NORDAKADEMIE
HOCHSCHULE DER WIRTSCHAFT

Einführung in die KI - Hausarbeit

Matrikelnummer	8558, 8240
Thema	Performanceoptimierungen für die Suche optimaler Züge über Min-Max-Bäume
Studiengang, Zenturie	Angewandte Informatik, A17a

Inhaltsverzeichnis

1 Einführung **1**
1.1 Übertragbarkeit . . . 1

2 Repräsentation & Strategie **2**
2.1 Darstellung von Spielen durch Wurzelbäume . . . 2
2.2 Implementation von Strategie durch Exploration . . . 2
2.3 Rekursion . . . 2
2.3.1 Performance . . . 3

3 Methodik **3**
3.1 Analysierte Spiele . . . 4
3.1.1 Deterministische Strategiespiele . . . 4
3.2 Architektur . . . 5
3.3 Austauschbare Komponenten . . . 5
3.3.1 Caches . . . 5
3.3.2 Traversierungsstrategien . . . 5
3.3.3 Abbruchbedingungen . . . 6
3.3.4 Bewertung von Zügen . . . 6

4 Untersuchungsergebnisse **8**
4.1 Gewinnraten . . . 8
4.1.1 Tic-Tac-Toe & Vier Gewinnt . . . 8
4.1.2 Schach . . . 8
4.2 Verlustraten . . . 8
4.3 Zuganzahl . . . 8
4.4 Performance . . . 9
4.4.1 Anzahl der evaluierten Knoten . . . 9
4.4.2 Laufzeit . . . 9

5 Zusammenfassung **10**
5.1 Forschungsempfehlung . . . 10
5.2 Fazit . . . 10

6 Anhang **11**
6.1 Diagramme . . . 11
6.2 Literaturverzeichnis . . . 33

1 Einführung

Mit der Absicht einen Computerspieler für das klassische Brettspiel Go zu entwickeln, kreierte Googles DeepMind Team ein Programm namens AlphaGo, welches Entscheidungsbäume nutzt, um den bestmöglichen Zug in einer gegebenen Spielsituation zu finden. Bei Spielen wie Tic-Tac-Toe und Vier Gewinnt ist es aufgrund der kleinen Feldgröße und limitierten Anzahl in annehmbarer Zeit möglich den kompletten Baum zu berechnen und folglich den idealen Zug zu tätigen. Dies ist bei Spielen wie Go oder Schach auf einem normalen Brett aufgrund des exponentiellen Wachstums der möglichen Züge mit der steigenden Brettgröße nicht mehr möglich. Bei Schach zeigte Claude Shannon bereits 1950, dass es bei Schach mindestens 10^{120} mögliche Spiele gibt[7]. Dies ist selbst auf moderner Hardware nicht in einer annehmbaren Zeit zu berechnen. Entsprechend wird nach anderen Methoden geforscht. AlphaGo verfolgt den Ansatz die Entscheidung, ob ein Unterbaum exploriert werden soll und die Bewertung der einzelnen Unterbäume durch einen Machine Learning Algorithmus zu lösen[8]. Dieser Ansatz ist so erfolgreich, dass selbst der Weltmeister geschlagen werden konnte[4]. In dieser Arbeit werden mögliche Abbruchbedingungen zur Traversierung des Spielbaums, sowie Bewertungsalgorithmen für die entstehenden Züge analysiert mit dem Ziel eine Empfehlung auszusprechen, welcher der geprüften Algorithmen unter welchen Performance-Bedingungen das bestmögliche Ergebnis liefert.

1.1 Übertragbarkeit

Entscheidungsbäume wie sie bei z. B. Schach, Tic-Tac-Toe oder auch Vier Gewinnt Anwendung finden, lassen sich auch auf einige Probleme aus der realen Welt anwenden [6] und ein Algorithmus zur optimierten Entscheidungsfindung ließe sich potenziell auf diese übertragen. Dabei muss allerdings beachtet werden, dass in den meisten realen Situationen keine vollständige Kenntnis der Situation vorhanden ist, während dies in den meisten klassischen Brettspielen Spielen der Fall ist und zusätzlich meist nicht nur ein Gegenspieler teilnimmt, sondern mehrere. Dies limitiert die Übertragbarkeit. Ein weiterer Anwendungszweck für Computerspieler ist die Anwendung in modernen Computerspielen als Gegenspieler für Menschen. Sofern sich ein gegebenes Spiel als Entscheidungsbaum modellieren lässt, kann die Erkenntnis dieser Arbeit angewendet werden.

2 Repräsentation & Strategie

Um eine Strategie für Spiele umzusetzen, muss zunächst eine Abstraktion durchgeführt werden.

2.1 Darstellung von Spielen durch Wurzelbäume

Die erste Abstraktionsebene stellt die Repräsentation von Spielzuständen und Zügen in einem Wurzelbaum [6] dar. Alle möglichen Spielzustände werden als Knoten repräsentiert, während die möglichen Spielzüge durch Kanten dargestellt werden. Spielzustände die auf mehreren Wegen erreicht werden können, werden zur Vereinfachung der Implementation durch mehrere Knoten abgebildet. Ein Spiel lässt sich nur unter bestimmten Bedingungen als Baum darstellen. Auf diese wird in Sektion 3.1.1 näher eingegangen.

2.2 Implementation von Strategie durch Exploration

Die Umsetzung von Strategie wird durch die Exploration des zuvor erstellen Baums umgesetzt. Dies kann sowohl durch Breadth-First-Traversal als auch Depth-First-Traversal umgesetzt werden. Durch diese Methode wird das menschliche Verhalten bei Strategiespielen nachgebildet, um mögliche Situationen zu evaluieren und die bestmögliche zu wählen.

2.3 Rekursion

In einem formalisierten Spielbaum ist das erwartete Ergebnis eines Knotens:

- Der Knoten selbst, wenn er ein Blatt ist.
- Der Unterknoten mit dem besten erwarteten Ergebnis für den Spieler, der den nächsten Zug tätigt.

Dies ist umgesetzt durch:

```
private GameStateTreeNode<T> choice(GameStateTreeNode<T> start) {
    Set<GameStateTreeNode<T>> children = start.getChildren();

    //recursion anchor
    if (children.isEmpty()) {
        return start;
    }

    //recursion step
    GameStateTreeNode<T> result = children.stream()
            .peek(this::choice)
            .max(Comparator.comparingDouble(node -> getPoints(node, start.g
```

```
            .orElse(start);

    return result;
}
```

Wobei

```
getPoints(GameStateTreeNode, Player)
```

die Bewertung des jeweiligen Knotens durch die gecachten Werte ausführt, die im Vorraus mit `this::choice` ermittelt wurden.

2.3.1 Performance

Mit einer rein rekursiven Strategie tritt sehr schnell ein Performance-Problem auf. Bei einem $3x3$ Tic-Tac-Toe Feld sind es $(3 * 3)! = 362.880$ mögliche Zustände, während es bei einem $7x6$ VierGewinnt bereits mindestens $7!^6 \approx 1.64 * 10^{22}$ sind. Bei klassischem Schach bewegt sich die Anzahl der möglichen Zustände in der Größenordnung von 10^{120} [7]. Da dies selbst auf moderner Hardware nicht in einer annehmbaren Zeit berechenbar ist, wird die Traversierung im Rahmen dieser Arbeit anhand von verschiedenen Kriterien abgebrochen.

3 Methodik

Um die Varianten der Baumexploration zu ergründen, haben wir den Algorithmus in mehrerer kleine Teile zerlegt. Für diese wurden mehrere beispielhafte Varianten implementiert (siehe Sektion 3.3). Die gefundenen und implementierten Algorithmen werden anschließend in jeder möglichen Kombination, die in einer für den Umfang dieser Arbeit annehmbaren Zeit berechnet werden kann, ausgeführt. Die Messwerte in dieser Arbeit wurden auf einem 16" MacBook Pro der ersten Generation mit einem Intel Core i9-9880H und einem stabilen Takt unter Last von $3.10 - 3.20$ GHz gesammelt (siehe Taktfrequenz Diagramm im Anhang).

Dabei werden folgende Daten für jedes Spiel in eine Datei geschrieben:

- Parameter
  - Spielbrettgröße
  - Spieltyp
  - Spielkonfiguration
    - Gravitation und Gewinnlänge für Tic-Tac-Toe / VierGewinnt
  - Initiales Brett
  - Spielerkonfigurationen
    - Traversierungsstrategie
    - Cachestrategie
    - Bewertungsstrategie
    - Abbruchbedingung
- Metriken
  - Anzahl der Züge
  - Finales Brett

  - Endergebnis
  - Errors
  - Zeit des Threads
  - Spielermetriken
    * Betrachtete Knoten
    * Zeit des Spielers

Die gesammelten Metriken werden anschließend jeweils nach den genutzten Abbruchbedingung und Bewertungsstrategie gruppiert, akkumuliert, das arithmetische Mittel gebildet (bis auf die Endergebnisse) und grafisch aufgearbeitet.

3.1 Analysierte Spiele

3.1.1 Deterministische Strategiespiele

Damit ein Spiel mittels Wurzelbäume gelöst werden kann, müssen folgende Bedingungen erfüllt sein [5]:

Vollständiges Wissen Der komplette Zustand des Spiels muss bekannt sein.
Nullsummenspiel Spiele bei denen die Summe der Gewinne und Verluste aller Spieler zusammengenommen gleich null ist.
Endlich Die Anzahl der möglichen Spielzustände muss endlich und ein Gewinnzustand klar definiert sein.
Formalisierbar Keine kreativen oder zufälligen Komponenten und klar definierte Regeln.

In dieser Arbeit wurden anhand von Komplexität und Implementationsaufwand die in den folgenden Sektionen erklärten Spiele gewählt. Des Weiteren wurde Go, das in Erwägung gezogen wurde, aufgrund der hohen Spielbaum Breite und der schwierig zu implementierenden Heuristik zur Bewertung von Spielzügen nicht gewählt.

3.1.1.1 Tic Tac Toe

Bei Tic-Tac-Toe werden Spieler-spezifische Steine auf einem quadratischen Feld mit einer Seitenlänge von drei abwechselnd von zwei Spielern gesetzt. Gewonnen hat derjenige, dessen Steine eine zusammenhängende vertikale, horizontale oder diagonale Reihe mit einer Länge von drei bilden. Eine übliche Strategie ist es, Steine in Positionen zu setzen, wo mehrere Möglichkeiten einer Reihe entstehen. Dies sind im Normalfall die Mitte und die Ecken. [3]

3.1.1.2 Vier gewinnt

Bei Vier gewinnt ist es das Ziel, ähnlich wie bei Tic-Tac-Toe, eine zusammenhängende vertikale, horizontale oder diagonale Reihe mit einer Länge von vier zu bilden. Die Rahmenbedingungen sind jedoch dahingehend anders, dass das Feld sieben Spalten breit und sechs Zeilen hoch ist und Steine nach unten fallen. Hier gilt ebenfalls die Strategie, dass es von Vorteil ist die Felder zu belegen, die einem mehr Optionen geben eine Reihe zu vervollständigen. [1]

3.1.1.3 Schach

Das dritte Spiel, was hier betrachtet wird, ist Schach. Bei diesem Spiel gibt es sechs verschiedene Figuren, die jeweils eigene Bewegungsregeln haben. Jede Figur gibt es in Schwarz und Weiß für die beiden Spieler. Die zwei Spieler ziehen abwechselnd und es beginnt immer Weiß. Ein gegnerischer Stein kann geschlagen werden, indem eine eigene Figur auf das Feld gezogen wird. Das Ziel ist die Königsfigur des Gegners Schach Matt zu setzen, also in eine Position zu bringen, in der sie geschlagen werden und nicht ausweichen kann. [9]

Für diese Arbeit wurden einige Einschränkungen getätigt:

- Keine Rochade
- Kein Schlagen von Passanten
- Kein Spielende durch Remis außer es sind nur noch die Könige auf dem Feld
- Aufgeben nicht möglich

Aufgrund der hohen Komplexität von Schach können nur wenige Schritte im Voraus berechnet werden. Hier werden nur maximal drei Schritte berechnet, um grundlegende Strategien wie das Schlagen gegnerischer Figuren, dieses zu verhindern und einen Schlagabtausch zu ermöglichen.

3.2 Architektur

Ein Spiel besteht aus zwei Controllern, die auf austauschbare Komponenten zugreifen. Diese austauschbaren Komponenten werden dabei durch eine Factory erstellt, genauere Informationen sind dem UML Diagramm im Anhang zu entnehmen.

3.3 Austauschbare Komponenten

3.3.1 Caches

Die Baum-Caches sind lediglich zur Beschleunigung der Berechnung implementiert worden und wurden durch transitive HashMaps umgesetzt. Dabei gibt es zwei Varianten:

Komprimiert Transitivität wird bei .put() sichergestellt, schnellere .get() Operation.
Unkomprimiert Transitivität wird bei .get() berechnet, schnellere .put() Operation.

