Algorithmen und Zufall

Jochen Geiger

20. Februar 2006

Contents

I Randomisierte Algorithmen

Ein *zufälliger* oder *randomisierter Algorithmus* verfügt über einen Vorrat an unabhängigen zufälligen Bits, die er verwenden kann um bei seiner Durchführung zufällige Entscheidung zu treffen. Die Vorzüge eines randomisierten Algorithmus sind im Allgemeinen kurze Laufzeit und geringer Speicherplatzbedarf. Ein zufälliger Algorithmus ist oft schneller oder braucht weniger Speicherplatz als der beste bekannte deterministische Algorithmus. Zudem ist er meist einfacher zu verstehen und zu implementieren. Häufig kann man einen naiven deterministischen Algorithmus mit fatalem worst case Verhalten durch Randomisieren in einen zufälligen Algorithmus verwandeln, der sich unabhängig von der Eingabe mit hoher Wahrscheinlichkeit gut verhält. Für besonders schwierige Probleme sind zufällige Algorithmen, die mit einer kleinen Fehlerwahrscheinlichkeit eine approximative Lösung liefern, oft gar die einzig bekannten effizienten Verfahren. Die Kunst des Entwerfens zufälliger Algorithmen besteht grundsätzlich darin, die Entscheidungen des Algorithmus so zu randomisieren, dass ungünstige Verläufe möglichst unwahrscheinlich sind.

Wir werden grundlegende Ideen und Methoden beim Entwurf und der Analyse zufälliger Algorithmen an Hand konkreter Beispiele kennen lernen. Vorab diskutieren wir einige Prinzipien, die der Konstruktion zufälliger Algorithmen typischerweise zugrunde liegen (nach R. Karp [14]).

Dem Widersacher einen Strich durch die Rechnung machen. Zum Verständnis der Vorzüge zufälliger Algorithmen ist eine spieltheoretische Sicht hilfreich. Bei der Komplexität eines Algorithmus kann man an den Wert eines Zwei-Personen Spiels denken, bei dem der eine Spieler den Algorithmus auswählt, und der andere Spieler (der Widersacher) versucht, eine Eingabe zu wählen, bei der der Algorithmus möglichst schlecht aussieht. Die Auszahlung des Widersachers sind die Kosten des Algorithmus (etwa die Laufzeit) bei der gewählten Eingabe. Eine natürliche Schwäche deterministischer Algorithmen ist, dass man relativ leicht eine einzelne Eingabe konstruieren kann, die den Algorithmus teuer zu stehen kommt. Ein zufälliger Algorithmus hingegen kann als Wahrscheinlichkeitsverteilung auf einer Menge von determistischen Algorithmen, also als eine gemischte Strategie, aufgefasst werden. Eine gemischte Strategie lässt den Widersacher im Ungewissen, wie sich der Algorithmus tatsächlich verhält. Dies erschwert dem Widersacher die Wahl einer Eingabe, die dem zufälligen Algorithmus Schwierigkeiten bereitet, denn dies muss eine Eingabe sein, die vielen deterministischen Algorithmen zugleich Schwierigkeiten bereitet.

Zufälliges Ziehen, Ordnen und Partitionieren. Ein zufälliger Algorithmus kann durch das Ziehen einer zufälligen Stichprobe Informationen über seine Eingabe gewinnen oder aber die Anordnung der Eingabe randomisieren, um ungünstige Anordnungen zu verhindern. Bei randomisierten *Divide & Conquer Algorithmen* ("Teile und Herrsche" Algorithmen) wird die Eingabe zufällig partitioniert.

Lastenverteilung. Stehen einem Algorithmus verschiedene Resourcen zur Verfügung, dann ist es oft hilfreich, die Lasten zufällig aufzuteilen.

Fülle von Zeugen. Soll ein Algorithmus herausfinden, ob die Eingabe eine gewisse Eigenschaft besitzt ("Ist n faktorisierbar?"), dann kann der Algorithmus diese Aufgabe lösen, indem er einen so genannten Zeugen findet. Auch wenn es schwer ist, einen Zeugen deterministisch zu konstruieren, kennt man manchmal eine (sehr große) Menge, in der diese Zeugen reichlich vorhanden sind. Dann kann man effizient nach einem Zeugen suchen, indem man wiederholt zufällig aus dieser Menge zieht. Besitzt die Eingabe die gefragte Eigenschaft, so wird mit großer Wahrscheinlichkeit ein Zeuge gefunden. Befindet sich kein Zeuge in der Stichprobe, so ist das ein starker Hinweis darauf, dass die Eingabe die gefragte Eigenschaft nicht besitzt.

Fingerabdrücke nehmen. Hier ist die Idee, ein komplexes Objekt durch einen kurzen "Fingerabdruck" zu repräsentieren; von welchem Finger der Abdruck genommen wird, das entscheidet der Zufall. Stimmen die Fingerabdrücke zweier Objekte überein, so auch mit hoher Wahrscheinlichkeit die beiden Objekte selbst.

1 Such- und Sortieralgorithmen

In diesem Kapitel behandeln wir einige ausgesuchte Such- und Sortieralgorithmen.

1.1 Der randomisierte Quicksort

Der Algorithmus *Quicksort* (Hoare [12], 1962) ist einer der einfachsten und meist benutzten Sortieralgorithmen. Das Verfahren arbeitet rekursiv nach dem Divide & Qonquer-Prinzip und ist ein sogenannter *in place* Algorithmus, das heißt er benötigt nur sehr wenig zusätzlichen Speicherplatz (anders als etwa der Merge Sort). Die randomisierte Version des Algorithmus verfährt wie folgt.

Randomisierter Quicksort

Eingabe: Eine Menge $\mathcal{S}$ von n paarweise verschiedenen Zahlen
Ausgabe: Die Elemente von $\mathcal{S}$ in aufsteigender Ordnung

1) Wähle rein zufällig ein Element Y aus $\mathcal{S}$.

2) Bestimme die Mengen $\mathcal{S}_<$ und $\mathcal{S}_>$, bestehend aus den Elementen, die kleiner bzw. größer als Y sind.

3) Ordne die Mengen $\mathcal{S}_<$ und $\mathcal{S}_>$ rekursiv.

4) Gebe die sortierte Menge $\mathcal{S}_<$ gefolgt von Y und der sortierten Menge $\mathcal{S}_>$ aus.

Es ist bei Sortieralgorithmen üblich, die Gesamtzahl der vom Algorithmus durchgeführten Vergleiche als Maß für die Laufzeit zu nehmen (bei vernünftiger Implementierung sind das die dominierenden Kosten). Bezeichne X_n die Anzahl der von Quicksort durchgeführten Vergleiche zwischen Elementen aus $\mathcal{S}$. Um Aussagen über die Laufzeit von Quicksort zu machen, müssen wir die Verteilung der Zufallsvariable X_n studieren. Man beachte, dass X_n Werte im Bereich $\mathbb{E}X_n$ bis $n^2/2$ annehmen kann. Ersteres ist der Fall, wenn stets der Median der zu sortierenden Teilmenge als Vergleichselement gewählt wird. Letzteres ist der Fall, wenn stets das kleinste (oder größte) Element gewählt wird. Der *worst case* ist ein großes Manko des deterministischen Quicksort. Wird etwa stets das erste Element der Liste als Vergleichselement gewählt, denn ergibt sich gerade bei vorsortierten Listen eine ungünstige Rekursionstiefe. Der folgende Satz besagt, dass beim randomisierten Quicksort ein ungünstiger Verlauf sehr unwahrscheinlich ist.

Satz 1.1 *Für alle $n \in \mathbb{N}$ gilt*

$$\mathbb{E}X_n \;=\; 2(n+1)H_n - 4n.$$

Dabei ist $H_n := \sum_{k=1}^{n} \frac{1}{k}$ die n-te harmonische Zahl. Insbesondere gilt

$$\lim_{n\to\infty} \frac{\mathbb{E}X_n}{n \log_2 n} = 1.386...$$

Beweis A. Wir schreiben X_n als Summe von Indikatorfunktionen,

$$X_n = \sum_{1\le i<j\le n} I_{A_{ij}},$$

wobei

$$A_{ij} = \{\text{es kommt zum Vergleich von } s_{(i)} \text{ und } s_{(j)}\}.$$

Dabei bezeichnet $s_{(i)}$ das Element in $\mathcal{S}$ vom Rang i (das i-kleinste Element). Für $i < j$ gilt

$$\mathbb{P}\{A_{ij}\} = \frac{2}{j-i+1},$$

denn

1) A_{ij} ist das Ereignis, dass $s_{(i)}$ oder $s_{(j)}$ als erstes der Elemente $s_{(i)}, s_{(i+1)}, \ldots, s_{(j)}$ als Vergleichselement ausgewählt wird.

2) Jedes der $j-i+1$ Elemente $s_{(i)}, s_{(i+1)}, \ldots, s_{(j)}$ wird mit gleicher Wahrscheinlichkeit zuerst als Vergleichselement ausgewählt.

Folglich gilt

$$\begin{aligned}
\mathbb{E}X_n &= \sum_{1\le i<j\le n} \mathbb{P}\{A_{ij}\} \\
&= \sum_{1\le i<j\le n} \frac{2}{j-i+1} \\
&= 2\sum_{k=1}^{n-1} \frac{n-k}{k+1} \\
&= 2(n+1)(H_n-1) - 2(n-1) \\
&= 2(n+1)H_n - 4n.
\end{aligned}$$

Beweis B. Die Verteilung der Zufallsvariable X_n erfüllt die folgende Rekursionsgleichung,

$$X_n \stackrel{d}{=} n-1 \;+\; X^{(1)}_{U_n-1} \;+\; X^{(2)}_{n-U_n}, \quad n \ge 2. \tag{1.1}$$

Hierbei ist U_n eine uniform auf $\{1,\dots,n\}$ verteilte Zufallsgröße, und $X_k^{(i)}$, $i = 1,2$ und $k = 0,\dots,n-1$ sind unabhängige Zufallsvariablen mit Verteilung $\mathcal{L}(X_k)$, unabhängig U_n. Die Notation $\stackrel{d}{=}$ bedeutet, dass die Verteilung der Zufallsgrößen übereinstimmt. Der Beitrag $n-1$ steht für die Anzahl der Vergleiche in Schritt 2 mit dem in Schritt 1 zufällig gewählten Element Y vom Rang U_n. Für das Ordnen der zufälligen Menge $\mathcal{S}_<$ benötigt der Algorithmus $X^{(1)}_{U_n-1}$ Vergleiche, für das Ordnen von $\mathcal{S}_>$ benötigt er $X^{(2)}_{n-U_n}$ Vergleiche.

Aus der Darstellung (1.1) ergibt sich mit dem Satz von der totalen Wahrscheinlichkeit, der Linearität des Erwartungswertes und der Unabhängigkeit der $X_k^{(i)}$ und U_n, dass

$$\begin{aligned}
\mathbb{E}X_n &= n-1 \;+\; \sum_{k=1}^{n} \mathbb{E}\Big[X^{(1)}_{U_n-1} \big|\, U_n = k\Big]\, \mathbb{P}\{U_n = k\} \\
&\qquad + \sum_{k=1}^{n} \mathbb{E}\Big[X^{(2)}_{n-U_n} \big|\, U_n = k\Big]\, \mathbb{P}\{U_n = k\} \\
&= n-1 \;+\; \frac{1}{n}\sum_{k=1}^{n} \Big(\mathbb{E}X^{(1)}_{k-1} + \mathbb{E}X^{(2)}_{n-k}\Big) \\
&= n-1 \;+\; \frac{2}{n}\sum_{k=0}^{n-1} \mathbb{E}X_k, \quad n \ge 2
\end{aligned}$$

mit $\mathbb{E}X_0 = \mathbb{E}X_1 = 0$. Es folgt

$$\begin{aligned}
\mathbb{E}X_n &= \frac{2}{n}\,\mathbb{E}X_{n-1} \;+\; \frac{n-1}{n}\Big(\frac{2}{n-1}\sum_{k=0}^{n-2}\mathbb{E}X_k \;+\; n-2\Big) \\
&\qquad + n-1 \;-\; \frac{(n-1)(n-2)}{n} \\
&= \frac{n+1}{n}\,\mathbb{E}X_{n-1} \;+\; \frac{2(n-1)}{n}
\end{aligned}$$

und somit

$$\begin{aligned}\frac{\mathbb{E}X_n}{n+1} &= \frac{\mathbb{E}X_{n-1}}{n} + \frac{2(n-1)}{n(n+1)} \\ &= \sum_{k=2}^{n} \frac{2(k-1)}{k(k+1)} \\ &= 2\sum_{k=1}^{n}\left(\frac{2}{k+1} - \frac{1}{k}\right) \\ &= 2\left(H_n + \frac{2}{n+1} - 2\right) \;=\; 2H_n - \frac{4n}{n+1}.\end{aligned}$$

Bemerkungen.

- Die erwartete Laufzeit ist für alle Eingaben gleich, denn die Verteilung von X_n hängt nicht von der Eingabe ab. Alternativ könnte man die Menge $\mathcal{S}$ zunächst rein zufällig permutieren und dann stets das erste (oder das letzte) Element als Vergleichselement auswählen.

- Wir haben lediglich die erwartete Laufzeit abgeschätzt. Man kann aber zeigen (vgl. [24]), dass für beliebige $\lambda, \varepsilon > 0$ Konstanten $c_{\lambda,\varepsilon}$ existieren, so dass

  $$\mathbb{P}\{X_n \geq (1+\varepsilon)\,\mathbb{E}X_n\} \leq\; c_{\lambda,\varepsilon} n^{-\lambda\varepsilon},\; n \in \mathbb{N}.$$

  Die Wahrscheinlichkeit für ungünstiges Verhalten von Quicksort wird für großes n also verschwindend klein (von beliebiger polynomialer Ordnung). Wir werden später sehen, dass man einen Algorithmus stets so modifizieren kann, dass der Schwanz der Laufzeitverteilung des modifizierten Algorithmus exponentiell abfällt und die erwartete Laufzeit, wenn überhaupt, nur unmerklich erhöht wird (siehe Kapitel 2).

- Man kann den randomisierten Quicksort noch verbessern, indem man etwa den Median dreier rein zufällig gewählter Elemente als Vergleichselement nimmt. Dadurch wird die Wahrscheinlichkeit einer ungünstigen Rekursionstiefe weiter verringert.

Ein enger Verwandter des randomisierten Quicksort ist der Suchalgorithmus *Find*, der das Element vom Rang k einer Menge $\mathcal{S}$ bestimmt. Auch der Algorithmus Find wählt zunächst rein zufällig ein Element Y aus $\mathcal{S}$ und zerlegt $\mathcal{S}\setminus\{Y\}$ in die Mengen $\mathcal{S}_<$ und $\mathcal{S}_>$ der Elemente, die kleiner bzw. größer als Y sind. Ist $|\mathcal{S}_<| = k-1$, so ist Y das gesuchte Element. Ist $|\mathcal{S}_<| \neq k-1$, dann bestimmt der Algorithmus rekursiv das Element vom Rang k in $\mathcal{S}_<$, falls $|\mathcal{S}_<| \geq k$, bzw. das Element vom Rang $k-|\mathcal{S}_<|-1$ in $\mathcal{S}_>$, falls $|\mathcal{S}_<| < k-1$. Unter Ausnutzung der rekursiven Struktur von Find zeigt man

Satz 1.2 *Sei $X_{n,k}$ die Anzahl der vom Algorithmus Find bei der Bestimmung des Elements vom Rang k in einer n-elementigen Menge $\mathcal{S}$ durchgeführten Vergleiche. Dann gilt*

$$\max_{1\leq k\leq n} \mathbb{E}X_{n,k} \;\leq\; 4n,\; n \in \mathbb{N}.$$

Die Laufzeit von Find besitzt Momente beliebiger Ordnung. Genauer gilt (vgl. [5]),

$$\mathbb{E}X_{n,k}^j \le c_j n^j,\ j \in \mathbb{N}.$$

Dabei kennt man explizite obere Schranken für die c_j.

1.2 Der Lazy Select Algorithmus

Wie Find bestimmt auch der Algorithmus *Lazy Select* (Floyd und Rivest [7], 1975) das Element $s_{(k)}$ in einer Menge $\mathcal{S} = \{s_1, \ldots, s_n\}$ von n paarweise verschiedenen Zahlen. Die dem Lazy Select zu Grunde liegende Idee ist, durch Ziehen einer Stichprobe $R = (R_1, \ldots, R_N)$ von Elementen aus $\mathcal{S}$ ein Intervall I zu bestimmen, so dass mit großer Wahrscheinlichkeit,

- das Intervall I das gesuchte Element $s_{(k)}$ überdeckt;
- der Schnitt von I mit $\mathcal{S}$ relativ klein, also kostengünstig zu sortieren ist.

Den Rang von $s_{(k)}$ in der Menge (bzw. des Multisets) $R \cup \{s_{(k)}\}$ wird man in der Nähe von $x := k\frac{N}{n}$ erwarten. Ein natürlicher Kandidat für das zufällige Intervall $I \subset \mathbb{R}$ ist daher $[R_{(\lfloor x \rfloor - m)}, R_{(\lceil x \rceil + m)}]$, wobei die Parameter N und m noch geeignet zu wählen sind. (Mit $R_{(j)}$ bezeichnen wir das Element vom Rang j in der Stichprobe R.) In der folgenden Version des Lazy Select sind die Parameter N und m universell eingestellt. Hat man eine bestimmte Größenordnung von k im Sinn (etwa $k_n = \lfloor \frac{n+1}{2} \rfloor$, also $s_{(k_n)} =$ Median von $\mathcal{S}$), dann empfiehlt es sich, die Wahl der Parameter spezifisch anzupassen.

Lazy Select

Eingabe: Eine Menge $\mathcal{S}$ von n paarweise verschiedenen Zahlen und ein $k \in \{1, \ldots, n\}$
Ausgabe: Das Element $s_{(k)}$ vom Rang k in $\mathcal{S}$

1) Ziehe mit Zurücklegen eine Stichprobe R vom Umfang $N = \lfloor n^{\frac{3}{4}} \rfloor$ aus $\mathcal{S}$.

2) Sortiere die Stichprobe R unter Benutzung eines optimalen Sortieralgorithmus.

3) Setze $x := kn^{-\frac{1}{4}},\ \ell := \max(\lfloor x - \sqrt{n} \rfloor, 1),\ h := \min(\lceil x + \sqrt{n}\, \rceil, n)$ und

$$Q := \begin{cases} \mathcal{S} \cap (-\infty, R_{(h)}], & \text{falls } k < n^{\frac{1}{4}} \log n; \\ \mathcal{S} \cap [R_{(\ell)}, R_{(h)}], & \text{falls } k \in [n^{\frac{1}{4}} \log n, n - n^{\frac{1}{4}} \log n]; \\ \mathcal{S} \cap [R_{(\ell)}, \infty), & \text{falls } k > n - n^{\frac{1}{4}} \log n. \end{cases} \tag{1.2}$$

4) Bestimme die Ränge von $R_{(\ell)}$ und $R_{(h)}$ in $\mathcal{S}$ durch Vergleich aller Elemente aus $\mathcal{S}$ mit $R_{(\ell)}$ und $R_{(h)}$.

5) Überprüfe, ob $s_{(k)} \in Q$ (durch Vergleich von k mit den Rängen von $R_{(\ell)}$ und $R_{(h)}$ in $\mathcal{S}$) und ob $|Q| \le 4n^{\frac{3}{4}} + 2$. Falls nicht, wiederhole die Schritte 1 bis 5.

6) Sortiere die Menge Q und identifiziere das Element $s_{(k)}$ als das Element aus Q mit Rang $k - \mathrm{Rang}_{\mathcal{S}}(R_{(\ell)}) + 1$, falls $k \geq n^{\frac{1}{4}} \log n$, bzw. als jenes mit Rang k, falls $k < n^{\frac{1}{4}} \log n$.

Satz 1.3 *Der Algorithmus Lazy Select findet das Element $s_{(k)}$ mit Wahrscheinlichkeit $1 - O(n^{-\frac{1}{4}})$ bei einmaligem Durchlaufen der Schritte 1 bis 5. Er führt dabei maximal $2n + o(n)$ Vergleiche durch.*

Beweis. Für das Sortieren der Stichprobe R bzw. der Menge Q in den Schritten 2 und 6 benötigt ein optimaler Sortieralgorithmus wie etwa der Heapsort $O(n^{\frac{3}{4}} \log n)$ Vergleiche. Die Kosten von Lazy Select werden also durch die maximal $2(n-2)$ Vergleiche in Schritt 4 dominiert.

Betrachten wir nun das Ereignis, dass Lazy Select das Element $s_{(k)}$ nicht bei erstmaligem Durchlaufen der Schritte 1 bis 5 findet. Wir beschränken uns auf den Fall $k \in [n^{\frac{1}{4}} \log n, n - n^{\frac{1}{4}} \log n]$, die beiden anderen Fälle behandelt man analog. Es gibt zwei Möglichkeiten für das Scheitern des Algorithmus beim Durchlaufen der Schritte 1 bis 5: Das gesuchte Element $s_{(k)}$ ist nicht in der Menge Q oder aber die Menge Q ist zu groß. Wir betrachten zunächst das Ereignis, dass $s_{(k)} \notin Q$, also $s_{(k)} < R_{(\ell)}$ oder $s_{(k)} > R_{(h)}$. Sei $Y = Y_{n,k,N}$ die Anzahl der Elemente in der Stichprobe, die kleiner als $s_{(k)}$ sind,

$$Y := \sum_{i=1}^{N} I_{\{R_i \leq s_{(k)}\}}.$$

Dabei bezeichne R_i das i-te Element der Stichprobe in der Reihenfolge der Ziehung. Offensichtlich gilt

$$\{R_{(\ell)} > s_{(k)}\} \;=\; \{Y \leq \ell - 1\}. \tag{1.3}$$

Die Zufallsvariable Y ist binomialverteilt zum Parameter $(N, \frac{k}{n})$. Folglich ist

$$\mathbb{E}Y = N\frac{k}{n} = x \quad \text{und} \quad \mathbb{V}\mathrm{ar}\, Y = \lfloor n^{\frac{3}{4}} \rfloor \tfrac{k}{n}(1 - \tfrac{k}{n}) \leq \tfrac{1}{4} n^{\frac{3}{4}}.$$

Im Fall $\ell = \lfloor x - \sqrt{n} \rfloor > 1$ erhält man nun mit Hilfe der Chebyshev Ungleichung

$$\begin{aligned} \mathbb{P}\{s_{(k)} < R_{(\ell)}\} \;&=\; \mathbb{P}\{Y < \ell\} \\ &\leq\; \mathbb{P}\{|Y - \mathbb{E}Y| > \sqrt{n}\} \;\leq\; n^{-1}\, \mathbb{V}\mathrm{ar}\, Y \;\leq\; \tfrac{1}{4} n^{-\frac{1}{4}}. \end{aligned}$$

Ist $\ell = 1$, dann ergibt sich die Abschätzung

$$\begin{aligned} \mathbb{P}\{s_{(k)} < R_{(\ell)}\} \;&=\; \mathbb{P}\{Y = 0\} \;=\; (1 - \tfrac{k}{n})^{n^{\frac{3}{4}}} \\ &\leq\; (1 - n^{-\frac{3}{4}} \log n)^{n^{\frac{3}{4}}} \;\leq\; \exp(-\log n) \;=\; n^{-1}. \end{aligned}$$

Völlig analog zeigt man, dass $\mathbb{P}\{s_{(k)} > R_{(h)}\} = O(n^{-\frac{1}{4}})$.

Betrachten wir nun die zweite Möglichkeit des Scheiterns, also das Ereignis, dass Q mehr als $4n^{\frac{3}{4}} + 2$ Elemente enthält. Offensichtlich gilt für alle i und j,

$$\{|Q| > i + 2\} \;=\; \{|\mathcal{S} \cap [R_{(\ell)}, R_{(h)}]| > i + 2\} \;\subset\; \{R_{(\ell)} \leq s_{(j)}\} \cup \{R_{(h)} \geq s_{(j+i+2)}\}.$$

Insbesondere gilt also

$$\mathbb{P}\{|Q| > 4n^{\frac{3}{4}} + 2\} \;\leq\; \mathbb{P}\{R_{(\ell)} \leq s_{(k_\ell)}\} + \mathbb{P}\{R_{(h)} \geq s_{(k_h)}\},$$

wobei $k_\ell := \max(\lfloor k - 2n^{\frac{3}{4}} \rfloor, 1)$ und $k_h := \min(\lceil k + 2n^{\frac{3}{4}} \rceil, n)$, Nun zeigt man analog zur Abschätzung des Scheiterns der ersten Art, dass beide Terme höchstens von der Größenordnung $n^{-\frac{1}{4}}$ sind. Dabei übernehmen k_ℓ und k_h die Rolle von k.

Bemerkungen.

- Die in Satz 1.3 angegebene obere Schranke ist offensichtlich nicht die wahre Größenordnung für die Wahrscheinlichkeit des Scheiterns beim erstmaligen Durchlaufen der Schritte 1 bis 5. (In Kapitel 4 werden wir feinere Abschätzungen als die Chebyshev Ungleichung kennen lernen.)

- Will man die Wahrscheinlichkeit für ein Scheitern bei Durchlaufen der Schritte 1 bis 5 durch Änderung der Parameter N und m verringern, so geht das zu Lasten der Laufzeit und schlägt sich entsprechend in dem $o(n)$ Term nieder.

- Offensichtlich braucht in Schritt 4 ein Element, welches kleiner als $R_{(\ell)}$ (größer als $R_{(h)}$) ist, nicht mehr mit $R_{(h)}$ (mit $R_{(\ell)}$) verglichen zu werden. Ist $k \leq \frac{n}{2}$, dann empfiehlt es sich, die Elemente aus $\mathcal{S}$ zunächst mit $R_{(h)}$ zu vergleichen, ansonsten zunächst mit $R_{(\ell)}$. Die erwartete Anzahl der von Lazy Select durchgeführten Vergleiche ist dann durch $\frac{3}{2}n + o(n)$ beschränkt. Ist k bzw. $n - k$ von kleinerer Ordnung als n, so ist die erwartete Anzahl der Vergleiche sogar nur $n + o(n)$. Mit der Idee, einen Algorithmus bei ungünstigem Verlauf abzubrechen und neu zu starten (wie am Ende von Schritt 5 des Lazy Select), werden wir uns ausführlich im nächsten Kapitel beschäftigen.

- Die besten bekannten deterministischen Algorithmen benötigen $3n$ Vergleiche als worst case und ihre Implementierung ist relativ kompliziert. Für die Anzahl der durchzuführenden Vergleiche zur Bestimmung des Medians kennt man eine theoretische untere Schranke von $2n$.

