Dokumentation Projekt „2048“

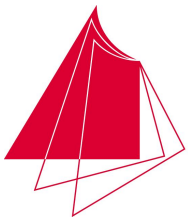
Algorithmen und Datenstrukturen

Kevin Kettinger (45701)

Janko Varga (44431)

Steffen Kappler (45472)

Stand: 16.02.2017



Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Inhaltsverzeichnis

1 Aufgabenstellung....3
2 „2048“ Spielprinzip....3
3 Realisierung und Funktionsumfang....3
3.1 Funktion „New Game“....3
3.2 Funktion „Load Game“....3
3.3 Funktion „Solver“....4
4 Lösungsalgorithmen....4
4.1 Zufälliger Algorithmus....4
4.2 Right Down Algorithmus....5
4.3 Pure Monte Carlo Algorithmus....6
5 Benchmark....6
5.1 Zufälliger Algorithmus....7
5.2 Right Down Algorithmus....8
5.3 Pure Monte Carlo Algorithmus....9
5.4 Gegenüberstellung....10
5.4.1 Vergleich Zyklus zu Punkten....10
5.4.2 Vergleich Punkte und Spielzüge....11

1 Aufgabenstellung

Für das Spiel „2048“ sollen mehrere Lösungsalgorithmen entwickelt und gegenüber gestellt werden.

2 „2048“ Spielprinzip

Das Spiel „2048“ wird auf einem quadratischen Spielfeld mit n-Kästchen gespielt, auf diesen sich Kacheln mit 2er Potenzen befinden. Diese können in alle vier Richtungen verschoben werden. Kacheln mit gleicher Potenz werden addiert. Nach jedem Zug wird eine neue Kachel mit dem Wert 2 (Wahrscheinlichkeit P=0.9) oder 4 (Wahrscheinlichkeit P=0.1). Zu Beginn werden zwei Kacheln erzeugt.

Ziel des Spieles ist das der Spieler eine Kachel mit dem Wert 2048 erreicht. Der Spieler verliert das Spiel, wenn die 2048 nicht erreicht hat und kein Zug mehr möglich ist. Hat der Spieler die 2048 erreicht, kann er trotzdem das Spiel fortsetzen.

3 Realisierung und Funktionsumfang

Das Spiel „2048“, so wie die Lösungsalgorithmen, wurden mit der Entwicklungsumgebung Qt Creator 4.1.0 basierend auf der plattformübergreifenden C++ Klassenbibliothek Qt 5.7.0 (MSVC 2013), zur grafischen Entwicklung von Benutzeroberflächen entwickelt.

Im folgenden sollen die einzelnen Menüpunkte der grafischen Oberfläche erläutert werden.

3.1 Funktion „New Game“

Mit dieser Funktion kann ein neues Spiel begonnen werden. Als Parameter kann die Dimension des Spielfeldes angegeben werden.

3.2 Funktion „Load Game“

Mit dieser Funktion können Spielstände geladen werden, um etwa ein vorheriges Spiel fortsetzen zu können oder eine bestimmte Problemstellung mit Hilfe einer der Algorithmen zu lösen.

Die Datei kann mit Hilfe eines einfachen Texteditors erstellt werden und als .txt Datei abgespeichert werden. Die einzelnen Datensätze müssen folgende Syntax aufweisen:

Spielfeldgröße: Die Größe des quadratischen Spielfeldes wird mit der Variablen *size* angegeben, nach dieser Variablen folgt die eigentliche Größe als Integer Zahl.

Belegung für die Kacheln: Die Datensätze für die Kacheln werden mit Hilfe ihrer Koordinaten x und y eingegeben, gefolgt durch ihren jeweiligen Zahlenwert der Kachel.

Kommentare: Kommentare können mit // begonnen werden und werden beim Einlesen ignoriert.

```
size 4;                 //Spielgroesse
x0, y0, z2048;          //Spieldaten
x0, y2, z2;
x1, y1, z2049;
x2, y0, z4096;
x3, y0, z128;
x3, y1, z1024;
```

3.3 Funktion „Solver“

Mit dieser Funktion kann ein Algorithmus ausgewählt werden und auf die aktuelle Spielsituation angewendet werden. Der Zeitabstand zwischen zwei Zügen kann eingestellt werden, sowie der Algorithmus der angewendet werden soll. Ebenfalls werden die Anzahl der bisher durchgeführten Züge dargestellt.