3.3.2 Traversierungsstrategien

Die Traversierungsstrategie definiert die Reihenfolge, in der die Knoten des Spielbaums berechnet werden. Dabei gibt es zwei Varianten, die Breitentraversierung und Tiefentraversierung, wobei erstere den Baum Ebene für Ebene berechnet, während letztere Pfad für Pfad bis zum Blatt berechnet. Die Art der Traversierung bestimmt dabei, welche Daten für die Abbruchbedingungen (siehe Sektion 3.3.3) zur Verfügung stehen.

3.3.3 Abbruchbedingungen

Um den Baum nicht vollständig zu berechnen, werden Abbruchbedingungen definiert. Wie früh abgebrochen wird, hat dabei eine Auswirkung auf sowohl die Performance als auch auf die Präzision der Bewertung. Somit stellen die Bedingungen einen Trade-off zwischen Performance und Präzision dar. Im Rahmen dieser Arbeit wurden zwei Varianten betrachtet:

Zuganzahl Bricht nach einer festgelegten Anzahl von Zügen ab. Dabei ist die Performance immer gleich und es ist kein Wissen über das Spiel notwendig.

Heuristik Bricht bei Unterbäumen ab, die keine absehbar vorteilhafte Spielsituation darstellen, und umgeht somit unter anderem Teilbäume, die zu einem garantierten Verlust führen.

3.3.4 Bewertung von Zügen

Bestimmt die Güte einer Spielsituation ohne den Teilbaum vollständig zu berechnen und ist möglichst minimal implementiert, um nicht die Exploration des Baums vom Zeitaufwand her zu übersteigen. Die Umsetzung ist jedoch sehr stark von dem Spiel abhängig und wird im Folgenden erläutert.

3.3.4.1 Tic-Tac-Toe & Vier Gewinnt

Die Anzahl der Steine in einer Reihe werden gezählt. Dabei gibt es zwei Varianten:

1. Anzahl der zusammenhängenden Steine wird gewertet

- Performance linear zu der Anzahl an Steinen
- Ignoriert Reihen mit freien Feldern dazwischen
- Betrachtet Reihen, die nicht vervollständigt werden können

2. Anzahl der Steine und leeren Felder in einer Linie

- Performance linear zu der Anzahl an Feldern
- Betrachtet mehrere Möglichkeiten eine Reihe zu vervollständigen

Sofern mehr als eine Reihe existiert gibt es zwei Möglichkeiten diese zu akkumulieren:

1. Die quadrierten Summen der Länge aufaddieren

- Legt Wert darauf mehrere Optionen zu haben

2. Maximale Länge verwenden

- Fokussiert sich nur auf die beste Strategie

3.3.4.2 Schach

Es werden die Figuren auf dem Brett mit Zahlenwerten aus einer Tabelle belegt und aufsummiert [2]. Dabei werden Aspekte wie die Positionierung der Figuren vernachlässigt. Außerdem wird dem König kein Wert zugewiesen, da dieser sich während einer Partie immer auf dem Spielfeld befindet.

Figur	Wert
König	–

Figur	Wert
Dame	9
Turm	5
Läufer	3
Springer	3
Bauer	1

4 Untersuchungsergebnisse

Die Diagramme zu den folgenden Abschnitten befinden sich nach Spiel gruppiert im Anhang.

4.1 Gewinnraten

Die Anzahl der Spiele, die eine Spielerkonfiguration gewonnen hat.

4.1.1 Tic-Tac-Toe & Vier Gewinnt

Entgegen der Erwartungen gab es keinerlei nennenswerte Abweichung durch die Erhöhung der Explorationstiefe. Dies ließe sich dadurch erklären, dass diese Spielarten meistens in einem Patt enden, wenn beide Spieler auf Basis einer grundlegenen Strategie handeln. Außerdem erwartet die Baumstrategie, dass der Gegenspieler den bestmöglichen Zug tätigt. Ein Ausnutzen von den Schwächen in der Strategie des Gegners oder diesen zu Fehlern zu verleiten ist dabei nicht möglich. Das Ergebnis ist dabei abhängig von der Art der Bewertung. Die besten Resultate wurden bei dem Zählen von zusammenhängenden Steinen erreicht.

4.1.2 Schach

Bei Schach hingegen entsteht ein signifikanter Unterschied durch die Anpassung der Rekursionstiefe. Dies lässt vermuten, dass bei Schach eine komplizierte Strategie über mehrere Züge hinweg notwendig ist, um zu gewinnen, während Spiele wie Tic-Tac-Toe eher darauf basieren den Gegner nicht gewinnen zu lassen. Des Weiteren hatten Abbruchbedingungen auf Basis von Heuristiken keine starke Auswirkung, was darauf schließen lässt, dass sich die Güte einer Spielsituation nicht durch einfache Algorithmen bestimmen lässt.

4.2 Verlustraten

Hier gab es im Gegensatz zu den Gewinnraten keine Unterschiede zwischen Schach und Tic-Tac-Toe / Vier Gewinnt. Eine höhere Rekursionstiefe hat in jedem Fall die Anzahl der verlorenen Spiele reduziert. Dies ist darauf zurückzuführen, dass Situationen in denen der Gegner gewinnt früher erkannt und verhindert werden können. Diese Vermutung wird durch den proportionalen Zusammenhang zwischen der Rekursionstiefe und der Anzahl von Pattsituationen bestärkt.

Außerdem ist ersichtlich, dass ein Abbruch, durch den Vergleich der Bewertung von verschiedenen Unterbäumen, einen negativen Effekt hat. Dies lässt darauf schließen, dass die Bewertungsmethoden von Knoten nicht optimal gewählt sind und entsprechend schlechtere Bäume exploriert werden.

4.3 Zuganzahl

Die Anzahl der Züge verhält sich anti-proportional zu der Anzahl der verlorenen Spiele. Vermutlich ist das ein Effekt davon, dass Spiele so lange laufen, bis eine Seite verliert. Da die evaluierten Strategien

laut vorher analysierten Daten sehr gut darin sind nicht zu verlieren aber auch nicht zu gewinnen, gehen die Spiele entsprechend so lange bis eine Partei verliert oder eine Pattsituation entsteht. Abgesehen davon wird dieser Wert lediglich als Teiler für andere Messwerte verwendet, um die Metriken pro Zug zu erhalten.

4.4 Performance

Lediglich die Daten für Spieler 1 weisen signifikante Veränderungen auf, was nicht verwunderlich ist, da die Werte nach den Konfigurationen von diesem gruppiert wurden. Abgesehen davon lassen sich keine weiterführenden Ergebnisse aus den Daten ableiten.

4.4.1 Anzahl der evaluierten Knoten

Die Anzahl steigt proportional zu der Anzahl der Züge. Deswegen wird im Folgenden der Wert durch die Anzahl der Züge geteilt verwendet. Die Anzahl der betrachteten Knoten pro Zug steigt exponentiell zu der Rekursionstiefe. Ein Abbruch durch Vergleich der Bewertung verschiedener Teilbäume hat nur einen marginalen Effekt. Dies lässt auf eine zu hohe Toleranz schließen oder darauf, dass es bei der Größe der betrachteten Bäume keine messbare Auswirkung hat.

4.4.2 Laufzeit

Das Ergebnis der Laufzeitmessung zeigt bei beiden Spielarten, dass es in den gemessenen Situationen wesentlich teurer ist, eine Abbruchbedingung auf Basis der Knotenbewertung zu evaluieren, als es wäre den betreffenden Teilbaum zu berechnen. Wie erwartet steigt die Laufzeit mit einer tieferen Traversierung.

5 Zusammenfassung

Strategie ist in jedem Fall eine Abwägung zwischen Zeitaufwand und Qualität. Mit genug Rechenkapazität könnte man die ideale Lösung für jede Situation in Schach finden, doch dies ist nicht realistisch. Ansätze die Strategie einer Baumsuche zu optimieren hatten im Laufe dieser Arbeit nur minimale Auswirkungen. Wie zuvor erwähnt führen Bäume in den geprüften Szenarien nur dazu, dass der aktive Spieler weniger verliert. Allerdings gab es kaum Veränderungen in der Anzahl der gewonnenen Spiele. Dies kann man darauf zurückführen, dass eine optimierte Baumsuche nicht in der Lage ist die Schwächen des Gegenspielers auszunutzen und ihn zu Fehlern zu verleiten, die aufgrund seiner Strategie möglich sind. Außerdem lässt sich ableiten, dass eine einfachere Heuristik für die Bewertung von Zügen effektiver ist als eine komplexere.

Abgesehen davon lässt sich keine direkte Schlussfolgerung auf die Auswirkung von bestimmten Heuristiken und Abbruchbedingungen in größeren Bäumen treffen. Stattdessen haben die Ergebnisse eher eine Aussage zu dem jeweils getesteten Spiel zur Folge, da die unterschiedlichen Resultate für Vier Gewinnt, Tic-Tac-Toe sowie Schach eher auf die Art des Spiels zurückzuführen sind als auf die angewendete Heuristik und Abbruchbedingung.

5.1 Forschungsempfehlung

Auch, wenn die Ergebnisse diese Arbeit selbst keinen Rückschluss auf die Anwendbarkeit bei größeren Bäumen zulassen, hat sie einen Einblick in zukünftige Forschungsrichtungen gegeben. Dabei könnten zum Beispiel verschiedene Spiele auf ähnliches Verhalten analysiert und darauf basierend in Gruppen eingeordnet werden in denen eine tiefgreifendere Forschungen zur Optimierung betrieben werden kann. Außerdem können zusätzlich zu den hier angewendeten Algorithmen weitere evaluiert werden um eventuell auch im Kontext der zuvor genannten Gruppen neue Erkenntnisse zu gewinnen. Schlussendlich kann diese Art von Forschung mit größeren Bäumen auf leistungsstärkerer Infrastruktur durchgeführt werden, um potenziell eine generalisierbare Aussage zu treffen.

5.2 Fazit

Die angewendeten Algorithmen können aufgrund des Performance-Overhead nicht mit einer reinen Brute-Force Lösung mithalten. Entsprechend liegt der Wert dieser Arbeit nicht in den Eingangs erwarteten Empfehlungen bezüglich der Algorithmen, sondern darin, dass ein Framework geschaffen wurde, welches zukünftige Forschungen ermöglicht, die durch die Erkenntnisse aus dieser Arbeit ermöglicht werden.