- Das Ziehen mit Zurücklegen ist nicht eben wirtschaftlich. Es ist aber einfacher und platzsparender zu implementieren, da man sich die bereits gezogenen Elemente nicht merken muss. Wir haben diese Annahme hier vor allem der übersichtlicheren Analysis zuliebe gemacht.

2 Das Beschleunigen von Las-Vegas-Algorithmen

Ein randomisierter Algorithmus, der (wenn er terminiert) stets eine korrekte Lösung ausgibt, heißt *Las-Vegas-Algorithmus*. Einen Algorithmus, der mit positiver Wahrscheinlichkeit eine falsche Lösung liefert, nennt man hingegen einen *Monte-Carlo-Algorithmus*. (Alle bisher betrachteten Algorithmen waren Las-Vegas-Algorithmen, in den Kapiteln 5 und 8 werden wir aber auch Monte-Carlo-Algorithmen kennen lernen.)

Betrachten wir einen Las-Vegas-Algorithmus mit der folgenden (zugegebenermaßen etwas degenerierten) Laufzeitverteilung. Der Algorithmus liefert mit Wahrscheinlichkeit p nach genau k Schritten eine Lösung, mit Wahrscheinlichkeit $1-p$ gerät er hingegen in eine Endlosschleife. Bei Kenntnis dieses Laufzeitverhaltens ist man sicherlich gut beraten, den Algorithmus gegebenenfalls, d.h. bei nicht erfolgter Ausgabe einer Lösung nach k Schritten, abzubrechen und neu zu starten. Dies ist bereits die grundlegende Idee zur Beschleunigung von Las-Vegas-Algorithmen [16]. Um zu vermeiden, dass einem der Zufall besonders übel mitspielt, startet man den Algorithmus nach k Schritten jeweils unabhängig neu, bis schließlich ein Durchlauf mit der Ausgabe einer Lösung terminiert. Die Anzahl der Starts ist dann eine geometrisch verteilte Zufallsvariable mit Erfolgsparameter $\mathbb{P}\{X \leq k\}$. Dabei ist X die zufällige Laufzeit des Las-Vegas-Algorithmus. Bezeichnet Y_k die Laufzeit des modifizierten Verfahrens, dann wird Y_k/k von der Gesamtzahl der Starts stochastisch dominiert. Es gilt also

$$\mathbb{E}Y_k \leq \frac{k}{\mathbb{P}\{X \leq k\}}$$

und

$$\mathbb{P}\{Y_k > n\} \;\leq\; (\mathbb{P}\{X > k\})^{\lfloor n/k \rfloor} \;\leq\; c\,\theta_k^n,\; n \geq 0,$$

mit $\theta_k = (\mathbb{P}\{X > k\})^{\frac{1}{k}}$. Selbst wenn also die Laufzeit eines randomisierten Algorithmus mit nur kleiner Wahrscheinlichkeit gering ist, kann man den Algorithmus stets so modifizieren, dass ungünstige Verläufe extrem unwahrscheinlich werden. Insbesondere ist dies bei Algorithmen mit geringer erwarteter Laufzeit der Fall. Wir werden sehen, dass die obige Strategie mit geeignetem von der Verteilung abhängenden k in gewissem Sinn optimal ist.

In der Praxis kennt man die genaue Verteilung der Laufzeit eines Algorithmus im Allgemeinen jedoch nicht, manchmal kennt man nicht einmal eine obere Schranke für die Mindestschrittzahl. Auch hängt die Verteilung der Laufzeit möglicherweise stark von der Eingabe ab. In solchen Fällen kann man nach einer universellen Strategie verfahren, bei der die Abbruchzeiten von Lauf zu Lauf variieren und deren erwartete Laufzeit die erwartete Laufzeit der spezifischen optimalen Strategie nur um einen logarithmischen Faktor übersteigt.

2.1 Die optimale Strategie bei bekannter Verteilung der Laufzeit

Wir betrachten etwas allgemeiner Strategien in denen die Schrittzahl bis zum nächsten Neustart variieren kann. Eine solche Strategie wird durch einen Folge $(k_i)_{i\geq 1} \in (\mathbb{N} \cup \{\infty\})^{\mathbb{N}}$ beschrieben. Terminiert der Algorithmus nicht spätestens nach k_1 Schritten, so wird er erneut und dieses Mal für höchstens k_2 Schritte gestartet. Liefert er auch beim zweiten Versuch keine

Lösung, dann wird er ein drittes Mal für maximal k_3 Schritte gestartet, etc. Wir bezeichnen die Laufzeit dieses Verfahrens mit $Y_{(k_i)}$ und schreiben $Y_{(k_i)} = Y_k$, falls $k_i = k$ für alle $i \geq 1$. Man beachte, dass $Y_\infty \stackrel{d}{=} X$.

Lemma 2.1 *Für $k \in \mathbb{N} \cup \{\infty\}$ gilt*

$$\mathbb{E}Y_k \;=\; \frac{\mathbb{E}[\min(X,k)]}{\mathbb{P}\{X \leq k\}}. \tag{2.1}$$

Beweis. Sei $k_{\min} := \min\{j \geq 1 : \mathbb{P}\{X = j\} > 0\}$ die Mindestlaufzeit des ursprünglichen Algorithmus. Für $k < k_{\min}$ und $k = \infty$ ist die Behauptung (2.1) offensichtlich.

Sei also $k_{\min} \leq k < \infty$. Dann gilt

$$Y_k \;=\; k(N_k - 1) + L_k,$$

wobei N_k die Gesamtzahl der Starts und L_k die Anzahl der Schritte vom letztmaligen Neustart bis zur Ausgabe der Lösung bezeichnet. Nach Konstruktion sind die Zufallsvariablen N_k und L_k unabhängig mit

$$\mathbb{P}\{N_k = n\} = \mathbb{P}\{X \leq k\}(\mathbb{P}\{X > k\})^{n-1},\ n \geq 1$$

und $\mathcal{L}(L_k) = \mathcal{L}(X|X \leq k)$. Insbesondere folgt mit dem Satz von der totalen Wahrscheinlichkeit

$$\begin{aligned}
\mathbb{P}\{X \leq k\}\,\mathbb{E}Y_k &= \mathbb{P}\{X \leq k\}(k\,\mathbb{E}[N_k - 1] \,+\, \mathbb{E}L_k) \\
&= k\,\mathbb{P}\{X > k\} + \mathbb{P}\{X \leq k\}\,\mathbb{E}[X \,|\, X \leq k] \\
&= \mathbb{E}[\,\min(X,k) \,|\, X > k\,]\,\mathbb{P}\{X > k\} + \mathbb{E}[\,\min(X,k) \,|\, X \leq k\,]\,\mathbb{P}\{X \leq k\} \\
&= \mathbb{E}[\min(X,k)]. \qquad \square
\end{aligned}$$

Sei nun A die Menge der k mit minimalem $\mathbb{E}Y_k =: \alpha(k)$,

$$A := \{1 \leq k \leq \infty : \alpha(k) = \alpha^*\},\ \text{ wobei } \alpha^* = \inf_{1 \leq j \leq \infty} \alpha(j).$$

Wegen

$$\lim_{k \to \infty} \alpha(k) \;=\; \frac{\mathbb{E}X}{\mathbb{P}\{X < \infty\}} \;=\; \mathbb{E}X \;=\; \alpha(\infty) \tag{2.2}$$

ist die Menge A nicht leer. Hat X eine langschwänzige Verteilung, also etwa $\mathbb{E}X = \infty$, dann wird man die Läufe nach nicht allzu langer Zeit abbrechen (das oder die Elemente in A sind klein). Hat die Verteilung hingegen einen sehr kurzen Schwanz, dann gibt es keinen Grund für einen Abbruch ($A = \{\infty\}$). Der Grenzfall ist die geometrische Verteilung, hier gilt $A = \mathbb{N} \cup \{\infty\}$.

Wir nennen im Folgenden eine Strategie $(k_i)_{i \geq 1}$ *optimal*, wenn sie die erwartete Laufzeit minimiert, wenn also $\mathbb{E}Y_{(k_i)} = \inf_{(\ell_i)} \mathbb{E}Y_{(\ell_i)}$.

Satz 2.2 *Gilt $k_i \in A$ für alle $i \geq 1$, dann ist die Strategie $(k_i)_{i\geq 1}$ optimal.*

Beweis. Sei $(k_i)_{i\geq 1}$ eine Strategie mit $k_i \geq k_{\min}$ für alle $i \geq 1$. Aus der Zerlegung nach dem Ergebnis des ersten Durchlaufs erhält man mit Lemma 2.1

$$\begin{aligned}
\mathbb{E}Y_{(k_i)} &= \mathbb{E}[\min(Y_{(k_i)}, k_1)] + \mathbb{E}(Y_{(k_i)} - k_1)^+ \\
&= \mathbb{E}[\min(X, k_1)] + \mathbb{E}[(Y_{(k_i)} - k_1)^+ \,|\, Y_{(k_i)} > k_1]\, \mathbb{P}\{Y_{(k_i)} > k_1\} \\
&= \mathbb{E}[\min(X, k_1)] + \mathbb{P}\{X > k_1\}\, \mathbb{E}Y_{(k_{i+1})} \\
&= \mathbb{P}\{X \leq k_1\}\, \alpha(k_1) + \mathbb{P}\{X > k_1\}\, \mathbb{E}Y_{(k_{i+1})} \\
&= \sum_{j=1}^{\infty} p_j\, \alpha(k_j), \qquad (2.3)
\end{aligned}$$

wobei

$$p_j = \mathbb{P}\{X \leq k_j\} \prod_{i=1}^{j-1} \mathbb{P}\{X > k_i\}.$$

Man beachte, dass p_j die Wahrscheinlichkeit für genau $j-1$ Neustarts ist. Auf Grund der Annahme $k_i \geq k_{\min}$ gilt $\sum_{j=1}^{\infty} p_j = 1$. Aus (2.3) folgt dann unmittelbar, dass

$$\mathbb{E}Y_{(k_i)} \;\geq\; \inf_j \alpha(k_j) \;\geq\; \alpha^*.$$

Diese untere Schranke gilt für die erwartete Laufzeit beliebiger Strategien, da eine Strategie durch Entfernen derjenigen k_i mit $k_i < k_{\min}$ offensichtlich nicht schlechter wird. Umgekehrt ist nach (2.3) die erwartete Laufzeit der in Satz 2.2 beschriebenen Strategien gleich α^*. Folglich sind solche Strategien optimal.

Bemerkungen. Satz 2.2 gewährleistet insbesondere, dass man sich auf konstante Strategien Y_k mit $k \in A$ beschränken kann. Der Gleichung (2.3) entnimmt man die folgende Umkehrung zu Satz 2.2: Ist die Strategie $(k_i)_{i\geq 1}$ optimal, dann gilt $k_i \in A$ für alle $i \leq \sup\{j : p_j > 0\}$.

Wir betrachten nun eine größere Klasse von Strategien. Wir nehmen an, dass Läufe unterbrochen und später an der entsprechenden Stelle fortgesetzt werden können. Eine solche Strategie wird durch eine Folge $(m_i, k_i)_{i\geq 1}$ beschrieben. Hierbei ist m_i die "Nummer" des Laufs auf "Stufe" i und k_i die maximale Schrittzahl dieser Fortsetzung.

Satz 2.3 *Die Strategien aus Satz 2.2 sind auch innerhalb der oben beschriebenen Klasse von Strategien optimal: Für alle Folgen $(m_i, k_i)_{i\geq 1}$ gilt*

$$\mathbb{E}Y_{(m_i,k_i)} \;\geq\; \alpha^*. \qquad (2.4)$$

Die Aussage von Satz 2.3 lässt sich wie folgt einsehen. Auf Grund der Unabhängigkeit der Läufe kann eine Strategie $(m_i, k_i)_{i\geq 1}$ für jeden Lauf getrennt betrachtet werden (man gewinnt durch das Starten bzw. Fortsetzen von Läufen mit anderer Startnummer keinerlei Information

über den Fortgang eines Laufes). Für einen einzelnen Lauf gibt es aber keinen Grund von der optimalen Strategie aus Satz 2.2 abzuweichen, also etwa den Lauf bereits nach $j < k \in A$ Schritten zu unterbrechen oder über $j \geq k \in A$ Schritte hinaus fortzuführen.

Der formale Nachweis von (2.4) ist etwas mühsam, weil man die Rekursion (2.3) nicht mehr zur Verfügung hat.

Beweis. Fixiere eine beliebige Folge $(m_i, k_i)_{i\geq 1}$. Wir zeigen wiederum, dass $\mathbb{E}Y$ eine Konvexkombination der $\alpha(j)$ ist, dass also $\mathbb{E}Y = \sum_{j=1}^{\infty} \lambda_j \alpha(j)$ mit $\lambda_j \geq 0$ und $\sum \lambda_j = 1$. Setze dazu $n_0 = 0$ und $n_i = \sum_{j=1}^{i} k_j$, $i \geq 1$. Mit ℓ_i bezeichnen wir die maximale Gesamtschrittzahl des Laufs mit der Nummer m_i vor der Fortsetzung auf Stufe i (also $\ell_i := \sum_{j=1}^{i-1} k_j 1_{\{m_j = m_i\}}$). Wir zerlegen $Y = Y_{(m_i,k_i)}$ wie folgt,

$$Y = \sum_{i=1}^{\infty} \min\Big((Y - n_{i-1})^+, k_i\Big).$$

Mit dem Satz von der totalen Wahrscheinlichkeit folgt

$$\begin{aligned}
\mathbb{E}Y &= \sum_{i=1}^{\infty} \mathbb{E}[\min((Y - n_{i-1})^+, k_i)] \\
&= \sum_{i=1}^{\infty} \mathbb{P}\{Y > n_{i-1}\}\, \mathbb{E}[\min((Y - n_{i-1})^+, k_i) \,|\, Y > n_{i-1}] \\
&= \sum_{i=1}^{\infty} \mathbb{P}\{Y > n_{i-1}\}\, \mathbb{E}[\min(X - \ell_i, k_i) \,|\, X > \ell_i] \\
&= \sum_{i=1}^{\infty} \mathbb{P}\{Y > n_{i-1}\}\, \frac{\mathbb{E}[\min(X - \ell_i)^+, k_i)]}{\mathbb{P}\{X > \ell_i\}} \\
&= \sum_{i=1}^{\infty} \frac{\mathbb{P}\{Y > n_{i-1}\}}{\mathbb{P}\{X > \ell_i\}} \Big(\mathbb{E}[\min(X, \ell_i + k_i)] \,-\, \mathbb{E}[\min(X, \ell_i)]\Big). \qquad (2.5)
\end{aligned}$$

Ohne Einschränkung sei $\ell_i + k_i \geq k_{\min}$, und $\ell_i = 0$ oder $\ell_i \geq k_{\min}$ (eine Strategie ohne diese Eigenschaften ist offensichtlich leicht zu verbessern). Mit der Konvention $0 \cdot \alpha(0) = 0$ ist der i-te Summand auf der rechten Seite von (2.5) nach Lemma 2.1 gleich

$$\begin{aligned}
&\frac{\mathbb{P}\{Y > n_{i-1}\}}{\mathbb{P}\{X > \ell_i\}} \Big(\mathbb{P}\{X \leq \ell_i + k_i\}\, \alpha(\ell_i + k_i) \,-\, \mathbb{P}\{X \leq \ell_i\}\, \alpha(\ell_i)\Big) \\
&\qquad =: \; q_i^{(1)} \alpha(\ell_i + k_i) \,-\, q_i^{(2)} \alpha(\ell_i).
\end{aligned}$$

Folglich gilt $\mathbb{E}Y = \sum_{j=1}^{\infty} \lambda_j \alpha(j)$, wobei

$$\lambda_j = \sum_{\{i:\, \ell_i + k_i = j\}} q_i^{(1)} - \sum_{\{i:\, \ell_i = j\}} q_i^{(2)}, \quad j \geq 1.$$

Wegen

$$q_i^{(1)} - q_i^{(2)} \;=\; \mathbb{P}\{Y > n_{i-1}\}\, \mathbb{P}\{X \leq \ell_i + k_i \,|\, X > \ell_i\} \;=\; \mathbb{P}\{n_{i-1} < Y \leq n_i\},$$

erhält man

$$\sum_{j=1}^{\infty} \lambda_j \;=\; \sum_{(i,j)\,:\,\ell_i+k_i=j} q_i^{(1)} - \sum_{(i,j)\,:\,\ell_i=j} q_i^{(2)} \;=\; \sum_{i=1}^{\infty} \Big(q_i^{(1)} - q_i^{(2)} \Big) \;=\; 1.$$

Hinsichtlich der Nichtnegativität der λ_j beachte man, dass $q_i^{(2)} = 0$, falls $\ell_i = 0$, wenn also der Lauf m_i auf Stufe i erstmalig gestartet wird. Bezeichne andernfalls i' die Stufe der jüngsten Fortsetzung des Laufs mit der Startnummer m_i vor Stufe i. Dann gilt $\ell_{i'} + k_{i'} = \ell_i$, und

$$\frac{q_{i'}^{(1)} - q_i^{(2)}}{\mathbb{P}\{X \le \ell_i\}} \;=\; \frac{\mathbb{P}\{Y > n_{i'-1}\}}{\mathbb{P}\{X > \ell_{i'}\}} \;-\; \frac{\mathbb{P}\{Y > n_{i-1}\}}{\mathbb{P}\{X > \ell_i\}} \;\ge\; 0,$$

denn die beiden Quotienten sind die bedingten Wahrscheinlichkeiten der Ereignisse $\{Y > n_{i'-1}\}$ bzw. $\{Y > n_{i-1}\}$ gegeben, dass der Lauf mit Startnummer m_i nicht vor Erreichen der Stufen i' bzw. i terminiert. Es folgt

$$\lambda_j \;=\; \sum_{i=1}^{\infty} \Big(q_i^{(1)} - q_i^{(2)} \Big) \;\ge\; \sum_{\{i\,:\,\ell_i>0\}} \Big(q_{i'}^{(1)} - q_i^{(2)} \Big) \;\ge\; 0$$

für alle $j \ge 1$ und Satz 2.3 ist bewiesen.

Bemerkung. Man beachte, dass auch eine zufällige Wahl der $(m_i, k_i)_{i\ge1}$ keinen Vorteil bringt, wenn die Wahl von (m_j, k_j) nur vom Verlauf der Stufen 1 bis $j-1$ abhängen darf.

2.2 Strategien bei unbekannter Verteilung der Laufzeit

In der Praxis wird man die Menge A zumeist nicht kennen. Es stellt sich daher die Frage, welche erwartete Laufzeit man ohne irgendwelche Informationen über die Verteilung der Laufzeit des Algorithmus erreichen kann. Wir werden universelle Strategien kennen lernen, die keinerlei Kenntnisse über die Verteilung von X zugrunde legen und deren erwartete Laufzeit dennoch nur um einen logarithmischen Faktor von der erwarteten Laufzeit der spezifischen optimalen Strategie abweicht.

Sei $(k_i)_{i\ge1}$ eine Folge von Zweier-Potenzen mit

$$|\{1 \le i \le 2^m - 1 : k_i = 2^j\}| = 2^{m-1-j} \;\text{ für alle }\; 0 \le j \le m-1 < \infty, \tag{2.6}$$

also etwa $(1,1,2,1,1,2,4,1,1,1,1,2,2,4,8,\dots)$. Man beachte, dass bei nicht erfolgter Ausgabe einer Lösung bis zur Stufe 2^m-1 ein solches Verfahren jeweils 2^{m-1} Schritte mit Läufen der Länge 2^j, $0 \le j \le m-1$, verbracht hat ("ausbalancierte" Strategie). Insbesondere gilt

$$n_{2^m-1} \;=\; \sum_{i=1}^{2^m-1} k_i \;=\; \sum_{j=0}^{m-1} 2^j 2^{m-j-1} \;=\; m2^{m-1}, \;\; m \ge 1. \tag{2.7}$$

Satz 2.4 *Sei $(k_i)_{i\ge1}$ eine Strategie mit der Eigenschaft (2.6). Dann gibt es von der Laufzeitverteilung $\mathcal{L}(X)$ unabhängige Konstanten c_1 und c_2, so dass*

$$\mathbb{E}Y_{(k_i)} \;\le\; c_1\alpha^*(\log_2\alpha^* + c_2). \tag{2.8}$$

Zum Beweis von Satz 2.4 erweist sich die Einführung der folgenden Größe als nützlich,

$$\beta^* := \inf_{k\geq 1} \beta(k), \quad \text{wobei } \beta(k) = \frac{k}{\mathbb{P}\{X \leq k\}},\ k \in \mathbb{N}.$$

Man beachte, dass das Infimum angenommen wird, da $\lim_{k\to\infty} \beta(k) = \infty$. Das folgende Lemma besagt, dass der Quotient β^*/α^* gleichmäßig beschränkt ist.

Lemma 2.5 *Unabhängig von der Laufzeitverteilung $\mathcal{L}(X)$ gilt*

$$\alpha^* \ \leq\ \beta^* \leq\ 4\alpha^*. \tag{2.9}$$

Beweis. Für beliebiges $k \geq 1$ gilt (!)

$$\mathbb{E}[\min(X,k)] > \frac{k}{2} \ \text{ oder } \ \mathbb{P}\{X \leq \lfloor 2\mathbb{E}[\min(X,k)]\rfloor\} \geq \frac{1}{2}.$$

Ist nämlich $2\,\mathbb{E}[\min(X,k)] \leq k$, dann folgt mit Hilfe der Markov Ungleichung

$$\mathbb{P}\{X \geq \lceil 2\mathbb{E}[\min(X,k)]\rceil\} = \mathbb{P}\{\,\min(X,k) \geq \lceil 2\mathbb{E}[\min(X,k)]\rceil\} \leq \frac{1}{2}.$$

Sei nun $k^* \in A$. Im ersten Fall folgt

$$\alpha^* \ =\ \frac{\mathbb{E}[\min(X,k^*)]}{\mathbb{P}\{X \leq k^*\}} \ >\ \frac{k^*}{2\,\mathbb{P}\{X \leq k^*\}} \ \geq\ \frac{\beta^*}{2}.$$

Im zweiten Fall folgt

$$\beta^* \ \leq\ \frac{\lfloor 2\mathbb{E}[\min(X,k^*)]\rfloor}{\mathbb{P}\{X \leq \lfloor 2\mathbb{E}[\min(X,k^*)]\rfloor\}} \ \leq\ 4\,\mathbb{E}[\min(X,k^*)] \ \leq\ 4\,\frac{\mathbb{E}[\min(X,k^*)]}{\mathbb{P}\{X \leq k^*\}} \ =\ 4\alpha^*.$$

Die erste Ungleichung in (2.9) ist wegen $\mathbb{E}[\min(X,k)] \leq k$ offensichtlich.

Die Intuition für die Schranke (2.8) in Satz 2.4 ist die folgende. Die Wahrscheinlichkeit, dass das Verfahren nach $c/\mathbb{P}\{X \leq k\}$ Läufen der Länge k noch keine Lösung ausgegeben hat, ist extrem klein, wenn c nur groß genug ist. Zu diesem Zeitpunkt hat das Verfahren jeweils etwa $c\beta(k)$ Zeit mit Läufen der Länge 2^j, $j = 0,\ldots,\log_2 c\beta(k)$, verbracht, also $O(\beta(k)(\log\beta(k) + c'))$ Schritte vollführt. Die Abschätzung gilt für beliebiges $k \geq k_{\min}$, es ergibt sich somit die Größenordnung $O(\beta^*(\log\beta^* + c'')) = O(\alpha^*(\log\alpha^* + c'''))$ als obere Schranke für $\mathbb{E}Y_{(k_i)}$.

Beweis von Satz 2.4. Sei a_ℓ, $\ell \geq 0$ eine streng monoton wachsende Folge nicht-negativer ganzer Zahlen und Y eine Zufallsvariable mit werten in $\mathbb{N}$. Dann gilt

$$\begin{aligned}
\mathbb{E}Y \ &= \ \sum_{n=0}^{\infty} \mathbb{P}\{Y > n\} \\
&= \ \sum_{n=0}^{a_0-1} \mathbb{P}\{Y > n\} \ + \ \sum_{\ell=0}^{\infty} \sum_{n=a_\ell}^{a_{\ell+1}-1} \mathbb{P}\{Y > n\} \\
&\leq \ a_0 \ + \ \sum_{\ell=0}^{\infty} a_{\ell+1}\,\mathbb{P}\{Y > a_\ell\}.
\end{aligned} \tag{2.10}$$

Sei nun $(k_i)_{i\geq 1}$ eine Folge mit der Eigenschaft (2.6). Wegen (2.7) gilt für $Y = Y_{(k_i)}$,

$$\mathbb{P}\{Y > m2^{m-1}\} \;=\; \mathbb{P}\{Y > n_{2^m-1}\} \;\leq\; \mathbb{P}\{X > 2^j\}^{2^{m-1-j}}, \quad 0 \leq j \leq m-1. \tag{2.11}$$

Wähle ein $k' \in \mathbb{N}$ mit $\beta(k') = \beta^*$ und setze $j' = \lceil \log_2 k' \rceil$, $n' = \lceil -\log_2 \mathbb{P}\{X \leq k'\} \rceil$. Nach (2.11) gilt

$$\begin{aligned} \mathbb{P}\{Y > (j'+n'+\ell+1)\, 2^{j'+n'+\ell}\} &\leq \mathbb{P}\{X > 2^{j'}\}^{2^{n'+\ell}} \\ &\leq (1 - \mathbb{P}\{X \leq k_0\})^{2^\ell / \mathbb{P}\{X \leq k_0\}} \\ &\leq \exp(-2^\ell), \quad \ell \geq 0. \end{aligned} \tag{2.12}$$

Ohne Einschränkung sei $j' + n' \geq 1$ (falls nicht, gilt $\mathbb{P}\{Y = 1\} = 1$). Aus (2.10) (2.12) folgt (setze $a_\ell ;=: (j'+n'+\ell+1)\, 2^{j'+n'+\ell}$)

$$\begin{aligned} \mathbb{E}Y &\leq (j'+n'+1)\, 2^{j'+n'} + \sum_{\ell=0}^{\infty} (j'+n'+\ell+2)\, 2^{j'+n'+\ell+1} \exp(-2^\ell) \\ &\leq (j'+n')\, 2^{j'+n'+1} \left(1 + \sum_{\ell=0}^{\infty} (\ell+3) \left(\frac{2}{e^2}\right)^\ell\right) \\ &= (j'+n')\, 2^{j'+n'}\, 2 \left(1 + \frac{e^2(3e^2-4)}{(e^2-2)^2}\right). \end{aligned} \tag{2.13}$$

Die Behauptung von Satz 2.4 folgt nunmehr aus (2.13), Lemma 2.5 und der Beobachtung

$$2^{j'+n'} \;\leq\; 2^{\log_2 k' \,-\, \log_2 \mathbb{P}\{X \leq k'\} + 2} \;=\; 4\beta^*.$$

Insbesondere gilt (2.8) mit $c_1 = 180$ und $c_2 = 4$.

