Dynamische Schwierigkeitsanpassung von Tetris mit logistischer Regression

Bachelor-Thesis, 29.06.2015

Johannes Schubert, Studienbereich Computational Engineering
Betreuer: Prof. Johannes Fürnkranz, Prof. Dr. Ulf Brefeld, Daniel Bengs

TECHNISCHE UNIVERSITÄT DARMSTADT

Fachgebiet Knowledge Engineering
Fachgebiet Knowledge Mining & Assessment

Dynamische Schwierigkeitsanpassung von Tetris mit logistischer Regression
Dynamic difficulty adaption of Tetris with logistic regression

Vorgelegte Bachelor-Thesis von Johannes Schubert

1. Gutachten: Professor Johannes Fürnkranz, Professor Dr. Ulf Brefeld
2. Gutachten: Daniel Bengs

Tag der Einreichung:

Erklärung zur Bachelor-Thesis

Hiermit versichere ich, die vorliegende Bachelor Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 29. Juni 2015

_____
(Johannes Schubert)

Inhaltsverzeichnis

1 Einführung **6**
- 1.1 Motivation . . . 6
- 1.2 Ziel . . . 6
- 1.3 Aufbau . . . 7

2 Grundlagen **8**
- 2.1 Interval Subdivision Agent . . . 8
- 2.2 Logistische Regression . . . 9
- 2.3 Logistische Regression als Lernalgorithmus . . . 11

3 Adaptris **13**
- 3.1 Aufbau Adaptris . . . 13
- 3.2 Aufseher . . . 13
- 3.3 Nutzerstudie . . . 14
- 3.4 Infrastruktur . . . 14

4 Anwendung der Lernalgorithmen auf Adaptris **15**
- 4.1 Flow Modell . . . 15
- 4.2 Schwierigkeitsdefinition in Tetris . . . 15
- 4.3 Anwendung des ISA Algorithmus in Tetris . . . 16
- 4.4 Ermittlung der vorgegebenen Rate für schlechte Züge . . . 16
- 4.5 Anwendung der Logistischen Regression in Tetris . . . 18
  - 4.5.1 Eine unabhängige Variable . . . 18
  - 4.5.2 Zwei unabhängige Variablen . . . 20
- 4.6 Reaktionsbasierter Aufseher . . . 21

5 Heuristiken für binäres Feedback und optimale KI **23**
- 5.1 Heuristiken zur Bewertung eines Spielzuges . . . 23
  - 5.1.1 Fallbewertung . . . 23
  - 5.1.2 Feldbewertung . . . 24
  - 5.1.3 Feldbewertung mit Lücken . . . 24
  - 5.1.4 Feldbewertung nach Yiyuan . . . 24
  - 5.1.5 Ein-Zug-Lookahead . . . 25
  - 5.1.6 Heuristik mit dem Ein-Zug-Lookahead . . . 25
  - 5.1.7 Zwei-Züge-Lookahead . . . 26
  - 5.1.8 Heuristik mit dem Zwei-Züge-Lookahead . . . 27
  - 5.1.9 Heuristik für Feedback realer Spieler . . . 27
- 5.2 Optimale KI . . . 28
  - 5.2.1 Unterschiede im Aufbau . . . 28
  - 5.2.2 Ergebnisse mit der optimalen KI . . . 28

6 Vergleich der Aufseher mit Spielersimulation **29**
- 6.1 Simulierte Spieler . . . 29

6.2 Simulierte Spiele . . . 31
6.3 Ergebnisse aus der Spielersimulation . . . 31
6.3.1 Vergleichswerte Spiele ohne Anpassung und optimale Ergebniswerte . . . 32
6.3.2 Simulationsauswertung LR1UV-Aufseher . . . 32
6.3.3 Simulationsauswertung LR2UV-Aufseher . . . 33
6.3.4 Simulationsauswertung ISA-Aufseher . . . 34
6.3.5 Vergleich zwischen LR1UV-, LR2UV- und ISA-Aufseher . . . 34
6.3.6 Vergleich zwischen ISA- und LR2UV-50-Aufseher . . . 35

7 Nutzerstudie 37
7.1 Aufbau . . . 37
7.2 Teilnehmerverteilung . . . 38
7.3 Teilnehmer aufgeteilt nach Kompetenz . . . 39
7.4 Ergebnisse und Interpretation . . . 39
7.4.1 Auswertung der Hintergrunddaten . . . 40
7.4.2 Auswertung der Fragebögen . . . 41
7.4.3 Auswertung nach Kompetenz . . . 43

8 Fazit 46

9 Ausblick 47

Abbildungsverzeichnis

2.1 Lineare Regression, logistische Regression 9
2.2 Hinzufügen eines Datenpunktes bei logistischer Regression 12

3.1 Spielfeld Adaptris 13
3.2 Tetrominos 14

4.1 Flow Modell 15
4.2 Boxplot Diagramme zur Verteilung der Anteile schlechter Züge 17
4.3 Initiale Datenmenge des LR1UV und Ablesen der Geschwindigkeit 19
4.4 Update der logistischen Regression 20
4.5 Initiale Datenmenge des LR2UV und Ablesen der Geschwindigkeit 20

5.1 Parameter der Feldbewertung nach Yiyuan 25
5.2 Vorsprung 26

6.1 Ablesen der Wahrscheinlichkeit aus dem Profil eines simulierten Spielers 30
6.2 Simulierte Spieler 30
6.3 Simulationsdaten zu LR1UV, LR2UV und ISA 33
6.4 Simulationsdaten zu ISA und LR2UV-50 35

7.1 Nutzerstudie: Ergebnisse des ersten Fragebogens 39
7.2 Nutzerstudie: Ergebnisse der Hintergrunddaten 41
7.3 Nutzerstudie: Fragebogenauswertung zu den Aufsehern 42
7.4 Nutzerstudie: Auswertung nach Kompetenz 44

Abkürzungsverzeichnis

ISA Interval Subdivision Agent

LR logistische Regression

UV unabhängige Variable

KI künstliche Intelligenz

LR1UV logistische Regression mit einer unabhängigen Variablen

LR2UV logistische Regression mit zwei unabhängigen Variablen

LR2UV-50 logistische Regression mit zwei unabhängigen Variablen; Annäherung an 50% schlechte Züge

1 Einführung

Diese Arbeit beschäftigt sich mit dynamischer Schwierigkeitsanpassung von Tetris mit binärem Feedback. Es geht darum, die Schwierigkeit des Spiels automatisch während des Spiels an die Spieler anzupassen. Dies wird mit Algorithmen praktiziert, die Feedbacks mit zwei möglichen Werten nutzen.

1.1 Motivation

Die Qualität von kommerziellen Computerspielen hängt direkt von ihrem Unterhaltungswert ab [1]. Um ein Spiel unterhaltsam zu machen, muss ein Verhältnis zwischen der Kompetenz der Spieler[1] und der Herausforderung des Spiels gefunden werden. Zu schwierige Herausforderungen führen bei Spielern zu Aufregung und Frustration, wohingegen zu wenig Herausforderung zu Langeweile führt [2]. Die Kompetenz der Spieler ist nicht beeinflussbar, daher muss die Schwierigkeit des Spiels angepasst werden. Da diese Kompetenz bei jedem Spieler anders ist, ist es nicht möglich, eine feste Schwierigkeit einzustellen, um allen Spielern ein gutes Spielerlebnis zu liefern. Aus diesem Grund muss eine KI (künstliche Intelligenz) die Spielschwierigkeit so anpassen, dass sie für jeden Spieler weder zu herausfordernd noch zu einfach ist. Dies ist das Ziel der Lernalgorithmen, die in dieser Arbeit vorgestellt werden.

2013 wurde von Brefeld und Bengs der Interval Subdivision Agent (ISA) vorgestellt. Dieses Lernverfahren soll in der Umgebung von Tetris getestet werden [3]. Das Spiel Tetris wurde gewählt, weil es ein relativ einfaches und sehr bekanntes Spiel ist. Der ISA und die in der Arbeit entwickelten Lernverfahren werden in der Arbeit im Hinblick auf das Spielerlebnis der Spieler evaluiert.

1.2 Ziel

Das erste Ziel dieser Arbeit ist, ein Lernverfahren basierend auf logistischer Regression für Adaptris[2] zu entwickeln. Dieses Verfahren ist Teil eines sogenannten Aufsehers, der den Spielstatus beobachtet und die Schwierigkeit in Adaptris einstellt. Das Lernverfahren lernt eine gute Schwierigkeitseinstellung und adaptiert die Schwierigkeit des Spieles auf den Spieler, sodass der Spieler unabhängig von seiner Kompetenz weder über- noch unterfordert ist. Das Spielerlebnis soll dadurch verbessert werden. Dazu wird in dieser Arbeit eine Schwierigkeitsdefinition ermittelt, die von den Aufsehern angenähert werden soll.
Es werden zwei verschiedene Aufseher basierend auf logistischer Regression entwickelt. Der eine Aufseher beruht auf logistischer Regression mit einer unabhängigen Variablen, der andere Aufseher beruht auf logistischer Regression mit zwei unabhängigen Variablen.

Das zweite Ziel ist, den ISA und das entwickelte Lernverfahren in Adaptris im Hinblick auf das Spielerlebnis der Spieler zu testen und zu vergleichen.
Dazu wird zum einen eine Nutzerstudie erhoben und zum anderen werden Spiele simuliert. Anhand der Nutzerstudie und der simulierten Spiele wird ermittelt, wie die Spieler das Spielerlebnis mit den verschiedenen Algorithmen empfinden und wie gut die Algorithmen die Schwierigkeit an verschiedene Kompetenzen anpassen.

[1] Werden Personenbezeichnungen aus Gründen der besseren Lesbarkeit lediglich in der männlichen oder weiblichen Form verwendet, so schließt dies das jeweils andere Geschlecht mit ein.

[2] Adaptris: Tetris, bei dem die Fallgeschwindigkeit in jedem Zug neu adaptiert werden kann

1.3 Aufbau

Zu Beginn der Arbeit, in Kapitel 2, werden Grundlagen zum ISA, zur logistischen Regression und zur logistischen Regression als Lernalgorithmus erläutert. In Kapitel 3 wird der Aufbau von Adaptris aufgeführt, gefolgt von Kapitel 4, in welchem die Anwendung der Lernalgorithmen auf Adaptris erklärt wird. In Kapitel 5 werden mögliche Heuristiken der Lernalgorithmen vorgestellt und eine optimal spielende KI erläutert.
Um die Algorithmen vergleichen zu können, werden zum einen simulierte Spiele durchgeführt, deren Aufbau in Kapitel 6 beschrieben wird. Anschließend werden die Ergebnisse beschrieben und interpretiert. Zum anderen wird eine Nutzerstudie erhoben, deren Aufbau in Kapitel 7 dargestellt wird und deren Ergebnisse dort dargestellt und interpretiert werden.
Kapitel 8 liefert das Fazit der Arbeit, wonach in Kapitel 9 mit einem Ausblick abgeschlossen wird.

2 Grundlagen

In diesem Kapitel werden der ISA und die logistische Regression erläutert. Für letztere wird gezeigt, wie diese als Lernalgorithmus verwendet wird.

2.1 Interval Subdivision Agent

Der ISA wurde von Bengs und Brefeld entwickelt [3]. Die nachfolgenden Ausführungen beruhen auf der selben Quelle. Der Agent ist ein Algorithmus, der eine Schätzung der Kompetenz einer Testperson bei „speeded tests“ ermittelt.[1] Solche Tests umfassen mehrere Runden, bei denen das Feedback jeder Runde gespeichert wird. Damit wird die Schätzung der Kompetenz aktualisiert. Nach jeder Runde wird die Schwierigkeit der Testrunde an die neue Schätzung angepasst.
Die Schwierigkeitskonfiguration der Runden wird im ISA mit dem Parameter θ modelliert. Die geschätzte Schwierigkeitskonfiguration $\hat{\theta}$ entspricht der Schwierigkeit, bei der die Testperson die Testrunde zu 50% mit Erfolg und zu 50% ohne Erfolg abschließt.
Das Feedback ρ, welches nach jeder Runde generiert wird, nimmt folgende Werte mit beistehender Definition an:

- $\rho = 1$: Schwierigkeit der Runde war zu niedrig (*zu leicht*)
- $\rho = 0$: Schwierigkeit der Runde war *genau richtig*
- $\rho = -1$: Schwierigkeit der Runde war zu hoch (*zu schwer*).

Da sich die Kompetenz der Testperson über die Zeit verändern kann, beispielsweise durch Lerneffekte oder Müdigkeit, hat das aktuellste Feedback einen größeren Einfluss, genau 50%, auf die Schätzung $\hat{\theta}$.[2] Diese starke Gewichtung hängt mit der Annahme zusammen, dass bei einem Feedback *zu leicht*, zu einer Schwierigkeitseinstellung zur Schätzung $\hat{\theta}$, davon ausgegangen werden kann, dass für alle Schwierigkeitskonfigurationen $\theta < \hat{\theta}$ die Testrunde ebenso *zu leicht* ist. Selbiges gilt vice versa für das Feedback *zu schwer*. Das Feedback *genau richtig* hat keinen Einfluss auf die Schätzung.
Die Schätzung der optimalen Schwierigkeitskonfiguration zur Zeit t wird im Agenten durch die Funktion $w_t : [a, b] \to (0, \infty)$ modelliert, wobei a den Minimalwert der Schwierigkeitskonfiguration und b den Maximalwert darstellt. Wird ein Feedback $\rho_t = 1$ (*zu leicht*) zur Schätzung $\hat{\theta}$ ermittelt, lernt der Agent Information zum Punkt $\hat{\theta}$. Zusätzlich wird die Schätzung im Intervall $[a, \hat{\theta}]$ aktualisiert. Die Menge der Schätzungen, die aktualisiert werden, ist gegeben durch

$$A_t(\hat{\theta}_t) = \int_a^{\hat{\theta}_t} w_t(x)dx\,. \tag{2.1}$$

Für ein Feedback $\rho_t = -1$ (*zu schwer*) wird das Intervall $[\hat{\theta}, b]$ aktualisiert. Die Menge der Schätzungen ist definiert durch

$$B_t(\hat{\theta}_t) = \int_{\hat{\theta}_t}^{b} w_t(x)dx\,. \tag{2.2}$$

[1] speeded tests: Tests bei denen den Testpersonen homogen schwierige Aufgaben gegeben werden, die einfach zu lösen sind. Die Testpersonen werden anhand der benötigten Zeit, die Aufgaben zu lösen, bewertet.

[2] In dieser Arbeit wird davon ausgegangen, dass sich die Kompetenz eines Spielers über den kurzen Zeitraum eines Spiels nur vernachlässigbar verändert.

Für das Feedback $\rho_t = 0$ (*genau richtig*) wird die Schätzung nicht aktualisiert.

Der kontinuierliche Bereich zwischen dem Minimalwert a und dem Maximalwert b wird in Intervalle unterteilt. Um die Schwierigkeitseinstellung zur aktuellen Kompetenzschätzung $\hat{\theta}$ zu ermitteln, wird zunächst das Intervall ausgewählt, in dem der Wert liegt für den gilt:

$$A_t(\hat{\theta}_t) = \frac{1}{2} \int_a^b w_t(x) dx \,. \tag{2.3}$$

In dem ausgewählten Intervall wird dann der passende Wert für $\hat{\theta}$ durch Interpolation ermittelt.

2.2 Logistische Regression

Die Logistische Regression (auch „Logit-Modell“) wird häufig verwendet, um die Ergebnisse von nominal[3] oder ordinal[4] abhängigen Variablen zu modellieren [5]. Die lineare Regression ist in diesen Modellen nicht anwendbar, da die Wertemenge der linearen Funktion zum Teil nicht interpretierbar ist. Dies ist beispielsweise die Teilmenge $M = \mathbb{R} \setminus [0, 1]$ bei einer abhängigen Variablen y mit den Werten „trifft zu“ ($y = 1$) und „trifft nicht zu“ ($y = 0$). Die logistische Regressionskurve (Logit) passt sich dem interpretierbaren Wertebereich $W = [0, 1]$ hingegen an (siehe Abbildung 2.1).

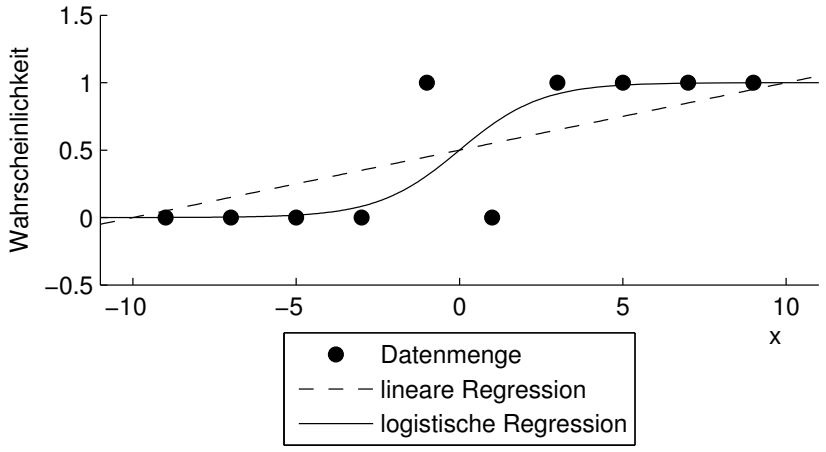


Abbildung 2.1: Lineare Regression, logistische Regression

Die logistische Regression wird in zwei Varianten unterschieden:

- Die *binär-logistische Regression* wird für nominal abhängige Variablen mit zwei Werten (*trifft zu* und *trifft nicht zu*) verwendet.
- Die *multinominale logistische Regression* wird verwendet, wenn die abhängige Variable mehr als zwei Werte annehmen kann, wie zum Beispiel Farben (Werte *rot*, *grün* und *schwarz*). [6]

Im Folgenden wird die binär logistische Regression erläutert. Die multinominale logistische Regression wird nicht in dieser Arbeit nicht verwendet.

[3] nominaler Datentyp: „Rein qualitative Merkmalsausprägungen ohne natürliche Ordnung“[4]; z.B. Geschlecht (männlich/ weiblich) oder dichotome Antwort vom Typ „ja“/ „nein“

[4] ordinaler Datentyp: „Qualitative Merkmalsausprägungen mit natürlicher Ordnung“[4]; Beispiel: Schulnoten (sehr gut, gut, befriedigend, ausreichend, mangelhaft, ungenügend)

Die (binär) logistische Regression ist ein Regressionsverfahren, bei welchem die Wahrscheinlichkeitsverteilung für das Eintreten des Ereignisses E, basierend auf den unabhängigen Variablen, ermittelt wird. Für die unabhängigen Variablen muss vorausgesetzt sein, dass keine Multikollinearität[5] besteht [6] und es darf keine Autokorrelation[6] vorliegen [8].
Das Ziel der logistischen Regression ist, für gegebene unabhängige Variablen die Eintrittswahrscheinlichkeit p des vorher definierten Ereignisses E zu ermitteln. Dazu müssen die Regressionskonstante b_0 und die Regressionsgewichte b_j bestimmt werden.
Im logistischen Regressionsmodell wird angenommen, dass die unabhängigen Variablen x_j linear auf eine nicht empirisch beobachtbare latente[7] Variable z wirken, die eine binäre Ausprägung der abhängigen Variablen y_k erzeugen kann [6]. Für einen Beobachtungsfall k wird die Variable z definiert durch Gleichung 2.4.

$$z_k = b_0 + \sum_{j=1}^{J} b_j \cdot x_{j,k} \tag{2.4}$$

mit
z = nicht beobachtbare latente Variable,
x = unabhängige Variable,
b_0 = Regressionskonstante,
b_j = Regressionsgewicht der unabhängigen Variablen,
j = Index der unabhängigen Variablen,
k = Index des Beobachtungsfalls.

Bei positiven Werten von z soll der Wert 1 = „Ereignis tritt ein“ und bei negativen z-Werten der Wert 0 = „Ereignis tritt nicht ein“ erzeugt werden. In Gleichung 2.5 wird der y-Wert (der Wert des Ereignisses E) für den Beobachtungsfall k dargestellt. [6]

$$y_k = \begin{cases} 1, z_k > 0 \\ 0, z_k \geq 0 \end{cases} \tag{2.5}$$

Die logistische Funktion 2.6 wird als Verknüpfungsfunktion eingesetzt, damit die errechneten z-Werte eine binäre Ausprägung erzeugen können [6]:

$$p = \frac{1}{1 + e^{-z}} \tag{2.6}$$

mit
p = Eintrittswahrscheinlichkeit von Ereignis E,
e = Eulersche Zahl.