6 Anhang

6.1 Diagramme

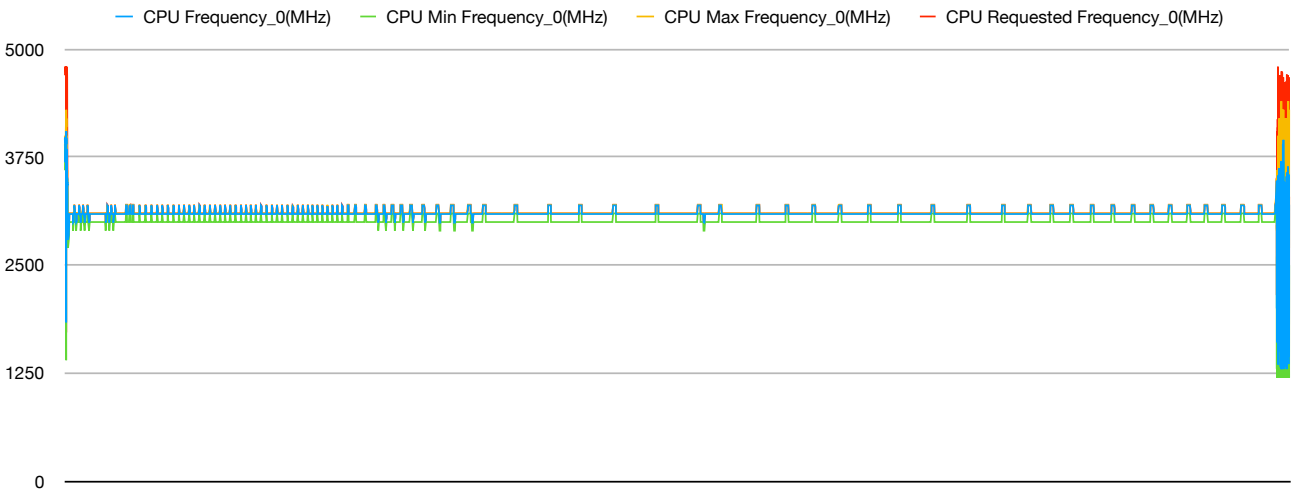

Abbildung 1: Taktfrequenz - Vier Gewinnt

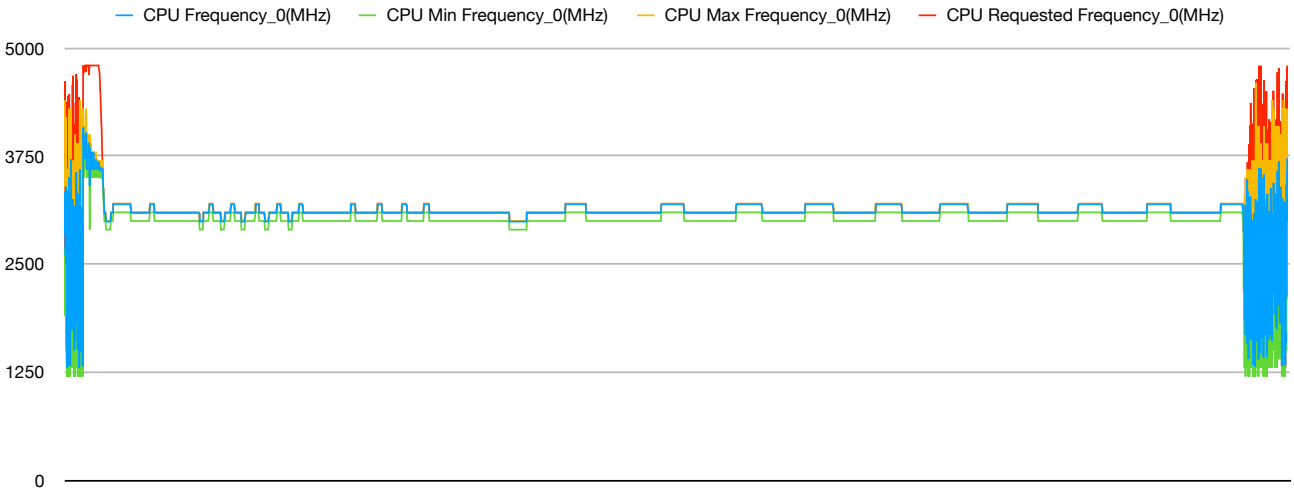

Abbildung 2: Taktfrequenz - Tic-Tac-Toe

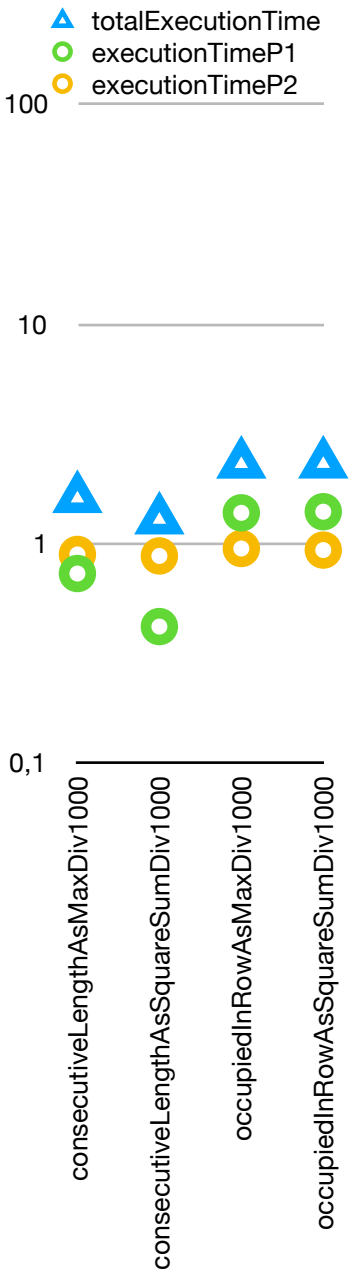

Abbildung 3: Ausführungszeit - Vier Gewinnt - Gruppiert nach Bewertungsstrategie

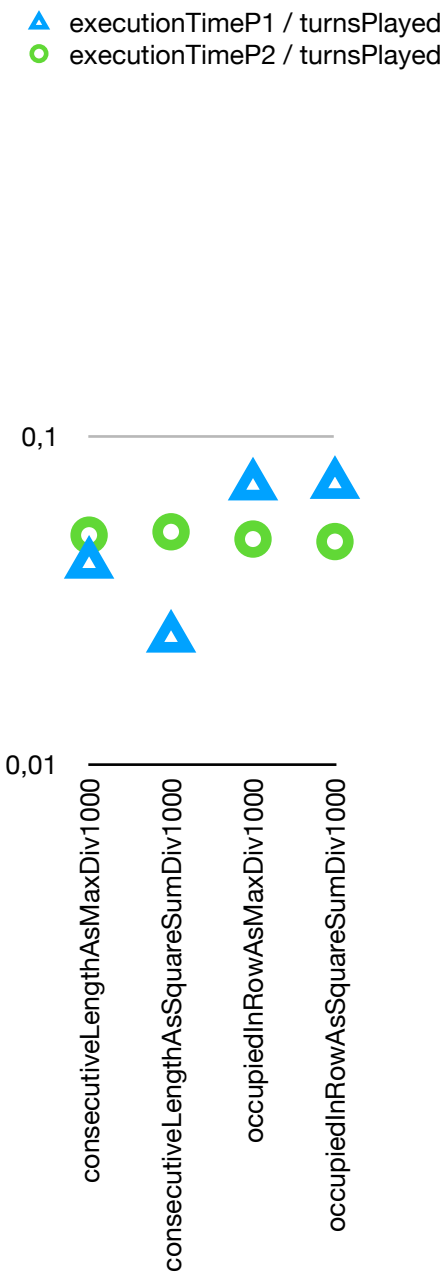

Abbildung 4: Ausführungszeit pro Zug - Vier Gewinnt - Gruppiert nach Bewertungsstrategie

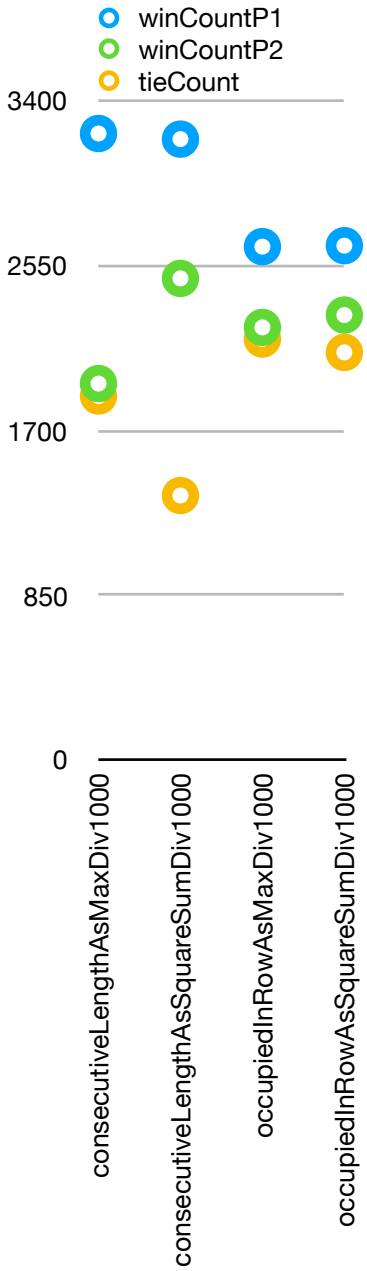

Abbildung 5: Spielende - Vier Gewinnt - Gruppiert nach Bewertungsstrategie

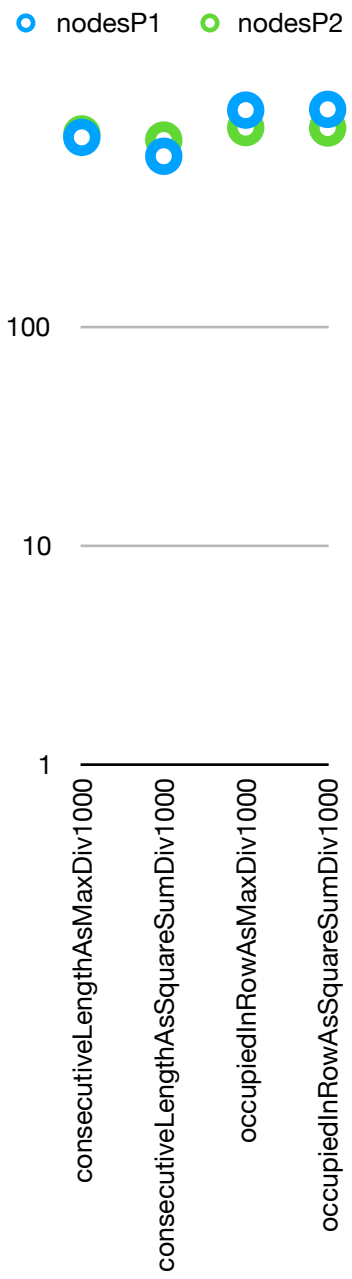

Abbildung 6: Betrachtete Knoten - Vier Gewinnt - Gruppiert nach Bewertungsstrategie

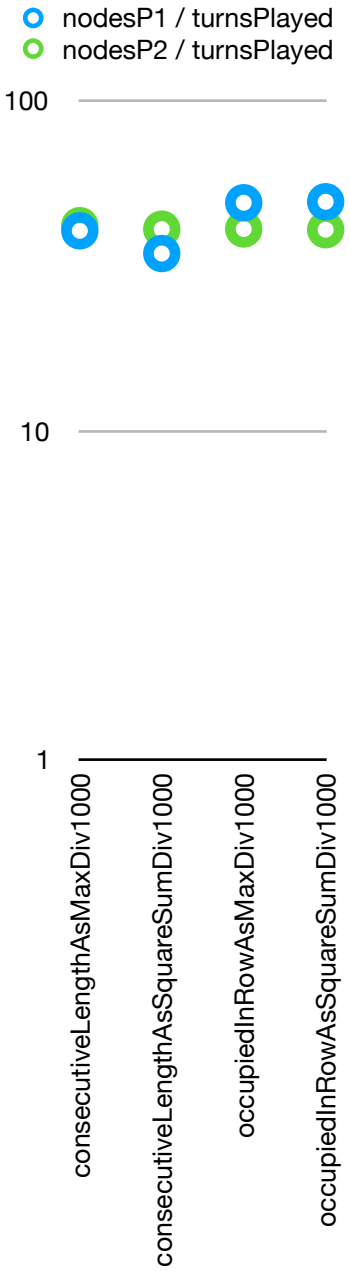

Abbildung 7: Betrachtete Knoten pro Zug - Vier Gewinnt - Gruppiert nach Bewertungsstrategie

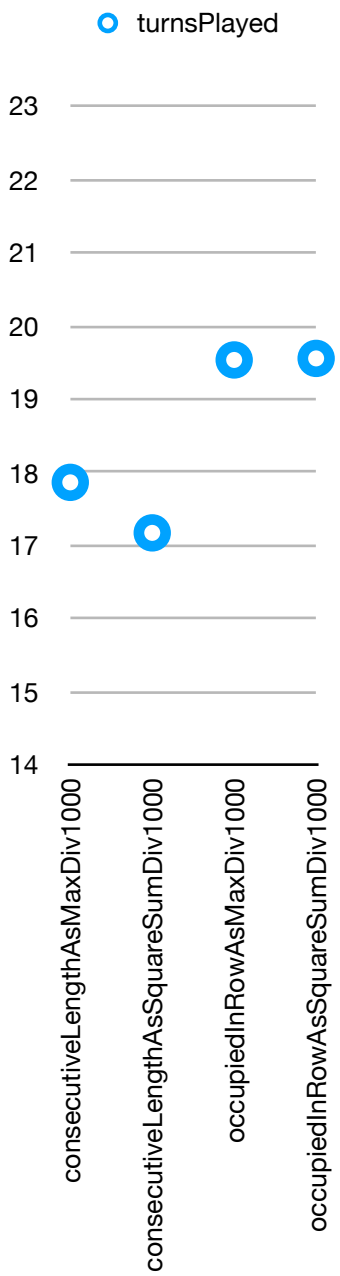

Abbildung 8: Anzahl von Zügen - Vier Gewinnt - Gruppiert nach Bewertungsstrategie