4 Lösungsalgorithmen

Zur Lösung bestimmter Problemstellungen stehen eine Auswahl verschiedener Algorithmen zur Verfügung. Diese sollen im folgenden beschrieben werden.

4.1 Zufälliger Algorithmus

Der zufällige Algorithmus wendet für jeden Zug eine neue zufällige Bewegungsrichtung an (rechts, links, unten oder oben). Das aktuelle Spielfeld wird nicht ausgewertet. Durch die einmalige Verwendung der Funktion „rand“ pro Zug ist die Laufzeit konstant.

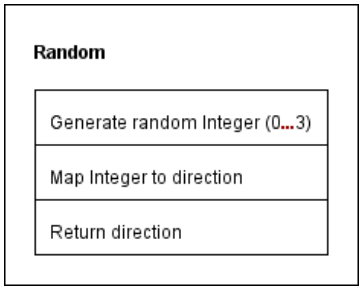


Abbildung 4.1: Struktogramm für Random Algorithmus

4.2 Right Down Algorithmus

Das Grundprinzip dieses Algorithmus ist es möglichst viele Kacheln in eine Ecke zu schieben, in diesem Fall nach rechts unten.

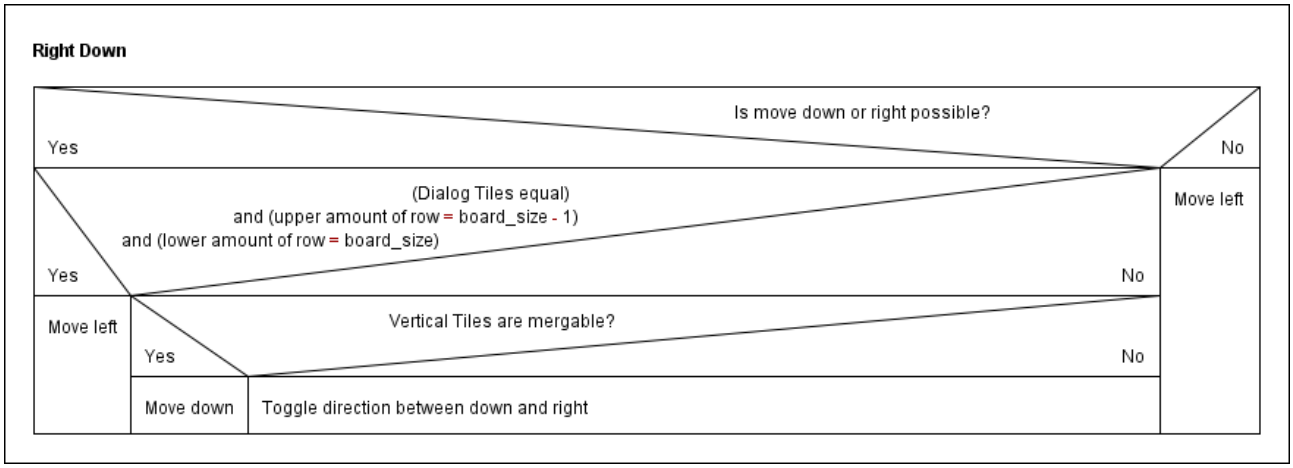


Abbildung 4.2: Struktogramm für Right Down Algorithmus

4.3 Pure Monte Carlo Algorithmus

Für jeden Zug wird für jede mögliche Richtung das Spiel *N*-mal zufällig zu Ende gespielt. Die Punktzahl für jede Richtung wird gemittelt und die Richtung mir der höchsten Punktzahl ist der nächste Zug.