Um die Funktion p zu bestimmen, kann der Vektor **b** mit der Maximum-Likelihood Methode ermittelt werden. Minka vergleicht sieben Ansätze in Bezug auf Laufzeit, von denen das Newton-Verfahren zusammen mit den Line-Search Algorithmen am besten abschneidet [9].
Aus diesem Grund wird in dieser Bachelorarbeit das Newton-Verfahren verwendet, welches hier nicht näher erläutert wird.

[5] Multikollinearität liegt vor, wenn zwischen den unabhängigen Variablen lineare Abhängigkeiten bestehen.
[6] Autokorrelation bedeutet, dass die Störvariablen in einem Regressionsmodell bei gegebenen unabhängigen Variablen zwischenzeitlich korreliert sind. [7]
[7] Eine latente Variable ist eine Variable, die nicht direkt gemessen werden kann, aber nach der Hypothese den beobachteten Variablen zugrunde liegt.

2.3 Logistische Regression als Lernalgorithmus

Die Logistische Regression (LR) ist in erster Linie kein Lernverfahren, sondern ein Regressionsverfahren, um die Wahrscheinlichkeit eines Ereignisses vorherzusagen. Um einen Lernalgorithmus auf der Grundlage der logistischen Regression zu entwickeln, muss zunächst eine Datenmenge vorhanden sein. Die Datenmenge muss so gewählt sein, dass eine logische Funktion bestimmt werden kann. Diese Datenmenge kann beispielsweise erzeugt werden, indem mit einer Stichprobe aus zufällig gewählten unabhängigen Variablen zugehörige Werte ermittelt werden.
Aus dieser Datenmenge wird eine initiale logistische Regressionskurve mit der Maximum-Likelihood-Methode ermittelt. Diese Datenmenge mit Logistischer Regressionsfunktion kann dann wie in Abbildung 2.1 dargestellt aussehen.

Der Lernalgorithmus soll den Wert eines unabhängigen Parameters (in Adaptris die Geschwindigkeit) bei gegebenen anderen unabhängigen Parametern (in Adaptris gegebenenfalls die Höhe der Spielsteine im Spielfeld) lernen, bei dem eine vorgegebene Wahrscheinlichkeit p (in Adaptris die Wahrscheinlichkeit, dass der Spieler den Stein schlecht setzt) erreicht wird.

Dazu wird die Formel

$$p = \frac{1}{1 + e^{-b_0 - b_1 x_1}} \tag{2.7}$$

umgeformt zu

$$x_1 = -\frac{\ln(\frac{1}{p} - 1) + b_0}{b_1}, \tag{2.8}$$

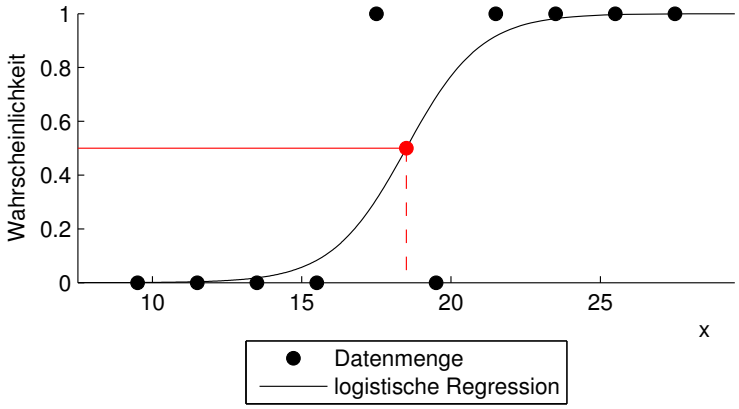
beziehungsweise im Falle von zwei unabhängigen Variablen wird

$$p = \frac{1}{1 + e^{-b_0 - b_1 x_1 - b_2 x_2}} \tag{2.9}$$

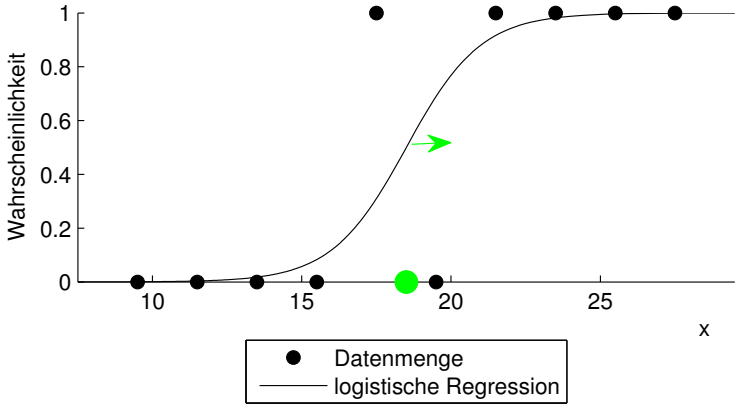
umgeformt zu

$$x_1 = -\frac{\ln(\frac{1}{p} - 1) + b_0 + b_2 x_2}{b_1}. \tag{2.10}$$

Alle Werte der Formeln 2.8 und 2.10 auf der rechten Seite sind gegeben und der Wert der unabhängigen Variablen x_1 wird ausgegeben. Mit diesem Wert wird eine Situation mit allen unabhängigen Variablen b_j vorgegeben und das Ereignis E tritt ein. Die Datenmenge wird um einen Punkt erweitert und die logistische Regression wieder angewendet. Dabei verschiebt sich die logistische Regressionskurve bei jeder Iteration, je nach Ereignis (0 oder 1), nach links oder rechts und der Parameter x_1 wird angepasst wie in Abbildung 2.2 für LR mit einer unabhängigen Variablen dargestellt.
In Abbildung 2.2a wird der Parameter x_1 ausgelesen bei gegebener Wahrscheinlichkeit $p = 50\%$. Der zugehörige x_1-Wert beträgt in der Abbildung etwa 18,5. Mit diesem Wert wird das nächste Ereignis ermittelt. In Abbildung 2.2b tritt das Ereignis 0 ein. Der Punkt (grün) wird der Datenmenge hinzugefügt, die logistische Regression wird erneut ausgeführt und die neue Regressionskurve ist leicht nach rechts verschoben im Vergleich zur Vorherigen. Der folgende x_1-Wert wird dementsprechend höher sein als der Letzte.



(a) Auslesen von x_1



(b) Datenmenge nach der Aktualisierung

Abbildung 2.2: Hinzufügen eines Datenpunktes bei logistischer Regression

3 Adaptris

Adaptris ist die Bezeichnung für adaptiertes Tetris. Anders als im originalen Spiel Tetris wird bei Adaptris die Spielgeschwindigkeit für jeden Zug neu angepasst. Es gibt mittlerweile unzählige verschiedene Varianten von Tetris. In diesem Kapitel werden die Regeln und das Setup in Adaptris beschrieben.

3.1 Aufbau Adaptris

Adaptris ist ein Spiel für genau einen Spieler. Das Spielfeld ist 10 Einheiten breit und 22 Einheiten (inklusive 2 versteckte Einheiten) hoch (siehe Abbildung 3.1).
Die Einheit der Fallgeschwindigkeit der Tetrominos[1] ist Einheiten/Sekunde. Im Spiel fällt der aktuelle Tetromino mit einer konstanten Geschwindigkeit herunter. Der Spieler kann den Tetromino während des Falles nach links und rechts bewegen, gegen und mit dem Uhrzeigersinn rotieren und er kann den Fall beschleunigen. Wie in der Abbildung 3.1 zu erkennen, wird links der Tetromino angezeigt, der als nächstes von oben herab fallen wird. Dieser wird Vorschautetromino genannt. Nach einem Zug[2] wird der nächste Vorschautetromino vom System zufällig gewählt und jeder der sieben Tetrominos (siehe Abbildung 3.2) wird mit gleicher Wahrscheinlichkeit ausgewählt. Außerdem wird ein Schatten angezeigt, wo der aktuelle Tetromino landen würde.
Das Spiel gilt als verloren, wenn ein Tetromino so platziert wurde, dass ein Teil des Tetrominos über der Höhe 20, im verdeckten Bereich oberhalb des Feldes, liegt.

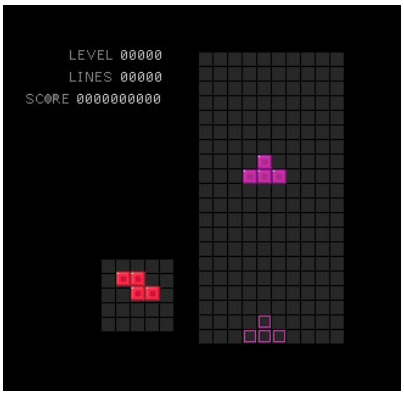


Abbildung 3.1: Der Spielfeldaufbau in Adaptris: Oben links Punkte und vervollständigte Linien, links unten der Vorschautetromino und rechts das Spielfeld.

3.2 Aufseher

Die Spielgeschwindigkeit in Adaptris wird von sogenannten Aufsehern gesteuert. Diese bekommen von einer Heuristik nach jedem Zug ein binäres Feedback (*zu schwer* oder *zu leicht*), je nachdem wie der Spieler den Stein gesteuert und gesetzt hat.
Die Aufseher haben verschiedene Methoden, um die Spielgeschwindigkeit an einen Spieler anzupassen.

[1] Ein Tetromino ist eine Form aus vier Quadraten und die Bezeichnung für einen Tetrisstein.
[2] Ein Zug ist der Spielabschnitt in dem ein Tetromino herunter fällt und vom Spieler platziert wird.

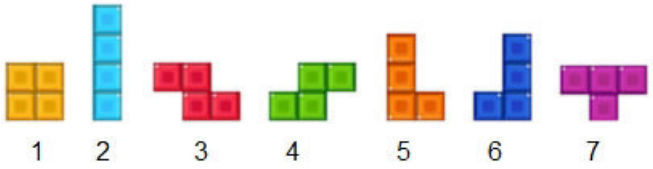


Abbildung 3.2: Die verschiedenen Tetrominos: Tetromino O (1), Tetromino I (2), Tetromino Z (3), Tetromino S (4), Tetromino L (5), Tetromino J (6), Tetromino T (7).

Ein Aufseher basiert auf dem ISA (Abschnitt 4.3), zwei auf der logistischen Regression (Abschnitt 4.5) und einer steuert die Geschwindigkeit reaktionsbasiert (Abschnitt 4.6).

3.3 Nutzerstudie

Der Aufbau von Adaptris ermöglicht, eine Nutzerstudie durchzuführen. Diese dient dazu festzustellen, wie Spieler auf das System reagieren. Die Testpersonen sollen drei Spiele mit verschiedenen Aufsehern spielen und werden nach jedem Spiel aufgefordert, einen kurzen Fragebogen auszufüllen. Eine genaue Beschreibung der Nutzerstudie befindet sich in Kapitel 7. Die Nutzerstudie wurde von Leischnig entwickelt und in dieser Arbeit erweitert [10].

3.4 Infrastruktur

Adaptris basiert auf dem open source Projekt „simple tetris clone“[3] und ist in TypeScript/Javascript geschrieben. Es wurde von Leischnig in eine Nutzerstudie eingebunden [10]. Das System läuft auf der Seite des Nutzers und ist in Typescript und HTML/CSS geschrieben und kann von jedem modernen Webbrowser aufgerufen werden.
Der Webserver ist ein Pythonskript, welches die HTML-/Typescript Seiten für das Spiel und die Nutzerstudie zur Verfügung stellt. Außerdem nimmt das Skript die Antworten der Fragebögen entgegen und speichert diese in einer sqlite3 Datenbank.
Der Server, der die Nutzerstudie hostet, ist Teil der Knowledge Mining & Assessment Group (KMA) der Technischen Universität Darmstadt. [10]

[3] Eine Implementierung von Tetris in verschiedenen Sprachen: https://code.google.com/p/simple-tetris-clone/, die jedoch nicht mehr aufrufbar ist (Stand 18.04.2015).

4 Anwendung der Lernalgorithmen auf Adaptris

In diesem Abschnitt wird gezeigt, wie die Spielschwierigkeit für ein faires, herausforderndes Spiel ermittelt wird und wie der ISA und die Logistische Regression als Lernalgorithmen auf das Spiel angewendet werden. Außerdem wird ein reaktionsbasierter Aufseher vorgestellt, der das Spiel beobachtet und die Schwierigkeit einstellt.

4.1 Flow Modell

Um die Spielschwierigkeit auf den einzelnen Spieler anzupassen, muss zunächst herausgefunden werden, wie schwer das Spiel sein muss, damit der Spieler weder gelangweilt noch überfordert ist. Csíkszentmihályi hat hierzu das Flow Modell (siehe Abbildung 4.1) entwickelt, welches sich sehr gut auch auf Spiele anwenden lässt. Dieses Modell besagt, dass einem Spieler, der unterfordert ist, langweilig wird und dass Überforderung zu Aufregung führt. Um den Spielspaß über die gesamte Zeit zu optimieren, sollte sich der Spieler in der „Flow-Zone“ befinden. Dies bedeutet, dass die Spielschwierigkeit für einen Fortgeschrittenen Spieler schwerer sein sollte als für einen Anfänger. [2]
Da ein Tetrisspiel in der Regel nicht lange dauert, ist der Lerneffekt des Spielers im Spiel zu vernachlässigen. Es wird für jeden Spieler die Schwierigkeit angenähert, die für seine Spielstärke in der „Flow-Zone“ ist.

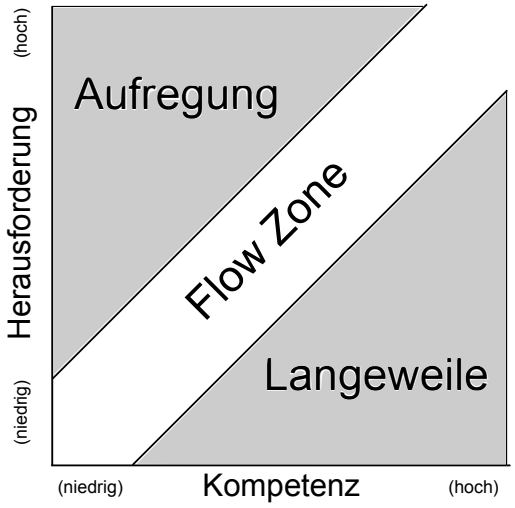


Abbildung 4.1: Flow Modell nach Csíkszentmihályi

4.2 Schwierigkeitsdefinition in Tetris

In Tetris ist die Hauptkomponente der Spielschwierigkeit die Fallgeschwindigkeit der Tetrominos. Fällt ein Tetromino schnell, ist das Platzieren dieses Tetrominos schwerer als bei einem langsam fallenden Tetromino.

Eine weitere Komponente der Spielschwierigkeit ist die Bauhöhe der Tetrominos im Spielfeld. Sie ist definiert als die Höhe des aktuell höchstgelegenen Tetrominoteiles. Ist diese Höhe insgesamt sehr hoch, hat der Spieler weniger Zeit zu reagieren und zu agieren als bei einem freien Spielfeld. Auf die Höhe der Tetromino im Spielfeld haben die Algorithmen keinen direkten Einfluss, weshalb die Anpassung der Spielschwierigkeit über die Geschwindigkeit geschieht.

Bei dieser Herangehensweise wird die Form des aktuellen Tetrominos vernachlässigt. Außerdem wird nicht darauf eingegangen, wie gut der Tetromino auf das aktuelle Spielfeld passt.

4.3 Anwendung des ISA Algorithmus in Tetris

Der ISA wurde von Leischnig[10] in Form des ISA-Aufsehers in Adaptris eingebunden.
Die Schwierigkeit ist festgelegt als die Zeit, die benötigt wird, um einen Tetromino zu platzieren. Die Fallgeschwindigkeit wird demnach in Abhängigkeit der maximalen Höhe der Tetrominos eingestellt.
In der Arbeit von Leischnig werden verschiedene Heuristiken vorgestellt, um den Spielzug des Spielers zu bewerten, von denen jedoch nur ein Teil in der vorliegenden Arbeit angewendet wird. In dieser Arbeit wird die Heuristik für Feedback realer Spieler (Abschnitt 5.1.9) verwendet. Diese gibt Feedback für einen Zug in Form von *zu schwer*, der Wert wird für den ISA zu -1, oder *zu leicht*, der Wert wird für den ISA zu 1, zurück. Der ISA könnte auch das Feedback genau richtig (Feedback = 0) verwenden, jedoch kann die logistische Regression als Lernalgorithmus diesen Wert nicht verwenden. Um einen guten Vergleich zwischen dem ISA und der logistischen Regression als Lernalgorithmus herstellen zu können, wird dieselbe Heuristik verwendet.

Die bereits implementierten Heuristiken für den ISA sind bei der Entwicklung nicht an Spielern getestet worden und sind nur bedingt zielführend, um eine qualifizierte Aussage darüber zu treffen, ob ein Zug gut oder schlecht ist. Eine auf den ISA abgestimmte Heuristik, die den Wert *genau richtig* mit einbezieht, wird kurz im Ausblick diskutiert.
Der Algorithmus wird angewendet, wie im Abschnitt ISA beschrieben (Abschnitt 2.1).

4.4 Ermittlung der vorgegebenen Rate für schlechte Züge

Für die logistische Regression als Lernalgorithmus wird eine Vorgabe benötigt, mit welcher Wahrscheinlichkeit das Ereignis $E = \textit{schlechter Zug des Spielers}$ eintreffen soll. In diesem Abschnitt wird vorgestellt, wie in dieser Arbeit eine Schwierigkeitsdefinition gefunden wird, bei der die Spieler ein möglichst faires und herausforderndes Spiel erleben.

Die Aufseher geben die Spielgeschwindigkeit vor. Ein Spieler macht, abhängig von dieser Geschwindigkeit und der Höhe der Tetrominos im Spielfeld (Schwierigkeitsdefinition in Tetris 4.2) und abhängig von seiner Kompetenz, *gute Züge* (der Spieler setzt den Stein gut) oder *schlechte Züge* (der Spieler setzt den Stein nicht gut). Die Evaluierung, ob ein guter oder schlechter Zug gemacht wurde, wird im Abschnitt 5.1 beschrieben.

Gute und schlechte Züge sollen in einem gewissen Verhältnis stehen. Kein Spieler funktioniert deterministisch und jede Situation ist anders. Aus diesem Grund wird davon ausgegangen, dass der Spieler zu einer gegebenen Schwierigkeitskonfiguration eine Chance hat, den aktuellen Tetromino gut zu platzieren (guter Zug) oder ihn schlecht zu platzieren (schlechter Zug).

Insgesamt soll der Spieler ein faires Spielerlebnis erleben. Um dies zu erreichen, darf der Spieler nicht unterfordert sein. Daher sollte die Schwierigkeit so eingestellt sein, dass der Spieler ab und zu einen

schlechten Zug macht. Das Spiel darf den Spieler auch nicht überfordern. Deshalb muss die Schwierigkeit so eingestellt sein, dass der Spieler auch in der Lage ist, gute Züge zu machen.

Es wird ermittelt, in welchem Verhältnis gute Züge zu schlechten Zügen stehen müssen. Um Herauszufinden, bei wie viel Prozent schlechter Züge ein Spiel herausfordernd und gleichzeitig gut spielbar ist, wurden in dieser Arbeit verschiedene Tests gemacht. Für diese Tests ist eine optimal spielende KI notwendig, welche in Abschnitt 5.2 vorgestellt wird. Bei den folgenden Tests wird davon ausgegangen, dass diese KI unendlich lange spielt.

