4 Routing in Hyperwürfeln

Anhand einer Problemstellung, die bei Parallelrechnern mit einer entsprechenden Verbindungsstruktur aufträte, sollen in diesem Kapitel zwei weitere Aspekte vertieft werden, die bei randomisierten Algorithmen bzw. ihrer Analyse von Bedeutung sind: Chernoff-Schranken und die probabilistische Methode.

4.1 Das Problem und ein deterministischer Algorithmus

4.1 Definition Für $d \geqslant 1$ ist der d-*dimensionale Hyperwürfel* H_d gegeben durch die Menge der Knoten $V_d = \{0,1\}^d$ und die Menge der Kanten E_d, die zwei Knoten x und y genau dann miteinander verbinden, wenn sich die Bitfolgen x und y an genau einer Stelle unterscheiden. ◇

4.2 H_d hat also $N = 2^d$ Knoten und $d \cdot 2^{d-1} \in \Theta(N \log N)$ Kanten.

Der Durchmesser von H_d ist $d = \log N$. Ist nämlich $x = (x_1 x_2 \cdots x_d)$ und $y = (y_1 y_2 \cdots y_d)$ (mit $x_i, y_j \in \{0,1\}$), so ergibt sich aus der Knotenfolge

$$(x_1 x_2 \cdots x_d), (y_1 x_2 \cdots x_d), \ldots, (y_1 y_2 \cdots y_{d-1} x_d), (y_1 y_2 \cdots y_d)$$

nach Entfernen aller aufeinander folgender doppelter Knoten ein Pfad von x nach y. Er hat maximal Länge d, da höchstens so viele Bits unterschiedlich sind.

4.3 Problem. Beim *Permutationsrouting* stellt man sich vor, die Knoten x von H_d seien Prozessoren, auf denen jeweils eine „Nachricht“ (oder auch ein „Paket“) vorliege, die auf einem Pfad in H_d zu einem Zielknoten $f(x)$ transportiert werden muss. Das Präfix „Permutation“ bedeutet dabei, dass $f\colon V \to V$ eine Bijektion ist, also eine Permutation der Knoten beschreibt.

Es gibt die Einschränkung, dass in jedem Schritt über jede Kante maximal ein Paket transportiert werden kann. Entsteht die Situation, dass mehrere Pakete gleichzeitig über die gleiche Kante weitergeschickt werden sollen, so werde ein Paket bedient, während die anderen in einer FIFO-Warteschlange gespeichert und später bedient werden.

Die Aufgabe besteht darin, für jedes Paar $(x, f(x))$ einen „Reiseplan“ (Kanten und Zeitpunkte für deren Benutzung) von x nach $f(x)$ festzulegen, so dass alle Pakete möglichst schnell ans Ziel kommen und dabei obiger Einschränkung Genüge getan wird.

4.4 Insbesondere wollen wir uns im folgenden nur für Algorithmen interessieren, die im Englischen als *oblivious* und im Deutschen (manchmal) als *datenunabhängig* bezeichnet werden. Damit ist gemeint, dass die Route für Paket x nicht von den Routen der anderen Pakete abhängt.

4.5 Die Argumentation in Punkt 4.2 für den Durchmesser von H_d liefert auch gleich ein Verfahren für die Routenwahl. Dieser „Bit-Fixing-Algorithmus“ geht auf Valiant zurück.

4.6 Müsste man insgesamt nur *ein* Paket transportieren, so würden offensichtlich $d = \log N$ Schritte dafür ausreichen. Schwierig wird es dadurch, dass mehrere Pakete gleichzeitig transportiert werden müssen und sie sich unter Umständen gegenseitig behindern.

Für den Bit-Fixing-Algorithmus liefert zum Beispiel die folgende Permutation („Matrix-Transposition“) sehr lange Transportzeiten: $f(x_1 \cdots x_{d/2} x_{d/2+1} \cdots x_d) = (x_{d/2+1} \cdots x_d x_1 \cdots x_{d/2})$. Der Grund ist schnell zu sehen: Für jedes beliebige Bitmuster $z_{d/2+1} \cdots z_d$ werden alle $2^{d/2} = \sqrt{N}$ Pakete, die in einem der Knoten $x_1 \cdots x_{d/2} z_{d/2+1} \cdots z_d$ starten, über den gleichen Knoten $z_{d/2+1} \cdots z_d z_{d/2+1} \cdots z_d$ geschickt. Da in jedem Schritt maximal d Pakete diesen Knoten verlassen können, ergibt sich für diesen Algorithmus eine untere Schranke von $\sqrt{N}/d$ Schritten.