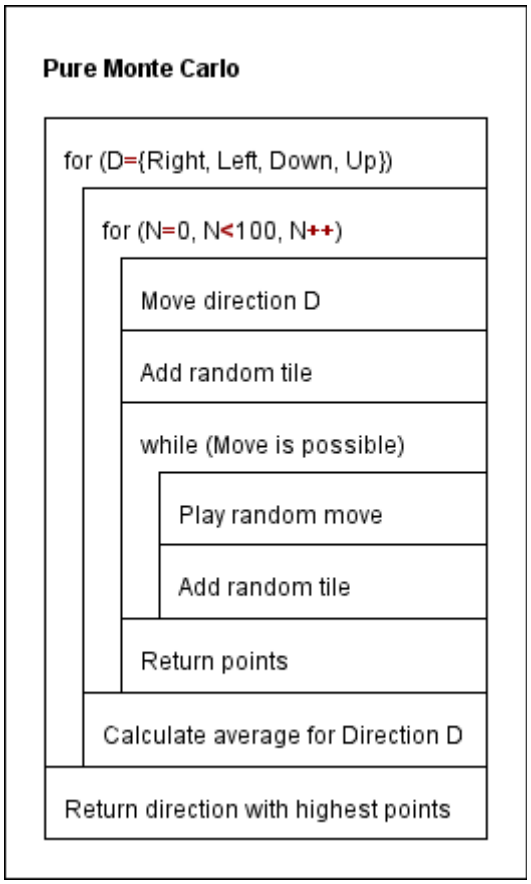


Abbildung 4.3: Struktogramm für Pure Monte Carlo Algorithmus

5 Benchmark

Im folgenden sollen die verschiedenen Algorithmen hinsichtlich ihres Berechnungsaufwandes und der maximal erreichbaren Punktzahl verglichen werden.

Zur Beurteilung des Berechnungsaufwandes wird die Anzahl an CPU-Zyklen innerhalb des Programms ermittelt. Hierbei wird nur der Algorithmus beurteilt und nicht andere Programmteile, wie zum Beispiel die grafische Oberfläche.

Zum Vergleich wurden die unterschiedlichen Algorithmen auf verschieden große Spielfelder jeweils drei Mal angewandt und die Ergebnisse gemittelt. Die Ergebnisse sind in den nachfolgenden Diagrammen dargestellt.

