Humboldt-Universität zu Berlin
Institut für Informatik
Lehrstuhl für Algorithmen und Komplexität

Randomisierte Algorithmen

Studienarbeit

Daniel Rolf [1]

5. Dezember 2002

Betreuer: Dr. Deryk Osthus

[1] rolf@informatik.hu-berlin.de

Zusammenfassung

Kapitel 1. Ein genügend großer, zufälliger Graph hat bei richtiger Parametrisierung die Eigenschaft, einerseits wenige kurze Kreise zu enthalten und andererseits jedoch genug Kanten, um einen beliebigen Durchschnittsgrad nach Zerstören der kurzen Kreise zu produzieren. Durch die Methode der bedingten Erwartungswerte lässt sich der Algorithmus derandomisieren und zur expliziten Konstruktion eines gewünschten Graphen benutzen.

Kapitel 2. Wie schon Paul Erdös im Jahre 1957 bewiesen hat, lassen sich auch Graphen würfeln, die wiederum kaum kurze Kreise enthalten jedoch eine hohe chromatische Zahl aufweisen. Durch einen etwas anderen Ansatz, als Erdös in seinem Beweis benutzte, kann man hier den Beweis auf lineare Erwartungswerte zurückführen, der sich dann mittels der Methode der bedingten Erwartungswerte derandomisieren lässt.

Kapitel 3. Polynome über einem beliebigen Körper haben die Eigenschaft, maximal so viele Nullstellen zu besitzen, wie ihr Grad ist, oder konstant Null zu sein. Daraus lässt sich eine effiziente Methode zum randomisierten Erzeugen von Fingerprints ableiten.

Kapitel 4. Das Paging-Problem, bei dem kostenintensive Zugriffe auf einen langsamen Lesespeicher durch kostenlose Zugriffe auf einen kleinen Cache der Größe k minimiert werden, ist durch einen einfachen randomisierten Algorithmus realisierbar, welcher im Erwartungswert etwa $2 \ln k$ mal so schlecht ist wie ein deterministischer Algorithmus, welcher die komplette Anfragefolge im Voraus sehen kann.

Inhaltsverzeichnis

1 Existenz von Graphen mit hoher Taillenweite und hohem Durchschnittsgrad

1.1 Randomisierter Beweis

Sei V eine beliebige Menge mit n Elementen, außerdem sei $\omega : \binom{V}{2} \mapsto \{0,1\}$ für alle $e \in \binom{V}{2}$ durch ein Zufallsexperiment mit $\mathbb{P}(\omega(e) = 1) = p$ bzw. $\mathbb{P}(\omega(e) = 0) = 1 - p$ bestimmt worden. Dann ist $G = (V, E)$ mit $E = \{e|\omega(e) = 1\}$ ein *zufälliger Graph* auf V mit der Kantenwahrscheinlichkeit p. Der Wahrscheinlichkeitsraum aller zufälligen Graph auf einer n-elementigen Knotenmenge mit der Kantenwahrscheinlichkeit p wird mit $\mathcal{G}(n,p)$ bezeichnet.

Die widersprüchliche Frage, ob es einen Graphen gibt, dessen Kreise mindestens die Länge g haben und dessen Durchschnittsgrad gleichzeitig mindestens d ist, beantwortet folgende Proposition.

Proposition 1.
Für ein gegebenes g und d existiert ein Graph H, dessen Taillenweite mindestens g und Durchschnittsgrad mindestens d ist.

Beweis. Der Beweis verläuft ähnlich zum Beweis des Theorems von Erdös in [1]. Für $g \leq 2$ oder $d = 0$ ist das offensichtlich. Sei also $g \geq 3$ und $d \geq 1$ gegeben. Sei $\epsilon := 1/(2g)$ und $p := n^{\epsilon - 1}$. Mit $X(G)$ wird die Anzahl der Kreise kürzer als g und mit $Z(G)$ die Anzahl der Kanten in einem zufälligen Graphen $G \in \mathcal{G}(n,p)$ bezeichnet.

Für den Erwartungswert $\mathbb{E}[X]$ von $X(G)$ wird für alle $i \geq 3$ die Anzahl der erwarteten Kreise der Länge i gezählt und durch die Anzahl der $2i$ symmetrischen Varianten eines Kreises dividiert (Rotation und Drehsinn). $\mathbb{E}[X]$ ergibt sich zu

$$\mathbb{E}[X] = \sum_{i=3}^{g} \frac{n!}{2i(n-i)!} p^i \leq \frac{1}{2} \sum_{i=3}^{g} n^i p^i \overset{(*)}{\leq} \frac{1}{2}(g-2) n^g p^g = \frac{1}{2}(g-2) n^{1/2}, \quad (1)$$

wobei sich die Ungleichung $(*)$ mit $(np)^i \leq (np)^g$ ergibt.

