GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN

SOMMERSEMESTER 2012

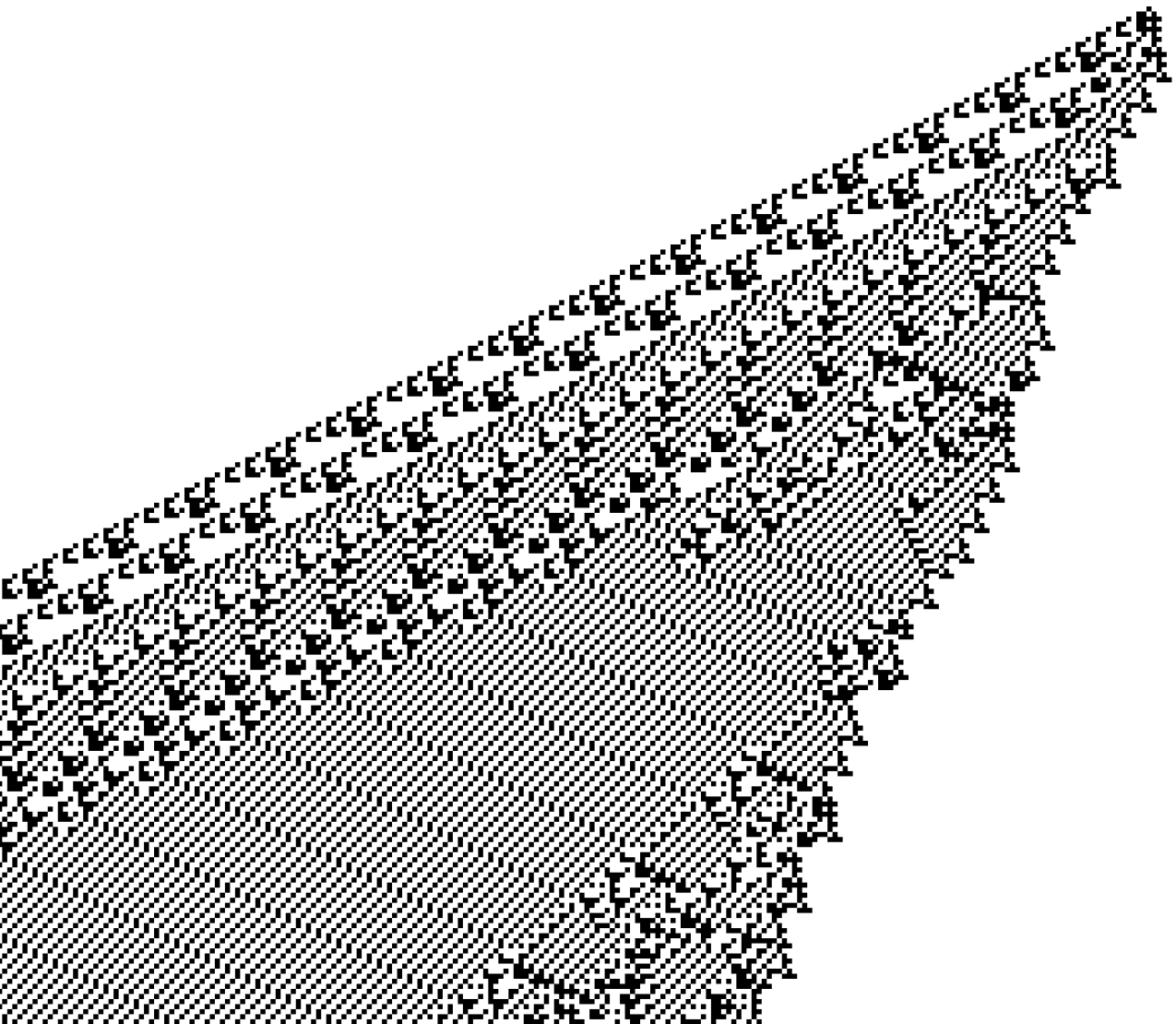
Shigeru Ninagawa

Evolution of one-dimensional cellular automata by $1/f$ noise

Ausarbeitung zum Proseminar „Artificial Life“

Pascal Schmitt

17. September 2012



Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Göttingen, den 17. September 2012 

Titelbild 200 Zeitschritte der 1-2-5-Regel 1 289 134 406

Georg-August-Universität Göttingen
Fakultät für Forstwissenschaften und Waldökologie
Abteilung Ökoinformatik, Biometrie und Waldwachstum

Büsgenweg 4
D-37077 Göttingen

Tel. +49 (5 51) 39 9715
Fax +49 (5 51) 39 3465
Email fbiom@uni-forst.gwdg.de
WWW http://www.uni-goettingen.de/de/67072.html

Email pascal.schmitt1@stud.uni-goettingen.de

Inhaltsverzeichnis

1 Einleitung

Zelluläre Automaten, insbesondere das „Game of Life“, faszinieren durch die trotz ihrer einfachen Regeln hohe Komplexität der Ausgabe. Bestimmte Regeln eignen sich sogar zur Simulation universeller Turingmaschinen, die somit lediglich zu speziellen Eingaben zellulärer Automaten werden.

Stephen Wolfram postulierte 1985, dass Regel 110 der einfachsten Klasse eindimensionaler zellulärer Automaten diesen Universalitätscharakter besitzt. Matthew Cook veröffentlichte 2004 einen Beweis [1], in dem Regel 110 ein *cyclic tag system* emuliert, eine universelle Zustandsmaschine, deren Band eine *queue* ist.

Shigeru Ninagawa untersuchte in [8] das Frequenzspektrum der Ausgabe von Regel 110 und fand ein charakteristisches $1/f$-Muster, das einen Hinweis auf die Universalität einer Regel geben könnte. Um diese These zu prüfen, sucht er im vorliegenden Paper [7] im Raum komplexerer zellulärer Automaten nach Regeln, die ein ähnliches Frequenzspektrum erzeugen, in der Erwartung, Hinweise auf eine Fähigkeit zur universellen Berechnung zu finden.

Dabei wird nicht lediglich eine Stichprobe des Raums betrachtet, sondern mittels eines genetischen Algorithmus Regeln mit möglichst passendem Frequenzspektrum gesucht und schließlich qualitativ auf Hinweise auf universelle Berechnungsfähigkeiten untersucht, etwa *glider*.

2 Funktionsweise zellulärer Automaten

Zelluläre Automaten sind Ersetzungssysteme auf gleichmäßigen Rastern: eine feste Regel ϕ wird auf die Nachbarschaft ν jeder Zelle x_i^t angewendet um deren Farbe $x_i^{t+1} \in \Sigma$ im nächsten Zeitschritt $t+1$ zu erhalten, also $x_i^{t+1} := \phi\left(\nu\left(x_i^t\right)\right)$. Hier werden eindimensionale Automaten betrachtet, d.h. Zeilen $x^t \in \Sigma^W$ der Breite W und Nachbarschaften im Radius r. Jede Zelle hat also $N = 2r+1$ direkte Nachbarn:

$$\nu\left(x_i^t\right) = \left(x_{i-r}^t, \dots, x_i^t, \dots, x_{i+r}^t\right)$$
$$\phi : \Sigma^N \to \Sigma$$

Das Alphabet Σ ist durch die Anzahl der Farben $k = |\Sigma|$ definiert, hier ist $k = 2$ und $\Sigma = \{0, 1\}$. Für elementare zelluläre Automaten (ECAs) ist $r = 1$ festgelegt.

