Kombinatorik bei „Schiffe versenken“

Henning Thielemann

31. Mai 2019

Zusammenfassung

Wir berechnen die Anzahl der Möglichkeiten, um die Schiffe beim Spiel „Schiffe versenken“ anzuordnen. Die Antwort: Auf einem 10×10-Spielfeld gibt es 26 509 655 816 984 (etwa 26,5 Billionen) Möglichkeiten, ein Schiff der Länge 5, zwei Schiffe der Länge 4, drei Schiffe der Länge 3 und vier Schiffe der Länge 2 mit Zwischenräumen zu plazieren. Dabei sind Drehungen und Spiegelungen nicht herausgerechnet.

Die gute Nachricht: Auf einem aktuellen Rechner lässt sich die Antwort in etwa 6 Minuten berechnen. Die schlechte Nachricht: Bei dem verwendeten Ansatz wachsen die Rechenzeit und der Speicherverbrauch sehr stark mit der Breite des Spielfeldes. Die gute Nachricht: Die Rechenzeit wächst nur proportional mit der Höhe des Spielfeldes und der Speicherverbrauch hängt gar nicht von der Höhe des Spielfeldes ab.

1 Motivation

Am 4. Dezember 2011 erhielt ich von Manuela Kugel[1] die Frage, wie wahrscheinlich es ist, dass zwei Spieler beim Spiel „Schiffe versenken“[2] unabhängig voneinander die Schiffe gleich aufstellen. Dies war an jenem Tag bei einem Spiel zwischen ihrem Sohn und dessen Opa passiert. Die Frage lässt sich streng genommen nur beantworten, wenn man die Wahrscheinlichkeitsverteilung kennt, die die Spieler beim Aufstellen der Schiffe befolgen. Der Einfachheit halber nahm ich Gleichverteilung an. Mehr zu dieser Annahme in Unterabschnitt 6.2. Unter dieser Annahme bleibt „nur“ noch die Frage zu klären, wieviele verschiedene Möglichkeiten es gibt, einen vorgegebenen Satz an Schiffen auf einem Spielfeld zu verteilen.

In zahlreichen WWW-Foren wurde die Frage nach der Anzahl der Möglichkeiten bereits für eine Vielzahl von Spielvarianten gestellt (Spielfeldgröße, Auswahl Schiffe, dürfen sich Schiffe berühren oder nicht), aber nie beantwortet. Es wurden einige Vorschläge gemacht, wie man das Problem mit Hilfe von Inklusion und Exklusion lösen könnte. Mit diesem Ansatz bekommt man schnell sehr großzügige Abschätzungen, aber die genaue Berechnung stellt selbst schon wieder ein kombinatorisches Problem dar, denn es müssen alle Möglichkeiten untersucht werden, wie sich bis zu zehn Schiffe überlappen können.

Also begann ich meine eigenen Lösungsversuche mit der rein funktionalen Programmiersprache Haskell.[3] Zuerst probierte ich mich an einem eigenen Ansatz, bei dem ich bald feststellte, dass ich Möglichkeiten doppelt zähle. Dann hatte ich einen korrekten Ansatz, der aber nur bis Breite 9 funktionierte und danach wegen Speichermangels ausstieg. Beim letzten und erfolgreichen Ansatz lagerte ich eine große Menge an Daten auf die Festplatte aus. Damit konnte ich am 8. Januar 2012 endlich das Problem lösen.

[1] `http://www.olympiade-mathematik.de/`
[2] `http://de.wikipedia.org/wiki/Schiffe_versenken`
[3] `http://hackage.haskell.org/package/battleship-combinatorics`

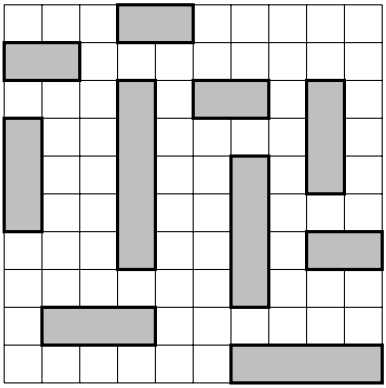

Abbildung 1: Beispiel für eine Aufstellung

2 Abschätzungen

Wir wollen mit Abschätzungen für die Anzahl der Aufstellungen beginnen. Angenommen wir haben ein Spielfeld der Größe $m \times n$, die Schiffe lassen sich alle unterscheiden und sie dürfen sich zudem überschneiden. Dann ist die Anzahl $N^*_{m,n}$ möglicher Aufstellungen:

$$N^*_{m,n} = f_{m,n}(5) \cdot f_{m,n}(4)^2 \cdot f_{m,n}(3)^3 \cdot f_{m,n}(2)^4 \quad (1)$$

$$f_{m,n}(k) = \max(0, n-k+1) \cdot m + \max(0, m-k+1) \cdot n \quad (2)$$

Dabei steht $f_{m,n}(k)$ für die Anzahl der Möglichkeiten, um ein Schiff der Länge k auf einem $m \times n$-Spielfeld zu plazieren. Für die Anzahl $N_{m,n}$ der Aufstellungen ohne Unterscheidung der Schiffe und ohne Berührungen gilt entsprechend:

$$2! \cdot 3! \cdot 4! \cdot N_{m,n} \leq N^*_{m,n} \quad (3)$$

Der höchste Term des Polynoms $N^*_{m,n}$ ist $(2 \cdot n \cdot m)^{10}$. Dies entspricht der asymptotischen Anzahl für sehr große Spielfelder. Bei sehr großen Spielfeldern fallen Überschneidungen und selbst die Länge der Schiffe nicht mehr ins Gewicht. Dieser Term ist auch der erste Summand bei einer Berechnung nach dem Prinzip von Inklusion und Exklusion.