5.1 Zufälliger Algorithmus

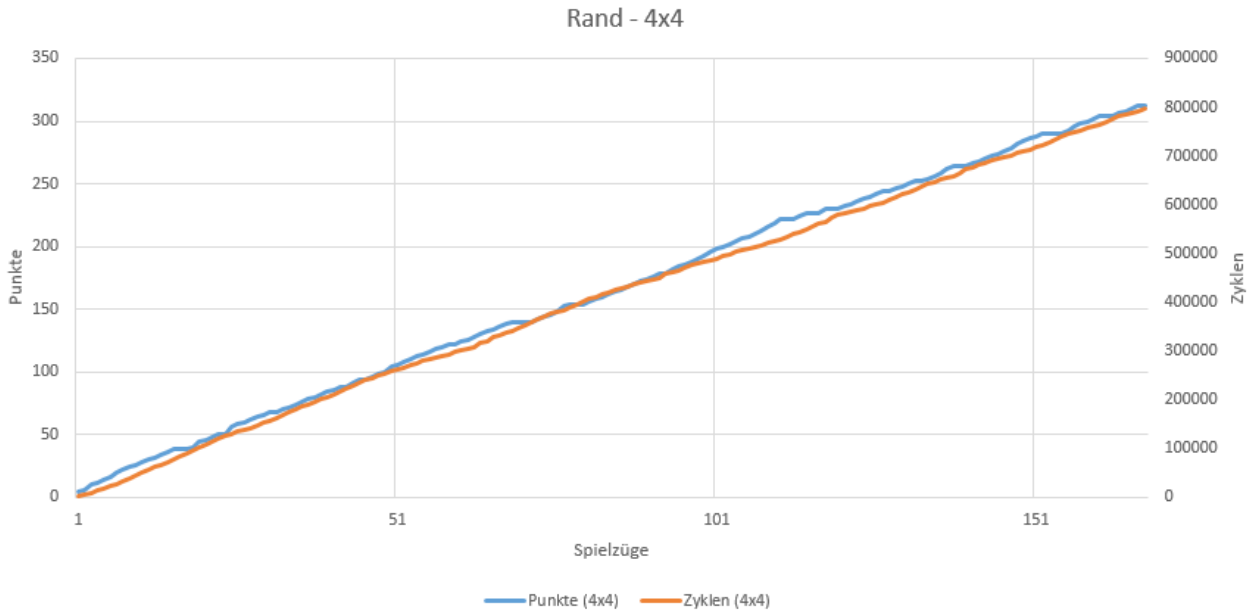


Abbildung 5.1: Zufälliger Algorithmus (4x4), Gegenüberstellung Zyklen und Punkte

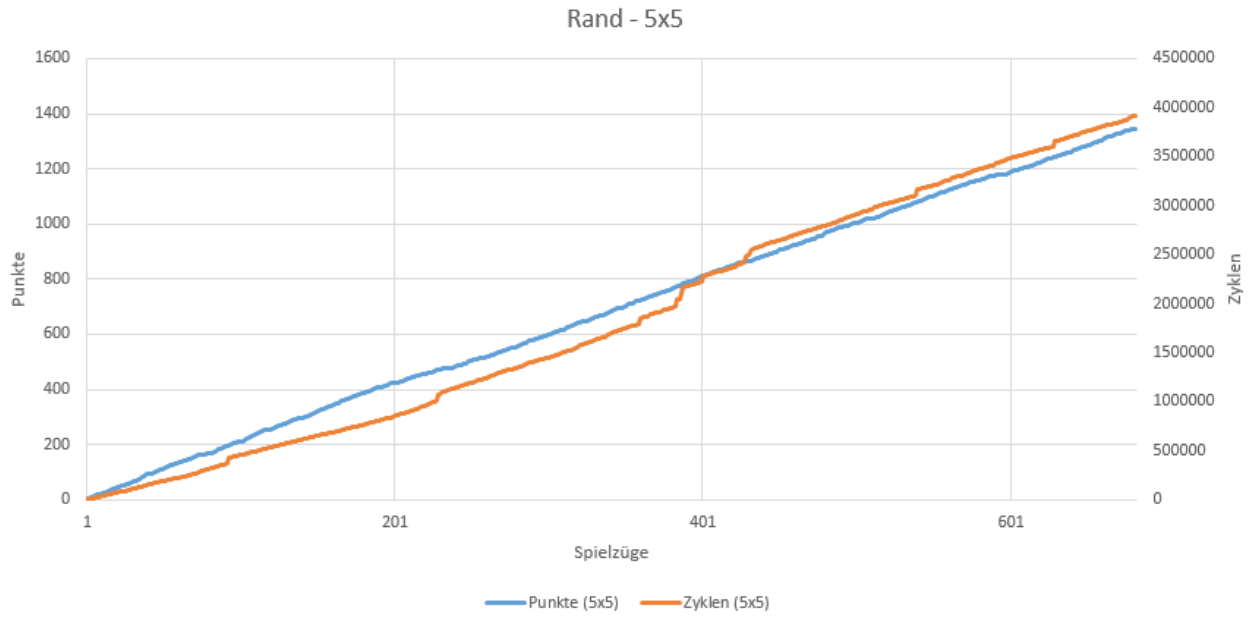


Abbildung 5.2: Zufälliger Algorithmus (5x5), Gegenüberstellung Zyklen und Punkte