Nach Wolframs Benennungsschema aus [10] ergibt sich somit z.B. für den ECA $110_{10} = 01101110_2$ folgende Regeltabelle:

ν	111	110	101	100	011	010	001	000
$\phi(\nu)$	0	1	1	0	1	1	1	0

Für $r = 2$ werden entsprechend fünf Nachbarn jedes Pixels verwendet, es gibt also $|\nu| = 2^5 = 32$ verschiedene Nachbarschaften.

ν	11111	11110	11101	$\cdots$	00011	00010	00001	00000
$\phi(\nu)$	R_{31}	R_{30}	R_{29}	$\cdots$	R_3	R_2	R_1	0

In der Tradition des „Game of Life“ wird $s_q = 0$ als der „Totzustand“ bzw. Hintergrund betrachtet, aus dem immer s_q folgt, es interagieren sozusagen nur die „Lebendzustände“ $\Sigma \setminus \{s_q\} = \{1\}$ miteinander.

Im Folgenden werden Durchläufe ausgehend von x^0 mit $p(0) = p(1) = ½$ mit Breite $W = 700$ über $T = 3000$ Zeitschritte betrachtet. Dabei wird die Nachbarschaft mit periodischen Randbedingungen verwendet, d.h.

$$x_i^t = x_{i \bmod W}^t$$
$$\nu\left(x_i^t\right) = \left(x_{(i-r) \bmod W}^t, \dots, x_i^t, \dots, x_{(i+r) \bmod W}^t\right)$$

2.1 Komplexitätsklassen

In [11] teilt Wolfram zelluläre Automaten qualitativ in vier Klassen ein:

Klasse 1 endet für beliebige Eingaben in einem homogenen Zustand, d.h. Zufall aus der Eingabe verschwindet. Das Verhalten muss nicht unbedingt trivial sein, es existieren z.B. Automaten deren homogener Endzustand davon abhängt, welches Symbol in der Eingabe am häufigsten vertreten ist.

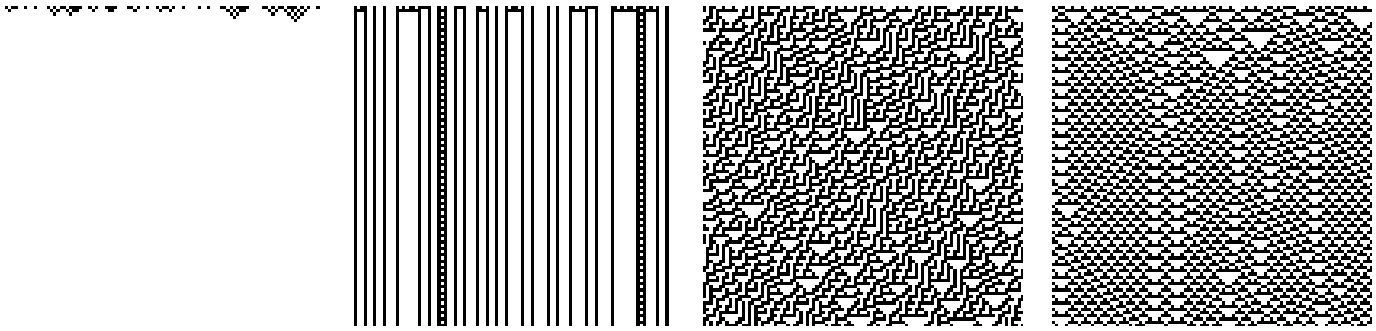
Klasse 2 führt zu einfachen getrennten stabilen oder periodischen Strukturen, dazu gehören Automaten, die den Eingabezustand unverändert kopieren oder verschieben, und evtl. einfache *blinker* erzeugen.

Klasse 4 führt zu komplexen lokalen, teils langlebigen, interagierenden Strukturen, etwa die aus dem „Game of Life“ bekannten *blinker* und festen Strukturen, *glider*, die interagieren können und neue Strukturen erzeugen, z.B. *glider guns*, etc.

Klasse 3 führt in chaotische Muster, etwa Regel 30, die aus einem 1-Pixel auf 0-Hintergrund auf einer Seite Pseudozufall erzeugt. Bei zufälliger Eingabe nimmt die „Zufälligkeit“ nicht ab.

Wie [3] erläutert, deckt diese Einteilung allerdings nicht alle möglichen Automaten ab, Clusterbildung ist z.B. ähnlich Klasse 4 an der Grenze zwischen Klasse 1 und Klasse 2 zu finden. In [5] wird die qualitative Klasseneinteilung durch quantitative Bewertungen in Abhängigkeit von λ ergänzt.

Wolfram vermutet, alle Klasse 4-Automaten seien universell, bislang wurde dies allerdings nur für Regel 110 und das „Game of Life“ bewiesen. Diese sind lediglich



(a) Klasse 1: Regel 160 (b) Klasse 2: Regel 108 (c) Klasse 3: Regel 30 (d) Klasse 4: Regel 54

Abb. 1: Ausgabe beispielhafter Regeln für jede Klasse.

„Turing vollständig", können also universelle Turingmaschinen simulieren, aber z.B. nicht beliebige zelluläre Automaten, die unendlich viele Turingmaschinen benötigen würden.

3 Spektrale Leistungsdichte der Ausgabe

Um die spektrale Leistungsdichte (*power spectrum*) $S(f)$ eines Durchlaufs zu erhalten, wird die diskrete Fouriertransformation auf die Zeitreihe $x_i(t) := x_i^t$ jeder Spalte i angewendet um deren Frequenzreihe $\hat{x}_i(f)$ zu erhalten:

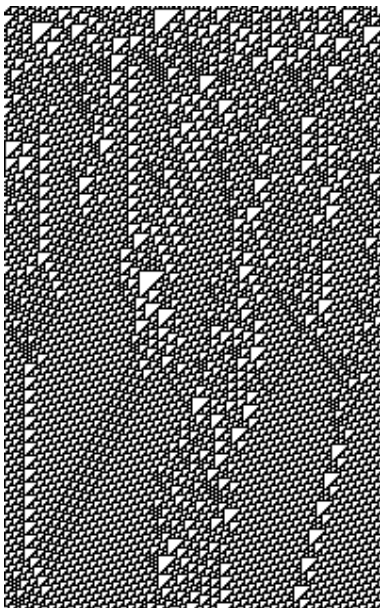
$$\hat{x}_i(f) = \frac{1}{T}\sum_{t=0}^{T-1} x_i(t)\, \mathrm{e}^{-\mathrm{i}\, 2\pi t f/T} \qquad S(f) = \sum_{i=0}^{W-1} |\hat{x}_i(f)|^2$$

Die Größe $S(f)$ beschreibt nun die Stärke eines Musters mit Periode T/f, ein beispielhafter Verlauf für Regel 110 ist in Abb. 2 zu sehen. Ein Peak bei $f = 428$ deutet dort z.B. auf oft erscheinende Muster mit Periode $3000/428 = 7$ hin. In Abb. 3 ist $S(f)$ über T/f dargestellt. Für die gewählte Eingabe x^0 bei $W = 700$ wiederholt sich die Ausgabe nach 2655 Schritten, weshalb das über 300000 Schritte berechnete Spektrum dort einen starken Peak gefolgt von einem flachen Verlauf zeigt: es gibt keine Muster mit längerer Periode.

