

---

# Klassifikation der Schwierigkeitsgrade von Sudokus mit Methoden des maschinellen Lernens

---

Classification of Sudoku difficulty levels using methods of machine learning  
Bachelor-Thesis von Michael Bräunlein

---



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

Fachbereich Informatik  
Knowledge Engineering Group  
Betreuer Prof. Dr. Johannes Fürnkranz

---

---

## **Zusammenfassung**

---

Das Zahlenrätsel Sudoku ist weltweit bei Rätsel-Liebhabern bekannt und beliebt. Seit seiner Veröffentlichung 1986 begeistern sich immer mehr Menschen für mehr oder weniger schwierige Exemplare. Sudokus finden sich im Internet, in der Rätsecke der Tageszeitungen und sogar als ganze Bücher, um nur einige Erscheinungsorte zu nennen.

Die Regeln sind einfach zu lernen und doch kann man sich sehr lange mit Sudokus beschäftigen, da die schwersten Sudokus meist nur von Profis gelöst werden können.

Der Spielspaß ist sehr stark davon abhängig, dass die Schwierigkeit zum persönlichen Können passt. Ist das Sudoku zu leicht, stellt es keine Herausforderung dar. Ist es zu schwer, dann kommt schnell ein Gefühl der Überforderung auf.

Die ausgewiesenen Schwierigkeitsstufen von Sudokus aus verschiedenen Quellen haben zwar oft die gleichen Namen wie zum Beispiel "Mittel" oder "Schwer", unterscheiden sich aber dennoch häufig nach Meinung des Spielers.

Das Ziel dieser Bachelorarbeit ist, Merkmale aus Sudokus zu extrahieren, anhand derer die Sudokus von einem Klassifizierer möglichst zuverlässig in Schwierigkeitsstufen eingeteilt werden können.

---

## Inhaltsverzeichnis

---

<b>1</b>	<b>Einleitung</b>	<b>3</b>
1.1	Motivation . . . . .	3
1.2	Fragestellung . . . . .	3
1.3	Struktur . . . . .	3
<b>2</b>	<b>Grundlagen</b>	<b>4</b>
2.1	Die Regeln . . . . .	4
2.2	Begriffserklärung . . . . .	4
2.3	Existierende Einteilungsschemata . . . . .	4
2.4	Lösungsmethoden . . . . .	6
2.4.1	Kandidatenlisten . . . . .	6
2.4.2	Full House . . . . .	7
2.4.3	Naked Single . . . . .	8
2.4.4	Hidden Single . . . . .	9
2.4.5	Pointing Pair / Triple . . . . .	10
2.4.6	Box-Line Reduction . . . . .	11
2.4.7	Naked Subset . . . . .	12
2.4.8	Hidden Subset . . . . .	13
2.4.9	Fish . . . . .	14
2.4.10	Single Digit Patterns . . . . .	16
2.4.11	Wings . . . . .	20
2.4.12	Sue de Coq . . . . .	23
2.4.13	Coloring . . . . .	24
2.4.14	Almost Locked Set . . . . .	25
2.4.15	Backtracking . . . . .	29
<b>3</b>	<b>Maschinelles Lernen zur Charakterisierung des Schwierigkeitsgrades</b>	<b>30</b>
3.1	Klassifikation . . . . .	30
3.2	Merkmalsextrahierung . . . . .	30
3.3	Aufbau des Featurevectors . . . . .	31
3.4	Entkopplung von konkreten Zahlen . . . . .	33
<b>4</b>	<b>Software</b>	<b>34</b>
4.1	Eigene Software . . . . .	35
4.2	Fremdsoftware . . . . .	35
4.3	Trainingsdaten . . . . .	35
<b>5</b>	<b>Ergebnisse</b>	<b>37</b>
<b>6</b>	<b>Zusammenfassung und Ausblick</b>	<b>38</b>

---

---

## **1 Einleitung**

---

### **1.1 Motivation**

---

### **1.2 Fragestellung**

---

### **1.3 Struktur**

---

---

## 2 Grundlagen

---

### 2.1 Die Regeln

---

Diese Arbeit beschäftigt sich nur mit der meist verbreiteten Art von Sudoku. Dabei spielt man auf einem 9x9 Felder großen Spielfeld, das wiederum in neun 3x3 Felder große Blöcke eingeteilt ist. Weiter handelt es sich nur dann um ein Sudoku, wenn genau eine Lösung vorhanden ist. Ein Sudoku gilt dann als gelöst, wenn jede Zeile, jede Spalte und jeder Block die Ziffern 1 bis 9 genau einmal enthält.

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

Abbildung 2.1: Sudoku

Ein Sudoku hat immer eine eindeutige Lösung, sonst ist es kein Sudoku.

---

### 2.2 Begriffserklärung

---

Ein Sudoku besteht aus 81 *Feldern* oder *Zellen*. In diese werden die *Ziffern* oder *Zahlen* eingetragen und sie bilden ein Quadrat der Größe 9x9, das *Grid*. Aufgrund dieser Aufteilung hat ein Sudoku 9 *Zeilen* und 9 *Spalten*. Das Grid wird in 9 Unterquadrate geteilt, die jeweils 3x3 Felder groß sind. Diese werden *Blöcke* genannt. Zeilen, Spalten und Blöcke werden unter dem Begriff *Figur* zusammengefasst. Die Nummerierung der Blöcke erfolgt zeilenweise von links oben nach rechts unten.

*Vorgaben* sind Zahlen, die schon von Anfang an gegeben sind.

In **Abbildung 2.2** sieht man im mittleren Block sogenannte *Kandidaten*. Ein Kandidat ist eine Zahl, die in der Zelle noch möglich ist. Jede Zelle hat ihre eigene Liste mit Kandidaten.

In der Beschreibung der Lösungstechniken ist es notwendig bestimmte Felder zu betrachten. Hierzu wird eine Abkürzung verwendet, die Zeile und Spalte enthält und somit eine Zelle eindeutig indentifiziert. *z2s3* meint zum Beispiel die Zelle in Zeile 2 und Spalte 3.

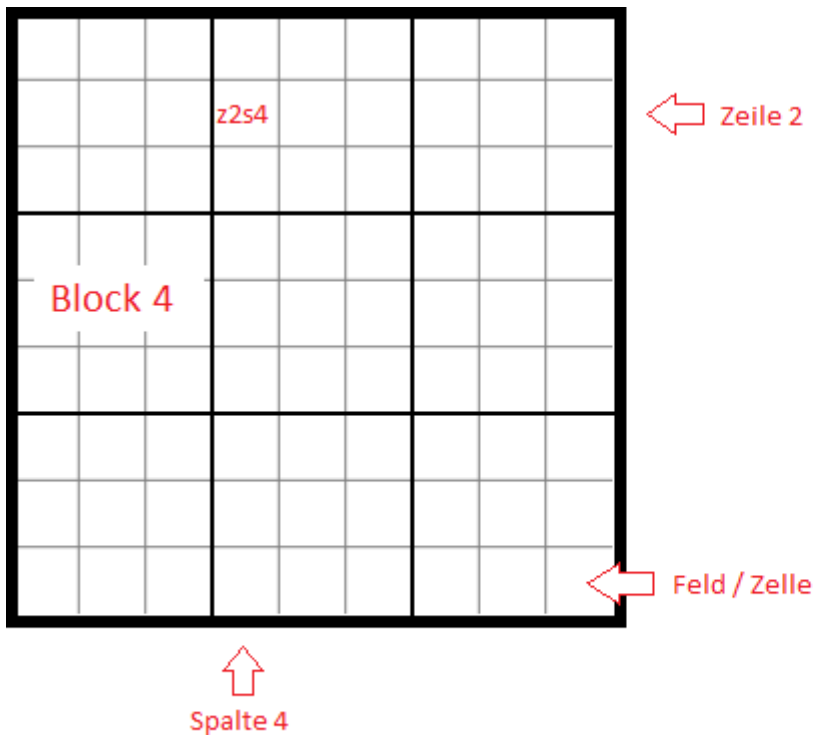
In der folgenden Abbildung sind die erläuterten Begriffe zum besseren Verständniss eingetragen.

---

### 2.3 Existierende Einteilungsschemata

---

Für die Zuordnung eines Schwierigkeitsgrades zu einem Sudoku existieren bereits verschiedene Methoden. Die einfachsten orientieren sich nur an Merkmalen, die auf den ersten Blick sichtbar sind, wie etwa die Anzahl der vorgegebenen Ziffern. Diese Methoden haben den Vorteil, dass das Sudoku bewertet werden kann, ohne es zu lösen. Ausserdem sind sie sehr einfach zu implementieren und auch sehr schnell.



**Abbildung 2.2:** Begriffe

Die Genauigkeit solcher Methoden ist allerdings nicht besonders hoch, da die offensichtlichen Merkmale nicht mit einbeziehen, wie schwer das Lösen eines Sudokus ist, da es ja während der Bewertung des Sudokus nicht gelöst wird und somit dem Lösungsweg kein Schwierigkeitsgrad zugeordnet werden kann. Daher ist es sinnvoll, das Sudoku bei der Bewertung zu lösen und die Schwierigkeit des Lösungsweges zu schätzen. Ein Lösungsweg besteht aus einer Reihe von Lösungsschritten. Mit jedem Schritt erfährt der Spieler mehr über das Sudoku, das bedeutet, dass er eine Ziffer einsetzen kann oder dass er ausschließt, dass eine Ziffer in einem bestimmten Feld stehen kann. Gegebenenfalls kann er mit einem Lösungsschritt sogar mehrere Ziffern in einem Feld, oder eine Ziffer in mehreren Felder ausschließen. Sehr komplexe Methoden erlauben auch den gleichzeitigen Ausschluss von mehreren Ziffern in mehreren Feldern.

Es gibt also verschiedene Arten von Lösungsschritten, diese haben auch eine verschiedene Schwierigkeit. Dabei entscheidend ist für diese Arbeit die vom menschlichen Spieler empfundene Schwierigkeit und nicht etwa die Komplexität der Implementierung oder die Laufzeit der Lösungsmethode auf einem Rechner, da die Bewertung später als Orientierung für Menschen dienen soll.

Die Website <http://sudoku.soeinding.de/strategie/strategie03.php> ist ein Beispiel für ein solches Vorgehen.

Eine weitere Methode ist die Messung der Zeit, die menschliche Spieler zum Lösen von Sudokus brauchen. Hierbei ist das Problem, dass die Klassifikation eines Sudokus sehr lange dauert, da es erst von mehreren Spielern gelöst werden muss. Ausserdem ist die Erfahrung und damit die Spielstärke von Sudoku Spielern sehr unterschiedlich und sollte in die Bewertung mit einfließen.

Ein Beispiel für diesen Ansatz der Klassifikation ist diese Website <http://sudokugarden.de/de/online/dstats>.

In dieser Arbeit wird versucht, ein Einteilungsschema zu finden, dass auf der Schwierigkeit des Lösungsweges beruht, aber auch offensichtliche Merkmale einbezieht.