Abschätzung 3 ist für unsere Zwecke aber zu grob. Sie sagt uns, dass es auf einem 10×6-Feld höchstens 80 Billiarden Möglichkeiten gibt. Diese Aussage ist zwar richtig, tatsächlich gibt es aber überhaupt keine Möglichkeit. Dies kann man leicht sehen, wenn man sich den Flächenverbrauch der Schiffe anschaut. Wir tragen dem Flächenverbrauch Rechnung, in dem wir jedes Schiff und auch das Spielfeld unten und rechts um einen Sicherheitsstreifen erweitern (vgl. Abbildung 11).

Fläche Schiffe mit Abstand		Fläche Rechteck
$2 \cdot ((5+1) + 2 \cdot (4+1) + 3 \cdot (3+1) + 4 \cdot (2+1))$	$>$	$(10+1) \cdot (6+1)$
80	$>$	77 (4)

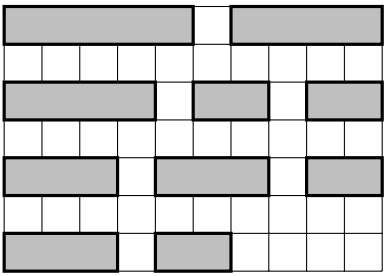

Abbildung 2: Minimale Feldgröße für die normale Schiffsflotte

Es gibt erst ab der Spielfeldgröße 10×7 Möglichkeiten, siehe Abbildung 2. Daraus folgt, dass man die Überschneidungen unbedingt berücksichtigen muss, wenn man aussagekräftige Abschätzungen haben möchte.

3 Induktive Berechnung

Wir wollen die Anzahl der Möglichkeiten per Induktion[4] über die Höhe des Spielfeldes berechnen. Dazu müssen wir von der Anzahl von Aufstellungen auf einem Spielfeld der Größe $10 \times n$ auf eines mit der Größe $10 \times (n + 1)$ schließen. Tatsächlich reicht es aber nicht, abgeschlossene Spielfelder zu betrachten. Vielmehr müssen wir $10 \times n$-Felder betrachten, die durch Abschneiden eines 10×10-Feldes entstehen. Für die folgenden Erklärungen benötigen wir die Begriffe „Flotte“ und „Grenze“.

[4] Man kann es auch dynamische Programmierung nennen.

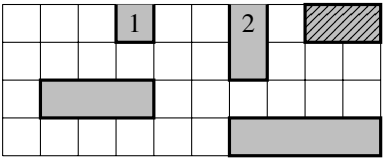

Abbildung 3: Abgeschnittene Schiffsaufstellung, wie sie in einem Induktionsschritt auftreten kann.

Definition 1 (Flotte)**.** Unter Flotte verstehen wir eine Multimenge von Schiffslängen. Die Standardflotte ist $[2, 2, 2, 2, 3, 3, 3, 4, 4, 5]$.

Definition 2 (Grenze)**.** In unserem induktiven Ansatz betrachten wir Teile des Spielfeldes, die entstehen, wenn man das Spielfeld waagerecht entlang einer Gitterlinie abschneidet. Dann betrachten wir die Schiffe, die unterhalb entlang der Schnittlinie aufgestellt sind. Dies wollen wir als „Grenze“ bezeichnen. Die in Abbildung 3 dargestellte Grenze würden wir so kodieren: $(\square, \square, \square, 1, \square, \square, 2, \square, \blacksquare, \blacksquare)$.

Für die Induktion schließen wir von abgeschnittenen Spielfeldern der Größe $10 \times n$ auf welche der Größe $10 \times (n + 1)$. Dabei müssen wir alle diejenigen Details mitführen, die darüber entscheiden, wie die hinzukommende Spielfeldzeile aussehen kann. Diese Details sind vollständig mit einem Paar (Grenze, Flotte) abgedeckt. Für die Kombinatorik müssen wir für jedes $10 \times n$-Teilspielfeld eine Funktion von (Grenze, Flotte) nach der Anzahl der Möglichkeiten berechnen. Der Induktionsschritt besteht im Wesentlichen daraus, für Paare der Form (Grenze, Flotte) für ein $10 \times n$-Teilspielfeld alle möglichen Paare von (Grenze, Flotte) für ein $10 \times (n + 1)$-Teilspielfeld zu finden und die entsprechenden Funktionswerte

zu addieren.

Die Induktion beginnt bei einem Feld der Größe 10×0. Für dieses Feld gibt es genau eine Aufstellung mit dem Paar $((\square, \square, \square, \square, \square, \square, \square, \square, \square, \square), [])$.

Für den Induktionsschritt gehen wir für jedes Paar von (Grenze, Flotte) die neu aufzubauende Zeile von links nach rechts ab und überlegen uns für jede Zelle, wie sie belegt werden kann. Die aktuelle Zelle kann belegt werden mit:

- einem leeren Feld:
  In jedem Fall. Falls die darunter liegende Zelle Teil eines senkrecht angeordneten Schiffes ist, dann wird das Schiff abgeschlossen und in die Flotte aufgenommen.

- einem Teil eines waagerecht angeordneten Schiffs:
  Wenn die links, rechts und direkt darunter liegenden Zellen frei sind. Jedes waagerecht aufgestellte Schiff wird in die Flotte aufgenommen.