Für $\mathbb{E}[Z]$ gilt

$$\mathbb{E}[Z] = p\binom{n}{2} = \frac{n(n-1)p}{2} = \frac{(n-1)n^{1/(2g)}}{2}. \quad (2)$$

Mit $M(G)$ wird die Anzahl der Kanten bezeichnet, die übrig bleiben, wenn aus jedem „kurzen“ Kreis in G eine Kante entfernt wird. Dann folgt

$$\begin{aligned} &M(G) \geq Z(G) - X(G) \text{ und} \\ &\mathbb{E}[M] \geq \mathbb{E}[Z] - \mathbb{E}[X] \\ &\qquad = \frac{(n-1)n^{1/(2g)} - (g-2)n^{1/2}}{2}. \end{aligned}$$

Sei $n := (3d)^{2g}$. Sei $D(G)$ der Durchschnittsgrad für einen zufälligen Graph $G \in \mathcal{G}(n,p)$. Für seinen Erwartungswert $\mathbb{E}[D]$ gilt dann

$$\begin{aligned}
\mathbb{E}[D] &= 2\mathbb{E}[M]/n \\
&\geq \frac{1}{n}\left((n-1)n^{1/(2g)} - (g-2)n^{1/2}\right) \\
&= n^{1/(2g)}(1-1/n) - (g-2)n^{-1/2} \\
&\geq \frac{1}{2}n^{1/(2g)} - (g-2)n^{-1/2} \\
&\geq \frac{3}{2}d - (g-2)(3d)^{-g} \\
&\geq \frac{3}{2}d - 1/3 \\
&\geq d.
\end{aligned}$$

Da $\mathbb{E}[D] \geq d$ ist, muss es einen Graphen $G \in \mathcal{G}(n,p)$ geben, für den $D(G) \geq d$ gilt.

□

1.2 Derandomisierung

Die Bezeichnungen des vorherigen Abschnittes sollen in diesem Abschnitt weiterhin gelten. Es werden nun jedoch zufällige Graphen betrachtet, deren Kanten nicht mehr mit einer uniformen Wahrscheinlichkeit auftreten.

Der Wahrscheinlichkeitsraum aller *zufälligen Graphen* auf einer n-elementigen Knotenmenge V mit den Kantenwahrscheinlichkeiten $p : \binom{V}{2} \mapsto [0..1]$ wird mit $\mathcal{G}(n,p)$ bezeichnet, d.h. für $G = G(V,E) \in \mathcal{G}(n,p)$ ist die Kante $e \in \binom{V}{2}$ mit der Wahrscheinlichkeit $p(e)$ in E enthalten.

Dann gilt

$$\mathbb{E}[Z] = z(p) := \sum_{e \in \binom{V}{2}} p(e), \tag{3}$$

$$\mathbb{E}[X] = x(p) := \mathbb{E}[X] = \sum_{l=3}^{g} \frac{1}{2l} \sum_{(v_1,\ldots,v_l) \in \binom{V}{l}} p(\{v_1,v_2\}) \cdot \ldots \cdot p(\{v_l,v_1\}) \text{ und} \tag{4}$$

$$m(p) := z(p) - x(p).$$

Die Funktionen z, x und m bekommen als Parameter die Kantenwahrscheinlichkeiten p. Gleichung (3) folgt unmittelbar daraus, das die erwartete Anzahl der Kanten die Summe der einzelnen Kantenwahrscheinlichkeiten ist. Die innere Summe der Gleichung (4) berechnet die erwartete Anzahl aller Kreise der Länge l, was in der äußeren Summe für alle Längen von 3 bis g aufsummiert wird. Die

innere Summe muss durch $2l$ dividiert werden, da jeder Kreis $2l$ mal gezählt wird (Rotation und Drehsinn).

Folgender deterministischer Algorithmus findet nun einen gewünschten Graphen:

Algorithmus 1.1: FindeGraphenDG(d, g)

$n \leftarrow (3d)^{2g}$
for $e \in \binom{V}{2}$ **do** $p(e) \leftarrow n^{1/(2g)-1}$
for $e \in \binom{V}{2}$
do $\begin{cases} m_0 \leftarrow m(p|p(e) \leftarrow 0) \\ m_1 \leftarrow m(p|p(e) \leftarrow 1) \\ \textbf{if } m_1 < m_0 \\ \quad \textbf{then } p(e) \leftarrow 0 \\ \quad \textbf{else } p(e) \leftarrow 1 \end{cases}$
$G \leftarrow (V, \{e \in \binom{V}{2} | p(e) = 1\})$
Entferne je eine Seite jedes kurzen Kreises in G
return (G)

Der Ausdruck $p|p(e) \leftarrow i$ ist gleichwertig zu p' mit $p'(e') := p(e')$ für alle $e' \in \binom{V}{2}$ mit $e \neq e'$ und $p'(e) := i$.

Proposition 2.
Für gegebenes $d > 0$ und $g > 2$ findet Algorithmus 1.1 eine Graphen G auf $n := (3d)^{2g}$ Knoten, dessen Durchschnittsgrad mindestens d und dessen Taillenweite mindestens g ist. Die Laufzeit ist

$$\begin{aligned} &\mathcal{O}(n^{g+2}) \\ &= \mathcal{O}\left((3d)^{2g(g+2)} g\right). \end{aligned}$$

Beweis. Offensichtlich sind am Ende aller Durchläufe alle Kantenwahrscheinlichkeiten 0 oder 1. Somit ist $M(G)$ für den zurückgegebene Graphen keine Zufallsvariable mehr. Die Funktion $m(p)$ ist linear in $p(e)$ für alle $e \in \binom{V}{2}$. Deshalb gilt für beliebiges $e \in \binom{V}{2}$

$$\begin{aligned} m(p) &= m(p|p(e) \leftarrow 0) + (m(p|p(e) \leftarrow 1) - m(p|p(e) \leftarrow 0))p(e), \\ &= m(p|p(e) \leftarrow 0)(1 - p(e)) + m(p(e) \leftarrow 1)p(e) \text{ und} \\ &\leq \max\{m(p(e) \leftarrow 0), m(p(e) \leftarrow 1)\}. \end{aligned} \tag{5}$$

Für das initiale p_0 und das finale p_1 gilt somit $m(p_0) \leq m(p_1)$ und damit $M(G) \geq m(p_1) \geq m(p_0)$.