Wir zeigen nun, dass die erwartete Laufzeit der Strategien aus Satz 2.4 von der bestmöglichen Größenordnung ist: Zu einer universellen Strategie gibt es stets einen Las-Vegas-Algorithmus, so dass die erwartete Laufzeit dieser Strategie die erwartete Laufzeit der spezifischen optimalen Strategie um einen logarithmischen Faktor übersteigt.

Satz 2.6 *Für jede Strategie $(k_i)_{i\geq 1}$ und jedes $1 < \alpha < \infty$ gilt*

$$\sup_{\mathcal{L}(X):\, \alpha^* = \alpha} \mathbb{E}Y_{(k_i)} \;\geq\; \frac{1}{8}\, \alpha \log_2 \alpha.$$

Beweis. Zu festem $1 < \alpha < \infty$ konstruieren wir eine endliche Familie von Verteilungen, so dass für jede dieser Verteilungen $\alpha^* = \alpha$, und für zumindest eine der Verteilungen $\mathbb{E}Y_{(k_i)} \geq \frac{1}{8}\, \alpha \log_2 \alpha$. Setze $n^* = \lfloor \log_2 \alpha \rfloor$ und definiere für $0 \leq j \leq n^*$,

$$\mathbb{P}_j\{X = k\} = \begin{cases} 2^j/\alpha, & \text{falls } k = 2^j; \\ 1 - 2^j/\alpha, & \text{falls } k = \infty; \\ 0, & \text{sonst.} \end{cases} \tag{2.14}$$

Wegen $\mathbb{E}_j[\min(X, 2^j)] = 2^j$ gilt $\alpha^* = \alpha$ für jede der $n^* + 1$ Verteilungen. Betrachte nun eine beliebige Strategie (k_i). Für alle $t \in \mathbb{N}$ und $0 \le j \le n^*$ gilt

$$\mathbb{P}_j\{Y_{(k_i)} > t\} \;=\; \mathbb{P}_j\{X > 2^j\}^{c(j,t)}, \tag{2.15}$$

wobei $c(j,t)$ die Anzahl der bis zur Zeit t (maximal) beendeten Läufe der Mindestlänge 2^j bezeichnet,

$$c(j,t) = |\{1 \le i \le m \colon k_i \ge 2^j\}| \;+\; 1_{\{t - n_m \ge 2^j\}}, \quad \text{falls } n_m \le t < n_{m+1}.$$

Für $0 \le j \le n^*$ gilt

$$\mathbb{P}_j\{Y_{(k_i)} > t\} \;\ge\; \frac{1}{2}, \quad \text{falls } c(j,t) \le \frac{\alpha}{2^{j+1}}. \tag{2.16}$$

Für $j = n^*$ ergibt sich (2.16) unmittelbar aus (2.15), für $0 \le j \le n^* - 1$ erhält man (2.16) aus (2.14), (2.15) und der Ungleichung $(1-x)^{\frac{1}{2x}} \ge \frac{1}{2},\ x \in (0, \frac{1}{2}]$. Aus (2.16) folgt insbesondere

$$\mathbb{E}_j Y_{(k_i)} \;\ge\; \frac{t+1}{2}, \quad \text{falls } c(j,t) \le \frac{\alpha}{2^{j+1}}.$$

Ist andererseits $c(j,t) \ge \alpha/2^{j+1}$ für alle $0 \le j \le n^*$, dann folgt

$$\begin{aligned} t \;&\ge\; \frac{\alpha}{2^{n^*+1}}\, 2^{n^*} \;+\; \left(\frac{\alpha}{2^{n^*}} - \frac{\alpha}{2^{n^*+1}}\right) 2^{n^*-1} \;+\; \cdots \;+\; \left(\frac{\alpha}{2} - \frac{\alpha}{4}\right) 2^0 \\ &=\; \alpha\left(\frac{1}{2} + \sum_{j=0}^{n^*-1} \frac{1}{4}\right) \;=\; \frac{\alpha(n^*+2)}{4} \;\ge\; \frac{\alpha \log_2 \alpha}{4}. \end{aligned}$$

Somit erhält man für eine beliebige Folge (k_i) und $t \in \mathbb{R}^+$

$$\begin{aligned} \mathbb{E}_j Y_{(k_i)} \;<\; \frac{t}{2},\ 0 \le j \le n^* \;&\Longrightarrow\; c(j, \lfloor t \rfloor) \ge \frac{\alpha}{2^{j+1}},\ 0 \le j \le n^* \\ &\Longrightarrow\; \lfloor t \rfloor \ge \frac{\alpha \log_2 \alpha}{4}, \end{aligned}$$

also

$$\max_{0 \le j \le n^*} \mathbb{E}_j Y_{(k_i)} \;\ge\; \frac{\alpha \log_2 \alpha}{8}.$$

3 Techniken aus der Spieltheorie

In diesem Kapitel beschäftigen wir uns mit spieltheoretischen Ansätzen bei der Analyse von Las-Vegas-Algorithmen. Diese münden in Yao's Minimax-Prinzip, das die Herleitung einer unteren Schranke für die Güte randomisierter Algorithmen ermöglicht. Zur Veranschaulichung der Technik wird uns als Beispiel die Auswertung sogenannter Spielbäume begleiten.

3.1 Das Auswerten von Spielbäumen

Ein Spielbaum ist ein verwurzelter und beschrifteter Baum. Innere Knoten aus geradzahligen Generationen und die Wurzel sind mit MIN beschriftet, innere Knoten aus ungeraden Generationen mit MAX. Jedem Blatt des Baums ist ein reeller Wert zugeordnet. Der Wert eines inneren Knoten mit der Beschriftung MAX ist das Maximum der Werte seiner Kinder, der Wert eines mit MIN beschrifteten inneren Knoten ist das Minimum der Werte seiner Kinder. Unter der Auswertung eines Spielbaums versteht man das Bestimmen des Wertes an der Wurzel des Baums.

Ein Spielbaum beschreibt das folgende Spiel: Zwei Spieler A und B bestimmen einen in der Wurzel startenden Pfad durch den Baum. Spieler A bekommt von Spieler B eine Zahlung in Höhe des Wertes des Blattes am Ende des Pfades. Spieler B darf beginnen. Die Kinder eines inneren Knoten entsprechen den verschiedenen Spielfortführungen, die dem am Zug befindlichen Spieler zur Auswahl stehen. Der Wert eines inneren Knoten beschreibt die bei kluger zukünftiger Spielführung garantierte bzw. maximal drohende Auszahlung am Ende des Spiels.

Ein Algorithmus zur Auswertung eines Spielbaums liest bei jedem Schritt den Wert eines Blattes ein, wobei die Wahl des als nächstes einzulesenden Blattes von den Werten der bereits inspizierten Blätter abhängen darf. Als Maß für die Laufzeit eines solchen Algorithmus nehmen wir die Anzahl der eingelesenen Werte.

Wir beschränken uns der Einfachheit halber auf den Fall binärer Spielbäume deren Blätter den Wert 0 oder 1 haben. In diesem Fall kann man Minimum- und Maximumbildung durch die Booleschen Operationen AND und OR ersetzen. Wir bezeichnen mit $\mathbf{t}_k$ einen solchen Spielbaum der Höhe $2k$,

$$\mathbf{t}_k = (v_1, \ldots, v_{4^k}) \in \{0,1\}^{4^k}.$$

Dabei ist v_i der Wert des i-ten Blattes bzgl. einer beliebigen aber festen Abzählung der Blätter. Den Wert des Baums $\mathbf{t}_k$ an seiner Wurzel bezeichnen wir mit $w(\mathbf{t}_k)$.

Die (deterministische) Komplexität des Auswertens eines binären 0-1 Spielbaums ist $n = 4^k$: Zu jedem deterministischen Algorithmus gibt es eine Belegung der Blätter von $\mathbf{t}_k$, die den Algorithmus zwingt, jedes einzelne Blatt zu inspizieren. Das liegt daran, dass man den Algorithmus stets zwingen kann, die Werte beider Kinder eines inneren Knoten zu bestimmen, indem man ihm an einem AND-Knoten zunächst eine 1 und an einem OR-Knoten zunächst eine 0 liefert.

Im nachfolgenden randomisierten Algorithmus zur Auswertung eines Spielbaums (Snir [29], 1985) werden die Kinder der Knoten in zufälliger Reihenfolge ausgewertet.

Randomisierter Algorithmus zur Auswertung eines Spielbaums
Zur Auswertung eines AND-Knoten wähle man rein zufällig eines der beiden Kinder und bestimme rekursiv dessen Wert (= Wert eines in einem OR-Knoten verwurzelten Teilbaums). Ist der Wert des Kindes 0, so ist dies auch der zu bestimmende Wert des AND-Knoten. Ist der Wert des Kindes 1, dann bestimme man auch noch den Wert des anderen Kindes. Die Auswertung eines OR-Knoten verläuft analog mit vertauschten Rollen von 0 und 1.

Satz 3.1 *Sei $L(\mathbf{t}_k)$ die Anzahl der bei der Auswertung des Spielbaums $\mathbf{t}_k$ inspizierten Blätter, dann gilt*

$$\max_{\mathbf{t}_k \in \{0,1\}^{4^k}} \mathbb{E}L(\mathbf{t}_k) \leq 3^k, \ k \in \mathbb{N}. \tag{3.1}$$

Bemerkungen.

- Die Eingabe hat die Länge $n = 4^k$. Die erwarteten Kosten des randomisierten Algorithmus sind also gleichmäßig durch $n^{\log_4 3}$ beschränkt ($\log_4 3 = 0.7924\ldots$).
- Der Erwartungswert $\mathbb{E}L(\mathbf{t}_k)$ ist nicht für alle $\mathbf{t}_k$ gleich, es gibt durchaus unterschiedlich schwierige Belegungen der Blätter.

Beweis. Wir beweisen (3.1) per Induktion und betrachten zunächst den Fall $k = 1$.

Ist $w(\mathbf{t}_1) = 0$, so ist auch der Wert zumindest eines der beiden Kinder der Wurzel 0. Der randomisierte Algorithmus wählt also mit Wahrscheinlichkeit $\geq \frac{1}{2}$ zuerst ein Kind mit dem Wert 0, die Auswertung des zweiten Kindes ist dann hinfällig. Die erwartete Anzahl der auszuwertenden Kinder ist also $\leq \frac{3}{2}$ und bei der Auswertung eines Kindes müssen schlimmstenfalls die zugehörigen zwei Blätter inspiziert werden.

Ist $w(\mathbf{t}_1) = 1$, dann müssen beide Kinder der Wurzel ausgewertet werden. Da diese beiden OR-Kinder jedoch den Wert 1 haben, kostet die Auswertung jedes Kindes in Erwartung höchstens $\frac{3}{2}$, da mit Wahrscheinlichkeit $\geq \frac{1}{2}$ das zuerst gewählte Blatt den Wert 1 hat.

Wir kommen nun zum Induktionsschritt von k auf $k+1$. Seien $\mathbf{t}_k^{(i)}$, $i = 1, \ldots, 4$, die in den Enkeln der Wurzel von $\mathbf{t}_{k+1}$ verwurzelten binären Spielbäume der Höhe $2k$. Vom Fall $k = 1$ her wissen wir, dass in Erwartung höchstens drei der vier Enkel inspiziert werden müssen. Die Bestimmung jedes der Werte $w(\mathbf{t}_k^{(i)})$ kostet nach Induktionsvoraussetzung in Erwartung höchstens 3^k. Da die Reihenfolge, in der die Enkel inspiziert werden, unabhängig ist von den $L(\mathbf{t}_k^{(i)})$, folgt die Behauptung für $k + 1$.

3.2 Das Minimax-Prinzip

Im vorigen Abschnitt haben wir gesehen, dass die erwarteten Kosten des randomisierten Algorithmus von Snir zur Auswertung eines binären 0-1 Spielbaums bei beliebiger Blattbelegung durch $n^{\log_4 3}$ beschränkt sind. Wie ist die Güte dieses Algorithmus zu beurteilen? Was also ist

die "randomisierte Komplexität" des Problems des Auswertens von Spielbäumen? In diesem Abschnitt werden wir ganz allgemein eine Technik zur Herleitung einer unteren Schranke für die maximale erwartete Laufzeit von Las-Vegas-Algorithmen kennen lernen. Das Maximum wird dabei über die möglichen Eingaben gebildet.

Wir erinnern zunächst an einige grundlegende Begriffe aus der Spieltheorie. Wir betrachten ein endliches Zwei-Personen-Nullsummen-Spiel bei dem der Spieler A über die reinen Strategien $a_i,\ 1 \leq i \leq n$ und der Spieler B über die reinen Strategien $b_j,\ 1 \leq j \leq m$ verfügt. Wählt der Spieler A die Strategie a_i und der Spieler B die Strategie b_j, dann erhält der Spieler A von Spieler B den Betrag c_{ij}. Die Matrix (c_{ij}) nennt man die *Auszahlungs-* oder *Gewinnmatrix*. Wählt ein Spieler zufällig eine der ihm zur Verfügung stehenden Strategien, so spricht man von einer *gemischten* Strategie. Wählt der Spieler A mit Wahrscheinlichkeit p_i die Strategie a_i und der Spieler B davon unabhängig mit Wahrscheinlichkeit q_j die Strategie b_j, dann ist die erwartete Höhe der von Spieler B an den Spieler A zu leistenden Zahlung gleich

$$\mathbb{E}_{p\otimes q}\, c(I,J) \;=\; \sum_{i,j} p_i q_j c_{ij}, \quad \text{wobei } p=(p_1,\ldots,p_n) \text{ und } q=(q_1,\ldots,q_m).$$

Hierbei ist I die Nummer der vom Spieler A zufällig gewählten Strategie und J die Nummer der Strategie von B. Verfolgt der Spieler A die gemischte Strategie p, so ist sein erwarteter Gewinn mindestens $\min_q \mathbb{E}_{p\otimes q}\, c(I,J)$. Den Wert

$$V_A := \max_p \min_q \mathbb{E}_{p\otimes q}\, c(I,J)$$

nennt man den *unteren Wert* des Spiels. Eine Strategie p^*, die $\min_q \mathbb{E}_{p\otimes q}\, c(I,J)$ maximiert, nennt man eine *optimale* gemischte Strategie für den Spieler A. Der Wert V_A ist eine untere Schranke für den erwarteten Gewinn des Spielers A bei Verwendung einer optimalen Strategie, selbst wenn der Spieler diese öffentlich macht. Umgekehrt ist $\max_p \mathbb{E}_{p\otimes q}\, c(I,J)$ die maximale Höhe der von Spieler B zu leistenden Zahlung, wenn er die Strategie q verfolgt. Der Wert

$$V_B := \min_q \max_p \mathbb{E}_{p\otimes q}\, c(I,J)$$

heißt *oberer Wert* des Spiels. Eine Strategie q^*, die $\max_p \mathbb{E}_{p\otimes q}\, c(I,J)$ minimiert, nennt man eine optimale Strategie für den Spieler B. Der Wert V_B ist der maximal zu erwartende Betrag, den der Spieler B bei bei Verfolgen einer optimalen Strategie an A zahlen muss. Existieren Strategien p^* und q^*, so dass

$$\mathbb{E}_{p\otimes q^*}\, c(I,J) \;\leq\; \mathbb{E}_{p^*\otimes q^*}\, c(I,J) \;\leq\; \mathbb{E}_{p^*\otimes q}\, c(I,J) \text{ für alle } p \text{ und } q,$$

dann sind diese Strategien notwendigerweise optimal und man sagt, das Spiel hat die *Lösung* bzw. den *Sattelpunkt* (p^*,q^*). Die Größen $\mathbb{E}_{p^*\otimes q^*}\, c(I,J)$, V_A und V_B stimmen dann überein und werden als der *Wert* des Spiels bezeichnet.

Dürfen die Spieler lediglich reine Strategien verwenden, dann können der obere und der untere Wert eines Spiels differieren. Hingegen besagt der folgende Fundamentalsatz der Spieltheorie, dass bei Verwendung gemischter Strategien der untere und der obere Wert eines endlichen

Zwei-Personen-Nullsummen-Spiels stets übereinstimmen (einen Beweis dieses Satzes findet man etwa in [19]).

Satz 3.2 (von Neumann's Minimax-Theorem) *Jedes durch eine endliche Matrix* (c_{ij}) *beschriebene Zwei-Personen-Nullsummen-Spiel besitzt eine Lösung und hat den Wert*

$$\max_p \min_q \mathbb{E}_{p\otimes q}\, c(I,J) \;=\; \min_q \max_p \mathbb{E}_{p\otimes q}\, c(I,J). \tag{3.2}$$

Für festes p ist die Funktion $\mathbb{E}_{p\otimes q}c(I,J) = \sum_j q_j \sum_i p_i c_{ij}$ linear in q und nimmt ihr Minimum in einem Extremalpunkt des Simplex der Verteilungen auf $\{1,\ldots,m\}$ an. Folglich gilt

$$\min_q \mathbb{E}_{p\otimes q}\, c(I,J) = \min_j \mathbb{E}_p\, c(I,j).$$

Mit anderen Worten, kennt der Spieler B die gemischte Strategie p des Spielers A, so kann er mit einer reinen Strategie dagegenhalten (i.A. ist diese Strategie keine optimale Strategie). Entsprechendes gilt für den Spieler A, wenn er die Strategie q des Spielers B kennt. Eine Variante von (3.2) ist daher die folgende Gleichheit **(Satz von Loomis)**,

$$\max_p \min_j \mathbb{E}_p c(I,j) \;=\; \min_q \max_i \mathbb{E}_q\, c(i,J). \tag{3.3}$$

Wir benutzen nun das Minimax-Prinzip zur Analyse der Güte von Las-Vegas-Algorithmen. Wir fassen die Kosten des Algorithmus als die Höhe der Zahlung des den Algorithmus entwerfenden Spielers an den die Eingabe wählenden Widersacher auf. Wir betrachten nur Probleme bei denen sowohl die Zahl der möglichen Eingaben als auch die Zahl der deterministischen (immer korrekten, in endlicher Zeit terminierenden) Algorithmen zur Lösung des Problems endlich sind. Alle bisher in der Vorlesung betrachteten Probleme wie etwa das Sortieren von Listen oder das Bestimmen des Medians besitzen diese Eigenschaft.

Sei $\mathcal{A}$ die Menge der deterministischen Algorithmen zur Lösung eines Problems und $\mathcal{I}$ die Menge der möglichen Eingaben. Für $a \in \mathcal{A}$ und $i \in \mathcal{I}$ bezeichne $c(i,a)$ die Kosten des Algorithmus a bei der Eingabe i. Die *Komplexität in Verteilung* des Problems ist definiert als

$$c_{dist} := \max_p \min_a \mathbb{E}_p c(I,a),$$

wobei das Maximum über sämtliche Verteilungen p auf $\mathcal{I}$ gebildet wird und die zufällige Eingabe I unter $\mathbb{P}_p$ die Verteilung p hat. Bei bekannter Eingabeverteilung p kann stets ein deterministischer Algorithmus $a(p)$ gefunden werden, dessen erwartete Kosten (gemittelt über die Eingaben!) durch c_{dist} nach oben beschränkt sind. Die Komplexität in Verteilung ist stets geringer als die *deterministische Komplexität* des Problems, $c_{det} = \min_a \max_i c(i,a)$ (= bestes worst-case-Verhalten eines deterministischen Algorithmus). Unter der *randomisierten Komplexität* eines Problems versteht man die Größe

$$c_{rand} := \min_q \max_i \mathbb{E}_q\, c(i,A),$$

wobei das Minimum über alle Verteilungen q auf $\mathcal{A}$ gebildet wird und der zufällige Algorithmus A unter $\mathbb{P}_q$ die Verteilung q hat. c_{rand} ist eine untere Schranke für die Güte (=

maximalen erwarteten Kosten) eines randomisierten Algorithmus. (In diesem Abschnitt ist es zweckmäßig, einen randomisierten Algorithmus als eine Wahrscheinlichkeitsverteilung auf einer Menge von deterministischen Algorithmen aufzufassen.)

Das Minimax-Prinzip in der Form (3.3) besagt, dass die beiden Komplexitätsbegriffe c_{dist} und c_{rand} übereinstimmen. Insbesondere gilt

Proposition 3.3 (Yao's Minimax-Prinzip) *Für alle Verteilungen p auf der Menge $\mathcal{I}$ der möglichen Eingaben und alle Verteilungen q auf der Menge $\mathcal{A}$ der deterministischen Algorithmen zur Lösung eines endlichen Problems gilt*

$$\min_{a\in\mathcal{A}} \mathbb{E}_p c(I,a) \;\le\; \max_{i\in\mathcal{I}} \mathbb{E}_q c(i,A).$$

Die maximale erwartete Laufzeit eines jeden randomisierten Algorithmus ist also durch die mittlere Laufzeit eines optimalen deterministischen Algorithmus bei Eingabeverteilung p nach unten beschränkt. Man kann daher eine untere Schranke für die randomisierte Komplexität herleiten, indem man die erwartete Laufzeit eines optimalen deterministischen Algorithmus (bei freier aber öffentlich zu machender Wahl der Eingabeverteilung p) nach unten abschätzt.

Wir kehren nun zu unserem Beispiel des Auswertens binärer 0-1 Spielbäume zurück. Zunächst bemerken wir, dass der Wert eines binären Spielbaums und der Wert eines binären Baums gleicher Höhe, dessen innere Knoten alle mit NOR beschriftet sind, bei gleicher Belegung der Blätter übereinstimmen. Man verifiziere das für binäre Bäume der Höhe 2 und erinnere sich, dass man einen Spielbaum der Höhe $2(k+1)$, als einen Spielbaum der Höhe 2 auffassen kann, an dessen vier Blättern Spielbäume der Höhe $2k$ angebracht sind. Um eine untere Schranke für die maximale erwartete Anzahl der durch einen zufälligen Algorithmus inspizierten Blätter herzuleiten, müssen wir zunächst eine Verteilung der Blattbelegung spezifizieren und dann eine untere Schranke für die erwarteten Kosten eines optimalen deterministischen Algorithmus beweisen. Wir wählen die Verteilung p wie folgt. Jedem Blatt des NOR-Baums wird unabhängig mit Wahrscheinlichkeit $\bar{p}$ der Wert 1 zugewiesen wird, wobei

$$\bar{p} = (1-\bar{p})^2,$$

also $\bar{p} = \frac{3-\sqrt{5}}{2} = 0.3819\ldots$ Insbesondere ist die Wahrscheinlichkeit, dass ein innerer Knoten den Wert 1 besitzt (also beide Teilbäume den Wert 0), ebenfalls $\bar{p}$. Die Werte von Knoten, die nicht in einer Vorfahr-Nachkomme-Beziehung stehen, sind unabhängig, also insbesondere die Werte von Knoten aus ein und derselben Generation. Aufgrund dieser Tatsache wird jeder optimale deterministische Algorithmus zunächst den Wert eines von zwei Geschwisterteilbäumen bestimmen, bevor er ein Blatt des anderen Teilbaums inspiziert. Der Baum wird also entsprechend einer "depth-first search" durchlaufen, wobei die Untersuchung eines Teilbaums in dem Moment abgebrochen wird, in dem der Wert des Teilbaums bestimmt ist. Bezeichne Z_h die Anzahl der Blätter, die ein solcher Algorithmus zur Bestimmung des Wertes eines Knoten in der Generation $2k-h$ inspiziert. Dann gilt

$$Z_h \;\stackrel{d}{=}\; Z_{h-1}^{(1)} + Z_{h-1}^{(2)}\, I_{A_{h-1}}.$$

Dabei sind die $Z_{h-1}^{(i)}$, $i = 1, 2$ unabhängige Kopien von Z_{h-1}. Das Ereignis A_{h-1}, dass der zuerst ausgewertete Teilbaum den Wert 0 hat, ist unabhängig von $Z_{h-1}^{(2)}$ und es gilt $\mathbb{P}\{A_{h-1}\} = 1 - \bar{p}$. Es folgt

$$\mathbb{E}Z_h \;=\; \mathbb{E}Z_{h-1} + (1-\bar{p})\,\mathbb{E}Z_{h-1} \;=\; (2-\bar{p})\,\mathbb{E}Z_{h-1},$$

also $\mathbb{E}Z_{2k} = (2-\bar{p})^{2k}$. Aus Yao's Minimax-Prinzip folgt nunmehr

Satz 3.4 *Die maximale erwartete Anzahl der von einem zufälligen Algorithmus bei der Auswertung eines binären 0-1 Spielbaums inspizierten Blätter ist mindestens n^α. Dabei ist n die Anzahl der Blätter des Baums und $\alpha = \log_2(2-\bar{p}) = 0.6942\ldots$*

Die Schranken in den Sätzen 3.1 und 3.4 differieren. Unser randomisierter Algorithmus kann also nur optimal sein, wenn die Schranke in Satz 3.1 nicht scharf ist und/oder die Wahl der Verteilung der Blattbelegung nicht die bestmögliche war (= schwierigste für deterministische Algorithmen). Letzteres ist aber offensichtlich, denn ein vernünftiger Algorithmus wird, wenn er den Wert eines Kindes eines NOR-Knoten als 1 bestimmt hat, nicht auch noch den Teilbaum des anderen Kindes auswerten. Bei einer möglichst schwierigen Verteilung auf den Blattbelegungen werden also Geschwister nicht beide den Wert 1 haben. Aber auch die obere Schranke aus Satz 3.1 ist nicht die bestmögliche. Es gilt nämlich (Theorem 1.5 in [25])

Satz 3.5 *Der randomisierte Algorithmus von Snir zur Auswertung binärer 0-1 Spielbäume ist optimal. Die Komplexität des Auswertens eines binären Spielbaums mit n Blättern ist n^β, wobei $\beta = \log_2(\frac{1+\sqrt{33}}{4}) = 0.7537\ldots$*

4 Abweichungen vom erwarteten Wert

4.1 Die Bernstein-Chernoff-Schranke

Die erwarteten Kosten eines randomisierten Algorithmus sind oft relativ einfach zu berechnen bzw. nach oben abzuschätzen. Aus naheliegenden Gründen ist es wünschenswert, die Wahrscheinlichkeit für Abweichungen der Kosten von ihrem erwarteten Wert zu kontrollieren. Im Allgemeinen hat man zur Abschätzung der Abweichungen vom erwarteten Wert lediglich die Markov-Ungleichung zur Verfügung. Eine weit bessere Abschätzung erhält man für Summen unabhängiger Bernoulli-Zufallsvariablen.