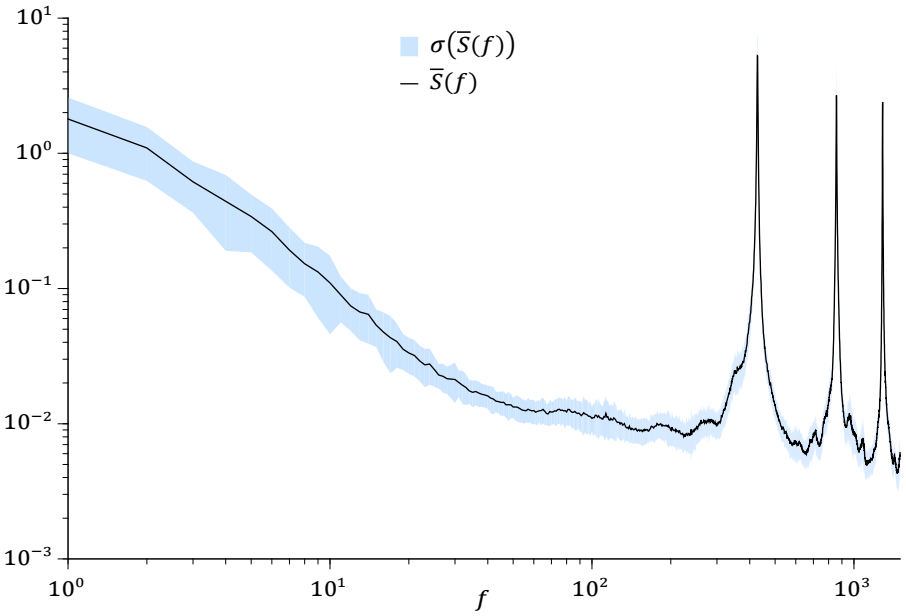
3.1 $1/f$-Noise

„Pinkes Rauschen" oder „$1/f$-Noise" ist in [4] definiert als ein Zufallsprozess, dessen spektrale Leistungsdichte sich annähernd umgekehrt proportional zur Frequenz verhält:

$$S(f) \sim \frac{1}{f^\gamma} \qquad 0 < \gamma < 2$$



(a) $W = 200, T = 320$



(b) Frequenzspektrum $W = 700, T = 3000$, 40 Eingaben

Abb. 2: Regel 110.

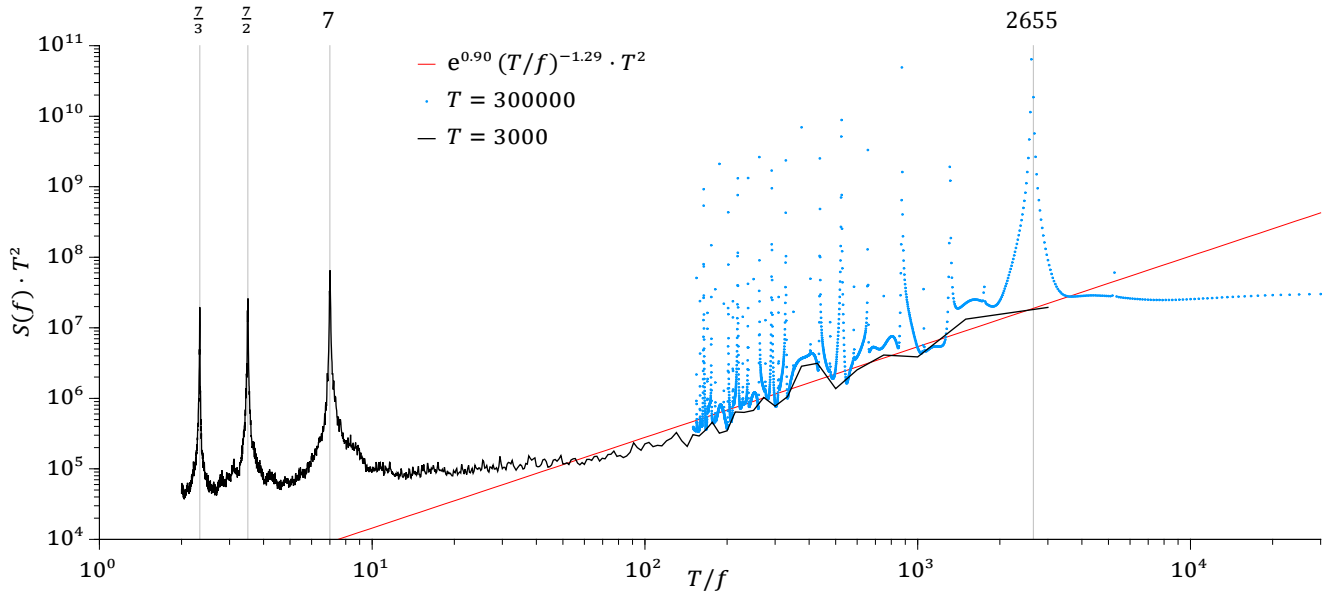


Abb. 3: Peaks im Spektrum zeigen die Periode von Mustern an.

Ein System, das $1/f$-Noise erzeugt, wird von seinen vorherigen Zuständen beeinflusst, verfügt also über eine Speichermöglichkeit, in elektrischen Schaltungen etwa Kondensatoren, in einem Zellulären Automat etwa die Muster *glider* oder *blinker*, die Informationen über mehrere Zeitschritte speichern, um sie zu einem späteren Zeitpunkt zu verwerten (z.B. Kollision zweier *glider*). Der Exponent γ beschreibt, wie groß das Gedächtnis des Systems ist: Für $\gamma \approx 1$ wird auf weit zurückliegende Information gleich oft zugegriffen wie auf neuere, für $\gamma \approx 0$ oder $\gamma \approx 2$ fast nur auf neuere.

Ninagawa vermutet $1/f$-Noise als Indiz für Klasse 4-Automaten, da diese durch ihren Wechsel zwischen Klasse 2 und Klasse 3-Verhalten immer $1/f$-Noise erzeugen müssen, während anderen Klassen das erforderliche „Gedächtnis“ fehlt.

4 Genetischer Algorithmus zur Suche nach Regeln

Der genetische Algorithmus versucht die durch die Funktion F bestimmte Fitness für die gesamte Population P zu maximieren. Dazu wird zunächst eine Startpopulation von P Regeln bzw. Genomen generiert. Bis das Abbruchkriterium erreicht ist wird für jede Regel F berechnet und mit Hilfe der Fitnesswerte durch Kreuzung und Mutation die jeweils nächste Generation erzeugt.

