganz im Gegensatz zum folgenden.

Das Anfangsgitter für das in Abbildung 7 gezeigte Rätsel wird generiert durch die MATLAB-Anweisung

```
x = diag(1:4)
```

Da es in diesem Rätsel zu Anfang keine Singletons gibt (Abb. 8), lösen wir es durch Rekursives Backtracking. Wir wählen eine der leeren Zellen und setzen versuchsweise einen der Kandidaten ein. Dabei gehen wir die Zellen in der Reihenfolge der von MATLAB vorgegebenen eindimensionalen Indizierung, X(:), durch und probieren darin die Kandidaten in numerischer Reihenfolge aus. In Zelle (2,1) setzen wir demzufolge eine „3“ ein und erzeugen so ein neues Rätsel (Abb. 9). Nun rufen wir das Programm rekursiv auf.

Das neue Rätsel ist leicht und Abbildung 10 zeigt sein Ergebnis. Diese Lösung hängt aber von der vor dem rekursiven Aufruf getroffenen Auswahl ab. Durch andere Einsetzungen könnten sich andere Lösungen ergeben. Für diese einfache diagonale Ausgangssituation gibt es beispielsweise zwei mögliche Lösungen, die zufällig die Matrixtransponierten der jeweils anderen sind. Da die Lösung nicht eindeutig ist, ist das Gitter aus Abbildung 7 kein gültiges Rätsel.

Existenz und Eindeutigkeit

Als Mathematiker möchten wir beweisen, dass für ein Problem eine Lösung existiert und dass diese Lösung eindeutig ist. Beim Sudoku lassen sich aber weder Existenz noch Eindeutigkeit auf einfache Weise aus den zu Anfang vorhandenen Hinweisen ableiten. Würden wir bei dem in Abbildung 1 gezeigten Rätsel etwa eine „1“, „5“ oder „7“ in die Zelle (1,1) einsetzen

1			
		3	
	2		
			4

ABB. 3: Ein Shidoku ist ein Sudoku auf einem 4-mal-4-Gitter.

1	3 4	2 4	2
2 4	4	3	1 2
3 4	2	1	1 3
3	1 3	1 2	4

ABB. 4: Die Kandidaten; rote Kandidaten sind Singletons.

1	3 4	2 4	2
2 4	4	3	1 2
4	2	1	1 3
3	1	1 2	4

ABB. 5: Eintragen des Singleton „3“ und Neuberechnung der Kandidaten.

1	3	4	2
2	4	3	1
4	2	1	3
3	1	2	4

ABB. 6: Mit dem Eintragen der restlichen Singletons ist das Rätsel gelöst.

1			
	2		
		3	
			4

ABB. 7: (diag(1:4))

1	3 4	2 4	2 3
	2	1 4	1 3
2 4	1 4	3	1 2
2 3	1 3	1 2	4

ABB. 8: Die Kandidaten. Es gibt keine Singletons.

1			
3	2		
		3	
			4

ABB. 9: Durch probeweises Eintragen von „3“ wird ein neues Rätsel erzeugt. Anschließend folgt das Backtracking.

1	4	2	3
3	2	4	1
4	1	3	2
2	3	1	4

ABB. 10: Die resultierende Lösung. Sie ist nicht eindeutig, denn ihre Transponierte ist eine weitere Lösung.

zen, wären zwar die Zeilen-, Spalten- und Bedingungen für die Unterquadrante erfüllt, aber das so erzeugte Rätsel hätte keine Lösung. Sie würden sich wahrscheinlich ärgern, wenn Sie dieses Rätsel in ihrer Zeitung fänden.

Backtracking erzeugt viele unmögliche Anordnungen. Unser Programm beendet darum die Rekursion, sobald es eine Zelle findet, für die es keine Kandidaten gibt. Solche Rätsel haben keine Lösung.

Eindeutigkeit ist eine nur schwer fassbare Eigenschaft. In den meisten Beschreibungen für Sudokus wird nicht explizit erwähnt, dass es nur eine Lösung geben darf. Aber natürlich würde man sich ärgern, wenn man eine andere Lösung fände als die angegebene. Einige der Rätsel-Generatoren, die man auf MATLAB Central findet, überprüfen nicht, ob die erzeugten Rätsel eindeutig sind. Die einzige mir bekannte Möglichkeit auf

Eindeutigkeit zu prüfen ist, sämtliche möglichen Lösungen aufzulisten.

Der Lösungsalgorithmus für Sudokus

Unser MATLAB-Programm umfasst nur vier Schritte:

1. Trage alle Singletons ein.
2. Breche ab, wenn es für eine Zelle keine Kandidaten gibt.
3. Trage in eine leere Zelle einen Probestwert ein.
4. Rufe das Programm rekursiv auf.