Satz 4.1 (Bernstein-Chernoff-Schranke) *Seien X_i, $i \geq 1$ unabhängige Bernoulli-verteilte Zufallsvariablen zum Parameter $p_i := \mathbb{P}\{X_i = 1\}$. Sei $S_n := \sum_{i=1}^n X_i$ und $\mu_n := \mathbb{E}S_n = \sum_{i=1}^n p_i$, $n \geq 1$. Dann gilt*

$$\mathbb{P}\{S_n \geq (1+\delta)\mu_n\} \leq \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^{\mu_n} \text{ für alle } \delta \geq 0 \text{ und } n \geq 1; \tag{4.1}$$

$$\mathbb{P}\{S_n \leq (1-\delta)\mu_n\} \leq \exp\left(-\frac{\delta^2}{2}\,\mu_n\right) \text{ für alle } 0 \leq \delta < 1 \text{ und } n \geq 1. \tag{4.2}$$

Bemerkungen.

- Die Schranken sind nicht symmetrisch, d.h. die Schranke in (4.2) und die Schranke (4.1) für Abweichungen vom Erwartungswert nach oben der Zufallsgröße $\overline{S}_n := n - S_n = \sum_{i=1}^n (1 - X_i)$ sind verschieden.

- Man beachte, dass die Schranken nur von μ_n, nicht aber von den einzelnen p_i oder von n abhängen. Daher kann man die Abschätzungen auch dann verwenden, wenn man lediglich eine obere bzw. untere Schranke von μ_n kennt. Gilt etwa $\mu_n \leq \nu_n$ (bzw. $\mu_n \geq \nu_n$), dann behält die Ungleichung (4.1) (bzw. (4.2)) ihre Gültigkeit, wenn man μ_n auf beiden Seiten durch ν_n ersetzt: Im Fall $\mu_n \leq \nu_n$ wählt man $p_i' \geq p_i, 1 \leq i \leq n$ mit $\sum_{i=1}^n p_i' = \nu_n$, und konstruiert unabhängige Bernoulli-Zufallsvariablen X_i' mit

  $$\mathbb{P}\{X_i' = 1 \,|\, X_i = 1\} = 1 \text{ und } \mathbb{P}\{X_i' = 1 \,|\, X_i = 0\} = \frac{p_i' - p_i}{1 - p_i}.$$

  Für $S_n' := \sum_{i=1}^n X_i'$ gilt dann $\mathbb{P}\{S_n' \geq S_n\} = 1$ und die Behauptung folgt mittels der Bernstein-Chernoff Schranke für S_n'. Im Fall $\mu_n \geq \nu_n$ konstruiert man entsprechend unabhängige Bernoulli-verteilte Zufallsvariablen X_i'' mit der Eigenschaft $\mathbb{P}\{S_n'' \leq S_n\} = 1$.

Beweis. Wir beweisen die Schranke (4.1) für Abweichungen vom Erwartungswert nach oben. Mittels der Markov-Ungleichung folgt für beliebiges $\lambda > 0$

$$\begin{aligned}\mathbb{P}\{S_n \geq (1+\delta)\mu_n\} &= \mathbb{P}\{\exp(\lambda S_n) \geq \exp(\lambda(1+\delta)\mu_n)\} \\ &\leq \exp(-\lambda(1+\delta)\mu_n)\,\mathbb{E}\exp(\lambda S_n). \qquad (4.3)\end{aligned}$$

Mit Hilfe der Funktionalgleichung der Exponentialfunktion und der Unabhängigkeit der X_i ergibt sich

$$\begin{aligned}\mathbb{E}\exp(\lambda S_n) &= \mathbb{E}\exp\Big(\sum_{i=1}^{n}\lambda X_i\Big) = \mathbb{E}\Big(\prod_{i=1}^{n}\exp(\lambda X_i)\Big) \\ &= \prod_{i=1}^{n}\mathbb{E}\exp(\lambda X_i) = \prod_{i=1}^{n}(p_i e^{\lambda}+(1-p_i)) = \prod_{i=1}^{n}(1+p_i(e^{\lambda}-1)).\end{aligned}$$

Da $1+x \leq e^x$ für $x \geq 0$, folgt

$$\begin{aligned}\mathbb{E}\exp(\lambda S_n) &\leq \prod_{i=1}^{n}\exp\left(p_i(e^{\lambda}-1)\right) \\ &= \exp\Big(\sum_{i=1}^{n}p_i(e^{\lambda}-1)\Big) = \exp\left(\mu_n(e^{\lambda}-1)\right). \qquad (4.4)\end{aligned}$$

(Die Ungleichung in (4.4) entspricht dem Ersetzen der X_i durch Poisson-verteilte Zufallsvariablen $\bar{X}_i$ mit Erwartungswert p_i.) Durch Kombination von (4.3) mit (4.4) erhält man

$$\begin{aligned}\mathbb{P}\{S_n \geq (1+\delta)\mu_n\} &\leq \exp\left(-\lambda(1+\delta)\mu_n\right)\exp\left(\mu_n(e^{\lambda}-1)\right) \\ &= \exp\Big(\mu_n(e^{\lambda}-1-\lambda(1+\delta))\Big). \qquad (4.5)\end{aligned}$$

Die Wahl $\lambda = \log(1+\delta)$ minimiert den Ausdruck auf der rechten Seite von (4.5) (man differenziere $e^{\lambda}-1-\lambda(1+\delta)$ nach λ) und liefert die Schranke (4.1).

Zum Beweis der Schranke (4.2) verwendet man bei der Abschätzung mittels der Markov-Ungleichung die Funktion $\exp(-\lambda\,\cdot)$ statt $\exp(\lambda\,\cdot)$ und verfährt dann analog. Den schöneren geschlossenen Ausdruck erhält man auf Grund der Tatsache, dass $(1-\delta)^{1-\delta} \geq \exp\left(-\delta+\frac{\delta^2}{2}\right)$ für $0 \leq \delta < 1$.

4.2 Routenplanung in Netzwerken

Die Routenplanung in Netzwerken ist ein weiteres Problem, bei dessen Lösung ein zufälliger Algorithmus jedem deterministischen Algorithmus überlegen ist. Wir modellieren ein Netzwerk paralleler Prozessoren durch einen gerichteten Graph (V, E). Entlang jeder gerichteten Kante kann pro Zeitschritt eine Nachricht einer gewissen Einheitslänge (ein Paket) befördert werden. Die Knoten haben unbeschränkte Lagerkapazität und können mehrere Pakete gleichzeitig empfangen (eines pro eingehender Kante). Der Anzahl der Kanten sind natürliche Grenzen gesetzt, typischerweise gilt $|E| << |V|^2$. Wir betrachten als Spezialfall den Boolesche Würfel $V = \{0,1\}^n$ mit $E = \{(v,w) : \sum_{j=1}^{n}|v_j - w_j| = 1\}$. Der Boolesche Würfel hat $|V| = 2^n$ Knoten und $|E| = n2^n$ Kanten. Zwei Knoten, deren Adresse sich nur an einer Stelle unterscheidet (Hammingabstand 1), können direkt miteinander kommunizieren.

Jeder Prozessor $i,\ 1 \leq i \leq 2^n$ enthält zu Anfang genau ein Paket (wir identifizieren den Knoten v mit $i = 1 + \sum_{j=1}^{n} v_j 2^{j-1}$). Das von i aus versandte Paket soll dem Prozessor $\pi(i)$ zugestellt werden. Der Einfachheit halber nehmen wir an, dass π eine Permutation der Knoten aus V ist. Die Aufgabe ist die Bestimmung der Routen (= Folgen von Kanten) entlang derer die Pakete befördert werden sollen, und zwar so, dass der gesamte Zustellprozess möglichst zeitig abgeschlossen ist. Der Zeitpunkt, zu dem der letzte Knoten das an ihn adressierte Paket empfängt, soll also minimiert werden. Ein Algorithmus zur Lösung dieses Problems spezifiziert sowohl die Routen der einzelnen Pakete, als auch die Devise, nach der Staus an Knoten abgebaut werden, also die Reihenfolge in der angestaute Pakete, die einen Knoten durch dieselbe Kante verlassen, weiter befördert werden.

Wir betrachten nur sogenannte "Scheuklappen-Algorithmen". Das sind Algorithmen, bei denen die Route des Pakets nur vom Ursprungs- und Bestimmungsort, also vom "Ticket" $(i, \pi(i))$ abhängt, nicht aber von den Bestimmungsorten der anderen Pakete. Scheuklappenalgorithmen haben den Vorteil, dass sie einfach zu implementieren sind und dass zur Weiterleitung eines Pakets der Blick auf das Ticket genügt. Ein Beispiel für eine Scheuklappen-Strategie ist das "Sukzessive Ausrichten der Bits". Hier wird zu jedem Zeitpunkt das erste Bit, das in der Adresse des gegenwärtigen Knoten von der Adresse des Bestimmungsorts differiert, geflippt. Soll also beispielsweise das Paket vom Ausgangsknoten 0011 dem Knoten 1000 zugestellt werden, so wird es zunächst zum Knoten 1011, dann zum Knoten 1001 und schließlich zum Bestimmungsort 1000 befördert.

Wie lange braucht ein deterministischer Algorithmus, der diese Strategie verfolgt, bis das letzte Paket zugestellt ist? Sehr lange, wenn er es mit einem ungünstigen π zu tun hat: Wird beispielsweise jedem Paket mit Start in $V' = \{v : v_j = 0,\ \lfloor \frac{n}{2} \rfloor < j \leq n\}$ ein Bestimmungsort in $W' = \{w : w_j = 0,\ 1 \leq j \leq \lfloor \frac{n}{2} \rfloor\}$ zugeordnet, so passiert jedes dieser Pakete den Knoten $(0, \ldots, 0)$. Es müssen also $|V'| = 2^{\lfloor \frac{n}{2} \rfloor}$ Pakete durch diesen einen Knoten geschleust werden. Da pro Zeitschritt nur n Pakete einen Knoten verlassen können, ist die Zeit für den gesamten Zustellprozess mindestens $2^{\lfloor \frac{n}{2} \rfloor}/n$, also exponentiell in n. Das folgende Resultat besagt, dass dieses fatale worst-case Verhalten keine spezielle Eigenschaft der Strategie "Sukzessives Ausrichten der Bits" ist (siehe [13]).

Satz 4.2 *Zu jedem deterministischen Scheuklappen-Algorithmus existiert eine Permutation π bei der der Algorithmus $\Omega(\sqrt{2^n}/n)$ Schritte bis zur Zustellung des letzten Pakets benötigt.*

Wir stellen nun einen zufälligen Algorithmus (Valiant, Brebner [31]) mit einer Laufzeit von der bestmöglichen Größenordnung n vor. Man beachte, dass beispielsweise jede Route von $(0, \ldots, 0)$ nach $(1, \ldots, 1)$ mindestens die Länge n hat.

Randomisierter Scheuklappen-Algorithmus zur Routenplanung in Netzwerken

Seien $\sigma(i)$, $1 \leq i \leq 2^n$ unabhängige rein zufällige Knoten aus der Menge V. In einer ersten Phase werden die Pakete zunächst von i zu den Knoten $\sigma(i)$ gesandt. Anschließend werden in einer zweiten Phase die Pakete von den Knoten $\sigma(i)$ zu ihren Bestimmungsorten $\pi(i)$ verschickt. In beiden Phasen werden die Pakete gemäß der Strategie "Sukzessives Ausrichten

der Bits" befördert. Warteschlangen werden nach der Devise FIFO ("wer zuerst kommt, mahlt zuerst") abgebaut. Gleichzeitig eintreffende Pakete, die einen Knoten durch dieselbe Kante verlassen, werden in beliebiger Reihenfolge weiterbefördert.

Bemerkungen.

- In der Praxis wird man das Ende der ersten Phase nicht abwarten und im Zwischenlager $\sigma(i)$ eintreffende Pakete sofort weiterbefördern. Eventuelle Konflikte zwischen angestauten Paketen der verschiedenen Phasen löst man dann nach der Devise "Pakete aus Phase 1 zuerst".
- Der Algorithmus ist offensichtlich leicht so zu modifizieren, dass ein Knoten mehrere Pakete (oder auch gar keines) empfangen und versenden kann. Wir machen die Annahme, dass π eine Permutation ist, nur der einfacheren Analysis zuliebe.

Satz 4.3 *Sei X_n der Zeitpunkt der Zustellung des letzten der 2^n Pakete durch den randomisierten Algorithmus. Für jede Permutation $\pi = \pi_n$ auf $\{0,1\}^n$ und jedes $c > 5.081...$ gilt*

$$\mathbb{P}_\pi\{X_n \geq cn\} \leq 2\,\theta_c^n, \quad n \geq 1, \tag{4.6}$$

wobei

$$\theta_c = \frac{2e^{\frac{c-3}{2}}}{(c-2)^{\frac{c}{2}-1}}.$$

Bemerkungen.

- Der Wert 5.081... erklärt sich als Lösung der Gleichung $\theta_c = 1$.
- Die Schranke in (4.6) hängt nicht von π ab, die Verteilung von X_n hingegen schon. Beispielsweise gilt $\mathbb{P}_\pi\{X_n = 0\} > 0$ nur dann, wenn $\pi = \mathrm{id}$.
- Man beachte, dass θ_c super-exponentiell abfällt, d.h. $\lim_{c\to\infty} \frac{1}{c} \log \theta_c = -\infty$. Anders als in der im Kapitel 2 diskutierten Situation empfiehlt es sich daher nicht, einen Zustellversuch abzubrechen und unabhängig neu zu starten.

Beweis. Wir analysieren zunächst die erste Phase. Für $1 \leq i \leq 2^n$ sei

$$\begin{aligned} T_i &:= \text{Ankunftszeit des in } i \text{ startenden Pakets im Zwischenlager } \sigma(i), \\ R_i &:= \text{Route des Pakets von } i \text{ nach } \sigma(i). \end{aligned}$$

Die Zufallsvariable $\max_{1\leq i\leq 2^n} T_i$ beschreibt die Dauer der ersten Phase. Die Zufallsgröße T_i setzt sich zusammen aus der Länge der Route R_i und dem Verzug V_i, d.h. der Anzahl der Zeitpunkte zu denen das in i startende Paket in einem Knoten warten muss, weil zunächst ein anderes Paket den Knoten durch die anvisierte Kante verlässt,

$$T_i = |R_i| + V_i.$$

Man beachte, dass das gemeinsame Wegstück zweier Routen R_i und R_j stets zusammenhängend ist, dass also ein Zusammenschluss nicht mehr möglich ist, sobald sich die Wege mal getrennt haben (das ist eine unmittelbare Konsequenz der Verwendung der Strategie "Sukzessives Ausrichten der Bits"). Es ist daher augenscheinlich, dass der bei der Zustellung Pakets i entstandene Verzug V_i durch die Anzahl der Pakete, deren Routen mit R_i ein gemeinsames Teilstück haben, nach oben beschränkt ist. (Zum formalen Nachweis dieser Eigenschaft stellt man injektiv jeden Zuwachs im Verzug des in i gestarteten Pakets einem Element aus $A_i := \{j \neq i : R_i \text{ und } R_j \text{ haben eine gemeinsame Wegstrecke}\}$ in Rechnung, siehe [20].) Mit $H_{ij} := I_{\{j \in A_i\}}$ gilt also

$$V_i \leq |A_i| = \sum_{j=1}^{2^n} H_{ij}. \tag{4.7}$$

Gegeben die Route R_i sind die $H_{ij}, 1 \leq j \leq 2^n$ bedingt unabhängig: Für alle möglichen Routen $(e_1, \ldots, e_\ell)$, $0 \leq \ell \leq n$, und alle $(x_1, \ldots, x_{2^n}) \in \{0,1\}^{2^n}$ gilt

$$\begin{aligned} &\mathbb{P}\{H_{ij} = x_j,\, 1 \leq j \leq 2^n \,|\, R_i = (e_1, \ldots, e_\ell)\} \\ &= \prod_{j=1}^{2^n} \mathbb{P}\{H_{ij} = x_j \,|\, R_i = (e_1, \ldots, e_\ell)\}. \end{aligned} \tag{4.8}$$

Die bedingte Unabhängigkeit (4.8) folgt unmittelbar aus der Unabhängigkeit der $\sigma(j)$. Das Bedingen ist aber unverzichtbar, denn die H_{ij} sind positiv korreliert (je länger die Route R_i, desto größer die erwartete Anzahl gemeinsamer Teilstücke). Mit der Bernstein-Chernoff-Schranke im Hinterkopf wollen wir den bedingten Erwatungswert

$$\mathbb{E}\Big(\sum_{j=1}^{2^n} H_{ij} \,\Big|\, R_i = (e_1, \ldots, e_\ell) \Big)$$

nach oben abschätzen. Der naheliegende Ansatz, die Wahrscheinlichkeiten $\mathbb{P}\{H_{ij} = 1 \,|\, R_i = (e_1, \ldots, e_\ell)\}$ zu berechnen, scheitert an der Abhängigkeit dieser Größen von der Route $(e_1, \ldots, e_\ell)$. Stattdessen schätzen wir die Zufallsvariable $\sum_{j=1}^{2^n} H_{ij}$ nach oben ab. Sei dazu $Y(e)$ die Anzahl der Routen, die die Kante e benutzen,

$$Y(e) := \sum_{j=1}^{2^n} I_{\{e \in R_j\}}.$$

Da die Indikatorvariable $I_{\{j \in A_i\}}$ durch die Länge des gemeinsamen Teilstücks $\sum_{e \in R_i} I_{\{e \in R_j\}}$ nach oben beschränkt ist, und da $H_{ii} = 0$ nach Definition, gilt

$$\sum_{j=1}^{2^n} H_{ij} \;\leq\; \sum_{j \neq i} \sum_{e \in R_i} I_{\{e \in R_j\}} \;=\; \sum_{e \in R_i} (Y(e) - 1).$$

Es folgt

$$\begin{aligned}
\mathbb{E}\Big(\sum_{j=1}^{2^n} H_{ij} \,\Big|\, R_i = (e_1,\ldots,e_\ell)\Big) &\le \mathbb{E}\Big(\sum_{e\in R_i} (Y(e)-1) \,\Big|\, R_i = (e_1,\ldots,e_\ell)\Big) \\
&= \sum_{k=1}^{\ell} \mathbb{E}(Y(e_k) - 1 \,|\, e_k \in R_i) \\
&= \sum_{k=1}^{\ell} \mathbb{E}\Big(\sum_{j\neq i} I_{\{e_k\in R_j\}} \,\Big|\, e_k \in R_i\Big) \\
&= \sum_{k=1}^{\ell} \mathbb{E}\Big(\sum_{j\neq i} I_{\{e_k\in R_j\}}\Big) \;\le\; \sum_{k=1}^{\ell} \mathbb{E}\, Y(e_k). \qquad (4.9)
\end{aligned}$$

Aus Symmetriegründen ist $\mathbb{E}\, Y(e)$ für alle Kanten e gleich, denn jeder Knoten enthält anfangs genau ein Paket und der Würfel sieht von allen Ecken aus betrachtet gleich aus.

Sei also $c' := \mathbb{E}Y(e)$. Da die Summe der Lasten $\sum_{e\in E} Y(e)$ gleich der Gesamtlänge der Routen $\sum_{i=1}^{2^n} |R_i|$ ist, gilt

$$c' \;=\; \frac{\mathbb{E}(\sum_{e\in E} Y(e))}{|E|} \;=\; \frac{|V|\,\mathbb{E}|R_1|}{|E|} \;=\; \frac{2^n\,\frac{n}{2}}{n\,2^n} \;=\; \frac{1}{2}. \qquad (4.10)$$

Aus (4.9) und (4.10) erhält man daher die Abschätzung

$$\mathbb{E}\Big(\sum_{j=1}^{2^n} H_{ij} \,\Big|\, R_i = (e_1,\ldots,e_\ell)\Big) \;\le\; \frac{\ell}{2} \;\le\; \frac{n}{2}. \qquad (4.11)$$

Mit (4.7), (4.12), der Bernstein-Chernoff Schranke (4.1) und dem Kopplungsargument in der anschließenden Bemerkung ergibt sich

$$\begin{aligned}
&\mathbb{P}\{V_i \ge (1+\delta)\tfrac{n}{2} \,|\, R_i = (e_1,\ldots,e_\ell)\} \\
&\quad\le\; \mathbb{P}\Big\{\sum_{j=1}^{2^n} H_{ij} \ge (1+\delta)\tfrac{n}{2} \,\Big|\, R_i = (e_1,\ldots,e_\ell)\Big\} \;\le\; \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^{\frac{n}{2}}
\end{aligned}$$

für alle möglichen Pfade $(e_1,\ldots,e_\ell)$. Mit dem Satz von der totalen Wahrscheinlichkeit folgt

$$\mathbb{P}\{V_i \ge (1+\delta)\tfrac{n}{2}\} \le \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^{\frac{n}{2}}. \qquad (4.12)$$

(Man beachte, dass diese Wahrscheinlichkeit nicht von der zuzustellenden Permutation π abhängt.) Abhängig von π ist lediglich die gemeinsame Verteilung der beiden Phasen.
Mit der Wahl $\delta = c - 3$ erhält man

$$\mathbb{P}\{V_i \ge (\tfrac{c}{2} - 1)\, n\} \le \left(\frac{e^{\frac{c-3}{2}}}{(c-2)^{\frac{c}{2}-1}}\right)^n \;=\; \left(\frac{\theta_c}{2}\right)^n.$$

Da jede Route höchstens die Länge n hat, folgt schließlich

$$\begin{aligned}\mathbb{P}\Big\{ \max_{1\le i\le 2^n} T_i \ge \tfrac{c}{2}n\Big\} &\le \mathbb{P}\Big\{ \max_{1\le i\le 2^n} V_i \ge (\tfrac{c}{2}-1)\Big\} \\ &\le 2^n\,\mathbb{P}\{V_1 \ge (\tfrac{c}{2}-1)\} \le \theta_c^n.\end{aligned}$$

Die Wahrscheinlichkeit, dass die erste Phase nach $(c/2)\,n$ Schritten noch nicht beendet ist, ist also höchstens θ_c^n. Da bei unserer Analyse die Richtung der Zeit keine Rolle gespielt hat, folgt aus Symmetriegründen (π ist eine Permutation!), dass auch die Wahrscheinlichkeit, dass die zweite Phase nach $(c/2)\,n$ Schritten noch nicht beendet ist, durch θ_c^n beschränkt ist. Die Wahrscheinlichkeit, dass eines der Pakete nach cn Schritten noch nicht seinen Bestimmungsort erreicht hat, ist also höchstens $\theta_c^n + \theta_c^n = 2\,\theta_c^n$.

5 Fingerabdrücke

Stellen wir uns vor, wir müssten entscheiden, ob zwei Objekte x und y aus einer Menge $\mathcal{U}$ gleich sind oder nicht. Ist die Menge $\mathcal{U}$ sehr groß, dann empfiehlt es sich, statt der Objekte x und y deren Bilder unter einer zufälligen Abbildung F von $\mathcal{U}$ in eine wesentlich kleinere Menge $\mathcal{V}$ zu überprüfen. Die Übereinstimmung der "Fingerabdrücke" $F(x)$ und $F(y)$ lässt sich in $O(\log|\mathcal{V}|)$ Schritten verifizieren, wohingegen die Komplexität des ursprünglichen Problems von der Ordnung $\log|\mathcal{U}|$ ist. Da aus der Übereinstimmung der Fingerabdrücke nicht zwingend auf die Gleichheit der Objekte geschlossen werden kann, liegt die Kunst bei dieser Technik in der geeigneten Wahl des Fingerabdrucks F. Die Verteilung der zufälligen Abbildung sollte so gewählt werden, dass die Wahrscheinlichkeit, dass zwei unterschiedliche Objekte denselben Fingerabdruck haben, gleichmäßig klein ist.

5.1 Die Technik von Freivalds

Nehmen wir an, wir sollen entscheiden, ob $AB = C$ für gegebene $n \times n$ Matrizen A, B und C über einem Körper $\mathbb{F}$. Die naheliegende Idee, die beiden Matrizen A und B miteinander zu multiplizieren und die Einträge des Produkts mit denen der Matrix C zu vergleichen, ist uns zu teuer (der schnellste bekannte deterministische Algorithmus zur Multiplikation zweier $n \times n$ Matrizen hat Laufzeit $O(n^{2.376\ldots})$). Eine alternative Lösung bietet die folgende auf Freivalds [8] zurückgehende Technik.

Probabilistischer Test der Gleichheit von Matrizen

Eingabe: Drei $n \times n$ Matrizen A, B und C

1) Wähle rein zufällig einen Vektor R aus $\mathcal{S}^n$, $\mathcal{S} \subset \mathbb{F}$ endlich.

2) Berechne die Vektoren $X = BR$, $Y = AX$ und $Z = CR$.

3) Ist $Y \neq Z$, dann verkünde "$AB \neq C$". Falls $Y = Z$, dann vermute "$AB = C$".

Zur Berechnung von X, Y und Z werden jeweils $O(n^2)$ Schritte benötigt. Die Ausgabe "$AB \neq C$" ist definitiv richtig. Das Urteil "$AB = C$" ist möglicherweise falsch, der nachfolgende Satz besagt aber, dass die Wahrscheinlichkeit hierfür gering ist.

Satz 5.1 *A, B und C seien $n \times n$ Matrizen über $\mathbb{F}$ mit $AB \neq C$ und R ein rein zufälliger Vektor aus $\mathcal{S}^n$, $\mathcal{S} \subset \mathbb{F}$ endlich. Dann gilt*

$$\mathbb{P}\{ABR = CR\} \leq \frac{1}{|\mathcal{S}|}.$$

Beweis. Die Matrix $D = AB - C$ ist nach Voraussetzung ungleich der Nullmatrix. Also existiert ein $d_{ik} \neq 0$. Falls $DR = 0$, dann gilt insbesondere $\sum_{j=1}^{n} d_{ij} R_j = 0$, also

$$R_k = -\frac{\sum_{j \neq k} d_{ij} R_j}{d_{ik}}. \tag{5.1}$$

Da die Komponenten $R_j, 1 \leq j \leq n$ unabhängige uniform auf $\mathcal{S}$ verteilte Zufallsvariablen sind, hat das Ereignis (5.1) höchstens Wahrscheinlichkeit $1/|\mathcal{S}|$ (man bedinge auf beliebige Werte der R_j, $j \neq k$).

Bemerkung. Man beachte, dass durch unabhängiges k-maliges Ziehen eines uniformen Vektors aus $\mathcal{S}^n$ die Wahrscheinlichkeit einer falschen Entscheidung auf $|\mathcal{S}|^{-k}$ gedrückt werden kann.

5.2 Das Testen der Gleichheit von Strings

Zwei Personen A und B an den Enden eines Nachrichtenkanals möchten zwei Strings (0-1 Folgen) der Länge n auf ihre Gleichheit überprüfen. Die Übertragung entlang des Kanals ist zuverlässig, aber kostspielig. Die Möglichkeit, dass Person A den String $a = a_1 \cdots a_n$ an Person B schickt und diese ihn dann auf Gleichheit mit $b = b_1 \cdots b_n$ überprüft, ist den beiden zu teuer. Das folgende randomisierte Verfahren reduziert die Übertragungskosten erheblich und fällt dennoch mit hoher Wahrscheinlichkeit das richtige Urteil.