---

## 2.4 Lösungsmethoden

---

Alle in dieser Bachelorarbeit beschriebenen Techniken sind nicht im Rahmen dieser Arbeit entwickelt worden, sondern wurden aus verschiedenen Quellen zusammengetragen. Die Beschreibung der Lösungstechniken lehnt sich an die Beschreibung der Quellen an. Teile der Beispiele wurden aus den Quellen entnommen, dies ist entsprechend gekennzeichnet.

Grob kann man die Techniken zum Lösen von Sudokus in zwei Kategorien einteilen. Die erste Kategorie findet Zahlen heraus, die direkt in das Sudoku eingetragen werden können. Die Techniken der zweiten Kategorie entfernen Bedingungen in einzelnen Zellen des Sudokus.

---

### 2.4.1 Kandidatenlisten

---

Beim Lösen von Sudokus ist es üblich, in jedes Feld die Kandidaten einzutragen, die dort stehen können. Dabei wird vorerst nur die Sudoku Regel berücksichtigt, die besagt, dass in jeder Zeile die Zahlen 1 bis 9 vorkommen müssen. Wenn in einer Zeile nun die Zahl 3 vorkommt, dann kann sie in der selben Zeile nicht nochmal vorkommen, daher kann sie aus allen Kandidatenlisten der Zellen in der selben Zeile gelöscht werden. Dasselbe gilt für Spalten und Blöcke. Immer, wenn eine Ziffer in ein Feld eingetragen wird, dann muss der Spieler die Liste der Kandidaten aktualisieren.

Kandidatenlisten sind keine eigene Lösungstechnik, sind aber wesentlicher Bestandteil vieler Techniken.

## 2.4.2 Full House

Wenn in einer Figur 8 Zahlen eingetragen sind, dann kann die Technik *Full House* angewendet werden. Da in jeder Figur die Zahlen 1 bis 9 stehen müssen, kann die fehlende Zahl einfach per Ausschluss ermittelt werden.

2	<sup>1</sup> 4 5 6	7	<sup>1</sup> 4 5 8	<sup>1 3</sup> 5	<sup>3</sup> 8	<sup>1</sup> 6	<sup>1 3</sup> 4 5 6	<sup>1 3</sup> 5 6 9
<sup>1</sup> 4 5	8	4 5 6	<sup>1 2</sup> 4 5	9	<sup>2 3</sup> 7	<sup>1 2</sup> 6 7	<sup>1 3</sup> 4 5 6 7	<sup>1 2 3</sup> 5 6 7
<sup>1</sup> 4 5 9	3	4 5	6	<sup>1</sup> 5 7	<sup>2</sup> 7	8	<sup>1</sup> 4 5 7	<sup>1 2</sup> 5 7 9
<sup>3</sup> 5 7	<sup>5</sup> 7	8	<sup>1</sup>	6	4	9	<sup>1</sup> 7	<sup>1 2</sup> 7
6	9	2	7	8	5	3	<sup>1</sup>	4
<sup>4</sup> 7	<sup>4</sup> 7	1	3	2	<sup>9</sup>	5	<sup>6</sup> 7 8	<sup>6</sup> 7
<sup>3</sup> 4 5 7 8	<sup>4 5 6</sup> 7	9	<sup>5</sup> 8 7	<sup>5 3</sup> 7	1	<sup>6</sup> 7	2	<sup>3</sup> 5 6 7
<sup>1 3</sup> 5 7 8	<sup>1 2</sup> 5 6 7	<sup>3</sup> 5 6	<sup>2</sup> 5 8	4	<sup>2 3</sup> 6 7 8	<sup>1</sup> 6 7	9	<sup>1 3</sup> 5 6 7
<sup>1 3</sup> 5 7	<sup>1 2</sup> 5 6 7	<sup>3</sup> 5 6	<sup>2</sup> 5 9 7	<sup>3</sup> 5 7	<sup>2 3</sup> 6 7 9	4	<sup>1 3</sup> 5 6 7	8

Abbildung 2.3: Full House

In **Abbildung 3.1** fehlt in Zeile 5 nur noch eine Ziffer. Da die Zahlen 2 bis 9 bereits vorhanden sind, kann in das Feld z5s8 die Zahl 1 eingetragen werden.



### 2.4.3 Naked Single

Bei der Technik *Naked Single* werden Kandidatenlisten verwendet. Diese Technik kann angewendet werden, wenn in der Kandidatenliste einer Zelle nur noch ein Kandidat steht. Dieser Kandidat kann dann in die Zelle eingetragen werden. Das funktioniert aufgrund des Aufbaus der Kandidatenlisten. Diese enthalten zuerst alle Kandidaten und es werden immer dann Kandidaten entfernt, wenn dieser Kandidat nicht mehr als Ziffer in der Zelle stehen könnte weil er dort eine Regel verletzen würde. Wenn also nur noch ein Kandidat in der Kandidatenliste steht, dann bedeutet das, dass dieser Kandidat die einzige Ziffer zwischen 1 und 9 ist, die in der Zelle stehen kann ohne eine Regel zu verletzen.

<div>2 5 6 8</div>	<b>1</b>	<div>2 3 5 6</div>	<b>9</b>	<div>5 6</div>	<div>5 6</div>	<b>7</b>	<b>4</b>	<div>2 5 8</div>
<div>2 4 5 6 9</div>	<div>2 4 5 9</div>	<div>2 5 6</div>	<b>8</b>	<div>4 5 6</div>	<div>1 4 5 6 7</div>	<div>1 5</div>	<div>1 5</div>	<b>3</b>
<div>4 5 8</div>	<b>7</b>	<div>5</div>	<div>3</div>	<b>2</b>	<div>1 4 5</div>	<b>6</b>	<b>9</b>	<div>1 5 8</div>
<div>1 5 6 7 9</div>	<div>5 9</div>	<b>4</b>	<div>5 7</div>	<b>3</b>	<div>5 7 9</div>	<b>2</b>	<div>1 5 6 7 8</div>	<div>1 5 6 7 8 9</div>
<div>1 5 7 9</div>	<div>3 5 7 9</div>	<div>3 5</div>	<b>6</b>	<div>4 5 8 9</div>	<b>2</b>	<div>1 4 5 8 9</div>	<div>1 5 7 8</div>	<div>1 4 5 7 8 9</div>
<div>2 5 6 7 9</div>	<div>2 5 9</div>	<b>8</b>	<div>4 5 7</div>	<b>1</b>	<div>4 5 7 9</div>	<b>3</b>	<div>5 6 7</div>	<div>4 5 6 7 9</div>
<div>2 4 5</div>	<b>8</b>	<b>1</b>	<div>2 4 5</div>	<b>7</b>	<div>4 5 6 9</div>	<div>4 5 9</div>	<b>3</b>	<div>4 5 6 9</div>
<b>3</b>	<div>2 4 5</div>	<div>2 5 7</div>	<div>1 2 4 5</div>	<div>4 5 6 9</div>	<b>8</b>	<div>1 4 5 9</div>	<div>1 5 6 7</div>	<div>1 4 5 6 7 9</div>
<div>4 5 7</div>	<b>6</b>	<b>9</b>	<div>1 4 5</div>	<div>4 5</div>	<b>3</b>	<div>1 4 5 8</div>	<b>2</b>	<div>1 4 5 7 8</div>

Abbildung 2.4: Naked Single

Im oben stehenden Beispiel **Abbildung 3.2** sieht man sofort, dass die Kandidatenliste in z3s3 nur noch einen Eintrag enthält. Dieser kann nun einfach eingetragen werden.

#### 2.4.4 Hidden Single

Auch die Technck *Hidden Single* arbeitet wieder mit Kandidatenlisten. Wenn in einer Figur eine Kandidatenliste die einzige ist, in der eine bestimmte Zahl vorkommt, dann kann diese Zahl direkt in die Zelle eingetragen werden. Wenn in dieser Zelle die Zahl nicht stünde, dann gäbe es in der Figur keine Möglichkeit mehr, dass die Zahl auftaucht und damit wäre die Sudoku Regel verletzt, nach der jede Zahl genau einmal enthalten sein muss.

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

Abbildung 2.5: Hidden Single

In **Abbildung 3.3** sieht man, dass die Zahl 6 in der Zeile 3 nur in z3s4 vorkommen kann. Daher kann man sie dort eintragen.

### 2.4.5 Pointing Pair / Triple

Bei der Technik *Pointing Pair / Triple* müssen zum ersten mal die Kandidatenlisten mehrerer Felder gleichzeitig betrachtet werden, was diese Technik etwas schwerer macht. Ausserdem ist diese Technik die erste, die Kandidaten aus Kandidatenlisten entfernt und nur bedingt zum Einsetzen von Zahlen in das Sudoku führt.

Es werden die Kandidatenlisten in Blöcken jeweils zeilen- und spaltenweise betrachtet. Die Technik *Pointing Pair / Triple* kann angewendet werden, wenn in einem Block eine Kandidat nur in Kandidatenlisten der selben Zeile oder Spalte vorkommt. Dann kann jedes weitere vorkommen der Zahl in einer Kandidatenliste der selben Zeile oder Spalte entfernt werden. Das gilt, da die Zahl genau einmal in dem Block vorkommen muss. Da alle möglichen Vorkommen der Zahl in der selben Zeile oder Spalte liegen ist klar, dass die Zahl in dieser Zeile oder Spalte vorkommt. Da sie aber kein zweites mal in der Zeile oder Spalte vorkommen darf muss sie aus den Kandidatenlisten entfernt werden, die nicht im selben Block liegen.