Mit der optimalen KI wurden Spiele gespielt mit einer Vorgabe von 0%, 4%, 5%, 7,5%, 10% und 50% Chance einen schlechten Zug zu machen. Da ein zufälliger Zug auch gut sein kann, ist ein schlechter Zug definiert als ein zufälliger Zug der 70% schlechtesten möglichen Züge. Die Stichprobe umfasst je 1000 Spiele für die Chancen 4%, 5%, 7,5%, 10% und 50% und 4 Spiele für 0%. Die Stichprobe der 0%-Vorgabe ist sehr gering. Dies liegt zum einen daran, dass die Simulation dieser Spiele sehr lange dauert, da sie optimal gespielt werden und die Spiele lange nicht verloren werden. Zum anderen wurde auf eine größere Stichprobe der 0%-Vorgabe verzichtet, da sie nicht weiter von Belang ist, wie im Folgenden erläutert wird.

Die vorgegebenen Chancen werden durch Zufallswerte für jeden Zug realisiert. Es kommt dazu, dass die Vorgaben nicht eingehalten werden und sich das Verhältnis *schlechter Züge* zu *Züge insgesamt* im Verhältnis 0% bis 100% bewegt. Der größte Anteil liegt jedoch zwischen 4% und 50%.

Das Auswertungskriterium ist die Spiellänge. Der grundlegende Gedanke ist, dass in einem kurzen Spiel der Spieler überfordert ist. Aufgrund vieler schlechter Züge wird das Spiel schnell verloren. In einem Spiel, das sehr lange dauert, wird hingegen davon ausgegangen, dass der Spieler unterfordert ist und somit das Spiel für ihn langweilig ist. In einem langen Spiel hat der Spieler sehr wenig schlechte Züge gemacht. Die Spiellänge wird in der Einheit „vervollständigte Linien“ gemessen.
In Abbildung 4.2 sind die Spiellängen zu den zugehörigen Verhältnissen $\frac{\textit{schlechte Züge}}{\textit{Züge insgesamt}}$ in Boxplotdiagrammen dargestellt. In grau stehen die zugehörigen Mediane des Anteils schlechter Züge.

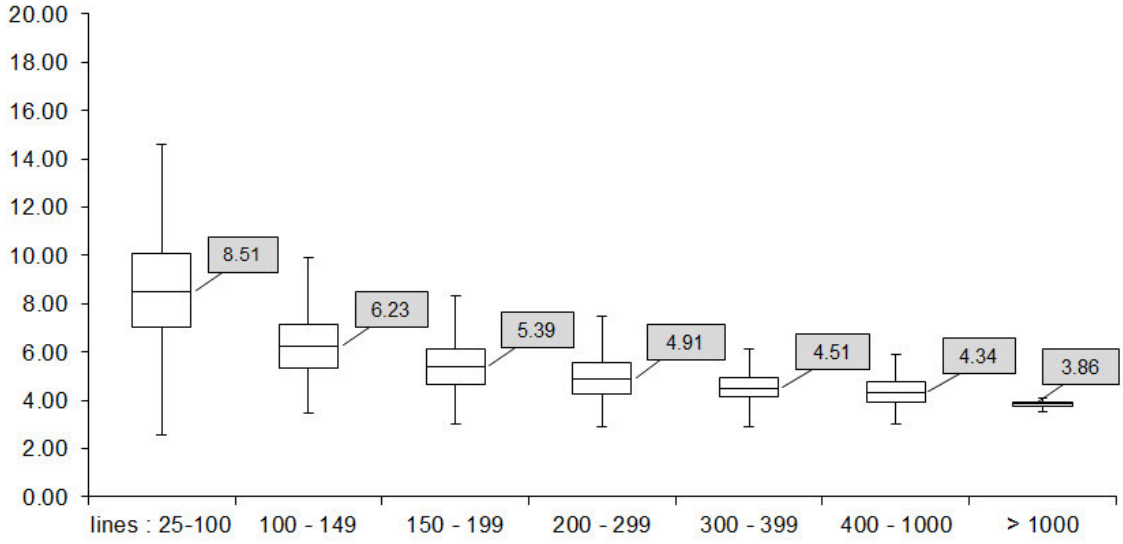


Abbildung 4.2: Verteilung der Verhältnisse $\frac{\textit{schlechte Züge}}{\textit{Züge insgesamt}}$ in Bezug auf die Spiellänge

In vielen Tetrisspielen wird das Level, und somit auch die Geschwindigkeit, jede 10 vervollständigten Linien um einen Tick pro Sekunde erhöht. Das Maximallevel ist 30. In einem Spiel, bei dem das Maximallevel erreicht wird, werden dementsprechend mindestens 300 Linien vervollständigt.

Adaptierte Spiele, bei denen weniger als 10 Linien vervollständigt werden, sind als zu kurz anzusehen. Der Spieler hat nur wenig Spielerlebnis.
Bei Spielen, bei denen mehr als 400 Linien vervollständigt werden, wird davon ausgegangen, dass der Spieler keine große Herausforderung erlebt.
In Adaptris wird eine Spiellänge von 150 bis 199 vervollständigten Linien als Vorgabe gewählt. Der Median dieser Verteilung liegt bei dem Verhältnis 5,39%. Dieser Wert ist die vorgegebene Rate schlechter Züge bei den Aufsehern basierend auf logistischer Regression.

4.5 Anwendung der Logistischen Regression in Tetris

Die logistische Regression wurde in dieser Arbeit auf zwei verschiedenen Arten als Aufseher entwickelt und implementiert. Ein Aufseher hat nur eine unabhängige Variable (UV), diese ist die Fallgeschwindigkeit der Tetrominos. Der andere Aufseher hat zwei unabhängige Variablen: die Fallgeschwindigkeit und die Höhe der Tetrominos im Spielfeld (siehe Schwierigkeitsdefinition in Tetris 4.2).
Die logistische Regression kann nur dann angewendet werden, wenn schon eine gewisse Datenmenge gegeben ist. Für weniger als vier Punkte ist nicht einmal die logistische Regression mit einer UV definiert. Außerdem darf es bei einer UV keine Grenze $\hat{x}$ geben, sodass

$$\forall x < \hat{x}; E = 0 \text{ und} \tag{4.1}$$
$$\forall x > \hat{x}; E = 1, \tag{4.2}$$

da sonst eine Schrittfunktion angenähert wird, welche eine unendlich große Steigung an einer Stelle hat. Für die komplett entgegengesetzten Ergebnisse gilt dasselbe.
Für die logistische Regression mit 2 UVn darf keine Ebene parallel zur Ergebnisachse die Ergebnisse teilen können.
Um die Aufseher zu initialisieren, muss zunächst eine hinreichende Datenmenge vorhanden sein. Diese kann mit einer zufälligen Stichprobe (siehe Logistische Regression als Lernalgorithmus 2.3) initialisiert werden. Dazu würden im Spiel zunächst Tetrominos mit zufälliger Geschwindigkeit oder mit vorher bestimmten Geschwindigkeiten fallen, bis eine logistische Regression durchgeführt werden kann. Dies ist jedoch schlecht zu praktizieren, da der Spieler zum einen ein merkwürdiges Spielerlebnis hätte, wenn Tetrominos mit stark verschiedenen Geschwindigkeiten fielen. Zum anderen kann es viele Züge dauern, bis die oben genannten Anforderungen an die Datenmenge erreicht sind.
Aus diesem Grund wird statt einer zufälligen Stichprobe, deren Ereignisse E ermittelt werden, eine initiale Datenmenge vorgegeben mit Punkten, die über den Definitionsbereich verteilt sind und deren Ereignisse bereits festgelegt sind.

4.5.1 Eine unabhängige Variable

Der Aufseher basierend auf logistischer Regression mit einer UV (LR1UV) betrachtet nur die Fallgeschwindigkeit als UV. Die abhängige Variable ist das Ereignis E mit den Werten 1 (*zu schwer*) und 0 (*zu leicht*). Es ist zu beachten, dass diese Feedbackwerte anders sind als beim ISA.
In Abbildung 4.3 ist der Aufbau der initialen Datenmenge dargestellt. Die x-Achse gibt die Geschwindigkeit an und die y-Achse die Wahrscheinlichkeit des Eintreffens vom Ergebnis „Zug zu schwer“. Die schwarzen Punkte markieren die Punkte der initialen Datenmenge. Die dicken schwarzen Punkte stellen jeweils drei Punkte dar. Die schwarze Linie ist die logistische Regressionsfunktion.

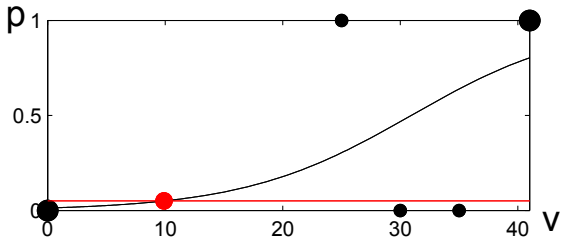


Abbildung 4.3: Initiale Datenmenge des LR1UV mit logistischer Funktion und Ablesen der Geschwindigkeit

Auf die Datenmenge wird die logistische Regression angewendet. Dadurch entsteht die Funktion

$$p = \frac{1}{1 + e^{-b_0 - b_1 v}} \tag{4.3}$$

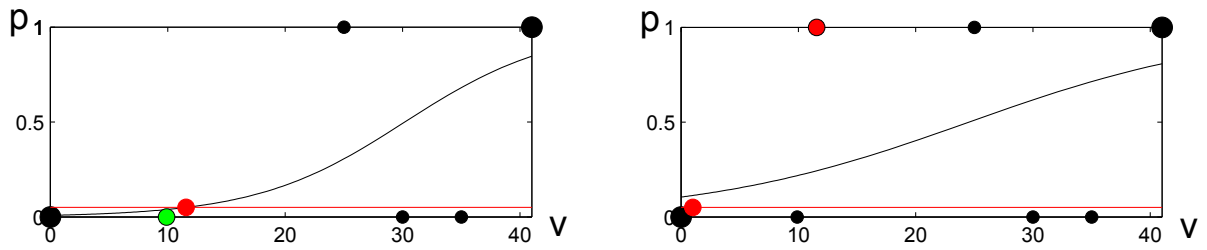
mit aus der LR ermittelten Werten für b_0 und b_1.
In der Abbildung 4.3 ist außerdem zu sehen, wie die Geschwindigkeit v für den nächsten Zug ausgewählt wird. Die rote Linie markiert den optimalen Prozentsatz schlechter Züge von 5,39%, der in Abschnitt 4.4 ermittelt wurde. An dem Schnittpunkt der roten Linie und der Regressionskurve markiert ein roter Punkt die Stelle, an der die Geschwindigkeit für den nächsten Zug abgelesen wird. Diese beträgt in der Abbildung etwa 10 Einheiten pro Sekunde.
Im Algorithmus wird die Geschwindigkeit v ermittelt, wie in Abschnitt 2.3 beschrieben, mit Einsetzen der vorgegeben Chance $p = 5,39\%$ in die Formel 2.8:

$$v = -\frac{\ln(\frac{1}{0,0539} - 1) + b_0}{b_1}. \tag{4.4}$$

b_0 und b_1 sind aus der logistischen Funktion gegebene Parameter.
Falls $v < 1$ wird $v = 1$ gesetzt. Eine Minimalgeschwindigkeit größer 0 muss definiert werden, da sonst negative Geschwindigkeiten ermittelt werden können. Solche Geschwindigkeiten kann Adaptris nicht verwerten.

Für den Zug wird die Fallgeschwindigkeit v eingestellt und der Tetromino im Zug vom Spieler platziert. Eine Heuristik, die im nächsten Kapitel vorgestellt wird, wertet nach dem Zug aus, ob der Spieler einen schlechten Zug gemacht hat. Dann gibt die Heuristik dem Aufseher das Ereignis $E = 1$ (*zu schwer*) zurück. Für einen guten Zug gibt sie $E = 0$ (*zu leicht*) zurück.
Aus der Geschwindigkeit und dem Ereigniswert setzt sich ein Punkt zusammen, der der Datenmenge hinzugefügt wird. Die logistische Regression wird auf die Datenmenge angewendet und eine neue Geschwindigkeit wird für den nächsten Zug ermittelt.

Zu beachten ist, dass sich bei einem Update die logistische Regressionsfunktion je nach Feedback nach links oder rechts verschiebt. In Abbildung 4.4a stellt der grüne Punkt ein Feedback „Ereignis *zu leicht*“ für den letzten Zug dar. Dadurch verschiebt sich die initiale Regressionsfunktion nach rechts. Die nächste Geschwindigkeit wird daher ein wenig schneller sein. In dem Beispiel beträgt die nächste Geschwindigkeit etwa 11,5 Einheiten pro Sekunde.
In der Abbildung 4.4b wird dargestellt, wie sich die Kurve nach einem Feedback „Ereignis *zu schwer*“ verschiebt. Der rote Punkt mit schwarzem Rand wurde der Datenmenge hinzugefügt.
Da es im Wertebereich $v > 1$ keinen Schnitt mit dem optimalen Prozentsatz gibt, wird die Geschwindigkeit für den nächsten Zug auf die Minimalgeschwindigkeit von einer Einheit pro Sekunde festgelegt.



(a) Update nach einem Feedback „zu leicht“ **(b)** Update nach einem Feedback „zu schwer“

Abbildung 4.4: Update der logistischen Regression

Auffällig ist, dass sich die Kurve für ein Feedback *zu schwer* deutlich mehr verschiebt als bei einem Feedback *zu leicht*. Das liegt daran, dass der ein neuer Punkt für ein das Feedback *zu leicht* bereits deutlich näher an der Regressionsfunktion liegt als ein neuer Punkt für das Feedback *zu schwer*.
Eine Anpassung der Geschwindigkeit nach oben dauert daher länger als eine Anpassung nach unten.

4.5.2 Zwei unabhängige Variablen

Der Aufseher basierend auf logistischer Regression mit 2 UVn (LR2UV) betrachtet zusätzlich zur UV Fallgeschwindigkeit v auch die Höhe h der Tetrominos im Spielfeld als zweite UV. Auch bei der LR2UV ist die abhängige Variable das Ereignis E mit den Werten 1 (*zu schwer*) und 0 (*zu leicht*). Die initiale Datenmenge ist durch die schwarzen Punkte in Abbildung 4.5 gekennzeichnet. Die Funktion in der Abbildung ist die initiale logistische Regressionsfunktion. Die rote Linie markiert die Geschwindigkeiten, die bei gegebenem h für einen Zug ausgewählt werden.

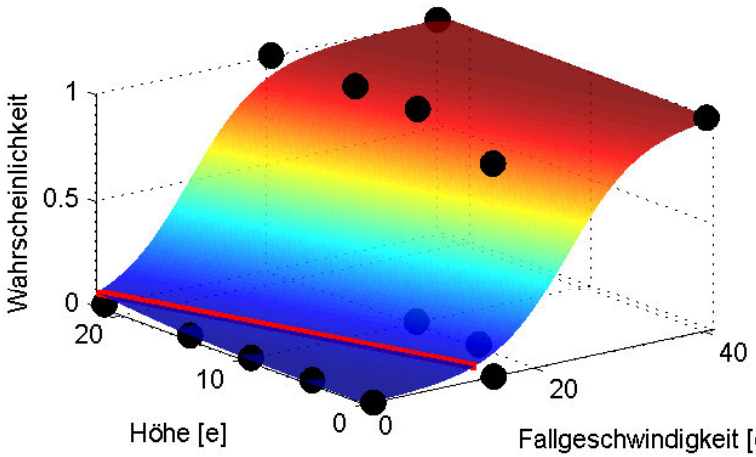


Abbildung 4.5: Initiale Datenmenge des LR2UV mit logistischer Funktion und Ablesen der Geschwindigkeit

Auf die Datenmenge wird die logistische Regression angewendet. Dadurch entsteht die Funktion

$$p = \frac{1}{1 + e^{-b_0 - b_1 v - b_2 h}}. \tag{4.5}$$

Die Werten b_0, b_1 und b_2 sind aus der logistischen Funktion gegeben.
In der Abbildung ist außerdem zu sehen, wie die Geschwindigkeit v für den nächsten Zug ausgewählt

wird. Für eine gegebene Höhe h der Tetrominos im Spielfeld und der Wahrscheinlichkeit 0.0539, wird der Schnittpunkt mit der Funktion gesucht. Dieser liegt für unterschiedliche Werte von h auf der roten Geraden.

Im Algorithmus wird die Geschwindigkeit v ausgerechnet wie in Formel 2.10 beschrieben. Durch Einsetzten von $p = 5.39\%$ wird diese umgeformt zu

$$v = -\frac{\ln(\frac{1}{0{,}0539} - 1) + b_0 + b_2 h}{b_1}. \tag{4.6}$$

Die Werte b_0, b_1 und b_2 sind aus der logistischen Funktion gegeben und h ist aus dem aktuellen Spielfeld ablesbar.
Falls $v < 1$ wird $v = 1$ gesetzt.

Für den Zug wird die Fallgeschwindigkeit v eingestellt und der Tetromino im Zug vom Spieler platziert. Eine Heuristik, die im nächsten Kapitel vorgestellt wird, wertet nach dem Zug aus, ob der Spieler einen schlechten Zug gemacht hat. Dann gibt die Heuristik dem Aufseher das Ereignis $E = 1$ (*zu schwer*) zurück oder $E = 0$ (*zu leicht*) für einen guten Zug.
Aus der der Geschwindigkeit, der Höhe vor dem Zug und dem Ergebnis setzt sich ein Punkt zusammen, der der Datenmenge hinzugefügt wird. Die logistische Regression wird auf die Datenmenge angewendet und eine neue Geschwindigkeit wird für den nächsten Zug ermittelt.

Die Verschiebung der Funktion nach einem Update ist analog zur Verschiebung der logistischen Regression mit einer Variablen. Die Anpassung an eine langsame Geschwindigkeit ist ebenso wie beim LR1UV-Aufseher schneller als an eine höhere Geschwindigkeit.

4.6 Reaktionsbasierter Aufseher

Dem reaktionsbasierten Aufseher liegt kein Lernalgorithmus zugrunde und wird auch in der Auswertung nicht mit den anderen Aufsehern verglichen. Er wurde im Umfang dieser Arbeit entwickelt, um die Spielstärke der Spieler einschätzen zu können.
Der Aufseher funktioniert reaktionsbasiert. Das bedeutet, dass er unabhängig vom Spielfeld nur auf den letzten Spielzug des Spielers reagiert. Er soll die Spielgeschwindigkeit so anpassen, dass der Spieler zu 5,39% einen schlechten Zug macht. Er hat eine Startgeschwindigkeit von 10 Einheiten pro Sekunde und ändert die Geschwindigkeit wie folgt:

$$v_{neu} = v_{alt} \cdot 1{,}0234 \textit{ für Feedback } E = 0 \textit{ (zu leicht)} \tag{4.7}$$

$$v_{neu} = v_{alt} \cdot 0{,}6667 \textit{ für Feedback } E = 0 \textit{ (zu schwer)} \tag{4.8}$$

Daraus ergibt sich, dass jedes Mal wenn ein schlechter Zug bei Geschwindigkeit $\hat{v}$ gemacht wird, der Spieler 17,55 gute Züge in Folge machen muss, um die Geschwindigkeit $\hat{v}$ wieder zu erreichen.
Die Geschwindigkeit pendelt sich daher, abhängig von der Kompetenz des Spielers, so ein, dass er ein Verhältnis von einem schlechten Zug zu 17,55 guten Zügen macht. Dieses Verhältnis ist gegeben durch die 5,39% Vorgabe schlechter Züge $(1/(1 + 17{,}55) = 5{,}39\%)$.
Die Anpassung der Geschwindigkeit bei $E = 1$ ist frei gewählt und wird auf $v_{neu} = v_{alt} \cdot 0{,}6667$ festgelegt, was einer Geschwindigkeitssenkung auf $\frac{2}{3}$ der vorherigen Geschwindigkeit entspricht. Um das gegebene

Verhältnis anzunähern, muss bei $E = 0$ die Geschwindigkeit bei jedem Zug um 2,34% ($v_{neu} = v_{alt} \cdot 1,0234$) erhöht werden, wie in der folgenden Berechnung gezeigt wird.

$$\hat{v} = \hat{v} \cdot \frac{2}{3}(1 + \frac{p}{100})^{17,55} \tag{4.9}$$

$$\sqrt[17,55]{\frac{3}{2}} = 1 + \frac{p}{100} \tag{4.10}$$

$$p = (\sqrt[17,55]{\frac{3}{2}} - 1) \cdot 100 \tag{4.11}$$

$$p = 2,34\% \tag{4.12}$$

5 Heuristiken für binäres Feedback und optimale KI

Um binäres Feedback zu generieren muss eine Heuristik bestimmen, ob ein Spieler mit einem Zug überfordert ist (Feedback *zu schwer*) oder unterfordert ist (Feedback *zu leicht*). Dafür ist es notwendig zu wissen, ob ein Zug gut oder schlecht ist. In einem guten Zug werden beispielsweise in der Regel keine neuen Lücken gebaut. Um die Qualität eines Zuges zu bewerten, werden im Folgenden verschiedene Heuristiken vorgestellt und am Ende zu einer zusammengeführt.
Ein Feedback *genau richtig* kann von der logistischen Regression nicht verarbeitet werden und wird daher nicht verwendet.
In diesem Kapitel wird außerdem eine optimal spielende KI vorgestellt, welche auf einer optimalen Feldbewertungsfunktion basiert.

