Softwarepraktikum SS 2018

Assignment 4

Group - 3

Ramil Sabirov	369500	ramil.sabirov@rwth-aachen.de
Joel Choi	345575	joel.choi@rwth-aachen.de
Eric Remigius	366895	eric.remigius@rwth-aachen.de

Task 1

In unserer Implementierung der Vorsortierung der Züge nutzt der PruningParanoidCalculator das Interface MoveSorter, welche die Methode *sortMoves*() stellt. Wir haben bisher zwei konkrete Implementierungen dieses Interfaces. Einmal den BogoSorter und einmal den NaturalSorter (siehe Abbildung 1).

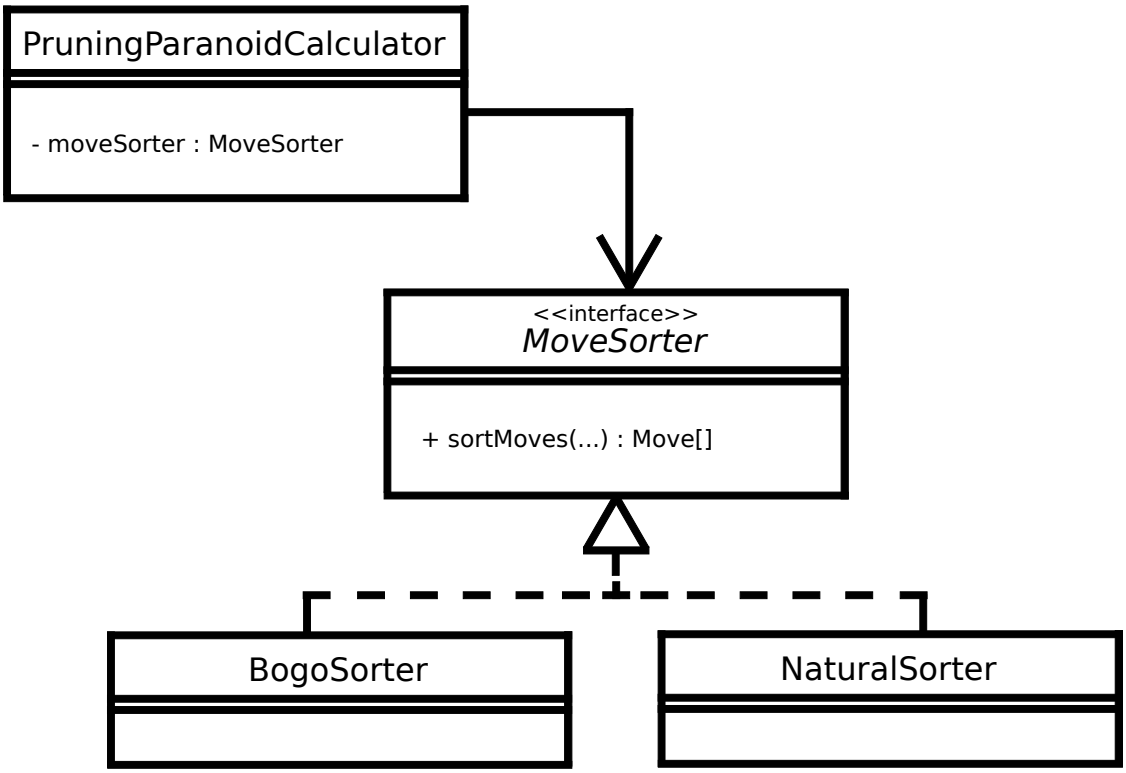

Figure 1: Klassendiagramm für das MoveSorting

BogoSorter

Der Name ist Programm. Der BogoSorter sortiert die Züge nicht, sondern konvertiert einfach die Datenstruktur. Dementsprechend ist die Implementierung nicht besonders spannend.