3	4	<sup>1</sup> 5 <sub>9</sub>	<sup>1 2</sup> 5 <sub>8</sub>	<sup>5</sup> 8 <sub>9</sub>	6	<sup>1 2</sup> 5	7	<sup>1 2</sup> 8
<sup>1</sup> 5 <sub>7</sub> 6	8	<sup>1</sup> 5 <sub>6</sub>	<sup>1 2</sup> 4 <sub>5</sub> 7	<sup>4</sup> 5 <sub>7</sub>	<sup>1</sup> 5 <sub>7</sub>	9	3	<sup>1 2</sup> 4
<sup>1</sup> 5 <sub>7</sub> 9	<sup>1</sup> 5 <sub>7</sub>	2	<sup>1</sup> 4 <sub>5</sub> 7	3	<sup>1</sup> 5 <sub>7</sub> 9	<sup>1</sup> 4 <sub>5</sub>	6	<sup>1</sup> 4 <sub>8</sub>
<sup>2</sup> 4 <sub>5</sub> 8	<sup>2 3</sup> 5 <sub>6</sub>	<sup>3</sup> 4 <sub>5</sub> 8	<sup>5</sup> 7 <sub>8</sub>	1	<sup>5</sup> 7 <sub>9</sub>	<sup>2 3</sup> 4 <sub>6</sub>	<sup>2</sup> 4 <sub>7</sub>	<sup>2 3</sup> 6 <sub>9</sub>
<sup>1 2</sup>	9	7	3	6	4	8	5	<sup>1 2</sup>
<sup>1</sup> 4 <sub>5</sub> 8	<sup>1</sup> 3 <sub>5</sub> 6	<sup>1</sup> 3 <sub>4</sub> 5	<sup>5</sup> 7 <sub>8</sub>	<sup>5</sup> 7 <sub>8</sub> 9	2	<sup>1</sup> 3 <sub>4</sub> 6	<sup>1</sup> 4 <sub>6</sub>	<sup>1</sup> 3 <sub>4</sub> 6
<sup>1 2</sup> 4 <sub>5</sub> 7	<sup>1 2 3</sup> 5 <sub>6</sub> 8	<sup>1</sup> 3 <sub>4</sub> 5	<sup>1</sup> 4 <sub>5</sub> 7	<sup>4</sup> 5 <sub>7</sub>	<sup>5</sup> 7	<sup>1 2 3</sup> 4 <sub>6</sub>	<sup>1 2</sup> 4 <sub>6</sub>	<sup>1 2 3</sup> 4 <sub>6</sub>
<sup>1 2</sup> 4 <sub>5</sub> 7	<sup>1 2 3</sup> 5 <sub>7</sub>	<sup>1</sup> 3 <sub>4</sub> 5	6	<sup>4</sup> 5 <sub>7</sub>	8	<sup>1 2 3</sup> 4	9	<sup>1 2 3</sup> 4
<sup>1</sup> 4 <sub>6</sub>	<sup>1</sup> 6 <sub>4</sub>	<sup>1</sup> 4 <sub>6</sub>	9	2	3	7	8	5

Abbildung 2.6: Pointing Pair / Triple

In **Abbildung 3.4** betrachten wir Block 8. Hier ist das Vorkommen der Zahl 1 in den Kandidatenlisten auf Zeile 7 beschränkt. Wie oben beschrieben können nun alle weiteren vorkommen in der selben Zeile, die nicht in Block 8 liegen aus den Kandidatenlisten entfernt werden. Im vorliegenden Beispiel führt das allerdings nicht dazu, dass eine neue Zahl in das Sudoku eingetragen wird. Dennoch ist das Sudoku nun genauer bestimmt, da weniger Möglichkeiten übrig sind.

## 2.4.6 Box-Line Reduction

Die Technik *Box-Line-Reduction* ist verwandt mit der Technik *Pointing Pair / Triple*. Hier wird das Sudoku zeilen- und spaltenweise betrachtet. Ist das Vorkommen einer Zahl in den Kandidatenlisten auf einen Block beschränkt, dann kann jedes weitere Vorkommen der Zahl aus den Kandidatenlisten der Zellen des selben Blocks gestrichen werden, die nicht in der Zeile oder Spalte liegen. Die Begründung dafür ist ähnlich der Begründung bei *Pointing Pair / Triple*. Da die Zahlen 1 bis 9 jeweils genau einmal in der Zeile oder Spalte vorkommen müssen und dieses Vorkommen auf einen Block beschränkt ist, ist klar, dass die Zahl letztendlich in diesem Block vorkommt und zwar in der gefundenen Zeile oder Spalte. Die Zahl kann aber nicht zweimal in dem Block vorkommen, daher kann sie aus den Kandidatenlisten des Blocks gelöscht werden, deren Zellen sich nicht in der Reihe oder Spalte befinden.

7	6	2	4 <sup>3</sup> 5	4 <sup>3</sup> 5 <sup>9</sup>	8	5 <sup>9</sup> 4 <sup>3</sup>		1
9	8	4 <sup>3</sup>	1 <sup>2</sup> 3 <sup>4</sup>	1 <sup>2</sup> 4 <sup>5</sup> 7	4 <sup>5</sup> 7	2 <sup>5</sup> 4 <sup>3</sup>		6
1	5	4 <sup>3</sup>	2 <sup>3</sup> 4 <sup>6</sup> 9	2 <sup>6</sup> 4 <sup>9</sup>	4 <sup>6</sup> 6	2 <sup>9</sup>	8	7
4	7	8	2 <sup>5</sup>	2 <sup>5</sup>	3	1	6	9
5	2	6	1 <sup>4</sup>	1 <sup>4</sup>	9	8	7	3
3	1	9	8	7 <sup>6</sup> 7 <sup>6</sup>		4	2	5
8	3	5	4 <sup>7</sup>	4 <sup>7</sup>	1	6	9	2
2	9	7	6	8	5	3	1	4
6	4	1	9	3	2	7	5	8

Abbildung 2.7: Box-Line Reduction

Wir betrachten Spalte 6 in **Abbildung 3.5**. Hier sieht man, dass das Vorkommen der Zahl 4 in dieser Spalte auf Block 2, den oberen Block, beschränkt ist. Anhand dieser Spalte sieht man also, dass die Zahl 4 entweder in z2s6 oder in z3s6 steht, also in jedem Fall in Block 2. Daher kann die Zahl 4 aus den Kandidatenlisten der anderen Zellen in Block 2 gestrichen werden.

### 2.4.7 Naked Subset

Die Technik *Naked Subset* ist ein Überbegriff für die Techniken *Naked Pair*, *Naked Triple* und *Naked Quadruple*. Alle Techniken arbeiten nach dem selben Prinzip, der Unterschied liegt in der Anzahl der verwendeten Kandidatenlisten. Bei *NakedSubsets* sucht man nach Paaren, Tripeln oder Quadrupeln von Zellen in Figuren, nach Kandidatenlisten einer bestimmten Eigenschaft. Die Vereinigung der Listen muss eine bestimmte Anzahl Elemente enthalten. Bei Paaren sind das zwei, bei Tripeln drei und bei Quadrupeln vier Einträge in den Kandidatenlisten.

Findet man zum Beispiel ein Paar, das nur noch die selben beiden Zahlen enthalten kann dann ist klar, dass keine der Zahlen anderswo in der Figur stehen kann, da sonst für eine der Zellen keine Zahl mehr übrig bleibt. Daher können die beiden Zahlen dann aus den Kandidatenlisten aller anderen Zahlen aus der Figur entfernt werden. Die Begründung für Tripel und Quadrupel ist analog.

Es ist nicht nötig nach mehr als Quadrupeln zu suchen, da für jedes Naked Quintupel ein Hidden Quadrupel existiert.

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

Abbildung 2.8: Naked Subset - Naked Triple

In **Abbildung 3.6** findet man das *Naked Subset*, bei dem es sich um ein *Naked Triple* handelt, in Spalte 2. Hier hat die Vereinigung der Kandidatenlisten der Zellen z2s2, z4s2 und z5s2 genau drei Einträge: 3, 6 und 9. Es gibt offensichtlich keine andere Möglichkeit, als die drei Zahlen auf diese Felder zu verteilen, Demnach können sie in der Reihe sonst nicht vorkommen und können aus den Kandidatenlisten der anderen Zellen entfernt werden.

## 2.4.8 Hidden Subset

Analog zu den *Naked Subset*-Techniken ist auch *Hidden Subset* ein Sammelbegriff. Er beinhaltet die Techniken *Hidden Pair*, *Hidden Triple* und *Hidden Quadruple*. Auch hier ändert sich nur die Anzahl der betrachteten Kandidatenlisten. Hier soll exemplarisch die Technik *Hidden Tuple* erklärt werden, im folgenden Beispiel wird dann die Technik *Hidden Quadruple* angewendet.

Wenn man in einer Figur zwei Zahlen findet, die ausschließlich in den zwei gleichen Zellen vorkommen können, dann müssen diese beiden Zahlen in die beiden Zellen gesetzt werden. Daher kann man alle anderen Kandidaten in den Zellen von der Kandidatenliste streichen.

8	1	6	5	7	3	2	9	4
3	9	2	<sup>1</sup>	<sup>1</sup> <sub>6 4 6 4</sub>	<sup>1</sup> <sub>8</sub>	<sup>1</sup> <sub>7 8</sub>	<sup>1</sup> <sub>5</sub>	<sup>5</sup> <sub>7</sub>
4	5	7	2	<sup>1</sup> <sub>8</sub>	9	<sup>3</sup> <sub>8</sub>	<sup>1</sup> <sub>3</sub>	6
9	4	1	<sup>3</sup> <sub>7</sub>	<sup>2 3</sup> <sub>7</sub>	2	5	6	8
7	8	5	4	9	6	1	2	3
6	2	3	8	<sup>1</sup> <sub>5</sub>	<sup>1</sup> <sub>5</sub>	<sup>7</sup> <sub>9</sub>	4	<sup>7</sup> <sub>9</sub>
2	7	9	<sup>3</sup> <sub>6</sub>	<sup>3</sup> <sub>4 5 6 8</sub>	<sup>3</sup> <sub>4 5 8</sub>	<sup>4</sup> <sub>6</sub>	<sup>5</sup> <sub>3</sub>	1
1	3	8	<sup>6</sup> <sub>9</sub>	<sup>2</sup> <sub>4 5 6 8</sub>	<sup>2</sup> <sub>4 5</sub>	<sup>4</sup> <sub>6</sub>	7	<sup>5</sup> <sub>9</sub>
5	6	4	<sup>1</sup> <sub>7</sub>	<sup>3</sup> <sub>9</sub>	<sup>1</sup> <sub>7</sub>	<sup>3</sup> <sub>9</sub>	8	2

Abbildung 2.9: Hidden Subset - Hidden Quadruple

In **Abbildung 3.7** betrachten wir den Block 8 und in diesem Block die Zellen z6s5, z6s6, z7s5 und z7s6. Nur in diesen Zellen können die Zahlen 4, 5, 6 und 8 vorkommen. Da wir diese vier Zahlen nun auf die vier Zellen verteilen müssen gibt es dort keinen Platz mehr für andere Zahlen. Diese können also aus den Kandidatenlisten entfernt werden.

---

## 2.4.9 Fish

---

Die *Fish* Methoden sind ein Sammelbegriff für eine ganze Gruppe von Methoden, die alle nach dem gleichen Prinzip arbeiten. Wie echte Fische hat dieses Prinzip eine sehr große Anzahl Unterarten hervorgebracht. Kleine Fische wie zum Beispiel X-Wing sind von geübten Sudoku Spielern noch zu finden, wenn die Fische allerdings größer werden, dann sind sie nur noch mit sehr hohem Aufwand manuell zu finden und daher eher zur Verarbeitung mit dem Computer gedacht.

Auf einer Internetseite, die sich unter anderem mit den Lösungsmethoden für Sudokus befasst, findet sich die folgende Erklärung zur Funktionsweise von Fischen.