- einem Teil eines senkrecht angeordneten Schiffs:
  Wenn die direkt darunter liegende Zelle ebenfalls Teil eines senkrecht angeordneten Schiffes ist. Die Schiffsgröße wird in der Grenze um eins erhöht. Wenn die links, rechts und direkt unter der aktuellen Zelle liegenden Zellen frei sind, dann können wir ein neues senkrecht angeordnetes Schiff mit der Länge 1 beginnen.

Die wichtige Erkenntnis ist, dass wir bei der Induktion immer Flotte und Anordnung gleichzeitig betrachten müssen. Wir können keine Induktion allein über die Feldgröße oder allein über die Flottengröße durchführen. Das macht die Berechnung so aufwändig. Anhang A zeigt die Ergebnisse unserer Berechnung.

4 Optimierungen

4.1 Mathematische Optimierungen

Die Induktion funktioniert problemlos, wenn man die entstehenden Flotten nirgendwo beschränkt. Wir können dann nach Ablauf des Algorithmus sehen, welche Flotten sich überhaupt auf dem Spielfeld unterbringen lassen. Wenn wir umgekehrt die Flotte vorgeben, dann lässt sich die Rechenzeit deutlich verkürzen, wenn wir frühzeitig zu große Flotten aussortieren. Dabei gibt es zwei Möglichkeiten für eine Flotte, zu groß zu sein. Entweder überschreitet für eine bestimmte Schiffsgröße die Anzahl der Schiffe diejenige in der vorgebenen Flotte oder aber die Länge eines Schiffes wird zu groß.

Der erste Fall ist einfach: Es sei y die vorgegebene Flotte. Im Induktionsschritt fügen wir einer Flotte ein neues Schiff hinzu, sodass Flotte x entsteht. Nun müssen wir testen, ob $\forall j : x_j \leq y_j$ zutrifft. Falls ja, können wir fortfahren, andernfalls brauchen wir das Schiff nicht hinzuzufügen.

Der zweite Fall ist etwas kniffliger: Angenommen, die vorgegebene Flotte besteht nur aus einem Schiff der Länge 5 und wir haben eine Grenze mit einem vertikal angeordneten Schiff, das bei Länge 3 abgeschnitten ist. Da das Schiff noch auf Länge 5 wachsen kann, müssen wir diesen Fall weiterverfolgen. Ein direkter Vergleich der Tupel $(0, 0, 1, 0, 0)$ mit $(0, 0, 0, 0, 1)$ würde fälschlicherweise ergeben, dass die Teilflotte bereits zu groß geworden sei. Hätten wir hingegen zwei bei Länge 3 abgeschnittene Schiffe an der Grenze, dann wäre die Flotte tatsächlich bereits zu groß, allein schon wegen der Gesamtanzahl der Schiffe.

Wie können wir nun schnell testen, ob jedes abgeschnittene Schiff noch auf eine zulässige Länge wachsen kann? Zunächst einmal stellen wir fest, dass wir nicht zwischen abgeschnittenen und abgeschlossenen Schiffen unterscheiden müssen. Wir erlauben es fertigen Schiffen, noch zu wachsen, auch wenn sie es nicht tun.

Für den Test beginnen wir mit der maximalen Schiffslänge k, betrachten

schrittweise kleiner werdende Längen und berechnen für jede Länge j den Überschuss d_j an Schiffen mit mindestens Länge j in der Zielflotte.

$$d_{k+1} = 0, \qquad d_j \quad = \quad d_{j+1} + y_j - x_j \tag{5}$$

Wenn für ein j gilt $d_j < 0$, dann bedeutet dies, dass für ein abgeschnittenes Schiff kein längeres Schiff mehr in der Zielflotte zu haben ist.

Zusätzlich wollen wir die sogenannten kumulierten Flotten X und Y einführen:

$$X_{k+1} = 0, \qquad \forall j \leq k: \quad X_j \quad = \quad X_{j+1} + x_j \tag{6}$$

$$Y_{k+1} = 0, \qquad \forall j \leq k: \quad Y_j \quad = \quad Y_{j+1} + y_j \tag{7}$$

Die Zahlen X_j und Y_j sind gewissermaßen Partialsummen in umgekehrter Richtung der Tupel x bzw. y. Es gilt

$$d_j \quad = \quad Y_j - X_j \tag{8}$$

Wir müssen d_j gar nicht direkt berechnen, es reicht der Vergleich $X_j \leq Y_j$, d.h. wir müssen einen Tupel-Vergleich wie im ersten Fall lediglich zusätzlich auf die kumulierten Flotten anwenden. Vorteil dieser Herangehensweise ist, dass wir Y einmal berechnen und speichern können, denn die Zielflotte ist während der gesamten Berechnung immer die gleiche.

Wir kommen zu einer Änderung, die die Anzahl der möglichen Grenzen verringert. Bisher haben wir als Grenze die Aufstellung der Schiffe direkt unterhalb der Schnittstelle verwendet. Alternativ können wir als Grenze diejenigen Zellen betrachten, die in der nächsten Zeile nicht mit Schiffen belegt werden können. In Abbildung 4 zeigen wir, wie sich die Grenzdefinition auf diese Weise ändert. Der Vorteil dieser Definition ist, dass manche vorher verschiedene Schiffsaufstellungen nun die gleiche Grenze markieren. Ein Beispiel dafür zeigen wir in Abbildung 5.