5.1 Heuristiken zur Bewertung eines Spielzuges

Der ISA und das Lernverfahren mittels logistischer Regression basieren beide auf binärem Feedback.
Dieses Feedback wird in Adaptris nach jedem Zug ermittelt und dem Aufseher übergeben. Das Feedback kann die Werte *zu leicht* und *zu schwer* annehmen. Bei dem ISA kann zusätzlich noch ein Wert *genau richtig* verwendet werden. Da in dieser Arbeit die Verfahren möglichst gut vergleichbar sein sollen, wird für beide Verfahren die selbe Heuristik verwendet, die nur die beiden erstgenannten Werte zurückgibt.

5.1.1 Fallbewertung

Fallbewertungsfunktionen geben die Möglichkeit, den Spieler zu beobachten, während dieser den Stein bewegt.
Ein Spieler, der einen Tetromino beschleunigt, ist nicht stark gefordert. Die Fallgeschwindigkeit ist zu gering um den Spieler zu fordern. Die Fallbeschleunigungsheuristik wertet einen Zug als *zu leicht*, wenn der Spieler einen Tetromino über die Hälfte des Weges zur Endposition beschleunigt hat. Diese Heuristik wurde bereits von Leischnig implementiert und kann im Original auch die Werte *genau richtig* und *zu schwer* zurückgeben [10]. Dies wird jedoch im Folgenden nicht weiter betrachtet, da diese Werte nicht genutzt werden. Außerdem wurde der Parameter geändert, ab wie viel Prozent Fallbeschleunigung die Heuristik greift. Im Original wurde der Wert *zu leicht* ab 20% Fallbeschleunigung (zeitlich gesehen) zurückgegeben. Dies entspricht einer Abkürzung von mindestens 20% des Weges, den der Tetromino vertikal zurückgelegt hat.

Die Steuerung in Adaptris ist so, dass nach dem Platzieren eines Tetrominos, der am Ende des Falls beschleunigt wird, der nächste Tetromino direkt oben am Spielfeld auftaucht und auch beschleunigt wird. Da die Reaktion des Spielers auf diesen Effekt eine gewisse Zeit in Anspruch nimmt, wird der folgende Tetromino manchmal unabsichtlich beschleunigt. In der bereits implementierten Heuristik, wurde der Tetromino als beschleunigt angesehen, wenn er mehr als 20% der Strecke beschleunigt wurde. Dies führt dazu, dass Züge ohne *absichtliche* Beschleunigung von der Fallbewertung als beschleunigt betrachtet werden. In dem Fall wird gegebenenfalls fälschlicherweise der Wert *zu leicht* von der Heuristik zurückgegeben. Aus diesem Grund wurde die Schwelle in dieser Arbeit auf 50% angehoben.

5.1.2 Feldbewertung

Feldbewertungsfunktionen gehen davon aus, dass ein Zug für den Spieler *zu leicht* war, wenn der Stein nach ihrer Heuristik gut gesetzt ist. Dies bedeutet, dass der Spieler genug Zeit hatte zu überlegen, wo der aktuelle Tetromino gut passt und auch genug Zeit hatte, den Tetromino nach seinem Plan zu platzieren. Der Wert *zu schwer* wird zurückgegeben, wenn der Stein schlecht gesetzt wurde. Es wird angenommen, dass der Spieler bei mehr verfügbarer Zeit einen guten Zug hätte machen können.
Um zu evaluieren, ob der Tetromino gut oder schlecht gesetzt wurde, wird die Feldbewertung des Spielfeldes vor dem Zug mit der Feldbewertung des Spielfeldes nach dem Zug verglichen.

5.1.3 Feldbewertung mit Lücken

Zum Verdeutlichen, wie eine Feldbewertungsheuristik funktioniert, wird eine von Leischnig implementierte Heuristik vorgestellt [10].
Die Heuristik gibt dem Feld eine Bewertung, basierend auf der Anzahl der Lücken im Feld. Hat ein Spielfeld nach dem Zug mehr Lücken als vorher, ist die Bewertung nach dem Zug schlechter als vorher und die Heuristik gibt für den beschriebenen Zug den Wert *zu schwer* zurück. Sind weniger Lücken als vorher vorhanden, wird der Wert *zu leicht* zurückgegeben. Bei derselben Anzahl Lücken wird je nach Verwendung der Heuristik entweder der Wert *genau richtig* zurückgegeben oder eine andere Heuristik tritt in Kraft.

In einigen Fällen ist es nicht möglich, einen Zug zu machen, ohne neue Lücken zu bauen. Mit dieser Heuristik wäre es dann nicht möglich, das Feedback *zu leicht* entsprechend einem guten Zug zu bekommen. Außerdem gibt es Spielsituationen in denen keine Lücken dazu kommen können, hier ist nur ein Feedback *zu leicht* möglich. Daraus lässt sich schließen, dass sich alleine auf Grund dieser Feldbewertung keine fundierte Aussage über die Güte eines Zuges bestimmen lässt. In dieser Arbeit wird eine andere Heuristik verwendet.

5.1.4 Feldbewertung nach Yiyuan

Die Feldbewertungsfunktion, die in Adaptris genutzt wird, wurde 2013 von Lee Yiyuan entwickelt. Die Heuristik geht von vier Parametern aus. Diese sind

1. die aufsummierte Höhe aller Spalten,
2. die Anzahl vervollständigter Linien,
3. die Anzahl der Löcher und
4. die Unebenheit.[11]

Die aufsummierte Höhe wird ermittelt, indem die Höhe aller Spalten ermittelt und summiert wird. Für ein gutes Feld ist diese minimal. Beispielhafte Berechnung für Abbildung 5.1a: $hoehe = 48 = 3+5+5+5+6+6+5+4+4+5$.

Die Anzahl vervollständigter Linien ist die Anzahl vervollständigter Linien mit dem aktuellen Zug. Im eigentlichen Sinne ist dies kein Parameter einer reinen Feldbewertungsfunktion, da für diesen Wert die Änderung vom Feld vor dem Zug mit einbezogen wird.
Für einen guten Zug ist dieser Parameter maximal. Beispiel Abbildung 5.1b: $linien = 2$.

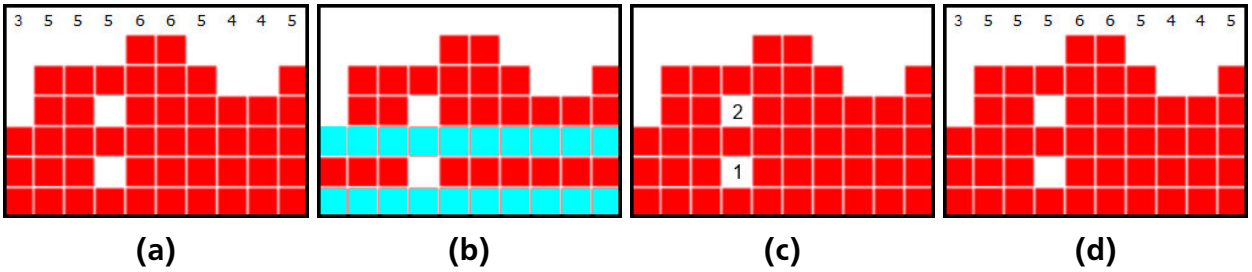


Abbildung 5.1: Parameter der Feldbewertung nach Yiyuan (Quelle: [11])

Die Anzahl der Löcher ist der Wert der Lücken im Spielfeld. Für ein gutes Feld ist diese minimal. Beispiel Abbildung 5.1c: $loecher = 2$.

Die Unebenheit eines Feldes wird ermittelt, indem die Höhenunterschiede aller benachbarten Spalten des Feldes aufsummiert werden. Da ein ebenes Feld mehr Möglichkeiten offen lässt weitere Tetrominos gut, beziehungsweise dicht, zu verbauen, ist die Unebenheit für ein gutes Feld minimal. Beispielhafte Berechnung für Abbildung 5.1d: $unebenheit = 6 = 2+0+0+1+0+1+1+0+1$.

Um die Heuristik zusammenzusetzen, wird die Bewertung r eines Feldes definiert als:

$$r = a \cdot hoehe + b \cdot linien + c \cdot loecher + d \cdot unebenheit \tag{5.1}$$

mit $a = -0.510066$, $b = 0.760666$, $c = -0.35663$ und $d = -0.184483$.
Diese Parameter wurden von Yiyuan durch einen genetischen Algorithmus optimiert. [11]
Mit dieser Feldbewertung werden sehr gute Ergebnisse erzielt. Diese werden im Abschnitt Optimale KI (5.2) erläutert. Diese Feldbewertungsheuristik wird aufgrund der guten Ergebnisse im Folgenden als optimal betrachtet.

5.1.5 Ein-Zug-Lookahead

Der Ein-Zug-Lookahead lässt sich mit jeder Feldbewertung kombinieren. Er simuliert jeden möglichen Zug, den der Spieler machen kann. Dies bedeutet, dass der aktuelle Tetromino auf höchster Höhe jede mögliche Rotation mit jeder möglichen x-Position annimmt und von dort aus herunter fällt. Ausgenommen von der Simulation sind daher Züge, bei denen ein Tetromino um einen Vorsprung herum gelenkt wird (siehe Abbildung 5.2).

Durch den Lookahead entstehen je nach Anzahl äquivalenter Rotationen des Tetrominos 9 (Tetromino O), 17 (Tetrominos I, S und Z) oder 34 (Tetrominos T, L und J) mögliche Züge mit je einem zugehörigen simulierten Feld F (Tetrominos siehe Abbildung 3.2). Zu jedem der entstandenen Felder F_i wird mit einer vorher bestimmten Feldbewertungsfunktion eine zugehörige Feldbewertung f_i ermittelt.
Je größer eine Feldbewertung ist, desto besser ist das Feld.
Die Felder werden nach ihrer Feldbewertung absteigend sortiert und neu indiziert zu $\hat{F}_1$ bis $\hat{F}_n$ mit zugehörigen Feldbewertungen $\hat{f}_1$ bis $\hat{f}_n$. Für n gilt je nach Art des Tetrominos $n = 9$, $n = 17$ oder $n = 34$.

5.1.6 Heuristik mit dem Ein-Zug-Lookahead

Um mit dem Ein-Zug-Lookahead zu ermitteln, ob ein Zug für einen Spieler *zu leicht* oder *zu schwer* war, wird zunächst der Ein-Zug-Lookahead mit einer übergebenen Feldbewertungsfunktion durchgeführt. Dann wird eine Grenzfeldbewertung $f_g = \hat{f}_3$ gesetzt.

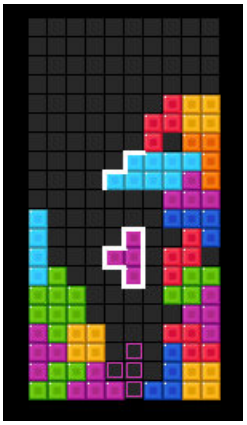


Abbildung 5.2: Vorsprung in Tetris

Hat der Spieler nach seinem Zug ein Feld F_s mit einer Feldbewertung $f_s >= f_g$ erspielt, so liefert die Heuristik den Wert *zu leicht*. Dies geschieht, wenn der Spieler einen Zug macht, der nach der vorgegebenen Feldbewertungsfunktion besser oder ebenso gut wie der drittbeste Zug ist. Ist $f_s < f_g$, liefert die Heuristik *zu schwer*. Dies bedeutet, dass der vom Spieler durchgeführte Zug schlechter als der drittbeste nach der Feldbewertungsfunktion ist.

Der Parameter $g = 3$ zur Grenzfeldbewertung f_g ist frei gewählt. Da es je nach Tetromino verschieden viele Möglichkeiten gibt (9 bis 34), einen Stein zu platzieren, ist es möglich einen prozentuale Herangehensweise zu entwickeln. Aus eigenen Spielerfahrungen mit dem Lookahead hat sich ergeben, dass eine fixe Anzahl bessere Ergebnisse liefert als ein prozentuale Herangehensweise. Außerdem hat sich der Parameter $g = 3$ bewährt. Dies lässt sich damit erklären, dass sich in einem Zeitfenster, welches dem Spieler für einen Zug zur Verfügung steht, der Spieler kaum die Chance hat, mehr als 3 gute Positionen für den aktuellen Tetromino zu finden. Hat ein Spieler einen Stein gesetzt, sodass $f_s < f_3$, so ist davon auszugehen, dass der Zug entweder nicht gut war oder der Spieler bereits mit dem Vorschautetromino (siehe Adaptris Aufbau 3.1) geplant hat. Dies ist zum Beispiel möglich, wenn der Spieler in einem Zug neue Lücken baut, aber mit dem nächsten Tetromino diese Lücken wieder beseitigen kann.

5.1.7 Zwei-Züge-Lookahead

Der Zwei-Züge-Lookahead lässt sich mit jeder Feldbewertung kombinieren und funktioniert wie der Ein-Zug-Lookahead, mit der Änderung, dass der Vorschautetromino mit in Betracht gezogen wird. Das bedeutet, dass zunächst alle möglichen Felder mit dem aktuellen Tetromino simuliert werden. Auf alle simulierten Felder werden weitere Simulationen mit dem Vorschautetromino angewendet. Ausgenommen von der Simulation sind die Züge, bei denen ein Tetromino um einen Vorsprung herum gelenkt wird.
Für den Zwei-Züge-Lookahead wird zunächst der Ein-Zug-Lookahead mit dem aktuellen Tetromino durchgeführt, um alle simulierten Felder F_1 bis F_n zu generieren. Diese werden zunächst nicht geordnet. Die simulierten Felder F_1 bis F_n werden anhand ihrer bestmöglichen Folgefelder verglichen. Für jedes der Felder F_i wird dazu ein weiter Ein-Zug-Lookahead mit dem Vorschautetromino und der Feldbewertungsfunktion durchgeführt. Die Felder, die durch den zweiten Lookahead erstellt werden, sind bereits durch den Ein-Zug-Lookahead nach der übergebenen Feldbewertungsfunktion sortiert. Zur Bewertung der Felder F_1 bis F_n wird die größte Feldbewertung aus den generierten Folgefeldern jedes Feldes F_i übernommen. Jedes Feld F_i hat nun eine Feldbewertung f_i, welche dem Wert des besten Folgefeldes entspricht.

Die Felder werden nach ihrer Feldbewertung absteigend sortiert und neu indiziert zu $\hat{F}_1$ bis $\hat{F}_n$ mit zugehörigen Feldbewertungen $\hat{f}_1$ bis $\hat{f}_n$. Für n gilt je nach Art des Tetrominos $n = 9$, $n = 17$ oder $n = 34$.

5.1.8 Heuristik mit dem Zwei-Züge-Lookahead

Um mit dem Zwei-Züge-Lookahead zu ermitteln, ob ein Zug für einen Spieler *zu leicht* oder *zu schwer* war, wird zunächst der Zwei-Züge-Lookahead mit einer übergebenen Feldbewertungsfunktion durchgeführt. Dann wird eine Grenzfeldbewertung $f_g = \hat{f}_3$ gesetzt.
Hat der Spieler nach seinem Zug ein Feld F_s mit einer Feldbewertung $f_s >= f_g$ erspielt, so liefert die Heuristik den Wert *zu leicht*. Dies ist der Fall, wenn der Spieler einen Zug macht, der unter Berücksichtigung des Vorschautetrominos nach der vorgegebenen Feldbewertungsfunktion besser oder ebenso gut wie der drittbeste Zug ist. Ist $f_s < f_g$, liefert die Heuristik *zu schwer*. Dies bedeutet, dass der vom Spieler durchgeführte Zug schlechter als der drittbeste nach der Feldbewertungsfunktion unter Berücksichtigung des Vorschautetrominos ist.

In eindeutigen Situationen ist der beste Zug nach dem Zwei-Züge-Lookahead nicht anders als bei dem Ein-Zug-Lookahead. Allerdings lässt sich mit dem Zwei-Züge-Lookahead weiter vorausplanen, wodurch der beste Zug des Zwei-Züge-Lookaheads in vielen Fällen besser ist als bei dem Ein-Zug-Lookahead.
Der Parameter $g = 3$ zur Grenzfeldbewertung f_g ist frei gewählt. Da es je nach Tetromino verschieden viele Möglichkeiten gibt (9 bis 34), einen Stein zu platzieren, wäre es möglich einen prozentualen Ansatz zu entwickeln. Dies wird im Abschnitt Ein-Zug-Lookahead (5.1.5) diskutiert.
Im Unterschied zum Ein-Zug-Lookahead kann der Zwei-Züge-Lookahead erkennen, ob ein vermeintlich schlechter Zug gegebenenfalls ein guter Zug war, bei dem der Spieler vorausgeplant hat. Dies ist beispielsweise möglich, wenn der Spieler in einem Zug neue Lücken baut, aber mit dem nächsten Tetromino diese Lücken wieder beseitigen kann.

5.1.9 Heuristik für Feedback realer Spieler

Die in Adaptris verwendete Heuristik zur Bewertung eines Zuges von einem Spieler wird aus bereits vorgestellten Heuristiken zusammengesetzt. Diese Heuristik entscheidet, ob der Spieler mit dem Zug überfordert war oder nicht. Dazu wird wie folgt vorgegangen:

1. Hat der Spieler den Fall mehr als 50% beschleunigt, so wird *zu leicht* zurückgegeben. (siehe Fallbewertung 5.1.1)

2. Mit dem Ein-Zug-Lookahead mit Feldbewertung nach Yiyuan wird überprüft, ob der Spieler einen Zug gemacht hat, der besser oder ebenso gut war, wie der drittbeste mögliche Zug nach dieser Heuristik. Ist dies der Fall so wird *zu leicht* zurückgegeben.

3. Mit dem Zwei-Züge-Lookahead mit Feldbewertung nach Yiyuan wird überprüft, ob der Spieler einen Zug gemacht hat, der besser oder ebenso gut war, wie der drittbeste mögliche Zug nach dieser Heuristik. Ist dies der Fall so wird *zu leicht* zurückgegeben.

4. Treffen die bisherigen Punkte nicht zu, so wird *zu schwer* zurückgegeben.

Diese Heuristik lässt sich wie folgt interpretieren: Wird der Zug beschleunigt, ist es nicht von Belang, wie der Stein gesetzt wurde. Die Geschwindigkeit war nicht herausfordernd.
Hat der Spieler den Tetromino gut platziert, aber nicht mit dem nächsten Tetromino weiter geplant, war er mit der Situation nicht überfordert und hat einen guten Zug gemacht.
Hat der Spieler mit dem Vorschautetromino vorausgeplant und so einen guten Zug gemacht, so hat ihn

diese Situation auch nicht überfordert.
Treffen diese Punkte nicht zu, so hat der Spieler den Stein schlecht platziert. Es ist davon auszugehen, dass ein Spieler mit unbegrenzter Zeit einen Stein optimal, oder zumindest nach der Heuristik gut, platzieren kann. Daher zeugt eine schlechte Platzierung von einer zu knappen Zeit und die Spielsituation war *zu schwer*.

