

4 C-rekursive Folgen – Anwendungen

4.1 Zwei Beispiele

4.1.1 Nochmals *Frisbee*

Jetzt wird Frisbee so gespielt, dass Das Spiel nicht beendet wird, sobald die beiden Frisbee-Scheiben bei einem Spieler zusammentreffen. Tritt dieser Fall ein, soll jede der Scheiben mit Wahrscheinlichkeit $1/2$ zu rechten oder linken Mitspieler weiterfliegen In zwei von vier möglichen Fällen landen beide Scheiben bei dem gleichen Mitspieler. In den anderen beiden Fällen haben die beiden Scheiben dann den Abstand 2. Der folgende Transitionsgraph beschreibt die Situation:

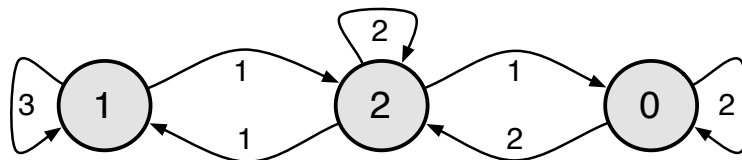


Abbildung 5: Modifiziertes Frisbee-Spiel

Als Transitionsmatrix geschrieben:

$$A = \begin{bmatrix} 3 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 2 & 2 \end{bmatrix}$$

Potenzen dieser Matrix zählen wieder die möglichen Spielverläufe auf:

$$A^2 = \begin{bmatrix} 10 & 5 & 1 \\ 5 & 7 & 4 \\ 2 & 8 & 6 \end{bmatrix} \quad A^3 = \begin{bmatrix} 35 & 22 & 7 \\ 22 & 27 & 15 \\ 14 & 30 & 20 \end{bmatrix} \quad A^4 = \begin{bmatrix} 127 & 93 & 36 \\ 93 & 106 & 57 \\ 72 & 114 & 70 \end{bmatrix}$$

$$A^5 = \begin{bmatrix} 474 & 385 & 165 \\ 385 & 419 & 220 \\ 330 & 440 & 254 \end{bmatrix} \quad A^6 = \begin{bmatrix} 1807 & 1574 & 715 \\ 1574 & 1663 & 859 \\ 1430 & 1718 & 948 \end{bmatrix} \quad A^7 = \begin{bmatrix} 6995 & 6385 & 3004 \\ 6385 & 6618 & 3381 \\ 6008 & 6762 & 3614 \end{bmatrix}$$

$$A^8 = \begin{bmatrix} 27370 & 25773 & 12393 \\ 25773 & 26383 & 13380 \\ 24786 & 26760 & 13990 \end{bmatrix} \quad A^9 = \begin{bmatrix} 107883 & 103702 & 50559 \\ 103702 & 105299 & 53143 \\ 101118 & 106286 & 54740 \end{bmatrix}$$

Für das Wachstum der Koeffizienten sind die Eigenwerte der Matrix zuständig:

$$\chi_A(z) = z^3 - 7z^2 + 13z - 4$$

hat die Nullstellen

$$\lambda_1 = 4, \lambda_2 = \frac{3 + \sqrt{5}}{2} = 2.618033988\dots, \lambda_3 = \frac{3 - \sqrt{5}}{2} = 0.381966012\dots$$

Interessanter wird die Angelegenheit, wenn man an stelle der Matrix A die Matrix

$$B = \frac{1}{4} A = \begin{bmatrix} 3/4 & 1/4 & 0 \\ 1/4 & 1/2 & 1/4 \\ 0 & 1/2 & 1/2 \end{bmatrix}$$

betrachtet. Das ist eine *stochastische Matrix*, also eine Matrix mit nichtnegativen Koeffizienten, bei der die Zeilensumme sämtlich gleich 1 sind. Mehr dazu im Verlauf dieses Kapitels. Die Matrix A beschreibt das Frisbee-Spiel aus stochastischer Sicht: aus den Anzahlen von Pfaden werden Wahrscheinlichkeiten von Pfaden.

Auch hier kann man die Potenzen der Matrix bilden:

$$B^2 = \begin{bmatrix} 5/8 & \frac{5}{16} & 1/16 \\ \frac{5}{16} & \frac{7}{16} & 1/4 \\ 1/8 & 1/2 & 3/8 \end{bmatrix} \quad B^3 = \begin{bmatrix} \frac{35}{64} & \frac{11}{32} & \frac{7}{64} \\ \frac{11}{32} & \frac{27}{64} & \frac{15}{64} \\ \frac{7}{32} & \frac{15}{32} & \frac{5}{16} \end{bmatrix} \quad B^4 = \begin{bmatrix} \frac{127}{256} & \frac{93}{256} & \frac{9}{64} \\ \frac{93}{256} & \frac{53}{128} & \frac{57}{256} \\ \frac{9}{32} & \frac{57}{128} & \frac{35}{128} \end{bmatrix}$$

$$B^5 = \begin{bmatrix} 0.4628906250 & 0.3759765625 & 0.1611328125 \\ 0.3759765625 & 0.4091796875 & 0.2148437500 \\ 0.3222656250 & 0.4296875000 & 0.2480468750 \end{bmatrix}$$

$$B^{10} = \begin{bmatrix} 0.4075536728 & 0.3971147537 & 0.1953315735 \\ 0.3971147537 & 0.4011020660 & 0.2017831802 \\ 0.3906631470 & 0.4035663605 & 0.2057704926 \end{bmatrix}$$

$$B^{20} = \begin{bmatrix} 0.4001089710 & 0.3999583768 & 0.1999326522 \\ 0.3999583768 & 0.4000158987 & 0.2000257246 \\ 0.3998653044 & 0.4000514491 & 0.2000832465 \end{bmatrix}$$

$$B^{50} = \begin{bmatrix} 0.4000000003 & 0.3999999999 & 0.1999999998 \\ 0.3999999999 & 0.4000000000 & 0.2000000001 \\ 0.3999999996 & 0.4000000002 & 0.2000000002 \end{bmatrix}$$

Es ist nicht zu übersehen, dass die Potenzen B^n gegen die Matrix

$$\lim_{n \rightarrow \infty} B^n = \begin{bmatrix} 0.4 & 0.4 & 0.2 \\ 0.4 & 0.4 & 0.2 \\ 0.4 & 0.4 & 0.2 \end{bmatrix}$$

konvergieren, bei der die Zeilen identisch sind. Aber warum ist das so und was zeichnet die Zeilen dieser Matrix aus?

4.1.2 *Drun kard's Walk*: Zufallspfade in einem Graphen

Es sei $G = (V, E)$ ein ungerichteter Graph. Ohne Einschränkung sei $V = \{1, 2, \dots, k\}$ – im Beispiel dieses Abschnitts ist $k = 8$. Für jeden der Knoten $v \in V$ sei d_v die Anzahl der Kanten, die mit v inzidieren, also der *Grad* des Knotens v .

Ein Betrunkener befindet sich in einem der Knoten “ v ”. Er wählt zufällig und gleichverteilt eine der Kanten aus, die mit v inzidieren, und benutzt sie, um zu dem Knoten am anderen Ende zu gelangen. Dort verhält er sich – unabhängig davon, von woher er gekommen ist – genauso, und immer so weiter \dots : er durchstreift also den Graphen auf einem *Zufallspfad* (*random walk*). Die Frage ist dann: wo wird man den Betrunkenen langfristig mit welcher Häufigkeit antreffen?

Hier ein Beispiel:

Die Adjazenzmatrix des Graphen ist also

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Befindet sich der Betrunkene beispielsweise im Knoten “3”, so wird er mit Wahrscheinlichkeit von jeweils $1/3$ zu einem der Knoten “2” oder “4” oder “6”, wechselt. Vom Knoten “2” aus wird er mit Wahrscheinlichkeit von jeweils $1/5$ zu einem der

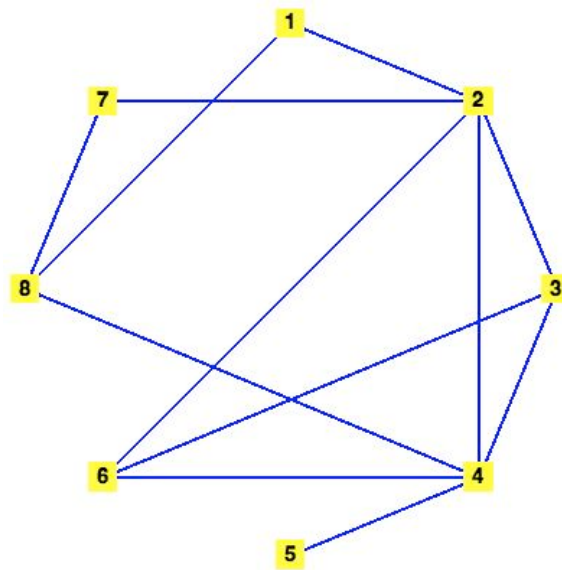


Abbildung 6: Beispielgraph

Knoten "1". "3", "4", "6", "7" taumeln. Wenn er in "5" ist, wird er sich mit Sicherheit zu "4" bewegen.