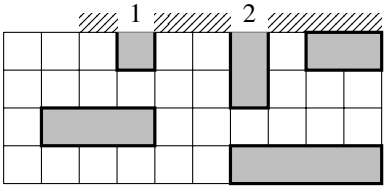

Abbildung 4: Grenze wie in Abbildung 3, wobei wir blockierte Felder markieren und nicht nur durch Schiffe belegte

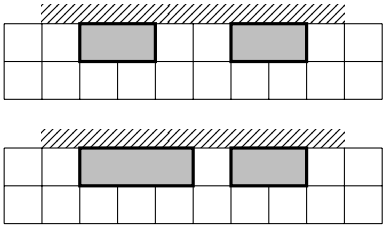

Abbildung 5: Zwei verschiedene Aufstellungen mit gleichem Muster blockierter Zellen in der nächsten Zeile

Der Änderung der Grenzdefinition entsprechend müssen wir die Fallunterscheidung in Abschnitt 3 anpassen.

Eine weitere Optimierung besteht darin, die Grenzen nach jedem Induktionsschritt immer dann zu spiegeln, wenn die Grenze dadurch lexikografisch kleiner wird. Wir erreichen damit fast eine Halbierung der Anzahl zu bearbeitender Grenzen.

Wir könnten dafür sorgen, dass die gefundenen Aufstellungen bis auf Rotation und Spiegeln eindeutig sind. Da sich das 5er-Schiff nicht rotations- oder spiegelsymmetrisch plazieren lässt, würde es reichen, das 5er-Schiff nur waagerecht und immer in die untere Hälfte des Spielbrettes zu setzen. Wir tun das in unserem Programm aus verschiedenen Gründen nicht: Erstens ist es im Spiel natürlich ein großer Unterschied, ob das Brett gedreht oder gespiegelt wird. Zweitens würde es das Programm komplizierter machen. Wir müssten zwei Spielbretthälften unterschiedlich behandeln und wir müssten waagerecht und senkrecht angeordnete Schiffe in der Flotte unterscheiden. Drittens behindert es die Verallgemeinerung auf andere Spielfeldgrößen und Flotten. Die Standardflotte lässt auf einem Feld mit ungeraden Seitenlängen symmetrische Aufstellungen zu. Gleiches gilt für Flotten mit geraden Anzahlen für alle Schiffe einer ungeraden Länge auf einer beliebigen Spielfeldgröße. Anstatt die Rotationen und Spiegelungen herauszurechnen, nutzen wir die Symmetrien zur Plausibilitätskontrolle. Wenn der Algorithmus korrekt ist, dann muss die Anzahl der Aufstellungen durch acht teilbar sein.

4.2 Maschinennahe Optimierungen

Neben den mathematischen Optimierungen wollen wir auch maschinennahe Optimierungen einsetzen. Vorteil dieser Optimierungen sind eine geringere Rechenzeit und ein geringerer Speicherverbrauch. Ein Nachteil ist hingegen ein geringerer Schutz gegen Fehler wie Überläufe. Im Folgenden möchte ich eine Reihe von Optimierungen angeben, die nicht so offensichtlich sind.

0	0	0	1	2	3	4	0

Abbildung 6: Hexadezimale Kodierung der Standardflotte.

4.2.1 Darstellung der Flotte

Für die Schiffsflotte können wir ein 32-Bit-Wort verwenden, das wir in 8 Elemente zu 4 Bits unterteilen. Diese Unterteilung lässt sich als achtstellige Hexadezimalzahl auffassen (siehe Abbildung 6). Damit können wir Schiffe bis zur Länge 8 verwalten, wobei von jeder Länge bis zu 15 Schiffe vorhanden sein können. Eine häufig benötigte Operation ist der Vergleich zweier Schiffsflotten, seien es einfache oder kumulierte Flotten. Für zwei Flotten x und y wollen wir testen ob $\forall j : x_j \leq y_j$. Dies kann man mit einer konstanten Anzahl Operation erreichen, d.h. unabhängig von der Wortgröße und der Unterteilung bleibt die Anzahl der Operationen immer gleich. Als Operationen benötigen wir lediglich eine Subtraktion und übliche Bitmanipulationsroutinen.

Der Test funktioniert so: Wir interpretieren die beiden Wörter für x und y als vorzeichenlose Zahlen und subtrahieren sie voneinander. Sollte für ein j $x_j > y_j$ gelten, so würden wir das daran bemerken, dass die Einerstelle für das Schiff der Größe $j + 1$ nicht korrekt ist. Die korrekte Einerstelle ermitteln wir durch ein bitweises Entweder-Oder beider Wörter. Für die größte Schiffsgröße haben wir keine nächsthöhere Größe, an der wir den Fehler in der Einerstelle prüfen können. Deswegen brauchen wir noch einen normalen Größenvergleich beider Wörter. Dieser entscheidet sich an der höchsten Schiffsgröße, wo beide Flotten eine verschiedene Anzahl an Schiffen besitzt. Der vollständige Haskell-Code für den Test sieht so aus:

0	0	0	1	3	6	A	A

Abbildung 7: Kumulierte Standardflotte.

```
subset :: Word32 -> Word32 -> Bool
subset x y =
   x<=y  &&  xor (xor x y) (y-x) .&. 0x11111111 == 0
```

Bemerkung: Dieser Ansatz ist nicht dazu geeignet, alle diejenigen Schiffsgrößen j zu finden, wo $x_j > y_j$ gilt. Das liegt daran, dass sich ein Übertrag auf höhere Schiffsgrößen als nur die nächsthöhere auswirken kann. Beispiel: `0x00010000`$-$`0x00000001` $=$ `0x0000FFFF`. Unser Test findet nur sicher die kleinste Schiffsgröße j, für die $x_j > y_j$ gilt.