5.2 Optimale KI

Im Umfang der Bachelorarbeit wurde die Feldbewertung nach Yiyuan implementiert und mit dem Zwei-Züge-Lookahead angewendet, was auch Yiyuan in seiner Arbeit macht [11]. Die KI nutzt fast alle möglichen Informationen und Spielmöglichkeiten. Ausgenommen sind Spielzüge, in denen ein Tetromino um einen Vorsprung herum gelenkt wird, und Parameter, die bisher nicht aufgeführt wurden, wie zum Beispiel eine Ermittlung von leeren Nachbarfeldern, wie Leischnig in seiner Arbeit vorschlägt [10], oder die aktuelle maximale Bauhöhe im Spielfeld.
Yiyuan gibt an, weitere Parameter getestet zu haben, die jedoch das Ergebnis der KI nicht verbessert haben. Yiyuan bezeichnet diese KI als (fast) optimal und sie wird in dieser Arbeit als optimal spielend angenommen. Das Tetris, auf welches die KI von Yiyuan ausgelegt ist, ist minimal verschieden zu Adaptris. [11]

5.2.1 Unterschiede im Aufbau

Wie im Spielaufbau angegeben, werden in Adaptris die Tetrominos rein zufällig ausgewählt. In dem Tetris, auf das die KI von Yiyuan optimiert wurde, werden für je sieben Züge alle sieben möglichen Tetrominos in zufälliger Reihenfolge ausgewählt. Durch rein zufällige Züge kann es dazu kommen, dass für eine lange Sequenz nur Tetrominos des Typen S oder Z ausgewählt werden oder ein bestimmter Tetromino lange Zeit nicht kommt. Daher erhöht die rein zufällige Auswahl die Schwierigkeit von Tetris. [11]

5.2.2 Ergebnisse mit der optimalen KI

Wie Yiyuan darlegt, spielt sein Algorithmus so gut, dass selbst nach zwei Wochen und 2183277 vervollständigten Linien das Spiel immer noch lief [11].
In dem Setup von Adaptris wurde der Algorithmus getestet und liefert Spiellängen zwischen 30650 und 58098 vervollständigten Linien. Die Stichprobe umfasst lediglich vier Spiele mit einem Mittelwert von 42871 vervollständigten Linien. Da die Simulation dieser Spiele lange dauert, die Daten jedoch ausreichend sind, um festzustellen, dass ein durchschnittliches Spiel von maximal 300 vervollständigten Linien quasi immer bewältigt werden kann, wurden keine weiteren Spiele mit der optimalen KI simuliert.

6 Vergleich der Aufseher mit Spielersimulation

Um die Aufseher ISA, LR1UV und LR2UV miteinander im Hinblick auf Spielspaß vergleichen zu können, wird zum einen eine Nutzerstudie erhoben (Kapitel 7) und zum anderen werden Spieler simuliert, um eine größere Datenmenge zum Vergleichen zu haben. Dieses Kapitel geht der Frage nach, wie gut sich die Aufseher an unterschiedlich gute (simulierte) Spieler anpassen und wie der ISA-Aufseher im Vergleich zu dem LR2UV-Aufseher konvergiert.
Es wird zunächst erklärt, wie Spieler simuliert werden. Im darauf folgenden Abschnitt wird gezeigt, wie Spiele simuliert werden. Anschließend werden die Ergebnisse der Spiele dargestellt und abschließend werden diese interpretiert.

6.1 Simulierte Spieler

Um einen Computerspieler so real („menschlich“) wie möglich zu simulieren, werden dieselben Annahmen getroffen, wie bei dem ISA- und den LR-Aufsehern: Ein Spieler setzt abhängig von seiner Kompetenz und abhängig von der Schwierigkeit der Situation den Tetromino zu einem gewissen Prozentsatz p gut und zu einem entsprechenden Prozentsatz $\bar{p} = 1 - p$ schlecht.

Da die Schwierigkeit eines Zuges in Tetris von der Geschwindigkeit und der Höhe der Tetrominos im Spielfeld abhängig ist, ergibt sich aus diesen Annahmen eine logistische Funktion

$$p = \frac{1}{1 + e^{-b_0 - b_1 v - b_2 h}} \tag{6.1}$$

mit p = Wahrscheinlichkeit, einen *schlechten Zug* zu machen. b_0, b_1 und b_2 sind interne Parameter, welche die Kompetenz des Spielers im Allgemeinen (b_0), relativ zur Geschwindigkeit v (b_1) und relativ zur Höhe der Tetrominos im Spielfeld h (b_2) ausdrücken.

Bei den simulierten Spielern wird diese Wahrscheinlichkeit nur in Abhängigkeit der Fallgeschwindigkeit und der Höhe der Tetrominos ausgedrückt. Sie hängt auch von anderen Parametern ab, zum Beispiel davon, wie gut der aktuelle Tetromino zu gebrauchen ist. Dies ist jedoch schwer messbar und wird daher in der Simulation nicht berücksichtigt.

In einem Zug sind die Parameter Fallgeschwindigkeit und Höhe der Tetrominos gegeben. Ein realer Spieler, der mit der Situation überfordert ist, macht wahrscheinlich einen *schlechten Zug*, wohingegen ein unterforderter Spieler wahrscheinlich einen *guten Zug* macht. Auszurechnen ist der Prozentsatz p für den *schlechten Zug* des simulierten Spielers, indem die Parameter v und h in die Funktion p einsetzt werden. Die Wahrscheinlichkeit für einen *guten Zug* ist dementsprechend $\bar{p} = 1 - p$.
Dies wird in der Abbildung 6.1 dargestellt. Die Werte in der Abbildung, $v = 17,1$ und $h = 12$, sind gegeben und der Prozentsatz $p = 0.39$ lässt sich ablesen beziehungsweise ausrechnen.

Da getestet werden soll, wie gut die Aufseher sich an die Kompetenz verschiedener Spieler anpassen können, wurden drei verschiedene Spieler simuliert.

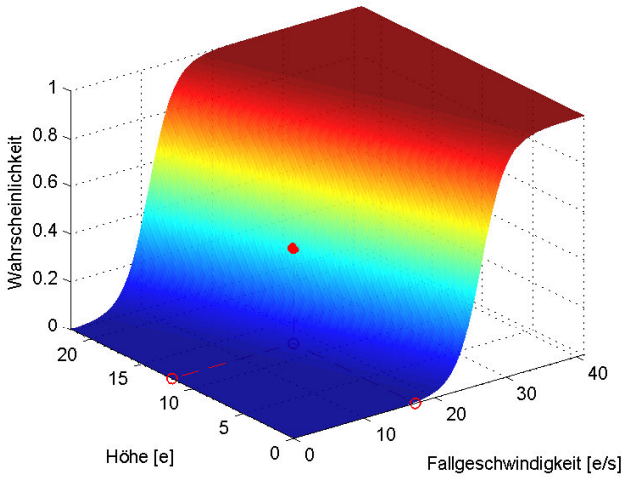


Abbildung 6.1: Ablesen der Wahrscheinlichkeit aus dem Profil eines simulierten Spielers

Ein Anfänger (siehe Abbildung 6.2a) wurde simuliert mit der Funktion

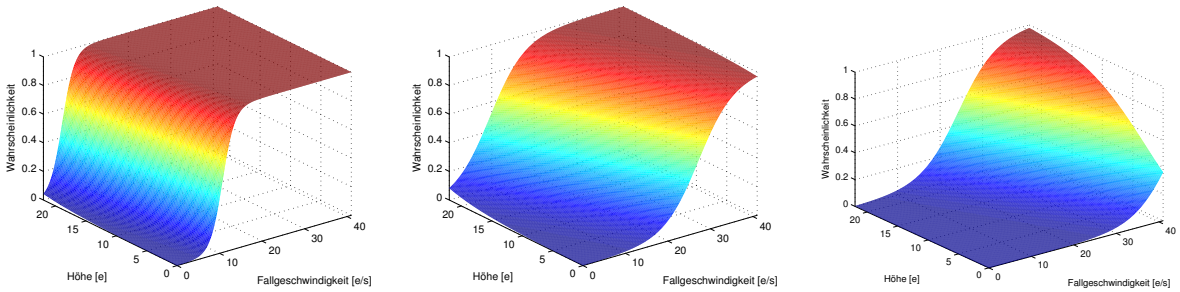
$$p_a = \frac{1}{1 + e^{7.5 - 0.7v - 0.2h}}, \tag{6.2}$$

ein Fortgeschrittener (siehe Abbildung 6.2b) mit der Funktion

$$p_f = \frac{1}{1 + e^{6.8 - 0.25v - 0.2h}} \tag{6.3}$$

und eine Experte (siehe Abbildung 6.2c) mit der Funktion

$$p_e = \frac{1}{1 + e^{10 - 0.23v - 0.2h}}. \tag{6.4}$$



(a) Anfänger **(b)** Fortgeschrittener **(c)** Experte

Abbildung 6.2: Simulierte Spieler

Wie zu erkennen, ist die Schwelle für den Anfänger, einen *schlechten Zug* zu einem Prozentsatz p zu setzen, bereits bei geringerer Geschwindigkeit und Höhe der Tetrominos höher als bei dem Fortgeschrittenen. Gleiches gilt für den Fortgeschrittenen im Vergleich zum Experten.

6.2 Simulierte Spiele

In einem simulierten Spiel steuert der simulierte Spieler den Tetromino. Dazu wird zunächst durch Einsetzen der aktuellen Spielparameter die Wahrscheinlichkeit p ermittelt, wie im vorigen Abschnitt beschrieben. Nun wird eine Zufallszahl $z \in [0,1]$ erstellt. Ist $z >= p$ so wird der optimale Zug mit der optimalen KI durchgeführt. Ist $z < p$, so wird ein schlechter Zug durchgeführt.
Ein zufälliger Zug kann auch ein *guter Zug* sein. Aus diesem Grund werden für einen simulierten *schlechten Zug* alle möglichen Züge ermittelt und einer der 70% schlechtesten Züge zufällig ausgewählt. Dies geschieht mit der optimalen Heuristik (Zwei-Züge-Lookahead mit Feldbewertung nach Yiyuan, vergleiche Abschnitt 5.1.4).

Ein Aufseher (zum Beispiel ISA, LR1UV oder LR2UV) registriert die Züge und ermittelt mit seiner eigenen Heuristik, der Heuristik für Feedback reale Spieler (siehe Abschnitt 5.1.9), ob der Zug *gut* oder *schlecht* war.
Bei der Simulation folgt aus einem schlechten Zug immer das Feedback *zu schwer* und aus einem guten Zug immer das Feedback *zu leicht*.
Dieses Feedback wird in die Datenmenge des Aufsehers aufgenommen und die Geschwindigkeit für den nächsten Zug eingestellt. Die Wahrscheinlichkeit für die neue Geschwindigkeit und die neue Höhe wird wieder in die Funktion des simulierten Spielers eingesetzt und es geht weiter mit dem nächsten Zug. Das Spiel wird so lange gespielt, bis der simulierte Spieler das Spiel verliert.

Um möglichst viele Daten in geringer Zeit zu bekommen, werden die Steine in der Simulation direkt in ihre Zielposition eingefügt. Die aufgenommenen Spielstatistiken bestehen aus der Spiellänge und dem Prozentsatz an schlechten Zügen.

6.3 Ergebnisse aus der Spielersimulation

Zu vier ausgewählten Aufsehern wurden je 200 Spiele mit dem simulierten Anfänger, dem Fortgeschrittenen und dem Experten gespielt. Die vier Aufseher sind der ISA-, der LR1UV-, LR2UV-, und der LR2UV-50 Aufseher. Die ersten drei wurden bereits beschrieben. Der LR2UV-50-Aufseher funktioniert genauso, wie der LR2UV-Aufseher, mit dem Unterschied, dass er einen Prozentsatz schlechter Züge von 50% annähert anstatt von 5,39%. Der LR2UV-50-Aufseher wird in die Tests mit einbezogen, um eine gute Vergleichbarkeit zu dem ISA-Aufseher herzustellen, da der ISA einen Prozentsatz schlechter Züge von 50% annähert. Dieser Wert ist beim ISA nicht variabel.

Die Ergebnisse der Spiellängen und der Prozentsätze schlechter Züge werden für jeden der Aufseher veranschaulicht und interpretiert. Die Auswertung basiert auf den Daten der Diagramme aus den Abbildungen 6.3a und 6.3b. Im Diagramm aus Abbildung 6.3a ist die durchschnittliche Anzahl der vervollständigten Linien in den simulierten Spielen zu den drei Aufsehern mit jedem der simulierten Spieler (Anfänger, Fortgeschrittener, Experte) abgebildet. Das Diagramm aus Abbildung 6.3b zeigt den durchschnittlichen Prozentsatz schlechter Züge.

Zunächst wird erläutert, wie Spiele mit den Spielern verschiedener Kompetenz ablaufen, wenn die Schwierigkeit nicht angepasst wird. Danach wird die Auswertung der simulierten Spiele aufgeführt. Diese wird aufgegliedert in einen Vergleich zwischen dem LR1UV-, dem LR2UV- und dem ISA-Aufseher und einen Vergleich zwischen dem ISA-Aufseher und dem LR2UV-50-Aufseher.

6.3.1 Vergleichswerte Spiele ohne Anpassung und optimale Ergebniswerte

Spiele ohne Anpassung sind Spiele, bei denen sich die Fallgeschwindigkeit nicht ändert.
Eine konstante niedrige Fallgeschwindigkeit ist für Anfänger sehr gut. Für Fortgeschrittene und Experten ist diese Einstellung jedoch langweilig. Wird eine konstante hohe Geschwindigkeit gewählt, sind Anfänger stark überfordert. Eine solche Geschwindigkeit ist jedoch für Experten sehr gut. Bei Spielen mit mittlerer Fallgeschwindigkeit vervollständigt ein Anfänger keine bis sehr wenige Linien, bevor er das Spiel verliert. Ein Fortgeschrittener kann bei dieser Einstellung recht viele Linien vervollständigen. Eine mittlere Fallgeschwindigkeit kann für ihn genau richtig sein. Experten sind mit der Einstellung eher unterfordert und können daher endlos lange spielen oder sehr viele Linien vervollständigen, bevor sie verlieren.

Der angenommene optimal Wert für die Spiellänge simulierter Spiele liegt zwischen 150 und 199 vervollständigen Linien (vergleiche Abschnitt 4.4), aber auch bei Spielen ab circa 50 bis 149 vervollständigten Linien ist davon auszugehen, dass ein vernünftiges Spiel zustande gekommen ist. Bei Spielen mit mehr als 200 vervollständigten Linien hingegen ist davon auszugehen, dass das Spiel für einen Spieler eher wenig herausfordernd und langweilig war.
Eine gute Anpassung der Spielschwierigkeit zeichnet sich zusätzlich dadurch aus, dass die Werte der Spiellängen wenig Abweichungen zwischen dem simulierten Anfänger, dem Fortgeschrittenen und dem Experten aufweisen.

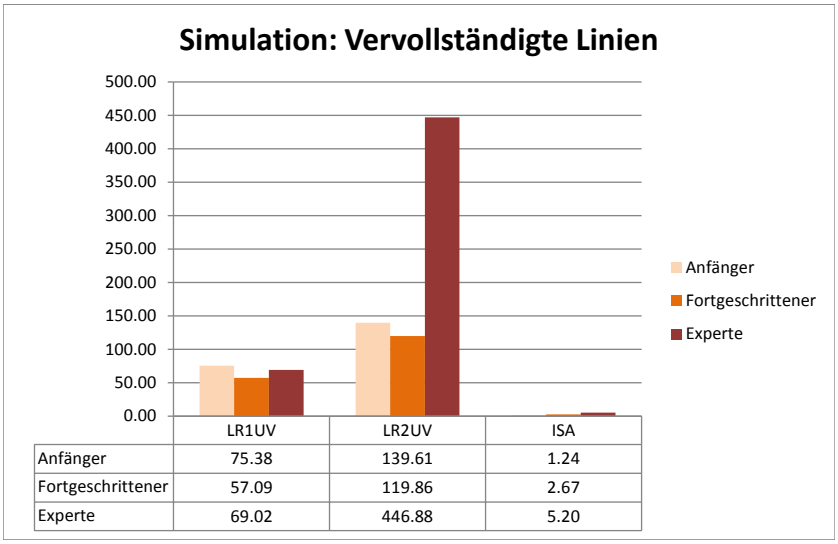
Der optimale Prozentsatz schlechter Züge liegt bei 5,39% (vergleiche Abschnitt 4.4). Bei normalen Spielen ohne Anpassung würde dieser Prozentsatz sehr hoch sein, bei Fortgeschrittenen mittelhoch und bei Experten gegen 0 gehen, da die Experten erst bei hohen Fallgeschwindigkeiten gefordert sind.
Eine gute Anpassung zeichnet sich dadurch aus, dass der vorgegebene Prozentsatz gut angenähert wird und dass die Spanne zwischen den Prozentsätzen des Anfängers, des Fortgeschrittenen und des Experten klein ist.

6.3.2 Simulationsauswertung LR1UV-Aufseher

Mit dem LR1UV-Aufseher wurden im Schnitt zwischen 50 und 80 Linien vervollständigt (siehe Abbildung 6.3a), die Werte zu den verschiedenen Spielern sind ähnlich. Diese Spiele haben alle eine passende Länge (über 50), nähern sich dem angepeilten Wert von circa 150 Linien jedoch nicht gut an. Die Werte liegen sehr nah zusammen. Das lässt auf eine gute Anpassung schließen.

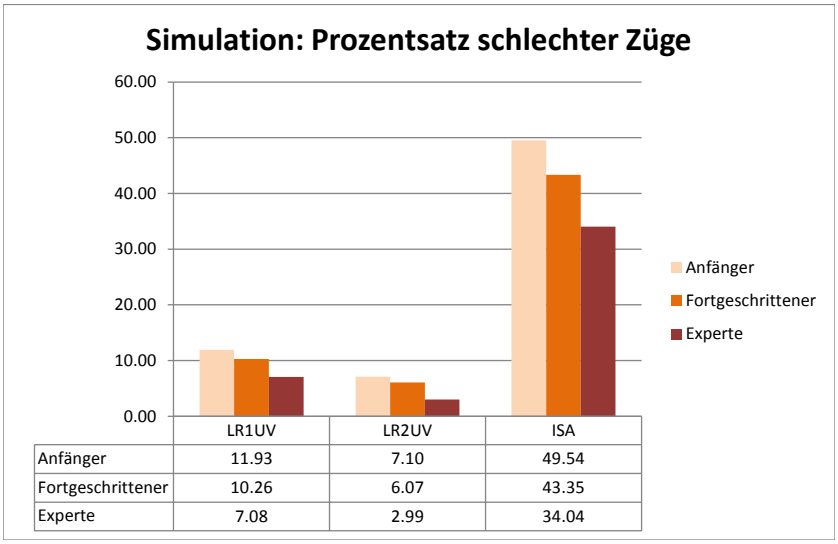
Der durchschnittliche Prozentsatz schlechter Züge liegt zwischen 7% (Experte) und 12% (Anfänger), der optimale und angepeilte Wert von 5.39% wurde vor allem bei den Spielen des Anfängers (11.93%) und des Fortgeschrittenen (10,36%) stark überschritten, ist aber ziemlich gut für Spiele des Experten (7.08%) (siehe Abbildung 6.3b).
Der höhere Prozentsatz im Vergleich zum LR2UV-Aufseher (Abbildung 6.3b) kommt dadurch zustande, dass der LR1UV-Aufseher nicht auf die aktuelle Höhe der Tetrominos eingeht. Daher ist die Chance des simulierten Spielers für einen *schlechten Zug* bei einem höher gebauten Feld größer als vom LR1UV-Aufseher angenommen.
Der durchschnittliche Prozentsatz schlechter Züge der drei Spieler liegt relativ dicht zusammen. Der Unterschied beträgt weniger als 5%.

Eine Anpassung an alle Kompetenzen verschiedener Spieler ist gegeben. Es wird jedoch ein deutlich zu hoher Prozentsatz schlechter Züge angenähert.



	LR1UV	LR2UV	ISA
Anfänger	75.38	139.61	1.24
Fortgeschrittener	57.09	119.86	2.67
Experte	69.02	446.88	5.20

(a) Simulation: Vervollständigte Linien LR1UV, LR2UV, ISA



	LR1UV	LR2UV	ISA
Anfänger	11.93	7.10	49.54
Fortgeschrittener	10.26	6.07	43.35
Experte	7.08	2.99	34.04

(b) Simulation: Prozentsatz schlechter Züge LR1UV, LR2UV, ISA

Abbildung 6.3: Simulationsdaten zu LR1UV, LR2UV und ISA

6.3.3 Simulationsauswertung LR2UV-Aufseher

Bei Spielen mit dem LR2UV-Aufseher wurden mit dem Anfänger und dem Fortgeschrittenen im Schnitt zischen 100 und 150 Linien vervollständigt (siehe Abbildung 6.3a), was eine passable Annäherung an den gewünschten Wert ist. Der Experte spielte im Schnitt bis etwa 450 Linien vervollständigt wurden. Dies lässt darauf schließen, dass die Schwierigkeit für den Experten nicht so gut angepasst wurde, wie bei den weniger guten Spielern.