Unter der Annahme, dass Addition und Multiplikation in $\mathcal{O}(1)$ zu realisieren sind, ergibt sich die Laufzeit folgendermaßen.

Die äußere Schleife wird $n(n-1)/2 = \mathcal{O}(n^2)$ mal durchlaufen. Die Laufzeit zur Berechnung von m ist in jedem Durchlauf invariant. z wird in $\mathcal{O}(n^2)$ berechnet. Die Laufzeit zur Berechnung von x ergibt sich zu

$$\begin{aligned}
&\sum_{l=3}^{g} \sum_{(v_1,\ldots,v_l)\in\binom{V}{l}} \mathcal{O}(l) \\
&= \sum_{l=3}^{g} \binom{n}{l} l!\mathcal{O}(l) \\
&= \sum_{l=3}^{g} \frac{n!}{(n-l)!}\mathcal{O}(l) \\
&= \mathcal{O}(n^g g).
\end{aligned}$$

Somit ist die zur Berechnung von m benötigte Laufzeit $\mathcal{O}(n^g g)$.

Das Entfernen der kurzen Kreise benötigt, durch Aufzählen aller kurzen Kreise wie in Gleichung (4) geschehen, auch die Zeit $\mathcal{O}(n^g g)$.

Das ergibt eine Gesamtlaufzeit von

$$\mathcal{O}(n^{g+2}g) = \mathcal{O}\left((3d)^{(2g)(g+2)}g\right).$$

□

2 Existenz von Graphen mit hoher Taillenweite und hoher chromatischer Zahl

2.1 Randomisierter Beweis

Wie in Abschnitt 1.1 werden zufällige Graphen betrachtet. Folgendes Theorem, welches eine Abwandlung des Theorems von Erdös aus dem Jahre 1959 ([1, Kapitel 11.1]) ist, wird das zentrale Element dieses Kapitels sein.

Theorem 1.
Für ein gegebenes g und k existiert ein Graph H, dessen Taillenweite mindestens g und chromatische Zahl mindestens k ist.

Zum Beweis von Theorem 1 wird folgende Proposition gebraucht.

Proposition 3. *Sei $G = G(V, E)$ ein beliebiger Graph und sei $H = G(V, E')$ ein Graph, der aus G durch Entfernen von k beliebigen Kanten hervorgegangen ist. Dann gilt $\alpha(H) \leq \alpha(G) + k$.*

Beweis. Es genügt zu zeigen, dass durch das Entfernen genau einer Kante die Stabilitätszahl um maximal 1 fallen kann. Sei deshalb $k = 1$.

Seien dazu die zu G und H komplementären Graphen $\overline{G}$ und $\overline{H}$ betrachtet. $\overline{H}$ geht aus $\overline{G}$ durch Hinzufügen genau einer Kante $e = \{u, v\}$ hervor. Zu zeigen ist, dass $\omega(H) \leq \omega(G) + 1$. Sei $\omega(H) > \omega(G) + 1$ angenommen. Da sich die Cliquenzahl geändert hat, muss e offenbar Bestandteil der neuen Clique C der Größe $\omega(H)$ sein und es kann keine weiter Clique der Größe mindestens $\omega(G)+1$ geben, da diese sonst auch schon in G existiert hätte. C in H ist aus zwei disjunkten Cliquen A und B in G entstanden. O.B.d.A. sei $|A| \geq |B|$. Da A maximal die Größe $\omega(G)$ haben kann, muss B mindestens 2 Knoten enthalten. Es gilt entweder $u \in A \wedge v \in B$ oder $u \in B \wedge v \in A$. O.B.d.A sei $u \in B$. Dann gibt es einen weiteren Knoten $w \in B$. Da die neue Kante e zwischen u und v besteht, muss für $w \in C$ auch w selbst schon in G mit allen Knoten aus A verbunden sein, d.h. w ist in A. Widerspruch zu $A \cap B = \emptyset$. □

Damit kann nun Theorem 1 bewiesen werden.

Beweis. Theorem 1. Sei $G = G(V, E) \in \mathcal{G}(n, p)$ ein zufälliger Graph mit den Kantenwahrscheinlichkeiten $p : \binom{V}{2} \mapsto [0..1]$. Sei $L_a(G)$ die Anzahl der unabhängigen Mengen der Größe a in G. Für $\mathbb{E}[L_a]$ gilt

$$\mathbb{E}[L_a] = \sum_{C \in \binom{V}{a}} \prod_{e \in \binom{C}{2}} (1 - p(e)). \tag{6}$$

Gleichung (6) summiert für alle möglichen Knotenmengen $C \subseteq V$ der Größe a die Wahrscheinlichkeiten auf, dass diese eine stabile Menge bilden, was der Fall

ist, wenn es keine Kante in C gibt. Für $p(e) = p$ gilt

$$\begin{aligned}\mathbb{E}[L_a] &= \sum_{\binom{n}{a}\binom{a}{2}} \prod (1-p) \\ &= \binom{n}{a}(1-p)^{\binom{a}{2}}.\end{aligned}$$

Sei $\epsilon := 1/(2g)$ und $p := n^{\epsilon-1}$. Mit $X(G)$ wird die Anzahl Kreis kürzer als g und mit $Y(G)$ die Größe der größten stabilen Menge in $G \in \mathcal{G}(n,p)$ bezeichnet. Aus dem Beweis von Lemma 11.2.1 in ([1, Kapitel 11.2]) folgt für

$$p \geq (6k \ln n) n^{-1} \tag{7}$$

und $a \geq \frac{n}{2k}$

$$\begin{aligned}\mathbb{E}[L_a] &= \binom{n}{a}(1-p)^{\binom{a}{2}} \\ &\leq \sqrt{e/n}^{a}.\end{aligned}$$