Die andere nicht-triviale Operation für Flotten ist die Berechnung der kumulativen Flotte (Abbildung 7). Auch diese lässt sich mit einer konstanten Anzahl Operationen berechnen, wenn der Prozessor die Ganzzahldivision beherrscht. Wir machen uns dabei die binomische Formel zunutze:

$$\frac{t^n - 1}{t - 1} = t^{n-1} + \ldots + t + 1$$

Der entsprechende Haskell-Code sieht so aus:

```
cumulate :: Word32 -> Word32
cumulate x =
   case divMod x 15 of (q,r) -> shiftL q 4 .|. r
```

Bemerkung: Die Einerstelle des Ergebnisses wird auf `mod x 15` gesetzt. Dies entspricht dem bekannten Umstand, dass bis auf Überläufe die Quersumme einer

0	0	0	1	0	0	2	0	7	7

Abbildung 8: Oktale Kodierung der Grenze in Abbildung 3

Zahl dem Rest bei der Division durch (Basis-1) entspricht. Um Überläufe zu vermeiden, muss die Anzahl aller Schiffe kleiner als 15 sein. 15 Schiffe sind also schon zuviel, obwohl man sie noch mit unserem kompakten Bitformat darstellen könnte.

Der Divisionstrick ließe sich übrigens selbst dann anwenden, wenn man trotz Binärdarstellung in einer anderen Basis rechnete. Beispielsweise ließen sich in einem 32-Bit-Wort mit Hilfe einer Dezimaldarstellung auch Schiffe bis zur Größe 9 verwalten, wobei von jeder Größe maximal 8 Schiffe existieren dürfen.

4.2.2 Darstellung der Grenze

Für die Darstellung der Situation an der Grenze des abgeschnittenen Spielfeldes genügt ebenfalls ein 32-Bit-Wort. Wir unterteilen es in 10 Elemente zu 3 Bits (Abbildung 8). Dies entspricht einer zehnstelligen Oktalzahl. Dabei stehen die Oktalziffern 1 bis 6 für die Länge eines abgeschnittenen senkrechten Bootes, 0 für ein freies Feld und 7 für ein blockiertes Feld.

Besonders interessant ist die Operation zum Spiegeln der Grenze, um zueinander spiegelverkehrte Spielfelder zusammenfassen zu können. Hierzu gibt es zwei effiziente Implementationen. Eine setzt schnelle Ganzzahlmultiplikationen und -divisionen voraus. Mit der Ganzzahlmultiplikation verteilt man die Oktalziffern auf geeignete Stellen in einem großen Maschinenwort, dann maskiert man die Ziffern aus, die an passenden Stellen stehen und zum Schluss fasst man mit einer Di-

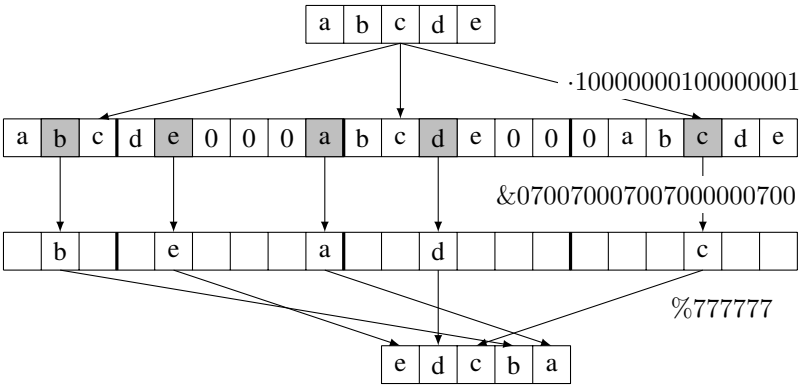

Abbildung 9: Verteilen, Auswählen und Zusammenführen von Oktalziffern für die Umkehrung einer Grenze der Breite 5

vision mit Rest die verteilten Ziffern wieder zu einer Oktalzahl zusammen.[5] Auf einem 64-Bit-Rechner können wir auf diese Weise maximal 5 Oktalziffern effizient spiegeln. Das entspricht 15 Bit. Abbildung 9 zeigt, wie die fünf Oktalziffern im 64-Bit-Wort vervielfältigt und welche davon für das Ergebnis ausgewählt werden. Hier der zugehörige Haskell-Code:

```
reverse5spread :: Word64 -> Word64
reverse5spread bits =
   mod
      ((bits * 0o10000000100000001)
```

[5] https://graphics.stanford.edu/~seander/bithacks.html

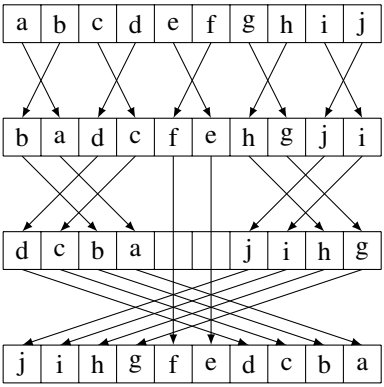

Abbildung 10: Oktalziffern schrittweise umordnen für die Umkehrung einer Grenze der Breite 10

```
      .&. 0o70070007007000000700)
    0o777777
```

Für eine Grenze der Breite 10 müssen wir die Funktion `reverse5spread` zweimal anwenden. Es zeigt sich aber, dass die Multiplikationen und Divisionen für 64 Bit auf einem x86_64-Prozessor vergleichsweise langsam sind. Die näher liegende Implementierung mit Hilfe von Maskieren und Verschieben wie in Abbildung 10 ist doch schneller:

```
swap :: Int -> Word32 -> Word32 -> Word32
swap n m bits =
   shiftL (bits .&. m) (n*3)  .|.
```

```
   shiftR bits (n*3) .&. m

reverse10up :: Word32 -> Word32
reverse10up bits0 =
   let bits1 = swap 1 0o0707070707 bits0
       bits2 = swap 2 0o0077000077 bits1
   in  bits1 .&. 0o0000770000
           .|. swap 6 0o0000007777 bits2
```