Weniger offensichtlich ist, dass man diese Beobachtung verallgemeinern kann.

Die erste diesbezügliche Arbeit stammt von Borodin und Hopcroft (1985). Die nachfolgende Verschärfung geht im wesentlichen auf Kaklamanis, Krizanc und Tsantilas (1990) zurück.

4.7 Satz. *Zu jedem deterministischen datenunabhängigen Algorithmus für Permutationsrouting in einem Graphen mit N Knoten, die alle Ausgangsgrad d haben, gibt es eine Permutation, für die der Algorithmus $\Omega(\sqrt{N}/d)$ Schritte benötigt.*

4.8 Beweis. Der folgende Beweis stammt aus dem Buch von Leighton (1992).

Es sei A ein deterministischer datenunabhängiger Algorithmus für Permutationsrouting. Der von A für ein Paket von u nach v gewählte Pfad werde mit $P_{u,v}$ bezeichnet. A ist also durch die N^2 Pfade $P_{u,v}$ für alle Knoten $u, v \in V$ eindeutig charakterisiert.

Die Beweisidee besteht darin, „große“ Mengen von Quellknoten $U' = \{u_1, \ldots, u_k\}$ und zugehörigen Zielknoten $V' = \{v_1, \ldots, v_k\}$ zu finden, so dass alle Pfade P_{u_i,v_i} über eine gleiche Kante e führen. Da jede Kante in jedem Schritt nur je ein Paket in jede Richtung transportieren kann, ergibt sich hieraus eine untere Schranke von $k/2$. Wir werden sehen, dass man $k = \sqrt{N}/d$ solche Pfade finden kann.

Man betrachte einen beliebigen Knoten v und alle $N-1$ Pfade $P_{u,v}$, die von allen anderen Knoten zu ihm führen. Für $k \geqslant 1$ bezeichne $S_k(v)$ die Menge aller Kanten, durch die mindestens k dieser Pfade führen. $S_k^*(v)$ bezeichne die Menge aller Endknoten der Kanten in $S_k(v)$. Offensichtlich ist $|S_k^*(v)| \leqslant 2|S_k(v)|$.

Der Beweis wird in mehreren Schritten geführt.

1. Da $N-1$ Pfade zu v hinführen, aber nur d Kanten, müssen über mindestens eine dieser Kanten mindestens $\frac{N-1}{d}$ Pfade führen. Also ist für $k \leqslant \frac{N-1}{d}$ auch $v \in S_k^*(v)$.
2. Es sei von nun an stets $k \leqslant \frac{N-1}{d}$ und daher $v \in S_k^*(v)$.
3. Als nächstes soll gezeigt werden:

$$|V \setminus S_k^*(v)| \leqslant (d-1)(k-1)|S_k^*(v)| \tag{4.1}$$

Wegen der eben gemachten Annahme führt jeder Pfad $P_{u,v}$ von einem Knoten $u \in V \setminus S_k^*(v)$ „nach $S_k^*(v)$ hinein“. Für das jeweils erste solche „Hineinführen“ über eine Kante $(w, w') \in V \setminus S_k^*(v) \times S_k^*(v)$ gilt:

- Eine solche Kante gehört nicht zu $S_k(v)$, denn sonst wäre ja schon $w \in S_k^*(v)$.
- Es gibt $|S_k^*(v)|$ mögliche w'.
- Zu jedem solchen w' führen maximal $d-1$ Kanten „von außerhalb“, denn mindestens eine der mit w' inzidenten Kanten muss ja in $S_k(v)$ liegen.
- Über eine solche Kante (w, w') können höchstens $k-1$ Pfade führen, denn sonst würde diese Kante ja schon zu $S_k(v)$ gehören.

Also kann es „außerhalb“ von $S_k^*(v)$, also in $V \setminus S_k^*(v)$, nur die in Gleichung 4.1 behauptete Anzahl von Knoten geben.