Da $L_a(G)$ für einen fixen Graphen G offenbar streng monoton fallend in a ist und $L_a(G) = 0$ die Ungleichung $Y(G) < a$ impliziert, gilt $Y(G) \leq a + L_a(G)$ und somit auch

$$\mathbb{E}[Y] \leq a + \mathbb{E}[L_a]$$

Damit G keine kurzen Kreise mehr enthält, müssen aus G alle kurzen Kreise entfernt werden. Sei $Z(G)$ die Stabilitätszahl nach Entfernen maximal einer Kante jedes der $X(G)$ kurzen Kreise. Nach Proposition 3 gilt dann

$$\begin{aligned}Z(G) &\leq Y(G) + X(G) \\ &\leq a + L_a(G) + X(G) \\ \mathbb{E}[Z] &\leq a + \mathbb{E}[L_a] + \mathbb{E}[X].\end{aligned} \tag{8}$$

Sei $a := n/(2k)$ und $p := n^{1/(2g)-1}$. Mit

$$n \geq \left(k(g-2)+1\right)^2 \tag{9}$$

folgt

$$\begin{aligned}n^{1/2} &\geq 2n^{-1/2}k\sqrt{e/n}^{n/(2k)} + k(g-2) \\ p^{-g}n^{1-g} &\geq 2n^{-g}p^{-g}k\sqrt{e/n}^{n/(2k)} + k(g-2) \\ n/(2k) &\geq \sqrt{e/n}^{n/(2k)} + \frac{1}{2}(g-2)n^g p^g \\ \mathbb{E}[Z] &\leq n/k.\end{aligned} \tag{10}$$

Der Übergang von Ungleichung (9) zu Ungleichung (10) ist für $k > 2$ und $g > 2$ offensichtlich. Falls dieses n nicht groß genug für Ungleichung (7) ist, so muss es entsprechend erhöht werden.

Somit gibt es einen Graphen, der nach Entfernen der kurzen Kreise eine chromatische Zahl mindestens k aufweist und auch keine Kreise kürzer als g mehr enthält. □

2.2 Derandomisierung

Die Bezeichnungen des vorherigen Abschnittes sollen in diesem Abschnitt weiterhin gelten. Außerdem werden zufällige Graphen wie in Abschnitt 1.2 mit unterschiedlichen Kantenwahrscheinlichkeiten verwendet.

Für die Kantenwahrscheinlichkeiten p und einen zufälligen Graphen $G = G(V, E) \in \mathcal{G}(n, p)$ gilt

$$\mathbb{E}[L_a] = l(p) := \sum_{C \in \binom{V}{a}} \prod_{e \in C} (1 - p(e)), \tag{11}$$

$$\mathbb{E}[X] = x(p) := \sum_{l=3}^{g} \frac{1}{2l} \sum_{(v_1,\ldots,v_l) \in \binom{V}{l}} p(\{v_1, v_2\}) \cdot \ldots \cdot p(\{v_l, v_1\}), \tag{12}$$

$$\mathbb{E}[Z] \leq z(p) := a + l(p) + x(p) \tag{13}$$

und außerdem der aus dem Beweis von Theorem 1 essentielle Fakt

$$z(n^{1/(2g)-1}) \leq n/k.$$

Die Funktionen l, x und z bekommen als Parameter die Kantenwahrscheinlichkeiten p. Gleichung (12) folgt wie Gleichung (4), Gleichung (11) wurde bereits im Beweis von Theorem 1 vorgestellt und Ungleichung (13) folgt aus Ungleichung (8).

Folgender deterministischer Algorithmus findet nun einen gewünschten Graphen.

Algorithmus 2.1: FindeGraphenCG(k, g)

$n \leftarrow (24kg)^{4g}$
for $e \in \binom{V}{2}$ **do** $p(e) \leftarrow n^{1/(2g)-1}$
for $(i, j) \in A$
do
$z_0 \leftarrow z(p|p(e) \leftarrow 0)$
$z_1 \leftarrow z(p|p(e) \leftarrow 1)$
if $z_1 < z_0$
then $p(e) \leftarrow 0$
else $p(e) \leftarrow 1$
$G \leftarrow (V, \{e \in \binom{V}{2} | p(e) = 1\})$
Entferne je eine Seite jedes kurzen Kreises in G
return (G)

Proposition 4.
Für gegebenes $k > 2$ und $g > 2$ findet Algorithmus 2.1 eine Graphen G auf

$n := (24kg)^{4g}$ Knoten, dessen chromatische Zahl mindestens k und dessen Taillenweite mindestens g sind. Die Laufzeit ist

$$\begin{aligned}&\mathcal{O}(n^{g+2}g + n^{a+2}a^2)\\ &= \mathcal{O}\left((24kg)^{4g(g+2)}g + (24kg)^{4g((24kg)^{4g}/k+4)}/k^2\right)\end{aligned}$$

mit $a := n/(2k)$.

Beweis. Offensichtlich sind am Ende aller Durchläufe alle Kantenwahrscheinlichkeiten 0 oder 1. Somit ist $Z(G)$ für den zurückgegebene Graphen keine Zufallsvariable mehr. Die Funktion $z(p)$ ist linear in $p(e)$ für alle $e \in \binom{V}{2}$. Deshalb gilt (analog zu Ungleichung 5) für beliebiges $e \in \binom{V}{2}$

$$z(p) \geq \min\{z(p|p(e) \leftarrow 0), z(p|p(e) \leftarrow 1)\}.$$

Für das initiale p_0 und das finale p_1 gilt somit $z(p_0) \geq z(p_1)$ und damit $Z(G) \leq z(p_1) \leq z(p_0)$.