4.2.3 Externes Sortieren

Im Speicher stellen wir die temporären Daten dar als Liste mit Elementen vom Typ `(((Frontier, Fleet), Word64)`. Die Daten erreichen für ein 10×10-Brett einen Umfang, der zu groß für den Hauptspeicher werden kann. Wir müssen dann zum Verarbeiten der Daten auf die Festplatte ausweichen. Im Wesentlichen geht es darum, alle Paare mit der gleichen Kombination aus Grenze und Flotte zusammenzufassen und deren Zähler zu addieren. Dazu zerteilen wir die durch Bedarfsauswertung erzeugte Liste in Stücke, die sich im Hauptspeicher sortieren und zusammenfassen lassen. Für das Sortieren nutzen wir den Typen `Map` aus der GHC-Grundbibliothek `containers`, einen binären Suchbaum. Die Stücke wandeln wir in kompakte Felder bestehend aus Elementen mit 128-Bit (32+32+64) um, die wir als Dateien auf die Festplatte schreiben. Dann verknüpfen wir die Dateien mit einer Art Merge-Sort.

5 Alternative Lösungsverfahren

Das in Abschnitt 3 vorgestellte Verfahren lässt sich als wiederholte Matrix-Vektor-Multiplikation mit einer schwach besetzten Matrix der Dimension 1 333 747 dar-

stellen.[6] Allerdings ist unser Vektor am Anfang noch schwach besetzt. Erst ab der Brettgröße 10×8 treten überhaupt alle Kombinationen aus Grenze und Flotte auf, d.h. erst dann ist der Vektor voll besetzt. Unsere Implementierung stellt gewissermaßen nur denjenigen Teil der Matrix auf, der für die besetzten Vektorelemente nötig ist.

Kann man aus der Matrix-Darstellung weitere Erkenntnisse gewinnen? Eher kann man aus unseren Abschätzungen in 3 etwas über die Eigenschaften der Matrix lernen. Da sich die Anzahl der Möglichkeiten mit einem Polynom vom Grad 10 abschätzen lässt, muss die Matrix für den größten Eigenwert einen Jordanblock der Größe 10 besitzen.

Da es, wie wir nun wissen, nur 26 Billionen mögliche Aufstellungen gibt, könnten wir vermutlich mit maschinennaher Bitmanipulation auch alle Aufstellungen in erträglicher Zeit durchprobieren.

6 Verwandte Probleme

6.1 Sich berührende Schiffe

In einer Spielvariante dürfen sich die Schiffe berühren. Der Induktionsansatz aus Abschnitt 3 lässt sich auch auf diese Variante übertragen. Allerdings gibt es im Detail Unterschiede:

- Ein waagerechtes Schiff blockiert keine Schiffe in der nächsten Zeile. Es gibt also an der Grenze kein ■. Das verringert die Anzahl der möglichen Grenzen.

[6] Diese Zahl ist übrigens kein Vielfaches von 120, der Anzahl der Teilflotten der Standardflotte. Das ist so, weil nicht jede Flotte jede Grenzsituation erzeugen kann. Beispielsweise kann eine Flotte bestehend aus nur einem 2er-Schiff keine vollbesetzte Grenze erzeugen.

- Statt höchstens 5 nebeneinander angeordneten senkrecht ausgerichteten Schiffen kann es jetzt bis zu 10 geben. Das steigert die Anzahl der möglichen Grenzen enorm.

Im Prinzip lässt sich die Spielform mit Zwischenräumen auf diejenige ohne Zwischenräume zurückführen, wenn man Schiffe mit Breite 2 zulässt. Dazu muss man nur alle Schiffe und das Spielfeld um jeweils einen zusätzlichen Streifen rechts und unten erweitern. Die Aufstellung aus Abbildung 1 würde dann zur Aufstellung wie in Abbildung 11 werden.

Dennoch könnte man nicht den gleichen Algorithmus für beide Varianten verwenden, denn sowohl waagerechte wie senkrechte Schiffe der Breite 2 bräuchten eine gesonderte Behandlung. Wenn man weiß, dass alle Schiffe die Breite 2 haben, könnte man als Optimierung verwenden, dass in einen Einerzwischenraum kein weiteres Schiff passt. Dieses Wissen ist bereits in unseren Algorithmus für die Spielvariante mit Zwischenräumen eingebaut.

6.2 Wahrscheinlichkeitsverteilung auf dem Spielbrett

Für einen Spieler wäre es bei einem Angriff von großem Vorteil, zu wissen, an welchen Stellen sich wahrscheinlich ein Schiff des Gegners befindet. Auch ein Algorithmus für einen Computerspieler könnte dieses Wissen zu seinem Vorteil einsetzen. Wenn allerdings beide Spieler von solchen Wahrscheinlichkeitsverteilungen wissen, dann verliert dieses Wissen seinen Wert, denn dann kann der Gegner bei seiner Aufstellung die üblicherweise häufig belegten Felder meiden.

Bei der Suche nach einer Wahrscheinlichkeitsverteilung auf dem Spielfeld stellt sich das grundlegende Problem, dass wir nichts über die Wahrscheinlichkeitsverteilung der Schiffsaufstellungen wissen. Ich will im Folgenden von zwei verschiedene Annahmen über diese Verteilung ausgehen.