*[...] Man suche eine bestimmte Anzahl von Häusern, die sich nicht überschneiden. Diese Häuser werden als Base-Sets (Basismengen) bezeichnet (Set wird hier synonym für Haus verwendet), die in diesen Häusern enthaltenen Kandidaten sind die Basiskandidaten. Nicht überschneiden bedeutet hier, dass kein Basiskandidat in mehr als einem Haus enthalten sein darf, die Häuser selbst dürfen sich schon überlappen. Nun suche man eine gleiche Anzahl an sich nicht überschneidenden Häusern, die alle Basiskandidaten abdecken (engl.: cover). Diese neuen Häuser sind die Cover-Sets, sie enthalten die Coverkandidaten. Wenn eine solche Kombination existiert, können alle Coverkandidaten gelöscht werden, die nicht gleichzeitig Basiskandidaten sind.*<sup>1 2</sup>

---

<sup>1</sup> Quelle: [http://hodoku.sourceforge.net/de/tech\\_fishg.php](http://hodoku.sourceforge.net/de/tech_fishg.php)

<sup>2</sup> Häuser stehen hier für Figuren

## Basic Fish

Die Techniken *X-Wing*, *Swordfish* und *Jellyfish* sind die einfachsten Unterarten der Fische. Sie funktionieren nach dem selben Prinzip, nur dass eine unterschiedliche Anzahl an Figuren betrachtet wird, ähnlich zu *Naked Subset* und *Hidden Subset*. Hier wird stellvertretend die Technik *X-Wing* erklärt und am Beispiel gezeigt.

Dazu sucht man zwei Spalten oder Zeilen, die ausschließlich in den selben zwei Zellen einen bestimmten Kandidaten, die Fischziffer, beinhalten. Nun kann man aus dem jeweils anderen Paar von Figuren (Spalte oder Zeile), deren Position durch die zwei gefundenen Zellen festgelegt wird, alle Fischziffern löschen, die nicht gleichzeitig in einer der zuerst ausgesuchten Figuren liegen.

Da die Fischziffer in den beiden zuerst ausgesuchten Figuren nur an jeweils zwei Stellen liegen kann und diese sich paarweise gegenseitig ausschließen ist klar, dass jedes Vorkommen der Fischziffer in den zuletzt ausgesuchten Figuren in der Überschneidung mit den ersten Figuren liegen muss.

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

Abbildung 2.10: Basic Fish - X-Wing

Ein Beispiel für die Technik *X-Wing* findet sich in **Abbildung 3.8**. Hier wurden als erste Figuren die Zeilen 2 und 5 gewählt. Diese enthalten die Fischziffer 5 nur an den Stellen 5 und 8. Wichtig ist, dass es in den beiden Zeilen die gleichen Positionen sind. Da die Ziffer 5 und Zeile 2 nur an den Positionen 5 und 8 stehen kann werden beiden Fälle nun getrennt betrachtet. Steht in z2s5 die Ziffer 5, dann muss sie auch in z5s8 stehen, da sonst die Zeile 5 die Ziffer 5 nicht enthalten würde. Umgekehrt gilt: Steht in z2s8 die Ziffer 5, dann muss sie auch in z5s5 stehen. In jedem der beiden möglichen Fälle gilt, dass sowohl in Spalte 5 als auch in Spalte 8 die Ziffer 5 vorkommt und zwar in den Zeilen 2 und 5. Daher kann aus allen anderen Zellen der Spalten die Fischziffer 5 gelöscht werden, falls sie in den Kandidatenlisten vorhanden ist.



## 2.4.10 Single Digit Patterns

*Single Digit Patterns* ist ein Oberbegriff für Techniken, denen allen gemeinsam ist, dass sie nur eine Ziffer betrachten. Die Techniken *textitSkyscarper* und *2-String-Kite* sind Unterarten der Technik *Turbot Fish*, die aber für den Menschen einfacher zu finden sind. Da der Schwierigkeitsgrad des Sudokus für Menschen bewertet werden soll, werden sie hier mit aufgelistet und auch im Programm verwendet.

### Skyscarper

Die Technik *Skyscarper* bedeutet übersetzt Wolkenkratzer und leitet sich von der Anordnung der betrachteten Ziffern ab. Gesucht werden zwei Zeilen oder Spalten, in deren Kandidatenlisten die Ziffer jeweils noch genau zwei mal auftaucht. Wenn nun zwei der Kandidaten in der selben anderen Figur (Spalte oder Zeile) sind, dann hat man einen Wolkenkratzer gefunden. Die beiden Zahlen, die in der selben anderen Figur sind, bilden das Fundament des Wolkenkratzers, sie schließen sich gegenseitig aus. Das bedeutet wiederum, dass eine der beiden anderen gefundenen Ziffern dort stehen muss. Daher können alle Kandidaten, die von beiden Ziffern ausgeschlossen werden, aus den Kandidatenlisten gelöscht werden.

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

Abbildung 2.11: Skyscarper

In **Abbildung 3.9** betrachten wir die Spalten 6 und 9. Hier sind die Bedingungen für den Wolkenkratzer erfüllt, da in jeder Spalte die Ziffer 1 jeweils genau zwei mal vorkommt und sie in jeder Spalte an Position 5 auftaucht. Für das Feld z3s9 gibt es nun zwei Möglichkeiten. Entweder die Ziffer 1 steht in diesem Feld oder nicht. Diese beiden Möglichkeiten werden nun separat betrachtet. Wenn die Ziffer 1 in Feld z3s9 steht, dann schließt das bereits alle rot markierten Zahlen aus. Für den Fall, dass die Ziffer 1 nicht in z3s9 steht, muss sie in z5s9 stehen, das geht aus der Bedingung des Wolkenkratzers hervor. Da sich nun z5s9 und z5s6 laut Bedingung in der selben Zeile befinden, kann die Ziffer 1 nicht in z5s6 vorkommen. Deshalb muss sie in z1s6 stehen, wo sie alle rot markierten Felder ausschließt.

## 2-String Kite

Die Technik *2-String-Kite* wird auch Paierdrache genannt, was sich, wie beim Wolkenkratzer, aus der Anordnung der Zahlen ableitet. Auch hier wird nur eine einzige Ziffer betrachtet. Gesucht werden eine Zeile und eine Spalte, die nur noch zwei Kandidaten der betrachteten Ziffer enthalten, so dass ein Kandidat der Spalte und ein Kandidat der Zeile im selben Block liegen. Die Zeile und die Spalte nennt man die *Schnüre* des Drachens. Die Enden der Schnüre liegen im gleichen Block, die Position des Anfangs ist relevant für das Löschen des Kandidaten. Gelöscht werden kann nämlich der Kandidat in der Zelle, die von beiden Anfängen der Schnüre ausgeschlossen wird. Das kommt zustande, da die betrachtete Ziffer in jedem Fall am Anfang einer der beiden Schnüre stehen muss.

<sup>3</sup> 7 9	8	1	<sup>4 5</sup> 7 9	2	<sup>4 5</sup> 7 9	6	<sup>3</sup> 7	<sup>3</sup> 7 5
<sup>3</sup> 7	4	2	<sup>5</sup> 7	6	<sup>1 3</sup> 5	<sup>1</sup> 5	8	9
<sup>3</sup> 7 9	5	6	8	<sup>1</sup> 7 9	<sup>1 3</sup> 7 9	2	4	<sup>1 3</sup> 7
6	9	3	1	4	2	7	5	8
4	2	8	3	5	7	9	1	6
1	7	5	6	8	9	3	2	4
5	1	<sup>4</sup> 7	<sup>4</sup> 7	3	6	8	9	2
2	3	<sup>7 9</sup> 7 9	<sup>5</sup> 7 9	<sup>1</sup> 7 9	8	4	6	<sup>1</sup> 7 5
8	6	<sup>4</sup> 7 9	2	<sup>1</sup> 7 9	<sup>1 5</sup> 4 5	<sup>1</sup> 5	<sup>3</sup> 7	<sup>1 3</sup> 7 5

Abbildung 2.12: 2-String-Kite

Abbildung 3.10 zeigt einen *2-String-Kite*. Betrachtet wird die Ziffer 5, Zeile 8 und Spalte 7 fungieren als Schnüre des Drachens. In z2s7 betrachten wir zwei Fälle: Entweder die Ziffer 5 steht dort oder sie steht dort nicht. Für den ersten Fall gilt, dass dann die rot markierte Ziffer in z2s4 ausgeschlossen ist. Der zweite Fall ist etwas komplizierter. Wenn die Ziffer 5 nicht in z2s7 steht, dann muss sie an der einzig anderen möglichen Position der Spalte stehen, nämlich in z9s7. Daher kann sie nicht in z8s9 stehen, da diese Felder im selben Block liegen. Da in Zeile 8 auch nur noch zwei Kandidaten für die Ziffer 5 übrig waren, muss sie in z8s4 stehen, wo sie z2s4 ausschließt. In jedem der Fälle kann also die Ziffer 5 nicht in z2s4 stehen und daher kann sie dort gelöscht werden.

## Turbot Fish

Wie auch bei den vorherigen *Single Digit Patterns* wird auch beim *Turbot Fish* nur eine Ziffer betrachtet. Gesucht wird eine Kette, die vier Ziffern lang ist, so dass Anfang und Ende eines Kettenglieds in einer gemeinsamen Figur liegen. Wichtig ist dabei, dass im ersten und dritten gemeinsamen Figur die beiden betrachteten Kandidaten die einzigen verbliebenen sind. Da die Kette vier Glieder lang ist, muss entweder der Anfang oder das Ende der Kette wahr sein, daher können die Kandidaten gelöscht werden, die von beiden Feldern ausgeschlossen werden.

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

Abbildung 2.13: Turbot Fish

In der obigen **Abbildung 3.11** beginnt die Kette der Ziffer 8 im Feld z2s6. In der selben Spalte befindet sich die zweite Ziffer in z7s6, sie ist dort der einzige weitere Kandidat der Ziffer 8, was Voraussetzung für den Turbot Fish ist, da es sich hier um das erste Glied handelt. Die nächste Ziffer liegt in der gleichen Zeile, z7s9. Die letzte Ziffer befindet sich im selben Block wie auch ihr Vorgänger, in z9s8, und auch sie ist hier der einzige weitere Kandidat für diese Ziffer. Wir betrachten zwei Fälle: die Ziffer 8 steht in z2s6 oder sie steht dort nicht. Im ersten Fall kann der rot markierte Kandidat in z2s8 gelöscht werden, da er direkt ausgeschlossen wird. Wenn die 8 dort nicht steht, dann muss sie in z7s6 stehen, da das der einzige andere Kandidat in der Zeile ist. Darum kann die 8 dann nicht in z7s9 stehen. Daraus folgt, dass sie in z9s8 stehen muss, was dann im zweiten Fall die Ziffer 8 in z2s8 ausschließt, womit diese dort in jedem Fall nicht stehen kann.

## Empty Rectangle

Für die Technik *Empty Rectangle* gilt das selbe wie für alle *Single Digit Patterns*, es wird nur eine Ziffer betrachtet und der Name leitet sich aus der Form der Anordnung der Ziffern ab. Um ein *Empty Rectangle* zu finden, wird ein Block gesucht, in dem ein Kandidat ausschließlich noch in einer Zeile und in einer Spalte vorkommen kann. Dann bilden die verbliebenen Kandidaten eine Ecke eines *Empty Rectangles*.