5.2 Right Down Algorithmus

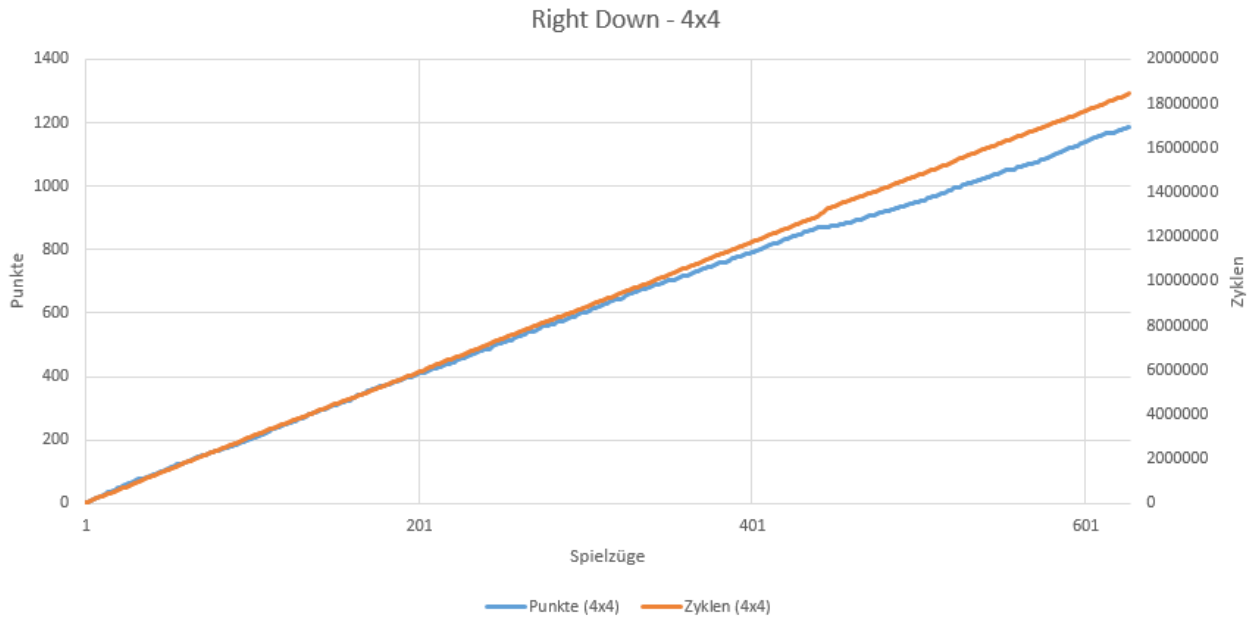


Abbildung 5.3: Right Down Algorithmus (4x4), Gegenüberstellung Zyklen und Punkte

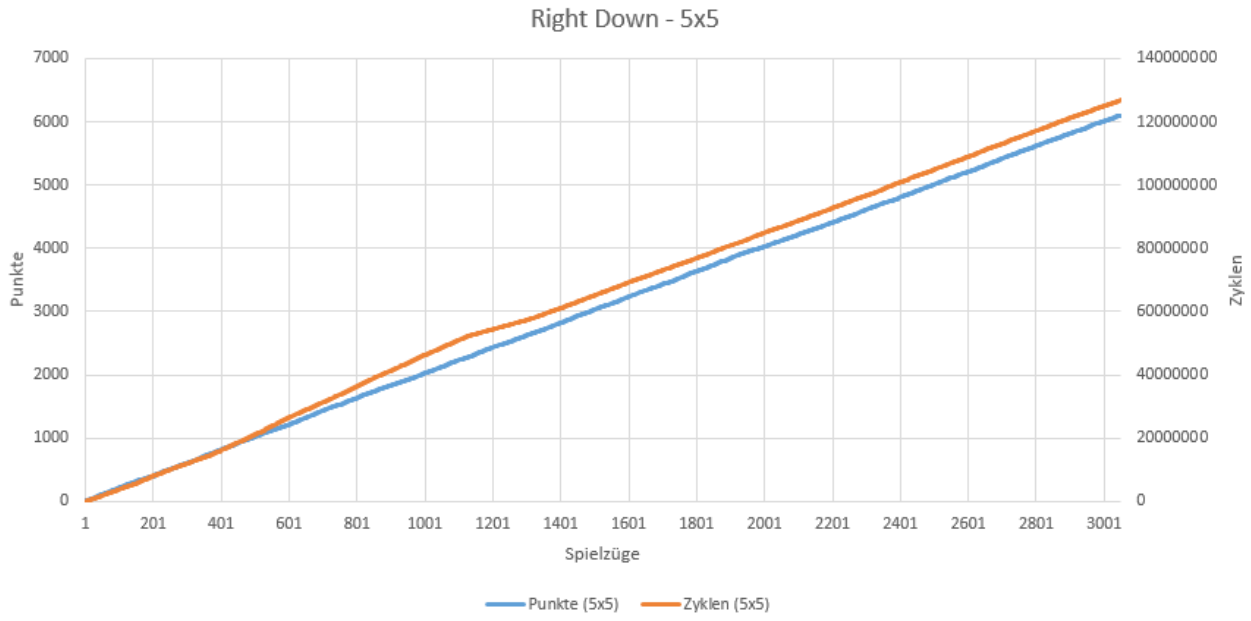


Abbildung 5.4: Right Down Algorithmus (5x5), Gegenüberstellung Zyklen und Punkte