Wir identifizieren die Strings a und b mit den Zahlen $\sum_{j=1}^n a_i 2^{n-j}$ und $\sum_{j=1}^n b_j 2^{n-j}$. Mit F_p bezeichnen wir die Funktion $F_p(x) = x \bmod p$ (hier und im Folgenden bezeichnet p stets eine Primzahl).

Randomisiertes Protokoll zum Testen der Gleichheit von Strings

1) Person A wählt rein zufällig eine Primzahl $X \leq k$ und berechnet $F_X(a)$.

2) Person A schickt X und $F_X(a)$ an Person B.

3) Person B berechnet $F_X(b)$ und vergleicht es mit $F_X(a)$.

4) Ist $F_X(a) \neq F_X(b)$, dann verkündet B, dass "$a \neq b$". Falls hingegen $F_X(a) = F_X(b)$, dann entscheidet sich B für die Meldung "$a = b$".

Stimmen die Strings a und b überein, dann ist die Meldung von Person B in jedem Fall korrekt. Die Wahrscheinlichkeit, dass B auf Grund des Indizes $F_X(a) = F_X(b)$ eine falsche Entscheidung trifft, ist dagegen gering:

Satz 5.2 *Seien $a, b \leq 2^n$ mit $a \neq b$, und sei X eine rein zufällige Primzahl $\leq k$, dann gilt*

$$\mathbb{P}\{F_X(a) = F_X(b)\} \;\leq\; \frac{n \log k}{k \log n} (\log 2 + o(1)) \text{ für } \min(n,k) \to \infty. \tag{5.2}$$

Wählt man $k = cn$ für ein $c > 1$, dann ist asymptotische obere Schranke in (5.2) gleich $(\log 2)/c$. Bei der Wahl $k = n^{1+\alpha}$ für ein $\alpha > 0$ erhält man gar eine obere Schranke der Ordnung $n^{-\alpha}$. In beiden Fällen ist die Anzahl der zu übertragenden Bits lediglich von der Ordnung $\log n$.

Beweis von Satz 5.2. Fixiere $a, b \leq 2^n$ mit $a \neq b$. Falls $F_p(a) = F_p(b)$, dann ist p Teiler von $|a-b|$. Da $|a-b| \leq 2^n$, folgt

$$\mathbb{P}\{F_X(a) = F_X(b)\} \leq \frac{\max_{1\leq j\leq 2^n} g(j)}{\pi(k)}, \tag{5.3}$$

wobei $g(j)$ die Anzahl der verschiedenen Primfaktoren von j ist, und $\pi(k)$ die Anzahl der Primzahlen $\leq k$. Die Dichte der Primzahlen wird durch den berühmten Primzahlsatz beschrieben (siehe etwa [3]). Dieser besagt, dass

$$\pi(k) \sim \frac{k}{\log k} \quad \text{für } k \to \infty. \tag{5.4}$$

Eine obere Schranke für $\max_{1\leq j\leq 2^n} g(j)$ erhält man aus der Beobachtung, dass $j \geq m!$, falls $g(j) \geq m$. Insbesondere gilt also $(\max_{1\leq j\leq 2^n} g(j))! \leq 2^n$. Mit Hilfe der Stirlingschen Formel erhält man

$$\max_{1\leq j\leq 2^n} g(j) \leq \log 2 \, \frac{n}{\log n}\,(1 + o(1)). \tag{5.5}$$

Die Behauptung von Satz 5.2 folgt nunmehr aus (5.3), (5.4) und (5.5).

Bemerkung. Die in diesem und dem vorigen Abschnitt vorgestellten Techniken sind in gewissem Sinn dual. Bei der Technik von Freivalds wählt man die Stelle, an der die Matrix über dem fixierten Körper $\mathbb{F}$ ausgewertet wird, zufällig. Beim probabilistischen Test der Gleichheit von Strings wertet man hingegen an einer festen Stelle aus, randomisiert dafür aber den Körper $\mathbb{Z}_p$.

5.3 Textsuche

Die Suche nach einem Ausdruck in einem Text ist ein klassisches Problem der Informatik ("Find/Change"). Gegeben seien ein langer und ein kürzerer String, $y = y_1 \cdots y_n$ und $z = z_1 \cdots z_m$, $m \leq n$. Die Aufgabe ist die Bestimmung aller Positionen, an denen der Ausdruck z im Text y vorkommt. Eine mögliche, aber teure Lösung ist, alle $y^{(i)} = y_i \cdots y_{i+m-1}$, $1 \leq i \leq n-m+1$ Bit für Bit mit z zu vergleichen (Laufzeit $O(mn)$). Der folgende zufällige Algorithmus von Karp und Rabin [15] basiert auf dieser plumpen Methode. Anstatt aber die $y^{(i)}$ mit z bitweise zu vergleichen, überprüft man, ob die Fingerabdrücke $F_X(y^{(i)})$ und $F_X(z)$ übereinstimmen, wobei X wiederum eine aus einem geeigneten Bereich zufällig gewählte Primzahl ist.

Randomisiertes Suchen eines Ausdrucks in einem Text

Eingabe: Zwei Strings $y = y_1 \cdots y_n$ und $z = z_1 \cdots z_m$, $m \leq n$.

1) Wähle eine rein zufällige Primzahl $X \leq k$ und berechne $F_X(z)$.
2) Für $i = 1, \ldots, n-m+1$
   - Berechne $F_X(y^{(i)})$ und vergleiche es mit $F_X(z)$.
   - Falls $F_X(y^{(i)}) = F_X(z)$, dann melde "z findet sich an Stelle i in y".

Ein wesentlicher Aspekt dieses Verfahrens ist die Tatsache, dass die Fingerabdrücke induktiv sehr einfach zu berechnen sind. Wenn wir $y^{(i)}$ mit $\sum_{j=1}^{m} y_{i+j-1} 2^{m-j}$ identifizieren, dann gilt

$$y^{(i+1)} = 2\,(y^{(i)} - y_i\, 2^{m-1}) + y_{i+m},$$

also

$$F_p(y^{(i+1)}) \;=\; (2F_p(y^{(i)}) - 2^m y_i + y_{i+m}) \bmod p.$$

Bei der sukzessiven Berechnung der Fingerabdrücke muss man also für jeden Update nur eine feste Anzahl arithmetischer Operationen durchführen. Auch die Wahrscheinlichkeit, dass der Algorithmus fälschlicherweise eine Übereinstimmung meldet, ist bei geeigneter Wahl von k gering.

Satz 5.3 *Für $k = mn^{1+\alpha}$, $\alpha > 0$ ist die Wahrscheinlichkeit, dass der randomisierte Algorithmus eine nicht vorhandene Übereinstimmung meldet, höchstens von der Ordnung $n^{-\alpha}$.*

Beweis. Der Algorithmus meldet eine nicht vorhandene Übereinstimmung, falls ein i existiert, so dass $F_X(y^{(i)}) = F_X(z)$, obwohl $y^{(i)} \neq z$. Eine falsche Ausgabe erfolgt also nur dann, wenn die zufällige Primzahl X Teiler von $\prod_{y^{(i)} \neq z} |z - y^{(i)}|$ ist. Da dieses Produkt durch 2^{mn} beschränkt ist, folgt die Behauptung von Satz 5.3 unmittelbar aus den Asymptotiken (5.4) und (5.5).

Bemerkung. Man kann den Algorithmus leicht in einen Las-Vegas-Algorithmus verwandeln, indem man bei einer durch die Fingerabdrücke indizierten potenziellen Übereinstimmung an der Stelle i, den Ausdruck $y^{(i)}$ Bit für Bit mit z vergleicht. Die erwartete Laufzeit dieses Las-Vegas-Algorithmus ist $O(n + \ell m)$, wobei ℓ die Anzahl der tatsächlichen Übereinstimmungen bezeichnet.

II Die Markovketten-Monte-Carlo-Methode

Wir wollen aus einer Menge $\mathcal{S}$ eine Stichprobe X mit Verteilung π ziehen. Dabei sei die Verteilung π nur bis auf die Normierungskonstante bekannt (man denke etwa an die uniforme Verteilung auf einer großen Menge unbekannter Kardinalität). Als *Markovketten-Monte-Carlo-Methode* (MCMC-Methode) bezeichnet man die folgende Vorgehensweise: Man konstruiert eine ergodische Markovkette auf $\mathcal{S}$ mit stationärer Verteilung π. Um einen (approximativ) nach π verteilten Wert zu generieren, lässt man die Markovkette dann so lange laufen bis die Verteilung des Zustands genügend nah an der Gleichgewichtsverteilung π ist. Zwei Fragen stellen sich dabei: Wie konstruiert man eine solche Markovkette und wie lange ist lang genug?

Wir erinnern zunächst an einige Grundbegriffe aus der Theorie der Markovketten. Sei $(X_k)_{k\geq 0}$ eine Markovkette mit abzählbarem Zustandsraum $\mathcal{S}$ und stochastischer Übergangsmatrix $P = (P_{xy})_{x,y\in\mathcal{S}}$, d.h.

$$\mathbb{P}\{X_{k+1} = x_{k+1} \,|\, X_0 = x_0, \ldots, X_k = x_k\} = P_{x_k x_{k+1}}$$

für alle $k \in \mathbb{N}_0$ und $x_0, \ldots, x_{k+1} \in \mathcal{S}$. Die Markovkette $(X_k)_{k\geq 0}$ heißt *irreduzibel*, falls

$$\forall\, x, y \in \mathcal{S}\ \exists\, m \in \mathbb{N}_0 \colon P^m_{xy} > 0$$

und *aperiodisch*, falls

$$ggT\,\{m \colon P^m_{xx} > 0\} = 1 \text{ für alle } x \in \mathcal{S}.$$

Man nennt eine Markovkette *ergodisch*, falls sie irreduzibel und aperiodisch ist, wenn also

$$\forall x, y, z \in \mathcal{S}\ \exists\, m \in \mathbb{N}_0 \colon P^m_{xz} > 0, P^m_{yz} > 0.$$

Ein Wahrscheinlichkeitsmaß π auf $\mathcal{S}$ heißt *stationäre Verteilung* oder *Gleichgewichtsverteilung* der Markovkette $(X_k)_{k\geq 0}$, falls

$$\pi(x) \;= \sum_{y\in\mathcal{S}} \pi(y) P_{yx} \text{ für alle } x \in \mathcal{S}.$$

Startet die Kette mit einer stationären Anfangsverteilung π, dann ist auch X_1 (und jedes andere X_k) nach π verteilt. Jede Markovkette mit endlichem Zustandsraum besitzt (mindestens)

eine stationäre Verteilung. Ist die Kette irreduzibel, dann ist die Gleichgewichtsverteilung eindeutig. Erfüllt die Markovkette $(X_k)_{k\geq 0}$ die *lokale Gleichgewichtsbedingung*

$$\pi(x)P_{xy} \;=\; \pi(y)P_{yx} \;\text{ für alle }\; x,y\in\mathcal{S}, \tag{A.1}$$

dann nennt man $(X_k)_{k\geq 0}$ *reversibel* bzgl. π. Insbesondere ist dann π eine stationäre Verteilung der Markovkette $(X_k)_{k\geq 0}$. Die MCMC-Methode basiert auf dem folgenden Grenzwertsatz (siehe etwa Theorem 1.8.3 in [21]).

Satz *Sei $(X_k)_{k\geq 0}$ eine ergodische Markovkette mit stationärer Verteilung π, dann gilt*

$$\lim_{k\to\infty} \mathbb{P}\{X_k = y \,|\, X_0 = x\} = \pi(y) \;\text{ für alle } x,y\in\mathcal{S}.$$

Eine ergodische Markovkette konvergiert also unabhängig von der Startverteilung $\mathcal{L}(X_0)$ in ihr (eindeutiges) Gleichgewicht.

6 Klassische Markovketten-Monte-Carlo-Verfahren

6.1 Der Metropolis-Algorithmus

Wie konstruiert man zu einer gegebenen Verteilung π auf einem endlichen Raum $\mathcal{S}$ eine Markovkette, die π als stationäre Verteilung besitzt? Der Metropolis-Algorithmus (oder auch Metropolis-Hastings-Algorithmus [11, 18]) ist eine Anleitung zum Entwurf einer solchen Markovkette. Dabei wählt man zunächst eine beliebige irreduzible "Referenzkette" auf $\mathcal{S}$ mit Übergangsmatrix $(Q_{xy})_{x,y\in\mathcal{S}}$. Die Übergangswahrscheinlichkeiten Q_{xy} werden nun so umgewichtet, dass man eine Markovkette mit π als stationärer Verteilung erhält. Für $x\neq y\in\mathcal{S}$ verkleinert man Q_{xy} zu $P_{xy}=\alpha_{xy}Q_{xy}$ derart, dass die Markovkette mit der Übergangsmatrix P die lokale Gleichgewichtsbedingung $(A.1)$ erfüllt. Mit dem Ansatz

$$\pi(x)P_{xy} \;\stackrel{!}{=}\; \min\left(\pi(x)Q_{xy}, \pi(y)Q_{yx}\right)$$

erhält man

$$P_{xy} = \begin{cases} \min\left(1, \dfrac{\pi(y)\,Q_{yx}}{\pi(x)\,Q_{xy}}\right) Q_{xy}, & \text{falls } x\neq y; \\[2ex] 1-\sum_{z\neq x} P_{xz}, & \text{falls } x=y. \end{cases} \tag{6.1}$$

Nach Konstruktion erfüllt P offenbar die lokale Gleichgewichtsbedingung $(A.1)$, insbesondere ist π die stationäre Verteilung der zugehörigen Markovkette. Man kann einen Übergang der P-Kette als zweistufiges Zufallsexperiment auffassen: Die Referenzkette schlägt zunächst mit Wahrscheinlichkeit Q_{xy} einen Übergang vom aktuellen Zustand x nach y vor. Anschließend wirft man eine Münze mit Erfolgswahrscheinlichkeit α_{xy}. Bei Erfolg wird der von der Referenzkette vorgeschlagene Übergang nach y akzeptiert, ansonsten verharrt man im momentanen Zustand x. Ist die Referenzkette *symmetrisch*, d.h. gilt $Q_{xy}=Q_{yx}$ für alle $x,y\in\mathcal{S}$, dann erhält man

$$\alpha_{xy} \;=\; \min\left(1, \frac{\pi(y)}{\pi(x)}\right),\; x\neq y.$$

Ein vorgeschlagener Übergang zu einem Zustand mit höherem Gewicht wird also stets akzeptiert.

Ein wesentlicher Aspekt des Metropolis-Algorithmus ist die Tatsache, dass man zur Konstruktion von P nur die Quotienten $\pi(y)/\pi(x)$ für $x \neq y \in \mathcal{S}$ mit $Q_{xy} > 0$ kennen muss. Die Ergodizität von P ist von Fall zu Fall klären. Dabei ist die Aperiodizität unproblematisch, sie lässt sich gegebenenfalls leicht einrichten. Die Irreduzibilität von P kann man in der Regel auf jene von Q zurückspielen.

Auch wenn man die Referenzkette Q grundsätzlich frei wählen kann, wird man mit Blick auf die Implementierung in der Praxis Dynamiken mit lokalen Übergängen bevorzugen.

Beispiel: Das Simulieren bedingter Verteilungen. Seien X_i, $1 \leq i \leq n$ unabhängig identisch verteilte Zufallsvariablen mit Werten in $\mathbb{N}_0$ und Verteilung $(p_x)_{x\in\mathbb{N}_0}$. Wir wollen die bedingte Verteilung von $(X_1, \ldots, X_n)$ gegeben das Ereignis $X_1 + \cdots + X_n = m,\ m \in \mathbb{N}_0$ simulieren. Der Zustandsraum ist in diesem Fall

$$\mathcal{S} = \Big\{ x = (x_1, \ldots, x_n) \in \mathbb{N}_0^n : \sum_{i=1}^{n} x_i = m \Big\}$$

und die zu simulierende Verteilung ist

$$\pi(x) = c_{m,n}\, p_{x_1} \cdots p_{x_n},\ x \in \mathcal{S},$$

wobei $c_{m,n}^{-1} = \mathbb{P}\{X_1 + \cdots + X_n = m\} = \sum_{y\in\mathcal{S}} p_{y_1} \cdots p_{y_n}$. Die Normierungskonstante $c_{m,n}$ ist nur in wenigen Fällen berechenbar (wenn die X_i Poisson oder geometrisch verteilte Zufallsvariablen sind).

Zur Konstruktion der Referenzkette wählen wir zunächst rein zufällig ein Paar (I, J) zweier verschiedener Komponenten aus $\{1, \ldots, n\}$. Ist $(I, J) = (i, j)$ und x der gegenwärtige Zustand, dann wird anschließend eine rein zufälliges U aus $\{0, \ldots, x_i + x_j\}$ gewählt. Ist $U = u$, dann wird x_i durch u, und x_j durch $x_i + x_j - u$ ersetzt. Als Übergangswahrscheinlichkeiten erhalten wir

$$Q_{xy} = \begin{cases} \binom{n}{2}^{-1} \dfrac{1}{x_i + x_j + 1}, & \text{falls } x_i \neq y_i,\ x_j \neq y_j \text{ und } x_k = y_k \text{ für alle } k \notin \{i, j\}; \\ 0\,, & \text{falls } |\{i : x_i \neq y_i\}| \geq 3; \\ 1 - \sum_{z\neq x} P_{xz}, & \text{falls } x = y. \end{cases}$$

Die Referenzkette ist offensichtlich symmetrisch, aperiodisch und irreduzibel. Der zugehörige Metropolis-Algorithmus akzeptiert einen vorgeschlagenen Übergang von x nach y mit Wahrscheinlichkeit 1, falls $\pi(y) \geq \pi(x)$, bzw. mit Wahrscheinlichkeit

$$\frac{\pi(y)}{\pi(x)} = \frac{p_{y_i} p_{y_j}}{p_{x_i} p_{x_j}}$$

falls $\pi(y) < \pi(x)$. Eine hinreichende Bedingung für die Irreduzibilität des Metropolis-Algorithmus ist

$$\{x \in \mathbb{N} \colon p_x > 0\} = \{1, \cdots, k\} \text{ für ein } k \in \mathbb{N} \cup \{\infty\}.$$

6.2 Der Gibbs-Sampler

Wie der Metropolis-Algorithmus ist auch der *Gibbs-Sampler* oder *heat bath algorithm* (siehe [9]) eine Anleitung zur Konstruktion einer Markovkette mit gegebener stationärer Verteilung π. Allerdings bedarf es dazu einer gewissen Struktur des Zustandsraums S. Wie im vorangegangenen Beispiel bestehe ein Zustand $x \in \mathcal{S}$ aus vielen Komponenten. Der Gibbs-Sampler passt dann den Zustand lokal an die stationäre Verteilung π an. Ist $x = (x_1, \ldots, x_n)$ der aktuelle Zustand, dann verfährt man wie folgt.

1) Wähle eine zufällige Komponente I gemäß einer Verteilung $q = (q_i)_{1 \le i \le n}$.

2) Ist $I = i$, dann ersetze die i-te Komponente x_i mit Wahrscheinlichkeit

$$r_{x_{-i},y_i} := \frac{\pi(x_1, \ldots, x_{i-1}, y_i, x_{i+1} \ldots, x_n)}{\pi(x_{-i})},$$

durch y_i (dabei ist $x_{-i} = \{(y_1, \ldots, y_n) \colon y_j = x_j \text{ für alle } j \neq i\}$). Die anderen Komponenten bleiben unverändert.

Bedingt auf $I = i$ hat der neue Zustand die Verteilung $\mathcal{L}(X \,|\, X \in x_{-i})$, wobei X eine Zufallsvariable mit Verteilung π bezeichnet. Man beachte, dass auch beim Gibbs-Sampler die Verteilung π nur bis auf die Normierungskonstante bekannt zu sein braucht. (In einem etwas allgemeineren Rahmen lassen sich der Gibbs-Sampler und der Metropolis-Algorithmus als Spezialfälle ein und desselben Verfahren auffassen, vgl. [10].) Für Zustände $x \neq y$, die sich nur in der i-ten Komponente unterscheiden, gilt

$$\pi(x)P_{xy} \;=\; \pi(x)\, q_i\, \frac{\pi(y)}{\pi(x_{-i})} \;=\; \pi(y)\, q_i\, \frac{\pi(x)}{\pi(y_{-i})} \;=\; \pi(y)P_{yx}.$$

Auch der Gibbs-Sampler erfüllt also die lokale Bilanzgleichung. Die Irreduzibilität des Gibbs-Samplers ist wiederum von Fall zu Fall zu klären. Für das Simulieren bedingter Verteilungen etwa ist er offensichtlich nicht geeignet.

Beispiel: Das Ising-Modell. Sei $D \subset \mathbb{Z}^d$ ein endliches Gebiet und

$$\mathcal{S} = \{-1, +1\}^D.$$

Jeder Gitterpunkt in D enthält also einen "Spin" ± 1. Einer Spinkonfiguration $x = (x_i)_{i \in D} \in \mathcal{S}$ wird die *Energie*

$$H(x) \;=\; -\sum_{i,j \in D} v_{ij}\, x_i x_j \;-\; b \sum_{i \in D} x_i$$

zugeordnet. Dabei beschreibt (v_{ij}) die Interaktion zwischen den Gitterpunkten und b die Wirkung eines externen Feldes. Als *Gibbsverteilung* bezeichnet man das Wahrscheinlichkeitsmaß

$$\pi(x) \;=\; \frac{1}{Z} \exp\big(-\beta H(x)\big),\; x \in \mathcal{S}.$$

Der Parameter β ist die inverse Temperatur, die Normierungskonstante $Z = Z(b, \beta)$ bezeichnet man auch als Zustandssumme. Befindet sich das System im thermodynamischen Gleichgewicht, dann ist $\pi(x)$ die Wahrscheinlichkeit für den Zustand x. Bei niedrigen Temperaturen befindet sich das System mit hoher Wahrscheinlichkeit in Zuständen mit geringer Energie. Bei steigender Temperatur nimmt die Unordnung des Systems zu, im Extremfall $\beta = 0$ ist π die Gleichverteilung auf $\mathcal{S}$. Die Zustandssumme Z ist nur in den Dimensionen $d = 1$ und 2 exakt bestimmbar, im Allgemeinen aber unzugänglich. In der Funktion $Z(b, \beta)$ ist die physikalisch relevante Information enthalten, wie etwa die mittlere Energie

$$\sum_{x \in \mathcal{S}} H(x)\, \pi(x) \;=\; -\,\frac{\partial \log Z}{\partial \beta}$$

oder die mittlere Magnetisierung

$$\sum_{x \in \mathcal{S}} \Big(\sum_{i \in D} x_i \Big) \pi(x) \;=\; \frac{1}{\beta}\, \frac{\partial \log Z}{\partial b}.$$

Wählt man beim zugehörigen Gibbs-Sampler als q die Gleichverteilung auf D, dann ergeben sich die Übergangswahrscheinlichkeiten

$$P_{xy} \;=\; \frac{1}{|D|}\, \frac{1}{1 + \exp(\beta(H(y) - H(x)))},$$

für $x \neq y$ mit $y_{-i} = x_{-i}$ für ein $i \in D$. Zum Vergleich: Beim Metropolis-Algorithmus mit der "Nächste-Nachbarn-Irrfahrt" auf $\{-1, +1\}^D$ als Referenzkette erhält man

$$P_{xy} \;=\; \frac{1}{|D|}\, \min\Big(1,\, \exp(-\beta(H(y) - H(x))\Big).$$

Die Ungleichung $z/(1+z) \leq \min(1, z)$ für $z \in \mathbb{R}_0^+$ zeigt, dass der Metropolis-Algorithmus hier (geringfügig) mobiler als der Gibbs-Sampler ist.

Man beachte, dass sich die Übergangswahrscheinlichkeiten effizient berechnen lassen: Für $x, y \in \mathcal{S}$ mit $x_{-i} = y_{-i}$ gilt

$$H(x) - H(y) \;=\; \pm 2 \Big(\sum_{j \neq i} (v_{ij} + v_{ji})\, x_j \;+\; b \Big). \tag{6.2}$$

Beide Markovketten sind offensichtlich ergodisch, versagen aber auf Grund ihrer lokalen Natur im interessanten Temperaturbereich, in dem Phasenübergänge stattfinden. Ein Verfahren, dass auch globale Änderungen der Spinkonfigurationen zulässt, ist der Swendsen-Wang-Algorithmus (siehe [30]).

7 Exaktes Simulieren von Verteilungen

Eine zentrales Problem bei der Markovketten-Monte-Carlo-Methode ist die Frage, wie viele Schritte man die Markovkette laufen lassen muss, bis die stationäre Verteilung "hinreichend gut" approximiert ist. Im Allgemeinen lässt sich durch bloßes Hinschauen nicht entscheiden, ob sich das System nahezu im Gleichgewicht befindet. Wir werden später Methoden zur Abschätzung der notwendigen Laufzeit am Rande des Kapitels 8 behandeln. In diesem Kapitel beschäftigen wir uns mit in jüngerer Zeit entwickelten Methoden, die es erlauben, eine Stichprobe zu generieren, die nicht nur approximativ sondern exakt die vorgegebene Verteilung besitzt.

In den folgenden Abschnitten bezeichnet $(X_k)_{k\geq 0}$ stets eine ergodische Markovkette mit endlichem Zustandsraum $\mathcal{S}$, Übergangsmatrix $P = (P_{xy})$ und stationärer Verteilung π.

7.1 Das Verfahren von Propp und Wilson

Das Konzept, das dem Verfahren "coupling from the past" ("Koppeln aus der Vergangenheit") von Propp und Wilson [22] zu Grunde liegt, ist das folgende: Man konstruiere eine Folge von Zufallsvariablen $(Y_k)_{k\geq 0}$ mit der Eigenschaft

$$\begin{aligned} &1) \quad Y_k \stackrel{d}{=} X_k \text{ für alle } k \in \mathbb{N}_0. \\ &2) \quad Y_k \stackrel{f.s.}{\to} Y_\infty \text{ für } k \to \infty. \end{aligned}$$

Offenbar gilt dann $Y_\infty \stackrel{d}{=} \pi$. Konvergenz einer Folge auf dem endlichen Raum $\mathcal{S}$ bedeutet, dass die Folgenglieder letztendlich konstant sind, d.h.

$$Y_k \stackrel{f.s.}{\to} Y_\infty \iff K := \inf\{k \geq 0 : Y_k = Y_{k+j} \text{ für alle } j \geq 0\} \stackrel{f.s.}{<} \infty.$$

Wegen $Y_K \stackrel{f.s.}{=} Y_\infty$ folgt $Y_K \stackrel{d}{=} \pi$. Wenn es also gelingt, auf Grund der Beobachtung von $Y_0, \ldots, Y_k$ festzustellen, ob $k \geq K$, dann kann man auf diese Weise eine exakt nach π verteilte Stichprobe generieren.