Die Spiele mit dem LR2UV-Aufseher wurden von dem Anfänger im Schnitt mit 7,1% schlechter Züge, von dem Fortgeschrittenen mit 6,07% und von dem Experten mit 2.99% schlechter Züge abgeschlossen. Die Werte für den Anfänger und den Fortgeschrittenen liegen sehr nah beieinander und sind zudem recht nah am angepeilten Wert von 5.39%. Hier hat die Anpassung sehr gut funktioniert. Bei dem Experten hingegen wurden im Schnitt nur 2.99% schlechte Züge gemacht. Der geringe Prozentsatz führt zu Spie-

len von circa 450 Linien (siehe Abbildung 6.3a).

Die Initialisierungsfunktion des LR2UV-Aufsehers schätzt für den Experten zunächst zu geringe Schwierigkeiten. Der Aufseher konvergiert deutlich langsamer gegen die Vorgabe von 5%, wenn er die passende Schwierigkeit zunächst zu gering schätzt als wenn er sie zunächst zu hoch schätzt. Letzteres ist beim Anfänger und beim Fortgeschrittenen der Fall. Selbst nach etwa 450 Zügen ist bei Spielen mit dem Experten nur ein durchschnittlicher Prozentsatz von 2,99% erreicht. Die Abweichung zum angepeilten Wert 5,39% ist größer als 2%. Die Spiele der Anfänger der Fortgeschrittenen erreichen durchschnittlich eine Abweichung von weniger als 2% bereits nach etwa 140 beziehungsweise 120 vervollständigten Linien.

Die Anpassung an einen Spieler, der besser ist als die initiale Schätzung, dauert also länger. Der Grund hierfür wurde in Abschnitt 4.5.1 anhand von Grafiken dargestellt und erläutert. Das Verhalten des LR2UV gleicht dem des LR1UV in dieser Hinsicht und wird mit den beschrieben Ergebnissen bestätigt.

Abschließend lässt sich für den LR2UV-Aufseher mit simulierten Spielern sagen, dass die Anpassung gut funktioniert. Diese Aussage basiert darauf, dass die Spanne zwischen den Prozentsätzen schlechter Züge mit weniger als 4,2% gering ist und der optimale Wert von 5,39% zwischen den Werten des Aufsehers liegt.

6.3.4 Simulationsauswertung ISA-Aufseher

Spiele mit dem ISA-Aufseher sind schnell verloren worden. In den Spielen wurden selbst mit dem Experten im Schnitt weniger als 6 Linien vervollständigt (siehe Abbildung 6.3a). Die Schwierigkeit ist insgesamt zu hoch. Das lässt auf wenig Anpassung schließen.
Der Prozentsatz schlechter Züge liegt bei dem ISA-Aufseher durchschnittlich zwischen 49,54% bei Spielen des Anfängers und 34,04% bei Spielen des Experten (siehe Abbildung 6.3b). Diese Werte liegen weit über dem optimalen Wert. Daher sind auch die Spiele sehr kurz.
Der ISA ist jedoch mit einer Heuristik, die nur die Werte *zu leicht* und *zu schwer* zurückgibt, darauf ausgelegt, gegen den Wert von 50% zu konvergieren.

Für den ISA-Aufseher mit der Heuristik, die in dieser Arbeit verwendet wurde, lässt sich daher feststellen, dass die Anpassung an die Spieler nicht funktioniert. Um einen weiteren Vergleich anführen zu können, wird ein direkterer Vergleich zwischen dem ISA- und dem LR2UV-50-Aufseher gemacht (Abschnitt 6.3.6).

6.3.5 Vergleich zwischen LR1UV-, LR2UV- und ISA-Aufseher

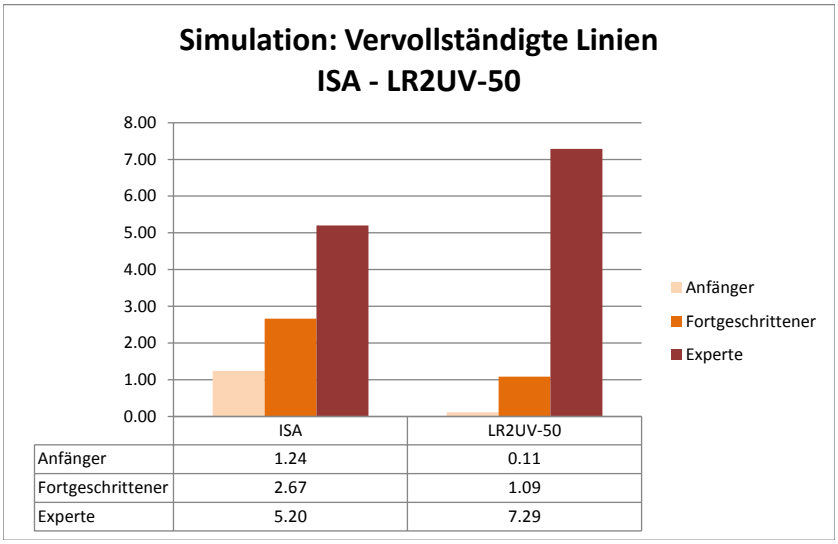
Von den drei vorgestellten Aufsehern schneidet der LR2UV-Aufseher in der Simulation am besten ab. Die vervollständigten Linien liegen für den Anfänger und den Fortgeschrittenen in einem sehr guten Bereich. Die Werte zum Prozentsatz schlechter Züge liegen am nächsten am angepeilten Wert, lediglich die langsame Konvergenz des Experten ist nicht optimal.

Der LR1UV-Aufseher ist der nächstbeste im Vergleich. Die Spiele mit dem Aufseher haben eine passable Länge und die Spanne der Prozentsätze schlechter Züge liegt nah beisammen. Allerdings liegen diese Prozentsätze alle zu hoch.

Der ISA-Aufseher nähert einen für Tetris unpassenden Prozentsatz schlechter Züge an und schneidet deshalb am schlechtesten ab. Es werden viel zu wenige Linien vervollständigt, bevor das Spiel zu Ende ist, sodass kein richtiges Spiel entsteht. Die Prozentsätze schlechter Züge liegen sehr hoch und variieren stark.

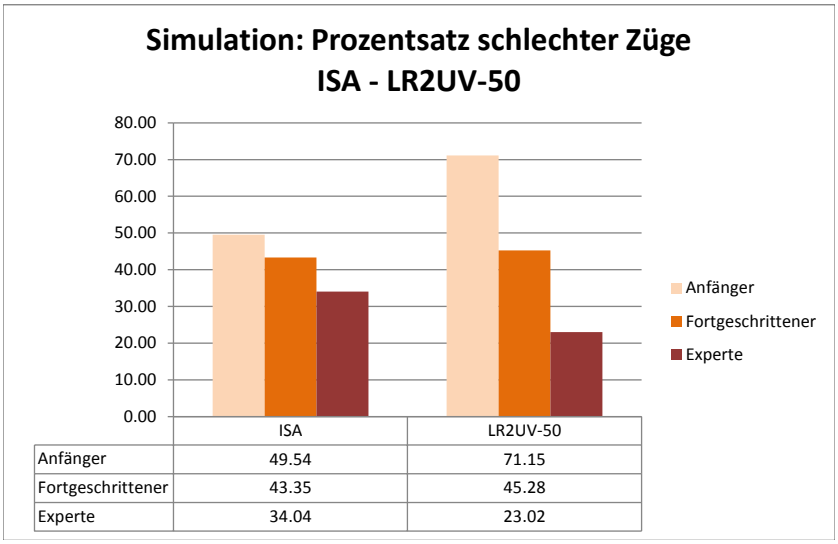
6.3.6 Vergleich zwischen ISA- und LR2UV-50-Aufseher

Im letzten Abschnitt wurde gezeigt, dass der ISA-Algorithmus mit der gegebenen Heuristik die Herausforderung auf die verschiedenen simulierten Spieler nicht gut anpasst. In diesem Abschnitt wird der ISA- mit dem LR2UV-50-Aufseher verglichen. Die Aufseher sind sehr ähnlich, da sie nicht nur die selbe Heuristik nutzen, sondern zusätzlich beide den Wert von 50% schlechten Zügen annähern.
Die Auswertung bezieht sich auf die Diagramme der Abbildungen 6.4a und 6.4b. Die Daten für den ISA-Aufseher sind dieselben wie in den Abbildungen 6.3a und 6.3b. Lediglich die Achsen sind anders skaliert.



	ISA	LR2UV-50
Anfänger	1.24	0.11
Fortgeschrittener	2.67	1.09
Experte	5.20	7.29

(a) Simulation: Vervollständigte Linien ISA, LR2UV-50



	ISA	LR2UV-50
Anfänger	49.54	71.15
Fortgeschrittener	43.35	45.28
Experte	34.04	23.02

(b) Simulation: Prozentsatz schlechter Züge ISA, LR2UV-50

Abbildung 6.4: Simulationsdaten zu ISA und LR2UV-50

An den vervollständigten Linien lässt sich erkennen, dass im Durchschnitt jeder der simulierten Spieler weniger als 10 Linien vervollständigt hat. Das zeigt zum einen, dass eine angepeilte Vorgabe von etwa 50% schlechten Zügen nicht zu einem Spiel in der Flow-Zone führt. Zum anderen lässt sich aus den Daten ableiten, dass es nicht am Algorithmus liegt, dass das Spiel nicht richtig angepasst wird, sondern an

der beschriebenen Vorgabe. Weitere Aussagen lassen sich besser am Prozentsatz schlechter Züge ableiten.

Die Spiele mit dem ISA-Aufseher wurden mit einem durchschnittlichen Prozentsatz schlechter Züge von 49,54% (Anfänger) bis 34,04% (Experte) abgeschlossen. Die Werte sinken mit steigender Kompetenz. Das liegt daran, dass der Experte besser anfängt und der Algorithmus seine initiale Schätzung stärker korrigieren muss als bei dem Anfänger. Der Experte setzt daher mehr Tetrominos gut bis der Aufseher sich an die Kompetenz angepasst hat. Jedoch ist auch bei einem Prozentsatz von 34% das Spiel schnell beendet, weshalb der Aufseher nur wenige Feedbacks bekommt, um sich dem Spieler anzupassen.
Selbiges gilt für den LR2UV-50-Aufseher, mit dem Unterschied, dass dieser auch für den Anfänger zu wenig Feedbacks bekommt, um sich der Kompetenz in der kurzen Zeit besser anzupassen. Die Werte liegen bei dem Aufseher zwischen 71,15% (Anfänger) und 23.02% (Experte).

Alle Spiele der verschiedenen Spieler mit den beiden Aufsehern haben etwa gleich viele Züge gedauert. Das Spiel ist beendet, sobald das Spielfeld bis oben voll gebaut ist. Ein bis zwei vervollständigte Linien fallen daher nicht ins Gewicht.

Aus den gewonnenen Daten lässt sich schließen, dass eine gute Anpassung mit keinem der beiden Aufseher möglich ist.
Die Spanne der Prozentsätze schlechter Züge ist bei dem ISA-Aufseher mit weniger als 16% wesentlich geringer als beim LR2UV-50-Aufseher mit mehr als 48%. Da die Aufseher etwa gleich viele Feedbacks bekommen haben, lässt sich daraus schließen, dass der ISA deutlich schneller gegen den angepeilten Wert konvergiert als der Lernalgorithmus basierend auf LR2UV.

7 Nutzerstudie

Um festzustellen, wie reale Spieler Adaptris mit den verschiedenen Aufsehern empfinden, wird eine Nutzerstudie durchgeführt. Diese geht der Fragestellung nach, ob und wie gut jeder der Aufseher die Schwierigkeit an die realen Spieler adaptiert. Des Weiteren soll festgestellt werden, wie die Spieler das Spielerlebnis empfinden und wie herausfordernd die Spiele für sie sind.
Die Umgebung für die Studie wurde von Leischnig entwickelt und implementiert [10], die Studie selber wurde zum Teil verändert.

Zunächst wird der Aufbau der Nutzerstudie erklärt, danach werden die Ergebnisse präsentiert und im Anschluss werden diese interpretiert.

7.1 Aufbau

Die Nutzerstudie fängt mit einer Willkommensseite an, in der den Teilnehmern der Aufbau der Studie erläutert wird. Durch Drücken von Buttons gelangen die Spieler von einer Seite zur nächsten. Die zweite Seite beinhaltet einen Fragebogen. Die Spieler werden aufgefordert, ihr Alter in eine der folgenden Kategorien einzuordnen

- <18
- 18-29
- 30-49
- >50

und eine Einschätzung über ihre Spielstärke abzugeben:

- 0: Nie zuvor gespielt (Neuling)
- 1: Ich weiß, wie das Spiel funktioniert (Anfänger)
- 2: Ich habe ab und zu mal gespielt (Fortgeschrittener Anfänger)
- 3: Ich bin ganz gut (Fortgeschrittener)
- 4: Ich habe viel gespielt oder spiele immer noch viel (Experte).

Den Spielern wird zugesichert, dass sich ihre Angaben nicht auf das Spiel auswirken. Dadurch wird sichergestellt, dass sich die Spieler nicht absichtlich schlechter einschätzen, da sie gegebenenfalls denken, sie könnten die Spielschwierigkeit auf diese Weise beeinflussen.

Auf der nächsten Seite ist das Spielfeld zu sehen. Den Spielern wird die Steuerung des Spiels erklärt: Das Spiel wird mit den Pfeiltasten gesteuert. Pfeiltaste links und Pfeiltaste rechts bewegen den aktiven Tetromino nach links beziehungsweise rechts, mit der Pfeiltaste nach oben wird der Tetromino im Uhrzeigersinn gedreht und mit der Pfeiltaste nach unten wird der Fall beschleunigt.
Die Teilnehmer haben außerdem die Möglichkeit zu lesen, wie Tetris allgemein funktioniert.

Um sich mit der Steuerung vertraut zu machen, soll jeder Spieler zunächst ein Tetrisspiel spielen, bei dem die Geschwindigkeit durchgehend langsam ist. Das Spiel endet nach einer Minute oder sobald das Spiel verloren ist.
Nachdem das Spiel beendet ist und sobald der Spieler bestätigt, dass er fertig ist, geht es mit dem Hauptteil der Studie los. In diesem Teil soll der Spieler drei Tetrisspiele absolvieren, die je von verschiedenen Aufsehern beobachtet und gesteuert werden. Die Spiele dauern drei Minuten oder enden, sobald das Spiel verloren wurde. Nach jedem der drei Spiele bekommt der Spieler einen Fragebogen mit den folgenden Fragen und Antwortmöglichkeiten:

- Wie viel Spaß hatten Sie im Spiel insgesamt?
  Fünf Antwortmöglichkeiten: 1 (kein Spaß) bis 5 (sehr viel Spaß)
- War das Spiel für Sie zu herausfordernd oder zu wenig herausfordernd?
  Fünf Antwortmöglichkeiten: -2 (viel zu wenig herausfordernd) bis 2 (viel zu herausfordernd)
- Wie fair war das Spiel am Ende?
  Fünf Antwortmöglichkeiten: 1 (unfair) bis 5 (sehr fair).

Der erste Aufseher ist der reaktionsbasierte Aufseher. Die gewonnenen Daten dienen bei der Auswertung dazu, den Spielern eine grobe Spielstärke zuweisen zu können. Die nächsten beiden Aufseher sind zwei verschiedene zufällige aus dem ISA-Aufseher, dem LR1UV-Aufseher und dem LR2UV-Aufseher.
Anders als bei Leischnig werden bei diesen drei Spielen, zusätzlich zu den Informationen der Fragebögen, folgende Werte ermittelt und in der Datenbank gespeichert:

- Die Anzahl der vervollständigten Linien im Spiel und
- der Prozentsatz an schlechten Zügen, die der Spieler im Spiel gemacht hat.

Diese Werte ermöglichen zusätzlich zu den Fragebögen objektive Daten zur Auswertung. Nach den drei Spielen ist die Studie beendet.

7.2 Teilnehmerverteilung

Insgesamt haben 62 Teilnehmer an der Studie teilgenommen. Da diese Stichprobe nicht sehr groß ist, sind die Ergebnisse nur eingeschränkt generalisierbar.

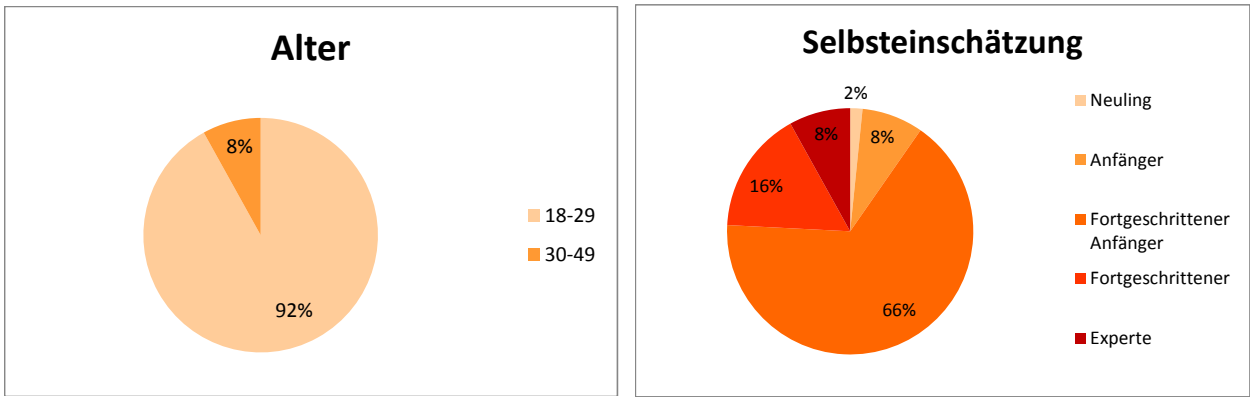
Im Folgenden wird zunächst die Teilnehmermenge beschrieben. Danach werden die Ergebnisse der Fragebögen zu dem ISA-, dem LR1UV- und dem LR2UV-Aufseher wiedergegeben und visualisiert. Anschließend werden die Ergebnisse der Hintergrunddaten dargestellt und erläutert.

Der erste Fragebogen enthält das angegebene Alter und die Selbsteinschätzung der Spieler. Die Verteilung der Antworten sind in Abbildung 7.1 dargestellt. Die meisten Teilnehmer waren zwischen 19 und 29 Jahren alt (92%), alle anderen zwischen 30 und 49 (8%).
Die Altersverteilung der Studie ist für Adaptris nicht repräsentativ, da niemand aus den Altersgruppen unter 19 oder über 50 teilgenommen hat. In dieser Studie geht es jedoch hauptsächlich darum, eine Variation von verschieden guten Spielern zu bekommen, da die Spielstärke von den Aufsehern analysiert wird. Daher hat die Spielstärke für die Repräsentativität der Studie eine größere Bedeutung als das Alter.

Die Teilnehmer haben sich zu 66% durchschnittlich gut eingeschätzt, zu 10% schlechter und zu 24% besser. Eine genaue Verteilung ist dem Diagramm in Abbildung 7.1b zu entnehmen. Jede der Gruppen ist vertreten. Die Selbsteinschätzung der Kompetenz ist jedoch sehr ungleich über die Gesamtheit verteilt, daher werden die Teilnehmer im Laufe der Auswertung anhand einer anderen Kompetenzeinschätzung

eingeteilt. Diese wird erst mit Hilfe der gewonnen Hintergrunddaten ermittelt.
Insgesamt ist die Studie nur bedingt repräsentativ.

Die Daten für die einzelnen Spiele mit den verschiedenen Aufsehern werden in Abbildungen 7.2 und 7.3 abgebildet. Die Daten sind prozentual in Säulendiagrammen dargestellt und die prozentualen Verteilungen sind in den Datentabellen mit abgebildet.