Die Wahrscheinlichkeiten für längere Wege ergeben sich Produktbildung (Unabhängigkeit!): befindet sich der Betrunkene in "4" so hat der Pfad

$$4 \longrightarrow 2 \longrightarrow 3 \longrightarrow 2 \longrightarrow 1 \longrightarrow 8$$

für ihn die Wahrscheinlichkeit

$$\frac{1}{5} \cdot \frac{1}{5} \cdot \frac{1}{3} \cdot \frac{1}{5} \cdot \frac{1}{2} = \frac{1}{750}$$

Um sein langfristigen Verhalten zu studieren, empfiehlt es sich, in der Matrix A alle Elemente durch die jeweilige Zeilensumme zu dividieren.

$$B = \begin{bmatrix} 0 & 1/2 & 0 & 0 & 0 & 0 & 0 & 1/2 \\ 1/5 & 0 & 1/5 & 1/5 & 0 & 1/5 & 1/5 & 0 \\ 0 & 1/3 & 0 & 1/3 & 0 & 1/3 & 0 & 0 \\ 0 & 1/5 & 1/5 & 0 & 1/5 & 1/5 & 0 & 1/5 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1/3 & 1/3 & 1/3 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 & 0 & 0 & 0 & 1/2 \\ 1/3 & 0 & 0 & 1/3 & 0 & 0 & 1/3 & 0 \end{bmatrix}$$

Induktiv sieht man leicht ein, dass in den n -ten Potenzen der Matrix B Information über die Wahrscheinlichkeiten aller Wege der Länge n von einem Knoten v zu einem Knoten w enthalten ist. Beispielsweise ist

$$B^5 = \begin{bmatrix} \frac{7}{375} & \frac{5446}{16875} & \frac{3817}{33750} & \frac{377}{3375} & \frac{332}{5625} & \frac{3817}{33750} & \frac{7}{375} & \frac{1367}{5625} \\ \frac{10892}{84375} & \frac{6916}{50625} & \frac{32536}{253125} & \frac{66101}{253125} & \frac{494}{16875} & \frac{32536}{253125} & \frac{10892}{84375} & \frac{974}{16875} \\ \frac{3817}{50625} & \frac{32536}{151875} & \frac{6334}{50625} & \frac{33271}{151875} & \frac{2072}{50625} & \frac{19627}{151875} & \frac{3817}{50625} & \frac{6107}{50625} \\ \frac{754}{16875} & \frac{66101}{253125} & \frac{33271}{253125} & \frac{7396}{50625} & \frac{5167}{84375} & \frac{33271}{253125} & \frac{754}{16875} & \frac{15127}{84375} \\ \frac{664}{5625} & \frac{494}{3375} & \frac{2072}{16875} & \frac{5167}{16875} & \frac{22}{1125} & \frac{2072}{16875} & \frac{664}{5625} & \frac{52}{1125} \\ \frac{3817}{50625} & \frac{32536}{151875} & \frac{19627}{151875} & \frac{33271}{151875} & \frac{2072}{50625} & \frac{6334}{50625} & \frac{3817}{50625} & \frac{6107}{50625} \\ \frac{7}{375} & \frac{5446}{16875} & \frac{3817}{33750} & \frac{377}{3375} & \frac{332}{5625} & \frac{3817}{33750} & \frac{7}{375} & \frac{1367}{5625} \\ \frac{2734}{16875} & \frac{974}{10125} & \frac{6107}{50625} & \frac{15127}{50625} & \frac{52}{3375} & \frac{6107}{50625} & \frac{2734}{16875} & \frac{82}{3375} \end{bmatrix}$$

oder approximiert auf 5 Dezimalstellen:

$$B^5 \approx \begin{bmatrix} 0.018667 & 0.32273 & 0.11310 & 0.11170 & 0.059022 & 0.11310 & 0.018667 & 0.24302 \\ 0.12909 & 0.13661 & 0.12854 & 0.26114 & 0.029274 & 0.12854 & 0.12909 & 0.057719 \\ 0.075398 & 0.21423 & 0.12512 & 0.21907 & 0.040928 & 0.12923 & 0.075398 & 0.12063 \\ 0.044681 & 0.26114 & 0.13144 & 0.14609 & 0.061239 & 0.13144 & 0.044681 & 0.17928 \\ 0.11804 & 0.14637 & 0.12279 & 0.30619 & 0.019556 & 0.12279 & 0.11804 & 0.046222 \\ 0.075398 & 0.21423 & 0.12923 & 0.21907 & 0.040928 & 0.12512 & 0.075398 & 0.12063 \\ 0.018667 & 0.32273 & 0.11310 & 0.11170 & 0.059022 & 0.11310 & 0.018667 & 0.24302 \\ 0.16201 & 0.096198 & 0.12063 & 0.29880 & 0.015407 & 0.12063 & 0.16201 & 0.024296 \end{bmatrix}$$

Somit ist $\frac{15127}{84375} \approx 0.17928$ die Wahrscheinlichkeit dafür, dass der Betrunkene in fünf “Schritten” vom Knoten “4” zum Knoten “8” kommt.

Schaut man sich nun

$$B^{30} \approx \begin{bmatrix} 0.083944 & 0.20738 & 0.12502 & 0.20917 & 0.041464 & 0.12502 & 0.083944 & 0.12406 \\ 0.082950 & 0.20893 & 0.12499 & 0.20781 & 0.041794 & 0.12499 & 0.082950 & 0.12559 \\ 0.083347 & 0.20831 & 0.12500 & 0.20835 & 0.041662 & 0.12500 & 0.083347 & 0.12498 \\ 0.083669 & 0.20781 & 0.12501 & 0.20880 & 0.041555 & 0.12501 & 0.083669 & 0.12448 \\ 0.082927 & 0.20897 & 0.12499 & 0.20777 & 0.041802 & 0.12499 & 0.082927 & 0.12563 \\ 0.083347 & 0.20831 & 0.12500 & 0.20835 & 0.041662 & 0.12500 & 0.083347 & 0.12498 \\ 0.083944 & 0.20738 & 0.12502 & 0.20917 & 0.041464 & 0.12502 & 0.083944 & 0.12406 \\ 0.082706 & 0.20932 & 0.12498 & 0.20747 & 0.041875 & 0.12498 & 0.082706 & 0.12597 \end{bmatrix}$$

an, so erkennt man, dass sich die Zeilen der Matrix kaum noch unterscheiden. Das bedeutet: die Wahrscheinlichkeit, dass sich der Betrunkene nach 30 Takten (oder generell: nach vielen Takten) in einem gewissen Zustand v befindet, hängt praktisch überhaupt nicht mehr davon an, wo er gestartet ist. Ausserdem erkennt man durch Vergleich mit anderen Potenzen von B , dass man es offensichtlich mit einem Konvergenzphänomen zu tun hat. Die Potenzen B^n konvergieren scheinbar gegen eine Matrix, deren Zeilen alle identisch sind, und zwar drängt sich dafür der Wert

$$\left[0.083333 \quad 0.20834 \quad 0.12500 \quad 0.20834 \quad 0.041666 \quad 0.12500 \quad 0.083333 \quad 0.12500 \right]$$

auf, und das könnte

$$\left[\frac{1}{12} \quad \frac{5}{24} \quad \frac{1}{8} \quad \frac{5}{24} \quad \frac{1}{24} \quad \frac{1}{8} \quad \frac{1}{12} \quad \frac{1}{8} \right] = \left[\frac{2}{24} \quad \frac{5}{24} \quad \frac{3}{24} \quad \frac{5}{24} \quad \frac{1}{24} \quad \frac{3}{24} \quad \frac{2}{24} \quad \frac{3}{24} \right]$$

sein. Aber warum ist das so?

4.2 Graphen und reguläre Sprachen

Definition 6. 1. Ein (*endlicher, gerichteter*) *Graph* $G = (V, E)$ besteht aus einer endlichen Knotenmenge V und einer endlichen Kantenmenge E , sowie zwei Abbildungen $h : E \rightarrow V$ (genannt *head*), $t : E \rightarrow V$ (genannt *tail*).

Für $e \in E$ besagt $h(e) = u, t(e) = v$, dass die Kante e von u nach v geht. Ist dabei $u = v$, so ist die Kante eine *Schleife* (ein *loop*). Es kann also mehrere Kanten zwischen zwei Knoten geben, ebenso mehrere Schleifen an einem Knoten. Kanten sind durch ihre “Namen” $e \in E$ unterschieden. E ist also als eine Menge von Symbolen, d.h. als ein Alphabet zu betrachten.

2. Ein *Pfad der Länge n von u nach v in G* ist eine Folge von (nicht notwendig verschiedenen) Kanten $\pi = (e_1 e_2, \dots, e_n) = e_1 e_2 \dots e_n \in E^n$ mit der Eigenschaft

$$t(e_1) = u, \quad h(e_i) = t(e_{i+1}) \quad (1 \leq i < n), \quad h(e_n) = v.$$

Die Funktionen h und t lassen sich auch für Pfade positiver Länge $|\pi| = n$ definieren:

$$t(\pi) = t(e_1) = u, \quad h(\pi) = h(e_n) = v.$$

Die Schreibweise $\pi = e_1 e_2 \dots e_n$ soll suggerieren, dass der Pfad π als ein Wort über dem Alphabet E aufgefasst wird.

3. Ein geschlossener Pfad π , d.h. $h(\pi) = t(\pi)$, ist ein *Zyklus* oder *Kreis*.
4. Für $u, v \in V$ und $n \in \mathbb{N}$ sei
 $[u, v]_G = \{\pi; t(\pi) = u, h(\pi) = v\} \subseteq E^*$ die Menge (formale Sprache) der Pfade von u nach v in G ,
 $[u, v]_G^n = \{\pi \in [u, v]_G; |\pi| = n\} = [u, v]_G \cap E^n$ die Menge der Pfade der Länge n von u nach v in G .

Bemerkung 7. Beachte die spezielle Situation für $k = 0$ und $k = 1$:

$$[u, v]_G^0 = \begin{cases} \emptyset & \text{falls } u \neq v \\ \{\varepsilon\} & \text{falls } u = v \end{cases}$$

$$[u, v]_G^1 = \{e \in E; t(e) = u, h(e) = v\}$$

ε ist das leere Wort, die Symbole $e \in E$ sind die Wörter der Länge 1.

Lemma 17.

$$[u, v]_G^{n+1} = \bigcup_{w \in V} [u, w]_G^n \times [w, v]_G^1$$

Theorem 18 (Kleene).

Ist $G = (V, E)$ ein Graph, so sind für $u, v \in V$ die "Pfadssprachen" $[u, v]_G \subseteq E^*$ rationale Sprachen.

Kommentar 8. Zur terminologischen Unterscheidung:

- Eine formale Sprache über einem Alphabet E heisst *regulär*, wenn sie durch einen endlichen (deterministischen oder nichtdeterministischen Automaten mit Alphabet E akzeptiert wird, bzw. durch eine Typ-3-Grammatik generiert wird.

- Eine formale Sprache über einem Alphabet E heisst *rational*, wenn sie aus der leeren Sprache \emptyset und den einelementigen Sprachen $\{\varepsilon\}$ und $\{e\}$ (für $e \in E$) mittels der *rationalen Operationen* Vereinigung, Konkatenation und Iteration (=Kleene-*) erzeugt werden kann – äquivalent: durch einen “regulären Ausdruck” beschrieben wird.

Beweis (KLEENES Theorem). Am einfachsten geht das per Induktion über die Anzahl der Kanten $\#E$.