Seien F_k, $k \geq 1$ unabhängige und identisch verteilte zufällige Abbildungen von $\mathcal{S}$ nach $\mathcal{S}$ mit

$$\mathbb{P}\{F_k(x) = y\} = P_{xy} \text{ für alle } x, y \in \mathcal{S}. \tag{7.1}$$

Man beachte, dass durch (7.1) die Verteilung von F_k, also die Wahrscheinlichkeiten

$$\mathbb{P}\{F_k(x_i) = y_i, 1 \leq i \leq r\}, \quad r \in \mathbb{N},\ x_i, y_i \in \mathcal{S} \tag{7.2}$$

noch nicht festgelegt sind. Setze $G_0 := Id_{\mathcal{S}}$ und

$$G_k := F_k \circ \cdots \circ F_1 = F_k \circ G_{k-1},\ k \geq 1.$$

Für beliebiges $x \in \mathcal{S}$ ist dann $(G_k(x))_{k\geq 0}$ eine in x startende Markovkette mit Übergangsmatrix P. Anstelle der G_k betrachten wir nun die zufälligen Abbildungen $H_0 := Id_{\mathcal{S}}$ und

$$H_k := F_1 \circ \cdots \circ F_k = H_{k-1} \circ F_k,\ k \geq 1. \tag{7.3}$$

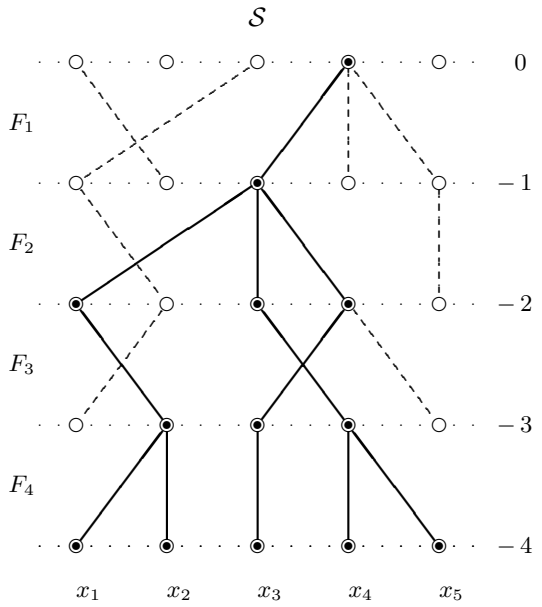

Abb. 7.1: In dieser Realisierung der F_k ist $T = 4$, $R_1 = \{x_1, x_3, x_4\}$, $R_2 = R_3 = \{x_3, x_4\}$ und $R_4 = \{x_4\}$.

Für $x \in \mathcal{S}$ ist die Folge $(H_k(x))_{k\geq 0}$ in aller Regel keine Markovkette. Da die zufälligen Abbildungen F_k aber unabhängig und identisch verteilt sind, gilt

$$H_k(x) \stackrel{d}{=} G_k(x) \text{ für alle } x \in \mathcal{S} \text{ und } k \in \mathbb{N}_0. \tag{7.4}$$

Bezeichnet $R_k := H_k(\mathcal{S})$ das Bild der zufälligen Abbildung H_k, dann gilt nach Konstruktion der H_k in (7.3)

$$R_k \subset R_{k-1},\ k \geq 1.$$

Der Limes

$$R_\infty := \lim_{k\to\infty} R_k = \bigcap_{k\geq 1} R_k$$

ist also wohl definiert. Insbesondere gilt, dass wenn H_k eine konstante Abbildung ist, dann sind dies auch H_ℓ, $\ell \geq k$. Als *(Rückwärt)-Kopplungszeit* T bezeichnet man die zufällige Größe

$$T := \min\{k \geq 0 \colon |R_k| = 1\}.$$

Satz 7.1 *Gilt $T < \infty$ f.s., dann folgt $R_T \stackrel{d}{=} \pi$.*

Beweis. Sei $x \in \mathcal{S}$. Für $k \geq T$ gilt $H_k(x) = R_T$. Wegen $T < \infty$ f.s. folgt

$$H_k(x) \stackrel{f.s.}{\to} R_T \text{ für } k \to \infty.$$

Andererseits gilt nach (7.4) und der Ergodizität von P

$$H_k(x) \stackrel{d}{=} G_k(x) \stackrel{d}{\to} \pi \text{ für } k \to \infty.$$

Bemerkungen.

- Die Verteilung von T (insbesondere, ob $T < \infty$ f.s.) hängt von der Verteilung (7.2) der F_k ab.
- Es ist von entscheidender Bedeutung aus der Vergangenheit zu koppeln: Sei T' die Vorwärtskopplungszeit,

$$T' := \min\{k \geq 0 \colon |G_k(\mathcal{S})| = 1\}.$$

  Zwar gilt $T \stackrel{d}{=} T'$ (Übung!), im Allgemeinen ist $G_{T'}(\mathcal{S})$ aber nicht nach π verteilt. Man mache sich das an einem Beispiel klar, bei dem es Zustände mit einem eindeutigen Vorgänger gibt, also $y \in \mathcal{S}$ mit $|\{x \in \mathcal{S} \colon P_{xy} > 0\}| = 1$.

Unser erstes Resultat bezüglich der Verteilung der Kopplungszeit T besagt, dass die Funktion $\mathbb{P}\{T > \cdot\}$ submultiplikativ ist. Die Verteilung von T hat also einen exponentiell abfallenden Schwanz (es sei denn, es gilt $T = \infty$ f.s.).

Lemma 7.2 *Für alle $m, n \in \mathbb{N}_0$ gilt*

$$\mathbb{P}\{T > m+n\} \leq \mathbb{P}\{T > m\}\,\mathbb{P}\{T > n\}. \tag{7.5}$$

Beweis. Setze

$$H_\ell^k := \begin{cases} F_{k+1} \circ \ldots \circ F_\ell, & \text{falls } \ 0 \leq k < \ell, \\ Id_{\mathcal{S}}, & \text{falls } \ 0 \leq k = \ell; \end{cases} \tag{7.6}$$

und beachte, dass

$$H_m^0 \circ H_{m+n}^m = H_{m+n} \ \text{ und } \ H_{m+n}^m \stackrel{d}{=} H_n.$$

Es folgt

$$\begin{aligned} \mathbb{P}\{T > m+n\} &= \mathbb{P}\{|H_{m+n}(\mathcal{S})| > 1\} \\ &= \mathbb{P}\{|H_m \circ H_{m+n}^m(\mathcal{S})| > 1\} \\ &\leq \mathbb{P}\{|H_m(\mathcal{S})| > 1,\, |H_{m+n}^m(\mathcal{S})| > 1\} \\ &= \mathbb{P}\{T > m\}\,\mathbb{P}\{T > n\}. \end{aligned}$$

Der unabhängige Fall. Wir betrachten nun zunächst den Fall, dass

$$\mathbb{P}\{F_1(x_i) = y_i, 1 \leq i \leq r\} \;=\; \prod_{i=1}^{r} \mathbb{P}\{F_1(x_i) = y_i\}$$

für paarweise verschiedene x_i. Auf Grund der Ergodizität von P existiert zu jedem $z \in \mathcal{S}$ ein k_0, so dass $P_{xz}^{k_0} > 0$ für alle $x \in \mathcal{S}$. Es folgt

$$\begin{aligned} \mathbb{P}\{T \leq k_0\} \;&=\; \mathbb{P}\{T' \leq k_0\} \;=\; \mathbb{P}\{|G_{k_0}(\mathcal{S})| = 1\} \\ &\geq\; \mathbb{P}\{G_{k_0}(x) = \{z\} \text{ für alle } x \in \mathcal{S}\} \;\geq\; \prod_{x \in \mathcal{S}} \mathbb{P}\{G_{k_0}(x) = z\} \;=\; \prod_{x \in \mathcal{S}} P_{xz}^{k_0} \;> 0\,. \end{aligned}$$

Bezüglich der zweiten Ungleichung beachte man, dass verschmelzende Markovketten eher in ein und demselben Zustand landen als unabhängige Markovketten. Mit (7.5) folgt

$$\mathbb{P}\{T \le jk_0\} \;\ge\; 1 - \mathbb{P}\{T > k_0\}^j \;\to\; 1 \;\text{ für } j \to \infty.$$

Im unabhängigen Fall gilt also $T < \infty$ f.s. Allerdings ist das Verfahren in diesem Fall höchst ineffizient. Zum einen kann T extrem große Werte annehmen, zum anderen muss jeweils die gesamte Liste $H_k(x)$, $x \in \mathcal{S}$ abgespeichert werden und auch der Aufwand für eine Aktualisierung beträgt $O(|\mathcal{S}|)$.

Monotones Monte-Carlo. Wir betrachten nun den Fall, dass die Menge $\mathcal{S}$ mit einer Halbordnung versehen ist. Zudem nehmen wir an, dass ein kleinstes Element $\hat{0}$ und ein größtes Element $\hat{1}$ existieren, d.h. $\hat{0} \le x \le \hat{1}$ für alle $x \in \mathcal{S}$. Eine Markovkette $(X_k)_{k\ge 0}$ mit Übergangsmatrix P auf $\mathcal{S}$ heißt *monoton*, falls

$$x \le y \implies P(x,\cdot\,) \preceq P(y,\cdot\,). \tag{7.7}$$

Dabei heißt ein Wahrscheinlichkeitsmaß μ auf dem halbgeordneten Raum $\mathcal{S}$ stochastisch kleiner als das Wahrscheinlichkeitsmaß ν ("$\mu \preceq \nu$"), falls $\mu(I) \ge \nu(I)$ für alle Mengen $I \subset \mathcal{S}$ mit der Eigenschaft $z \in I$, $z' \le z \Rightarrow z' \in I$.
Wir nehmen weiter an, dass die zufälligen Abbildungen F_k von der folgenden Gestalt sind,

$$F_k = \phi(\,\cdot\,, U_k),\; k \ge 1. \tag{7.8}$$

Dabei sind die U_k, $k \ge 1$ unabhängige und identisch verteilte Zufallsvariablen mit Werten in einer beliebigen (messbaren) Menge $\mathcal{U}$ und $\phi : \mathcal{S} \times \mathcal{U} \to \mathcal{S}$ ist eine messbare (deterministische) Funktion mit der Eigenschaft

$$x \le y \implies \phi(x,u) \le \phi(y,u) \;\text{ für alle }\; u \in \mathcal{U}.$$

Die *monotone Übergangsregel* (7.8) erlaubt es, Übergänge von unterschiedlichen Zuständen simultan so zu generieren, dass die Ordnung der Zustände erhalten bleibt. (Die Existenz einer monotonen Übergangsregel ist eine geringfügig stärkere Forderung als die Monotonie der Markovkette. Ist $\mathcal{S}$ linear geordnet, dann sind die beiden Forderungen äquivalent.) Insbesondere gilt unter der Annahme (7.8), dass

$$T = \min\{k \ge 0 \colon H_k(\hat{0}) = H_k(\hat{1})\}.$$

Um den Wert von T zu bestimmen muss man also nur die Bilder der beiden extremalen Elemente unter der Folge der zufälligen Abbildungen H_k, $k \ge 1$ betrachten.

Der Propp-Wilson-Algorithmus

Für $k = 1, 2, 4, 8, \ldots$ berechne

$$\begin{aligned} H_k(\hat{0}) &= \phi(\phi(\ldots\phi(\phi(\hat{0}, U_k), U_{k-1}), \ldots, U_2), U_1); \\ H_k(\hat{1}) &= \phi(\phi(\ldots\phi(\phi(\hat{1}, U_k), U_{k-1}), \ldots, U_2), U_1). \end{aligned}$$

Setze $T^* := \min\{k \in \{1, 2, 4, 8, \ldots\} \colon H_k(\hat{0}) = H_k(\hat{1})\}$.

Ausgabe: $H_{T^*}(\hat{0})$.

Wegen $T^* \geq T$ folgt aus Satz 7.1, dass $H_{T^*}(\hat{0}) \stackrel{d}{=} \pi$. Der Propp-Wilson-Algorithmus leistet also das Gewünschte.

Bemerkungen.

- Die Zahl der zu simulierenden Übergänge ist durch

$$2(1 + 2 + 4 + \cdots + T^*) \leq 4T^* \leq 8T$$

  nach oben beschränkt. Anstelle der Listen $H_k(x)$, $x \in \mathcal{S}$ müssen lediglich die Werte U_j, $1 \leq j \leq k$ abgespeichert werden.

- Es ist wesentlich, bei jedem Durchlauf der Schleife dieselben Realisierungen der U_k zu verwenden, um eine Verzerrung der Stichprobe zu vermeiden. Der Algorithmus darf auch nicht einfach aus Ungeduld abgebrochen und neu gestartet werden, weil zwischen T^* und der Ausgabe $H_{T^*}(\hat{0})$ eine Abhängigkeit besteht.

Beispiel: Attraktive Spinsysteme. Im Ising-Modell (vgl. Abschnitt 6.2) besitzt die Menge der Spinkonfigurationen eine natürliche Halbordnung. Für $x, y \in \mathcal{S} = \{-1, +1\}^D$ definiere

$$x \leq y \;:\Longleftrightarrow\; x_i \leq y_i \text{ für alle } i \in D.$$

Die extremalen Elemente sind $\hat{0} = \{-1\}^D$ bzw. $\hat{1} = \{+1\}^D$. Für $x \in \mathcal{S}$ seien $x^i_+, x^i_- \in \mathcal{S}$ die Spinkonfigurationen mit Komponenten

$$(x^i_{+/-})_j := \begin{cases} +1/-1, & \text{falls } j = i; \\ x_j, & \text{sonst.} \end{cases} \tag{7.9}$$

Definiere $\phi : \mathcal{S} \times [0,1] \times D \to \mathcal{S}$ durch

$$\phi(x, v, i) := \begin{cases} x^i_+, & \text{falls } v \leq \dfrac{\pi(x^i_+)}{\pi(x_{-i})}; \\ x^i_-, & \text{sonst.} \end{cases} \tag{7.10}$$

Ist $U_k = (V_k, I_k)$, wobei V_k uniform verteilt ist auf $[0,1]$ und I_k uniform in der Menge D, unabhängig von V_k, dann generieren die zufällige Abbildungen $\phi(\,\cdot\,, U_k)$ die Übergänge des Gibbs-Samplers. Die Funktion ϕ ist monoton, falls

$$\frac{\pi(x^i_+)}{\pi(x_{-i})} \quad \text{bzw.} \quad \frac{\pi(x^i_+)}{\pi(x^i_-)}$$

monoton wachsend ist in x. Solche Spinsysteme nennt man *attraktiv*. Beim Ising Modell mit auf die nächsten Nachbarn beschränkter Interaktion $v_{ij} = v\, 1_{\{|j-i|=1\}}$ ist (vgl. (6.2))

$$\log \frac{\pi(x^i_+)}{\pi(x^i_-)} \;=\; 2\beta\Big(b + 2v \sum_{j:\,|j-i|=1} x_j\Big).$$

Zum Gibbs-Sampler existiert also eine monotone Übergangsregel, falls $v \geq 0$ (*ferromagnetischer Fall*).

Wir wollen nun die Laufzeit des Propp-Wilson-Algorithmus im monotonen Fall abschätzen und erinnern zunächst an die Definition des *Totalvariationsabstands* zweier Wahrscheinlichkeitsmaße μ und ν auf einem (abzählbaren) Raum $\mathcal{S}$,

$$d_{TV}[\mu,\nu] := \sup_{B\subset\mathcal{S}} |\mu(B)-\nu(B)|. \tag{7.11}$$

Das Supremum wird für die Menge $B=\{x\in\mathcal{S}\colon \mu(x)>\nu(x)\}$ angenommen,

$$d_{TV}[\mu,\nu] \;=\; \sum_{x\in\mathcal{S}} (\mu(x)-\nu(x))^+ \;=\; \frac{1}{2}\sum_{x\in\mathcal{S}} |\mu(x)-\nu(x)|. \tag{7.12}$$

Für $m\in\mathbb{N}_0$ definieren wir nun

$$d(m) \;:=\; \sup_{\mu,\nu} d_{TV}[\,\mathbb{P}_\mu\{X_m\in\cdot\,\},\mathbb{P}_\nu\{X_m\in\cdot\,\}]. \tag{7.13}$$

Die Größe $d(m)$ misst, wie verschieden die Verteilung von X_m in Abhängigkeit von der Anfangsverteilung sein kann. Nach Definition von $d(m)$ und der Dreiecksungleichung gilt

$$\sup_\mu d_{TV}[\,\mathbb{P}_\mu\{X_m\in\cdot\,\},\pi] \;\le\; d(m) \;\le\; 2\sup_\mu d_{TV}[\,\mathbb{P}_\mu\{X_m\in\cdot\,\},\pi].$$

So wie die Folge $\mathbb{P}\{T>n\}$ ist auch d submultiplikativ:

Lemma 7.3 *Für alle $m,n\in\mathbb{N}_0$ gilt*

$$d(m+n)\le\, d(m)\,d(n).$$

Beweis. Seien $m,n\in\mathbb{N}_0$ und $B\subset\mathcal{S}$. Nach dem Satz von der totalen Wahrscheinlichkeit gilt

$$\begin{aligned}
&\mathbb{P}_\mu\{X_{m+n}\in B\} - \mathbb{P}_\nu\{X_{m+n}\in B\}\\
&= \sum_{x\in B}\Big(\mathbb{P}_\mu\{X_m=x\}-\mathbb{P}_\nu\{X_m=x\}\Big)\Big(\mathbb{P}_x\{X_n\in B\}-\min_{y\in\mathcal{S}}\mathbb{P}_y\{X_n\in B\}\Big)\\
&\le \Big(\max_{z\in\mathcal{S}}\mathbb{P}_z\{X_n\in B\}-\min_{y\in\mathcal{S}}\mathbb{P}_y\{X_n\in B\}\Big)\sum_{x\in B}\Big(\mathbb{P}_\mu\{X_m=x\}-\mathbb{P}_\nu\{X_m=x\}\Big)^+\\
&\le \max_{y,z\in\mathcal{S}} d_{TV}[\mathbb{P}_y\{X_n\in\cdot\,\},\mathbb{P}_z\{X_n\in\cdot\,\}]\cdot d_{TV}[\,\mathbb{P}_\mu\{X_m\in\cdot\,\},\mathbb{P}_\nu\{X_m\in\cdot\,\}]\\
&\le d(n)\,d(m).
\end{aligned}\tag{7.14}$$

Da die rechte Seite von (7.14) weder von B noch von μ oder ν abhängt, folgt die Behauptung nach Definition des Totalvariationsabstands in (7.11).

Satz 7.4 *Der Zustandsraum $\mathcal{S}$ sei halbgeordnet mit extremalen Elementen $\hat0$ bzw. $\hat1$ und es existiere eine monotone Übergangsregel (7.8). Dann gilt für alle $m\in\mathbb{N}_0$,*

$$d(m) \;\le\; \mathbb{P}\{T>m\} \;\le\; \ell_{max}\,d(m), \tag{7.15}$$

wobei ℓ_{max} die maximale Länge einer Kette in $\mathcal{S}$ bezeichnet.

Beweis. Sei $\ell(x)$ die maximale Länge einer Kette in $\mathcal{S}$ mit x als größtem Element. Für $x < y$ ist $\ell(y) - \ell(x) \geq 1$. Folglich gilt

$$\begin{aligned}
\mathbb{P}\{T > m\} &= \mathbb{P}\{H_m(\hat{0}) < H_m(\hat{1})\} \\
&= \mathbb{E}\, I_{\{H_m(\hat{0}) < H_m(\hat{1})\}} \\
&\leq \mathbb{E}\,\big[\ell(H_m(\hat{1})) - \ell(H_m(\hat{0}))\big] \\
&= \sum_{x\in\mathcal{S}} \ell(x)\Big(\mathbb{P}\{H_m(\hat{1}) = x\} - \mathbb{P}\{H_m(\hat{0}) = x\}\Big) \\
&\leq \ell_{max} \sum_{x\in\mathcal{S}} \Big(\mathbb{P}\{H_m(\hat{1}) = x\} - \mathbb{P}\{H_m(\hat{0}) = x\}\Big)^+ \\
&= \ell_{max}\, d_{TV}[\,\mathbb{P}_{\hat{1}}\{X_m \in \cdot\,\}, \mathbb{P}_{\hat{0}}\{X_m \in \cdot\,\}] \;\leq\; \ell_{max}\, d(m).
\end{aligned}$$

Zum Nachweis der anderen Ungleichung seien Z und Z' nach μ bzw. ν verteilte Zufallsvariablen, unabhängig von den F_k, $k \geq 1$. Wegen $\{T \leq m\} \subset \{H_m(Z) = H_m(Z')\}$ gilt für beliebiges $x \in \mathcal{S}$

$$\begin{aligned}
|\,\mathbb{P}_\mu\{X_m = x\} - \mathbb{P}_\nu\{X_m = x\}| &= |\,\mathbb{P}\{H_m(Z) = x\} - \mathbb{P}\{H_m(Z') = x\}| \\
&= |\,\mathbb{P}\{H_m(Z) = x,\, T > m\} - \mathbb{P}\{H_m(Z') = x,\, T > m\}| \\
&\leq \mathbb{P}\{H_m(Z) = x,\, T > m\} + \mathbb{P}\{H_m(Z') = x,\, T > m\}.
\end{aligned}$$

Summation über $x \in \mathcal{S}$ ergibt

$$2\, d_{TV}[\,\mathbb{P}_\mu\{X_m \in \cdot\,\}, \mathbb{P}_\nu\{X_m \in \cdot\,\}] \;\leq\; 2\,\mathbb{P}\{T > m\}.$$

Die *Mischzeit* τ_{mix} einer Markovkette ist eine Kenngröße, die beschreibt, wie schnell die Kette gegen ihre Gleichgewichtsverteilung konvergiert. Es gibt verschiedene verwandte Definitionen dieses Begriffs, eine davon ist

$$\tau_{mix} := \min\{k \geq 0 \colon d(k) \leq \tfrac{1}{2}\}.$$

Die Wahl des Schwellenwerts $\frac{1}{2}$ ist dabei relativ willkürlich, denn nach Lemma 7.3 gilt

$$d(j\,\tau_{mix}) \leq d(\tau_{mix})^j \leq 2^{-j} \;\text{ für } j \geq 1. \tag{7.16}$$

Der nachfolgende Satz gibt eine obere Schranke für die erwartete Kopplungszeit der Markovkette mittels ihrer Mischzeit.

Satz 7.5 *Unter den Voraussetzungen von Satz 7.4 gilt*

$$\mathbb{E}\,T \leq\; 2\,\tau_{mix}(1 + \lceil \log_2 \ell_{max} \rceil).$$

Beweis. Es gilt

$$\mathbb{E}\,T \;=\; \sum_{n=0}^{\infty} \mathbb{P}\{T > n\} \;=\; \sum_{j=0}^{\infty} \sum_{i=0}^{m-1} \mathbb{P}\{T > jm + i\} \;\leq\; m \sum_{j=0}^{\infty} \mathbb{P}\{T > jm\}$$

für $m \in \mathbb{N}$. Mit Lemma 7.2 und Satz 7.4 folgt

$$\mathbb{E}\,T \;\le\; m\sum_{j=0}^{\infty}\mathbb{P}\{T>m\}^j \;=\; \frac{m}{1-\mathbb{P}\{T>m\}} \;\le\; \frac{m}{1-\ell_{max}\,d(m)}. \tag{7.17}$$

Für $m = \tau_{mix}(1+\lceil\log_2 \ell_{max}\rceil)$ gilt nach (7.16)

$$d(m) \;\le\; 2^{-(1+\log_2 \ell_{max})} \;=\; \frac{1}{2\,\ell_{max}}\,. \tag{7.18}$$

Aus (7.17) und (7.18) folgt

$$\mathbb{E}\,T \;\le\; 2\,\tau_{mix}(1+\lceil\log_2 \ell_{max}\rceil).$$

Die Sätze 7.4 und 7.5 liefern brauchbare obere Schranken für die Kopplungszeit T, falls ℓ_{max} klein ist im Vergleich zu $|\mathcal{S}|$. Beim Ising-Modell ist $\ell_{max} = |D| = \log_2|\mathcal{S}|$. In solchen Fällen sind rasch mischende Markovketten auch schnell koppelnd.

7.2 Der Algorithmus von Fill

Der Algorithmus von Fill [6] generiert wie der Propp-Wilson-Algorithmus eine exakt nach einem vorgegebenen Wahrscheinlichkeitsmaß π verteilte Stichprobe. Das Verfahren basiert auf dem so genannten *rejection sampling*. Dabei generiert man einen nach π verteilten Wert wie folgt: Sei ν ein Wahrscheinlichkeitsmaß auf $\mathcal{S}$ und c eine Konstante ≥ 1, so dass

$$\pi(x) \;\le\; c\nu(x) \text{ für alle } x \in \mathcal{S}. \tag{7.19}$$

1) Ziehe einen nach ν verteilten Wert X.
2) Ist $X = x$, dann werfe eine Münze mit Erfolgswahrscheinlichkeit $\dfrac{\pi(x)}{c\nu(x)}$. Bei Erfolg akzeptiere den vorgeschlagenen Wert x, ansonsten lehne ihn ab und gehe zurück zu 1).

Die Korrektheit des Verfahrens ist leicht verifiziert: Es gilt

$$\mathbb{P}\{X = x,\, X \text{ wird akzeptiert}\} = \nu(x)\,\frac{\pi(x)}{c\nu(x)} = \frac{\pi(x)}{c},$$

also

$$\mathbb{P}\{X \text{ wird akzeptiert}\} = c^{-1} \;\text{ und }\; \mathbb{P}\{X = x \,|\, X \text{ wird akzeptiert}\} = \pi(x),\; x \in \mathcal{S}.$$

Die Aufgabe besteht somit darin, ein Wahrscheinlichkeitsmaß ν und eine Konstante c zu finden, so dass $c\nu$ die (nicht vollständig bekannte) Verteilung π majorisiert und ein Münzwurfexperiment mit Erfolgswahrscheinlichkeit $\pi(x)/(c\nu(x))$ durchgeführt werden kann.

Wir betrachten wieder den Fall, dass der Zustandsraum $\mathcal{S}$ der Markovkette halbgeordnet ist mit extremalen Elementen $\hat{0}$ und $\hat{1}$. Sei

$$\widetilde{P}_{xy} \;:=\; \frac{\pi(y)}{\pi(x)}\,P_{yx}, \quad x, y \in \mathcal{S} \tag{7.20}$$

die Übergangsmatrix der *zeitumgekehrten* Kette. Aus (7.20) folgt unmittelbar, dass

$$\widetilde{P}^k_{xy} \;=\; \frac{\pi(y)}{\pi(x)}\, P^k_{yx} \;\text{ für alle } x,y\in\mathcal{S} \text{ und } k\in\mathbb{N}_0, \tag{7.21}$$

und dass π eine stationäre Verteilung von $\widetilde{P}$ ist. Ist die Markovkette mit Übergangsmatrix P ergodisch, dann ist es auch die zeitumgekehrte Kette.