4. Folglich gilt für jedes $k \leqslant (N-1)/d$:

$$\begin{aligned} N &= |V \setminus S_k^*(v)| + |S_k^*(v)| \\ &\leqslant (d-1)(k-1)|S_k^*(v)| + |S_k^*(v)| \\ &\leqslant ((d-1)(k-1)+1) \cdot 2|S_k(v)| \\ &\leqslant 2kd|S_k(v)| \\ \text{und daher} \qquad |S_k(v)| &\geqslant \frac{N}{2kd} . \end{aligned}$$

Summation über alle Knoten ergibt

$$\sum_{v \in V} |S_k(v)| \geqslant \frac{N^2}{2kd} .$$

Da es aber maximal $Nd/2$ Kanten im Graphen gibt, muss mindestens eine Kante in mindestens

$$\frac{N^2/2kd}{Nd/2} = \frac{N}{kd^2}$$

Mengen $S_k(v)$ vorkommen. Wir wählen nun k so, dass diese Anzahl gerade wieder k ist, also $k = \sqrt{N}/d$. (Dieses k ist kleiner gleich $(N-1)/d$.)

5. Es sei nun e eine Kante, die in $k = \sqrt{N}/d$ Mengen $S_k(v_1), \ldots, S_k(v_k)$ liegt. D. h. über e führen mindestens $k = \sqrt{N}/d$ Pfade zu v_1, und über e führen mindestens $k = \sqrt{N}/d$ Pfade zu v_2, usw..

   Es sei u_1 einer der k Knoten, für die P_{u_1,v_1} über e führt.

   Nach unserer Wahl von k im vorangegangenen Punkt gibt es zu jedem v_i mindestens k Knoten, für die P_{u_i,v_i} über e führt. Daher können wir induktiv u_i festlegen, indem wir verlangen, dass u_i einer der mindestens $k-(i-1)$ Knoten ungleich $u_1, \ldots, u_{i-1}$ sei, für die P_{u_i,v_i} über e führt.

Also gibt es mindestens $k = \sqrt{N}/d$ Pfade $P_{u_1,v_1}, \ldots, P_{u_k,v_k}$, die alle über die gleiche Kante e führen. ∎

4.9 Man könnte argwöhnen, dass es zumindest einen deterministischen datenunabhängigen Algorithmus gibt, für den nur „sehr sehr wenige“ Permutationen tatsächlich „sehr schlimm“ sind, so dass man in Anwendungen „normalerweise“ gar nicht das konstruierte Problem hat. Leider ist das nicht so.

Man kann zeigen, dass es für jeden deterministischen datenunabhängigen Algorithmus sogar $(\sqrt{N}/d)!$ Permutationen gibt, die mindestens $\sqrt{N}/2d$ Routingschritte nötig machen.

Wir beenden den ersten Abschnitt mit einem Überblick über den Rest dieses Kapitels.

In Abschnitt 4.4 werden wir einen ersten randomisierten Algorithmus für Permutationsrouting kennenlernen, bei dem der Erwartungswert für den Zeit, alle Pakete an ihre Ziele zu transportieren, nur $O(\log N)$ ist. Man vergleiche dies mit der unteren Schranke von $\Omega(\sqrt{N}/\log N)$ für deterministische Algorithmen!

Für die Analyse des randomisierten Algorithmus werden wir Chernoff-Schranken benutzen, die Gegenstand von Abschnitt 4.3 sind. Für die dortigen Beweise werden die Ungleichungen von Markov und Chebyshev benötigt, auf die wir zur Vorbereitung in Abschnitt 4.2 kurz eingehen.

In Abschnitt 4.5 wird genauer auf die probabilistische Methode eingegangen werden, die in Abschnitt 4.6 bei der Analyse eines zweiten randomisierten Algorithmus für Permutationsrouting benutzt werden wird.