5.3 Pure Monte Carlo Algorithmus

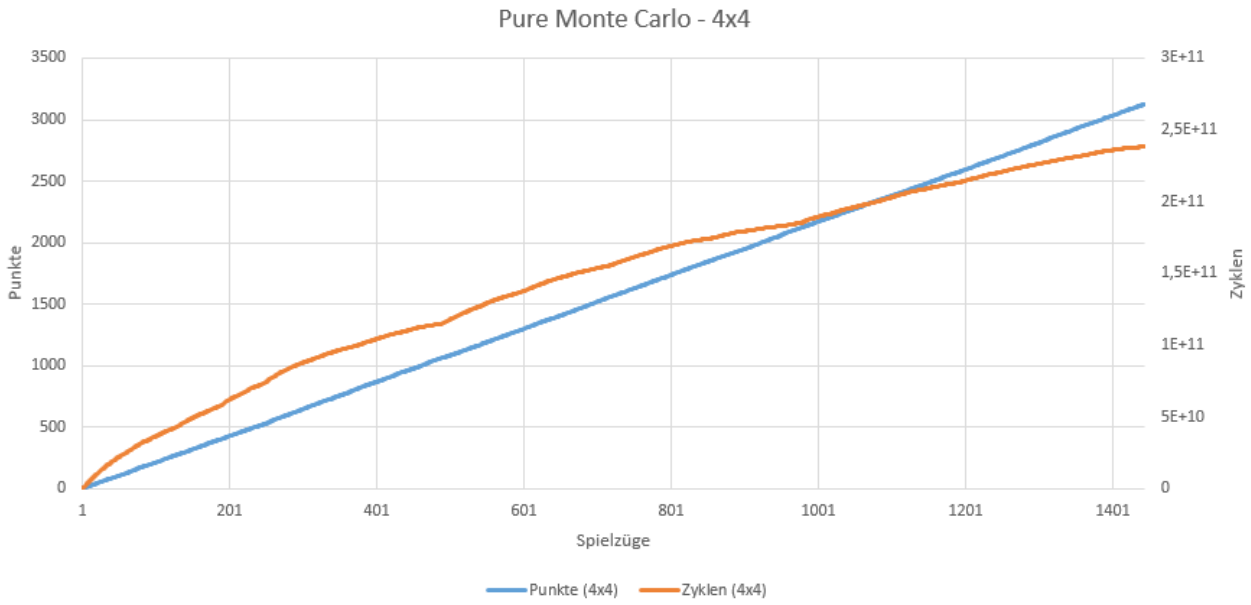


Abbildung 5.5: Pure Monte Carlo Algorithmus (4x4), Gegenüberstellung Zyklen und Punkte

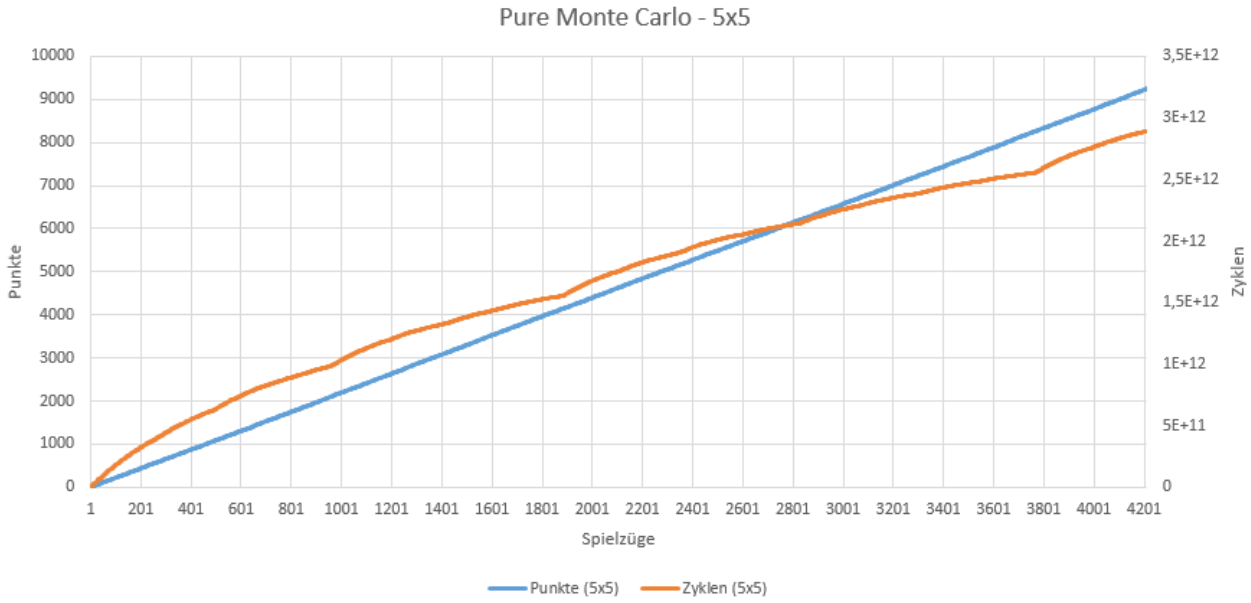


Abbildung 5.6: Pure Monte Carlo Algorithmus (5x5), Gegenüberstellung Zyklen und Punkte