Wir nehmen an, dass für die zeitumgekehrte (!) Markovkette eine monotone Übergangsregel $(\tilde{\phi},\widetilde{U})$ existiert (vgl. (7.8)). Wir können also mittels der zufälligen Abbildungen

$$\widetilde{G}_k = \tilde{\phi}(\ldots\tilde{\phi}(\tilde{\phi}(\,\cdot\,,\widetilde{U}_1),\widetilde{U}_2),\ldots,\widetilde{U}_k),\;\; k\geq 1 \tag{7.22}$$

gekoppelte Vorwärtspfade der$\widetilde{P}$-Kette generieren, so dass die Halbordnung der Zustände erhalten bleibt.

Beim Algorithmus von Fill simuliert man nun zunächst einen im Zustand $\hat{0}$ startenden Pfad $(X_j)_{0\leq j\leq k}$ der P-Kette. Diesen Pfad fasst man dann als einen in X_k startenden und in $\hat{0}$ endenden Pfad $(\widetilde{G}_j(X_k))_{0\leq j\leq k}$ der zeitumgekehrten $\widetilde{P}$-Kette auf. In einer zweiten Phase generiert man bedingt auf den in $\hat{0}$ endenden Pfad $(\widetilde{G}_j(X_k))_{0\leq j\leq k}$ den Pfad $(\widetilde{G}_j(\hat{1}))_{0\leq j\leq k}$. Verschmelzen die beiden Pfade der zeitumgekehrten Kette, dann wird der vorgeschlagene Wert X_k akzeptiert, ansonsten wird er abgelehnt.

Der Algorithmus von Fill

Für $k=1,2,4,8\ldots$ wiederhole unabhängig die folgende Prozedur bis eine Ausgabe erfolgt.

1) Simuliere einen in $\hat{0}$ startenden Pfad $(X_0,\ldots,X_k)$ der Markovkette mit Übergangsmatrix P.

2) Ist $X_j = x_j$ für $0\leq j\leq k$, dann generiere einen in $\hat{1}$ startenden Pfad $(\widetilde{Y}_j)_{0\leq j\leq k}$ entsprechend der folgenden Markovschen Dynamik (setze $\mathbf{x}:=(x_j)_{0\leq j\leq k}$),

$$\mathbb{P}_{\mathbf{x}}\{\,\widetilde{Y}_{j+1}=y' \,|\, \widetilde{Y}_j=y\} := \mathbb{P}\{\,\tilde{\phi}(y,\widetilde{U}_1)=y' \,|\, \tilde{\phi}(x_{k-j},\widetilde{U}_1)=x_{k-j-1}\},\;\; y,y'\in\mathcal{S}. \tag{7.23}$$

3) Ist $\widetilde{Y}_k=\hat{0}$, dann gebe X_k aus.

Satz 7.6 *Der Algorithmus von Fill terminiert f.s. in endlicher Zeit und gibt einen nach π verteilten Wert aus.*

Beweis. Die Verteilung ν_k des vorgeschlagenen Werts X_k hat Gewichte

$$\nu_k(x) \;=\; \mathbb{P}_{\hat{0}}\{X_k=x\} \;=\; P^k_{\hat{0}x},\;\; x\in\mathcal{S}.$$

Nach (7.21) gilt

$$\pi(x) \;=\; \frac{\pi(\hat{0})}{\widetilde{P}^k_{x\hat{0}}}\, P^k_{\hat{0}x}$$

und mit der Monotonie der $\widetilde{P}$-Kette folgt

$$\pi(x) \;\leq\; \frac{\pi(\hat{0})}{\widetilde{P}^k_{\hat{1}\hat{0}}}\, P^k_{\hat{0}x} =: \; c_k\nu_k(x),\;\; x\in\mathcal{S}.$$

Die Verteilung des vorgeschlagenen Werts erfüllt also die Bedingung (7.19) mit $c_k = \pi(\hat{0})/\widetilde{P}^k_{\hat{1}\hat{0}}$.

Wir zeigen nun, dass bedingt auf das Ereignis $\{X_k = x\}$ Schritt 2) des Algorithmus einem Münzwurfexperiment mit Erfolgswahrscheinlichkeit $\pi(x)/(c_k\nu_k(x))$ entspricht. Nach (7.22) und (7.23) gilt

$$\mathcal{L}_{\mathbf{x}}((\widetilde{Y}_j)_{0\le j\le k}) \;=\; \mathcal{L}((\widetilde{G}_j(\hat{1}))_{0\le j\le k} \,|\, \widetilde{G}_j(x_k) = x_{k-j},\, 0\le j\le k) \tag{7.24}$$

und aus (7.21) folgt

$$\mathbb{P}\{\widetilde{G}_j(x_k) = x_{k-j},\, 0\le j\le k \,|\, \widetilde{G}_k(x_k) = \hat{0}\} \;=\; \mathbb{P}\{(X_j)_{0\le j\le k} = \mathbf{x} \,|\, X_k = x_k\}. \tag{7.25}$$

Mit dem Satz von der totalen Wahrscheinlichkeit ergibt sich aus (7.24) und (7.25)

$$\begin{aligned}
&\mathbb{P}\{\,\widetilde{Y}_k = \hat{0} \,|\, X_k = x\} \\
&\quad= \sum_{\mathbf{x}:\, x_k = x} \mathbb{P}_{\mathbf{x}}\{\,\widetilde{Y}_k = \hat{0}\}\, \mathbb{P}\{(X_j)_{0\le j\le k} = \mathbf{x} \,|\, X_k = x\} \\
&\quad= \sum_{\mathbf{x}:\, x_k = x} \mathbb{P}\{\widetilde{G}_k(\hat{1}) = \hat{0} \,|\, \widetilde{G}_j(x_k) = x_{k-j},\, 0\le j\le k\} \\
&\qquad\qquad\qquad\qquad \cdot\, \mathbb{P}\{\widetilde{G}_j(x_k) = x_{k-j},\, 0\le j\le k \,|\, \widetilde{G}_k(x_k) = \hat{0}\} \\
&\quad= \mathbb{P}\{\widetilde{G}_k(\hat{1}) = \hat{0} \,|\, \widetilde{G}_k(x) = \hat{0}\}.
\end{aligned} \tag{7.26}$$

Angesichts der Monotonie der $\widetilde{P}$-Kette erhält man aus (7.26) unter Beachtung von (7.21)

$$\mathbb{P}\{\,\widetilde{Y}_k = \hat{0} \,|\, X_k = x\} \;=\; \frac{\widetilde{P}^k_{\hat{1}\hat{0}}}{\widetilde{P}^k_{x\hat{0}}} \;=\; \frac{\pi(x)\,\widetilde{P}^k_{\hat{1}\hat{0}}}{\pi(\hat{0})\,P^k_{\hat{0}x}} \;=\; \frac{\pi(x)}{c_k\nu_k(x)}.$$

Damit ist die Korrektheit des Verfahrens bewiesen. Begzüglich des f.s. Terminierens des Algorithmus von Fill beachte man, dass auf Grund der Ergodizität von $\widetilde{P}$

$$c_k^{-1} = \frac{\widetilde{P}^k_{\hat{1}\hat{0}}}{\pi(\hat{0})} \;\longrightarrow\; 1 \;\text{ für } k\to\infty.$$

Die Konvergenz der Akzeptanzwahrscheinlichkeit gegen 1 impliziert insbesondere, dass die Prozedur f.s. nur endlich oft wiederholt wird.

Bemerkungen.

- Im Gegensatz zum Propp-Wilson-Algorithmus kann der Algorithmus von Fill von einem ungeduldigen Benutzer abgebrochen werden, ohne dass die Stichprobe verzerrt wird.

- Hinsichtlich der Laufzeit und des Speicherplatzbedarfs sind die Verfahren von Fill und von Propp-Wilson in etwa vergleichbar (siehe die ausführliche Diskussion in [6]).

Aus den Rechnungen ist schwer ersichtlich, wieso sich beim in Schritt 2) durchgeführten Experiment die gewünschten Erfolgswahrscheinlichkeiten ergeben. Die folgende Diskussion soll als Erklärung dienen: Y und Z seien unabhängige Zufallsvariablen, und das Paar (Y', Z') habe Verteilung

$$\mathcal{L}(Y', Z') = \mathcal{L}(Y, Z \,|\, Y \in B),$$

insbesondere also $Z' \stackrel{d}{=} Z$. Wie kann man die Verteilung $\mathcal{L}(Y', Z')$ generieren? Eine Möglichkeit besteht darin, so lange unabhängige Kopien (Y_i, Z_i), $i = 1, 2, \ldots$ von (Y, Z) zu generieren bis das Ereignis $\{Y_i \in B\}$ eintritt, d.h.

$$(Y'', Z'') := (Y_N, Z_N) \text{ mit } N := \min\{i \geq 1 : Y_i \in B\}$$

hat die gewünschte Verteilung. In der Situation des Algorithmus von Fill ist

$$Y = (\widetilde{U}_j)_{1 \leq j \leq k},\ \{Y \in B\} = \{\widetilde{G}_k(\mathcal{S}) = \{\hat{0}\}\} \text{ und } Z \stackrel{d}{=} \pi.$$

Dabei beachte man, dass

$$\{\widetilde{G}_k(\mathcal{S}) = \{\hat{0}\}\} \subset \{\widetilde{G}_k(Z) = \hat{0}\} \text{ und } \mathcal{L}(Z \,|\, \widetilde{G}_k(Z) = \hat{0}) = \mathcal{L}(X_k).$$

In Schritt 1) wird also der Pfad $(\widetilde{G}_j(Z))_{0 \leq j \leq k}$ auf $\{\widetilde{G}_k(Z) = \hat{0}\}$ bedingt. Das erspart die Simulation von Y_i's, die von vorneherein zum Misserfolg verdammt sind.

8 Approximatives Zählen

In diesem Kapitel beschäftigen wir uns mit dem Problem des Abzählens der Elemente einer Menge $\mathcal{S}_n$ von kombinatorischen Objekten. In seiner exakten Form ist dieses Problem in der Regel unzugänglich, da $N_n := |\mathcal{S}_n|$ typischerweise exponentiell groß ist in n. Man ist daher an effizienten Algorithmen interessiert, die mit einer geringen Fehlerwahrscheinlichkeit eine approximative Lösung des Problems liefern. Gesucht sind zufällige Algorithmen, die zu gegebenem $\varepsilon, \delta > 0$ einen (zufälligen) Wert $\widehat{N}_n$ liefern, so dass

$$\mathbb{P}\Big\{(1+\varepsilon)^{-1}\widehat{N}_n \le N_n \le (1+\varepsilon)\widehat{N}_n\Big\} \;\ge\; 1-\delta. \tag{8.1}$$

In der Sprache der Statistik ist das Intervall $[(1+\varepsilon)^{-1}\widehat{N}_n, (1+\varepsilon)\widehat{N}_n]$ ein Konfidenzbereich für die unbekannte Größe N_n zum Sicherheitsniveau $1-\delta$. Ist die Laufzeit zur Bestimmung des Punktschätzers $\widehat{N}_n$ polynomial beschränkt in n, ε^{-1} und $\log \delta^{-1}$, dann nennt man das Verfahren ein *fully polynomial randomized approximation scheme* (FPRAS). Ein Verfahren, das bei polynomialer Laufzeit in n und ε^{-1} die Forderung (8.1) für ein $\delta < \frac{1}{2}$ erfüllt, lässt sich stets zu einem FPRAS modifizieren: Nimmt man den Median von m unabhängigen Kopien $\widehat{N}_{n,i}$, $1 \le i \le m$ von $\widehat{N}_n$ als Schätzer für N_n, dann fällt die Irrtumswahrscheinlichkeit exponentiell in m (vgl. das Argument zu (1.3)).

Das Problem des approximativen Zählens ist eng verbunden mit dem des Generierens einer rein zufälligen Stichprobe aus $\mathcal{S}_n$. Oft ist es relativ einfach, mit Hilfe einer nahezu uniform verteilten Stichprobe die approximative Kardinalität der Menge zu bestimmen (dann, wenn die kombinatorischen Strukturen "selbst-reduzibel" sind). Da $|\mathcal{S}_n|$ exponentiell groß ist, kann die Markovketten-Monte-Carlo-Methode aber nur effizient sein, wenn die Anzahl der zu simulierenden Schritte der Markovkette sehr viel kleiner ist als $|\mathcal{S}_n|$. Man muss also versuchen, Markovketten zu entwerfen, die bereits nach Auskundschaften eines nur winzigen Bruchteils des Zustandsraums sich nahezu im (uniformen) Gleichgewicht befinden. Markovketten mit dieser Eigenschaft nennt man *rasch mischend*.

8.1 Ein Rucksackproblem

Zur Illustration der Ideen (und der möglichen Schwierigkeiten) beim approximativen Zählen mittels der Markovketten-Monte-Carlo-Methode betrachten wir das folgende Problem. Gegeben sei ein Vektor $a = (a_1, \ldots, a_n) \in \mathbb{N}^n$ und eine natürliche Zahl b. Gesucht ist die Anzahl $N_n = N(a,b)$ der Elemente $x = (x_1, \ldots, x_n) \in \{0,1\}^n$ mit

$$\langle a, x\rangle \;:=\; \sum_{i=1}^{n} a_i x_i \;\le\; b. \tag{8.2}$$

Denken wir uns a_i als die Größe des i-ten von n von Gegenständen, dann ist N_n die Anzahl der Möglichkeiten, einen Rucksack der Kapazität b zu packen. (Unter *dem* Rucksackproblem versteht man das folgende Optimierungsproblem: Man maximiere $\langle v, x\rangle$ unter der Nebenbedingung (8.2), wobei v_i der "Wert" des i-ten Gegenstands ist.) Ein klassischer Monte-Carlo-Zugang zur Lösung unseres Rucksackproblems wäre, unabhängige Vektoren Y_j, $1 \le j \le m$ rein zufällig aus $\{0,1\}^n$ zu ziehen und N_n durch

$$\widetilde{N}_{n,m} \;:=\; \frac{\sum_{j=1}^{m} I\{\langle a, Y_j\rangle \le b\}}{m}\; 2^n$$

zu schätzen. Das starke Gesetz der großen Zahlen besagt, dass $\widetilde{N}_{n,m} \xrightarrow{f.s.} N_n$ für $m \to \infty$. Man braucht also m nur groß genug zu wählen, um N_n beliebig gut zu approximieren. In der Praxis scheitert dieses Verfahren jedoch kläglich, zumindest dann, wenn $N_n \ll 2^n$. Ist beispielsweise $a = (1, \ldots, 1)$ und $b = n/3$, dann ist der Schätzer $\widetilde{N}_{n,m}$ für polynomiales $m = m_n$ mit sehr hoher Wahrscheinlichkeit 0, und dass, obwohl $N(a, b)$ exponentiell groß ist in n.

Wir beschreiben nun eine Lösung dieses Rucksackproblems mittels der Markovketten-Monte-Carlo-Methode. Sei dazu $(X_k)_{k\geq 0}$ die Markovkette mit Zustandsraum $\mathcal{S}_n := \{x \in \{0,1\}^n : \langle a, x\rangle \leq b\}$ und der folgenden Übergangsdynamik. Wir werfen zunächst eine faire Münze. Zeigt die Münze Kopf, dann verharren wir im gegenwärtigen Zustand. Bei Zahl flippen wir eine rein zufällige Komponente, allerdings nur, falls der resultierende 01-Vektor wieder ein Element in $\mathcal{S}_n$ ist. Die zugehörigen Übergangswahrscheinlichkeiten sind

$$P_{xy} = \begin{cases} (2n)^{-1}, & \text{falls } |x-y| = 1; \\ 0, & \text{falls } |x-y| > 1; \\ 1 - \sum_{z \in \mathcal{S}_n \setminus \{x\}} P_{xz}, & \text{falls } x = y; \end{cases} \qquad x, y \in \mathcal{S}_n. \tag{8.3}$$

Die Markovkette $(X_k)_{k\geq 0}$ vollführt also eine Art Irrfahrt auf dem durch die Hyperebene $\{x \in \{0,1\}^n : \langle a, x\rangle = b\}$ beschnittenen n-dimensionalen Würfel. Die Möglichkeit, in einem Zustand verweilen zu können, sorgt dafür, dass die Markovkette aperiodisch ist. Zudem ist $(X_k)_{k\geq 0}$ irreduzibel, denn jedes Paar von Zuständen kann über den Zustand $\mathbf{0} := (0, \ldots, 0)$ miteinander kommunizieren. Die Kette ist also ergodisch. Auf Grund der Symmetrie von P ist die Gleichverteilung die stationäre Verteilung von $(X_k)_{k\geq 0}$. Um also ein nahezu uniform verteiltes Element aus $\mathcal{S}_n$ zu generieren, mauss man die Kette nur "ausreichend lang" laufen lassen.

Unsere eigentliche Aufgabe war jedoch die Bestimmung der Kardinalität der Lösungsmenge $\mathcal{S}_n$. Hierbei hilft ein *Schachtelungsprinzip*, das man so oder in ähnlicher Weise häufig anwenden kann. Wir fixieren den Vektor a, variieren aber die Kapazität b. Setze $b_0 := 0$ und

$$b_i := \min\Big(\sum_{j=1}^{i} a_{(j)},\, b\Big), \quad 1 \leq i \leq n,$$

wobei $a_{(j)}$ das j-kleinste der $a_1, \ldots, a_n$ bezeichnet. Sei $\mathcal{S}_{n,i} := \{x \in \{0,1\}^n : \langle a, x\rangle \leq b_i\}$ die Lösungsmenge des zu a und b_i gehörenden Rucksackproblems. Dann gilt $\mathcal{S}_{n,0} = \{\mathbf{0}\}$,

$$\mathcal{S}_{n,i-1} \subset \mathcal{S}_{n,i}, \quad 1 \leq i \leq n$$

und $\mathcal{S}_{n,n} = \mathcal{S}_n$. Wir schreiben nun N_n als

$$N_n \;=\; \prod_{i=1}^{n} \frac{|\mathcal{S}_{n,i}|}{|\mathcal{S}_{n,i-1}|}.$$

Ist $x \in \mathcal{S}_{n,i} \setminus \{\mathbf{0}\}$, dann erhält man durch das Flippen einer 1 an einer Stelle j mit $a_j x_j = \max_{1\leq k\leq n} a_k x_k$ (= Entfernen eines Gegenstands maximaler Größe) eine Lösung $y \in \mathcal{S}_{n,i-1}$. Folglich gilt

$$|\mathcal{S}_{n,i-1}| \;\leq\; |\mathcal{S}_{n,i}| \;\leq\; (n+1)|\mathcal{S}_{n,i-1}|, \quad 1 \leq i \leq n. \tag{8.4}$$

Die Größen $\beta_i := |\mathcal{S}_{n,i-1}| \,/\, |\mathcal{S}_{n,i}|$, $1 \le i \le n$ sind also gleichmässig nach oben und unten beschränkt und lassen sich daher gut schätzen. Jedes β_i schätzt man nun unabhängig durch nahezu uniformes Ziehen unter Verwendung der Markovkette $(X_k^{b_i})_{k\ge 0}$, dem Analogon zu $(X_k)_{k\ge 0}$ auf dem Raum $\mathcal{S}_{n,i}$. Jede dieser n Markovketten wird in $\mathbf{0}$ startend unabhängig jeweils m Mal für t Schritte simuliert. Setze

$$\hat\beta_i := \frac{1}{m}\sum_{j=1}^{m} I\{X_{t,\,j}^{b_i} \in \mathcal{S}_{n,i-1}\}, \quad 1 \le i \le n \tag{8.5}$$

und

$$\widehat{N}_n := \frac{1}{\hat\beta_1 \cdots \hat\beta_n}.$$

Dabei sind $X_{t,\,j}^{b_i}$, $1 \le j \le m$ unabhängige Kopien von $X_t^{b_i}$ (zur Wahl von $m, t \in \mathbb{N}$ später mehr). Die Zufallsvariablen $m\hat\beta_i$, $1 \le i \le n$ sind binomial-verteilt zum Parameter $(m, \mathbb{E}\hat\beta_i)$. Nehmen wir für den Moment an, dass

$$\mathbb{E}\hat\beta_i \;=\; \mathbb{P}\{X_t^{b_i} \in \mathcal{S}_{n,i-1}\} \;\stackrel{!}{=}\; \frac{|S_{n,i-1}|}{|S_{n,i}|} \;=\; \beta_i. \tag{8.6}$$

Mit der oberen Schranke aus (8.4) folgt dann

$$\frac{\mathbb{V}\mathrm{ar}\,\hat\beta_i}{(\mathbb{E}\hat\beta_i)^2} \;=\; \frac{\beta_i(1-\beta_i)}{m\beta_i^2} \;\le\; \frac{n}{m}. \tag{8.7}$$

Aus der Unabhängigkeit der $\hat\beta_i$ erhält man unter Verwendung von (8.6) und (8.7)

$$\begin{aligned}\frac{\mathbb{V}\mathrm{ar}(\widehat{N}_n^{-1})}{(\mathbb{E}[\widehat{N}_n^{-1}])^2} \;&=\; \frac{\mathbb{E}[\widehat{N}_n^{-2}] - (\mathbb{E}[\widehat{N}_n^{-1}])^2}{(\mathbb{E}[\widehat{N}_n^{-1}])^2} \;=\; \left(\prod_{i=1}^{n}\frac{\mathbb{E}\hat\beta_i^2}{(\mathbb{E}\hat\beta_i)^2}\right) - 1 \\ &=\; \prod_{i=1}^{n}\left(1 + \frac{\mathbb{V}\mathrm{ar}\,\hat\beta_i}{(\mathbb{E}\hat\beta_i)^2}\right) - 1 \;\le\; \Big(1+\frac{n}{m}\Big)^n - 1.\end{aligned}$$

Für $m \ge 37\varepsilon^{-2}n^2$ und $0 < \varepsilon < 1$ gilt $(1+\frac{n}{m})^n - 1 \le \frac{\varepsilon^2}{36}$ und somit

$$\frac{\mathbb{V}\mathrm{ar}(\widehat{N}^{-1})}{(\mathbb{E}[\widehat{N}^{-1}])^2} \;\le\; \frac{\varepsilon^2}{36}. \tag{8.8}$$

Unter der Annahme (8.6) gilt $\mathbb{E}[\widehat{N}_n^{-1}] = N_n^{-1}$. Daher ist

$$\begin{aligned}\mathbb{P}\Big\{(1+\tfrac{\varepsilon}{2})^{-1}\widehat{N}_n \le N_n \le (1+\tfrac{\varepsilon}{2})\widehat{N}_n\Big\} \;&=\; \mathbb{P}\Big\{-\tfrac{\varepsilon}{2+\varepsilon}\,N_n^{-1} \le \widehat{N}_n^{-1} - N_n^{-1} \le \tfrac{\varepsilon}{2}\,N_n^{-1}\Big\} \\ &\ge\; \mathbb{P}\Big\{\,|\widehat{N}_n^{-1} - \mathbb{E}[\widehat{N}_n^{-1}]\,| \le \tfrac{\varepsilon}{3}\,\mathbb{E}[\widehat{N}_n^{-1}]\Big\}.\end{aligned}$$

Mit der Chebyshev-Ungleichung und der Abschätzung (8.8) folgt

$$\mathbb{P}\Big\{(1+\tfrac{\varepsilon}{2})^{-1}\widehat{N}_n \le N_n \le (1+\tfrac{\varepsilon}{2})\widehat{N}_n\Big\} \;\ge\; 1 - \frac{9\,\mathbb{V}\mathrm{ar}(\widehat{N}_n^{-1})}{\varepsilon^2(\mathbb{E}[\widehat{N}_n^{-1}])^2} \;\ge\; \frac{3}{4}\,. \tag{8.9}$$

Bei der Herleitung von (8.9) haben wir in (8.6) fälschlicherweise die Annahme $\mathbb{E}\hat{\beta}_i = \beta_i$ getroffen (jedes endliche t erzeugt einen Bias). Man kann aber offensichtlich $t = t(\varepsilon)$ so groß wählen, dass die Abschätzung (8.9) gültig bleibt, wenn man $\varepsilon/2$ durch ε ersetzt. Die Frage, ob die Markovketten $(X_k^{b_i})_{k\geq 0}$ rasch mischend sind, d.h. ob $t(\varepsilon)$ polynomial beschränkt ist in ε^{-1}, ist ein offenes Problem. Man weiß also nicht, ob unser MCMC-Verfahren zur approximativen Lösung des Rucksackproblems ein FPRAS ist.

8.2 Selbstmeidende Pfade

Ein selbstmeidender Pfad im Gitter $\mathbb{Z}^d$ ist ein Pfad einer gewöhnlichen Irrfahrt, der jeden Gitterpunkt höchstens einmal besucht. Selbstmeidende Pfade sind das einfachste mathematische Modell für die Anordnung langer Polymerketten (ein solches Molekül besteht aus bis zu 10^5 Monomeren). Es gibt zahlreiche durch empirische Untersuchungen und Heuristiken gestützte Vermutungen über das Verhalten selbstmeidender Pfade. Mit Bestimmtheit (im mathematisch strengen Sinn) weiß man aber nur sehr wenig, gerade in den von den Anwendungen her interessanten niedrigen Dimensionen $2 \leq d \leq 4$. (In hohen Dimensionen reduziert sich die Einschränkung "selbstmeidend" im Wesentlichen darauf, dass "unmittelbare Umkehr" untersagt ist.) Grundlegende Probleme sind die Frage nach der Zahl der selbstmeidenden Pfade der Länge n, sowie die Bestimmung der Charakteristiken eines typischen Pfades, etwa des mittleren quadratischen Abstands des Endpunkts vom Ursprung.

Sei $\mathcal{S}_n$, $n \in \mathbb{N}_0$ die Menge der im Ursprung startenden selbstmeidenden Pfade der Länge n. Wir bezeichnen mit $|s|$ die Länge eines selbstmeidenden Pfades s und schreiben $s \prec s'$, falls s' den Pfad s fortsetzt, falls also $s_i = s'_i,\ 0 \leq i \leq |s|$.

Die genaue Zahl $N_n := |\mathcal{S}_n|$ der selbstmeidenden Pfade der Länge n kennt man nicht. Eine erste Näherung liefert das folgende Resultat.