5	9	8	6	4	3	<sup>1</sup>	<sup>1</sup>	2
<sub>1 2</sub>	<sub>1 2</sub>	3	7	5	9	6	4	8
6	7	4	1	2	8	5	9	3
4	5	7	2	<sup>1</sup>	<sup>1</sup>	8	3	<sup>1</sup>
9	<sup>1</sup>	6	3	<sup>1</sup>	7	4	2	5
<sup>1</sup>	3	2	4	<sup>1</sup>	5	<sup>1</sup>	6	<sup>1</sup>
<sup>2</sup>	<sup>2</sup>	5	9	<sup>1</sup>	4	3	8	<sup>1</sup>
3	4	1	8	7	2	9	5	6
<sub>7 8</sub>	<sub>6</sub>	9	5	3	<sup>1</sup>	2	<sup>1</sup>	4

Abbildung 2.14: Empty Rectangle

In **Abbildung 3.12** sehen wir ein *Empty Rectangle*, das von Block 5 ausgeht. Die verbleibenden Kandidaten der Ziffer 1 können hier nur noch in Zeile 4 und Spalte 5 stehen. Wenn wir Zeile 7 betrachten, dann sehen wir, dass die Ziffer 1 hier nur noch an zwei Positionen stehen kann, nämlich an der fünften und an der neunten. Wir betrachten nun diese beiden Fälle getrennt. Wenn die Ziffer 1 in z7s9 steht, dann ist die Ziffer 1 in z4s9 direkt ausgeschlossen. Wenn sie in z7s5 steht, dann kann sie in Block 5 nur noch an einer Position stehen, nämlich z4s6. Auch in diesem Fall ist die Ziffer 1 dann im Feld z4s9 ausgeschlossen. Daher kann sie dort als Kandidat gelöscht werden.

## 2.4.11 Wings

### XY-Wing

Die Technik *XY-Wing* wird manchmal auch nur *Y-Wing* genannt, da sie aussieht wie ein *X-Wing* (siehe Kapitel 2.4.9) nur mit drei Ecken. Zuerst sucht man eine Zelle, in der nur noch 2 Kandidaten verblieben sind. Diese Kandidaten nennt man dann x und y. Daher kommt der im Allgemeinen bekanntere Name der Strategie. Im nächsten Schritt sucht man nun eine Zelle, die in einer gemeinsamen Figur mit der ersten Zelle liegt und auch nur noch 2 Kandidaten der Form hat, dass einer der Kandidaten dem x aus der ersten Zelle entspricht und der zweite Kandidat ungleich dem y ist. Dieser wird nun z genannt. Anschließend sucht man eine dritte Zelle mit nur noch zwei verbliebenen Kandidaten, die ebenfalls in einer gemeinsamen Figur mit der ersten Zelle liegt, aber nicht in der selben wie die zweite gefundene Zelle. Wenn diese Zelle nun die Kandidaten y und z enthält, dann hat man einen *XY-Wing* gefunden. Gelöscht werden kann nun der Kandidat z aus der Zelle, die von der zweiten und dritten Zelle ausgeschlossen wird. Das funktioniert, da in der ersten Zelle entweder x oder y steht. Wenn in der ersten Zelle x steht, dann steht in der zweiten Zelle z, da dort nur x und z stehen kann, x aber durch die erste Zelle ausgeschlossen wird. Wenn in der ersten Zelle aber y steht, dann muss in der dritten Zelle z stehen, da dort nur y und z stehen können und y ausgeschlossen wird. Daher steht in einer der beiden Zellen z und alle Kandidaten von z, die durch beide Felder ausgeschlossen werden, können gelöscht werden.

8	<sup>1</sup> <sub>4 5</sub>	<sup>5</sup> <sub>7</sub>	3	6	<sup>2</sup> <sub>5</sub>	9	<sup>1</sup> <sub>4 7</sub>	<sup>1 2</sup>
<sup>2</sup> <sub>7</sub>	<sup>4 5</sup>	9	<sup>4 7</sup>	1	<sup>2</sup> <sub>5</sub>	8	6	3
<sup>1 2</sup> <sub>7</sub>	6	3	<sup>4 7</sup>	8	9	<sup>2</sup> <sub>4</sub>	<sup>1</sup> <sub>4 7</sub>	5
9	2	4	6	7	3	1	5	8
3	8	6	9	5	1	7	2	4
5	7	1	8	2	4	3	9	6
4	3	2	1	9	6	5	8	7
6	9	8	5	3	7	<sup>2</sup> <sub>4</sub>	<sup>1</sup> <sub>4</sub>	<sup>1 2</sup>
<sup>1</sup> <sub>7</sub>	<sup>1</sup> <sub>5</sub>	<sup>5</sup> <sub>7</sub>	2	4	8	6	3	9

Abbildung 2.15: XY-Wing

In der obigen **Abbildung 3.13** stehen in z1s3 die Kandidaten 5 für x und 7 für x. In Zelle z1s6 stehen 5 für x und 2 für z. in z2s1 stehen 7 für y und 2 für z. Wenn in z1s3 die Ziffer 5 steht, dann muss in z1s6 die Ziffer 2 stehen und den Kandidaten 2 in z2s6 ausschließen. Wenn in z1s3 die Ziffer 7 steht, dann steht in z2s1 die Ziffer 2 und schließt ebenfalls die Ziffer 2 in z2s6 aus. Somit kann diese dort in keinem der beiden Fälle stehen und kann gelöscht werden.

## XYZ-Wing

Bei der Technik *XYZ-Wing* handelt es sich um eine erweiterte Version der Technik *XY-Wing*. Zusätzlich zu den Kandidaten x und y steht hier in der ersten Zelle noch der Kandidat z, der Rest ändert sich nicht. Statt zwei Fällen werden hier drei Fälle betrachtet. Wenn der Kandidat x in der ersten Zelle steht, dann bleibt die Argumentation wie beim *XY-Wing*. Das selbe gilt für den Fall, dass der Kandidat y in der ersten Zelle steht. Im dritten Fall steht der Kandidat z in der ersten Zelle. In diesem Fall würde er alle Kandidaten ausschließen, die von dort aus direkt ausgeschlossen werden. Daher können alle Kandidaten der Ziffer z gelöscht werden, die von allen drei Zellen ausgeschlossen werden.

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

Abbildung 2.16: XYZ-Wing

In **Abbildung 3.14** sieht man einen *XYZ-Wing*, dessen erste Zelle z3s4 ist. Hier stehen die Kandidaten 4 für x, 7 für y und 6 für z. Bei z3s7 findet man die zweite Zelle mit den Kandidaten 4 und 6, also x und z. Die dritte Zelle ist z2s4, sie enthält die Kandidaten 6 und 7 und damit y und z. Wenn die Ziffer 4 in z3s4 steht, dann muss in z3s6 die Ziffer 6 stehen und damit sind die rot markierten Kandidaten ausgeschlossen. Wenn in z3s4 die Ziffer 7 steht, dann muss in z2s4 die Ziffer 6 stehen und auch in diesem Fall sind die rot markierten Kandidaten ausgeschlossen. Wenn in z3s4 die Ziffer 6 steht, dann sind ebenfalls die rot markierten Kandidaten ausgeschlossen. Somit können sie in jedem der möglichen Fälle ausgeschlossen werden und werden daher gelöscht.

## W-Wing

Die Technik *W-Wing* ist die letzte und schwierigste Technik der Wings. Hierbei werden immer zwei Ziffern betrachtet. Zuerst sucht man eine Zelle, in der nur noch zwei Kandidaten  $x$  und  $y$  möglich sind. Nun wird eine Figur gesucht, in der der Kandidat  $x$  nur noch zwei mal vorkommen kann und eines der möglichen Vorkommen von der ersten Zelle ausgeschlossen würde. Im letzten Schritt sucht man eine Zelle, die wieder ausschließlich die Kandidaten  $x$  und  $y$  enthält und die das andere mögliche Vorkommen der Ziffer  $x$  in der zuvor gefundenen Figur ausschließen würde. Findet man eine solche Konstellation, dann handelt es sich um einen *W-Wing*. Gelöscht werden können die Kandidaten  $y$ , die von beiden Zellen gleichzeitig ausgeschlossen werden, da der Kandidat  $y$  entweder in der ersten oder in der zweiten gefundenen Zelle stehen muss.

9	2	5	1	3	4	6	8	7
8	<sup>1</sup>	<sup>1</sup>	6	5	9	4	3	2
4	3	6	7	2	8	9	5	1
6	4	<sup>2</sup>	<sup>5</sup>	1	<sup>2 3</sup>	8	<sup>9</sup>	<sup>3</sup>
1	5	<sup>2</sup>	4	8	<sup>2 3</sup>	2	6	<sup>3</sup>
3	<sup>7 9</sup>	8	<sup>5 9</sup>	6	<sup>2</sup>	<sup>2 5</sup>	1	4
5	<sup>1</sup>	<sup>1</sup>	2	7	6	3	4	8
2	6	3	8	4	1	<sup>5</sup>	<sup>9</sup>	<sup>5</sup>
7	8	4	3	9	5	1	2	6

Abbildung 2.17: W-Wing

In **Abbildung 3.15** betrachten wir zuerst z8s9. Hier finden wir die Ziffer 9 für  $x$  und 5 für  $y$ . Die gesuchte Figur ist Spalte 8, da hier der Kandidat 9 nur noch zwei mal vorkommen kann und eines der Vorkommen von z8s9 ausgeschlossen wird. Das andere Vorkommen wird von z4s4 ausgeschlossen. In dieser Zelle befinden sich ausserdem nur noch die Kandidaten  $x$  und  $y$ . Damit ist die Bedingung für den *W-Wing* erfüllt und der rot markierte Kandidat kann gelöscht werden. Wir betrachten zwei Fälle, entweder die Ziffer 5 steht in z8s9 oder nicht. Im ersten Fall würde der rot markierte Kandidat direkt ausgeschlossen werden. Wenn die Ziffer 5 nicht in z8s9 steht, dann steht dort die Ziffer 9. Damit kann die in Spalte 8 nur noch an Position 4 stehen und damit muss in z4s4 die Ziffer 5 stehen, was wiederum den rot markierten Kandidaten ausschließt. In keinem der Fälle kann der Kandidat dort stehen und wird daher gelöscht..t

## 2.4.12 Sue de Coq

Die Technik *Sue de Coq* wurde ursprünglich unter dem Namen *Two-Sector Disjoint Subsets* in einem Forum<sup>3</sup> vorgestellt. Anderen Benutzern des Forums war dieser Name zu umständlich und sie begannen die Strategie als *Sue de Coq* zu bezeichnen, dem Benutzernamen des Vorstellers.

Um die Technik anzuwenden sucht man zuerst zwei oder drei Zellen in einem Block, die in der selben Reihe oder Spalte liegen. Die Vereinigung der Kandidatenlisten muss dabei bei zwei Zellen vier Kandidaten enthalten und bei drei Zellen müssen fünf Kandidaten vorhanden sein. Nun sucht man zusätzlich eine Zelle im selben Block, die ausschließlich zwei Kandidaten der Vereinigung enthält. Eine solche Zelle sucht man auch in der Reihe oder Spalte, in der die ersten Zellen liegen. Die Schnittmenge der Kandidatenlisten der beiden zusätzlichen Zellen muss leer sein.