Unter der Annahme, dass Addition und Multiplikation in $\mathcal{O}(1)$ zu realisieren sind, ergibt sich die Laufzeit folgendermaßen.

Die äußere Schleife wird $n(n-1)/2 = \mathcal{O}(n^2)$ mal durchlaufen. Die Laufzeit zur Berechnung von z ist in jedem Durchlauf invariant. Die Laufzeit zur Berechnung von x ergibt wie in dem Beweis von Algorithmus 1.1 zu $\mathcal{O}(n^g g)$.

Die Berechnung von l benötigt die Laufzeit

$$\begin{aligned}&\sum_{C \in \binom{V}{a}} \mathcal{O}(a^2)\\ &= \binom{n}{a}\mathcal{O}(a^2)\\ &= \mathcal{O}(n^a a^2).\end{aligned}$$

Somit ist die zur Berechnung von z benötigte Laufzeit $\mathcal{O}(n^g g + n^a a^2)$. Man beobachte, dass statt z auch direkt y in Algorithmus 2.1 hätte benutzt werden können, jedoch die Laufzeit zur Berechnung von y vermutlich exponentiell in n ist.

Das Entfernen der kurzen Kreise benötigt, durch Aufzählen aller kurzen Kreise wie in Gleichung (12) geschehen, auch die Zeit $\mathcal{O}(n^g g)$.

Mit

$$n := (24kg)^{4g}$$

wird Ungleichung 7 erfüllt, denn

$$\begin{aligned}\frac{n}{(\ln n)^{2g}} &= \frac{(24kg)^{4g}}{(4g\ln(24kg))^{2g}}\\ &\geq \frac{(24kg)^{4g}}{(4g)^{2g}(24kg)^{2g}}\\ &\geq \frac{(4g)^{2g}(6k)^{2g}}{(4g)^{2g}}\end{aligned}$$

und somit

$$\begin{aligned} \frac{n}{(\ln n)^{2g}} &\geq (6k)^{2g} \\ n &\geq (6k \ln n)^{2g} \\ p = n^{1/2g-1} &\geq (6k \ln n) n^{-1} \end{aligned}$$

gelten.

Das ergibt eine Gesamtlaufzeit von

$$\begin{aligned} \mathcal{O}(n^{g+2}g + n^{a+2}a^2) &= \mathcal{O}(n^{g+2}g + n^{n/k+4}/k^2) \\ &= \mathcal{O}\left((24kg)^{4g(g+2)}g + (24kg)^{4g((24kg)^{4g}/k+4)}/k^2\right). \end{aligned}$$

□

3 String Verifikation

3.1 Galoiskörper

Ein Galoiskörper ist ein endlicher Körper mit n Elementen und wird mit $\mathbb{F}_n$ bezeichnet. Interessanterweise liefert die Zerlegung der ganzen Zahlen in Restklassen modulo p einen Galoiskörper, wenn p prim ist, wobei Multiplikation und Addition wie üblich modulo p ausgeführt werden. Folgendes Theorem aus [2, Kapitel 14], dass hier ohne Beweis angegeben wird, verstärkt die Charakterisierung von Galoiskörpern noch beträchtlich.

Theorem 2. *Sei p eine Primzahl und sei $r \in \mathbb{N}\backslash\{0\}$. Dann gibt es bis auf Isomorphie genau einen Galoiskörper $\mathbb{F}_{p^r}$ mit $q := p^r$ Elementen. Die Elemente von $\mathbb{F}_{p^r}$ sind die Nullstellen des Polynoms $X^q - X$ über dem Restklassenkörper $\mathbb{Z}_p[x]$.*

Dieses Theorem liefert also den Schlüssel zum Bilden von „beliebigen“ Galoiskörpern. Darauf soll hier mit Verweis auf die umfangreiche Literatur zur Kodierungstheorie nicht näher eingegangen werden.

Im folgenden werden die Objekte, die zur Repräsentation der Elemente von $\mathbb{F}_{p^r}$ benutzt werden, auf das Niveau von „Zahlen“ abstrahiert, d.h. es interessiert nicht, wie die Zahlen dargestellt werden oder die Operationen definiert sind, sondern lediglich die Möglichkeit in einem Körper mit p^r Elementen zu rechnen. Insbesondere lassen sich über $\mathbb{F}_{p^r}$ wieder Polynome bilden, die im folgenden Abschnitt untersucht werden.

3.2 Polynome über Galoiskörpern

Wenn man ein Polynom betrachtet, so interessiert oftmals, wieviele Nullstellen dieses Polynom haben kann. Der Fundamentalsatz der Algebra bringt Licht ins Dunkle.

Theorem 3. *Ein Polynom vom Grad n hat in jedem Körper maximal n Nullstellen oder ist identisch Null.*

Ein Beweis des Theorems 3 kann man in [2, Kapitel 8] finden. Insbesondere kann aus diesem Theorem die folgende für den folgenden Abschnitt wichtige Folgerung aufstellen.