- Ist $\#E = \emptyset$, so ist

$$[u, v]_G = \begin{cases} \emptyset & \text{falls } u \neq v \\ \{\varepsilon\} & \text{falls } u = v \end{cases}$$

- Sei die Aussage des Theorems schon bewiesen für einen Graphen $G = (V, E)$ und sei $H = (V, F)$ ein Graph mit $F = E \cup \{f\}$, wobei f eine neue Kante ist mit $t(f) = x, h(f) = y$, mit $x, y \in V$. Dann gilt für alle $u, v \in V$:

$$[u, v]_H = [u, v]_G \cup [u, x]_G \cdot (f \cdot [y, x]_G)^* \cdot f \cdot [y, v]_G,$$

d.h. $[u, v]_H$ entsteht aus schon als rational bekannten Sprachen durch rationale Operationen. \square

Kommentar 9. Aus dem eben bewiesenen allgemeinen Theorem folgt durch Anwendung auf endliche Automaten insbesondere, dass jede *reguläre* Sprache auch eine *rationale* Sprache ist.

Die Umkehrung dieser Aussage, dass nämlich jede *rationale* Sprache auch eine *reguläre* Sprache ist, ist auch richtig und durch direkte Automatenkonstruktionen für den Abschluss unter rationalen Operationen zu beweisen. Zusammen ergibt sich:

Theorem 19 (Hauptsatz der Theorie der endlichen Automaten).

$$\boxed{\text{rationale Sprachen} \equiv \text{reguläre Sprachen}}$$

Dieser Satz hat sehr weitreichende Konsequenzen (auch praktische!), so z.B. den Abschluss der rationalen Sprachen unter booleschen Operationen und das Verhalten von regulären Sprachen unter diversen Transformationen, was auf Grund der jeweiligen *Definitionen* überhaupt nicht klar ist.

Fundamentale Probleme in diesem Kontext sind die Fragen nach den Kardinalitäten der endlichen Sprachen $[u, v]_G^n$ in Abhängigkeit von n (für fest gewählten Graphen $G = (V, E)$ und beliebig gewählte Knoten $u, v \in V$).

- Kann man $\# [u, v]_G^n$ *effizient* berechnen?
“Ausprobieren” ist mit Sicherheit *kein effizienter* Weg!
- Was kann man über das Verhalten von $\# [u, v]_G^n$ bei wachsendem n aussagen?
Oft ist man weniger an den exakten Zahlen, als an deren Wachstumsverhalten interessiert.

Der Bedeutung wegen für reguläre Sprachen nochmals separat formuliert:

Es sei L eine reguläre Sprache über einem Alphabet Σ , gegeben durch einen akzeptierenden Automaten, eine generierende Grammatik oder einen regulären Ausdruck. Für $n \in \mathbb{N}$ sei $\ell_n = \#(L \cap \Sigma^n)$ die Anzahl der Wörter der Länge n von L .

- Kann man die $\ell_n = \#(L \cap \Sigma^n)$ *effizient* berechnen?
- Was kann man über das Verhalten der $\ell_n = \#(L \cap \Sigma^n)$ bei wachsendem n aussagen?

4.3 Graphen und Matrizen

Ist $G = (V, E)$ ein Graph mit $\#V = k$, so kann man o.E. annehmen, dass die Knoten des Graphen G von 1 bis k durchnummeriert sind, also $V = \{1, 2, \dots, k\}$. Die Zahlen

$$a_{i,j} = \# [u, v]_G^1 = \#\{e \in E; t(e) = i, h(e) = j\}$$

kann man also als die Koeffizienten einer $(k \times k)$ -Matrix $A_G = [a_{i,j}]_{1 \leq i, j \leq k}$ auffassen, der sog. *Adjazenzmatrix* der Graphen G .

In dieser Notation sind die Zahlen $a_{i,j}^{(n)} = \# [i, j]_G^n$ die Koeffizienten der n -ten Matrixpotenz $(A_G)^n$, d.h. es gilt der

Satz 20.

$$(A_G)^n = \left[a_{i,j}^{(n)} \right]_{1 \leq i, j \leq k}$$

Beweis. Das ergibt sich ganz einfach per Induktion über n .

- Für $n = 0$ und $n = 1$ ist das offensichtlich richtig.
- Ist die Beziehung bis zu einem n bereits gezeigt, so übersetzt sich die Matrixgleichung

$$(A_G)^{n+1} = (A_G)^n \cdot A_G$$

dahingehend, dass an der Position (i, j) dieser Matrix

$$\sum_{1 \leq \ell \leq k} \# [i, \ell]_G^n \cdot \# [\ell, j]_G^1$$

steht. Das ist aber nichts anders als $\# [i, j]_G^{n+1}$, denn

$$\# [i, j]_G^{n+1} = \sum_{1 \leq \ell \leq k} \# [i, \ell]_G^n \cdot \# [\ell, j]_G^1$$

ist die quantitative Version der Sprach-Beziehung

$$[i, j]_G^{n+1} = \bigcup_{1 \leq \ell \leq k} [i, \ell]_G^n \times [\ell, j]_G^1$$

Definition 7. Es sei nun $A = [a_{i,j}]_{1 \leq i, j \leq k}$ eine *nichtnegative* $(k \times k)$ -Matrix, d.h. alle $a_{i,j}$ sind reelle Zahlen ≥ 0 . Der Graph $G_A = (V, E)$, der zu dieser Matrix gehört, ist definiert durch

$$V = \{1, 2, \dots, k\}, \quad \forall (i, j) \in V \times V : (i, j) \in E \Leftrightarrow a_{i,j} > 0$$

$a_{i,j}$ ist das *Gewicht* der Kante (i, j) .

Das Gewicht eines Pfades

$$\pi : i = i_0 \rightarrow i_1 \rightarrow i_2 \rightarrow \dots \rightarrow i_n = j$$

ist dann das Produkt der Kantengewichte

$$a(\pi) = \prod_{0 \leq \ell < n} a_{i_\ell, i_{\ell+1}}$$

Definiert man nun

$$a_{i,j}^{(n)} = \sum \{ a(\pi) ; \pi \in [i, j]_{G_A}^n \}$$

d.h. ist $a_{i,j}^{(n)}$ die Summe alle Pfadgewichte für Pfade der Länge n von i nach j in G_A .

Bemerkung 10. Es gilt in völliger Analogie zum Vorherigen:

$$A^n = \left[a_{i,j}^{(n)} \right]_{1 \leq i, j \leq k}$$

und insbesondere

$$a_{i,j}^{(n)} > 0 \Leftrightarrow \text{es gibt in } G_A \text{ einen Pfad der Länge } n \text{ von } i \text{ nach } j$$

Für reguläre (Typ-3)-Sprachen ergibt sich aus der Matrixbeschreibung zusammen mit den Aussagen des vorigen Kapitels:

Folgerung 21. *Ist $L \subseteq \Sigma^*$ eine reguläre Sprache, bezeichnet $\ell_n = \#(L \cap \Sigma^n)$ die Anzahl der Wörter der Länge n dieser Sprache, so ist die Folge $(\ell_n)_{n \geq 0}$ eine C-rekursive Folge, d.h. es gibt ein $k > 0$ und ganze Zahlen a_1, a_2, \dots, a_k mit*

$$\ell_n = a_1 \ell_{n-1} + a_2 \ell_{n-2} + \dots + a_k \ell_{n-k} \quad (n \geq k).$$

Die Koeffizienten a_1, a_2, \dots, a_k kann mit Hilfe eines endlichen Automaten (als Transitionsgraph) bestimmen, der die Sprache L akzeptiert: man muss das charakteristische Polynom der Adjazenzmatrix dieser Graphen bestimmen. (Nebenbei: es gibt auch noch andere Wege...)

Man beobachtet hier eine Art *Pumplemma*: Ist nämlich $(\ell_n)_{n \geq 0}$ eine C-rekursive Folge der Ordnung k und sind für ein $n \geq k$ die k aufeinanderfolgenden Glieder $\ell_{n-1}, \dots, \ell_{n-k}$ alle gleich 0, so sind auch alle folgenden ℓ_m ($m \geq n$) auch gleich 0. Die Folge hat also höchstens endlich-viele von 0 verschiedene Glieder.

Auf reguläre Sprachen übertragen heisst das: ist L eine solche Sprache und k die Anzahl der Zustände eines L akzeptierenden Automaten, so gilt

$$L \text{ ist unendlich} \Leftrightarrow \begin{cases} \text{von je } k \text{ aufeinanderfolgenden Zahlen} \\ \ell_n, \ell_{n+1}, \dots, \ell_{n+k-1} \text{ ist mindestens eine } \neq 0 \end{cases}$$

Wir werden später der Einfachheit halber nur spezielle Graphen und Matrizen betrachten. Alle Ergebnisse lassen sich auf die Situation allgemeiner gerichteter Graphen bzw nichtnegativer Matrizen verallgemeinern.

Definition 8. 1. Ein gerichteter Graph $G = (V, E)$ heisst *stark-zusammenhängend*, wenn es zu jedem Paar $(u, v) \in V \times V$ einen G -Pfad von u nach v gibt.

2. Eine nichtnegative Matrix A heisst *irreduzibel*, wenn der zugehörige Graph G_A stark-zusammenhängend ist, d.h.

$$\forall i, j \in V \exists n \in \mathbb{N} : a_{i,j}^{(n)} > 0$$

Definition 9. 1. Ein stark-zusammenhängender, gerichteter Graph $G = (V, E)$ heisst *primitiv*, wenn der grösste gemeinsame Teiler aller Längen von Zyklen in $G = 1$ ist.

2. Eine nichtnegative Matrix A heisst *primitiv*, wenn der zugehörige Graph G_A primitiv ist. Man kann das ausdrücken (dies ist ein Satz) durch

$$\exists n \in \mathbb{N} \forall i, j \in V : a_{i,j}^{(n)} > 0$$

d.h. die Matrix $A^{(n)}$ ist *positiv*.

Probleme in diesem allgemeineren Kontext sind nach wie vor:

$A = [a_{i,j}]_{1 \leq i,j \leq k}$ sei eine nichtnegative Matrix und für $n \in \mathbb{N}$ sei $A^n = [a_{i,j}^{(n)}]_{1 \leq i,j \leq k}$ ($n \geq 0$).