4.2 Markov- und Chebyshev-Ungleichung

4.10 Satz. (Markov-Ungleichung) *Es sei Y eine Zufallsvariable, die nur nichtnegative Werte annehme. Dann gilt für alle $t, k \in \mathbb{R}_+$:*

$$\mathbf{P}(Y \geqslant t) \leqslant \frac{\mathbf{E}[Y]}{t} \quad \textit{bzw.} \quad \mathbf{P}(Y \geqslant k\mathbf{E}[Y]) \leqslant \frac{1}{k} .$$

4.11 Beweis. Man betrachte die Zufallsvariable

$$X = \begin{cases} 0 & \text{falls } Y < t \\ 1 & \text{falls } Y \geqslant t \end{cases}$$

Dann ist $X \leqslant Y$ und $\mathbf{E}[X] \leqslant \mathbf{E}[Y]$. Offensichtlich ist $\mathbf{E}[X] = 0 \cdot \mathbf{P}(Y < t) + t \cdot \mathbf{P}(Y \geqslant t) = t \cdot \mathbf{P}(Y \geqslant t)$. Also ist $t \cdot \mathbf{P}(Y \geqslant t) \leqslant \mathbf{E}[Y]$ und $\mathbf{P}(Y \geqslant t) \leqslant \mathbf{E}[Y]/t$. ∎

Weiß man mehr über die zur Rede stehende Zufallsvariable, zum Beispiel ihre Varianz, dann kann man auch schärfere Abschätzungen angeben:

4.12 Satz. (Chebyshev-Ungleichung) *Es sei X eine Zufallsvariable mit Erwartungswert μ_X und Standardabweichung σ_X. Dann gilt für alle $t \in \mathbb{R}_+$:*

$$\mathbf{P}(|X - \mu_X| \geqslant t\sigma_X) \leqslant \frac{1}{t^2} .$$

4.13 Beweis. Die Zufallsvariable $Y = (X - \mu_X)^2$ hat Erwartungswert $\mu_Y = \sigma_X^2$. Die Markov-Ungleichung ist anwendbar und liefert

$$\mathbf{P}\left(Y \geqslant t^2\mu_Y\right) \leqslant \frac{1}{t^2} .$$

Der Wert auf der linken Seite ist aber

$$\mathbf{P}\left(Y \geqslant t^2\mu_Y\right) = \mathbf{P}\left((X - \mu_X)^2 \geqslant t^2\sigma_X^2\right) = \mathbf{P}(|X - \mu_X| \geqslant t\sigma_X)$$

∎

4.3 Chernoff-Schranken

Die Bezeichnung Chernoff-Schranken geht auf die Arbeit von Chernoff (1952) zurück, der die ersten derartigen Resultate bewies. Heute fasst man den Begriff etwas weiter. Eine kompakte Zusammenfassung entsprechender Ergebnisse stammt von Hagerup und Rüb (1990).

4.14 Im folgenden seien stets $X_1, \ldots, X_n$ unabhängige 0-1-Zufallsvariablen mit $\mathbf{P}(X_i = 1) = p_i$ für $1 \leqslant i \leqslant n$. Solche Zufallsvariablen heißen auch *Poisson-Versuche*. Außerdem sei $X = X_1 + \cdots + X_n$ und $\mu = \mathbf{E}[X] = \sum_{i=1}^{n} p_i$. Im Fall, dass alle $p_i = p$ sind, spricht man auch von *Bernoulli-Versuchen*, und X ist binomialverteilt.

In letzterem Fall ist also $\mu = np$,

$$\mathbf{P}(X = k) = \binom{n}{k} p^k (1-p)^{n-k} \qquad \text{und} \qquad \mathbf{P}(X \geqslant k) = \sum_{j=k}^{n} \binom{n}{j} p^j (1-p)^{n-j} .$$

4.15 Satz. *Mit den Bezeichnungen wie in Punkt 4.14 gilt:*

1. *für* $0 \leqslant \delta$: $\mathbf{P}(X \geqslant (1+\delta)\mu) \leqslant \left(\dfrac{e^{\delta}}{(1+\delta)^{(1+\delta)}} \right)^{\mu}$.
2. *für* $1 > \delta \geqslant 0$: $\mathbf{P}(X \leqslant (1-\delta)\mu) \leqslant \left(\dfrac{e^{-\delta}}{(1-\delta)^{(1-\delta)}} \right)^{\mu}$.

In Abbildung 4.1 sind die beiden auf den rechten Seiten auftretenden Abbildungen für den Bereich zwischen 0 und 3 bzw. zwischen 0 und 1 geplottet.

Abbildung 4.1: Die Funktionen $\frac{e^x}{(1+x)^{(1+x)}}$ und $\frac{e^{-x}}{(1-x)^{(1-x)}}$.