NaturalSorter

Der NaturalSorter sortiert die Züge entsprechend der "natürlichen" Güte der Züge im Hinblick auf unsere Bewertungsfuntkion. Diese behandelt z.B. Override-Stones als Joker und platziert sie ungern. Demnach werden diese Züge ans Ende sortiert. Im Gegensatz dazu, werden Züge auf Bonusfelder, welche die Möglichkeit geben einen weiteren Override-Stein zu bekommen, an den Anfang der Datenstruktur sortiert. Die vollständige Reihenfolge der Sortierung ist in der **Buildingphase**:

$$BONUS_{OVERRIDE} > CHOICE > BONUS_{BOMB} > INVERSION > NORMAL > OVERRIDEUSE > OVERRIDEUSE_{SELF}$$

wobei $OVERRIDEUSE_{SELF}$ einen Zug bezeichnet, welcher sowohl einen Override-Stein nutzt als auch einen eigenen Stein überschreibt. Und in der **BombingPhase**:

$$NORMAL > BOMBING_{SELF}$$

Die undifferenzierte Zugsortierung in der BombingPhase ist dadurch zu erklären, dass

1. wir das Pruning und eine große Tiefe in dieser Spielphase nicht als besonders wichtig erachten.
2. eine feinere Sortierung zusätzliche Berechnungen zur Folge hätte, da wir beim Generieren der möglichen Züge in dieser Phase nicht besonders viel Information bekommen.

Task 2

Die Abbildungen 2 und 3 zeigen die durchschnittlichen Bedenkzeiten pro Zug auf zehn verschiedenen Maps. Die Daten wurden erhoben indem jeweils ein gesamtes Spiel auf der Map gespielt wurde mit eingeschalteten Performance-Logging und anschließend der ausgegebene Log geparsed wurde.

Es war zu erwarten, dass die durchschnittliche Bedenkzeit beim Alpha-Beta Al-

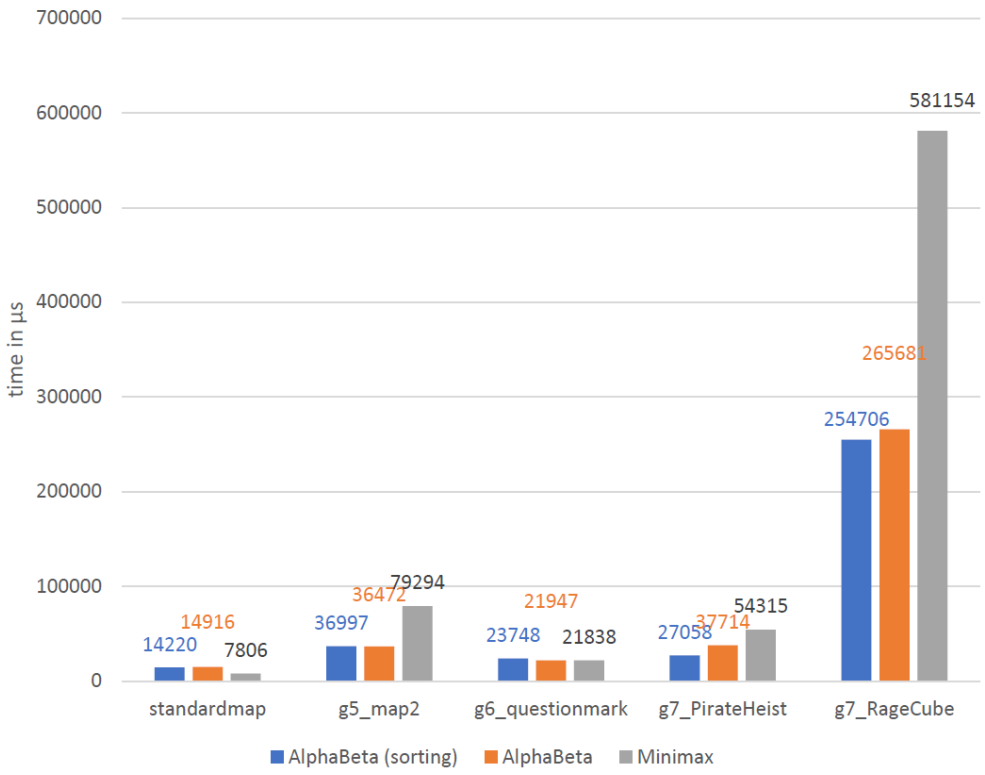

Figure 2: Durchschnittszeit bei Tiefe 2 (1/2)