5.4 Gegenüberstellung

In diesem Abschnitt werden die verschiedenen Algorithmen hinsichtlich ihrer Effektivität verglichen.

5.4.1 Vergleich Zyklus zu Punkten

In der Abbildung 5.7 sind die drei Algorithmen darstellt. In der horizontalen Achse ist die Punktzahl aufgetragen auf der vertikalen Achse die Anzahl der Zyklen. Jede Linie endet bei der maximal erreichten Punktzahl. Man erkennt das der Pure Monte Carlo Algorithmus deutlich mehr Zyklen gegenüber den Anderen benötigt, dafür aber eine deutlich höhere Endpunktzahl erreicht.

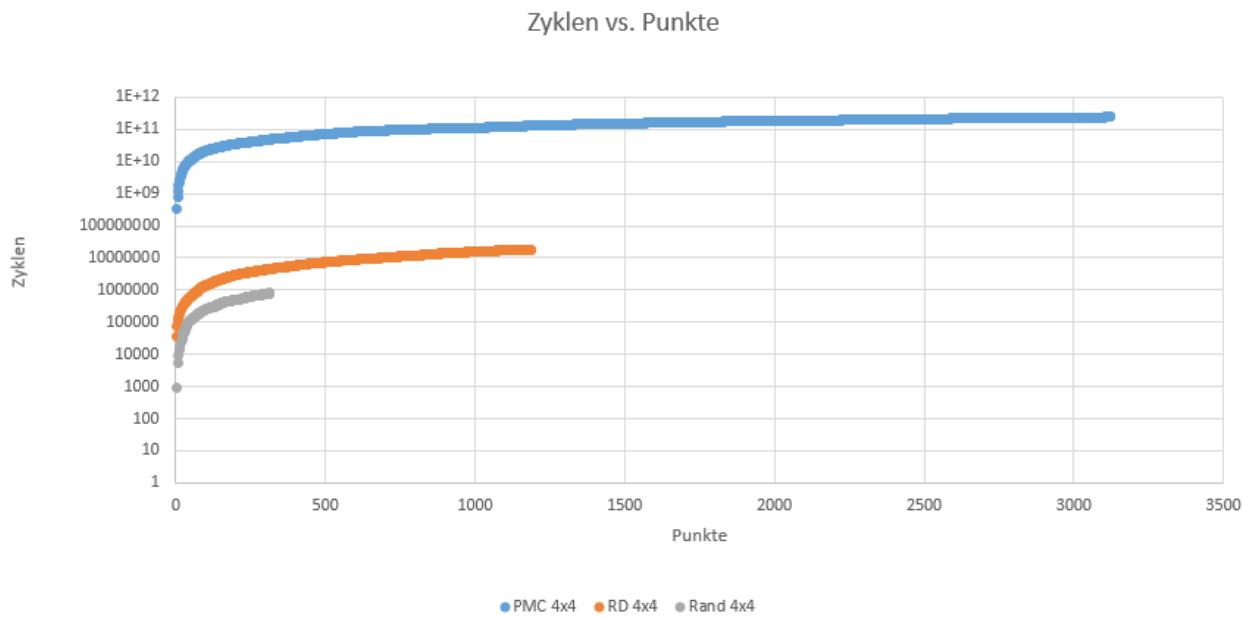


Abbildung 5.7: Gegenüberstellung Zyklen und Punkte

5.4.2 Vergleich Punkte und Spielzüge

In der Abbildung 5.8 sind die drei Algorithmen dargestellt. In der horizontalen Achse sind die Spielzüge aufgetragen auf der vertikalen Achse die Anzahl der Punkte. Jede Linie endet bei der maximal erreichten Punktzahl.

Zu Beginn des Spieles erhöht sich pro Zug die Punktzahl um zwei bzw. vier Punkte. Beim zufälligen und Pure Monte Carlo Algorithmus verläuft der Punkteanstieg linear, da vor jedem Zug überprüft wird ob es sich um eine gültigen Zahl handelt.