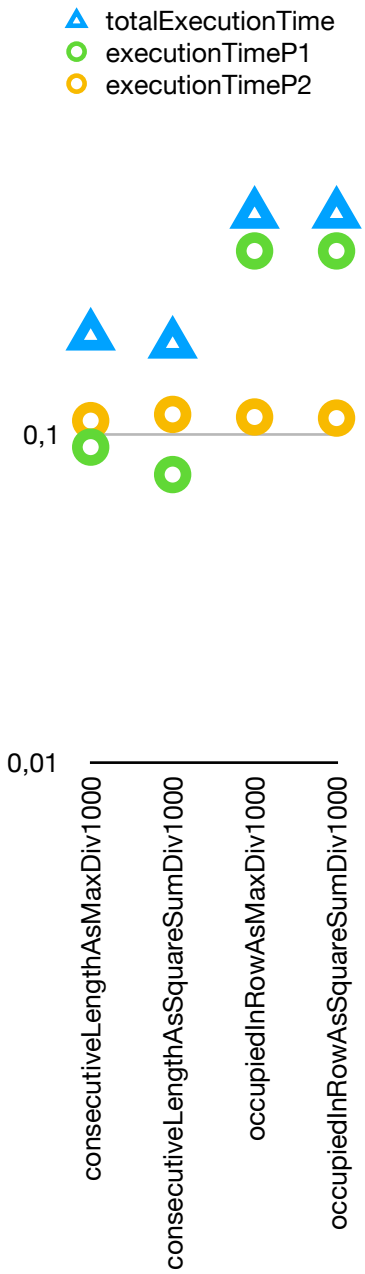

Abbildung 9: Ausführungszeit - Tic-Tac-Toe - Gruppiert nach Bewertungsstrategie

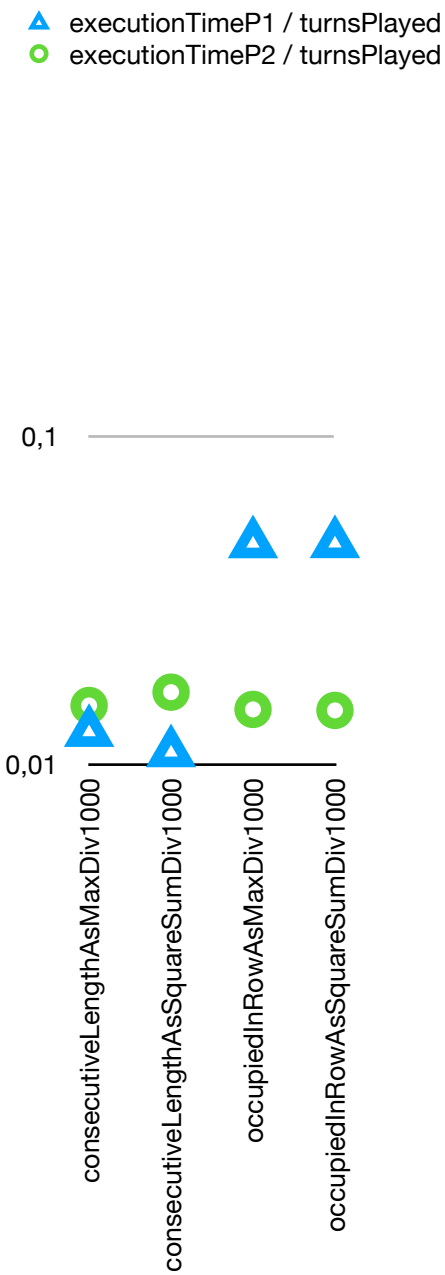

Abbildung 10: Ausführungszeit pro Zug - Tic-Tac-Toe - Gruppiert nach Bewertungsstrategie

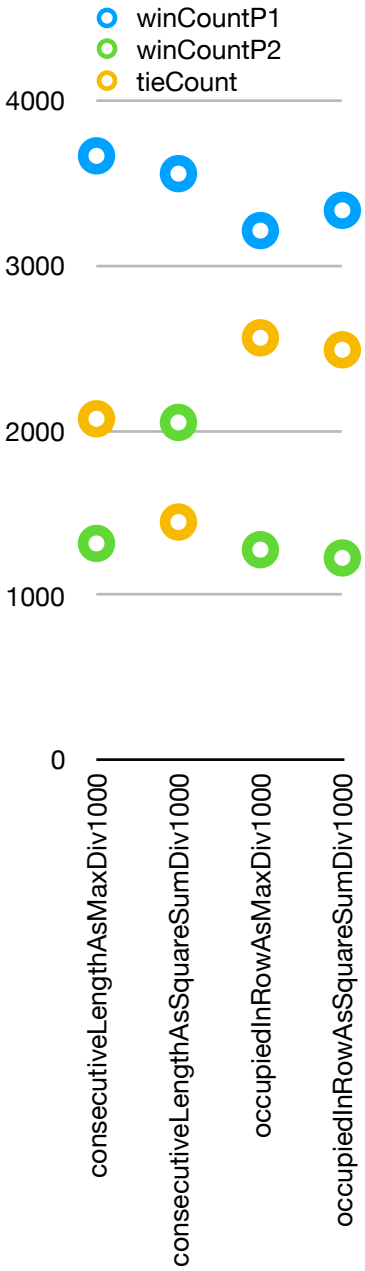

Abbildung 11: Spielende - Tic-Tac-Toe - Gruppiert nach Bewertungsstrategie

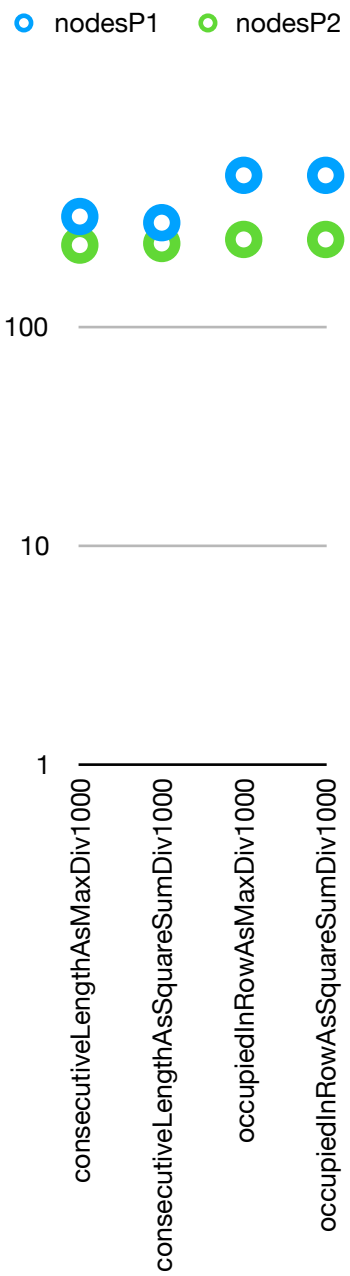

Abbildung 12: Betrachtete Knoten - Tic-Tac-Toe - Gruppiert nach Bewertungsstrategie

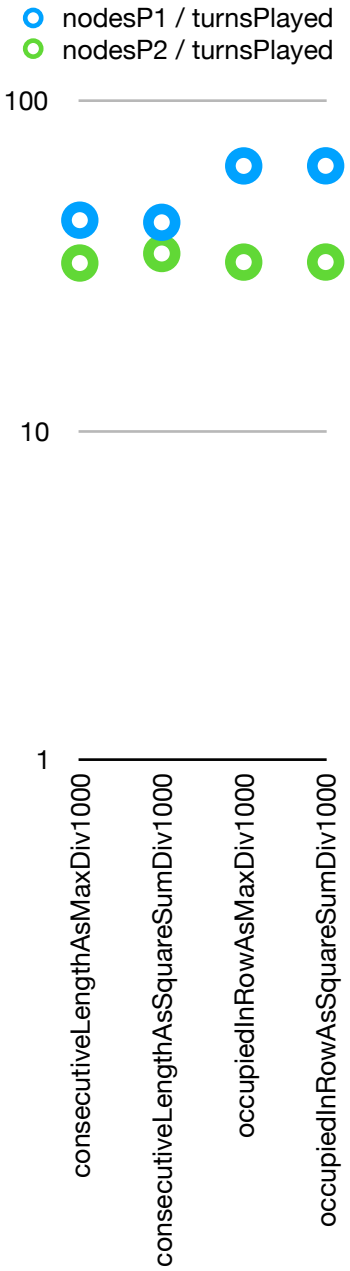

Abbildung 13: Betrachtete Knoten pro Zug - Tic-Tac-Toe - Gruppiert nach Bewertungsstrategie

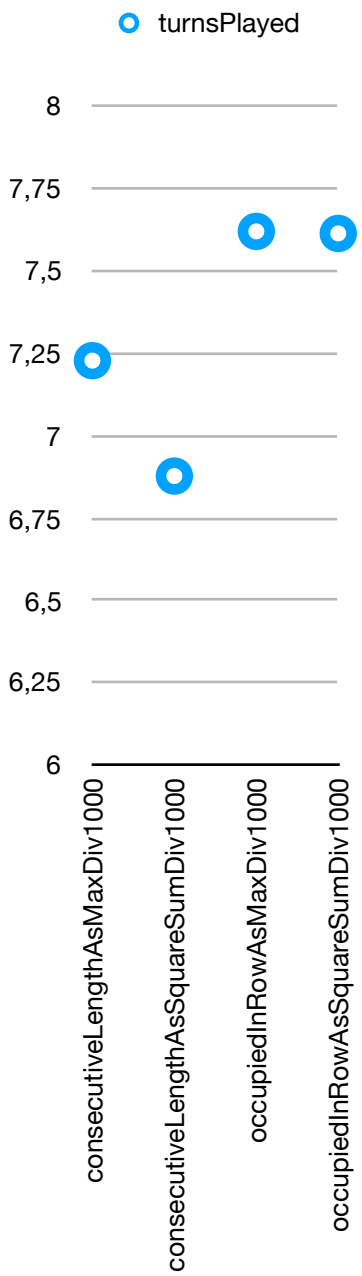

Abbildung 14: Anzahl von Zügen - Tic-Tac-Toe - Gruppiert nach Bewertungsstrategie