4.1 Startpopulation

Die Regeln ϕ lassen sich durch einen Parameter λ charakterisieren. Dieser ist in [5] als der Anteil der Übergänge definiert, die nicht in den Totzustand $s_q \in \Sigma$ führen:

$$\lambda := \frac{\#\{v \in \Sigma^N \mid \phi(v) \neq s_q\}}{k^N} = \frac{\#\{R_i \mid R_i = 1\}}{32}$$

Dies entspricht hier dem Anteil der 1-Bits im Regel-String $R_{31}, \dots, R_1$. Im Folgenden werden nur Regeln mit $\lambda \leq {}^{16}\!/_{32}$ betrachtet, da die Regeln ${}^{16}\!/_{32} < \lambda$ trivial durch Invertieren der Bits einer äquivalenten Regel abgeleitet werden können. Die weiteren in [6] beschriebenen Äquivalenzen, z.B. Spiegelung und Rotation, werden hier nicht verwendet.

Um den für diese Automatenklasse kritischen λ-Wert zu finden, werden für die Startpopulation Regeln mit zufälligen, gleichverteilten λ erzeugt, es gilt

$$p(R_i = 0) = \lambda \qquad \text{und} \qquad p(R_i = 1) = 1 - \lambda.$$

4.2 Bewertung der Regeln

Für jede Generation werden $m = 10$ zufällige Eingaben x^0 erzeugt und jede Regel mit diesen für $T = 3000$ Zeitschritte ausgeführt. Aus dem Leistungsspektrum $S(f)$ der Ausgabe wird dann die Fitness F für das Regel-Eingabe-Paar berechnet. Die Gesamtfitness $\overline{F}$ der Regel ergibt sich aus dem arithmetischen Mittel.

Gesucht werden Regeln, deren Ausgabe $1/f$-Noise erzeugt, es soll also gelten:

$$S(f) \sim \frac{1}{f^\gamma} \qquad \text{mit } \beta = -\gamma$$

$$S(f) = \mathrm{e}^\alpha \cdot \frac{1}{f^{-\beta}} = \mathrm{e}^\alpha f^\beta = \mathrm{e}^\alpha \mathrm{e}^{\beta \log f} = \mathrm{e}^{\alpha + \beta \log f}$$

Im doppeltlogarithmischen Plot erhalten wir eine Gerade mit Steigung β und Achsenabschnitt α:

$$\log S(f) = \alpha + \beta \log f$$

Durch lineare Regression [9] über die doppeltlogarithmischen Koordinaten lassen sich Werte für α und β bestimmen die den Fehler σ^2 minimieren:

$$x_f = \log f \qquad y_f = \log S(f)$$

$$\sigma^2_{1,\dots,f_b} = \sum_{f=1}^{f_b} \frac{\left(y_f - (\alpha + \beta\, x_f)\right)^2}{f_b}$$

$$\beta = \sum_{f=1}^{f_b} \left(x_f - \overline{x}\right) \cdot \left(y_f - \overline{y}\right) \Bigg/ \sum_{f=1}^{f_b} \left(x_f - \overline{x}\right)^2$$

$$\alpha = \overline{y} - \beta\, \overline{x}$$

Im Paper wird die lineare Regression über die ersten $f_b = 10$ Frequenzen berechnet, in der Fitnessfunktion allerdings die Residuen über $f_r = 100$ Frequenzen verwendet, wahrscheinlich um die ersten 10 Frequenzen höher zu gewichten. Die Fitnessfunktion F ordnet schließlich „besseren“ Regeln größere Zahlen zu, so dass der Algorithmus versucht β und σ^2 zu minimieren.

$$F = \begin{cases} -\beta / \left(\sigma^2_{1,\dots,f_r} + \delta\right) & \text{falls } \beta < 0 \qquad \delta = 10^{-6} \\ 0 & \text{sonst} \end{cases}$$

Entgegen der Beschreibung in [4] wird β also nur minimiert, anstatt es -1 anzunähern, d.h. $1/f$-Noise mit möglichst langem Gedächtnis. Im Paper wird dies nicht begründet. Die beste dort gefundene Regel hat $\beta = -2.0$, dürfte demnach kein „Gedächtnis“ besitzen. Sie erzeugt allerdings *glider*, speichert folglich Informationen, wahrscheinlich ist die physikalisch motivierte Definition aus [4] also nicht direkt auf Zelluläre Automaten anwendbar.

4.3 Erzeugen der nächsten Generation

Die $E = 20$ Regel mit der höchsten Fitness werden zunächst unverändert in die nächste Generation kopiert. Dies versichert, dass zumindest die Fitness der E besten Regeln maximiert wird, sofern sie nicht zu stark von der Eingabe abhängt.

Der Algorithmus simuliert zwei essentielle Prozesse der natürlichen Evolution: Paarung mit Crossover und Mutation. Zur Paarung werden zwei Eltern per *roulette selection* ausgewählt. Dabei ist die Wahrscheinlichkeit, eine Regel i auszuwählen proportional zu deren Fitness F_i:

$$P(\text{select } i) = \frac{\overline{F}_i}{\sum_{j=1}^{P} \overline{F}_j}$$

Der Name bezieht sich auf die übliche Implementierung: es wird eine gleichverteilte Zufallszahl $r \in \left[0, \sum_{j=1}^{P} \overline{F}_j\right]$ gezogen, und über die kumulative Summe der Fitnesswerte auf i umgerechnet:

$$\text{select } i \Leftrightarrow \sum_{j=1}^{i-1} \overline{F}_j \leq r < \sum_{j=1}^{i} \overline{F}_j$$

Wurden so zwei Eltern R^1 und R^2 selektiert, findet mit $P_c = 0.6$ ein *uniform crossover* statt. Dabei werden für alle $k \in [1, 31]$ mit jeweils $P_s = 0.5$ die Regelbits $R_k^1 \leftrightarrows R_k^2$ getauscht. Anschließend wird jedes Regelbit mit $P_m = 0.05$ mutiert, also invertiert. Die beiden so entstandenen neuen Regeln R^1 und R^2 werden der nächsten Generation hinzugefügt. Dies wird so lange wiederholt, bis diese wieder $P = 160$ Regeln enthält.