gorithmus mit Zugsortierung stets geringer ist, als die ohne Sortierung. Leider entsprechen unsere Messungen nicht ganz den Erwartungen. Die Bedenkzeiten liegen sehr oft auf gleicher Höhe. Eine enorme Verbesserung erzielen wir nur auf der 2-Spieler Random-Map sowie auf der Map3 unserer Gruppe. Besonders erschreckend ist das Messergebnis auf der Standardmap sowie auf g6-questionmark. Dort scheint der Minimax **schneller** zu sein als der Alpha-Beta und die Zugsortierung ist eine Einbuße von Performance. Auf der Suche nach einer Erklärung sind wir auf folgende Punkte gestoßen:

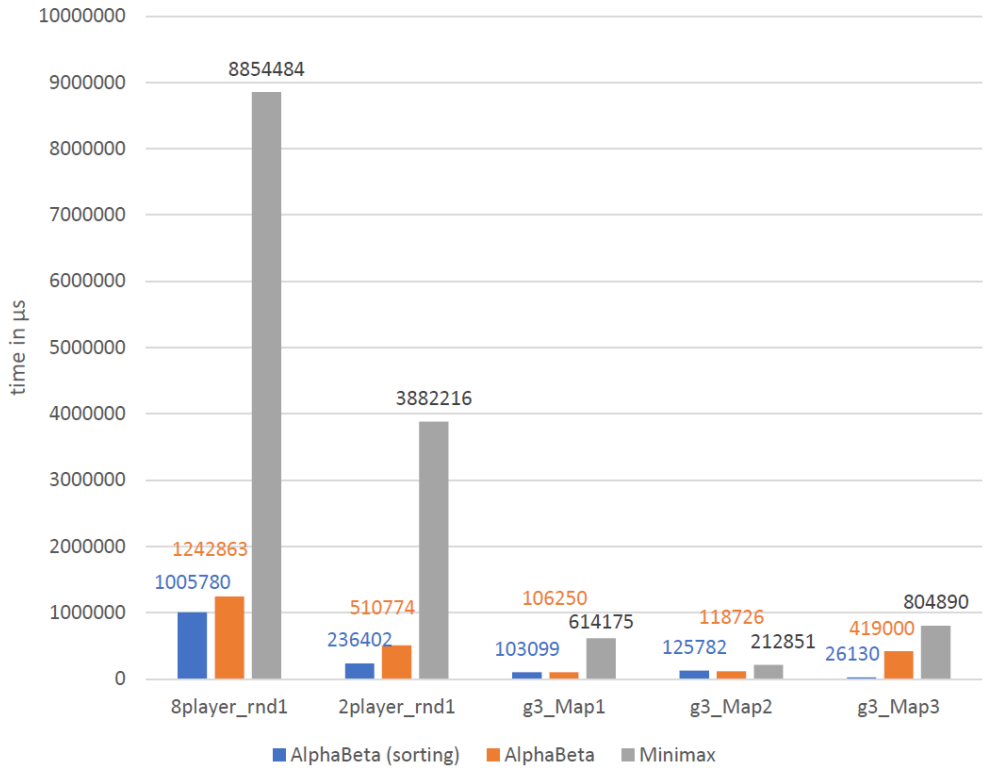

Figure 3: Durchschnittszeit bei Tiefe 2 (2/2)

1. **Die verschiedenen Suchverfahren hatten keine identischen Spiele:** Das passiert durch eine unterschiedliche Zugsortierung und bei gleichwertigen Zügen bzgl. der Bewertungsfunktion. Ein unterschiedlicher Spielverlauf kann unterschiedliche Komplexität mit sich führen und somit eine unterschiedliche Durchschnittszeit. Jedoch ist es sehr unwahrscheinlich, dass dieser Einfluss die Bedenkzeit des Alpha-Beta Algorithmus über die des MiniMax bringt.