Die Kandidaten aus der zusätzlichen Zelle in der Reihe können nun aus allen unbeteiligten Zellen der Reihe gelöscht werden. Das gilt ebenfalls für die Kandidaten der zusätzlichen Zelle des Blocks, diese können aus allen unbeteiligten Zellen des Blocks entfernt werden.

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

Abbildung 2.18: Sue de Coq

In Abbildung 3.16 sieht man einen *Sue de Coq* mit den zwei Basiszellen z7s1 und z7s3. Diese liegen im Block 7 in der selben Zeile. Die Vereinigung der Kandidaten enthält die Ziffern 3, 4, 5 und 9. Die zusätzliche Zelle in der Zeile ist z7s7. Sie enthält die Kandidaten 4 und 5 die Teilmenge der vorherigen Vereinigung ist. Die zusätzliche Zelle im Block 7 ist z8s3. Diese enthält die Kandidaten 3 und 9, die ebenfalls Teilmenge der Vereinigung ist. der Schnitt der Kandidatenlisten der zusätzlichen Zellen ist leer. Damit sind alle Bedingungen für den *Sue de Coq* erfüllt und die rot markierten Kandidaten können gelöscht werden.

<sup>3</sup> <http://forum.enjoysudoku.com/two-sector-disjoint-subsets-t2033.html>



## 2.4.13 Coloring

Die Technik *Coloring* konzentriert sich auf einzelne Ziffern. Es werden Zellen nach einer bestimmten Vorgehensweise Farben zugeordnet. Danach wird das gefärbte Sudoku nach Ausschlüssen durchsucht. Ähnlich zu den *Fischen* 2.4.9 ist *Coloring* eine Strategie, die viele Unterarten hervor gebracht hat. In diesem Kapitel wird ausschließlich auf *Simple Coloring* eingegangen, die einfachste der Strategien, die nur zwei Farben benutzt.

Beim *Simple Coloring* wird für die betrachtete Ziffer nach Figuren gesucht, in denen die Ziffer nur noch in zwei Kandidatenlisten vorkommt. Den Zellen dieser Kandidatenlisten werden dann zwei unterschiedliche Farben zugeordnet. Dadurch, dass sich die Kandidaten in den Figuren gegenseitig ausschließen, steht die betrachtete Ziffer immer in jedem Feld der entweder einen oder anderen Farbe.

Erster Widerspruch (Color Trap): In Zellen, die nicht gefärbt sind, die von beiden Farben gesehen werden und die die betrachtete Ziffer enthalten, kann diese gelöscht werden.

Zweiter Widerspruch (Color Wrap): Wenn eine Farbe in der gleichen Figur zwei mal vorkommt, dann können alle Vorkommen der Ziffer in Zellen mit dieser Farbe gelöscht werden, da die Ziffer entweder in allen Zellen steht oder in keiner. In allen Zellen kann die Ziffer nicht stehen, da eine Ziffer nicht zwei mal in einer Figur vorkommen kann.

2	1	4	5 <sup>3</sup>	5 <sup>3</sup>	6	5 <sup>3</sup>	5 <sup>3</sup>	5 <sup>3</sup>
5 <sup>3</sup>	5 <sup>3</sup>	7	9	5 <sup>3</sup>	2	5 <sup>3</sup>	5 <sup>3</sup>	4
5 <sup>3</sup>	5 <sup>3</sup>	5 <sup>3</sup>	4	5 <sup>3</sup>	7	5 <sup>3</sup>	5 <sup>3</sup>	5 <sup>3</sup>
5 <sup>3</sup>	5 <sup>3</sup>	1	8	7	4 <sup>3</sup>	4 <sup>3</sup>	3	2
5 <sup>3</sup>	5 <sup>3</sup>	2	6	9	4 <sup>3</sup>	4 <sup>3</sup>	5 <sup>3</sup>	5 <sup>3</sup>
5 <sup>3</sup>	4	8	5 <sup>3</sup>	2	1	5 <sup>3</sup>	5 <sup>3</sup>	6
4	2	5 <sup>3</sup>	7	5 <sup>3</sup>	9	8	6	1
5 <sup>3</sup>	5 <sup>3</sup>	9	1	6	8	4 <sup>3</sup>	4 <sup>3</sup>	5 <sup>3</sup>
1	8	6	2	4	5 <sup>3</sup>	5 <sup>3</sup>	5 <sup>3</sup>	9

Abbildung 2.19: Simple Coloring - Color Trap

In **Abbildung 3.17** wird die Ziffer 3 betrachtet. Sie kann entweder in allen dunkelgelb oder allen hellgelb gefärbten Zellen stehen. Wir betrachten nun zwei Fälle. Wenn die Ziffer 3 in z8s9 steht, dann kann sind die Kandidaten in z1s9 und z3s9 ausgeschlossen. Wenn nicht, dann muss sie in allen dunkelgelb gefärbten Zellen stehen, insbesondere in z1s4 und z3s3. Diese schließen ebenfalls die rot markierten Kandidaten aus. Damit können diese in keinem Fall in z1s9 und z3s9 stehen und können gelöscht werden.

---

#### 2.4.14 Almost Locked Set

---

Ein *Locked Set* bezeichnet eine Gruppe von  $n$  ungelösten Zellen, deren Vereinigung der Kandidatenlisten nur noch  $n$  Kandidaten enthält. Das erlaubt das Entfernen aller Kandidaten aus den anderen Zellen der Figur, die in der Vereinigung der Kandidatenlisten vorkommen.

Ein *Almost Locked Set* hat  $n + 1$  Kandidaten und ist daher einzeln nicht zu gebrauchen. Man kann allerdings Informationen erhalten, indem man zwei *Almost Locked Sets* kombiniert. Dazu müssen sie mindestens einen gemeinsamen Kandidaten haben für den zusätzlich gilt, dass alle Instanzen des Kandidaten in ALS 1 alle Instanzen des Kandidaten in ALS 2 sehen können. Dieser Kandidat kann dann nur einmal in beiden ALS gesetzt werden, da sich alle Instanzen gegenseitig ausschließen. Man nennt diese Ziffer *Restricted Common Candidate* oder *RCC*.

## ALS XZ

Die Technik *ALS XZ* ist die einfachste Unterart der *Almost Lockes Sets*. Man sucht zwei *ALS* mit einem *RCC*. Dieser wird *X* genannt. Wenn beide *ALS* noch einen gemeinsamen Kandidaten *Z* besitzen, der kein *RCC* ist, dann kann *Z* aus allen Zellen gelöscht werden, die nicht zum *ALS* gehören und die alle Instanzen von *Z* in beiden *ALS* sehen. Das funktioniert, da durch *X* ein *ALS* zum *Locked Set* wird. Da in beiden *ALS* die Ziffer *Z* vorkommt wird diese auf ein *ALS* beschränkt. Das bedeutet, dass *Z* in genau einem *ALS* vorkommt und daher können alle Kandidaten gelöscht werden, die von allen Instanzen gesehen werden.

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

Abbildung 2.20: ALS XZ

In **Abbildung 4.18** sehen wir die beiden *ALS* einmal in Zeile 1 Spalte 6 und 7 mit den Kandidaten 6, 7 und 9 und das zweite in Zeile 3 Spalte 2, 8 und 9 mit den Kandidaten 6, 7, 8 und 9. Der *RCC* ist hier die Ziffer 6, da alle Instanzen in beiden *ALS* in Block 3 liegen. Ausserdem kommt in beiden *ALS* die Ziffer 7 vor, die aber kein *RCC* ist, da sich zum Beispiel z1s6 und z3s9 nicht sehen können. Nun können alle Zellen die Ziffer 7 von ihren Kandidatenlisten, die alle Instanzen der Ziffer 7 in beiden *ALS* sehen, was z3s5 und z3s6 sind.

## ALS XY Wing

Bei der Technik *ALS XY Wing* benötigt man 3 *ALS* A, B und C. A und C haben den gemeinsamen *RCC* X, B und C haben den gemeinsamen *RCC* Y. X und Y sind unterschiedlich. *ALS* A und B haben die gemeinsame Ziffer Z, die kein *RCC* ist.

Nun werden zwei Fälle betrachtet. Wenn Z nicht in A steht, dann muss X in A stehen, da bei einem *ALS* nur eine Ziffer fehlen darf. Das bedeutet, C muss Y enthalten, da Z nicht in C steht. Daraus kann man folgern, dass B den Kandidaten Z enthalten muss. Im Fall zwei nehmen wir an, dass Z nicht in B steht. Dann muss Y in B stehen, woraus folgt, dass X in B steht. Daher muss dann Z in A stehen. In jedem möglichen Fall steht Z also entweder in A oder in B. Daher können alle Kandidaten von Z ausgeschlossen werden, die von allen Instanzen in A und B gesehen werden.

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

Abbildung 2.21: ALS XY Wing

Im oben stehenden Beispiel **Abbildung 4.19** sieht man einen *ALS XY Wing*, *ALS* A ist hier Zeile 7 Spalten 1, 5 und 6 mit den Kandidaten 3, 6, 7 und 8, B ist Spalte 8 Zeilen 5, 7 und 9 mit den Kandidaten 2, 3, 8 und 9. *ALS* C befindet sich in Zeile 9 Spalten 3 und 4 mit den Kandidaten 1, 7 und 9. X ist also 7, Y ist 9 und Z ist 3. Da Z wegen der oben genannten Begründung nur in *ALS* A und B vorkommen kann, können jetzt alle Kandidaten von Z gelöscht werden, die von allen Instanzen von Z in A und B gesehen werden. Das ist die Ziffer 3 in z7s7.

## ALS Chain

*ALS Chains* sind die Verallgemeinerung von *ALS XZ* und *ALS XY Wing*. Ein *ALS XZ* ist eine *ALS Chain* der Länge 2 und ein *ALS XY Wing* ist eine *ALS Chain* der Länge 3. Eine *ALS Chain* ist eine Kette von ALS verbunden durch *RCCs*, für die gilt, dass keine zwei aufeinanderfolgenden *RCCs* gleich sein dürfen. Die *ALS* am Anfang und Ende der Kette enthalten eine gemeinsame Ziffer *Z*. Diese Ziffer wird wie gewohnt dann aus allen Kandidatenlisten gelöscht, die von allen Instanzen der Ziffer *Z* im *ALS* am Anfang und am Ende der Kette gesehen werden.

Das geht, da durch die Verknüpfung durch *RCCs* die Ziffer *Z* entweder im *ALS* am Anfang der Kette stehen muss oder in dem am Ende.