5 Ergebnisse

Im Paper wird die $\lambda = {}^{13}\!/_{32}$-Regel 10000000110100111011100000011010 als bestes Ergebnis präsentiert. Wie zu erwarten handelt es sich um eine Klasse 4-Regel, die wie in Abb. 4 zu erkennen auf einem einfachen Hintergrund drei Arten periodischer Strukturen und einen *glider* hervorbringt, der sich in jedem Zeitschritt einen Schritt nach links bewegt und bei Kollision mit den Strukturen diese zu einem Sprung nach rechts anregt. Es ist fragwürdig, ob diese Regel tatsächlich zu interessanten Berechnungen fähig ist, Regel 110 erzeugt z.B. eine größere Vielfalt an Strukturen. Ninagawa untersucht die Fähigkeiten der gefundenen Regel nicht weiter, da das Ziel des Papers lediglich sei, $1/f$-Noise als gutes Kriterium zu bestätigen. Allerdings wirkt die Regel weniger interessant als einige ECAs, der zusätzliche Aufwand den $r = 2$ mit sich bringt bleibt daher unbegründet. Wolfram z.B. betrachtet hier nur die 2^5 totalistischen Automaten, die, wie das „Game of Life“ nur die Anzahl der 1 in Nachbarschaft betrachten, nicht deren genaue Anordnung.

Abb. 4: Die beste Regel über $W = 720$ für $T = 320$.

5.1 λ-Verteilung

Da nach Klasse 4-Regeln gesucht wurde, bewegt sich λ der besten Ergebnisse, wie aus [5] zu erwarten, in einem engen Rahmen, λ_c ist hier etwa $^{14}/_{32}$ wie das Histogram in Abb. 5 zeigt. Das Paper verwendet $f_b = 10, f_r = 100$, betrachtet also nur die ersten zehn Frequenzen. Werden stattdessen mit $f_b = 100, f_r = 400$ die ersten hundert betrachtet, ergibt sich ein stärkerer Peak um λ_c. Allerdings werden z.B. mit $f_b = f_r = 400$ keine Klasse 4-Regeln mehr gefunden.

Im zeitlichen Verlauf des Histograms in Abb. 6(a) ist zu beobachten, dass der Algorithmus ca. 30 Generationen benötigt, die Regeln in den gewünschten λ-Bereich zu optimieren. Interessant ist, dass obwohl in der Startgeneration λ-Gleichverteilung herrscht, in den folgenden fünf Generationen zunächst niedrige λ überwiegen. Da nun bekannt ist, in welchem λ-Bereich sich Klasse 4-Verhalten finden lässt, könnte der Suchraum direkt eingeschränkt werden.

Dass die Einschränkung des Startbereichs auf $^1/_{32} \leq \lambda \leq {}^{16}/_{32}$ sinnvoll ist, zeigt Abb. 6(b): Läufe enden mit λ um $^{12}/_{32}$ oder $^{20}/_{32}$, das Ergebnis ist also um $^{16}/_{32}$ spiegelsymmetrisch, was durch invertieren der Bits R_i erreicht werden kann.

5.2 Zeitlicher Ablauf der Optimierung

Abb. 7(a) zeigt den Verlauf der Fitness der „Elite“. Abgesehen von zufallsbedingten Schwankungen steigt sie wie zu erwarten, allerdings nicht durchgehend: der Algorithmus bewegt sich zwischen Plateaus in der Fitnesslandschaft und wechselt

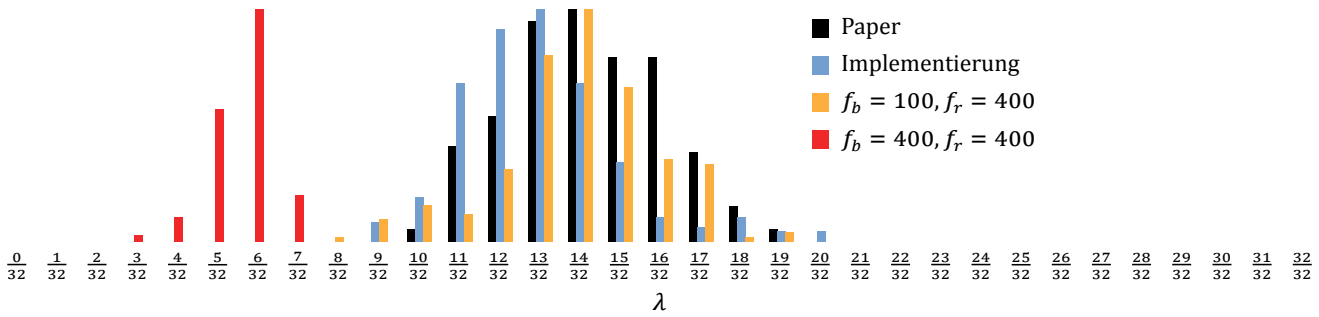
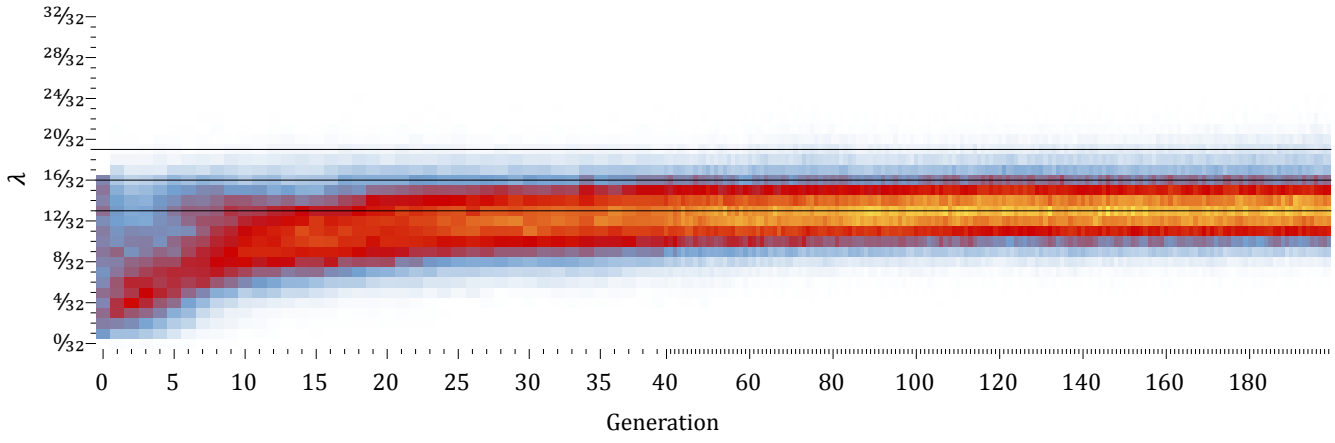
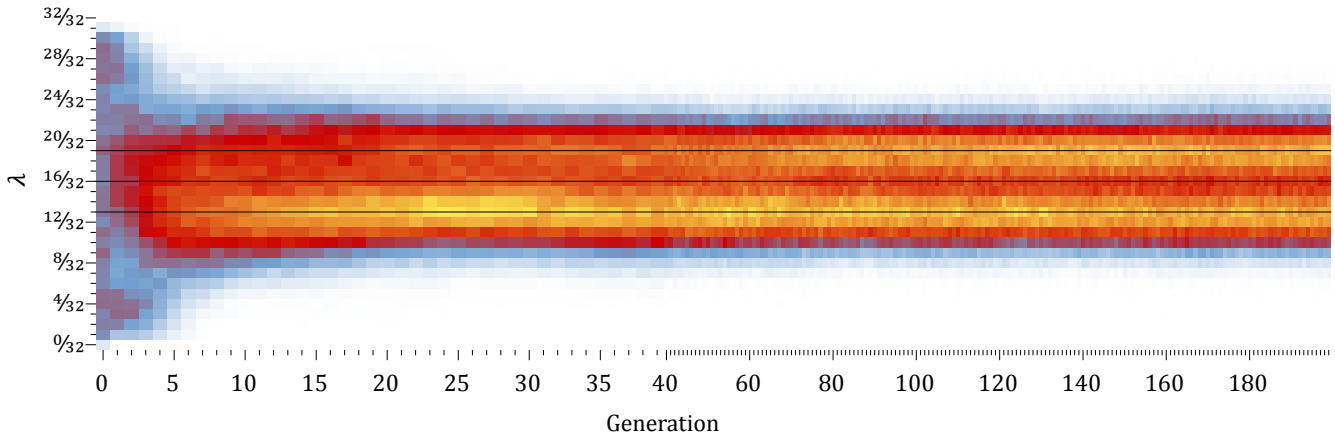