2. **Fehler in der Datenerhebung:** Wir können nicht komplett ausschließen, dass die Messung der Performance unentdeckte Fehler beherbergt.

3. **Fehler in der Implementierung:** Der Vollständigkeit halber sei dieser Punkt ebenfalls aufgeführt, wobei einige nachträgliche Tests darauf hindeuten, dass die Implementierung in Ordnung ist.

Da unsere Zugsortierung Züge mit speziellen Effekten ordnet (z.B. Override-uses oder Bonus-Felder), verliert sie jegliche Wirkung auf Karten, auf denen besonders

wenige Spezial-Felder oder Override-stones zur Verfügung stehen. Dennoch sollte sie insgesamt eine größere Auswirkung haben.

Task 3

Vorab erwähnt: Wir haben sowohl den Alpha-Beta Algorithmus als auch den Min-Max Algorithmus angepasst, dass dieser beim Überschreiten einer Zeitvorgabe sofort terminiert und den TimeOut auch im Rückgabewert kenntlich macht. Dafür haben wir ausgenutzt, dass unsere Bewertungsfunktion im Wertebereich $(0, 100)$ liegt. Den negativen Werten konnten wir dementsprechend eine besondere Bedeutung geben.
Nun kommen wir zum Iterative Deepening Algorithmus, in diesem Report als 'Deepener' bezeichnet. Der Deepener ist implementiert in der Klasse IterativeDeepeningCalculator, welcher selber keine Kalkulation durchführt sondern diese Arbeit an einen anderen Calculator delegiert.

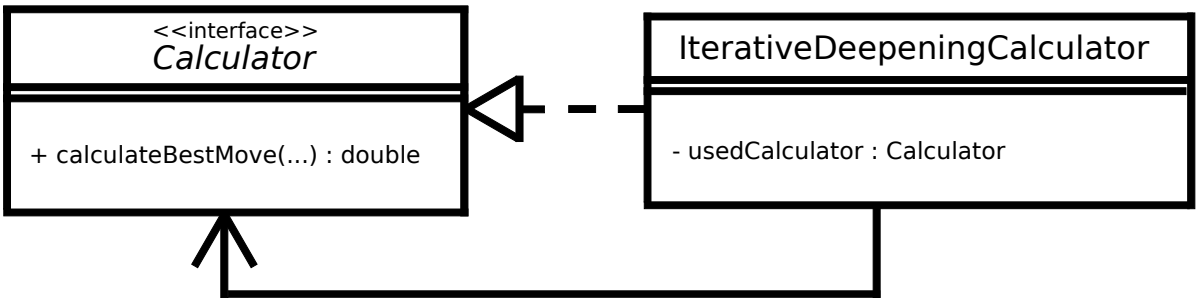

Figure 4: Klassendiagramm von IterativeDeepeningCalculator

Dabei erhält jeder Calculator ein "Formular" (Klasse CalculatorForm), welcher er beim Aufruf von $calculateBestMove()$ ausfüllen muss. Diese Informationen sind als Schnittstelle zu den aufrufenden Klassen gedacht. In dem Formular wird momentan der beste Zug, der maximale Verzweigungsgrad und ob der Algorithmus bis zum Ende der Spielphase gerechnet hat, hinterlegt. Diese Informationen nutzt der Deepener um effizienter mit der Bedenkzeit umzugehen. Seine Arbeitsweise ist in Abbildung 3 als Flussdiagramm gezeigt.

