Ico-Soku-Solver Vorgehensweise

Georg Jenschmischek, 14INM, Matrikelnummer: 64572

# Beginn mit simplem Back-Tracking

Zu Beginn meiner Lösung des Ico-Soku-Problems stand die Implementierung des grundlegenden Back-Tracking-Algorithmus. Die Rekursion erfolgt bei mir über alle zwanzig Flächen der Reihe nach von 0 bis 19. In jedem Rekursionsschritt werden alle noch verfügbaren Plättchen jeweils in allen drei Orientierungen auf die Fläche gesetzt. Anschließend werden die drei von der Fläche betroffenen Ecken betrachtet. Um diese zu erhalten wurde von mir die IcoSoku-Klasse um eine Zuordnungsmatrix von Fläche auf ihre Ecken sowie um eine spezielle Methode zum Auslesen dieser Beziehungen erweitert. Bei den von der Fläche betroffenen Ecken wird geprüft, ob die anliegende Summe an Punkten kleiner oder gleich der Soll-Summe ist. Sobald fünf Plättchen anliegen wird außerdem geprüft, ob die Summe der anliegenden Punkte gleich der Soll-Summe ist. Sollten beide Bedingungen erfüllt sein, wird ein Rekursionsschritt tiefer gegangen, also eine neue Fläche betrachtet. Wenn nicht wird das Plättchen gedreht, oder ein neues Plättchen ausprobiert. Sollte kein noch verfügbares Plättchen auf die Fläche passen, so wird Back-Tracking eingeleitet und eine Rekursionsebene zurückgegangen. Dort wird dann wiederum das aktuelle Plättchen gedreht, oder ein neues Plättchen ausprobiert.

Nachdem dieser grundsätzliche Back-Tracking-Algorithmus implementiert wurde, erfolgte eine erste Zeitmessung für die Dauer der Lösungsberechnung. Nach über 40 Minuten musste der Test allerdings erfolglos abgebrochen werden. Im aktuellen Status benötigt der Ico-Soku-Solver noch zu viel Zeit zum Lösen des Problems.

# Optimierungen durch Ausnutzen von Plättcheneigenschaften

Um die Lösung des Ico-Soku-Rätsels schneller zu erhalten wurden zuerst zwei Optimierungsverfahren verfolgt, die auf bestimmten Plättcheneigenschaften aufbauen.

Einerseits betrifft dies die symmetrischen Plättchen [1,1,1], [2,2,2] sowie [3,3,3]. Diese müssen im Gegensatz zu anderen Plättchen nicht gedreht werden, da sie in jeder Orientierung die gleiche Punktzahl besitzen.

Andererseits existieren die Plättchen [0,1,2] und [0,2,1] je dreimal und die Plättchen [1,2,3] und [1,3,2] je zweimal. Von jedem dieser vier Plättchengruppen muss immer nur je ein Plättchen überprüft werden, die doppelten können ignoriert werden.

Mit Hilfe dieser Optimierungen konnte zwar die Anzahl der Lösungsmöglichkeiten, welche der Algorithmus untersuchen muss, eingeschränkt werden, trotzdem terminierte der Ico-Soku-Solver-Algorithmus auch nach 40 Minuten Wartezeit nicht.

# Optimierung durch Betrachtung der Punkterestanzahl

Um zeitig zu erkennen, ob ein Plättchen überhaupt an die gerade betrachtete Fläche passt, wurde ein Zusammenhang zwischen den noch fehlenden Punkten an einer Ecke und den noch freien Flächen um eine Ecke ausgenutzt. Mit einem Plättchen können höchstens drei Punkte zur Punktesumme an einer Ecke hinzugefügt werden. Fehlen beispielsweise an einer Ecke noch zehn Punkte, es sind aber nur noch drei Flächen frei, so kann das aktuell betrachtete Plättchen nicht passen. Ein Plättchen passt also nur an eine Ecke, wenn die noch fehlende Punkteanzahl kleiner oder gleich der noch freien Flächen um die betreffende Ecke multipliziert mit drei ist. Da die Punkteobergrenze einer Ecke zwölf ist, greift diese Optimierung erst ab zwei Plättchen an einer Ecke.