Abb. 5: Histogram der λ-Werte für die jeweils 10 besten Regeln aus je 20 Durchläufen mit verschiedenen Parametern.

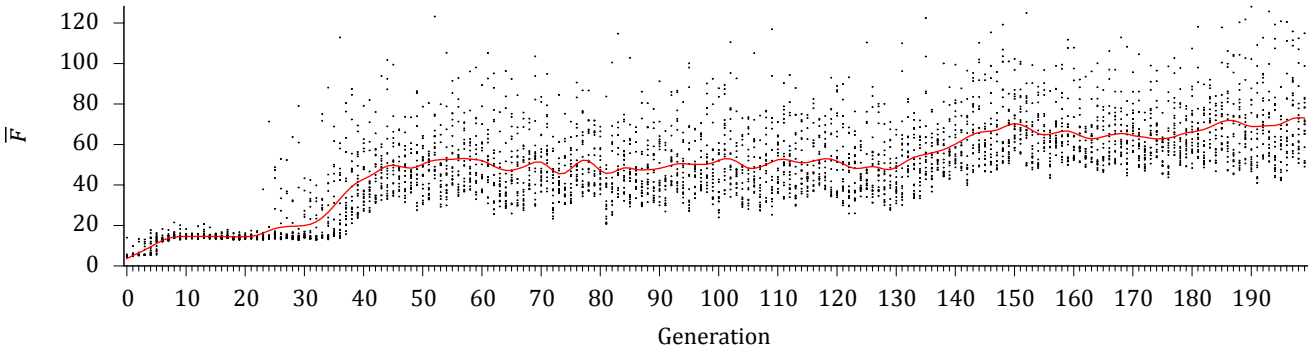


(a) Start mit $1/32 \leq \lambda \leq 16/32$, 20 Läufe

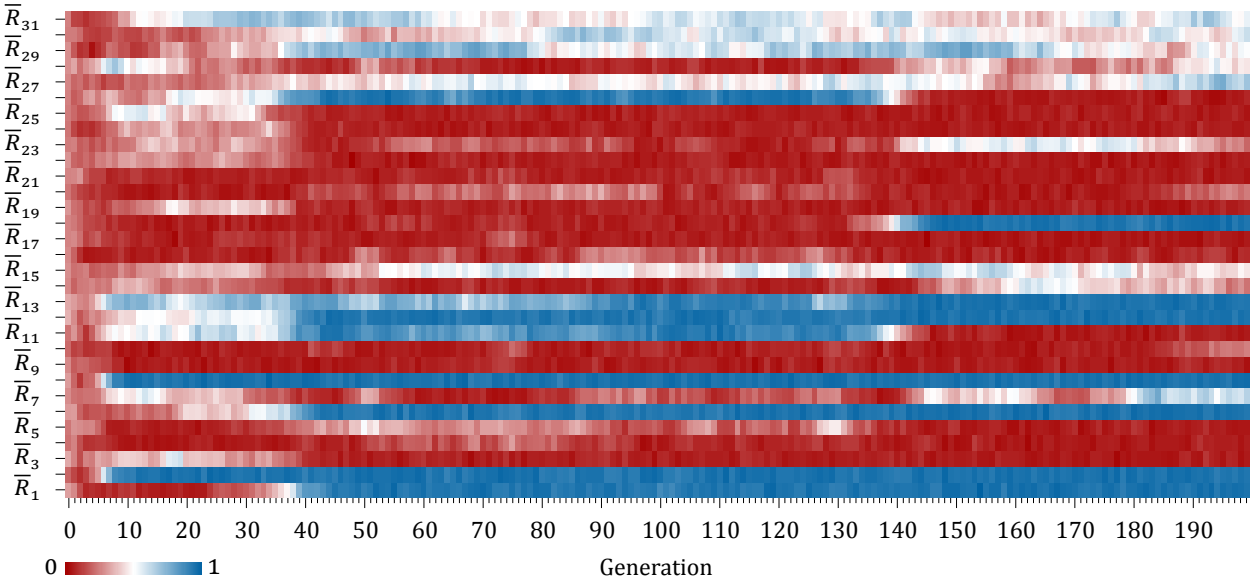


(b) Start mit $0/32 \leq \lambda \leq 31/32$, 40 Läufe

Abb. 6: Verlauf des Histograms der λ-Werte für jede Generation.



(a) Durchschnittliche Fitness der jeweils 20 besten Regeln jeder Generation.



(b) Verlauf der Durchschnittswerte $\overline{R}_i$ jeder Position über alle Regeln jeder Generation

Abb. 7: Fitness und Profil eines Durchlaufs. Starke Änderungen des Profils korellieren mit Sprüngen in der Fitness.

gelegentlich das Plateau als Folge eines Zufallsereignisses. In der Profilansicht Abb. 7(b) des gleichen Durchlaufs ist zu erkennen, dass die meisten Bits der Population auf einem Plateau konstant bleiben und nur beim Wechsel umgeworfen werden, z.B. um die Generationen 37 und 138, jeweils verbunden mit Sprüngen in der Durchschnittsfitness.