Korollar 1. *Seien $f(x)$ und $g(x)$ verschiedene Polynome vom Grad höchstens n über dem Körper $\mathbb{F}_q$. Dann ist bei zufälliger Wahl von $x_0 \in \mathbb{F}_q$ die Wahrscheinlichkeit für $f(x_0) = g(x_0)$ kleiner gleich n/q.*

Beweis. Sei $d(x) := f(x) - g(x)$. $d(x)$ ist vom Grad höchstens n. $d(x)$ hat somit maximal n Nullstellen in $\mathbb{F}_q$, da $d(x)$ nicht identisch Null ist. Bei beliebiger Wahl von $x_0 \in \mathbb{F}_q$ ist somit die Wahrscheinlichkeit, eine Nullstelle von $d(x)$ zu treffen, kleiner gleich n/q. □

3.3 Anwendung zur String Verifikation

Sei ein String der Länge n über $\mathbb{F}_p$ ein Vektor $s = (s_1, ..., s_i)$ mit $s_i \in \mathbb{F}_p$, wobei p prim ist. Informal soll ein String Verifikationsalgorithmus überprüfen, ob zwei gegebene Strings identisch sind. Dieser Abschnitt zeigt ähnlich wie in [4, Kapitel 7], wie man eine einfache Methodik und deren Eigenschaften zur Verifikation zweier Strings aus Korollar 1 gewinnen kann.

Zu einem gegebene String wird das k-Polynom

$$p_s(x) := \sum_{i=0}^{k-1} a_i x^i$$

definiert, wobei k ein beliebiger Teiler von n ist und $a_i \in \mathbb{F}_{p^r}$ sich mit $r = n/k$ zu

$$a_i := \sum_{j=0}^{r-1} s_{ri+j} p^j$$

ergibt. Die Koeffizienten von p_s ergeben sich also aus dem Zusammenfassen von jeweils r aufeinanderfolgenden Koeffizienten von s. r kann also als *Blocklänge* und k als *Blockanzahl* aufgefasst werden.

Zwei Strings s, t der Länge n sind nun genau dann identisch, wenn alle Koeffizienten von p_s und p_t übereinstimmen. Direkt aus Korollar 1 ergibt sich die folgende Proposition.

Proposition 5. *Seien s und t zwei Strings der Länge n über $\mathbb{F}_p$, p prim. Sei k beliebiger Teiler von n und $r = n/k$. Seien p_s und p_t die k-Polynome von s und t. Sei $x_0 \in \mathbb{F}_{p^r}$ zufällig gewählt. Falls sich s und t unterscheiden, so ist die Wahrscheinlichkeit für $p_s(x_0) = p_t(x_0)$ kleiner gleich $(k-1)/p^r = (k-1)/p^{n/k}$.*

Beweis. p_s und p_t sind Polynome vom Grad höchstens $k-1$ und Koeffizienten im Galoiskörper $\mathbb{F}_{p^r}$. Weil s und t verschieden sind, gilt dies auch für p_s und p_t. Mit Korollar 1 folgt die Aussage. □

Im Kontext von Strings über $\mathbb{F}_p$ bezeichnet im folgenden der Begriff Bit eine Komponente dieses Strings. Dies ist damit zu rechtfertigen, dass meist Binärstrings über $\mathbb{F}_2$ betrachtet werden.

In einem konkreten Anwendungsfall kann man sich vorstellen, dass zwei Rechner über ein schwaches Netzwerk verbunden sind und vergleichen wollen, ob sie zwei identische Strings vorliegen haben. Sei z gerade und außerdem $r := z/2$ Teiler von n ist. Ein Paar $(x_0, p_s(x_0))$ wird *Zufallsfingerprint* der Länge z von s genannt, wenn x_0 eine aus $\mathbb{F}_{p^r}$ gleichverteilt zufällig bestimmte Zahl ist. Statt sich gegenseitig die Strings zu schicken, schickt ein Rechner dem anderen einen Zufallsfingerprint der Länge z seines Strings. Der zweite Rechner kann nun sein $p_s(x_0)$ berechnen und mit dem im Zufallsfingerprint angegebenen Wert vergleichen. Ein Zufallsfingerprint *scheitert*, wenn ein ungünstiges x_0 gewählt wurde, so dass $p_s(x_0)$ den gleichen Wert für beide Strings annimmt, obwohl es sich um unterschiedliche Strings handelt.

Folgende Proposition berechnet nun die Wahrscheinlichkeit für das Scheitern in Abhängigkeit der Länge des Zufallsfingerprints.

Proposition 6. *Seien s und t zwei Strings der Länge n über $\mathbb{F}_p$, p prim. Sei z gerade und $z/2$ ein Teiler von n. Ein Zufallsfingerprint der Länge z scheitert mit einer Wahrscheinlichkeit von höchstens*

$$\frac{2n/z - 1}{p^{z/2}} \leq \frac{2n}{zp^{z/2}}.$$

Beweis. Sei $r = z/2$ und $k = n/r$. Die Blocklänge ist r und die Blockanzahl ist k. Mit Proposition 5 ergibt sich die Wahrscheinlichkeit $(k-1)/p^r = (2n/z - 1)/p^{z/2}$. □

Folgende Rechnung zeigt, wie die Wahrscheinlichkeit für das Scheitern auf Vergrößern des Zufallsfingerprints um 2 Bit reagiert.

$$\begin{aligned}\left(\frac{2n/z-1}{p^{z/2}}\right) / \left(\frac{2n/(z+2)-1}{p^{(z+2)/2}}\right) &= \left(\frac{2n/z-1}{p^{z/2}}\right)\left(\frac{p^{(z+2)/2}}{2n/(z+2)-1}\right) \\ &= p \cdot \frac{2n/z-1}{2n/(z+2)-1} \\ &\geq p\end{aligned}$$

D.h. bei Vergrößerung um 2 Bit nimmt die Wahrscheinlichkeit des Scheiterns um Faktor mindestens p ab. Die Länge des Zufallsfingerprints geht also exponentiell-reziprok in die Wahrscheinlichkeit ein.

