Matthias Rüster
Lena Kristin Zander


# Bundeswettbewerb Informatik: Aufgabe 3

## *Lösungsidee*

Diese Aufgabe kann, wie schon im *Taschenbuch der Algorithmen* beschrieben, mit Hilfe der simulierten Abkühlung (*simulated annealing*) gelöst werden.

Die Fitness eines Puzzles ist gleichbedeutend mit den darin enthaltenen Fehlern. Es wird angestrebt, dass die Fitnessfunktion den Wert 0 zurückliefert, das Puzzle also keine Fehler mehr enthält (Abbruchkriterium).

Wie es bei der simulierten Abkühlung üblich ist, gibt es eine *Starttemperatur*, die im Laufe der Simulation langsam abnimmt. Ein „*Verringerungswert*“ bestimmt, wie schnell die Temperatur abnehmen soll.

Um die Fitness eines Puzzle zu optimieren wählt man zufällig zwei verschiedene Steine des Puzzles und vertauscht diese. Man vergleicht dann die Strafpunkte, die das Puzzle vor der Vertauschung hatte mit den Strafpunkten, die das Puzzle nach der Vertauschung der Steine hat. Weist das Puzzle nach der Vertauschung mehr Strafpunkte auf als vorher, wird das „schlechtere“ Puzzle nur mit einer bestimmten Wahrscheinlichkeit behalten. Die Wahrscheinlichkeit, dass ein schlechteres Puzzle behalten wird ist abhängig von der aktuellen Temperatur:

$$P = \exp\left(-\frac{\Delta S}{T}\right) = \exp\left(-\frac{S_{nachher} - S_{vorher}}{T}\right)$$

*S* bezeichnet hierbei die Anzahl der Strafpunkte des Puzzles, *T* ist die aktuelle Temperatur und *exp* ist die natürliche Exponentialfunktion.

Ist also $P \geq random[0; 1]$ (zufällige Zahl zwischen 0 und 1), dann wird das schlechtere Puzzle behalten. Ist nach der Vertauschung ein Puzzle mit weniger Strafpunkten entstanden, dann wird dieses Puzzle ohne jegliche Prüfungen übernommen.

Das Vertauschen zweier Steine und die eventuelle Prüfung, ob das schlechtere Puzzle behalten wird, bezeichnet man als *Schritt*.

Das schwierige an der simulierten Abkühlung ist das Finden guter Parameter. Ein Parameter beispielsweise ist, wie lange eine bestimmte Temperatur beibehalten wird. Wie viele Schritte also bei einer bestimmten Temperatur ausgeführt werden (*Schrittanzahl*). Außerdem müssen *Starttemperatur* und *Verringerungswert* gut gewählt werden.

Hier nochmal der gesamte Algorithmus in Pseudocode:

```
Temperatur := Starttemperatur
SOLANGE AnzahlStrafpunkte(Puzzle) ≠ 0 UND Temperatur > 0
      FÜR JEDES i VON 0 BIS Schrittanzahl WIEDERHOLE
            vorher  := AnzahlStrafpunkte(Puzzle)
            Wähle zwei verschiedene Steine des Puzzles und vertausche sie
            nachher := AnzahlStrafpunkte(Puzzle)
            WENN nachher > vorher DANN
                  Behalte schlechteres Puzzle mit der Wahrscheinlichkeit P
            ENDE WENN
      ENDE WIEDERHOLE
      Temperatur := Temperatur - Verringerungswert
ENDE SOLANGE
```

Für dieses Puzzle (12 · 8 Steine, 4 Farben) haben wir folgende Parameter gewählt, mit denen ein schnelles Erreichen des Optimums möglich ist:

| | |
|---|---|
| *Starttemperatur* | 0,22 |
| *Schrittanzahl* | 20000 |
| *Verringerungswert* | 0,00002 |

## *Programm-Dokumentation*

Die Anfangsordnung wird mit der Funktion **`read_puzzle`** eingelesen. Diese Funktion bekommt den Dateinamen der Anfangsordnung als Parameter übergeben und gibt das eingelesene Puzzle als **`puzzle`**-Struktur zurück. Beim Einlesen werden die Farben in Zahlen umgewandelt, um diese später besser vergleichen zu können. Die **`puzzle`**-Struktur besteht aus einem zweidimensionalen Array von **`stone`**-Strukturen, die die Spielsteine darstellen. Zusätzlich werden auch Höhe und Breite des Arrays in der **`puzzle`**-Struktur gespeichert, sowie die Anzahl der Strafpunkte.

Eine **`stone`**-Struktur besteht aus vier **`char`**-Variablen, in der die vier Farben eines Spielsteins gespeichert werden.

Nachdem die Anfangsordnung eingelesen wurde, wird der Algorithmus zur simulierten Abkühlung gestartet: Die Funktion **`simulated_annealing`** bestimmt zunächst die Anzahl der Strafpunkte der Anfangsordnung. Dazu wird die Funktion **`count_faults`** aufgerufen. In dieser Funktion werden die Farben der jeweils nebeneinander liegenden Spielsteine miteinander verglichen. Sind diese verschiedenfarbig, wird ein Strafpunkt dazu addiert.