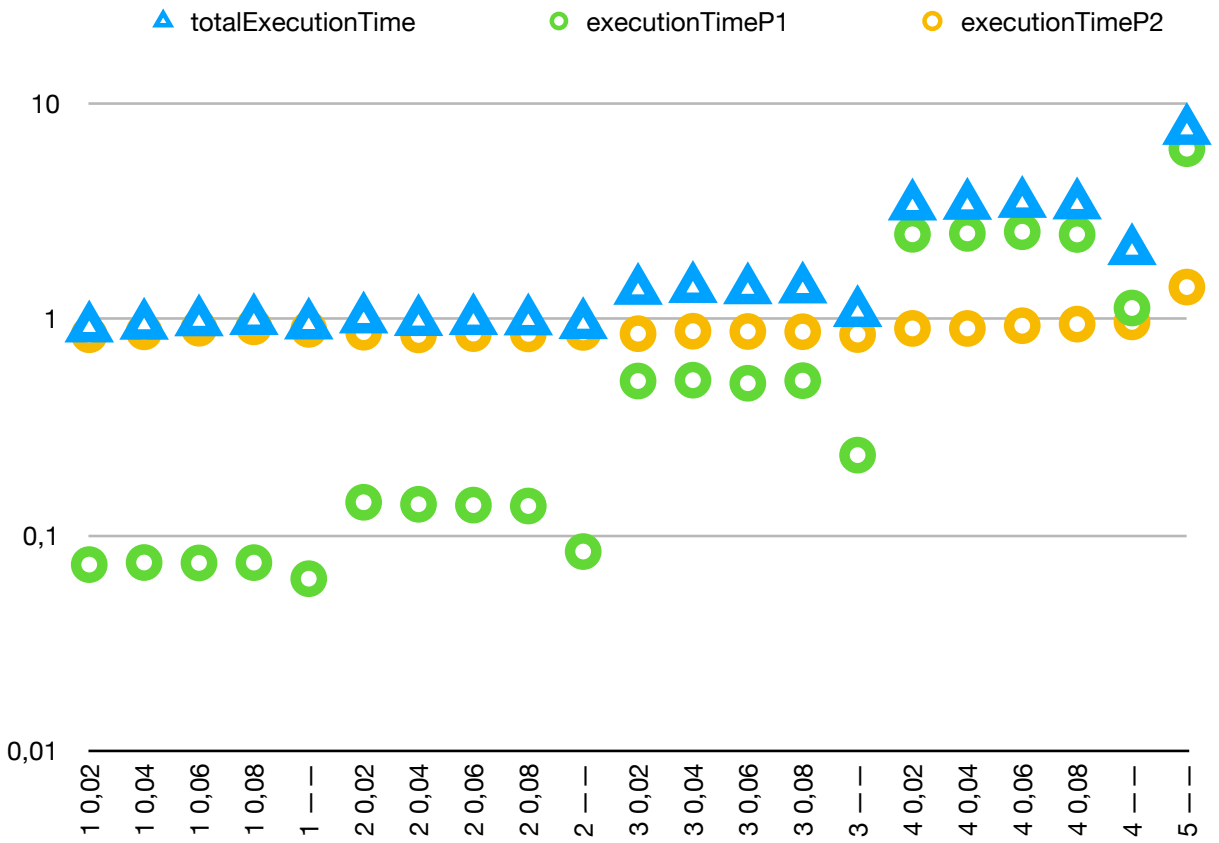

Abbildung 15: Ausführungszeit - Vier Gewinnt - Gruppiert nach Abbruchbedingung

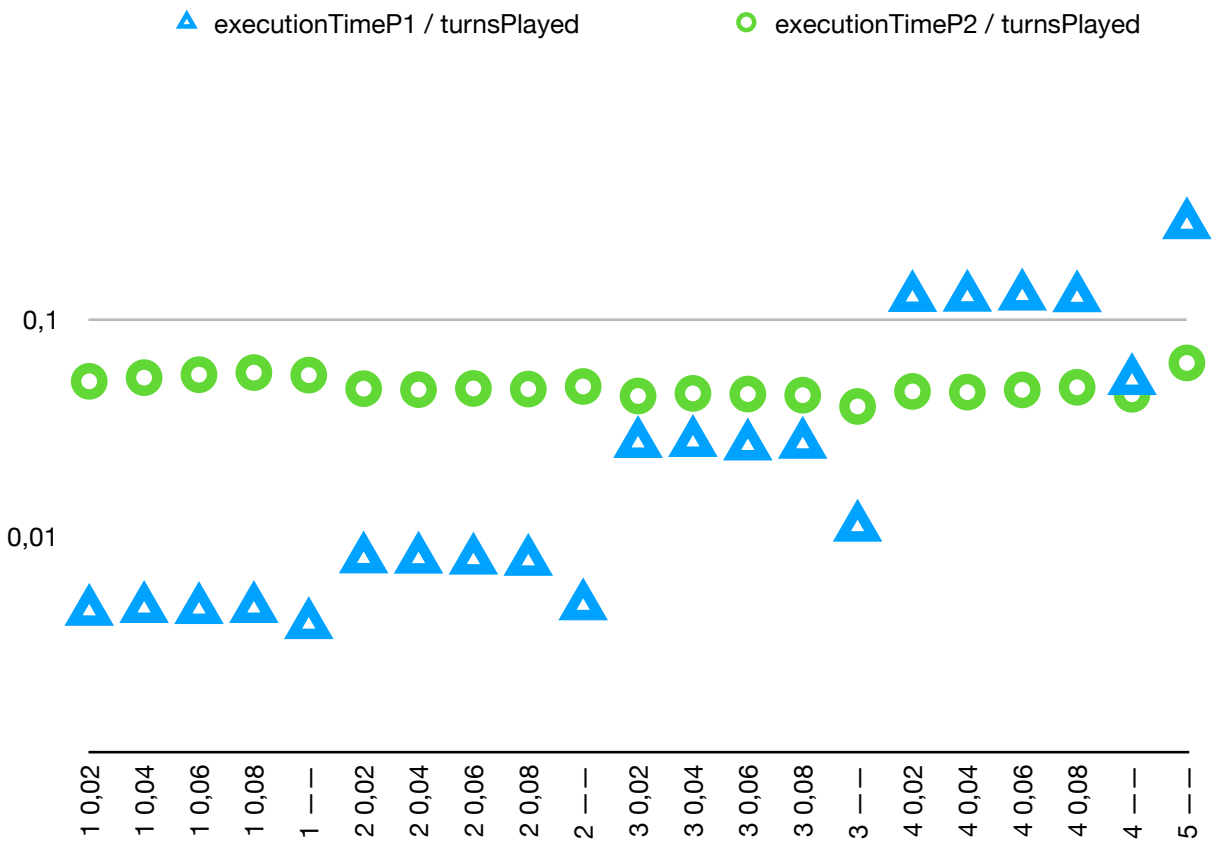

Abbildung 16: Ausführungszeit pro Zug - Vier Gewinnt - Gruppiert nach Abbruchbedingung

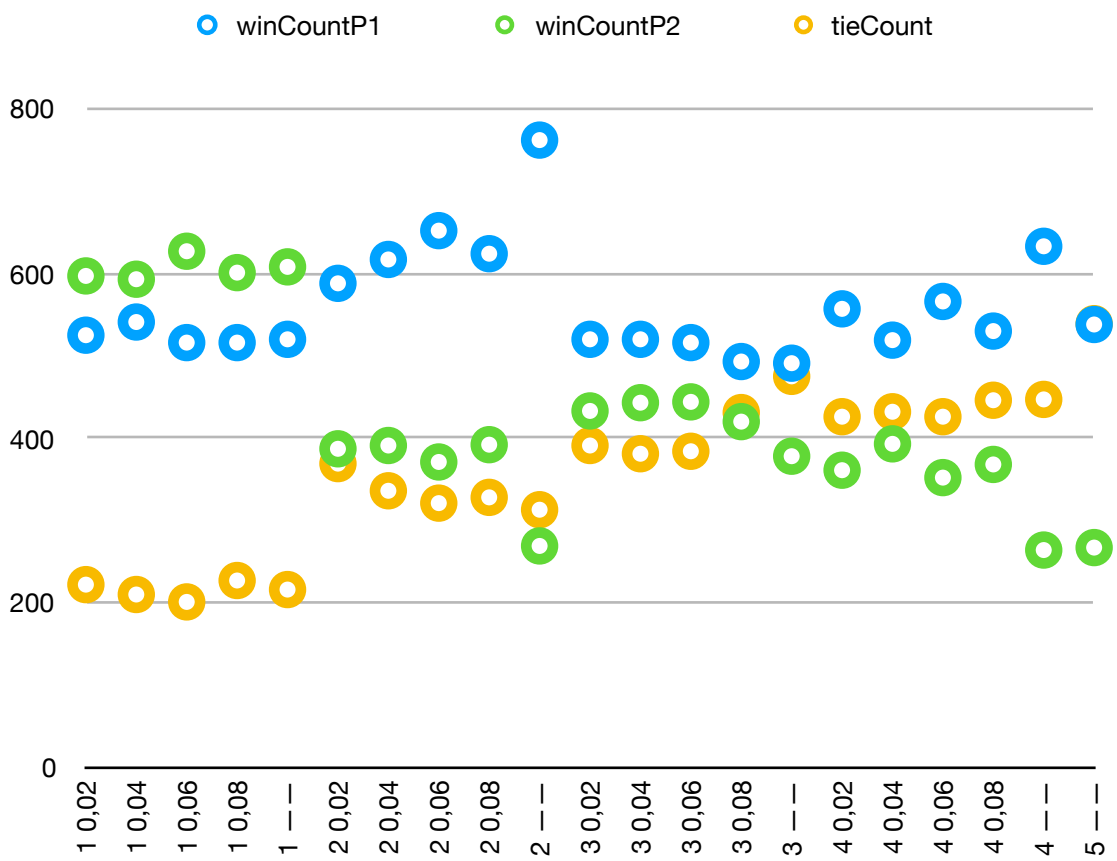

Abbildung 17: Spielende - Vier Gewinnt - Gruppiert nach Abbruchbedingung

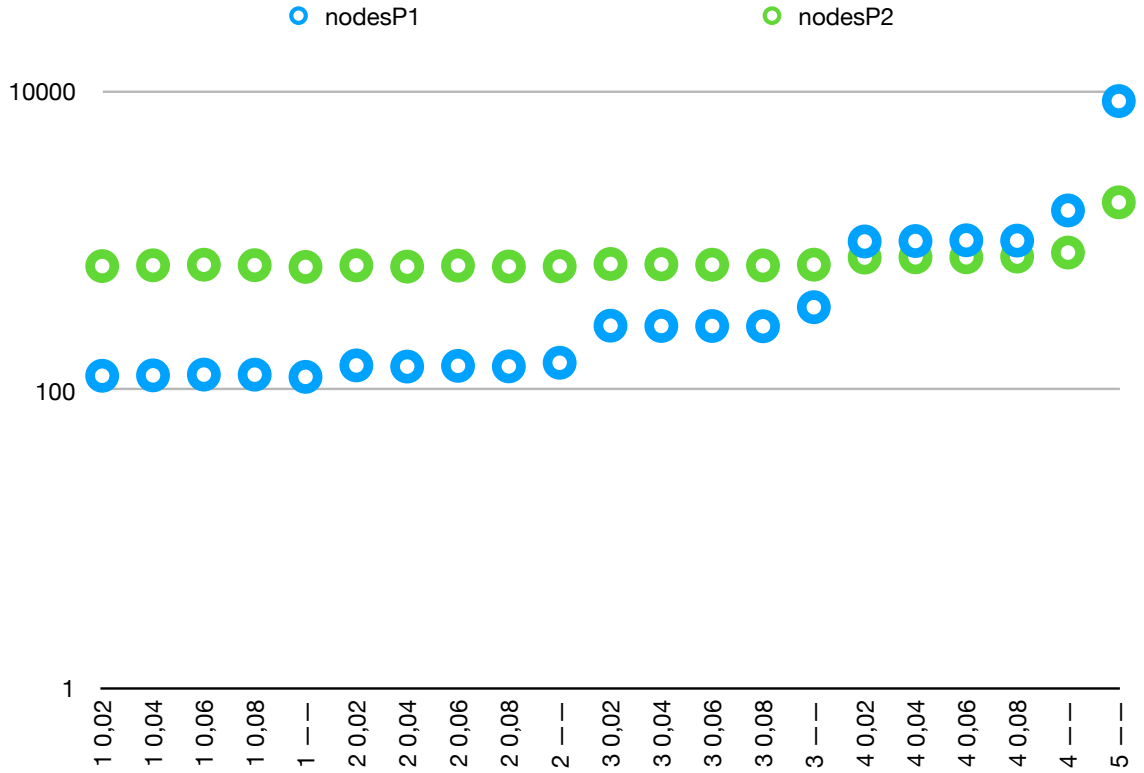

Abbildung 18: Betrachtete Knoten - Vier Gewinnt - Gruppiert nach Abbruchbedingung

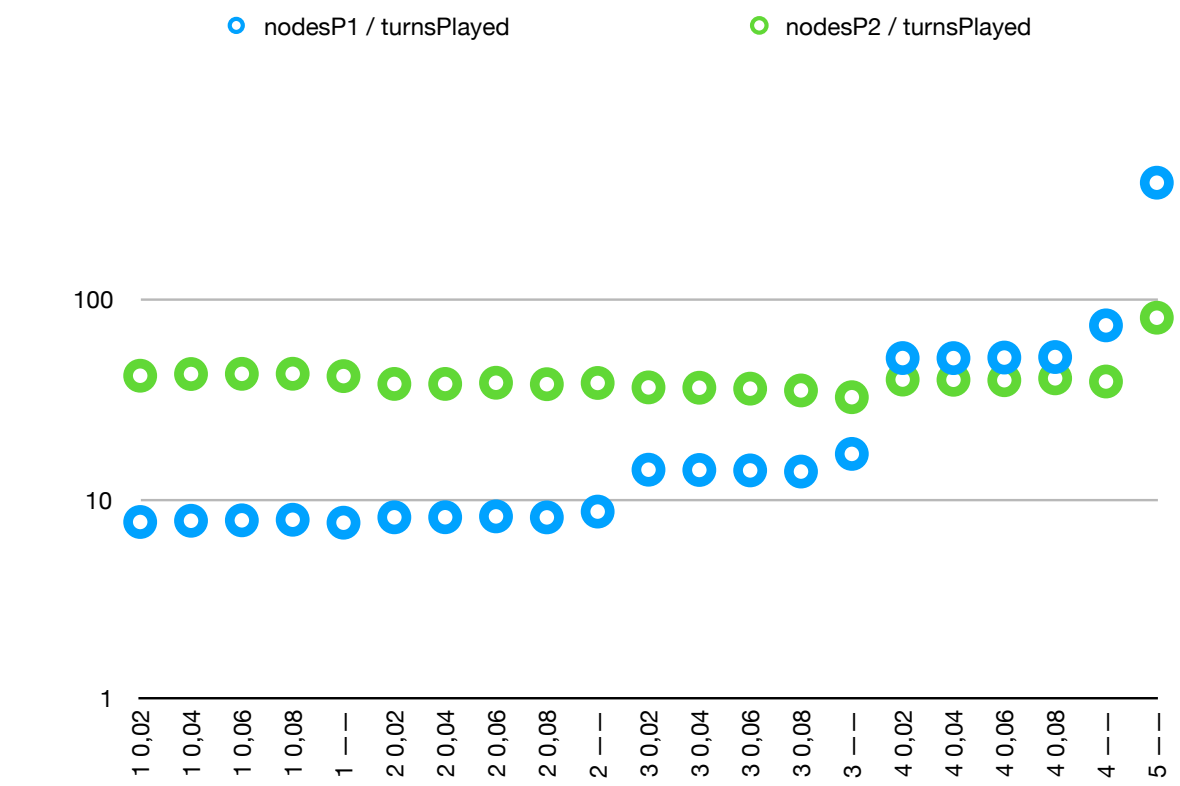

Abbildung 19: Betrachtete Knoten pro Zug - Vier Gewinnt - Gruppiert nach Abbruchbedingung