Nun kann man auch fragen, wie viele Bits Zufallsfingerprint man benötigt, um unter einer gewissen Scheiterwahrscheinlichkeit zu landen. Dazu gibt folgende Proposition Aufschluss.

Proposition 7. *Seien s und t zwei Strings der Länge n über $\mathbb{F}_p$, p prim. Sei $0 < w < 1$ die gewünschten maximale Scheiterwahrscheinlichkeit. Dann braucht man einen höchstens $2(\log_p n - \log_p w)$ Bits langen Zufallsfingerprint.*

Beweis. Es gilt $r = n/k$ zu finden mit $(k-1)/p^{n/k} \leq w$. Mit

$$r \geq \log_p n - \log_p w$$

folgt

$$\begin{aligned} r &\geq \log_p(n/w) \\ n/w &\leq p^r \\ n/w &\leq rp^r \\ \frac{n}{rp^r} &\leq w \\ \frac{k}{p^{n/k}} &\leq w \\ \frac{k-1}{p^{n/k}} &\leq w. \end{aligned}$$

□

Für eine fixe Wahrscheinlichkeit hängt also die Länge des Zufallsfingerprint logarithmisch von der Stringlänge ab. Für eine fixe Stringlänge hängt andererseits die Länge des Zufallsfingerprint negativ-logarithmisch von der Wahrscheinlichkeit ab.

Es gibt noch drei interessante Sonderfälle für die Wahl von x_0. Im Falle $x_0 = 0$ ist $p_s(x_0) = a_0$, hängt also lediglich von den ersten r Bits des Strings ab. Für $x_0 = 1$ werden die Blöcke einfach summiert und für $x_0 = p$ werden die Blöcke zyklisch rotiert und summiert.

4 Randomisiertes Paging

In [3] wird als Beispiel zum Paging-Problem ein randomisierter Algorithmus namens $RANDMARK$ vorgestellt. Die Analyse in diesem Kapitel ist aufgrund von Missverständnissen in [3], wie die Anzahl der Seitenfehler zustande kommt, geführt worden. Sie bestätigt die Ergebnisse in [3] und soll lediglich zur Beseitigung der Missverständnisse helfen.

Algorithmus 4.1: RANDMARK()

Demarkiere alle Cacheeinträge
for *ever*
do
- Anfrage : Cacheeintrag von p
- **if** p ist im Cache
  - **then** Markiere Cacheeintrag von p, wenn nicht schon markiert
  - **else**
    - **if** kein unmarkierter Cacheeintrag vorhanden
      - **then** Demarkiere alle Cacheeinträge
    - Überschreibe zufällig unmarkierten Cacheeintrag mit p
    - Markiere diesen
- Antwort : Cacheeintrag von p

Proposition 8. *$RANDMARK$ ist $2H_k$-kompetitiv gegen den blinden Gegner OBL.*

Beweis. Der Cache von $RANDMARK$ und OBL ist vor Beginn leer.

Die Anfragefolge σ wird in Phasen partitioniert. Eine neue Phase beginnt immer dann, wenn alle Seiten im Cache unmarkiert sind. Die erste Phase beginnt dann mit der ersten Anfrage. Alle Seiten die sich zu Beginn einer Phase im Cache befinden, werden alte Seiten genannt. Alle anderen Seiten die innerhalb einer Phase angefragt werden, werden neue Seiten genannt. Die Anzahl der alten Seiten wird mit l_i und die Anzahl der neuen Seiten mit m_i bezeichnet.

Eine Phase ist in Normalform, wenn alle neuen Anfragen unterschiedlich sind und die alten Anfragen sich in aufsteigender Reihenfolge innerhalb der Phase auf die Seiten der Cacheplätze 1 bis l_i beziehen. Die folgende Beobachtung zeigt, dass alle Phasen, die nicht in Normalform sind, durch Entfernen der wiederholten Anfragen und einer geeigneten Umbenennung der alten Seiten in eine Kosten-äquivalente Form gebracht werden können: Die erste Anfrage auf eine Seite hinterlässt diese markiert im Cache, d.h. sie wird während Phase nicht mehr entfernt. Alle weiteren Anfragen auf die Seite ändern weder Markierungen, noch erzeugen sie Seitenfehler. Die Reihenfolge der alten Seiten in einer wiederholungsbereinigten Phase ist genau wie die Nummerierung der Cacheplätze aus Symmetriegründen willkürlich. Es werden im folgenden nur noch Phasen in Normalform betrachtet.

Offensichtlich besteht eine Phase (in Normalform) aus $l_i + m_i$ verschiedenen Anfragen. Da ein Cache k Markierungen aufnehmen kann, gilt $k = m_i + l_i$. Man beobachte außerdem, dass die erste Anfrage in einer Phase immer eine neue Seite betrifft, sonst könnte die davor liegende Phase erweitert werden.

Es wird nun gezeigt, dass Phasen, in denen alle neuen Anfragen vor allen alten Anfragen kommen, die meisten erwarteten Seitenfehler liefern. Dazu wird eine Stelle in einer Phase betrachtet, an der hintereinander eine alte und eine neue Anfrage kommen. Die folgende Argumentation zeigt, dass das Vertauschen der Anfragen die erwarteten Seitenfehler erhöht. Folgende Fälle können eintreten:

1. Die alte Anfrage findet ihren Eintrag nicht im Cache. Alte und neue Anfrage erzeugen beide einen Seitenfehler und belegen zwei beliebige unmarkierte Cacheplätze. Die Reihenfolge von alter und neuer Anfrage spielt offensichtlich keine globale Rolle.