- Wie kann man $a_{i,j}^{(n)}$ für grosse n möglichst effizient berechnen?
- Was kann man über das Verhalten von $a_{i,j}^{(n)}$ aussagen, wenn n wächst?

4.4 Markov-Ketten

Ein wichtiger Spezialfall des beschriebenen Szenarios sind *Markov-Ketten* mit endlichem Zustandsraum. Als *Markov-Prozesse* bezeichnet man Zufallsprozesse, bei dem das Geschehen zu einem Zeitpunkt nur von dem aktuellen Zustand und der mit ihm zu diesem Zeitpunkt verbundenen “Zufallsregel” abhängt, *nicht jedoch* von der Vorgeschichte, wie man in diesen Zustand gelangt ist. Markov-Modelle haben “kein Gedächtnis”. *Markov-Modelle* haben in allen Bereichen stochastischer Simulation eine überragende Bedeutung. Dabei kann die Zeit diskret oder kontinuierlich modelliert sein, der Zustandsraum ebenfalls diskret oder kontinuierlich, wobei im diskreten Fall noch zwischen endlichen und unendlichen Zustandsräumen unterschieden. Wir betrachten hier nur den allereinfachsten Fall:

- diskrete Zeit $n = 0, 1, 2, 3, \dots$
- endlicher Zustandsraum $V = \{1, 2, \dots, k\}$
- Stationarität: die einem Zustand zugeordnete “Zufallsregel” ist zeitinvariant: sie bleibt immer gleich.

In dieser Situation wird das Zufallsgeschehen beschrieben durch eine nichtnegative $(k \times k)$ -Matrix $A = [a_{i,j}]_{1 \leq i,j \leq k}$ mit der Eigenschaft, dass alle Zeilen der Matrix Wahrscheinlichkeitsverteilungen auf dem Raum $V = \{1, 2, \dots, k\}$ sind, d.h. für die Zeilensummen gilt

$$r_i = \sum_{1 \leq j \leq k} a_{i,j} = 1 \quad (1 \leq i \leq k).$$

Man kann das auch so formulieren: ist $\mathbf{1} = (1, 1, \dots, 1)$ der Eins-Vektor der Länge k , so gilt

$$A \cdot \mathbf{1}^t = \mathbf{1}^t.$$

Dies bedeutet: $\mathbf{1}$ ist Rechts-Eigenvektor von A zum Eigenwert $\lambda = 1$.

Definition 10. 1. Eine nichtnegative Matrix ($k \times k$)-Matrix A mit $A \cdot \mathbf{1}^t = \mathbf{1}^t$ wird als *stochastische Matrix* bezeichnet.

2. Ein nichtnegativer Vektor $\mathbf{p} = (p_1, p_2, \dots, p_k)$ der Länge k mit $\mathbf{p} \cdot \mathbf{1}^t = 1$ wird als *stochastischer Vektor* bezeichnet.

Die Zeilen einer stochastischen Matrix sind also stochastische Vektoren. Die anschauliche Bedeutung dieser Begriffsbildung sind folgende:

- Ein stochastischer Vektor $\mathbf{p} = (p_1, p_2, \dots, p_k)$ stellt eine Wahrscheinlichkeitsverteilung auf dem Zustandsraum $V = \{1, 2, \dots, k\}$ dar: man befindet sich mit der Wahrscheinlichkeit p_j im Zustand “ j ”.

Man bezeichnet häufig die Elemente $1, 2, \dots, k$ von V als *reine Zustände* (*pure states* in der Physik) und die Wahrscheinlichkeitsverteilung von A als *gemischte Zustände* (*mixed states* in der Physik)

- Befindet sich ein Item zu Zeitpunkt t in einem Knoten “ i ” des Graphen G_A , so wird es sich zum Zeitpunkt $t + 1$ mit Wahrscheinlichkeit $a_{i,j}$ im Knoten “ j ” befinden.

Die gewichteten Kanten beschreiben also Transitionen in diesem Graphen, wobei die Kantengewichte die bedingten Wahrscheinlichkeiten darstellen, dass die entsprechende Transition vorkommt oder ausgeführt wird.

Lemma 22. *Ist A ein stochastische Matrix und $\mathbf{p} = (p_1, p_2, \dots, p_k)$ ein stochastischer (Zeilen-)vektor, so ist*

$$\mathbf{p}' = (p'_1, p'_2, \dots, p'_k) = \mathbf{p} \cdot A$$

wieder ein stochastischer Vektor.

Das ergibt sich ganz einfach:

$$\mathbf{p}' \cdot \mathbf{1}^t = \mathbf{p} \cdot A \cdot \mathbf{1}^t = \mathbf{p} \cdot \mathbf{1}^t = 1.$$

□

Beschreibt eine Wahrscheinlichkeitsverteilung $\mathbf{p} = (p_1, p_2, \dots, p_k)$ den (gemischten) Zustand zu einem Zeitpunkt t , so beschreibt $\mathbf{p}' = \mathbf{p} \cdot A$ den (gemischten) Zustand zum Zeitpunkt $t + 1$. Eine Zufallstransition wird also durch (Rechts-)Multiplikation mit der Matrix A dargestellt.

Induktiv ergibt sich daraus:

Ist mit dem Zufallsvektor $\mathbf{p} = (p_1, p_2, \dots, p_k)$ ein Anfangsverteilung auf V zum Zeitpunkt $t = 0$ gegeben, so sind die

$$(p_1^{(n)}, p_2^{(n)}, \dots, p_k^{(n)}) = (p_1, p_2, \dots, p_k) \cdot A^n \quad (n \geq 0)$$

die Verteilungen der Aufenthaltswahrscheinlichkeiten zu den Zeitpunkten $t = n$, also nach n Transitionen, die durch die in A benannten (bedingten) Wahrscheinlichkeiten gesteuert werden.

Man ist bei der Untersuchung von Markov-Modellen meist am *langfristigen* Verhalten interessiert, beispielsweise:

- In welchem Zustand befindet man sich mit welcher Wahrscheinlichkeit nach langer Zeit?
- Was geschieht generell, wenn man sehr viele Transitionen hintereinander ausführt: Stellt sich ein *stationärer* Zustand ein, also eine Verteilung $\mathbf{p} = (p_1, p_2, \dots, p_k)$ mit

$$\mathbf{p} \cdot A = \mathbf{p}$$

Das wäre ein Links-Eigenvektor von A zum Eigenwert $\lambda = 1$. Oder tritt oszillierendes Verhalten auf?

Das sind offensichtlich Fragen an die Matrizen A^n für grosses $n \in \mathbb{N}$

4.5 Asymptotisches Verhalten

Ein Blick auf die Formel

$$c_n = \alpha_1 \lambda_1^n + \alpha_2 \lambda_2^n + \dots + \alpha_k \lambda_k^n \quad (n \geq 0)$$

sagt kann man bezüglich des *Wachstums* der Folge $(c_n)_{n \geq 0}$ sofort interessante Feststellungen treffen. Dabei sollen die komplexen Zahlen $\lambda_1, \lambda_2, \dots, \lambda_k$ der absoluten Grösse nach geordnet, d.h. $|\lambda_1| \geq |\lambda_2| \geq \dots \geq |\lambda_k|$. Wir werden hier nur den Fall behandeln, dass sogar

$$|\lambda_1| > |\lambda_2| \geq \dots \geq |\lambda_k|$$

gilt. Man sagt dann, dass λ_1 *dominierende Nullstelle* (des charakteristischen Polynoms) ist. Dann gilt:

$$\frac{c_n}{\lambda_1^n} = \alpha_1 + \alpha_2 \left(\frac{\lambda_2}{\lambda_1}\right)^n + \dots + \alpha_k \left(\frac{\lambda_k}{\lambda_1}\right)^n$$

und die rechte Seite konvergiert für $n \rightarrow \infty$ gegen α_1 , und das exponentiell schnell! Ist $\alpha_1 \neq 0$, so hat man die asymptotische Aussage:

Für das Wachstum der Glieder einer C-rekursiven Folge gilt (sofern die beschriebenen Voraussetzungen gegeben sind)

$$c_n \sim \alpha_1 \lambda_1^n$$

Man hat also mit exponentiellem Wachstum zu rechnen, wobei der Betrag der dominante Nullstelle $|\lambda_1|$ die Wachstumsordnung bestimmt.

Was ist, wenn die einschränkenden Voraussetzungen nicht gegeben sind, wenn z.B. mehrere Eigenwerte denselben maximalen Absolutbetrag haben. Dann kann es zu Oszillationen kommen, also keine Konvergenz, was man auch mit Hilfe der obigen Darstellung quantitativ beschreiben kann.

Diese asymptotischen Aussagen über C-rekursive Folgen übertragen sich direkt auf das Verhalten der Potenzen A^n von Matrizen, deren Koeffizienten in gleiche Position ja C-rekursive Folgen mit demselben Rekursionspolynom sind.

Für eine Matrix A nennt man die Zahl

$$\rho(A) = \max\{|\lambda| ; \lambda \text{ Eigenwert von } A\}$$

den *Spektralradius* von A .

Lemma 23. Ist $A = [a_{i,j}]_{1 \leq i,j \leq k}$ irgendeine $(k \times k)$ -Matrix, bezeichnet $r_i = \sum_{1 \leq j \leq k} |a_{i,j}|$ die Summe der Absolutbeträge der Einträge in der i -ten Zeile, so gilt

$$\rho(A) \leq \max_{1 \leq i \leq k} r_i.$$

Analoges gilt für die Spaltensummen.

Beweis. Ist λ irgendein Eigenwert von A und $(x_1, \dots, x_k)^t$ ein (Rechts-)Eigenvektor zum Eigenwert λ , dann sei die ℓ -te Komponente die dem Absolutbetrag nach grösste der Komponenten dieses Vektors: $|x_\ell| = \max_j |x_j|$. Man erhält

$$|\lambda| |x_\ell| = |\lambda x_\ell| = \left| \sum_j a_{\ell,j} x_j \right| \leq \sum_j |a_{\ell,j}| |x_j| \leq \sum_j |a_{\ell,j}| \cdot |x_\ell| = r_\ell |x_\ell|$$

und somit $|\lambda| \leq r_\ell \leq \max_{1 \leq i \leq k} r_i$. □

Theorem 24 (PERRON-FROBENIUS).