Für die Zeitabschätzung der nächsten Tiefe der maximale Verzweigungsgrad k_{max} genutzt. Da die meiste Zeit der Kalkulation in den Blättern steckt und deren Anzahl sich in einem gleichmäßigen Variantenbaum um den Verzweigungsgrad vervielfacht ist die geschätzte Zeit für Tiefe $d + 1$:

$$t(d + 1) = t(d) * k_{max}$$

Dies ist eine einfache aber unter Umständen sehr konservative Abschätzung, gerade weil der Verzweigungsgrad nicht die Anzahl besuchter Teilbäume beschreibt (im

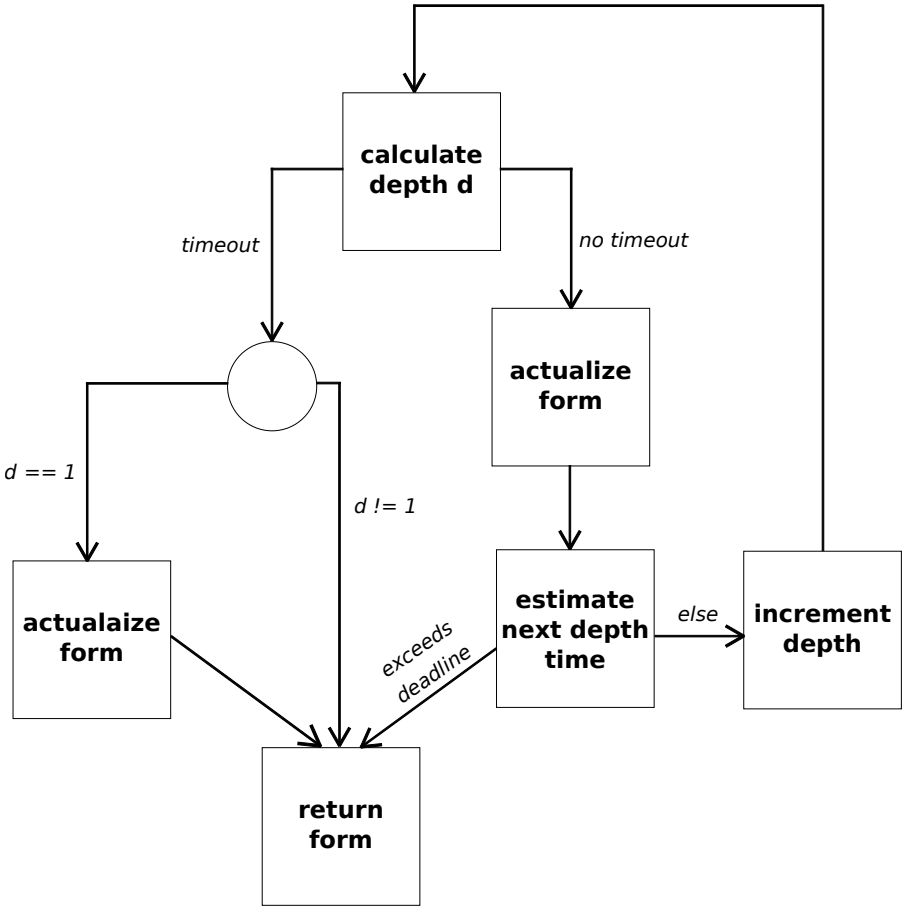

Figure 5: Programmfluss vom Deepener

Alpha-Beta können diese teilweise abgeschnitten werden). Außerdem wird außer Acht gelassen, dass man mit den Ergebnissen des Deepeners aus der vorhergehenden Tiefe eine bessere Zugsortierung machen kann um den Effekt des Alpha-Beta-Prunings zu verstärken und somit denselben Variantenbaum in kürzerer Zeit auskundschaften zu können.