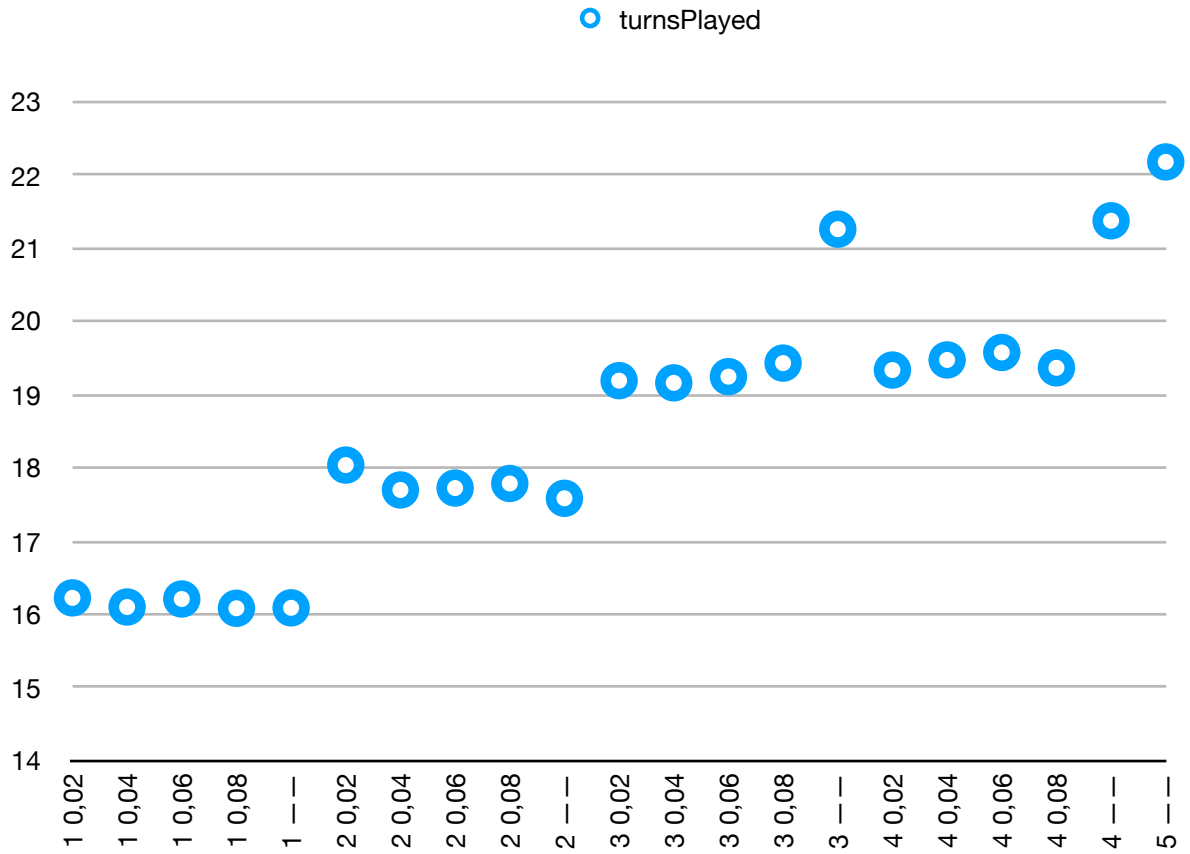

Abbildung 20: Anzahl von Zügen - Vier Gewinnt - Gruppiert nach Abbruchbedingung

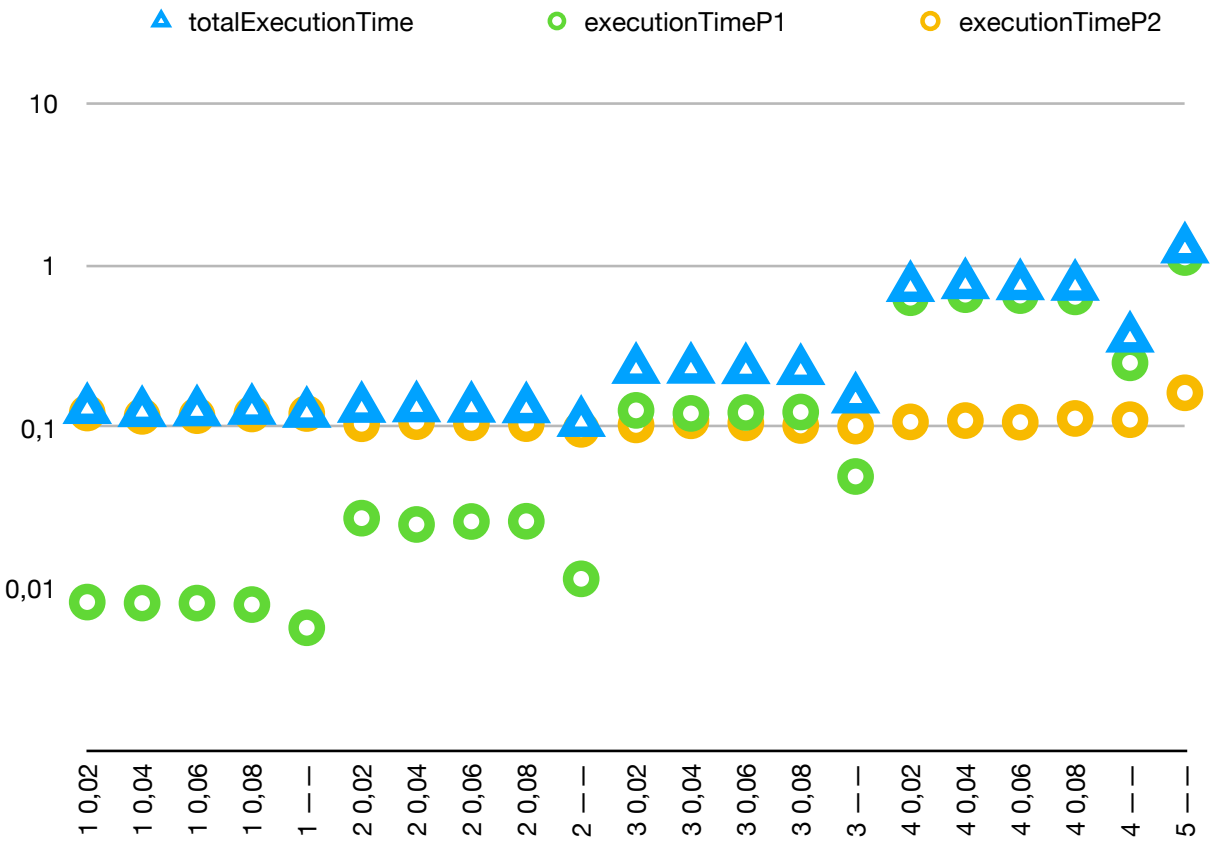

Abbildung 21: Ausführungszeit - Tic-Tac-Toe - Gruppiert nach Abbruchbedingung

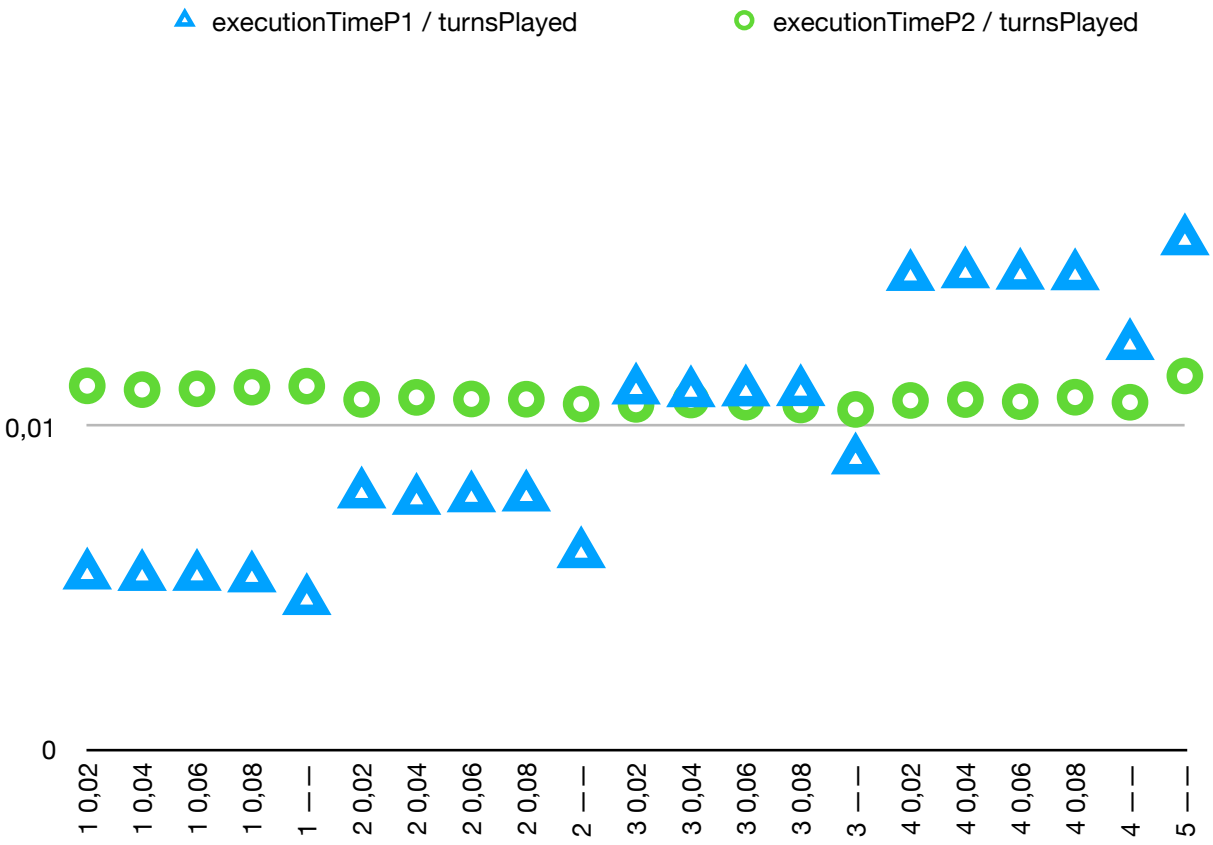

Abbildung 22: Ausführungszeit pro Zug - Tic-Tac-Toe - Gruppiert nach Abbruchbedingung