4.16 Dem Beweis sei kurz die folgende Überlegung vorangeschickt, die zeigt, dass für $x \geqslant 0$ stets $1 + x \leqslant e^x$ ist: Die Ableitung von $f(x) = e^x - (x+1)$ ist $e^x - 1$, ist also nichtnegativ für $x \geqslant 0$. Folglich nimmt $f(x)$ im Bereich $x \geqslant 0$ für $x = 0$ den minimalen Wert $f(0) = 0$ an. Also ist dort $f(x) \geqslant 0$.

Analog kann man zeigen, dass für $x \geqslant 0$ stets $1 - x \leqslant e^{-x}$ ist.

4.17 Beweis. (von Satz 4.15) Wie die ähnlichen Formeln in Satz 4.15 schon vermuten lassen, können die beiden Ungleichungen ähnlich bewiesen werden.

1. Mit Hilfe der Markov-Ungleichung erhält man zunächst für jede positive Konstante t:

$$\mathbf{P}(X \geqslant (1+\delta)\mu) = \mathbf{P}\left(e^{tX} \geqslant e^{t(1+\delta)\mu}\right) \leqslant \frac{\mathbf{E}\left[e^{tX}\right]}{e^{t(1+\delta)\mu}} .$$

Mit den X_i sind auch die e^{tX_i} unabhängige Zufallsvariablen. Deshalb kann man den Zähler umformen gemäß:

$$\mathbf{E}\left[e^{tX}\right] = \mathbf{E}\left[e^{t\sum X_i}\right] = \mathbf{E}\left[\prod e^{tX_i}\right] = \prod \mathbf{E}\left[e^{tX_i}\right] .$$

Weiter ist

$$\mathbf{E}\left[e^{tX_i}\right] = p_i \cdot e^t + (1-p_i) \cdot 1 = 1 + p_i(e^t - 1)$$

woraus sich mit der Abschätzung $1 + x \leqslant e^x$ (siehe Punkt 4.16) ergibt:

$$\mathbf{E}\left[e^{tX_i}\right] \leqslant e^{p_i(e^t-1)}$$

Damit erhält man:

$$\mathbf{P}(X \geqslant (1+\delta)\mu) \leqslant \frac{\prod e^{p_i(e^t-1)}}{e^{t(1+\delta)\mu}} = \frac{e^{\sum p_i(e^t-1)}}{e^{t(1+\delta)\mu}} = \frac{e^{\mu(e^t-1)}}{e^{t(1+\delta)\mu}} .$$

Die gewünschte Ungleichung ergibt sich durch Einsetzen von $t = \ln(1+\delta)$. Man beachte, dass dieser Wert tatsächlich positiv ist.

2. Für jede positive Konstante t ist

$$\mathbf{P}(X \leqslant (1-\delta)\mu) = \mathbf{P}(\mu - X \geqslant \delta\mu) = \mathbf{P}\Big(e^{t(\mu-X)} \geqslant e^{t\delta\mu}\Big) \leqslant \frac{\mathbf{E}\Big[e^{t(\mu-X)}\Big]}{e^{t\delta\mu}} = \frac{\mathbf{E}\big[e^{-tX}\big]}{e^{t(\delta-1)\mu}} \,.$$

Ähnlich wie im ersten Fall ist

$$\begin{aligned}\mathbf{E}\Big[e^{-tX}\Big] &= \prod \mathbf{E}\big[e^{-tX_i}\big] = \prod(p_i e^{-t} + (1-p_i)) = \prod(1 - p_i(1-e^{-t})) \\ &\leqslant \prod e^{-p_i(1-e^{-t})} = e^{\sum -p_i(1-e^{-t})} = e^{-\mu(1-e^{-t})} \,.\end{aligned}$$

Die gewünschte Ungleichung ergibt sich Einsetzen von $t = -\ln(1-\delta)$. Man beachte, dass dieser Wert tatsächlich positiv ist.

■

4.18 Bemerkung. Zur Vorbereitung von gelegentlich einfacher handzuhabenden Korollaren zeigen wir zunächst einige technische Ergebnisse. In Satz 4.15 spielt der Ausdruck $F(\mu,\delta) = \left(\frac{e^\delta}{(1+\delta)^{(1+\delta)}}\right)^\mu$ für $\delta > -1$ eine Rolle. In diesem Bereich ist $F(\mu,\delta) > 0$.