Beim ersten Ansatz simuliere ich das Verhalten eines menschlichen Spielers,

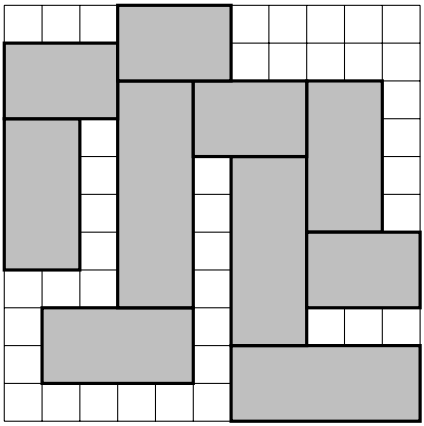

Abbildung 11: Aufstellung aus Abbildung 1 überführt in eine äquivalente Aufstellung ohne Zwischenräume

der zuerst gleichverteilt zufällig das größte Schiff plaziert, dann das nächstkleinere gleichverteilt zufällig auf dem verbleibenden Platz usw. Mit einem Monte-Carlo-Ansatz benötige ich 50 Millionen Versuche, damit die relativen Häufigkeiten maximal um 0,001 von symmetrisch liegenden Feldern abweichen. Ich erhalte dabei eine Verteilung wie in Abbildung 12 dargestellt.

Die Verteilung ändert sich schon, wenn ich die Reihenfolge der Schiffe ändere. Auch die Berechnung wird deutlich aufwändiger, denn wenn ich mit kleinen Schiffen beginne, ist die Gefahr größer, dass am Ende für das größte Schiff kein Platz mehr frei ist. Dort verwende ich nur 1 Million Versuche.

Beim zweiten Ansatz gehe ich von einer Gleichverteilung aller möglichen Aufstellungen aus. Die sich dabei ergebenden Häufigkeiten kann ich mit dem induktiven Ansatz aus Abschnitt 3 exakt berechnen. Dazu muss ich nur die Anzahl der Möglichkeiten durch die Häufigkeiten auf dem Feld ergänzen. Mathematisch ist das eine einfache Erweiterung, rechentechnisch bedeutet es aber einen hundertfachen Speicherbedarf. Beachten muss man, dass man die von Schiffen belegten Felder zählt und nicht die durch Schiffe blockierten Felder, wie ich es in Unterabschnitt 4.1 eingeführt habe. Auf diese Weise erhalte ich die Häufigkeitsverteilung in Abbildung 13.

Wir stellen fest, dass alle Verteilungen im Detail verschieden aussehen. Insbesondere sind bei der letzten Verteilung die Unterschiede zwischen minimaler und maximaler Häufigkeit deutlich größer. Wir können daraus schlussfolgern, dass beide Ansätze unterschiedliche Verteilungen von Aufstellungen beschreiben. Ich vermute aber, dass der erste eher dem Vorgehen eines menschlichen Spielers entspricht.

Allen Verteilungen ist gemein, dass die Randfelder am häufigsten belegt werden und die Eckfelder weniger häufig, dass sich im Inneren ein quadratischer 8×8-Ring mit besonders geringer Häufigkeit und in der Mitte ein 6×6-Quadrat mittlerer Häufigkeit befindet. Man tut also als Spieler gut daran, zuerst auf Felder am Rand zu zielen und danach auf Felder im inneren 6×6-Quadrat.

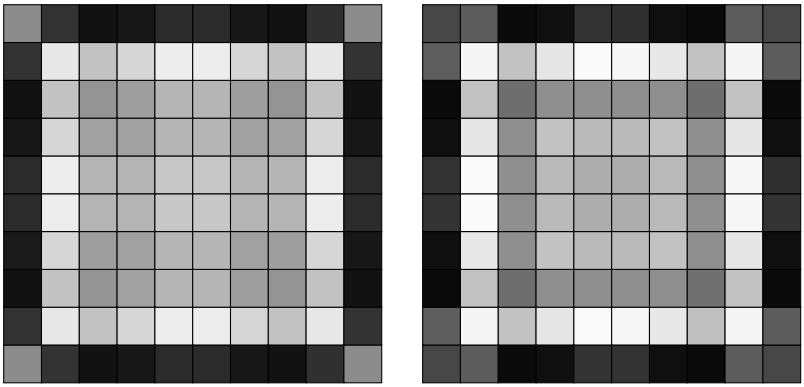

Abbildung 12: Relative Häufigkeiten für die Belegung von Feldern bei einem Monte-Carlo-Ansatz: Links: Die Schiffe werden in absteigender Größe plaziert. Weiß entspricht 0,25; Schwarz entspricht 0,35. Rechts: Die Schiffe werden in aufsteigender Größe plaziert. Weiß entspricht 0,17; Schwarz entspricht 0,45.

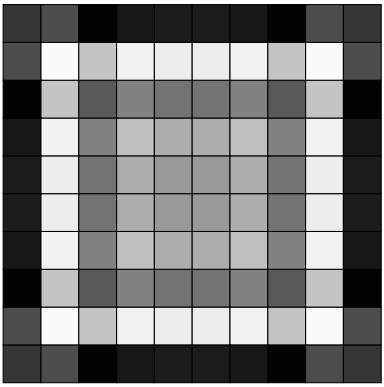

Abbildung 13: Relative Häufigkeiten für die Belegung von Feldern bei Annahme von Gleichverteilung aller möglichen Aufstellungen: Weiß entspricht 0,3; Schwarz entspricht 0,9.