Die interne Schlüsselfunktion ist `candidates`. Jede leere Zelle startet mit $z = 1:9$ und setzt alle Elemente in z auf Null, deren Zahlenwert in der zugehörigen Zeile, Spalte oder dem Unterquadrat vorkommen. Die verbleibenden Nicht-Nullwerte sind die Kandidaten. Nehmen Sie etwa die Zelle (1,1) in Abbildung 1. Wir beginnen mit

$z = 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9$

Mit den Werten aus der ersten Zeile wird z zu

$z = 1\ 0\ 0\ 0\ 5\ 6\ 7\ 8\ 9$

Die erste Spalte wiederum ändert z in

$z = 1\ 0\ 0\ 0\ 5\ 0\ 7\ 0\ 9$

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

ABB. 11: Die Situation nach nur 22 Schritten auf dem Weg zur Lösung von Abbildung 1. Cyanfarbene Werte sind durch Backtracking erzeugte Probestwerte, grüne Werte sind die dadurch entstehenden Singletons. Die „1“ ist die falsche Wahl für die Zelle (1,1).

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

ABB. 12: Nach 14781 Schritten scheint die Lösung nahe, aber es gibt keine Fortsetzungsmöglichkeit, weil für Zelle (1,9) keine Kandidaten übrig sind. Die „7“ ist die falsche Wahl für Zelle (1,1).

Lösung von Sudokus durch Rekursives Backtracking

```
function X = sudoku(X)
% SUDOKU Solve Sudoku using recursive backtracking.
% sudoku(X), expects a 9-by-9 array X.
% Fill in all "singletons".
% C is a cell array of candidate vectors for each cell.
% s is the first cell, if any, with one candidate.
% e is the first cell, if any, with no candidates.
[C,s,e] = candidates(X);
while ~isempty(s) && isempty(e)
    X(s) = C{s};
    [C,s,e] = candidates(X);
end
% Return for impossible puzzles.
if ~isempty(e)
    return
end
% Recursive backtracking.
if any(X(:) == 0)
    Y = X;
    z = find(X(:) == 0,1); % The first unfilled cell.
    for r = [C{z}] % Iterate over candidates.
        X = Y;
        X(z) = r; % Insert a tentative value.
        X = sudoku(X); % Recursive call.
        if all(X(:) > 0) % Found a solution.
            return
        end
    end
end
end
% -----
function [C,s,e] = candidates(X)
C = cell(9,9);
tri = @(k) 3*ceil(k/3-1) + (1:3);
for j = 1:9
    for i = 1:9
        if X(i,j)==0
            z = 1:9;
            z(nonzeros(X(i,:))) = 0;
            z(nonzeros(X(:,j))) = 0;
            z(nonzeros(X(tri(i),tri(j)))) = 0;
            C{i,j} = nonzeros(z)';
        end
    end
end
L = cellfun(@length,C); % Number of candidates.
s = find(X==0 & L==1,1);
e = find(X==0 & L==0,1);
end % candidates
end % sudoku
```

Das Unterquadrat, zu dem (1,1) gehört, enthält keine weiteren Informationen. Die Kandidaten für Zelle (1,1) sind damit

$$C\{1,1\} = [1\ 5\ 7\ 9]$$

Ein schwieriges Rätsel

Das in Abbildung 1 dargestellte Rätsel ist übrigens ein sehr schweres Rätsel, egal ob Sie es per Hand oder mit dem Computer lösen. In Abbildungen 11 und 12 sind Zwischenergebnisse seiner Berechnung dargestellt. Zu Anfang sind keine Singletons vorhanden und die Lösung beginnt sofort mit einem rekursiven Schritt. Wir probieren eine „1“ in Zelle (1,1) aus. Abbildung 11 zeigt, dass auf diese Wahl hin in Schritt 22 die erste Spalte gefüllt ist. Von der Lösung sind wir damit aber noch weit entfernt. Nach 3114 Schritten trägt die Rekursion eine „5“ in Zelle (1,1) ein und probiert nach 8172 Schritten eine „7“ aus.

Abbildung 12 zeigt die Situation nach 14781 Schritten. Es sieht so aus, als näherten wir uns langsam der Lösung, denn jetzt sind 73 der 81 Zellen mit Werten gefüllt. In der ersten Zeile und der letzten Spalte sind aber zusammengekommen bereits alle Ziffern von 1 bis 9 vorhanden, womit für die Zelle (1,9) oben rechts keine Werte mehr verfügbar sind. Die Kandidatenliste für diese Zelle ist leer und die Rekursion bricht ab. Nach 19229 Schritten probieren wir schließlich eine „9“ in der ersten Zelle. Diese „9“ ist eine gute Wahl, denn das Programm erreicht weniger als 200 Schritte später – nach 19422 Schritten – die in Abbildung 2 dargestellte Lösung. Das sind erheblich mehr Schritte, als man für die meisten Rätsel braucht. ■

Quellen

EXPERIMENTS WITH MATLAB
www.mathworks.de/nn9/exm

SUDOKU-QUADRATE UND CHROMATISCHE POLYNOME
www.ams.org/notices/200706/tx070600708p.pdf

STRATEGIE-FAMILIEN
www.scanraid.com/Strategy_Families