Bei dem Right Down Algorithmus können ungültige Spielzüge ausgeführt werden. Im Diagramm ist dieser Punkt ab 440 Punkten ersichtlich, ab diesem Punkt führt der Algorithmus ungültige Spielzüge aus.

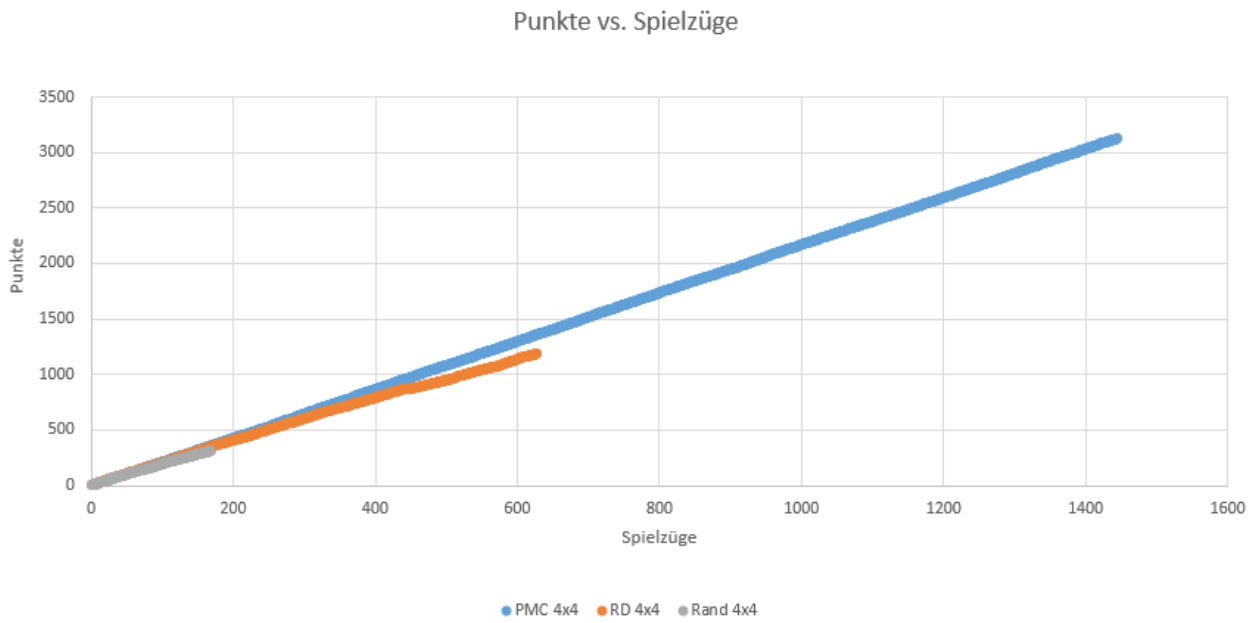


Abbildung 5.8: Gegenüberstellung Punkte und Spielzüge

Abbildungsverzeichnis

Abbildung 1:.....2
Abbildung 4.1: Struktogramm für Random Algorithmus.....4
Abbildung 4.2: Struktogramm für Right Down Algorithmus.....5
Abbildung 4.3: Struktogramm für Pure Monte Carlo Algorithmus.....6
Abbildung 5.1: Zufälliger Algorithmus (4x4), Gegenüberstellung Zyklen und Punkte.....7
Abbildung 5.2: Zufälliger Algorithmus (5x5), Gegenüberstellung Zyklen und Punkte.....7
Abbildung 5.3: Right Down Algorithmus (4x4), Gegenüberstellung Zyklen und Punkte.....8
Abbildung 5.4: Right Down Algorithmus (5x5), Gegenüberstellung Zyklen und Punkte.....8
Abbildung 5.5: Pure Monte Carlo Algorithmus (4x4), Gegenüberstellung Zyklen und Punkte.....9
Abbildung 5.6: Pure Monte Carlo Algorithmus (5x5), Gegenüberstellung Zyklen und Punkte.....9
Abbildung 5.7: Gegenüberstellung Zyklen und Punkte.....10
Abbildung 5.8: Gegenüberstellung Punkte und Spielzüge.....11