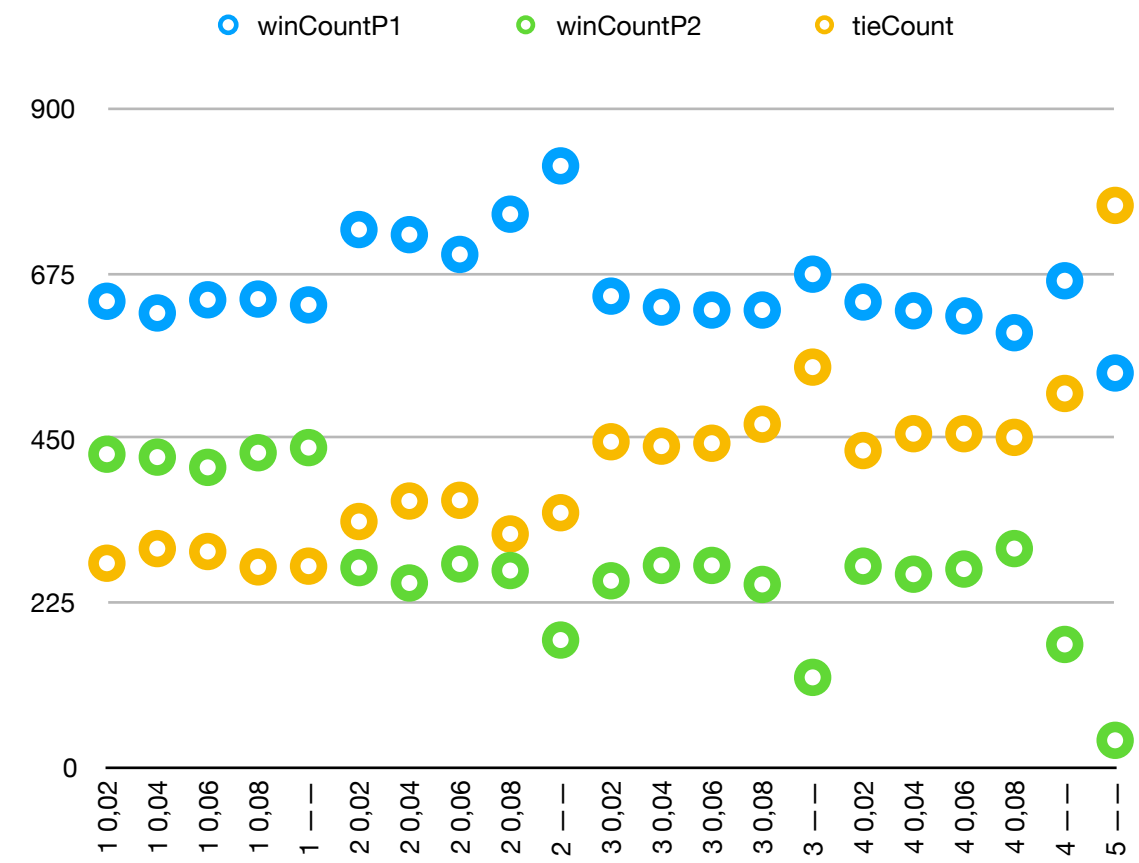

Abbildung 23: Spielende - Tic-Tac-Toe - Gruppiert nach Abbruchbedingung

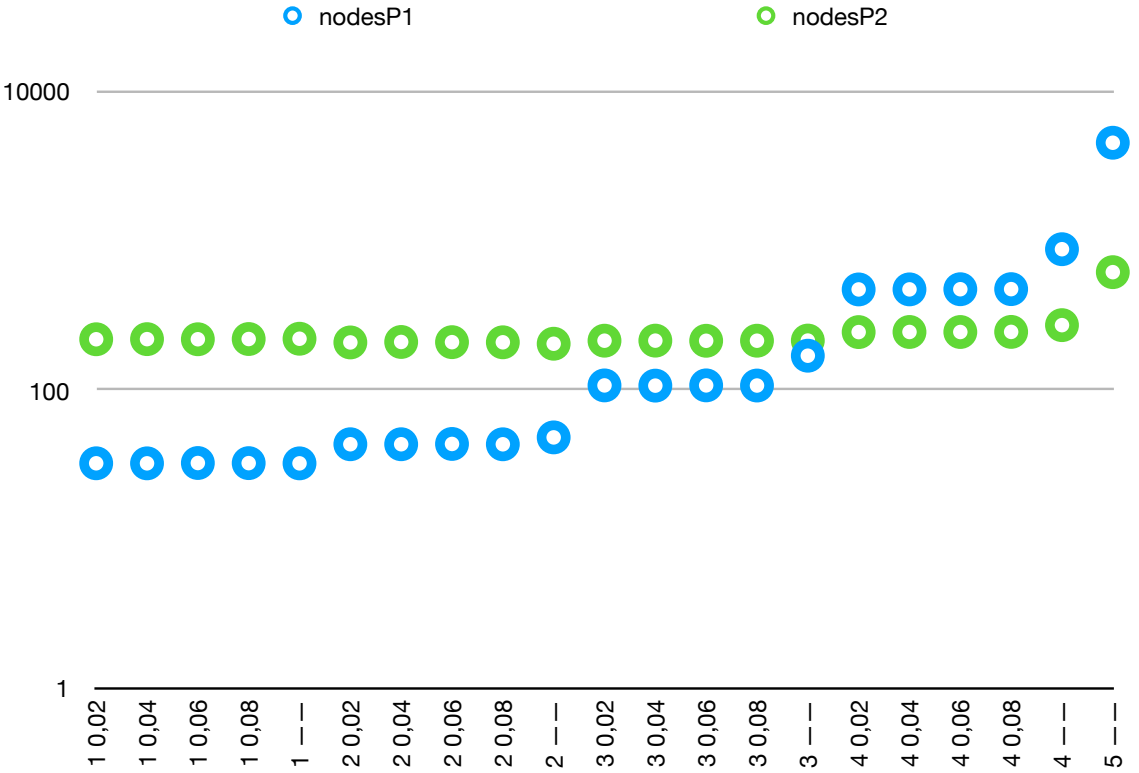

Abbildung 24: Betrachtete Knoten - Tic-Tac-Toe - Gruppiert nach Abbruchbedingung

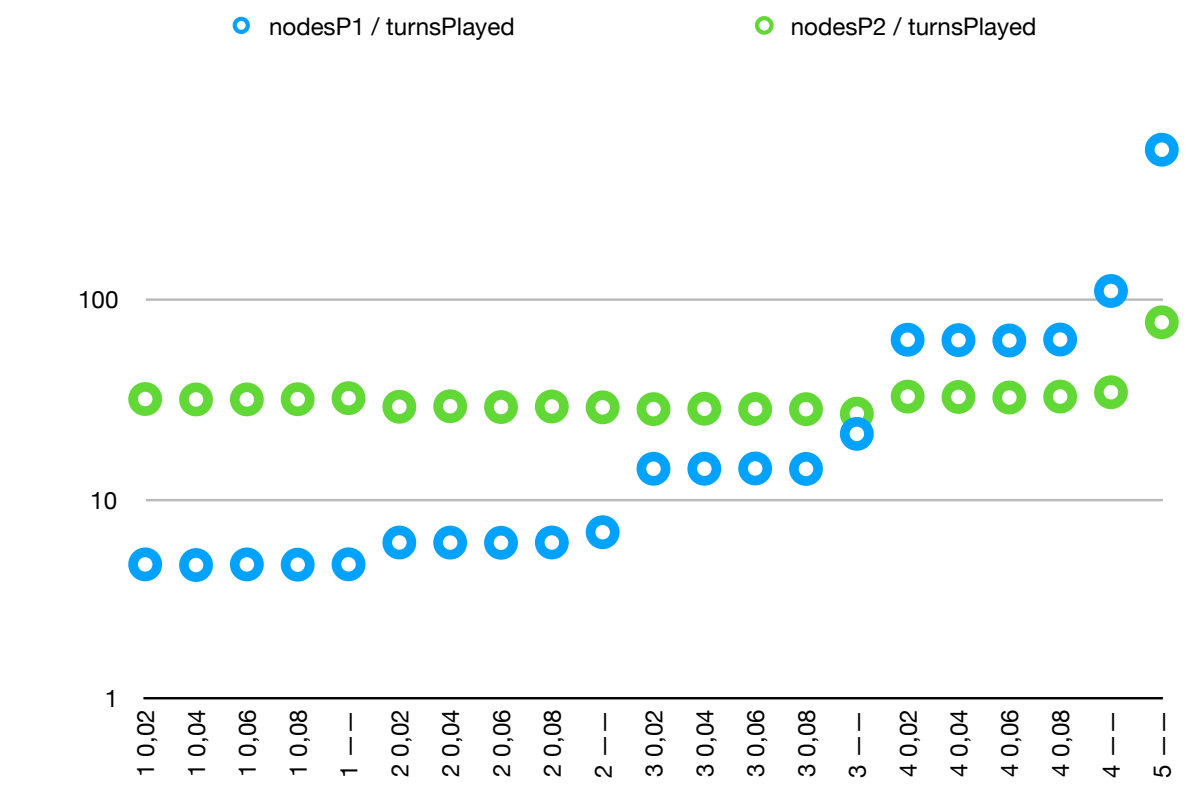

Abbildung 25: Betrachtete Knoten pro Zug - Tic-Tac-Toe - Gruppiert nach Abbruchbedingung