2. Die alte Anfrage findet ihren Eintrag (unmarkiert) im Cache. Allein die neue Anfrage erzeugt einen Seitenfehler und belegt einen beliebigen unmarkierten Cacheplatz. Danach sind also der Cacheplatz des alten Eintrags und ein beliebiger anderer unmarkierter Eintrag markiert. Falls man nun die Reihenfolge der Anfragen vertauscht, können folgende Fälle auftreten:

   (a) Die neue Anfrage belegt einen beliebigen unmarkierten Cacheplatz - jedoch nicht den der alten Anfrage. In diesem Fall erzeugt die Teilsequenz einen Seitenfehler, danach ist der alte Cacheplatz und ein beliebiger anderen unmarkierter Cacheplatz markiert - global gesehen ist die Vertauschung also irrelevant.

   (b) Die neue Anfrage belegt den unmarkierten Cacheplatz der alten Anfrage. Dann wird die alte Anfrage einen Seitenfehler erzeugen und einen beliebigen unmarkierten Cacheplatz belegen. Auch hier erkennt man, dass sich zwar die Anzahl der Seitenfehler erhöht, was gezeigt werden sollte, jedoch die Situation danach statistisch äquivalent zur Situation des Nicht-Vertauschens ist, da wiederum der alte und ein beliebiger anderer unmarkierter Cacheplatz markiert ist.

Im folgenden werden nur noch Phasen betrachtet, in denen alle Anfragen auf neue Seiten vor allen Anfragen auf alte Seiten kommen, da diese, wie oben gezeigt, die meiste Anzahl an erwarteten Seitenfehlern einbringen.

Bezeichne p_j die Wahrscheinlichkeit, dass der Cacheplatz j vor Eintreffen der j-ten alten Anfrage überschrieben wurde, dass ist gleich der Wahrscheinlichkeit, dass der Cacheplatz vor der j-ten alten Anfrage bereits markiert wurde. Mit Wahrscheinlichkeit p_j liefert die j-te Anfrage einen Seitenfehler. Per Induktion wird nun gezeigt, dass nach der j-ten Anfrage auf alte Seiten die Cacheplätze 1 bis j markiert sind. Außerdem wird innerhalb des Induktionsschritts p_j berechnet.

1. Für $j = 0$ (vor der ersten Anfrage) gilt dies offensichtlich, da hier nichts behauptet wird.

2. Sei nun die j-te alte Anfrage betrachtet. Es gab in der Phase bisher $m_i + j - 1$ Anfragen an den Cache und somit sind $m_i + j - 1$ Cacheplätze

bereits markiert. Nach Induktionsvoraussetzung sind die Cacheplätze 1 bis $j-1$ markiert, somit verteilen sich die restlichen $m_i + j - 1 - (j-1) = m_i$ Markierungen auf die Cacheplätze j bis k. Aus Symmetriegründen ist die Belegungswahrscheinlichkeit für alle Cacheplätze j bis k gleich, da keiner der Cachplätze j bis k bisher ausgezeichnet war. Somit muss die Belegung gleichverteilt auf die Cacheplätze j bis k sein. Dann ist die Wahrscheinlichkeit für die Markierung des j-ten Cacheplatzes vor der j-ten alten Anfrage natürlich $p_j = m_i/(k-j+1)$. Falls der j-te Cacheplatz nicht bereits markiert wurde, so wird er durch $RANDMARK$ markiert, da $RANDMARK$ dort die alte Seite findet. Somit sind nun die Cacheplätze 1 bis j markiert.

Für die erwartete Anzahl Seitenfehler der Phase erhält man

$$\begin{aligned}\mathbb{E}[X_i] &= m_i + \sum_{j=1}^{k-m_i} p_j \\ &= m_i + \sum_{j=1}^{k-m_i} \frac{m_i}{k-j+1} \\ &= m_i(H_k - H_{m_i} + 1) \leq m_i H_k.\end{aligned}$$

Mit $\mathbb{E}[RANDMARK(\sigma)] = \sum_i \mathbb{E}[X_i]$ gilt

$$\mathbb{E}[RANDMARK(\sigma)] \leq H_k \sum_i m_i \tag{14}$$

Innerhalb zweier aufeinanderfolgender Phasen werden $k + k - l_i = k + m_i$ Seiten abgefragt, wovon OPT nur k im Cache halten kann. Deshalb muss OPT mindestens m_i Seitenfehler in den zwei aufeinanderfolgenden Phasen $i-1$ und i haben. Deshalb gilt $OPT(\sigma) \geq \sum_i m_{2i}$ und $OPT(\sigma) \geq \sum_i m_{2i+1}$. Summiert ergibt sich

$$\begin{aligned}2OPT(\sigma) &\geq \left(\sum_i m_{2i} + \sum_i m_{2i+1}\right) \\ OPT(\sigma) &\geq \frac{1}{2}\sum_i m_i.\end{aligned} \tag{15}$$

Durch Vergleich von (15) und (14) ergibt sich die Behauptung.

□

5 Quellen

[1] Reinhard Diestel. *Graph theory*. Springer-Verlag, New York, 2 edition, 2000.

[2] Ina Kersten and Ole Riedlin. *Algebra*. Georg-August-Universität Göttingen, Göttingen, 2001.
http://www.uni-math.gwdg.de/skripten/Algebraskript/algebra.pdf.

[3] Sven O. Krumke and Jörg Rambau. *Online Optimierung*. Technische Universität Berlin, Berlin, 2000.
http://www.zib.de/krumke/Teaching/OnlineSS2000/skript-neu.pdf.

[4] Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Cambridge University Press, 1995.