5.3 Long Tail

Abb. 8 zeigt, dass die Fitness einer Regel einigermaßen unabhängig von den Eingaben x^0 ist: die durchschnittliche, mit einer Eingabe erreichte Fitness über alle Regeln ist für alle Eingaben etwa gleich, es hängt von der Regel ab, welche

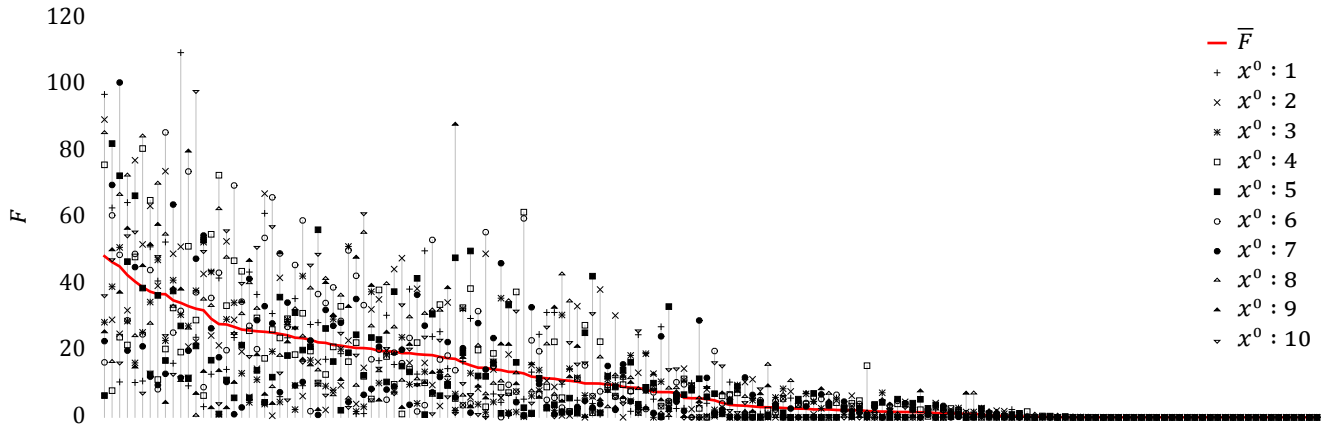


Abb. 8: Fitnesswerte einer Generation. Die für jede der m Eingaben x^0 berechneten F sind durch unterschiedliche Symbole markiert.

Eingabe zu hoher oder extrem niedriger Fitness führt, insbesondere ist keine Eingabe zu beobachten, die durchgehend hohe Fitness erzeugt. Somit ist auch die geringe Zahl von $m = 10$ Eingaben zu rechtfertigen, mit $m = 1000$ ist z.B. keine Verbesserung zu beobachten, im Verhältnis zur Gesamtzahl 2^W aller möglichen Eingaben ist dies ohnehin nicht signifikant.

Die Suche von einem genetischen Algorithmus steuern zu lassen ist allerdings relativ effizient: in 20 Durchläufen über 200 Generationen mit einer Population von 160 Regeln, insgesamt also 640 000 Bewertungen, wurden nur 17.2 % der erzeugten Regeln mehr als einmal betrachtet, insgesamt waren 37.3 % der Bewertungen redundant, weil mehrmals auftretende Regeln in jeder Generation neu bewertet wurden. Dies ist effizienter als die Anzahl m der Eingaben so weit zu erhöhen, dass Fitness-Werte unterschiedlicher Generationen vergleichbar werden.

Ein Großteil der in jeder Generation bewerteten Regeln bekommt extrem niedrige Fitnesswerte, üblicherweise mindestens die Hälfte (die Abbildung zeigt eine Ausnahmesituation, hier ist es lediglich ein Viertel). Ein Weg, die Suchzeit zu verringern wäre, nicht alle m Eingaben auszuwerten, sondern abzubrechen, falls die ersten m' Fitnesswerte nicht über einem bestimmten Schwellenwert liegen.

5.4 Dichte der Eingabe

Im Paper werden nur Eingaben mit $p(0) = p(1) = 0.5$ betrachtet. Abb. 9 zeigt die Spektralen Leistungsdichten für Regel 110 und die beste im Paper gefundene Regel für verschiedene Dichten $p(1)$ mit $p(0) = 1 - p(1)$. Im Gegensatz zu Regel 110 kann die beste Regel keine $1/f$-Noise mehr erzeugen, wenn weniger als 46 Zellen der $W = 700$-Eingabe 1 sind. Es könnte interessant sein, ähnlich

der verschiedenen λ auch unterschiedliche, und insbesondere extreme, Dichten zu evaluieren, z.B. 1 auf unendlichem 0-Hintergrund.

5.5 Fitnessfunktion

Für $\sigma^2 < 0.2$ liefert die Fitnessfunktion keine sinnvolle Bewertung der Regel mehr. So zeigt Abb. 10 vier sehr ähnliche $1/f$-Spektren, deren Fitness sich allerdings um eine Größenordnung unterscheidet. Statt, wie beabsichtigt von β hängt der Wert nur noch von σ^2 ab, der Algorithmus ist nicht mehr in der Lage, für β zu optimieren.

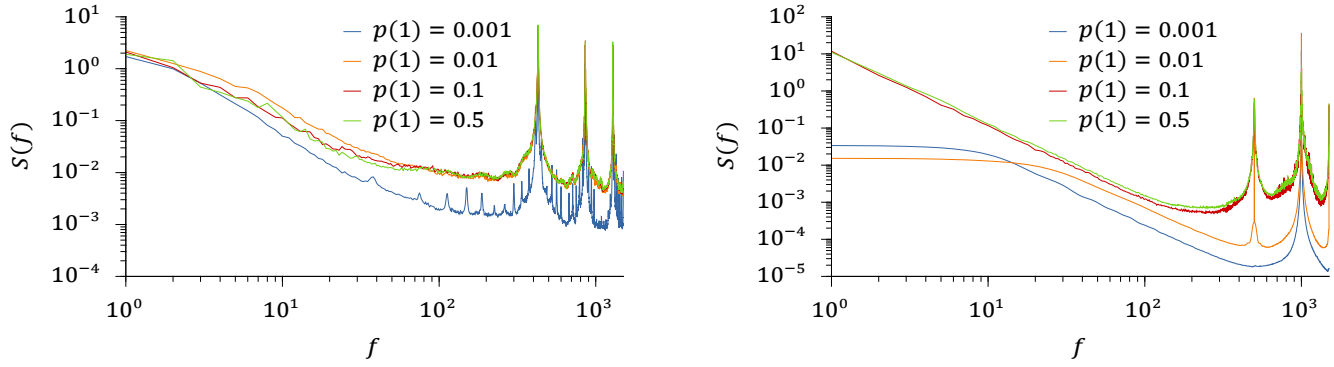
In Abb. 5 ist exemplarisch auch ein Lauf mit $f_b = f_r = 400$ dargestellt, der Regeln mit $3/32 \leq \lambda \leq 7/32$ ergibt, die beste mit einer Fitness von $F = 448.87$ ist 00000100000010000000100000011000 — eine Klasse 1-Regel mit flachem Spektrum. Das im Paper festgelegte $\delta = 10^{-6}$ sollte statt vor Division durch Null zu schützen, vielleicht besser verwendet werden, ein sinnvolles Minimum für σ^2 zu setzen, z.B. indem $\delta = 0.2$ gesetzt wird.

Wie in Abb. 11 zu erkennen ist, sind die Fitness-Werte logarithmisch verteilt, aber als linear verteilt verwendet, was Ausreißern nach oben überproportional viel Gewicht gibt und das System destabilisiert. Entsprechend fallen Regeln, die nicht mehr Teil der sowieso kopierten „Elite“ sind, zu schnell ab und haben wenig Chance auf Nachkommen.