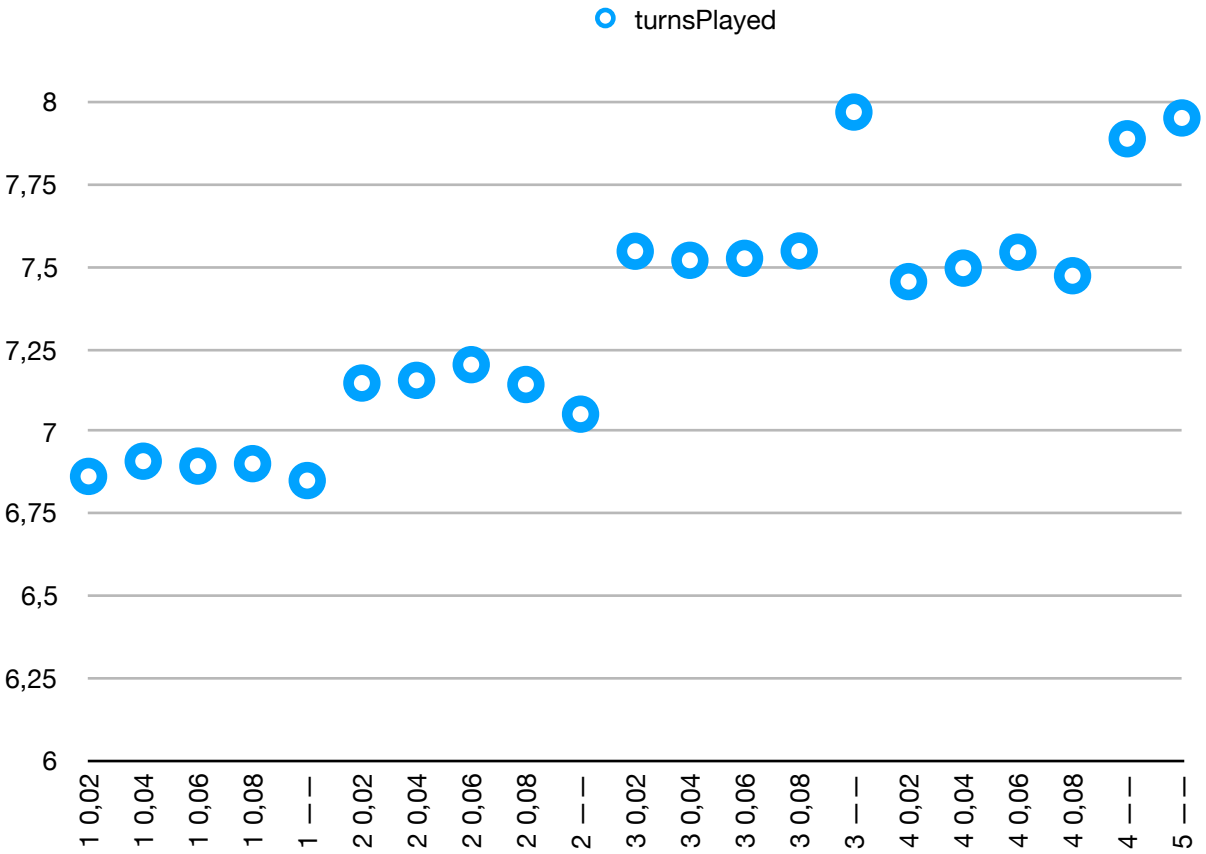

Abbildung 26: Anzahl von Zügen - Tic-Tac-Toe - Gruppiert nach Abbruchbedingung

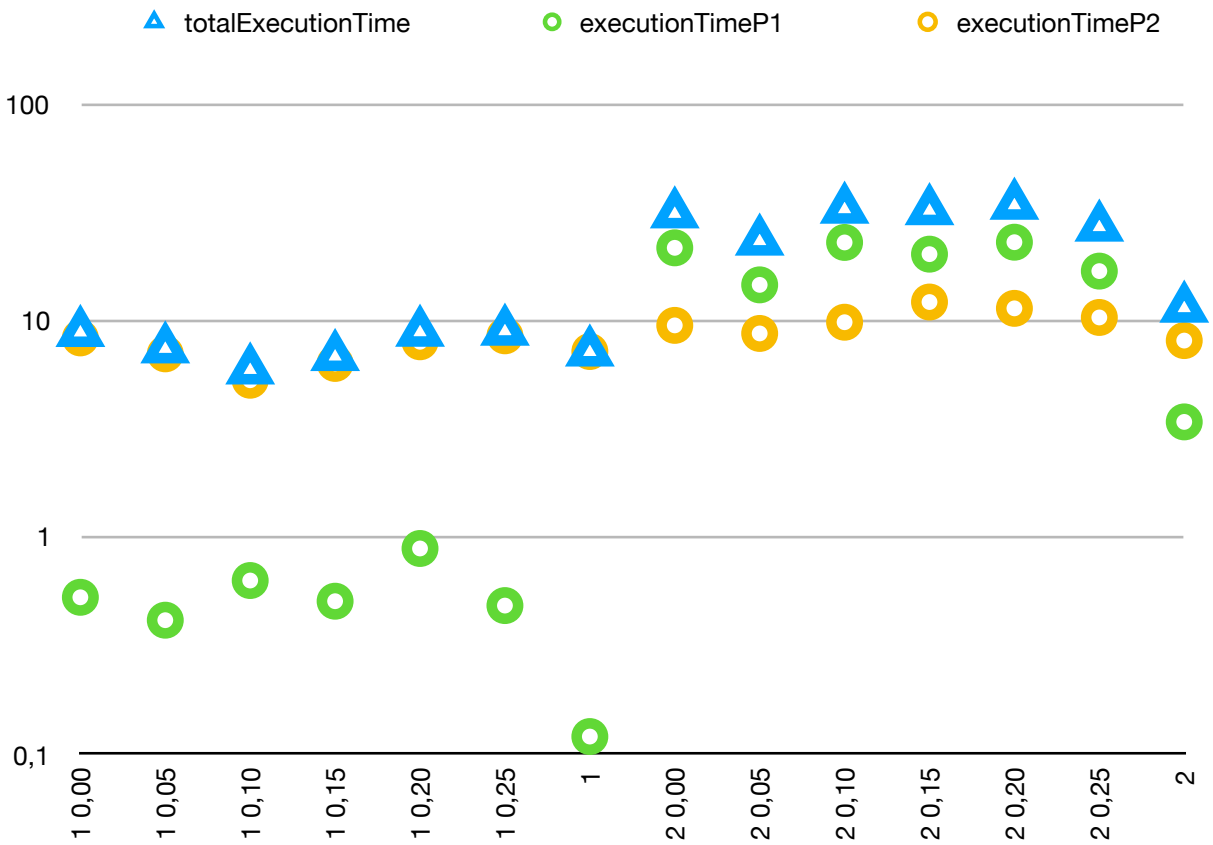

Abbildung 27: Ausführungszeit - Schach - Gruppiert nach Abbruchbedingung

Abbildung 28: Ausführungszeit pro Zug - Schach - Gruppiert nach Abbruchbedingung

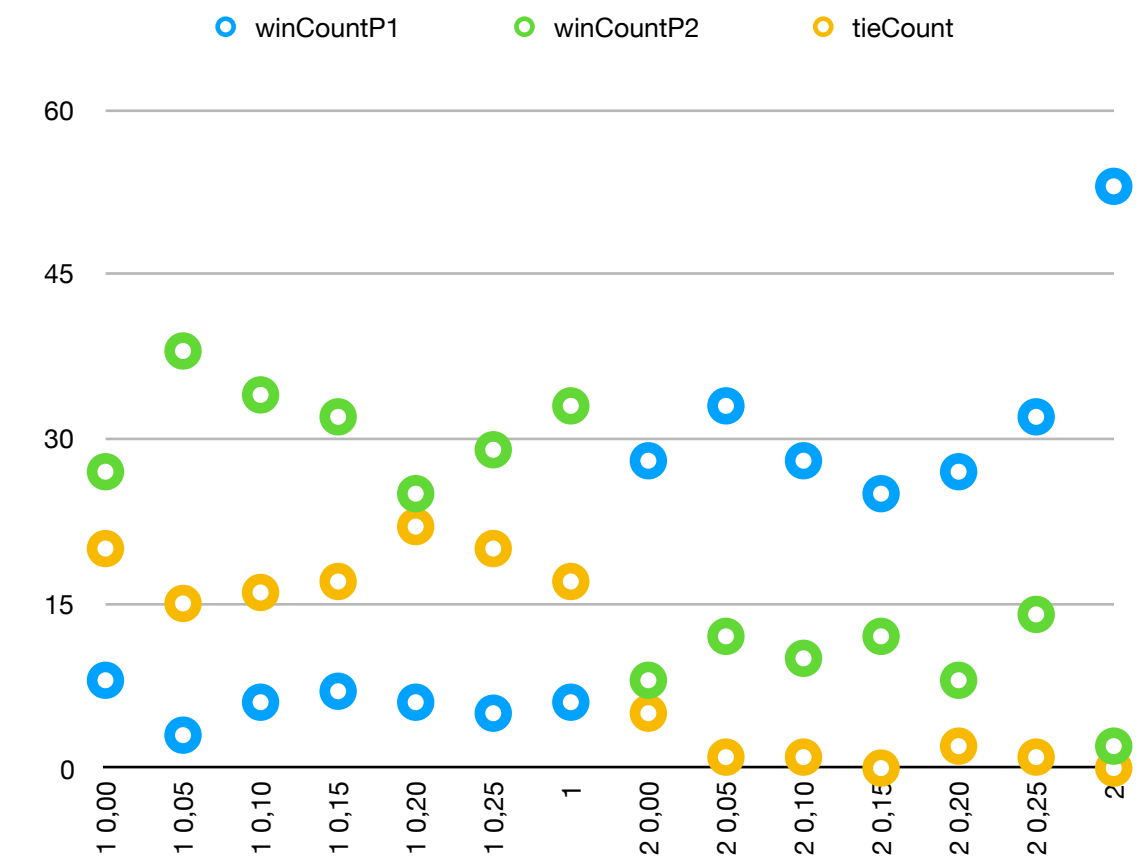

Abbildung 29: Spielende - Schach - Gruppiert nach Abbruchbedingung

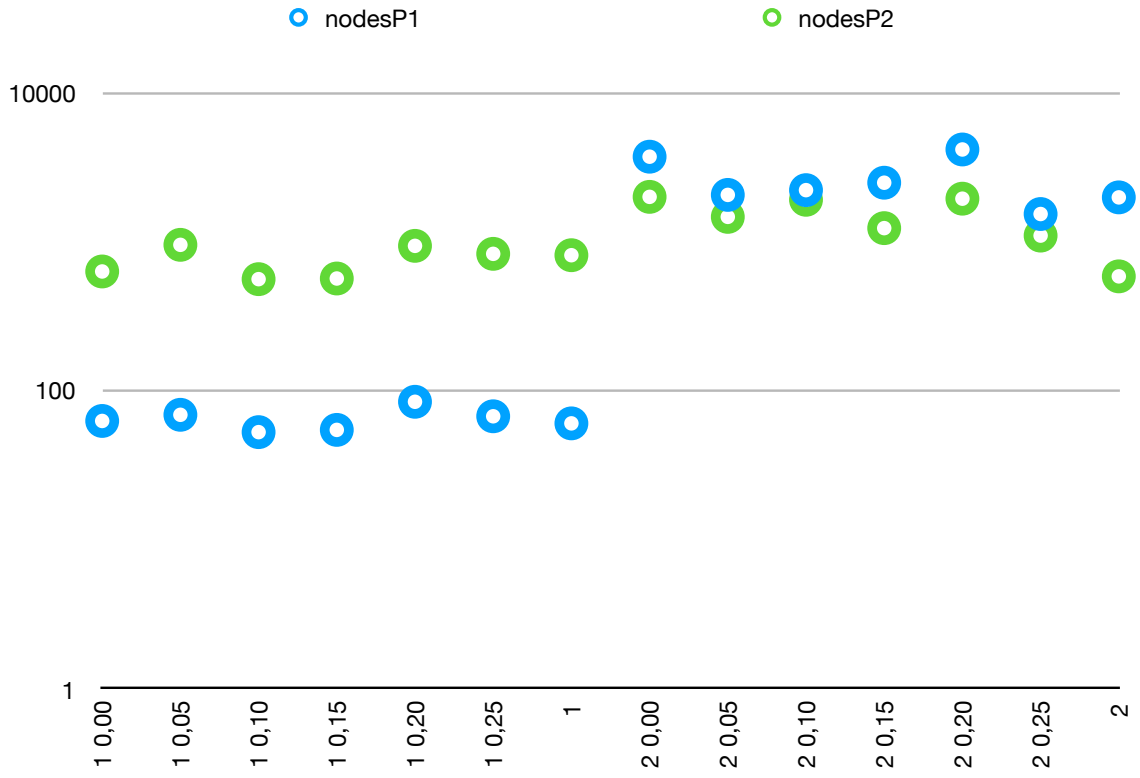

Abbildung 30: Betrachtete Knoten - Schach - Gruppiert nach Abbruchbedingung

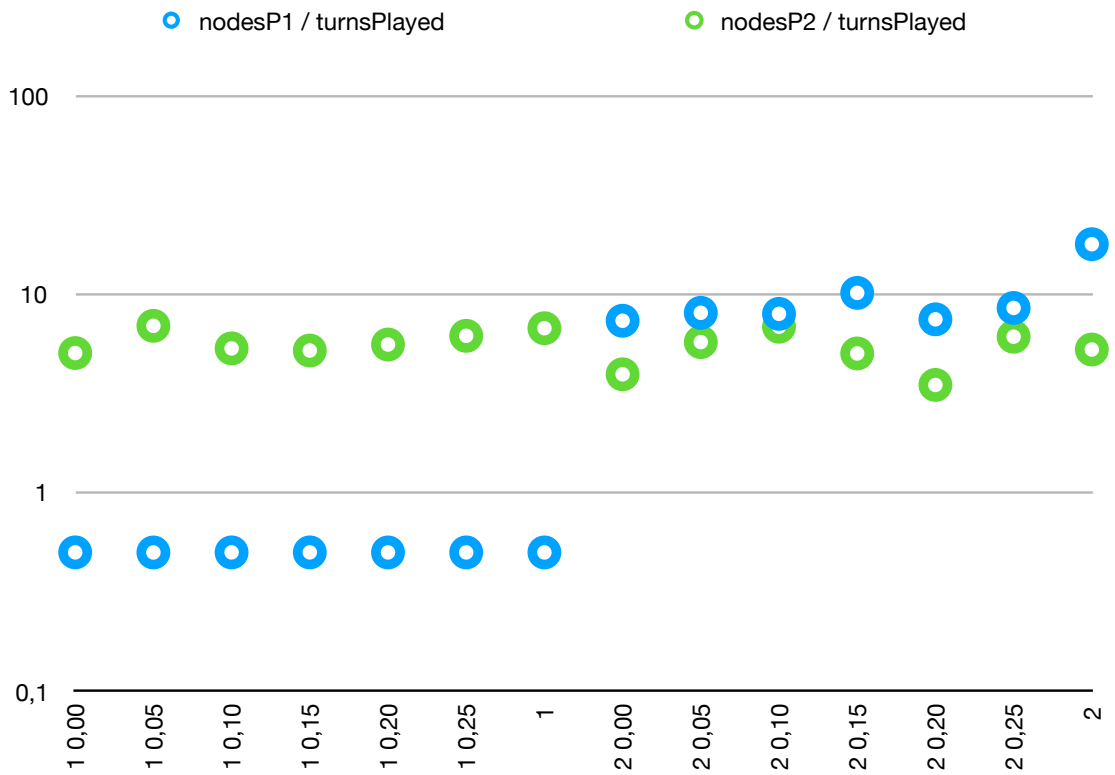

Abbildung 31: Betrachtete Knoten pro Zug - Schach - Gruppiert nach Abbruchbedingung

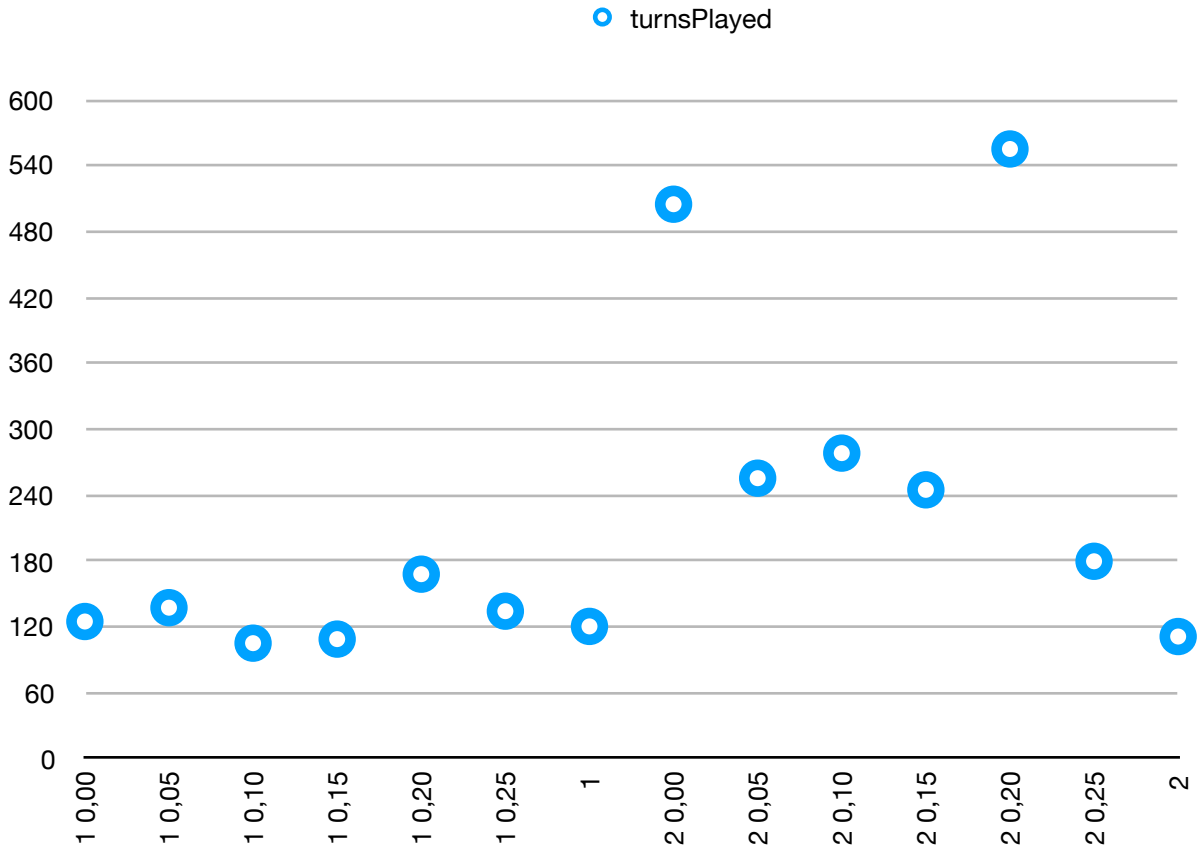

Abbildung 32: Anzahl von Zügen - Schach - Gruppiert nach Abbruchbedingung

6.2 Literaturverzeichnis

[1] Allis und Viktor, „A Knowledge-based Approach ofConnect-Four“. Okt-1988, http://www.informatik.uni-trier.de/~fernau/DSL0607/Masterthesis-Viergewinnt.pdf.

[2] J. Capablanca und N. de Firmian, „Chess Fundamentals (Completely Revised and Updated for the 21st century)“. 2006.

[3] Do und Norman, „How to Win at Tic-Tac-Toe“. https://www.austms.org.au/Gazette/2005/Jul05/mathellaneous.pdf.

[4] D. Ormerod, „AlphaGo defeats Lee Sedol in first game of historic man vs machine match“. [Online]. Verfügbar unter: https://web.archive.org/web/20160314112424/https://gogameguru.com/alphago-defeats-lee-sedol-game-1/. [Zugegriffen: 14-März-2016].

[5] J. Sauer, „KI Spielalgorithmen“. 11-Nov-2019, https://moodle2.nordakademie.de/course/view.php?id=2966#section-7.

[6] H. Sergiu, „Games in extensive and strategic forms“. Elsevier, 1992.

[7] C. Shannon, „Programming a Computer for Playing Chess“. Philosophical Magazine, März-1950, http://archive.computerhistory.org/projects/chess/related_materials/text/2-0%20and%202-1.Programming_a_computer_for_playing_chess.shannon/2-0%20and%202-1.Programming_a_computer_for_playing_chess.shannon.062303002.pdf.

[8] D. Silver, „Mastering the game of Go with Deep Neural Networks & Tree Search“. NATURE, Jan-2016, https://deepmind.com/research/publications/mastering-game-go-deep-neural-networks-tree-search.

[9] „FIDE LAWS OF CHESS TAKING EFFECT FROM 1 JANUARY 2018“. [Online]. Verfügbar unter: http://rules.fide.com/images/stories/Laws_of_Chess_2018_-EB_approved-_clean_version.pdf. [Zugegriffen: 19-Dez-2019].