Mit Hilfe dieses Zusammenhangs konnte die Eckenüberprüfung nach dem setzen eines Plättchens so umgeschrieben werden, dass zunächst die noch fehlende Punkteanzahl an allen betroffenen Ecken bestimmt wurde. Ist diese negativ (Ecken-Punktzahl überschritten) oder größer als die Anzahl der noch freien Flächen um die betreffende Ecke multipliziert mit drei, so passt das Plättchen nicht.

Mit Hilfe dieser Verbesserung konnte der Algorithmus auf eine Lösungszeit von unter zwei Sekunden verbessert werden.

# Optimierung durch Flächensortierung

Nachdem der Algorithmus in respektabler Zeit eine Lösung errechnen konnte, sollte mit der letzten Optimierung vor allem die stark schwankende Lösungszeit behoben werden. So brauchte der Algorithmus für den Best-Case ([1,2,3,4,5,6,7,8,9,10,11,12]) nur 1ms während er für den Worst-Case ([12,11,10,9,8,7,6,5,4,3,2,1]) 1,4s benötigte. Um dieses Problem zu beheben, sollte der Algorithmus nicht mehr die Flächen von 0 bis 19 der Reihe nach durchlaufen, sondern eine spezielle Sortierung der Flächen nutzen, um schneller eine Lösung zu finden.

Der erste Ansatz war dabei, die Flächen so zu sortieren, dass zuerst alle Fläche um die Ecke mit dem Wert 1, dann alle Fläche um die Ecke mit dem Wert 2 usw. betrachtet werden. Dies behob zwar den vergleichsweise lange Laufzeit im Fall [12,11,10,9,8,7,6,5,4,3,2,1], verschlechterte aber wiederum die Zeiten im Fall [6,12,9,10,7,2,8,1,3,5,4,11] auf 1,6s. Damit war das eigentliche Problem also nicht gelöst, sondern nur verschoben.

Aus diesem Grund wurde ein zweiter Ansatz betrachtet. Dieser sortiert die Flächen, ähnlich wie beim simplen Back-Tracking, so, dass zuerst eine alle fünf Flächen um eine Ecke betrachtet werden. Anschließend wird das „Mittelband“ bestehend aus den zehn anliegenden Flächen bearbeitet. Abschließend werden die fünf Flächen um die gegenüber der Startecke liegenden Ecke betrachtet. Mit dieser Reihenfolge wird besonders schnell an vielen Ecken eine Plättchenanzahl größer zwei erreicht, sodass die Optimierung durch Betrachtung der Punkterestanzahl greifen kann. Als Startecke gilt dabei im Gegensatz zum simplen Back-Tracking nicht die Ecke 0, sondern die Ecke, welche den Wert 1 besitzt. Dies hat sich als besonders günstiger Startwert herausgestellt.

# Zeitmessungen

Die Zeitmessung wurde mit den drei gegebenen Probleminstanzen, sowie dem selbst gefundenen Worst-Case, durchgeführt. Verglichen werden die Algorithmen mit und ohne Flächensortierung.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Problem 1 | Problem 2 | Problem 3 | Worst-Case | Summe |
| Ohne Flächensortierung | 13 ms | 1 ms | 218 ms | 1443 ms | 1675 ms |
| Mit Flächensortierung | 13 ms | 1 ms | 22 ms | 6 ms | 43 ms |

Wie deutlich zu erkennen ist, lohnt sich die Flächensortierung in der Summe vom Aufwand her sehr, da sie nur minimale zusätzliche Zeitkosten, dafür aber bei einigen Fällen hohen Zeitgewinn verursacht.

Die Zeitkosten könnten weiterhin verringert werden, indem die Flächensortierung nicht algorithmisch bestimmt, sondern bereits als Zuordnungsmatrix für direkten Zugriff bereitliegen würde. Dann müsste nur noch die Ecke mit dem Wert 1 bestimmt und die entsprechende Sortierung abgerufen werden.