5.6 Suchstrategie

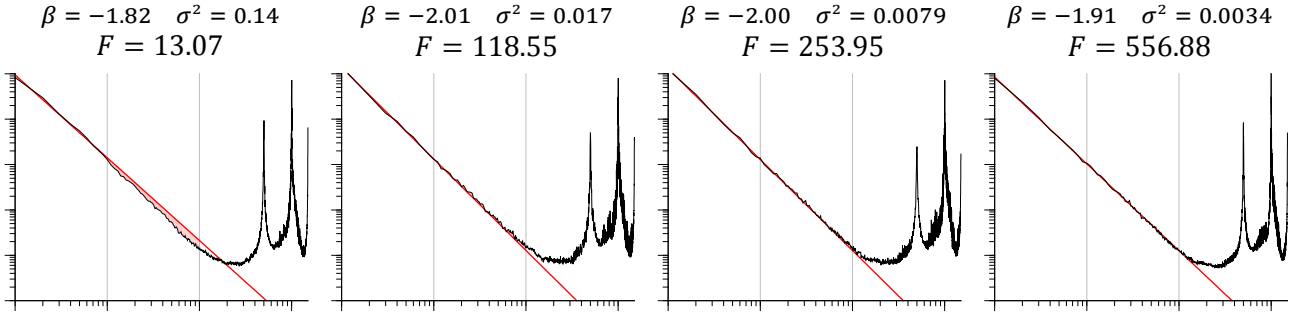
Meine Implementierung[1] des Papers verwendet die sehr effiziente Fourier-Transformation aus FFTw [2]; tatsächlich wird etwa 40 % der Laufzeit mit der Ausführung der Automaten verbracht, der Rest mit der Berechnung der DFT. Auf vier AMD Opteron 2216 Kernen (2.4 GHz, 1 MiB Cache, SSE2) kann die gemittelte Fitness über zehn Läufe von 160 Regeln in etwa 27 s berechnet werden, d.h. ca. 0.7 s pro Regel und Kern. Den kompletten Raum zu evaluieren würde also 2.9×10^9 s, ca. 90 CPU-Jahre, benötigen. Da die Regeln komplett unabhängig voneinander sind ist das Problem *embarassingly parallel*, d.h. über genügend Prozessoren verteilt durchaus in einem realistischen Zeitraum zu lösen, insbesondere da das Ausführen der Automaten und Berechnen des Spektrums ebenfalls gut parallelisierbar sind (so können etwa auf GPUs alle Zellen einer Zeile gleichzeitig und unabhängig voneinander berechnet werden, sowie diverse Phasen der FFT, besonders wenn darauf geachtet wird, T in Zweierpotenzen zu wählen und W in vielfachen der Registerbreite)

1 siehe `http://git.io/psal-noise`



(a) Regel 110 (b) Beste gefundene Regel

Abb. 9: Spektren, gemittelt über 6 Eingaben bestimmter Dichte.



(a) (b) (c) (d)

Abb. 10: Spektrum und daraus berechnete Fitnesswerte der besten gefundenen Regel für vier verschiedene Eingaben.

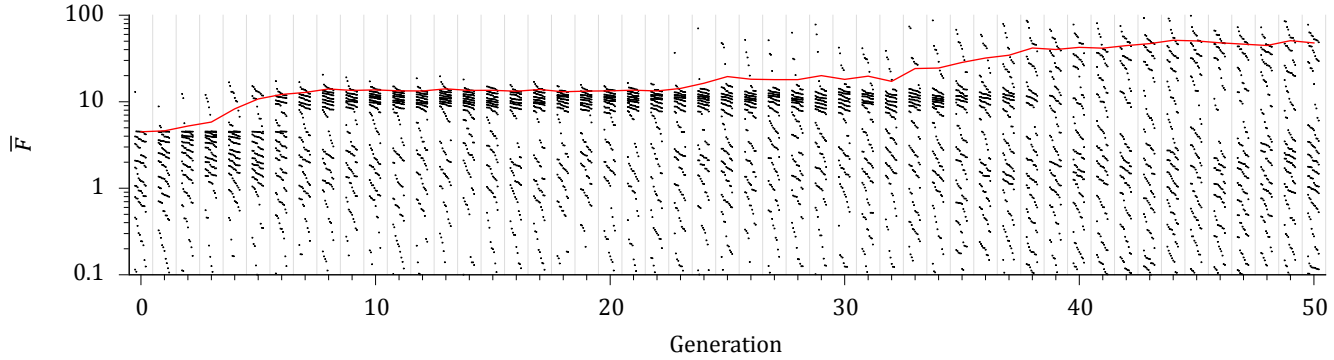


Abb. 11: Fitnesswerte, logarithmisch

Literatur

Alle Onlinequellen Stand 17. September 2012.

1 Matthew Cook. „Universality in Elementary Cellular Automata“. In: *Complex Systems* 15.1 (2004), S. 1–40.
URL: http://www.complex-systems.com/abstracts/v15_i01_a01.html.

2 Matteo Frigo und Steven G. Johnson. „The Design and Implementation of FFTW3“. In: *Proceedings of the IEEE* 93.2 (2005), S. 216–231.
DOI: 10.1109/JPROC.2004.840301. URL: http://fftw.org/fftw-paper-ieee.pdf.

3 Lawrence Gray. „A mathematician looks at Wolfram's new kind of science“. In: *Notices of the American Mathematical Society* 50.2 (2003), S. 200–211.
URL: http://www.ams.org/notices/200302/fea-gray.pdf.

4 Marvin S. Keshner. „$1/f$ Noise“. In: *Proceedings of the IEEE* 70.3 (1982), S. 212–218. DOI: 10.1109/PROC.1982.12282.

5 Chris G. Langton. „Computation at the edge of chaos: phase transitions and emergent computation“. In: *Physica D* 42.1 (1990), S. 12–37.
DOI: 10.1016/0167-2789(90)90064-V.

6 Weitan Li und Norman Packard. „The structure of the Elementary Cellular Automata rule space“. In: *Complex Systems* 4.3 (1990), S. 281–297.
URL: http://www.complex-systems.com/abstracts/v04_i03_a03.html.

7 Shigeru Ninagawa. „Evolution of one-dimensional Cellular Automata by $1/f$ noise“. In: *Proceedings of the 9th European conference on Advances in artificial life*. Lissabon, Portugal: Springer-Verlag, 2007, S. 905–914.
DOI: 10.1007/978-3-540-74913-4.

8 Shigeru Ninagawa. „Power Spectral Analysis of Elementary Cellular Automata“. In: *Complex Systems* 17.4 (2008), S. 399–411.
URL: http://www.complex-systems.com/abstracts/v17_i04_a05.html.

9 *Least Squares Fitting*. In: *MathWorld*. Hrsg. von Eric W. Weisstein.
URL: http://mathworld.wolfram.com/LeastSquaresFitting.html.

10 Stephen Wolfram. „Statistical mechanics of cellular automata“. In: *Reviews of Modern Physics* 55.3 (1983), S. 601–644. DOI: 10.1103/RevModPhys.55.601.

11 Stephen Wolfram. „Universality and complexity in cellular automata“. In: *Physica D* 10.1 (1984), S. 1–35. DOI: 10.1016/0167-2789(84)90245-8.