Danach beginnt der oben beschriebene Algorithmus: Es werden zufällig zwei verschiedene Spielsteine gewählt und vertauscht. Dann werden die Strafpunkte vor und nach der Vertauschung verglichen. Ist das Puzzle „schlechter“ geworden, sind also nach der Vertauschung Strafpunkte hinzugekommen, wird das schlechtere Puzzle nur mit der Wahrscheinlichkeit $P$ behalten. Andernfalls werden die Steine einfach wieder zurückgetauscht.

Anstatt das ganze zweidimensionale Array jedes Mal zu durchlaufen, um die Strafpunkte zu ermitteln, reicht es, nur die Strafpunkte der beiden Steine, die man vertauschen wird, zu betrachten. Dadurch wird einiges an Rechenzeit gespart.

Nebenbei wird sich außerdem das jeweils beste Puzzle gemerkt. Falls das optimale Ergebnis nämlich einmal nicht erreicht wird, kann man dem Benutzer immer noch die beste ermittelte Lösung präsentieren. Da wir aber gute Parameter für die simulierte Abkühlung gewählt haben, wird das optimale Ergebnis mit einer Wahrscheinlichkeit von ca. 96 % ermittelt.

Das Puzzle mit den wenigsten Strafpunkten wird als ein **`png`**-Bild abgespeichert. Dafür haben wir die GD Graphics Library[1] benutzt.

1 http://www.boutell.com/gd/

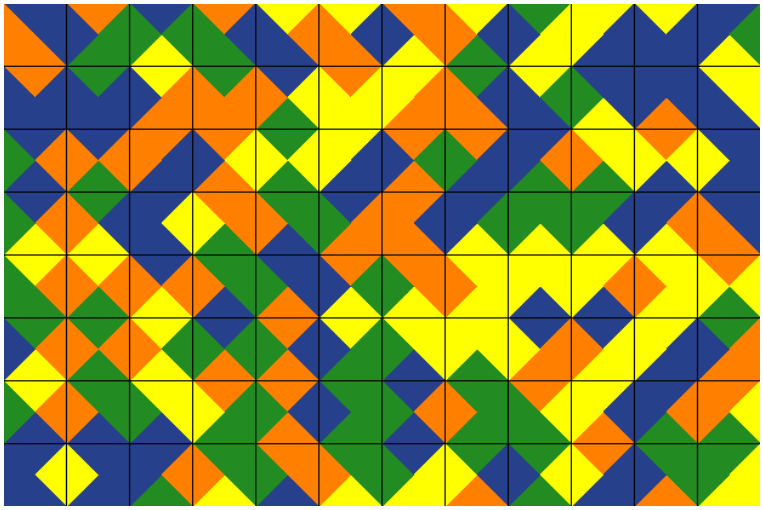
## *Programm-Ablaufprotokoll*

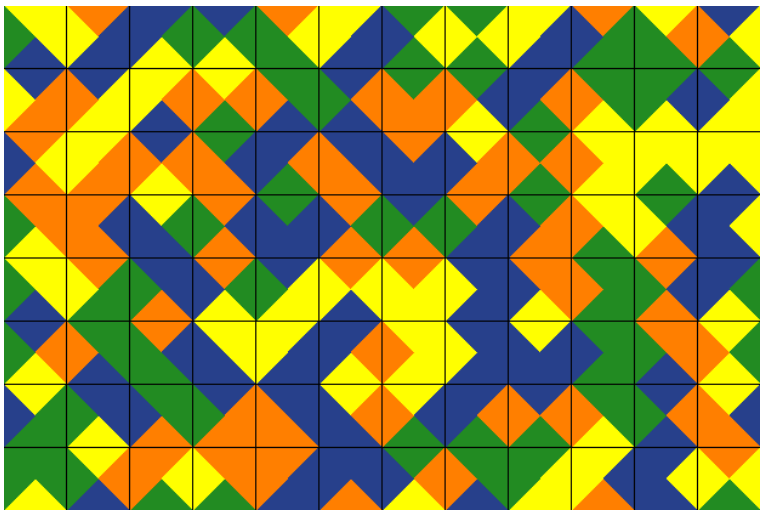
Das Programm wird mit dem Namen der Datei, die die Anfangsordnung des Puzzles beinhaltet, aufgerufen:

```
$ ./aufgabe3 anfangsanordnung.txt
Es wurde ein optimales Ergebnis gefunden.
Das Ergebnis wurde in der Datei 0.png gespeichert.
```

Falls das Programm mehr Zeit benötigt, sieht der Benutzer eine „Statuszeile“, die die aktuelle Temperatur und die Strafpunkte des Puzzles anzeigt.

Es gibt viele optimale Lösungen, deshalb werden hier nur zwei abgedruckt:





Mit der Option -h wird eine Hilfe zur Verwendung des Programms ausgegeben:

```
$ ./aufgabe3 -h
Verwendung: aufgabe3 [Optionen] Dateiname

moegliche Optionen:
    -t Temperatur    Starttemperatur festlegen
    -s Anzahl        Schrittanzahl festlegen
    -d Wert          Verringerungswert festlegen
    -h               diese Hilfe anzeigen
```

Die Parameter für die simulierte Abkühlung können also mit den Parametern **`-t`**, **`-s`** und **`-d`** gewählt werden. Möchte man beispielsweise eine höhere Starttemperatur, ruft man das Programm einfach mit dem Parameter **`-t 0.34`** auf.

Das Anpassen der Parameter wäre z. B. dann sinnvoll, wenn man ein anderes Puzzle lösen möchte, welches nicht aus $12 \cdot 8$ Feldern besteht.