Ist A eine primitive nichtnegative Matrix, so gilt

1. $\lambda_1 = \rho(A)$ ist ein einfacher Eigenwert von A .

2. Jeder andere Eigenwert λ von A ist absolut genommen strikt kleiner, d.h. $|\lambda| < \lambda_1$, d.h. λ_1 ist dominierender Eigenwert von A .
3. Zu λ_1 gibt es einen positiven Eigenvektor \mathbf{x} . Die Vektoren $\alpha\mathbf{x}$ mit $\alpha > 0$ sind die einzigen positiven Eigenvektoren von A .

Sei nun A eine primitive nichtnegative Matrix und λ_1 der dominierende Eigenwert, \mathbf{x} der positive linke Eigenvektor, \mathbf{y}^\top der positive rechte Eigenvektor zu λ_1 . Dann gilt folgendes:

- Die Matrizen $(\lambda_1^{-1}A)^n$ konvergieren gegen eine Matrix B :

$$\lim_{n \rightarrow \infty} \frac{A^n}{\lambda_1^n} = B$$

- Für die Matrix B muss $A \cdot B = \lambda_1 B$ gelten, denn

$$B \longleftarrow \frac{A^{n+1}}{\lambda_1^{n+1}} = \frac{A}{\lambda_1} \cdot \frac{A^n}{\lambda_1^n} \longrightarrow \frac{A}{\lambda_1} \cdot B$$

d.h. die Spalten der Matrix B sind Rechts-Eigenvektoren von A zum Eigenwert λ_1 , also skalare Vielfache von \mathbf{y}^\top .

- Analog zeigt man, dass die Zeilen von B Links-Eigenvektoren von A zum Eigenwert λ_1 sind, also skalare Vielfache von \mathbf{x} .
- Zusammengefasst ergibt das

$$B = \beta \mathbf{y}^\top \cdot \mathbf{x}$$

wobei $\beta > 0$ eine positive Konstante ist.

Theorem 25 (Hauptsatz über Markov-Ketten).

1. Ist A eine primitive stochastische Matrix, so ist $\lambda_1 = 1$ der dominante Eigenwert und es gilt

$$\lim_{n \rightarrow \infty} A^n = \begin{bmatrix} \mathbf{x} \\ \mathbf{x} \\ \vdots \\ \mathbf{x} \end{bmatrix}$$

wobei \mathbf{x} Links-Eigenvektor von A zu λ_1 ist. \mathbf{x} ist ein Wahrscheinlichkeitsvektor: die sog. stationäre Verteilung oder Gleichgewichtsverteilung der durch A gegebenen Markov-Kette.

2. Ist $\mathbf{p} = (p_1, p_2, \dots, p_k) = \mathbf{p}^{(0)}$ irgendeine Anfangsverteilung auf dem Zustandsraum $\{1, 2, \dots, k\}$, so konvergiert die Folge der transformierten Verteilungen (gemischte Zustände)

$$\mathbf{p}^{(n)} = \mathbf{p}^{(0)} \cdot A^n \quad (n = 1, 2, 3, \dots)$$

gegen die Gleichgewichtsverteilung

$$\lim_{n \rightarrow \infty} \mathbf{p}^{(n)} = \lim_{n \rightarrow \infty} \mathbf{p}^{(0)} \cdot A^n = \mathbf{p}^{(0)} \cdot \lim_{n \rightarrow \infty} A^n = \mathbf{p}^{(0)} \cdot \begin{bmatrix} \mathbf{x} \\ \mathbf{x} \\ \vdots \\ \mathbf{x} \end{bmatrix} = \mathbf{x}.$$

Bemerkung 11. Zur Konvergenzgeschwindigkeit: für eine C-rekursive Folge mit λ_1 als dominierender Nullstelle gilt, wie gesehen:

$$\frac{c_n}{\lambda_1^n} = \alpha_1 + \alpha_2 \left(\frac{\lambda_2}{\lambda_1}\right)^n + \dots + \alpha_k \left(\frac{\lambda_k}{\lambda_1}\right)^n$$

also

$$\lim_{n \rightarrow \infty} \frac{c_n}{\lambda_1^n} = \alpha_1$$

und bezüglich der Konvergenzgeschwindigkeit gilt

$$\begin{aligned} \left| \frac{c_n}{\lambda_1^n} - \alpha_1 \right| &\leq \left| \alpha_2 \left(\frac{\lambda_2}{\lambda_1}\right)^n + \dots + \alpha_k \left(\frac{\lambda_k}{\lambda_1}\right)^n \right| \\ &\leq \left| \frac{\lambda_2}{\lambda_1} \right|^n \cdot (|\alpha_2| + \dots + |\alpha_k|) \in \mathcal{O} \left(\left| \frac{\lambda_2}{\lambda_1} \right|^n \right). \end{aligned}$$

Die dokumentiert die exponentiell-schnelle Konvergenz und die Tatsache, dass es dabei auf das Verhältnis von dominierendem und (absolut genommen) zweitgrössten Eigenwert ankommt.

4.6 Googles PageRank-Algorithmus

Dies ist derzeit die mutmasslich populärste Anwendung der hier behandelten Begriffe und Methoden. Die Google-Suchmaschine verwendet zum *ranking* der Suchresultate zu einer Anfrage die stationäre Verteilung zu einer stochastischen Matrix G , der Google-Matrix, die aus den für die jeweilige Suche relevanten Webseiten gewonnen wird. In diesem abschnitt geht es nur um die Konstruktion dieser Matrix, für das ganze Szenario der Suchmaschine sollte man speziellere Literatur konsultieren.

Es sei also $\Gamma = (V, E)$ ein gerichteter Graph, wobei $V = \{1, 2, \dots, k\}$ angenommen werden kann und $E \subseteq V \times V$ ist. Unter den Knoten aus V hat man sich Webseiten vorzustellen, die durch Kanten aus E verlinkt sind. Von Γ werden keine besonderen Eigenschaften verlangt (Zusammenhang, starker Zusammenhang, Primitivität); Γ kann ohne weiteres auch Nullzeilen enthalten, also Webseiten, von denen aus keine Links weiterführen. In der Regel wird A eine dünn besetzte Matrix sein.

$$A = [A_{i,j}]_{1 \leq i,j \leq k} = A_\Gamma$$

sei die Adjazenzmatrix von G . Es gilt also:

$$A_{i,j} = \begin{cases} 1 & \text{falls } (i, j) \in E \\ 0 & \text{sonst} \end{cases}$$

Aus A wird nun schrittweise die Google-Matrix G konstruiert, wobei die Autoren Sergej BRIN und Larry PAGE und das mutmassliche Verhalten eines Websurfers im Auge hatten. Zunächst noch eine notationelle Bemerkung:

Mit $\mathbf{1}_k = (1, 1, \dots, 1)$ wird der Einsvektor der Länge k (als Zeilenvektor aufgefasst) bezeichnet. Es gilt:

$$\begin{aligned} \mathbf{1}_k \cdot \mathbf{1}_k^\top &= k \\ \mathbf{1}_k^\top \cdot \mathbf{1}_k &= \mathbb{J}_k = \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & 1 & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \dots & 1 \end{bmatrix} \end{aligned}$$

Der Vektor der Zeilensummen von A ist

$$A \cdot \mathbf{1}_k^\top = \mathbf{r}^\top = (r_1, r_2, \dots, r_k)^\top$$

Ferner seien \mathbf{u} bzw. \mathbf{v} die Indikatorvektoren für die Nicht-Nullzeilen bzw. die Nullzeilen von A :

$$\begin{aligned} \mathbf{u} &= (u_1, u_2, \dots, u_k) \quad \text{mit} \quad u_i = \begin{cases} 1 & \text{falls } r_i > 0 \\ 0 & \text{falls } r_i = 0 \end{cases} \\ \mathbf{v} &= (v_1, v_2, \dots, v_k) \quad \text{mit} \quad v_i = \begin{cases} 0 & \text{falls } r_i > 0 \\ 1 & \text{falls } r_i = 0 \end{cases} \end{aligned}$$

Somit ist $\mathbf{u} + \mathbf{v} = \mathbf{1}_k$.

- Die Matrix B entsteht aus A , indem man alle Nicht-Nullzeilen so normalisiert werden, dass die jeweiligen Zeilensumme $=1$ ist, d.h. die Nicht-Nullzeilen von B sind stochastische Vektoren: man dividiert durch die Zeilensumme, falls diese $\neq 0$ ist.

$$B = [B_{i,j}]_{1 \leq i,j \leq k} \quad \text{mit} \quad B_{i,j} = \begin{cases} \frac{1}{r_i} A_{i,j} & \text{falls } r_i > 0 \\ 0 & \text{falls } r_i = 0 \end{cases}$$

Es gilt also

$$B \cdot \mathbf{1}_k^t = \mathbf{u}^t$$

- Die Matrix

$$C = [C_{i,j}]_{1 \leq i,j \leq k}$$

entsteht dadurch, dass man in B eventuelle Nullzeilen durch stochastische Vektoren $\frac{1}{k} \mathbf{1}_k$ ersetzt:

$$C = B + \mathbf{v}^t \cdot \frac{1}{k} \mathbf{1}_k$$

C ist eine stochastische Matrix. Nichtnegativ ist sie offensichtlich und es ist

$$C \cdot \mathbf{1}_k^t = B \cdot \mathbf{1}_k^t + \frac{1}{k} \mathbf{v}^t \cdot \mathbf{1}_k \cdot \mathbf{1}_k^t = \mathbf{u}^t + \mathbf{v}^t = \mathbf{1}_k^t$$

C muss aber nicht primitiv sein!

- Sei nun γ mit $0 < \gamma < 1$ ein Parameter (bei Google ist er mit $\gamma = 0.15$ festgelegt), die sogenannte *Teleportationskonstante*. Sie gibt die Wahrscheinlichkeit dafür an, dass ein Surfer von einer Webseite “ i ” aus einem der r_i Links folgt, und zwar mit Gleichverteilung. Mit Wahrscheinlichkeit $1 - \gamma$ springt er zu irgendeiner der anderen Seiten (wiederum gleichverteilt). Die *Googlematrix* G ist nun

$$G = \gamma C + (1 - \gamma) \frac{1}{k} \mathbb{J}_k.$$

Als konvexe Kombination von stochastischen Matrizen ist G wiederum eine stochastische Matrix. Ausserdem ist G primitiv, da G eine positive Matrix ist.