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

Abbildung 2.22: ALS Chain

In **Abbildung 4.20** sieht man eine *ALS Chain* der Länge 4. Das erste *ALS* ist in Zeile 2 Spalten 1, 2, 4 und 9, es enthält die Kandidaten 2, 3, 5, 6 und 7. Durch den *RCC* 7 ist es verbunden mit dem einzelligen *ALS* in z2s7, das die Kandidaten 3 und 7 enthält. Dieses *ALS* ist durch den *RCC* 3 verbunden mit dem *ALS* in z7s7, wiederum mit den Kandidaten 3 und 7. Die letzte Verbindung besteht dann durch den *RCC* 7 zum *ALS* in Block 8 mit z7s4 und z8s5, das die Kandidaten 2, 3 und 7 enthält. Das erste und das letzte *ALS* enthalten den gemeinsamen Kandidaten 3. Dieser kann nun als Kandidat aus allen Zellen gelöscht werden, die alle Instanzen der Ziffer 3 in beiden *ALS* sehen. Das ist der Fall in z8s4 und z9s4.

---

#### 2.4.15 Backtracking

---

*Backtracking* arbeitet nach dem *trial and error* Prinzip. Es wird eine zufällige Zahl aus der Kandidatenliste eines Feldes in das Feld eingesetzt. Danach werden die Kandidatenlisten wieder aktualisiert und Backtracking beginnt von vorne. Wenn bemerkt wird, dass durch das Einsetzen einer Zahl eine Situation entsteht, die die Sudoku Regel verletzt, dann wird der letzte Schritt zurück genommen. *Backtracking* ist also ein rekursiver Algorithmus und führt eine Tiefensuche über den Lösungsraum des Sudokus durch, der als Baum dargestellt werden kann. Sobald ein Ast des Baumes komplett durchsucht wurde, ohne eine Lösung zu finden, dann wird im Baum so lange wieder nach oben gegangen, bis eine andere Abzweigung verfügbar ist. Wenn eine Lösung gefunden wurde, dann terminiert der Algorithmus. Die Terminierung ist also immer gegeben, da jedes Sudoku eine eindeutige Lösung hat, wie in den *Regeln* 2.1 definiert. Backtracking unterscheidet sich von den anderen Lösungstechniken dadurch, dass es das Sudoku vollständig löst, unabhängig davon, wie viele Zahlen dafür eingetragen werden müssen.

---

## 3 Maschinelles Lernen zur Charakterisierung des Schwierigkeitsgrades

---

### 3.1 Klassifikation

---

Unter einer Klassifikation versteht man in der Informatik das Einteilen von Objekten in vorher festgelegte Klassen. Diese Einteilung wird von einem Algorithmus durchgeführt, der anhand von festgelegten Merkmalen jedem Objekt eine Klasse zuordnet. Einen solchen Algorithmus nennt man Klassifikator. Um die Qualität eines Klassifikators zu analysieren gibt es verschiedene Metriken.

- Accuracy - Die Anzahl der richtig zugeordneten Klassen
- Recall - Der Anteil der positiven Beispiele, die auch positiv klassifiziert wurden
- Precision - Der Anteil der positiv klassifizierten Beispiele, die auch positiv sind

Ein Klassifikator benötigt vor der Phase der Klassifikation aber zunächst einmal eine Trainingsphase, in der er anhand von Beispielen lernt. Mit dem erlernten Wissen wird anschließend die Einteilung in die Klassen vorgenommen.

Es gibt viele verschiedene Ansätze für Klassifikatoren, von denen die wichtigsten in einem open source Framework implementiert sind. Dieses Framework heisst Weka<sup>1</sup> und wurde im praktischen Teil dieser Bachelorthesis verwendet.

Weka arbeitet unter anderem mit dem .arff<sup>2</sup> Format. In einer .arff Datei befindet sich neben den Metadaten hauptsächlich eine Sammlung von Featurevektoren. Jeder Featurevektor beschreibt ein zu klassifizierendes Objekt. Ein Eintrag in einem Featurevektor beschreibt eine Eigenschaft des beschriebenen Objekts. Das könnte bei Fahrzeugen zum Beispiel die Anzahl der Reifen sein. Bezogen auf Sudokus bedeutet das, dass jedes Sudoku durch einen Featurevektor beschrieben wird. Jede zur Klassifikation verwendete Eigenschaft eines Sudokus ist dann ein Wert im entsprechenden Featurevektor. Die Schwierigkeitsgrade der Sudokus sind die vorgegebenen Klassen.

Jeder Klassifikator in Weka hat als Eingabe eine Liste von Featurevektoren. Möchte man also das Zuordnen von Sudokus zu Schwierigkeitsgraden mit Weka realisieren, dann muss eine Methode entwickelt werden, die aus einem gegebenen Sudoku einen Featurevektor extrahiert.

Bei der Klassifikation wird ein Verfahren angewendet, das als *cross validation* bekannt ist und auch von Weka zur Verfügung gestellt wird. Dabei werden die Daten in eine vorgegebene Anzahl gleich großer Folds eingeteilt. Ein Fold ist eine Sammlung von Featurevektoren. Wenn die Einteilung in k Folds erfolgt ist, dann wird der Klassifikator k mal ausgewertet, einmal mit jedem Fold.

Die Qualität der resultierenden Klassifikation ist sehr stark von der Wahl der Einträge des Featurevektors abhängig. Daher lag das Hauptaugenmerk dieser Bachelorarbeit auf dem Herausarbeiten der passenden Einträge für den Featurevektor, der das Sudoku beschreibt. Später wird aber auch auf die Wahl des Klassifikators und die Optimierung der Parameter eingegangen.

---

### 3.2 Merkmalsextrahierung

---

Um Sudokus mit Hilfe von Weka nach ihrem Schwierigkeitsgrad zu klassifizieren, ist es nötig, Featurevektoren aus den Sudokus zu extrahieren. Ein Featurevektor repräsentiert ein Sudoku, ein Eintrag des Featurevektors steht für eine Eigenschaft des Sudokus. Einen Eintrag des Featurevektors nennt man ein Feature.

---

<sup>1</sup> <http://www.cs.waikato.ac.nz/ml/weka/index.html>

<sup>2</sup> <http://weka.wikispaces.com/ARFF>

---

Also ist die Frage: Welche Features hat ein Sudoku? Genauer gesagt werden Features gesucht, aus denen man Rückschlüsse auf den Schwierigkeitsgrad eines Sudokus ziehen kann.

Man kann ein solches Feature bereits finden, ohne überhaupt einen einzigen Lösungsschritt durchgeführt zu haben. Es handelt sich um die Anzahl der vorgegebenen Ziffern. Je mehr Ziffern vorgegeben sind, desto weniger muss der Spieler selbst finden und umso einfacher sollte das Sudoku für ihn werden.

---

### 3.3 Aufbau des Featurevectors

---

Wenn von einer Zahl kein Kandidat vorgegeben ist und von einer anderen Zahl bereits viele Kandidaten von Anfang an ausgefüllt sind, dann wird das Sudoku generell als schwerer empfunden, als Sudokus mit der gleichen Gesamtanzahl an vorgegebenen Kandidaten, die aber gleich verteilt sind. Daher ist es sinnvoll, die Anzahl der vorgegebenen Kandidaten für jede Zahl einzeln zu speichern. Die neun entstehenden Features werden als erstes in den Featurevector eingetragen.

Als nächstes werden Kandidatenlisten, wie in 2.4.1 angelegt. Diesen Listen kann man entnehmen, an wie vielen Stellen eine bestimmte Ziffer noch stehen kann. Das ergibt, für jede Ziffer einzeln, wieder neun Features, die in den Featurevector eingetragen werden.

Die Lösungsmethoden können generell in zwei Kategorien aufgeteilt werden. Die erste und einfachere Kategorie füllt Ziffern im Sudoku aus. Die zweite Kategorie Lösungsmethoden schließt Ziffern für bestimmte Felder aus, das bedeutet, dass sie aus den Kandidatenlisten gelöscht werden. Wenn eine Lösungsmethode auf das Sudoku angewendet wurde, dann wird das im Featurevector eingetragen. Hier existieren für jede Lösungsmethode neun Einträge - ein Eintrag für jede Ziffer. Jedes mal, wenn eine Lösungsmethode eine Ziffer ausfüllt, dann wird der entsprechende Eintrag um eins erhöht. Immer, wenn eine Lösungsmethode eine Ziffer aus einer Kandidatenliste entfernt, dann wird auch hier der zugehörige Eintrag im Featurevector um eins erhöht. Um das zu verdeutlichen sehen wir uns folgendes Beispiel an. Angenommen die Lösungsmethode Skyscarper 2.4.10 wird auf ein Sudoku angewendet und entfernt von den Kandidatenlisten von zwei Zellen die Ziffer 4. Dann wird im Featurevector der Abschnitt mit den neun Einträgen für die Lösungsmethode Skyscarper gesucht. Aus den neun Einträgen wird der Eintrag für die Ziffer 4 gesucht und sein Wert um zwei erhöht.

Nun stellt sich die Frage, welche der Lösungsmethoden zuerst angewendet wird. Um ein Sudoku mit den in dieser Arbeit beschriebenen Lösungsmethoden zu lösen, gibt es im Allgemeinen mehrere Wege. Diese entstehen durch eine unterschiedliche Anwendungsreihenfolge der Lösungsmethoden auf das Sudoku. Die Anwendungsreihenfolge ist aber entscheidend für den Aufbau des Featurevectors. Wenn zum Beispiel ein Sudoku nur mit den Methoden *Full House* 2.4.2 und *Naked Single* 2.4.3 gelöst werden kann, dann ist das für den Spieler sehr einfach. Allerdings könnte im Sudoku dennoch eine Stelle vorkommen, an der man *Coloring* 2.4.13 verwenden kann. Wenn bei der Klassifikation im Featurevector steht, dass die Methode *Coloring* verwendet wurde, die großen Aufwand erfordert und die meist nur von Computern angewendet wird, dann erscheint das Sudoku schwerer zu sein, als es tatsächlich war. Das würde das Ergebniss der Klassifikation verfälschen.

Um dieses Problem zu umgehen, wird eine Methode gesucht, um den einfachsten Lösungsweg zu finden. Der einfachste Lösungsweg ist die Folge von Lösungsschritten, die die einfachsten Lösungsschritte enthält, die im jeweiligen Zustand des Sudokus anwendbar waren. Dazu ist es nötig, den Lösungsmethoden einen Schwierigkeitsgrad zuzuweisen.

Die in dieser Arbeit vorgestellten Lösungsmethoden sind bereits von leicht nach schwer sortiert, die zuerst vorgestellten sind die Einfachsten, die zuletzt vorgestellten sind die Schwersten. Bei der Recherche nach dem Schwierigkeitsgrad der Lösungsmethoden bin ich auf verschiedene Quellen gestoßen, die sich alle mit meiner Reihenfolge übereinstimmen, wenn auch teilweise weniger oder mehr Lösungsmethoden vorgestellt wurden.