Proposition 8.1 *Es existiert ein $\mu = \mu_d \in [d, 2d-1]$ mit*

$$\lim_{n\to\infty} \frac{1}{n} \log N_n \;=\; \log \mu. \tag{8.10}$$

Im Schnitt gibt es also etwa μ Möglichkeiten, einen selbstmeidenden Pfad um einen Schritt zu verlängern. Man nennt μ die Zusammenhangskonstante (*connective constant*), ihr genauer Wert ist nicht bekannt. Sei die Funktion f definiert durch die Gleichung $N_n = \mu^n f(n)$, $n \in \mathbb{N}_0$. Bezüglich des asymptotischen Verhaltens des Korrekturterms $f(n)$ hat man die folgende Vermutung:

$$f(n) \sim \begin{cases} A_d\, n^{\gamma_d - 1}, & d = 2, 3; \\ A_4\, (\log n)^{\frac{1}{4}}, & d = 4; \\ A_d, & d \geq 5, \end{cases} \tag{8.11}$$

mit dimensionsabhängigen Konstanten A_d und γ_d. Im Unterschied zur Zusammenhangskonstante μ ist aber nicht einmal die Existenz der kritischen Exponenten γ_d bewiesen. Andere kritische Exponenten sind verbunden mit der "Rückkehrwahrscheinlichkeit" eines selbstmeidenden Pfades und dem mittleren quadratischen Abstand des Endpunkts vom Ursprung.

Was die Analyse der Charakteristiken selbstmeidender Pfade so schwierig macht, ist die Tatsache, dass so etwas wie eine selbstmeidende Irrfahrt, also ein stochastischer Prozess

$(Y_k)_{k\geq 0}$ mit Werten in $\bigcup_{n\geq 0}\mathcal{S}_n$ und der Eigenschaft

$$\mathbb{P}\{Y_k \prec Y_\ell\} = 1 \text{ für alle } k<\ell \quad \text{und} \quad \mathbb{P}\{Y_k = s\} > 0 \text{ für alle } s\in\mathcal{S}_k$$

nicht existiert. Der Grund hierfür ist, dass ein selbstmeidender Pfad in eine Falle geraten kann, in der keine Fortsetzung mehr möglich ist.

Beweis von Prop. 8.1. Da ein selbstmeidender Pfad nicht zu dem zuletzt besuchten Gitterpunkt zurückkehren kann, gilt $N_{n+1}\leq (2d-1)N_n$, $n\geq 1$. Andererseits ist ein Pfad, der in jeder Komponente monoton wächst, offensichtlich selbstmeidend. Da sich bei einem solchen Pfad zu jedem Zeitpunkt eine der d Komponenten erhöht, gilt $N_n\geq d^n$. Falls also der Grenzwert μ existiert, dann ist $\mu\in[d,2d-1]$.

Ein selbstmeidender Pfad der Länge $m+n$ besteht aus zwei selbstmeidenden Pfaden der Länge m bzw. n. Es gilt also

$$N_{m+n}\leq N_m N_n.$$

Die Existenz von μ ergibt sich aus dem folgenden Lemma angewandt auf die subadditive Folge $c_n=\log N_n$.

Lemma 8.2 *Sei c_n, $n\in\mathbb{N}$ eine Folge reeller Zahlen mit*

$$c_{m+n}\leq c_m + c_n. \tag{8.12}$$

Dann gilt

$$\lim_{n\to\infty}\frac{c_n}{n} = \inf_{k\geq 1}\frac{c_k}{k}.$$

Beweis. Offensichtlich gilt

$$\liminf_{n\to\infty}\frac{c_n}{n}\geq \inf_{k\geq 1}\frac{c_k}{k}.$$

Es bleibt also zu zeigen, dass

$$\limsup_{n\to\infty}\frac{c_n}{n} \;\leq\; \frac{c_k}{k} \text{ für alle } k\geq 1. \tag{8.13}$$

Zerlege $n=\ell_n k + r_n$ mit $0\leq r_n\leq k-1$. Wegen (8.12) gilt (setze $c_0:=0$)

$$c_n \;\leq\; \ell_n\, c_k + c_{r_n} \;\leq\; (n-r_n)\,\frac{c_k}{k} + \max_{0\leq i\leq k-1} c_i.$$

Division durch n und anschließender Grenzübergang $n\to\infty$ ergeben die Behauptung (8.13).

Wie simuliert man einen rein zufälligen selbstmeidenden Pfad der Länge n? Die naheliegende Idee ist, eine ergodische Markovkette auf $\mathcal{S}_n$ mit der Gleichverteilung als stationärer Verteilung zu konstruieren. Man muss dabei allerdings in jedem Schritt die Änderung großer Teilstücke des Pfades erlauben, denn es ist bekannt, dass eine Markovkette auf $\mathcal{S}_n$ nicht irreduzibel sein kann, solange lediglich "lokale" Änderungen zugelassen sind (siehe [17]).

Wir diskutieren hier einen alternativen Zugang, bei der die Länge der Pfade variiert: Konstruiert wird eine Markovkette $(X_k^{(n)})_{k\geq 0}$ auf dem Raum $\overline{\mathcal{S}}_n=\bigcup_{i=0}^n\mathcal{S}_i$ der selbstmeidenden Pfade mit maximaler Länge n. Die Dynamik der Kette lässt sich wie folgt beschreiben. Man

wählt zunächst rein zufällig eine der $2d$ mit dem Endpunkt des gegenwärtigen Pfades $s \in \overline{\mathcal{S}}_n$ verbundenen Kanten. Stimmt die zufällig gewählte Kante mit der letzten Kante des Pfades s überein, so wird die Kante aus dem Pfad entfernt. Setzt die zufällig gewählte Kante den Pfad s zu einem selbstmeidenden Pfad der Länge $|s|+1 \le n$ fort, dann werfen wir eine Münze mit Erfolgswahrscheinlichkeit $\beta_{|s|+1} > 0$ und verlängern bei Erfolg den Pfad um diese Kante. In allen anderen Fällen verbleiben wir im momentanen Zustand s. Die zugehörigen Übergangswahrscheinlichkeiten sind

$$P_{ss'}^{(n)} = \begin{cases} (2d)^{-1}, & \text{falls } s' \prec s,\ |s'| = |s| - 1; \\ (2d)^{-1}\beta_{|s|+1}, & \text{falls } s \prec s',\ |s'| = |s| + 1; \\ r(s), & \text{falls } s = s'; \\ 0, & \text{sonst;} \end{cases} \qquad s, s' \in \overline{\mathcal{S}}_n. \tag{8.14}$$

Dabei ist $r(s)$ so, dass $\sum_{s' \in \overline{\mathcal{S}}_n} P_{ss'}^{(n)} = 1$. Da je zwei Zustände über den Pfad $(0) \in \overline{\mathcal{S}}_0$ miteinander kommunizieren können, ist die Kette $(X_k^{(n)})_{k \ge 0}$ ergodisch.

Lemma 8.3 *Sei π_n die stationäre Verteilung der Markovkette $(X_k^{(n)})_{k\ge 0}$ mit Übergangsmatrix $P^{(n)}$. Dann gilt*

$$\pi_n(s) = \alpha_n \prod_{i=1}^{|s|} \beta_i, \quad s \in \overline{\mathcal{S}}_n, \tag{8.15}$$

wobei $\alpha_n = \Big(\sum_{j=0}^{n} N_j \prod_{i=1}^{j} \beta_i \Big)^{-1}$.

Beweis. Für $s \prec s' \in \overline{\mathcal{S}}_n$ mit $|s'| = |s| + 1$ gilt

$$\pi_n(s)\, P_{ss'}^{(n)} = \ \alpha_n \prod_{i=1}^{|s|} \beta_i \, \frac{\beta_{|s|+1}}{2d} \ = \ \alpha_n \prod_{i=1}^{|s'|} \beta_i \, \frac{1}{2d} \ = \ \pi_n(s')\, P_{s's}^{(n)}.$$

$P^{(n)}$ erfüllt also die lokale Gleichgewichtsbedingung. Insbesondere ist daher π_n die stationäre Verteilung der Markovkette $(X_k^{(n)})_{k\ge 0}$.

Das Gewicht $\pi_n(s)$ eines Pfades s hängt nur von dessen Länge $|s|$ ab. Für beliebige $\beta_i, 1 \le i \le n$ und alle $0 \le j \le n$ ist daher das Wahrscheinlichkeitsmaß

$$\nu_j(B) := \frac{\pi_n(\mathcal{S}_j \cap B)}{\pi_n(\mathcal{S}_j)}, \ B \subset \mathcal{S}_j \tag{8.16}$$

die Gleichverteilung auf $\mathcal{S}_j$. Diese Beobachtung legt das folgende, wiederum auf dem Schachtelungsprinzip basierende Verfahren zur Schätzung der N_n nahe (siehe [2, 23]).

MCMC-Verfahren zum approximativen Zählen der selbstmeidenden Pfade in $\mathbb{Z}^d$

Setze $\widehat{N}_0 := 1$ und $\widehat{N}_1 := 2d$. Für $n = 1, 2, 3, \ldots$

1) Erzeuge durch unabhängiges Simulieren der Markovkette $(X_k^{(n)})_{k\ge 0}$ für jeweils t_n Schritte eine Stichprobe vom Umfang m_n aus $\overline{\mathcal{S}}_n$.

2) Setze

$$\begin{aligned} Y_n &:= \{\,\text{Pfade der Länge } n \text{ in der Stichprobe}\,\}, \\ \mathrm{ext}(Y_n) &:= \sum_{s\in Y_n} |\{s'\in\mathcal{S}_{n+1}\colon s\prec s'\}|. \end{aligned}$$

3) Falls $|Y_n| < m'_n$ oder $\mathrm{ext}(Y_n)=0$, dann Abbruch. Sonst setze

$$\widehat{N}_{n+1} := \frac{\mathrm{ext}(Y_n)}{|Y_n|}\,\widehat{N}_n.$$

Zur Bestimmung der Gesamtzahl $\mathrm{ext}(Y_n)$ der möglichen Fortsetzungen prüft man für jeden Pfad aus Y_n, wie viele der $2d-1$ potenziellen Fortsetzungen einen selbstmeidenden Pfad der Länge $n+1$ ergeben. Die Wahl von m_n, m'_n und t_n spezifizieren wir später. Zunächst beschäftigen wir uns mit der Rolle der Parameter β_i, $1\le i\le n$. Mit Blick auf die Effizienz des Algorithmus müssen die β_i so gewählt werden, dass einerseits beim Simulieren der Markovkette $(X_k^{(n)})_{k\ge0}$ möglichst oft Pfade der Länge n erzeugt werden, andererseits eine relativ geringe Schrittzahl t_n zur Annäherung ans Gleichgewicht π_n genügt. Im Fall $\beta_i = N_{i-1}/N_i$, $1\le i\le n$ gilt

$$\pi_n(\mathcal{S}_j) \;=\; N_j\,\alpha_n \prod_{i=1}^{j}\frac{N_{i-1}}{N_i} \;=\; \alpha_n \;=\; \frac{1}{n+1},\quad 0\le j\le n. \tag{8.17}$$

Das Maß π_n legt also ausreichendes Gewicht auf $\mathcal{S}_n$.

Wir wenden uns jetzt der Frage zu, wie schnell die Kette $(X_k^{(n)})_{k\ge0}$ in ihr Gleichgewicht konvergiert. Betrachten wir zunächst eine beliebige ergodische reversible Markovkette mit Zustandsraum $\mathcal{S}$, Übergangsmatrix P und stationärer Verteilung π. Mit der Markovkette assoziieren wir einen gewichteten Graphen mit Knotenmenge $\mathcal{S}$ und Kantenmenge $E=\{(x,y)\colon P_{xy}>0\}$. Einer Kante (x,y) wird das Gewicht $q(x,y)=\pi(x)P_{xy}=q(y,x)$ zugeordnet. Ein *kanonischer Weg* γ_{xy} ist eine Folge von Kanten vom Knoten x zu y. Für eine Kollektion $\Gamma=\{\gamma_{xy}\colon x\ne y\in\mathcal{S}\}$ von kanonischen Wegen definieren wir

$$\rho(\Gamma) := \max_{e\in E}\;\frac{1}{q(e)}\sum_{\gamma_{xy}\ni e}\pi(x)\pi(y)|\gamma_{xy}|,$$

wobei $|\gamma_{xy}|$ die Länge (= Anzahl der Kanten) des Weges γ_{xy} bezeichnet. Die Größe $\rho(\Gamma)$ kann man als die maximale Belastung einer Kante e durch Wege in Γ relativ zu der *Kapazität* $q(e)$ der Kante ansehen.

Die Existenz einer Kollektion kanonischer Wege mit kleinem ρ impliziert, dass die zugehörige Markovkette rasch mischt. Dies zeigt die folgende Abschätzung (siehe [27]) für die Mischzeiten

$$\tau_x(\varepsilon) := \min\{k\ge0\colon d_{TV}[\,\mathbb{P}_x\{X_k\in\,\cdot\,\},\pi\,]\le\varepsilon\}.$$

Satz 8.4 *Für alle $\varepsilon>0$ und $x\in\mathcal{S}$ gilt*

$$\tau_x(\varepsilon) \;\le\; K\min_{\Gamma}\rho(\Gamma)\Big(\log\frac{1}{\pi(x)}+\log\frac{1}{\varepsilon}\Big),$$

Dabei ist K eine von ε und x unabhängige Konstante.

Wir werden Satz 8.4 zur Abschätzung der Mischzeit $\tau_{(0)}(\varepsilon)$ der Markovkette $(X_k^{(n)})_{k\geq 0}$ verwenden. Zwar beschränken wir uns dabei auf den idealisierten Fall $\beta_i = N_{i-1}/N_i$, $1 \leq i \leq n$, die Analyse ist aber stabil gegenüber leichten Änderungen der Parameter. Sei

$$a_n := \min_{j+k\leq n} \frac{N_{j+k}}{N_j N_k},\ n \in \mathbb{N}_0.$$

Man beachte, dass $N_{j+k}/(N_j N_k)$ die Wahrscheinlichkeit ist, dass man zwei unabhängige, rein zufällige selbstmeidende Pfade der Länge j bzw. k zu einem selbstmeidenden Pfad der Länge $j+k$ zusammenfügen kann. Ist die Vermutung (8.11) wahr, dann gilt

$$\frac{N_j N_k}{N_{j+k}} \sim A_d\Big(\frac{jk}{j+k}\Big)^{\gamma_d - 1} \leq A_d\Big(\frac{j+k}{4}\Big)^{\gamma_d - 1}$$

für $d = 2, 3$. Insbesondere ist dann a_n^{-1} polynomial beschränkt in n.

Satz 8.5 *Sei $\beta_i = N_{i-1}/N_i$ für $1 \leq i \leq n$, dann gilt für alle $n \in \mathbb{N}$ und $\varepsilon > 0$*

$$\tau_{(0)}(\varepsilon) \ \leq\ \frac{K' d\, n^2}{a_n}\Big(\log n + \log\frac{1}{\varepsilon}\Big). \tag{8.18}$$

Beweis. Der mit der Markovkette $(X_k^{(n)})_{k\geq 0}$ assoziierte Graph mit Knotenmenge $\overline{\mathcal{S}}_n$ ist der von der partiellen Ordnung $\prec$ induzierte Baum mit dem Pfad (0) als Wurzel. Die Kinder eines selbstmeidenden Pfades s sind alle Pfade s' mit $s \prec s'$ und $|s'| = |s| + 1$. Zu je zwei Pfaden s und s' gibt es einen eindeutigen (kürzesten) Weg $\gamma_{ss'}$ von s nach s'. Da der Baum die Höhe n hat, ist die Länge eines jeden solchen Weges durch $2n$ beschränkt. Nach (8.17) gilt $\pi_n((0)) = (n+1)^{-1}$. Nach Satz 8.4 genügt es daher zu zeigen, dass

$$\max_{e\in E} \frac{1}{q(e)} \sum_{\gamma_{s\bar{s}} \ni e} \pi_n(s)\pi_n(\bar{s}) \ \leq\ \frac{2dn}{a_n}. \tag{8.19}$$

Wir fixieren nun eine Kante $e = (s', s'')$ mit $s' \prec s''$, und bezeichnen mit $V_e = \{s : s'' \prec s\}$ die Knotenmenge des in s'' verwurzelten Teilbaums. Da die Kante e die Mengen V_e und V_e^c trennt, gilt

$$\frac{1}{q(e)} \sum_{\gamma_{s\bar{s}} \ni e} \pi_n(s)\pi_n(\bar{s}) \ =\ \frac{1}{q(e)} \sum_{s\in V_e,\, \bar{s}\in V_e^c} \pi_n(s)\pi_n(\bar{s}) \ \leq\ \frac{\pi_n(V_e)}{q(e)}. \tag{8.20}$$

Aus der Definition von q und den Beziehungen (8.14) und (8.17) folgt

$$\frac{1}{q(e)} \ =\ \frac{1}{\pi_n(s'')P_{s''s'}^{(n)}} \ =\ N_{|s''|}(n+1)\,2d. \tag{8.21}$$

Schließlich gilt für $k = |s''|$

$$\begin{aligned}\pi_n(V_e) \ &=\ \sum_{j=k}^{n} \pi_n(V_e \cap \mathcal{S}_j) \ \overset{(8.16)}{=}\ \sum_{j=k}^{n} \frac{|V_e \cap \mathcal{S}_j|}{N_j}\, \pi_n(\mathcal{S}_j) \\ &\leq\ \frac{1}{n+1} \sum_{j=k}^{n} \frac{N_{j-k}}{N_j} \ \leq\ \frac{n}{(n+1)N_k a_n}.\end{aligned} \tag{8.22}$$

Bei der ersten Ungleichung in (8.22) haben wir benutzt, dass jedes Teilstück eines selbstmeidenden Pfades wiederum selbstmeidend ist und folglich $|V_e \cap \mathcal{S}_j| \leq N_{j-k}$ für alle $k \leq j \leq n$. Die Behauptung (8.19) folgt durch Kombination von (8.20), (8.21) und (8.22).

Wir kehren nun zu der Analyse unseres Algorithmus zur approximativen Bestimmung von N_n zurück. Da wir die N_i ja nicht kennen, können wir auch die Parameter β_i nicht einfach als N_{i-1}/N_i wählen. Wir können aber die N_{i-1}/N_i nach und nach durch $\hat{\beta}_i := |Y_{i-1}|/\mathrm{ext}(Y_{i-1})$ schätzen und erhalten dadurch zum Ende der Stufe n des Algorithmus eine Approximation des optimalen Parameterwerts N_{i-1}/N_i. Man beachte, dass der Parameter β_n erstmalig auf Stufe $n+1$ benötigt wird. Als Schätzer für N_n erhalten wir

$$\widehat{N}_n = \Big(\prod_{i=1}^{n} \hat{\beta}_i\Big)^{-1}.$$

Haben wir jeden Quotient N_{i-1}/N_i bis auf einen relativen Fehler ε_i approximiert, d.h. ist

$$(1+\varepsilon_i)^{-1}\hat{\beta}_i \le N_{i-1}/N_i \le (1+\varepsilon_i)\hat{\beta}_i \text{ für } 1 \le i \le n,$$

dann gilt insbesondere

$$\prod_{i=1}^{n}(1+\varepsilon_i)^{-1}\,\widehat{N}_n \le N_n \le \prod_{i=1}^{n}(1+\varepsilon_i)\,\widehat{N}_n.$$

Mit $\varepsilon_i = \varepsilon/(4i^2)$, $1 \le i \le n$ folgt

$$(1+\varepsilon)^{-1}\widehat{N}_n \le N_n \le (1+\varepsilon)\widehat{N}_n.$$

Nach (8.16) ist π_j bedingt auf Werte in $\mathcal{S}_j$ die Gleichverteilung, unabhängig von etwaigen Abweichungen der Schätzer $\hat{\beta}_i$ vom optimalen Wert N_{i-1}/N_i. Die Ereignisse $\{(1+\varepsilon_i)^{-1}\hat{\beta}_i \le N_{i-1}/N_i \le (1+\varepsilon_i)\hat{\beta}_i\}$, $1 \le i \le n$ sind daher nahezu unabhängig. (Eine geringe Abhängigkeit entsteht durch den Einfluss der $\hat{\beta}_i$ auf die Mischzeiten der Ketten.) Ist die Irrtumswahrscheinlichkeit bei der Approximation der N_{i-1}/N_i durch $\delta_i = \delta/(2i^2), 1 \le i \le n$ beschränkt, dann folgt

$$\mathbb{P}\Big\{(1+\varepsilon)^{-1}\,\widehat{N}_n \le N_n \le (1+\varepsilon)\widehat{N}_n\Big\} \;\ge\; 1 - \sum_{i=1}^{n}\delta_i \;\ge\; 1-\delta.$$

Kommen wir nun zur Wahl von m_n, m_n' und t_n. Nach Satz 8.5 genügt es, die Markovkette $(X_k^{(n)})_{k\ge 0}$ für jeweils $t_n = O(n^2 a_n^{-1}(\log n + \log \varepsilon^{-1}))$ Schritte zu simulieren. Zur Wahl von m_n' beachte man, dass $\hat{\beta}_i^{-1}$ eine Summe (zufälliger Länge) von unabhängig identisch verteilten Zufallsvariablen ist. Die Summanden sind nicht-negativ, beschränkt und haben Erwartungswert ≥ 1. Eine einfache Verallgemeinerung der Bernstein-Chernoff-Schranke (Satz 4.1) zeigt, dass m_n' nicht allzu groß gewählt werden muss, um mit hoher Wahrscheinlichkeit eine gute Approximation des Erwartungswerts zu erhalten (ein Stichprobenumfang $m_n' \ge N_d\, n^4 \varepsilon^{-2}(\log n + \log \delta^{-1})$ erweist sich als ausreichend). Eine gute Wahl von m_n ist schließlich $2(n+1)m_n'$. Nach (8.17) ist dann $\mathbb{E}|Y_n| \approx 2m_n'$ und eine weitere Anwendung der Bernstein- Chernoff-Schranke zeigt, dass die die Wahrscheinlichkeit, dass das Verfahren abbricht, extrem gering ist. Zusammengefasst erhält man das folgende Resultat aus [23].

Satz 8.6 *Zu vorgegebenem $n \in \mathbb{N}$ und $\varepsilon, \delta > 0$ approximiert das MCMC-Verfahren die Zahl N_n der selbstmeidenden Pfade der Länge n mit Wahrscheinlichkeit $\ge 1-\delta$ bis auf einen relativen Fehler ε. Die Laufzeit des Algorithmus ist polynomial beschränkt in n, ε^{-1}, $\log \delta^{-1}$ und a_n^{-1}.*

Bibliography

[1] Aldous, D. und Diaconis, P. (1986) Shuffling cards and stopping times, *Amer. Math. Monthly* **93**, 333–348.

[2] Berretti, A. und Sokal, A. D. (1985) New Monte Carlo method for the self-avoiding walk, *J. Stat. Physics* **40**, 483–531.

[3] Bundschuh, P. (1992). *Einführung in die Zahlentheorie*, 2. Aufl., Springer, Berlin.

[4] Cormen, T. H., Leiserson, C. E. und Rivest, R. L. (1990) *Introduction to Algorithms*, The MIT Press, Cambridge.

[5] Devroye, L. (1984) Exponential bounds for the running time of a selection algorithm, *J. Comp. System Sciences* **29**, 1–7.

[6] Fill, J. A. (1998) An interruptible algorithm for perfect sampling via Markov chains, *Ann. Appl. Probab.* **8**, 131–162.

[7] Floyd, R. W. and Rivest, R. L.. (1975) Expected time bounds for selection. *Communications of the ACM* **18**, 165–172.

[8] Freivalds, R. (1977) Probabilistic machines can use less running time, *Information Processing 77, Proceedings of IFIP Congress 77*, 839–842.

[9] Geman, S. und Geman, D. (1984) Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **6**, 721–741.

[10] Gidas, B. (1995) Metropolis-type Monte Carlo simulation algorithms and simulated annealing, in L. Snell (ed.), *Topics in Contemporary Probability and its Applications*, CRC Press, Boca Raton, 159–232.

[11] Hastings, W. K. (1970) Monte Carlo sampling methods using Markov chains and their applications, *Biometrika* **57**, 97–109.

[12] Hoare, C.A.R. (1962) Quicksort. *Computer Journal* **5**, 10–15.

[13] Kaklamanis, C., Krizanc, D. und Tsantilas, T. (1991) Tight bounds for oblivious routing in the hypercube, *Proc. 3rd Annual ACM Symposium on Parallel Algorithms and Architectures*, 31–36.

[14] Karp, R. M. (1991) An introduction to randomized algorithms, *Disc. Appl. Math.* **34**, 165–201.

[15] Karp, R. M. und Rabin, M. (1987) Efficient randomized pattern-matching algorithms, *IBM J. Res. Develop.* **31**, 249–260.

[16] Luby, M., Sinclair, A. und Zuckerman, D. (1993) Optimal Speedup of Las-Vegas-Algorithms, *Inf. Proc. Lett.* **47**, 173–180.

[17] Madras, N. und Slade, G. (1993) *The self-avoiding walk*, Birkhäuser, Boston.

[18] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. und Teller, E. (1953) Equation of state calculations by fast computing machines, *Journal of Chemical Physics* **21**, 1087–1092.

[19] Mc Kinsey, J. C. C. (1952) *Introduction to the Theory of Games*, Mc Graw-Hill Book Company, New York.

[20] Motwani, R. und Raghavan, P. (1995) *Randomized Algorithms*, Cambridge University Press.

[21] Norris, J.R. (1997). *Markov chains*. Cambridge University Press, Cambridge.

[22] Propp, J. G. und Wilson, D. B. (1996) Exact sampling with coupled Markov chains and applications to statistical mechanics, *Random Structures and Algorithms* **10**, 223–252.

[23] Randall, D. und Sinclair, A. (1994) *Testable algorithms for self-avoiding walks*, Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, 593-602.

[24] Rösler, U. (1991) A limit theorem for "Quicksort", *Theor. Inf. Appl.* **25**, 85–100.

[25] Saks, M. und Wigderson, A. (1986) Probabilistic Boolean decision trees and the complexity of evaluating game trees, *Proceedings of the 27th Annual IEEE Symposium on Foundations of Computer Science*, Toronto, 29–38.

[26] Sedgewick, R. (1988) *Algorithms*, Addison-Wesley, Reading.

[27] Sinclair, A. (1992) Improved bounds for mixing rates of Markov chains and multicommodity flow, *Comb. Prob. Comp.* **1**, 351–370.

[28] Sinclair, A. (1993) *Algorithms for Random Generating and Counting*, Birkhäuser.

[29] Snir, M. (1985) Lower bounds for probabilistic linear decision trees, *Theor. Comp. Science* **38**, 69–82.

[30] Swendsen, R. H. and Wang, J.-S. (1987) Nonuniversal critical dynamics in Monte Carlo simulations, *Phys. Rev. Letters* **58**, 86–88.

[31] Valiant, L. G. und Brebner, G. J. (1981) Universal schemes for parallel computation, *Proc. 13th Annual ACM Symposium on Theory of Comput.*, Milwaukee, 263–277.