(a) Altersangabe der Teilnehmer **(b)** Selbsteinschätzung der Teilnehmer

Abbildung 7.1: Nutzerstudie: Ergebnisse des ersten Fragebogens

7.3 Teilnehmer aufgeteilt nach Kompetenz

Um festzustellen, ob die Aufseher sich auf die Spielstärke der Spieler einstellen, werden die Spieler in zwei Kategorien bezüglich ihrer Kompetenz aufgeteilt. Zur Einteilung wird die Selbsteinschätzung nicht betrachtet, um zum einen etwa gleich große Gruppen zu erhalten und zum anderen weil die Selbsteinschätzung subjektiv ist.

Die Kompetenz wird gemessen an der Anzahl vervollständigter Linien im Spiel mit dem reaktionsbasierten Aufseher. Diese Annahme beruht auf dem Gedanken, dass gute Spieler zum einen länger spielen (bis zu drei Minuten), da sie das Spiel nicht so schnell verlieren. In mehr Zeit ist es möglich mehr Linien zu vervollständigen. Zum anderen wird die Geschwindigkeit bei guten Spielern höher eingestellt. Daher fallen in der Spielzeit mehr Tetrominos, wodurch mehr Linien vervollständigt werden können.

Die Teilnehmer werden in zwei etwa gleich große Kategorien aufgeteilt: Kategorie 1 mit 33 Teilnehmern, die weniger oder genau 15 Linien vervollständigt haben (dies ist der Median) und Kategorie 2 mit 29 Teilnehmern, die mehr als 15 Linien vervollständigt haben. Die Gruppe der Spieler aus Kategorie 1 wird als Anfänger bezeichnet, die der Kategorie 2 als Fortgeschrittene. Dies trifft nicht bei jedem der Spieler zu, erleichtert aber das Verständnis der Arbeit.
Die Größe der Stichprobe lässt keine Einteilung in kleinere Gruppen zu.

7.4 Ergebnisse und Interpretation

In diesem Abschnitt werden die drei Aufseher zunächst allgemein anhand der gesammelten Hintergrunddaten verglichen. Es folgt ein Vergleich basierend auf der Auswertung der Fragebögen. Zuletzt werden die Aufseher einzeln betrachtet, wie gut sie sich an die Spielstärke der Spieler anpassen.

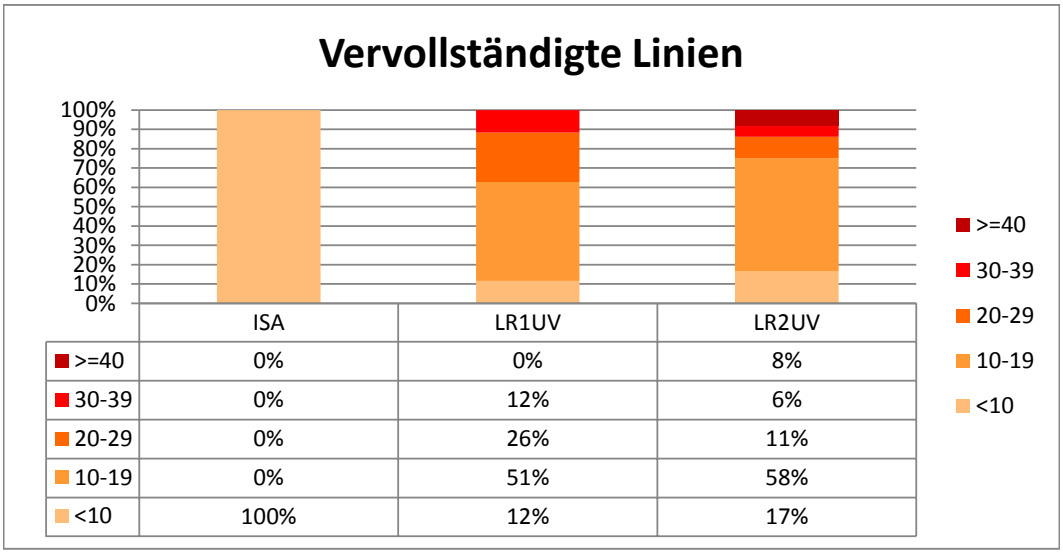
7.4.1 Auswertung der Hintergrunddaten

In Abbildung 7.2 werden die gesammelten Hintergrunddaten visualisiert. Die Verteilung der vervollständigten Linien wird in Abbildung 7.2a dargestellt. Bei Spielen mit dem ISA-Aufseher wurden in allen Spielen weniger als 10 Linien vervollständigt bevor das Spiel verloren wurde oder bevor die Zeit abgelaufen ist. Der Durchschnitt liegt bei 1,56 vervollständigten Linien.
Bei Spielen mit dem LR1UV-Aufseher wurden im Durchschnitt 17.91 Linien vervollständigt. Jeweils 12% der Teilnehmer haben weniger als 10 oder mehr als 30 Linien vervollständigt. Alle anderen Teilnehmer haben zwischen 10 und 30 Linien vervollständigt.
Spiele mit dem LR2UV-Aufseher wurden mit durchschnittlich 19,31 vervollständigten Linien abgeschlossen. 17% der Teilnehmer haben die Spiele mit weniger als 10 Linien beendet, 14% mit mehr als 30 Linien und 69% mit 10 bis 29 Linien.
Obwohl bei der Ermittlung des vorgegebenen Prozentsatzes schlechter Züge (Abschnitt 4.4) eine Vorgabe von 150 bis 199 vervollständigter Linien gesetzt wurde, sind die Werte der Teilnehmer deutlich geringer. Dies liegt zum einen daran, dass die Spiele in der Nutzerstudie nach drei Minuten beendet wurden. Zum anderen wurde bei der Vorgabe jeder gute Zug optimal gespielt. Die Spieler können aber auch andere Züge machen, die als *gute Züge* gewertet werden. Entsprechend nimmt die allgemeine Zugqualität ab, wodurch das Spiel im Allgemeinen schneller verloren wird.

Die Verteilung der Prozentsätze schlechter Züge ist in Abbildung 7.2b dargestellt. Bei Spielen mit dem ISA-Aufseher liegt der durchschnittliche Prozentsatz bei 47,23%. Nur ein Teilnehmer hat einen Prozentsatz von weniger als 15% erreicht.
Spiele mit dem LR1UV-Aufseher wurden mit einem durchschnittlichen Prozentsatz von 9,66% schlechter Züge beendet. Keiner hatte weniger als 5% oder mehr als 15% schlechte Züge. Die meisten Teilnehmer (77%) haben am Ende des Spiels zwischen 7,5% und 12,5% schlechte Züge gespielt.
Bei Spielen mit dem LR2UV-Aufseher wurden im Durchschnitt 7,56% schlechte Züge gemacht. Je 6% der Teilnehmer haben einen Prozentsatz von weniger als 5% schlechter Züge oder von mehr als 12,5% schlechter Züge. Die meisten Teilnehmer (88%) haben zwischen 5% und 10% schlechte Züge gemacht.

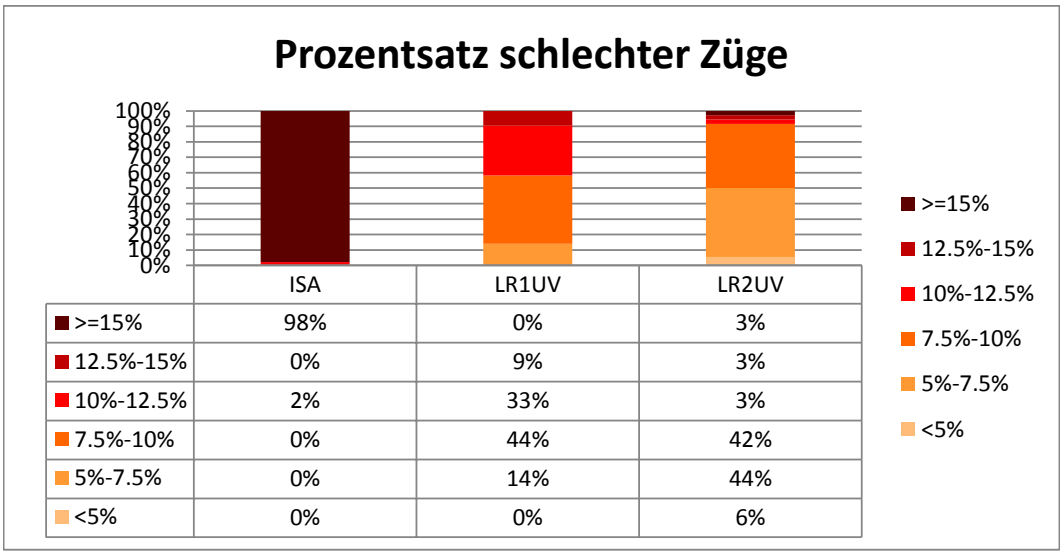
Es wird nun ermittelt, wie gut die Aufseher sich an vorgegebenen Werte angepasst haben.
Auffällig ist bei der Verteilung der vervollständigten Linien, dass keiner der Teilnehmer in den Spielen mit dem ISA-Aufseher mehr als 9 Linien vervollständigt hat. Die wenigen vervollständigten Linien bei den Spielen mit den ISA-Aufseher lassen sich mit dem Prozentsatz schlechter Züge erklären. Hier wurden in fast allen Fällen mehr als 15% schlechter Züge erkannt. Bei so einem hohen Verhältnis ist es kaum möglich, mehr als ein paar Linien zu vervollständigen. Der hohe Prozentsatz kommt dadurch Zustande, dass der ISA-Aufseher gegen 50% schlechte Züge konvergiert. Die Anpassung des ISA-Aufsehers an die Spieler funktioniert dementsprechend nicht gut.

Der durchschnittliche Prozentsatz schlechter Züge liegt beim LR1UV-Aufseher bei 9,66%. Bei dem LR2UV-Aufseher liegt dieser Wert mit 7,56% wesentlich näher am vorgegebenen Wert von 5,39%. Anhand dieser Werte, sowie aus der Verteilung der Prozentsätze zu den Aufsehern, ist zu erkennen, dass der LR2UV-Aufseher die vorgegebene Rate schlechter Züge wesentlich besser annähert als der LR1UV-Aufseher. Der Grund hierfür ist, dass er zusätzlich zur Geschwindigkeit die Höhe der Tetromino betrachtet. Dieser Schluss lässt sich ziehen, da die Betrachtung der Höhe der einzige wesentliche Unterschied des LR2UV-Aufsehers gegenüber dem LR1UV-Aufseher ist.



	ISA	LR1UV	LR2UV
>=40	0%	0%	8%
30-39	0%	12%	6%
20-29	0%	26%	11%
10-19	0%	51%	58%
<10	100%	12%	17%

(a) Spielauswertung: Vervollständigte Linien



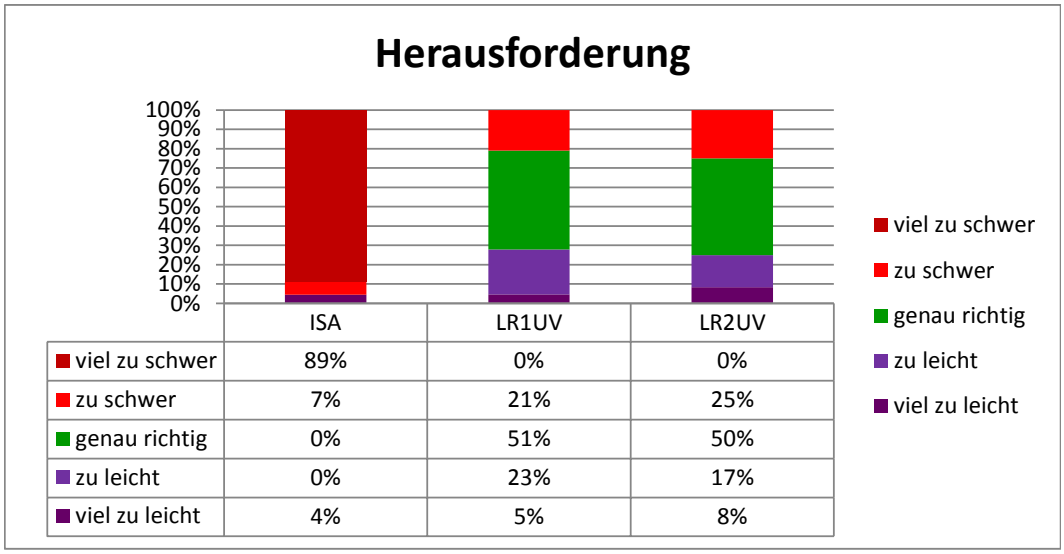
	ISA	LR1UV	LR2UV
>=15%	98%	0%	3%
12.5%-15%	0%	9%	3%
10%-12.5%	2%	33%	3%
7.5%-10%	0%	44%	42%
5%-7.5%	0%	14%	44%
<5%	0%	0%	6%

(b) Spielauswertung: Prozentsatz schlechter Züge

Abbildung 7.2: Nutzerstudie: Ergebnisse der Hintergrunddaten

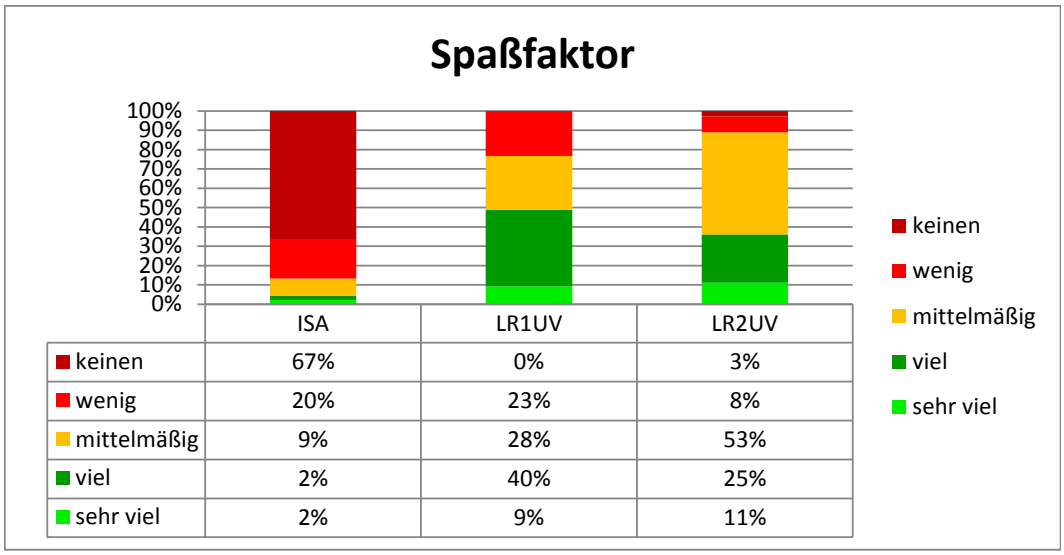
7.4.2 Auswertung der Fragebögen

In Abbildung 7.3 werden die Antworten der Studienteilnehmer dargestellt. Die Frage zur Herausforderung (siehe Abbildung 7.3a) wurde bei Spielen mit dem ISA zu 89% mit „viel zu schwer“ beantwortet. Die Verteilung der Antworten bei Spielen mit den LR-Aufsehern unterscheiden sich kaum. Weniger als jeweils 30% der Spieler empfanden das Spiel als „zu leicht“, jeweils etwa 50% als „genau richtig“ und jeweils weniger als 25% empfanden das Spiel als „zu schwer“.
Der Spielspaß (siehe Abbildung 7.3b) wurde bei Spielen mit dem ISA-Aufseher zu 87% mit „wenig“ bis „keinen“ Spaß angegeben. Nur 4% geben an „viel“ oder „sehr viel“ Spielspaß erlebt zu haben. Bei den Spielen mit dem LR1UV-Aufseher geben 23% an, das Spiel als „wenig“ unterhaltsam empfunden zu haben und 49% geben an, „viel“ oder „sehr viel“ Spaß gehabt zu haben. Alle anderen empfanden das Spiel als „mittelmäßig“ unterhaltsam. Bei Spielen mit dem LR2UV-Aufseher wurde die Frage zu 36% mit „viel“ oder „sehr viel“ Spaß, zu 53% mit „mittelmäßig“ und zu 11% mit „wenig“ bis „keinen“ Spaß beantwortet.



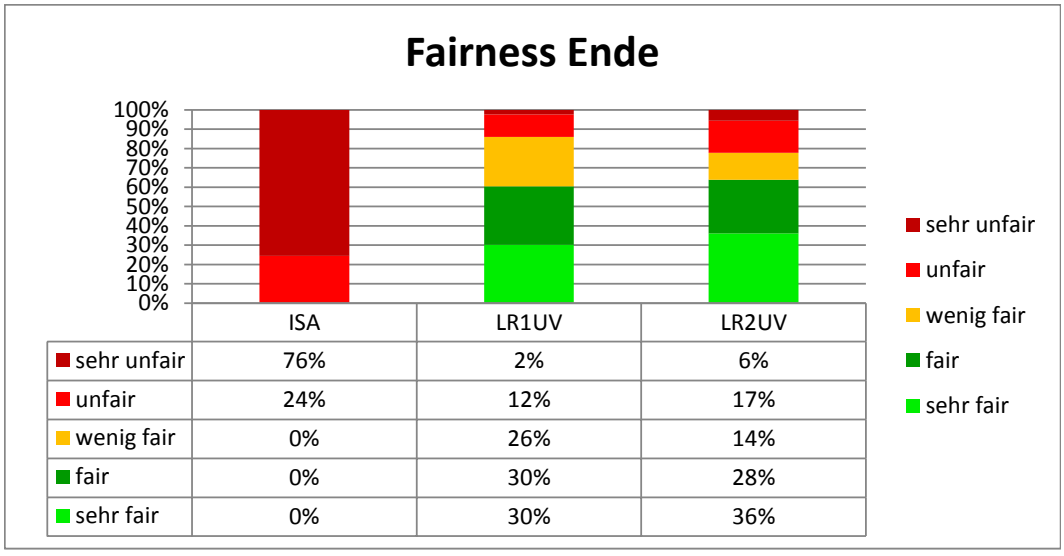
	ISA	LR1UV	LR2UV
viel zu schwer	89%	0%	0%
zu schwer	7%	21%	25%
genau richtig	0%	51%	50%
zu leicht	0%	23%	17%
viel zu leicht	4%	5%	8%

(a) Frageauswertung: Herausforderung



	ISA	LR1UV	LR2UV
keinen	67%	0%	3%
wenig	20%	23%	8%
mittelmäßig	9%	28%	53%
viel	2%	40%	25%
sehr viel	2%	9%	11%

(b) Frageauswertung: Spaßfaktor



	ISA	LR1UV	LR2UV
sehr unfair	76%	2%	6%
unfair	24%	12%	17%
wenig fair	0%	26%	14%
fair	0%	30%	28%
sehr fair	0%	30%	36%

(c) Frageauswertung: Fairness zum Ende des Spiels

Abbildung 7.3: Nutzerstudie: Fragebogenauswertung zu den Aufsehern

Die letzte Frage bezieht sich auf die empfundene Fairness gegen Ende des Spiels (siehe Abbildung 7.3c). Bei dem ISA wurde das Spiel zu 100% als „unfair“ oder „sehr unfair“ bewertet. Spiele mit den LR-Aufsehern wurden jeweils zu etwa 60% als „sehr fair“ oder „fair“ bewertet. Die restlichen Angaben verteilen sich jeweils auf die anderen Antwortmöglichkeiten.

Die zentralen Fragen sind:

- Wie viel Spaß haben die Teilnehmer bei den Spielen mit den jeweiligen Aufsehern gehabt?
- Wie hat sich die empfundene Herausforderung auf den Spielspaß ausgewirkt?
- Wie wirkt sich die Rate schlechter Züge auf die empfundene Herausforderung aus?

Anhand der vorgestellten Verteilungen lässt sich ablesen, dass bei den LR-Aufsehern der meiste Spielspaß empfunden wurde. Diese wurden auch überwiegend als fair und „genau richtig“ herausfordernd bewertet. Spiele mit dem ISA-Aufseher wurden nicht als unterhaltsam empfunden.