- Gesucht ist also der eindeutig bestimmte (positive) Linkseigenvektor \mathbf{x} von G mit $\|\mathbf{x}\|_1 = \sum_{1 \leq i \leq k} x_i = 1$ zum Eigenwert $\lambda = 1$: das ist Googles PageRank-Vektor.

Bleibt die Frage, wie man man \mathbf{x} berechnet? Das exakte Lösen eines riesigen Eigenwertproblems (k kann sehr gross sein!) kommt nicht in Frage. Man macht es durch Approximation! Dabei ist folgender Aspekt für die Effizienz relevant:

- Die Matrix G ist als *positive* Matrix so dicht besetzt, wie nur möglich. Bei der Iteration

$$\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} \cdot G \quad (n = 1, 2, 3, \dots)$$

hätte man immer der jeweils aktuellen Zustandsvektor $\mathbf{x}^{(n)}$ mit der vollen Matrix G zu multiplizieren. Das kann man billiger haben!

- Die Iteration kann auch als

$$\mathbf{x}^{(n+1)} = \gamma \mathbf{x}^{(n)} \cdot B + \frac{1}{k} (\mathbf{x}^{(n)} \cdot (\gamma \mathbf{v}^t + (1 - \gamma) \mathbf{1}_k^t)) \mathbf{1}_k$$

schreiben. Das hat den Vorteil, dass im Matrix-Vektorprodukt $\mathbf{x}^{(n)} \cdot B$ die Matrix B mutmasslich dünn besetzt ist, das Produkt also deutlich weniger kostet als das Produkt $\mathbf{x}^{(n)} \cdot G$. Ansonsten sind nur Vektoroperationen erforderlich.

Beispiel 4. Als erstes Beispiel wird folgender Graph betrachtet:

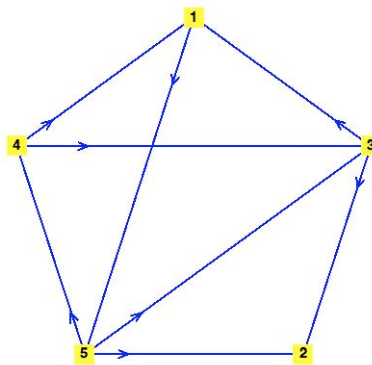


Abbildung 7: Gerichteter Graph mit 5 Knoten und 8 Kanten

Die Adjazenzmatrix ist

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Normalisieren der Nicht-Nullzeilen:

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \\ 0 & 1/3 & 1/3 & 1/3 & 0 \end{bmatrix}$$

Einfügen einer Gleichverteilung für Nullzeilen:

$$C = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \\ 0 & 1/3 & 1/3 & 1/3 & 0 \end{bmatrix}$$

Google-Matrix mit $\gamma = 1/3$:

$$G = \begin{bmatrix} 2/15 & 2/15 & 2/15 & 2/15 & \frac{7}{15} \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 3/10 & 3/10 & 2/15 & 2/15 & 2/15 \\ 3/10 & 2/15 & 3/10 & 2/15 & 2/15 \\ 2/15 & \frac{11}{45} & \frac{11}{45} & \frac{11}{45} & 2/15 \end{bmatrix}$$

Eine Potenz der Google-Matrix (auf 5 Stellen gerundet)

$$G^{10} = \begin{bmatrix} 0.20870 & 0.20426 & 0.19951 & 0.17101 & 0.21652 \\ 0.20870 & 0.20426 & 0.19951 & 0.17101 & 0.21652 \\ 0.20870 & 0.20426 & 0.19951 & 0.17101 & 0.21652 \\ 0.20870 & 0.20426 & 0.19951 & 0.17101 & 0.21652 \\ 0.20870 & 0.20426 & 0.19951 & 0.17101 & 0.21652 \end{bmatrix}$$

Die Eigenwerte von G :

$$\begin{bmatrix} -0.048808 - 0.042064 i \\ -0.048822 + 0.042064 i \\ -0.084515 - 0.20633 i \\ -0.084523 + 0.20633 i \\ 1.0 \end{bmatrix}$$

Der Linkseigenvektor von G zum Eigenwert $\lambda = 1$:

$$\left[0.96391 \quad 0.94338 \quad 0.92144 \quad 0.78981 \quad 1.0 \right]$$

Der Eigenvektor nach Normalisierung:

$$\left[0.2087044824 \quad 0.2042593547 \quad 0.1995089357 \quad 0.1710085871 \quad 0.2165186401 \right]$$

Vergleiche mit der Matrix G^{10} !

Beispiel 5. Als zweites Beispiel ein Graph mit 10 Knoten und 20 Kanten:

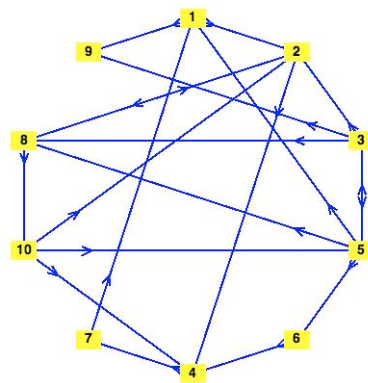


Abbildung 8: Gerichteter Graph mit 10 Knoten und 20 Kanten

Adjazenzmatrix

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Google-Matrix mit $\gamma = 0.2$:

$$G = \begin{bmatrix} 0.0800 & 0.180 & 0.0800 & 0.0800 & 0.0800 & 0.0800 & 0.0800 & 0.0800 & 0.180 & 0.0800 \\ 0.0800 & 0.0800 & 0.0800 & 0.180 & 0.0800 & 0.0800 & 0.0800 & 0.180 & 0.0800 & 0.0800 \\ 0.0800 & 0.130 & 0.0800 & 0.0800 & 0.130 & 0.0800 & 0.0800 & 0.130 & 0.130 & 0.0800 \\ 0.0800 & 0.0800 & 0.0800 & 0.0800 & 0.0800 & 0.0800 & 0.280 & 0.0800 & 0.0800 & 0.0800 \\ 0.130 & 0.0800 & 0.130 & 0.0800 & 0.0800 & 0.130 & 0.0800 & 0.130 & 0.0800 & 0.0800 \\ 0.0800 & 0.0800 & 0.0800 & 0.280 & 0.0800 & 0.0800 & 0.0800 & 0.0800 & 0.0800 & 0.0800 \\ 0.280 & 0.0800 & 0.0800 & 0.0800 & 0.0800 & 0.0800 & 0.0800 & 0.0800 & 0.0800 & 0.0800 \\ 0.0800 & 0.180 & 0.0800 & 0.0800 & 0.0800 & 0.0800 & 0.0800 & 0.0800 & 0.0800 & 0.180 \\ 0.100 & 0.100 & 0.100 & 0.100 & 0.100 & 0.100 & 0.100 & 0.100 & 0.100 & 0.100 \\ 0.0800 & 0.147 & 0.0800 & 0.147 & 0.147 & 0.0800 & 0.0800 & 0.0800 & 0.0800 & 0.0800 \end{bmatrix}$$

Die Matrix G^{20} (auf drei Stellen gerundet):

$$G^{20} = \begin{bmatrix} 0.108 & 0.113 & 0.0866 & 0.117 & 0.0924 & 0.0866 & 0.105 & 0.102 & 0.0970 & 0.0922 \\ 0.108 & 0.113 & 0.0866 & 0.117 & 0.0924 & 0.0866 & 0.105 & 0.102 & 0.0970 & 0.0922 \\ 0.108 & 0.113 & 0.0866 & 0.117 & 0.0924 & 0.0866 & 0.105 & 0.102 & 0.0970 & 0.0922 \\ 0.108 & 0.113 & 0.0866 & 0.117 & 0.0924 & 0.0866 & 0.105 & 0.102 & 0.0970 & 0.0922 \\ 0.108 & 0.113 & 0.0866 & 0.117 & 0.0924 & 0.0866 & 0.105 & 0.102 & 0.0970 & 0.0922 \\ 0.108 & 0.113 & 0.0866 & 0.117 & 0.0924 & 0.0866 & 0.105 & 0.102 & 0.0970 & 0.0922 \\ 0.108 & 0.113 & 0.0866 & 0.117 & 0.0924 & 0.0866 & 0.105 & 0.102 & 0.0970 & 0.0922 \\ 0.108 & 0.113 & 0.0866 & 0.117 & 0.0924 & 0.0866 & 0.105 & 0.102 & 0.0970 & 0.0922 \\ 0.108 & 0.113 & 0.0866 & 0.117 & 0.0924 & 0.0866 & 0.105 & 0.102 & 0.0970 & 0.0922 \\ 0.108 & 0.113 & 0.0866 & 0.117 & 0.0924 & 0.0866 & 0.105 & 0.102 & 0.0970 & 0.0922 \end{bmatrix}$$

Die Eigenwerte von G :

$$\begin{bmatrix} 1.0 + 0.0i \\ -0.15695 + 0.0i \\ 0.017904 + 0.11298i \\ 0.017904 - 0.11298i \\ -0.059122 + 0.081793i \\ -0.059122 - 0.081793i \\ 0.053037 + 0.0i \\ 0.0031723 + 0.048925i \\ 0.0031723 - 0.048925i \\ 3.9406 \times 10^{-17} + 0.0i \end{bmatrix}$$

Der Links-Eigenvektor zum Eigenwert $\lambda = 1$:

$$[-0.339 \quad -0.357 \quad -0.272 \quad -0.367 \quad -0.291 \quad -0.272 \quad -0.331 \quad -0.322 \quad -0.305 \quad -0.290]$$

Der normalisierte Eigenvektor:

$$[0.108 \quad 0.113 \quad 0.0865 \quad 0.117 \quad 0.0925 \quad 0.0865 \quad 0.105 \quad 0.102 \quad 0.0969 \quad 0.0922]$$

Vergleiche wiederum mit der Matrix G^{20} !

4.7 Komplexität

- Es geht in diesem Abschnitt um eine fundamentale algorithmische Aufgabe:
 - Für einen Graphen $G = (V, E)$ und zwei Knoten $i, j \in V$ und $n \in \mathbb{N}$ sind die (exakten!) Anzahlen $a_{i,j}^{(n)} = \# [i, j]_G^n$ der Pfade der Länge n von i nach j in G zu berechnen.