A Ergebnisse

In Tabelle 1 zeige ich die Anzahl der möglichen Aufstellungen auf einem 10×10-Spielfeld für eine Auswahl an Flotten, bei denen sich die Schiffe nicht berühren dürfen. Auf StackExchange[7] wurde auch nach der Anzahl der Möglichkeiten für eine Variante des Spieles gefragt, bei der sich die Schiffe berühren dürfen, es aber weniger Schiffe gibt und das Spielfeld kleiner ist (8×8). Tabelle 2 zeigt auch für diese Variante die Anzahl der Aufstellungen für verschiedene Flottengrößen.

[7] http://math.stackexchange.com/questions/58769/battleship

Längen			Anzahl von Zweierschiffen				
5	4	3	0	1	2	3	4
			1	180	13 952	614 372	17 086 631
		1	160	23 864	1 512 096	53 606 768	1 179 864 432
		2	10 124	1 230 092	62 495 628	1 745 117 572	29 650 630 088
		3	330 840	32 094 960	1 278 079 616	27 395 120 928	348 892 008 792
	1		140	20 072	1 219 008	41 283 320	864 647 388
	1	1	16 896	1 966 664	95 386 968	2 532 545 416	40 721 312 760
	1	2	786 968	72 831 964	2 755 256 320	55 828 223 540	668 256 987 440
	1	3	18 382 184	1 320 017 704	37 880 082 192	567 343 768 064	4 871 721 743 520
	2		6 996	779 752	36 085 692	910 464 972	13 847 185 918
	2	1	619 136	54 625 952	1 961 767 760	37 549 258 992	422 118 615 240
	2	2	20 505 284	1 396 400 720	37 804 399 592	530 879 625 444	4 242 772 005 768
	2	3	328 511 112	16 757 986 408	330 360 758 520	3 268 821 509 016	17 707 810 153 184
1			120	16 528	961 464	31 086 744	619 268 128
1		1	13 808	1 538 520	71 194 112	1 796 509 352	27 329 683 408
1		2	610 688	53 879 808	1 935 562 696	37 068 246 456	417 017 581 256
1		3	13 477 504	918 108 728	24 872 048 856	349 608 204 104	2 797 606 827 136
1	1		11 360	1 211 008	53 413 832	1 279 310 984	18 385 007 248
1	1	1	954 544	80 194 520	2 731 237 000	49 341 382 288	520 599 890 368
1	1	2	29 852 632	1 925 751 392	49 143 133 528	646 677 411 848	4 808 897 160 408
1	1	3	448 574 416	21 536 631 136	397 108 468 832	3 647 432 090 016	18 170 989 432 520
1	2		371 048	29 660 864	957 045 912	16 298 373 712	161 138 941 144
1	2	1	21 906 376	1 337 029 984	32 111 473 656	395 146 057 864	2 726 362 307 664
1	2	2	462 924 032	20 886 349 672	359 461 539 224	3 056 118 386 176	13 949 906 853 080
1	2	3	4 488 051 160	143 891 605 904	1 693 472 099 056	9 404 689 335 808	26 509 655 816 984

Tabelle 1: Anzahl Aufstellungen auf einem 10×10-Brett, Schiffe dürfen sich nicht berühren. In den linken drei Spalten stehen die Anzahlen der Schiffe mit den Längen 3-5.

Längen			Anzahl von Zweierschiffen				
5	4	3	0	1	2	3	4
			1	112	5 924	196 916	4 618 099
		1	96	9 896	480 384	14 610 448	312 491 752
		2	4 092	386 684	17 153 616	475 161 604	9 222 676 444
		3	102 744	8 861 336	357 531 992	8 974 172 752	157 199 292 416
	1		80	8 032	379 416	11 218 712	233 038 372
	1	1	6 576	604 584	26 067 936	701 107 544	13 197 612 600
	1	2	238 632	19 999 324	783 249 968	19 060 435 960	323 279 127 972
	1	3	5 068 400	385 267 768	13 630 527 520	298 346 692 624	4 529 859 007 656
	2		2 616	233 976	9 804 528	256 000 192	4 672 780 776
	2	1	183 072	14 907 192	566 605 600	13 365 380 928	219 437 755 336
	2	2	5 618 424	414 346 108	14 204 172 008	300 827 818 472	4 412 683 088 528
	2	3	100 165 608	6 651 115 200	204 328 789 840	3 858 124 586 760	50 170 227 411 520
1			64	6 256	287 472	8 260 856	166 599 688
1		1	5 072	453 528	19 000 600	496 028 168	9 052 849 616
1		2	177 344	14 438 392	548 719 744	12 942 460 344	212 486 920 032
1		3	3 626 480	267 421 536	9 167 065 880	194 148 133 872	2 847 991 701 064
1	1		4 000	347 920	14 164 168	358 916 040	6 350 436 776
1	1	1	270 000	21 354 072	787 447 784	17 998 784 464	285 958 820 800
1	1	2	7 984 992	571 126 760	18 964 189 704	388 477 085 496	5 502 878 939 936
1	1	3	137 007 920	8 808 033 472	261 588 795 232	4 766 987 266 368	59 714 361 918 560
1	2		102 240	7 853 360	280 927 120	6 220 700 184	95 607 277 488
1	2	1	5 833 360	404 574 056	13 008 315 336	257 637 328 288	3 522 498 833 168
1	2	2	144 604 696	8 997 481 576	258 206 849 656	4 538 580 803 200	54 728 030 543 744
1	2	3	2 059 524 032	114 123 399 112	2 899 461 695 864	44 824 464 971 792	471 929 328 258 128

Tabelle 2: Anzahl Aufstellungen auf einem 8×8-Brett, Schiffe dürfen sich berühren.