Die Annahme des Flow Modells kann in dieser Studie nur zum Teil bestätigt werden. Hierzu wurden alle Spiele betrachtet und nach empfundener Herausforderung sortiert. Die Spielbewertungen können der folgenden Tabelle entnommen werden. Der Wert 1 steht für „keinen“ Spaß und der Wert 5 für „sehr viel“ Spaß.

Herausforderung:	viel zu leicht	zu leicht	genau richtig	zu schwer	viel zu schwer
Spielspaß:	2,00	3,50	3,52	2,98	1,59

Die Spiele, deren Herausforderung als „genau richtig“ oder „zu leicht“ bewertet wurden, wurden am unterhaltsamsten gewertet. Dies steht leicht im Gegensatz zur Aussage des Flow Modells, welche besagt, dass der Spielspaß am positivsten empfunden wird, wenn die Herausforderung „genau richtig“ eingestellt ist.

Im Folgenden wird ausgewertet, wie sich die Rate schlechter Züge auf die empfundene Herausforderung auswirkt. Dazu wurden die Spiele anhand der Prozentsätze schlechter Züge aufgeteilt. In der folgenden Tabelle wird dargestellt, wie herausfordernd die Spiele empfunden wurden. Der Wert -2 entspricht „viel zu leicht“ und der Wert 2 „viel zu schwer“. Zusätzlich wird der durchschnittliche Spielspaß dieser Spiele angegeben

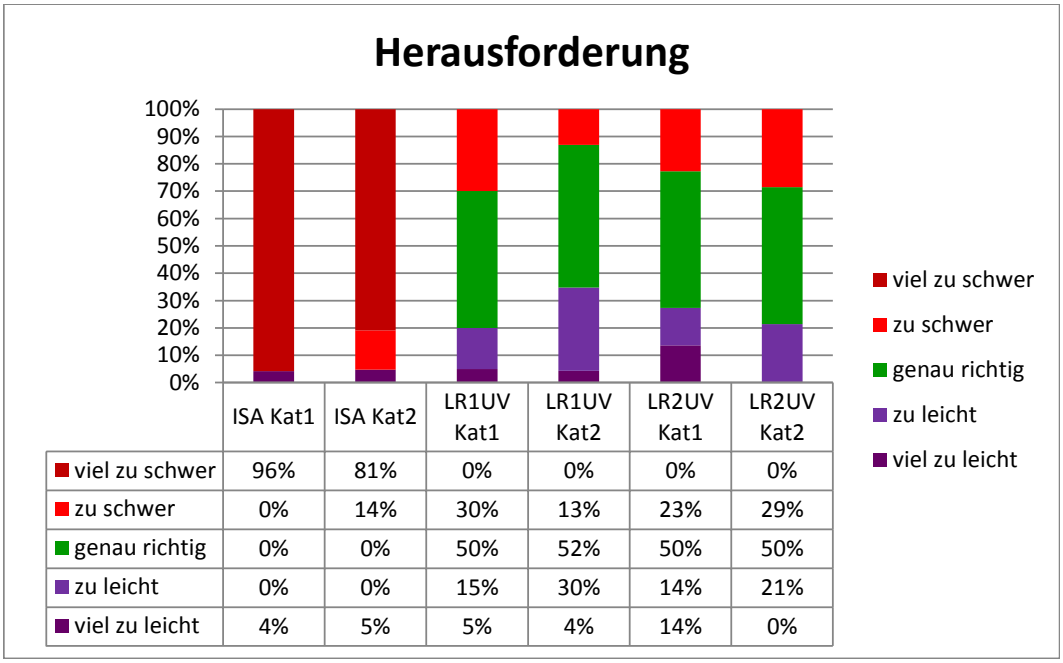
Prozentsatz:	<7,5%	7,5% - 10%	10% - 12,5%	12,5% - 15%	>15%
Herausforderung:	-0,5	0,05	0,5	0,67	1,77
Spielspaß:	3,39	3,38	3,17	3,08	1,77

Die empfundene Herausforderung hängt, wie angenommen, vom Prozentsatz schlechter Züge ab. Ein Prozentsatz zwischen 7,5% und 10% ist optimal, da die Teilnehmer diese Spiele im Schnitt als „genau richtig“ herausfordernd empfunden haben. Dies ist abweichend vom ermittelten optimalen Wert. Jedoch wurden Spiele mit weniger als 7,5% schlechter Züge am positivsten im Hinblick auf den Spielspaß bewertet.

Zusammenfassend lässt sich feststellen, dass die LR-Aufseher sich im Hinblick auf das Ziel, den Spielspaß, nicht viel unterscheiden und beide deutlich besser als der ISA-Aufseher abschneiden.

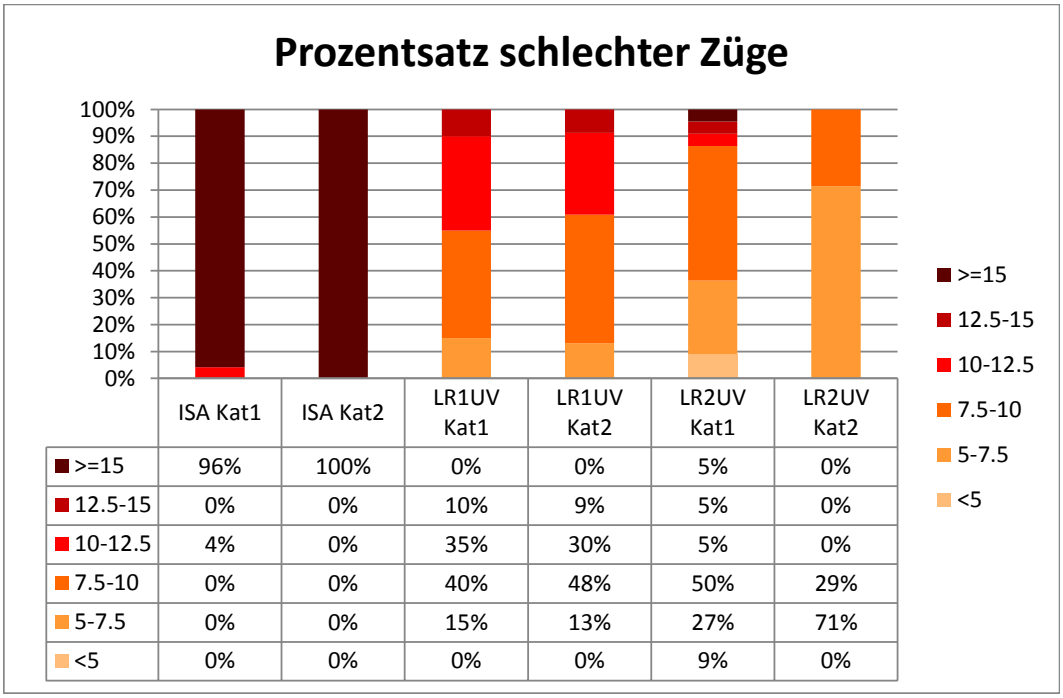
7.4.3 Auswertung nach Kompetenz

Anhand der in Abschnitt 7.3 vorgenommenen Aufteilung wird nun analysiert, wie gut die Aufseher das Spiel auf die verschiedenen Kompetenzniveaus anpassen. Ohne eine Anpassung der Schwierigkeit an die Kompetenz der Spieler ist zu erwarten, dass fortgeschrittene Spieler weniger gefordert sind als Anfänger.



	ISA Kat1	ISA Kat2	LR1UV Kat1	LR1UV Kat2	LR2UV Kat1	LR2UV Kat2
viel zu schwer	96%	81%	0%	0%	0%	0%
zu schwer	0%	14%	30%	13%	23%	29%
genau richtig	0%	0%	50%	52%	50%	50%
zu leicht	0%	0%	15%	30%	14%	21%
viel zu leicht	4%	5%	5%	4%	14%	0%

(a) Frageauswertung: Herausforderung, aufgeteilt nach Kompetenz



	ISA Kat1	ISA Kat2	LR1UV Kat1	LR1UV Kat2	LR2UV Kat1	LR2UV Kat2
>=15	96%	100%	0%	0%	5%	0%
12.5-15	0%	0%	10%	9%	5%	0%
10-12.5	4%	0%	35%	30%	5%	0%
7.5-10	0%	0%	40%	48%	50%	29%
5-7.5	0%	0%	15%	13%	27%	71%
<5	0%	0%	0%	0%	9%	0%

(b) Spielauswertung: Prozentsatz schlechter Züge, aufgeteilt nach Kompetenz

Abbildung 7.4: Nutzerstudie: Auswertung nach Kompetenz

Die Frage zur Herausforderung (siehe Abbildung 7.4a) zu den jeweiligen Aufsehern wurde von den Anfängern sehr ähnlich beantwortet, wie von der den Fortgeschrittenen.
Der ISA-Aufseher wurde von nahezu allen Anfängern als *viel zu schwer* bewertet. Auch die Gruppe der Fortgeschrittenen bewertet den ISA-Aufseher größtenteils als *viel zu schwer* oder *zu schwer*. Anhand dieser Daten lässt sich nicht feststellen, ob der Aufseher auf die Kompetenz der Spieler eingeht, da die Auswertung für beide Gruppen zwar ähnlich ist, aber nicht feststellbar ist, ob das Spiel für fortgeschrittene Spieler schwerer war als für Anfänger.
Der LR1UV- sowie der LR2UV-Aufseher wurden jeweils sehr ähnlich bewertet. Die Anfänger wie auch die Fortgeschrittenen haben zu einem großen Teil (jeweils 50% oder mehr) die Herausforderung als *genau richtig* empfunden. Mehr als jeweils 10% empfanden den Aufseher als *zu schwer* oder als *zu leicht*. Die Profile der Verteilung sind sehr ähnlich, obwohl die Kompetenz der Spieler sehr unterschiedlich ist. Daraus folgt, dass der LR1UV- und der LR2UV-Aufseher die Schwierigkeit gut an die Kompetenz der Spieler angepasst haben.

Ohne die Anpassung der Schwierigkeit an die Kompetenz ist zu erwarten, dass fortgeschrittene Spieler prozentual deutlich weniger schlechte Züge machen als Anfänger. Die Daten zum Prozentsatz schlechter Züge (siehe Abbildung 7.4b) von den Anfängern ist im Allgemeinen sehr ähnlich zu den Daten der Fortgeschrittenen.

Der Prozentsatz schlechter Züge bei dem ISA-Aufseher ist für fast alle Spieler größer als 15%. Dies liegt daran, dass der ISA-Algorithmus mit der gegebenen Heuristik darauf abzielt, dass der Spieler 50% der Züge gut und 50% der Züge schlecht setzt. Da Spiele mit einem so hohen Prozentsatz schlechter Züge sehr kurz sind und in der Regel maximal 1 bis 2 Linien vervollständigt werden bevor das Spiel verloren wird, ist es nicht möglich für den ISA-Aufseher eine Aussage über das Anpassungsverhalten an die verschiedenen Kompetenzen zu machen.
Die Verteilungen der schlechten Züge bei dem LR1UV- und dem LR2UV-Aufseher sind für Anfänger und Fortgeschrittene sehr ähnlich. Für den LR2UV-Aufseher ist eine Tendenz zu sehen, dass die Fortgeschrittenen Spieler prozentual weniger schlechte Züge machen als die Anfänger. Diese Ausprägung ist jedoch bei weitem nicht so stark, wie bei Spielen ohne Anpassung anzunehmen ist. Aus diesen Daten geht hervor, dass der LR1UV- und der LR2UV-Aufseher auf die Kompetenz der Spieler eingehen und die Schwierigkeit des Spieles dementsprechend anpassen.

8 Fazit

In dieser Arbeit wurden Aufseher für Adaptris, deren Ziel die automatische Schwierigkeitsanpassung an Spieler verschiedener Kompetenzen ist, getestet und verglichen. Sie basieren auf verschiedenen Lernalgorithmen.
Die Aufseher basierend auf logistischer Regression mit einer und mit zwei unabhängigen Variablen wurden in dieser Arbeit entwickelt und implementiert. Der bereits von Leischnig implementierte Aufseher basierend auf dem ISA wurde in der Arbeit mit den anderen Aufsehern verglichen.

Für die Lernalgorithmen wurde eine Heuristik entwickelt, die feststellt, ob ein Spieler einen guten oder ein schlechten Zug gemacht hat. Dazu wurde unter anderem eine Heuristik von Yiyuan implementiert. Für diese wurde gezeigt, dass sie in Adaptris als optimal spielend angenommen werden kann.
Durch simulierte Spiele wurde eine Schwierigkeitseinstellung gefunden und definiert, bei der Spieler in Adaptris weder unter- noch überfordert sind.

Für den Vergleich der Aufseher wurden Spieler mit unterschiedlicher Kompetenz simuliert und Spiele mit ihnen durchgeführt. Außerdem wurde die bereits von Leischnig implementierte Nutzerstudie weiterentwickelt und durchgeführt. Hierbei wurde getestet, wie Spieler die adaptierten Tetrisspiele wahrnehmen.

Anhand der Ergebnisse der Nutzerstudie und der simulierten Spiele konnte gezeigt werden, dass die Spielschwierigkeitsanpassung in Adaptris mit den Aufsehern basierend auf logistischer Regression generell funktioniert. Für den ISA-Aufseher ist dies mit der verwendeten Heuristik nicht der Fall. Es hat sich herausgestellt, dass der LR2UV-Aufseher die vorgegebene Schwierigkeitseinstellung besser annähert als der LR1UV-Aufseher. Die Parameter Fallgeschwindigkeit und Höhe des Spielfeldes oder das Äquivalent, die zur Verfügung stehende Zeit, einen Zug durchzuführen, sollten daher genutzt werden, um die Schwierigkeit anzupassen.

Es wurde festgestellt, dass mit der gegebenen initialen Datenmenge der LR2UV-Aufseher für sehr gute Spieler deutlich langsamer konvergiert als für Anfänger und Fortgeschrittene. Eine mögliche Erklärung wurde erläutert. Sie hängt damit zusammen, dass die logistische Funktion in einem Update nur geringfügig nach oben angepasst werden kann, wohingegen die Anpassung nach unten in einem Schritt deutlich größer ist. Dies wiederum beruht auf der Tatsache, dass Feedback für einen guten Zug deutlich näher an der logistischen Funktion liegt als das Feedback für einen schlechten Zug.

Die Ergebnisse zu dem ISA haben gezeigt, dass eine gute Schwierigkeitsanpassung mit der vorgestellten Heuristik nicht möglich ist, aber es konnte gezeigt werden, dass der ISA schneller konvergiert als der Lernalgorithmus basierend auf LR2UV.

9 Ausblick

Wie bereits beschrieben, bleibt zu prüfen, ob der ISA mit einer angepassten Heuristik, die zusätzlich das Feedback „genau richtig“ zurückgeben kann, eine bessere Anpassung liefern würde. Erste Tests mit einer Heuristik dieser Art schienen die Schwierigkeit recht zufällig anzupassen. Das liegt daran, dass das Feedback in vielen Fällen nicht richtig ist.
Es ist zu untersuchen, ob eine Heuristik gefunden werden kann, die präzise unterscheiden kann, ob ein Spieler unterfordert ist, die Schwierigkeit richtig eingestellt ist oder ob der Spieler überfordert ist. Vor allem die Unterscheidung zwischen den Feedbacks „zu leicht“ und „genau richtig“ ist schwierig. Wenn nur darauf geachtet wird, ob der Spieler den Tetromino beschleunigt, ist die Heuristik sehr leicht zu durchschauen, was dazu führt, dass die Spieler ihre Spielweise an die Heuristik leicht anpassen können. Dies ist nicht der Sinn der Aufseher.

Im ISA gibt es einen einstellbaren Parameter, durch den andere Werte ausgewählt werden würden. Erste Tests mit der Heuristik aus dieser Arbeit haben ergeben, dass der ISA auch mit anderen Parametereinstellungen gegen einen Prozentsatz von 50% konvergiert. Der Parameter hat bewirkt, dass dieser Wert von 50% langsamer oder schneller angenähert wurde. Es bleibt zu prüfen, ob auch andere Konvergenzwerte erreicht werden können.

Der prozentuale Wert schlechter Züge, für den die Herausforderung des Spieles als genau richtig angenommen wird, wurde durch Simulationen bestimmt. Die Nutzerstudie hat gezeigt, dass der Wert von 5,39% noch optimiert werden kann.

Die initialen Datenmengen für die Aufseher basierend auf logistischer Regression sind plausibel und haben sich bewährt. Zu untersuchen ist jedoch, wie gut andere initiale Datenmengen sind, insbesondere im Hinblick darauf, wie schnell die Aufseher konvergieren und wie robust sie dabei sind.

Ein weiterer Ansatz zur Verbesserung der Aufseher ist, die Faktoren der Schwierigkeit besser zu messen. Die Schwierigkeit eines Zuges kann von dem aktuellen Tetromino abhängig sein. Es ist leichter einen passenden Tetromino gut zu platzieren als bei einem nicht passenden Tetromino die Platzierung zu finden, die am wenigsten schlecht ist. Eine Möglichkeit ist, den aktuellen Tetromino mit in die Schwierigkeitsbewertung einzubeziehen. Eine weitere Möglichkeit ist, die Schwierigkeit nicht nur über die Fallgeschwindigkeit anzupassen, sondern auch zu steuern, welche Tetrominos der Spieler im Spiel bekommt.

Eine letzte Möglichkeit, die hier vorgeschlagen wird, ist die automatische Schwierigkeitsanpassung über mehrere Spiele zu machen. Das bedeutet, dass die Lernalgorithmen die Daten für einen Spieler über mehrere Spiele hinweg speichern und nutzten. Dadurch ist die initiale Datenmenge bei den Spielen nicht durch die in dieser Arbeit vorgestellte geschätzte Datenmenge gegeben, sondern durch die Datenmenge, die der Spieler in den vorherigen Spielen erzeugt hat. Dieser Ansatz ist vielversprechend, da die Lernalgorithmen mehr Feedback bekommen und die Schwierigkeit auch am Anfang der Spiele auf den Spieler abgestimmt ist.

Literaturverzeichnis

[1] P. Tozour, *AI Game Programming Wisdom*, ch. The Evolution of Game AI, pp. 3–15. Charles River Media, 2002.

[2] C. R. Snyder and S. J. Lopez, *Handbook of positive psychology*, ch. The concept of flow, pp. 89–105. Oxford University Press USA, 2002. Autor des Kapitels: J. Nakamura and M. Csikszentmihalyi.

[3] D. Bengs and U. Brefeld, "A Learning Agent for Parameter Estimation in Speeded Tests," in *Proceedings of the ECML/PKDD Workshop on Reinforcement Learning with Generalized Feedback: Beyond Numeric Rewards*, 2013.

[4] Zingel, "Nominal, ordinal und metrisch: kleine Übersicht über die Datentypen der Statistik." Online: http://www.bwl24.net/blog/2008/06/27/nominal-ordinal-und-metrisch-kleine-ubersicht-uber-die-datentypen-der-statistik/ Stand: 24.05.2015, 2008.

[5] S. A. Czepiel, "Maximum likelihood estimation of logistic regression models: Theory and implementation." Online: http://czep.net/stat/mlelr.pdf Stand: 24.05.2015.

[6] S. Albers *et al.*, eds., *Methodik der empirischen Forschung*, ch. Logistische und Ordinale Regression, pp. 267–282. Gabler, 3 ed., 2009. Autor des Kapitels: Christian Rohrlack.

[7] W. Ahlborn, *Prüfung auf Autokorrelation der Störvariablen in linearen Regressionsmodellen mit verzögerten endogenen Variablen als Regressoren: eine vergleichende Untersuchung*. O. Schwartz, 1982.

[8] J. H. Aldrich and O. D. Nelson, *Linear Probability, Logit, and Probit Models*, vol. 45, p. 49. Sage, 1984.

[9] T. P. Minka, "Algorithms for maximum-likelihood logistic regression," tech. rep., Carnegie Mellon University, 2003.

[10] S. Leischnig, "Adaptris - Tetris with dynamic difficulty as implementation of the ISA algorithm," Bachelorarbeit, Technische Universität Darmstadt, 2013.

[11] L. Yiyuan, "Tetris ai – the (near) perfect bot." Online: https://codemyroad.wordpress.com/2013/04/14/tetris-ai-the-near-perfect-player/ Stand: 24.05.2015, 2013.