Entsprechend formuliert man diese Aufgabe für Graphen mit Kantengewichten und auch für endliche Automaten. Wegen dieser Bedeutung wird nochmals separat formuliert:

- Eine reguläre Sprache $L \subseteq \Sigma^*$ sei spezifiziert durch einen endlichen Automaten, eine Typ-3-Grammatik oder einen regulären Ausdruck. $\ell_n = \#(L \cap \Sigma^n)$ bezeichne die Anzahl der Wörter der Länge n von L . Berechne ℓ_n (exakt!) für ein gegebenes (grosses) n .

Es werden im Folgenden drei Lösungen von drastisch unterschiedlicher Komplexität angeboten!

- Zunächst eine Beschreibung des Begriffes *Komplexität* (Siehe Intro-2006)
- Im Kontext dieses Kapitels wird die *Grösse* von ganzen Zahlen x zu einer Basis $b \geq 2$ gemessen durch

$$\mathbf{size}_b(x) = 1 + \lceil \log_b(1 + |x|) \rceil,$$

d.h. es gilt

$$\mathbf{size}_b(x) \in \Theta(\log |x|)$$

(unabhängig von der speziellen Basis b). Umgekehrt formuliert:

$$x \in \Theta(b^{\mathbf{size}(x)}),$$

d.h. x ist *exponentiell* in $\mathbf{size}(x)$.

Für die eingangs formulierte Aufgabenstellung wird der Graph G (oder der Automat usw.) mittels seiner Knotenanzahl k in die Betrachtungen eingehen. Insbesondere interessiert aber die Abhängigkeit von der Pfadlänge bzw. Wortlänge n .

- Erstes Verfahren:
 Probiere alle Möglichkeiten für Pfade bzw. Wörter der Länge n systematisch aus.

Man sieht sofort, dass das völlig unakzeptabel ist, da die Anzahl der Kandidaten *exponentiell* in n (und damit *doppelt-exponentiell* in $\mathbf{size}(n)$) wächst. Bei endlichen Automaten etwa würde man $(\#\Sigma)^n$ Läufe auf allen möglichen Inputs der Länge n machen. Da ist es unerheblich dass jeder Lauf noch einmal n Schritte erfordert.

- Zweites Verfahren (grafische Version):
Berechne iterativ für $t = 1, 2, 3, \dots, n$ den Vektor

$$\mathbf{a}_i^{(t)} = \left(a_{i,1}^{(t)}, a_{i,2}^{(t)}, \dots, a_{i,k}^{(t)} \right)$$

mittels Trellis-Diagramm.

Man sieht sofort, dass diese “klassische” Ingenieursmethode nichts anderes ist als die graphische Realisierung der Matrizenpotenzierung:

- Zweites Verfahren (algebraische Version):
Berechne iterativ für $t = 1, 2, 3, \dots, n$ den Vektor

$$\mathbf{a}_i^{(t)} = \left(a_{i,1}^{(t)}, a_{i,2}^{(t)}, \dots, a_{i,k}^{(t-1)} \right) = \mathbf{a}_i^{(t-1)} \cdot A$$

ausgehend vom Startvektor $\mathbf{a}_i^{(0)} = \mathbf{e}_i = (0, \dots, 0, 1, 0, \dots, 0)$:

$$\mathbf{a}_i^{(n)} = \mathbf{e}_i \cdot A^n$$

Jede Multiplikation eines k -Vektors mit einer $(k \times k)$ -Matrix kostet $\mathcal{O}(k^2)$ arithmetische Operationen (Multiplikationen und Additionen). Insgesamt hat man für die Berechnung von $\mathbf{a}_i^{(n)}$ einen Aufwand von $\mathcal{O}(n \cdot k^2)$ arithmetischen Operationen zu veranschlagen.

Man kann das aber auch so sehen: man berechnet erste die Matrixpotenz A^n zieht daraus die i -te Zeile heraus. Wenn man A^n iterativ durch sukzessive Multiplikation berechnet, sieht die Bilanz in arithmetischen Operationen gemessen folgendermassen aus

- Es sind $n - 1$ Multiplikationen von $(k \times k)$ -Matrizen auszuführen.
- Jede Multiplikation zweier $(k \times k)$ -Matrizen erfordert $\Theta(k^3)$ arithmetische Operationen

- Der Gesamtaufwand ist also $\Theta(k^3 \cdot n)$, oder $\Theta(n)$, wenn man k als Konstante behandelt.

- Drittes Verfahren :
Berechne iterativ für $t = 1, 2, 3, \dots, n$ die Matrizenpotenz A^t mittels “schneller Exponentiation” und ziehe dann die i -te Zeile heraus.

Im Vorgriff auf den folgenden Abschnitt wird hier zur Analyse die Tatsache herangezogen, dass die “schnelle Exponentiation” zur Berechnung von A^n mit $\Theta(\log n)$ Multiplikationen auskommt. Der Gesamtaufwand an arithmetischen Operationen ist also nur noch $\Theta(k^3 \cdot \log n)$, und das ist ein effizientes Verfahren: es ist *polynomiell in der Inputgröße*.

4.8 Schnelle Exponentiation

Generelles Szenario: gegeben sei ein Monoid (H, \circ, e) , d.h. $\circ : H \times H \rightarrow H$ ist eine *assoziative* Operation und e ist neutrales Element. Für $a \in H$ und $n \in \mathbb{N}_{>0}$ sind die Potenzen $\mathbf{exp}(a, n) = a^n$ induktiv definiert:

$$a^0 = e, \quad a^{n+1} = a^n \circ a$$

Folgt man dieser Definition, so werden für die Berechnung von a^n genau $n - 1$ Multiplikationen benötigt. Es geht aber besser, wenn man ausnutzt, dass

$$k + \ell = n \quad \Rightarrow \quad a^n = a^k \circ a^\ell$$

gilt. Ist nämlich

$$n = \sum_{i \geq 0} n_i 2^i = n_0 + 2n_1 + 4n_2 + 8n_3 + \dots \quad \text{mit } n_i \in \{0, 1\}$$

die Binärdarstellung von n , so gilt

$$a^n = a^{n_0} \circ (a^2)^{n_1} \circ (a^4)^{n_2} \dots$$

1. LR-Exponentiation

(a) Rekursive Definition von \mathbf{exp} (erste Version)

$$\mathbf{exp}(a, n) = \begin{cases} e & \text{falls } n = 0 \\ \mathbf{exp}(a^2, n/2) & \text{falls } n \text{ gerade} \\ a \circ \mathbf{exp}(a^2, (n-1)/2) & \text{falls } n \text{ ungerade} \end{cases}$$

Algorithm 1 LR-Exponentiation

```

procedure EXPLR( $a \in H, n \in \mathbb{N}$ )
   $y := e$ 
  while  $n > 0$  do
    if  $n$  odd then
       $y := a \circ y$ 
    end if
     $a := a \circ a$ 
     $n := \lfloor n/2 \rfloor$ 
  end while
  return  $y$ 
end procedure

```

(b) Iteratives Programm (*left-to-right exponentiation*)
(siehe Algorithmus 1)

2. Rekursive Definition von \exp (zweite Version)

(a)

$$\exp(a, n) = \begin{cases} e & \text{falls } n = 0 \\ \exp(a, n/2)^2 & \text{falls } n \text{ gerade} \\ a \circ \exp(a, (n-1)/2)^2 & \text{falls } n \text{ ungerade} \end{cases}$$

(b) Iteratives Programm (*right-to-left exponentiation*)
(siehe Algorithmus 2)

Algorithm 2 RL-Exponentiation

```

procedure EXPRL( $a \in H, n \in \mathbb{N}$ )
   $m := \log_2 n + 1$ 
   $y := e$ 
  while  $m > 1$  do
     $m := \lfloor m/2 \rfloor$ 
     $y := y \circ y$ 
    if  $n \geq m$  then
       $y := a \circ y$ 
       $n := n - m$ 
    end if
  end while
  return  $y$ 
end procedure

```

Zur Komplexität

Die Abhängigkeit des Algorithmus von der Binärdarstellung des Exponenten n macht klar, dass die Anzahl der Durchläufe der while-Schleife bei beiden Versionen $\log n$ ist. Pro Schleifendurchlauf hat man eine Quadrierung und zusätzlich noch eine weitere Multiplikation, wenn die entsprechende der Binäziffer von $n = 1$ ist. Damit ist die arithmetische Komplexität $\Theta(\log n)$. Genauer:

Die Anzahl der Multiplikationen (inklusive Quadrierungen) zur Berechnung von a^n mittels “Schneller Exponentiation” ist

$$\log_2 n + \nu_2(n) - 2.$$

Dabei ist $\nu_2(n) = \sum n_i$ die Anzahl der Einsen in der Binärdarstellung von n .

Eine Warnung

Man muss ausdrücklich darauf hinweisen, dass es sich bei der vorangehenden Betrachtung um die *arithmetische Komplexität* handelt, nicht um die *Bitkomplexität*! Bezieht man die Operandengröße in die Betrachtungen mit ein, so ist der Vorteil der “schnellen Methode gegenüber der schlichten iterativen Multiplikation keineswegs mehr klar! Anders formuliert: “Schnelle Exponentiation” bietet bezüglich der Bitkomplexität nur dann einen Vorteil in der Größenordnung, wenn man in Zahlbereichen rechnet, bei denen die Operandengröße beschränkt ist, z.B. in den Ringen \mathbb{Z}_n , in endlichen Körpern usw. Aber dort ist der Vorteil erheblich!

Die Überlegungen dieses Abschnitts betreffen natürlich auch die Aufgabe

Gegeben eine C-rekursive Folge $(c_n)_{n \geq 0}$ durch die Rekursion

$$c_n = a_1 c_{n-1} + a_2 c_{n-2} + \cdots + a_k c_{n-k} \quad (n \geq k),$$

d.h. durch Angabe der Koeffizienten a_1, a_2, \dots, a_k , sowie k Anfangswerte c_0, c_1, \dots, c_{k-1} .

Zu gegebenem $n \in \mathbb{N}$ berechne man c_n .