Nachdem also der Schwierigkeitsgrad der einzelnen Lösungsmethoden bekannt ist, muss jetzt zu jedem



---

Sudoku der einfachste Lösungsweg gefunden werden. Dazu wird der folgende Algorithmus verwendet.

```
sudoku: the current sudoku;
solvingMethod[]: Array of Solving methods sorted by difficulty;
int solvingMethodCounter = 0;
FeatureVector fv = new FeatureVector();
while sudoku not solved do
    apply solvingMethod[solvingMethodCounter] to sudoku;
    if sudoku changed then
        note changes in fv;
        solvingMethodCounter = 0;
    else
        solvingMethodCounter++;
    end
end
```

**Algorithm 1:** Build Featurevector

Der Algorithmus versucht also, immer die einfachste Lösungsmethode auf ein Sudoku anzuwenden so lange diese anwendbar ist. Wenn die einfachste Lösungsmethode nicht anwendbar ist, dann wird die nächst schwerere Methode versucht, so lange bis eine Methode anwendbar war und das Sudoku verändert hat. Dann fällt der Algorithmus zurück auf die leichteste Methode, da diese durch die Veränderung im Sudoku anwendbar geworden sein könnte. So wird sicher gestellt, dass immer zuerst die leichtest mögliche Methode angewendet wird und somit wird der leichteste Lösungsweg gewählt.

---

### 3.4 Entkopplung von konkreten Zahlen

---

Aus einem Sudoku kann leicht ein neues Sudoku erzeugt werden, indem man Ziffern tauscht. Das bedeutet zum Beispiel, dass man alle Vorkommen der Ziffer 7 mit der Ziffer 8 ersetzt und umgekehrt. Die Lösung des neuen Sudokus unterscheidet sich von der Lösung des alten Sudokus nur darin, dass dort ebenfalls die Ziffer 8 mit der Ziffer 7 vertauscht ist. An der Schwierigkeit des Sudokus ändert sich nichts. Das spiegelt sich allerdings nicht im Featurevector wieder, da dort die Einträge für die Ziffer 7 und die Ziffer 8 in jeder Lösungsmethode vertauscht sind. Bei einem Sudoku exakt gleicher Schwierigkeit ergibt sich also ein anderer Featurevector.

Das kann verhindert werden, indem die Einträge im Featurevector von den konkreten Zahlen entkoppelt werden. Dazu werden nach der vollständigen Ermittlung des Featurevectors die neun Einträge für jede Lösungsmethode, für die am Anfang vorgegebenen Ziffern und für die am Anfang möglichen Positionen nach ihrer Häufigkeit absteigend geordnet. Das wird am folgenden Beispiel deutlich.

Hier betrachten wir die Lösungsmethode *Coloring* 2.4.13. Vor der Sortierung hat der Vector die diese Darstellung

$(1, 0, 4, 15, 3, 0, 9, 2, 0)^T$

Das bedeutet zum Beispiel, die Ziffer 3 wurde vier mal mit der Methode *Coloring* ausgeschlossen. Würde man, wie im oben genannten Problemfall, die Ziffern 7 und 8 zu Anfang im Sudoku vertauschen, dann hätte der Vector die Form

$(1, 0, 4, 15, 3, 0, 2, 9, 0)^T$

Wenn man aber nach der Lösung des Sudokus diesen Vector nach Häufigkeit absteigend sortiert, dann erhält man in beiden Fällen

$(15, 9, 4, 3, 2, 1, 0, 0, 0)^T$

Der Vector sagt nun aus, dass die am dritthäufigsten von der Methode *Coloring* ausgeschlossene Zahl vier mal ausgeschlossen wurde. Für die Sudokus mit gleicher Schwierigkeit ist der Featurevector nun gleich, hat aber an für die Klassifizierung relevanter Aussagekraft nicht verloren.

---

## 4 Software

---

Um die Ergebnisse der Klassifikation ermitteln zu können, wurde im Rahmen dieser Bachelorarbeit eine Software entwickelt, die es erlaubt, Sudokus aus Dateien einzulesen und den Featurevector zu extrahieren. Mit den Featurevectoren wird dann ein Klassifizierer trainiert und evaluiert. Ausserdem erlaubt es die Software, dass ein Mapping zwischen zwei verschiedenen Bewertungsskalen vorgenommen wird. Darauf werde ich im Kapitel *Eigene Software* 4.1 eingehen.

Die ganze Software ist in Java geschrieben, der Quellcode ist bei der Abgabe beigelegt. Ausserdem ist die Software momentan verfügbar auf <https://github.com/mbraeunlein/ExtendedHodoku>. Die Software lässt sich einfach über ein Terminal starten.

Für eine *cross validation* ist der Befehl

```
java -jar ExtendedHodoku cross trainSudokus.txt
```

Um einen Klassifizierer mit einem Testset zu evaluieren

```
java -jar ExtendedHodoku test trainSudokus.txt testSudokus.txt
```

Um ein Mapping zwischen verschiedenen Skalen herzustellen

```
java -jar ExtendedHodoku map sudokusSkala1.txt sudokusSkala2.txt
```

Und um die Featurevectoren in eine .arff Datei zu schreiben

```
java -jar ExtendedHodoku write sudokus.txt
```

Die Sudokus in den .txt Dateien müssen in einem definierten Format gespeichert werden. In einer Zeile steht jeweils ein Sudoku, bestehend aus 81 Ziffern. Jede Ziffer ist entweder 1 bis 9 für die jeweilige eingesetzte Zahl oder 0 für ein leeres Feld. Die Klassifikation steht jeweils in einer extra Zeile vor den Sudokus. Zum Beispiel steht in einer Datei zu Anfang die Zeile *Einfach*, dann wird die Klassifikation des Schwierigkeitsgrads der folgenden Sudokus *Einfach* sein, bis eine andere Zeile mit einer Klassifikation folgt. Klassifikationen dürfen nicht mit einer Zahl anfangen.

---

## 4.1 Eigene Software

---

Trotz der verwendeten Software ist im Rahmen dieser Arbeit auch eigene Software entwickelt worden. Die Software ist komplett in Java geschrieben. Hier möchte ich kurz auf die einzelnen Komponenten und deren Funktionen eingehen.

Zuerst wurde eine Klasse entwickelt, die es ermöglicht, Sudokus in einem definierten Format 4 einzulesen und in die von Hodoku verwendeten *Sudoku2* Objekte zu parsen. Die nächste, wichtige und selbst entwickelte Funktionalität ist das extrahieren eines Featurevectors. Das grobe Vorgehen ist in 3.3 beschrieben, die Implementierung findet sich in der Klasse */src/FeatureVectorExtractor.java*. Diese nimmt ein Objekt von *Sudoku2* entgegen und liefert ein *FeatureVector* Objekt zurück, indem sie das Sudoku schrittweise und mit Hilfe von Hodoku löst. Eine weitere, selbst entwickelte Klasse schreibt eine Menge von Featurevectors in eine .arff Datei, die von Weka verarbeitet werden kann. Natürlich wäre es möglich gewesen, die Featurevectors auch direkt an den Klassifizierer zu übergeben, durch das schreiben in eine .arff Datei bleibt dem Nutzer die Möglichkeit erhalten, die Daten später mit Weka genauer zu analysieren.

In *src/analyze/Analyzer.java* wurden die Modi implementiert, die benötigt werden, um Sudokus zu analysieren. Hier wird die Weka library verwendet.

Eine Besonderheit der Software findet sich in der Klasse */src/FeatureVectorExtractor.java*. Hier würde als letzte mögliche Methode *Backtracking* 2.4.15 angewendet werden. Diese löst jedes Sudoku vollständig durch *trial and error*. Es gibt kein Sudoku, das nicht durch *Backtracking* gelöst werden kann. Da diese Methode allerdings das ganze Sudoku auf einmal löst, wird sie nur angewendet, wenn keine andere Methode funktioniert. Für die Klassifikation ist nur relevant, wie viele Zahlen mit *Backtracking* ermittelt wurden. Das kann man auch schon vor dem Anwenden der Methode errechnen, da sie für alle offenen Felder die entsprechenden Zahlen findet. Dies kann man ohne Ausführen der Methode in den *FeatureVector* eintragen, was zu einer deutlichen Verbesserung der Laufzeit führt.

---

## 4.2 Fremdsoftware

---

Es wurden zwei andere Projekte für diese Arbeit verwendet.

Zum Generieren und schrittweisen Lösen von Sudokus wurde Hodoku<sup>1</sup> benutzt. Dieses Programm steht unter der GPLv3 Lizenz<sup>2</sup>.

Zur Klassifikation der Featurevectors wurde Weka<sup>3</sup> verwendet. Weka steht unter der GNU General Public License<sup>4</sup>.

---

## 4.3 Trainingsdaten

---

Um fundierte Aussagen über die Qualität der Klassifikation treffen zu können, wird eine große Menge an Trainingsdaten benötigt. Diese müssen bereits vollständig in Schwierigkeitsstufen eingeteilt worden sein. Das ist nötig, da der Klassifikator eine Schwierigkeitsstufe zuordnet und die Qualität der Zuordnung evaluiert werden soll. Um also festzustellen, ob der Klassifikator die richtige Klasse zugeordnet hat, muss diese bekannt sein.

Kostenlose und frei verfügbare Sudokus in digitaler Form mit definiertem Schwierigkeitsgrad lassen sich nicht leicht finden. Daher habe ich bei einigen großen Zeitungen, auf deren Websites Sudokus zu finden waren, nachgefragt, ob es möglich ist, ihre Sudokusammlungen zur Verfügung zu stellen. Die Anfragen wurden aber leider abgelehnt. Auf eine Anfrage an die Website <http://sudoku.soeinding.de/> wurden von sieben Schwierigkeitsgraden jeweils 32 Sudokus bereitgestellt. Da diese Trainingsdaten nicht

---

<sup>1</sup> <http://hodoku.sourceforge.net/de/index.php>

<sup>2</sup> <http://www.gnu.org/licenses/gpl-3.0.html>

<sup>3</sup> <http://www.cs.waikato.ac.nz/ml/weka/>

<sup>4</sup> <http://www.gnu.org/licenses/gpl.html>

---

ausreichten wurden mit dem open source Programm Hodoku<sup>5</sup> jeweils 1000 Sudokus von fünf unterschiedlichen Schwierigkeitsgraden generiert.

Wie schon an der Anzahl der Schwierigkeitsstufen zu erkennen ist, unterscheiden sich die Skalen der beiden Quellen. Daher konnten die Sudokus nicht gemeinsam klassifiziert werden. Auch war es nicht möglich, eine Quelle als Trainingsdaten für den Klassifikator zu verwenden um ihn anschließend mit der anderen Menge auszuwerten. Allerdings kann man eine Verbindung zwischen den Skalen suchen, zum Beispiel Klassen mit gleich schweren Sudokus.

---

<sup>5</sup> <http://hodoku.sourceforge.net/de/index.php>

---

## 5 Ergebnisse

---

278	160	179	196	187
181	266	196	157	200
222	213	222	175	168
228	208	206	174	184
228	220	206	191	155

---

## 6 Zusammenfassung und Ausblick

---