Man kann die Technik der “Schnellen Exponentiation” auf die Begleitmatrix C_a anwenden oder alternativ auf die explizite Lösungsformel

$$c_n = \alpha_1 \lambda_1^n + \cdots + \alpha_k \lambda_k^n \quad (n \geq 0)$$

mit den λ_j als den Nullstellen des charakteristischen Polynoms von C_a . Letzteres hat den Nachteil, dass die λ_j in der Regel komplexe Zahlen sind, mit denen

man nicht mehr so ohne weiteres exakt rechnet. (Es geht schon, da es sich um sog. “algebraische Zahlen” handelt, aber das ist aufwendig). Rechnet man nicht exakt, sondern mit Fließkommazahlen, so wird man in beiden Fällen sehr schnell Probleme mit der Akkumulation von Rundungsfehlern bekommen.

Die exakte Formel taugt kaum zur expliziten Berechnung exakter Werte, sie hat aber einen ganz anderen und wichtigen Vorteil: in vielen Fällen ist man gar nicht an den exakten Werten interessiert, sondern an der Grössenordnung des Wachstums. Dafür ist diese Formel ideal geeignet.

4.9 Kommentare, Literatur, Ausblicke

4.9.1 Historie

Das grundlegende Theorem von PERRON (siehe [22, 23]) und FROBENIUS (siehe [9, 10, 11]) ist ziemlich genau hundert Jahre alt. Die Theorie der nichtnegativen Matrizen ist seither wegen ihrer vielseitigen Aspekte und Anwendungen in vielen verschiedenen Bereiche der Mathematik intensiv erforscht worden. Der Artikel [17] von MacCluer macht das schon im Titel deutlich. Das Buch [19] von MINC gibt einen guten Überblick. Das kürzlich erschienene Buch [5] von BRUALDI und CVETKOVIC bietet eine leicht verständliche Einführung.

4.9.2 Graphen und Automaten, Viterbi-Algorithmus

Dass Wege in Graphen, endliche Automaten, reguläre Sprachen Aspekte desselben Problems sind, ist in der Automatentheorie schon lange bekannt, es ist sozusagen "Folklore". Für tieferschürfende Aspekte wird auf den letzten Unterabschnitt (4.10.4) verwiesen. Eine elementare Darstellung findet man in dem Buch [18] von MCELIECE ET AL. im sechsten und siebten Kapitel. Insbesondere wird dort auch die Technik der Trellis-Diagramme besprochen. Als konkrete Anwendung wird ein berühmter Algorithmus aus der Nachrichtentechnik besprochen, der VITERBI-Algorithmus, und am Beispiel der Decodierung von sog. *Faltungscodes* (CONVOLUTIONAL CODES) demonstriert. Dieser Algorithmus hat wichtige Anwendungen im Bereich der Mustererkennung, Stichwort *Hidden-Markov-Modelle*. Wer sich genauer ansehen will, wie Trellis-Diagramme, Codes, endliche Automaten, VITERBI-Algorithmus und Faltungscodes zusammenspielen, findet im Kapitel 14 des Buches [24] eine hervorragende Einführung. Wer zurück an die Quellen gehen will sei auf den (für seine Zeit) enzyklopädischen "Klassiker" [27] von VITERBI und OMURA hingewiesen.

4.9.3 Markov-Ketten

Markov-Prozesse (erfunden von A. A. MARKOV im Jahr 1906), speziell der Fall von Prozessen mit diskreter Zeit und endlichem Zustandsraum, also Markov-Ketten, sind nicht nur ein, sondern sogar "das" Standardmodell der probabilistischen Modellbildung mit Anwendungen in vielen Bereichen. Entsprechend reichhaltig ist die Literatur, aus der hier nur auf die Bücher [12, 13] von HÄGGSTRÖM und [1] von Behrends hingewiesen werden soll. Das Buch [26] ist eine Monografie über die graphentheoretische Struktur von Markov-Ketten. Jedes Standardwerk über

Wahrscheinlichkeitsrechnung widmet sich diesem Thema. Auch einführende Literatur speziell für Informatiker gibt es, z.B. das Buch [7] von DÜMBGEN. Ein ältere deutschsprachiger Text [8] von FRITZ, HUPPERT und WILLEMS gibt eine immer noch lesenswerte Einführung in den Bereich der stochastischen Matrizen.

Markov-Prozesse spielen innerhalb der Informatik im Bereich der *Probabilistischen Algorithmen* eine besonders wichtige Rolle, das sieht man bspw. in den sehr gut zugänglichen Büchern [21] von MOTWANI und RAGHAVAN und [20] von MITZENMACHER und UPFAL.

4.9.4 Algorithmen für den Web-Graphen

Der Algorithmus PageRank von GOOGLE geht auf die Google-Gründer S. BRIN und L. PAGE zurück, siehe [4]. In Buchform ist Google in [16] beschrieben worden. Speziell die mathematischen Aspekte beschreibt ein schöner Artikel [6] von BRYAN und LEISE. Ein Konkurrent von PageRank, ebenfalls auf der Berechnung von Eigenvektoren basierend, ist der Algorithmus HITS von L. KLEINBERG (siehe [14]). PageRank und HITS sind auch in dem Buch von [3] beschrieben, das sich generell graphentheoretischen Aspekten des Internets (oder ähnlicher grosses Graphen mit Zufallsstruktur) widmet.

4.9.5 Reguläre und kontextfreie Sprachen

- Es gilt (mit der gebotenen Vorsicht) die Entsprechung

$$\text{Reguläre Sprachen} \Leftrightarrow \text{Lineare Gleichungssysteme}$$

Wenn man dabei “Lineare Gleichungssysteme mit nicht-kommutativen Variablen” (also: Lineare Gleichungssystem für Formale Sprachen) meint, besteht sogar eine tatsächliche präzise Entsprechung. Wer dieser Beziehung auf den Grund gehen will, findet in den Büchern [25] von A. Salomaa und M. Siottola, sowie von [2] von J. Berstel und Ch. Reutenauer, einen Einstieg (auf hohem Niveau) und in dem Buch [15] von W. Kuich und A. Salomaa eine umfassende Abhandlung

Auf der nächsten Stufe der Chomsky-Hierarchie gilt analog:

$$\text{Kontextfreie Sprachen} \Leftrightarrow \text{Polynomiale Gleichungssysteme}$$

An die Stelle der C-rekursiven Folgen treten dann sog. *P-rekursive Folgen*, das sind Folgen, die durch lineare Rekursionsgleichungen mit *polynomialen* Koeffizienten bestimmt sind.

All das verdanken wir M.P. SCHÜTZENBERGER – aber das ist eine andere Geschichte... Das schon genannte Buch [15] von Kuich und Salomaa deckt auch dies mit ab.

Literatur

- [1] Ehrhard Behrends. *Introduction to Markov Chains*. Advanced Lectures in Mathematics. Friedr. Vieweg & Sohn, Braunschweig, 2000. With special emphasis on rapid mixing.
- [2] Jean Berstel and Christophe Reutenauer. *Rational Series and Their Languages*, volume 12 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1984.
- [3] Anthony Bonato. *A Course on the Web Graph*, volume 89 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 2008.
- [4] Sergej Brin and Larry Page. Anatomy of a large-scale hypertextual web search engine. In *Proc. 7th International World Wide Web Conference*, 1998.
- [5] Richard A. Brualdi and Dragos Cvetkovic. *A Combinatorial Approach to Matrix Theory and Its Applications*. Discrete Mathematics and Its Applications. CRC Press, 2009.
- [6] Kurt Bryan and Tanya Leise. The \$25,000,000,000 eigenvector: The linear algebra behind google. *SIAM Rev.*, 48(3):569–581, 2006.
- [7] Lutz Dümbgen. *Stochastik für Informatiker*. Reihe: Statistik und ihre Anwendungen. Springer-Verlag, 2003.
- [8] Franz-Josef Fritz, Bertram Huppert, and Wolfgang Willems. *Stochastische Matrizen*. Springer-Verlag, Berlin, 1979. Hochschultext.
- [9] Georg Frobenius. Über Matrizen aus positiven Elementen. *Sitzungsber. Preuss. Akad. Wiss.*, pages 471–476 und 514–518, 1908-9.
- [10] Georg Frobenius. Über Matrizen aus nicht negativen Elementen. *Sitzungsber. Preuss. Akad. Wiss.*, pages 456–477, 1912.
- [11] Georg Frobenius. Über zerlegbare Determinanten. *Sitzungsber. Preuss. Akad. Wiss.*, pages 274–277, 1917.

-
- [12] Olle Häggström. *Finite Markov Chains and Algorithmic Applications*, volume 52 of *London Mathematical Society Student Texts*. Cambridge University Press, Cambridge, 2002.
- [13] Olle Häggström. *Streifzüge durch die Wahrscheinlichkeitstheorie*. Springer-Verlag, 2006.
- [14] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999.
- [15] Werner Kuich and Arto Salomaa. *Semirings, Automata, Languages*, volume 5 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, Berlin, 1986.
- [16] Amy N. Langville and Carl D. Meyer. *Google's PageRank and beyond: the science of search engine rankings*. Princeton University Press, Princeton, NJ, 2006.
- [17] C. R. MacCluer. The many proofs and applications of Perron's theorem. *SIAM Rev.*, 42(3):487–498, 2000.
- [18] Robert J. McEliece, Robert B. Ash, and Carol Ash. *Introduction to Discrete Mathematics*. McGraw-Hill, 1989.
- [19] Henryk Minc. *Nonnegative Matrices*. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons Inc., New York, 1988. A Wiley-Interscience Publication.
- [20] Michael Mitzenmacher and Eli Upfal. *Probability and Computing*. Cambridge University Press, Cambridge, 2005. Randomized algorithms and probabilistic analysis.
- [21] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, Cambridge, 1995.
- [22] Oskar Perron. Grundlagen für eine Theorie des Jacobischen Kettenbruchalgorithmus. *Math. Annalen*, 64:11–76, 1907.
- [23] Oskar Perron. Zur Theorie der Matrices. *Math. Annalen*, 64:248–263, 1907.
- [24] Ron M. Roth. *Introduction to Coding Theory*. Cambridge University Press, 2006.
- [25] Arto Salomaa and Matti Siottola. *Automata-Theoretic Aspects of Formal Power Series*. Texts and Monographs in Computer Science. Springer-Verlag, 1978.

- [26] Eugene Senata. *Non-negative Matrices and Markov Chains*. Springer-Verlag, second edition, 2006.
- [27] Andrew J. Viterbi and Jim K. Omura. *Principles of Digital Communication*. Mc-Graw-Hill, 1979.