Dieser Ausruck soll nun nach oben durch etwas gröbere, aber leichter handzuhabende Ausdrücke abgeschätzt werden. Durch Potenzieren mit $1/\mu$ und Logarithmieren ergibt sich der Ausdruck $f(\delta)$, wobei $f(x)$ die Funktion $f(x) = x - (1+x)\ln(1+x)$ ist.

Da Potenzieren mit $1/\mu$ und μ sowie Logarithmieren und Exponentiation für nichtnegative Argumente monotone Funktionen sind, folgt aus einer Abschätzung $f(x) \leqslant k(x)$ auch eine Abschätzung für den interessierenden Ausdruck $F(\mu,\delta) = e^{f(\delta)\mu} \leqslant e^{k(\delta)\mu}$.

Im folgenden werden $f(x)$ und $g(x) = f(x)/x^2$ näher untersucht. In Abbildung 4.2 sind sie und zwei quadratische Polynome geplottet.

Abbildung 4.2: Die Funktionen $f(x) = x - (1+x)\ln(1+x)$, $g(x) = f(x)/x^2$, $-x^2/5$ und $-x^2/2$ im Vergleich.

4.19 Lemma.

1. Für $-1 < x \leqslant 0$ gilt: $f(x) \leqslant -x^2/2$.
2. Für $0 < x$ gilt: $-x^2/2 \leqslant f(x)$.
3. Die Funktion $g(x) = f(x)/x^2$ ist monoton wachsend.
4. Für $0 < \delta < x$ gilt: $f(\delta) \leqslant g(x)\delta^2$.
5. Für $0 < \delta < 2e-1$ gilt: $f(\delta) \leqslant -\delta^2/5$.
6. Für $0 < \delta < 1$ gilt: $f(\delta) \leqslant -\delta^2/3$.

4.20 Beweis.

1. Es sei $-1 < x \leqslant 0$. Man betrachte $h(x) = f(x) + x^2/2$. Wegen $\frac{d}{dx} x\ln x = 1\cdot\ln x + x\frac{1}{x} = 1 + \ln x$ ist $h'(x) = 1 - 1 + \ln(1+x) + x = x + \ln(1+x)$. Daraus folgt $h''(x) = 1 - \frac{1}{1+x}$. Im betrachteten Intervall $-1 < x \leqslant 0$ ist $h''(x) \leqslant h''(0) = 0$. Also ist dort h' fallend, d. h. $h'(x) \geqslant h'(0) = 0$ und daher h steigend, d. h. $h(x) \leqslant h(0) = 0$.
2. Für $x \geqslant 0$ ist analog $h''(x) \geqslant h''(0) = 0$, also $h'(x) \geqslant h'(0) = 0$ und daher $h(x) \geqslant h(0) = 0$.

3. Im folgenden sei stets $x \geqslant 0$ und statt $\ln(1+x)$ schreiben wir kurz $l(x)$. Es ist $g(0) = 0$. Wir zeigen, dass $g'(x) \geqslant 0$ ist. Die Quotientenregel ergibt $\frac{dg}{dx} = \frac{-x^2 l(x) - 2x^2 + 2x(1+x)l(x)}{x^4}$. Dies ist offensichtlich genau dann größer gleich Null, wenn $h(x) = xl(x) + 2l(x) - 2x$ (Zähler durch x) größer gleich Null ist. Es ist $h(0) = 0$. Wir zeigen, dass $h'(x) \geqslant 0$ ist. Ableiten ergibt $h'(x) = -x/(1+x) + l(x)$, also $h'(0) = 0$ und $h''(x) = x/(1+x)^2 \geqslant 0$. Hieraus folgt das Gewünschte.
4. Die Behauptung ist eine einfache Umformulierung der eben gezeigten Monotonieeigenschaft.
5. Einfaches Nachrechnen ergibt

$$g(2e-1) = \frac{2e-1-2e\ln(2e)}{(2e-1)^2} = \frac{2e-1-2e(1+\ln 2)}{(2e-1)^2} < -\frac{1}{5}\,.$$

   Somit folgt die Behauptung aus dem vorgegangenen Punkt.
6. Analog zu eben berechnet man $g(1) = f(1) = 1 - 2\ln 2 < -\frac{1}{3}$ und die Behauptung folgt analog wie eben.

■

4.21 Korollar. Mit den Bezeichnungen wie in Punkt 4.14 gilt

$$\begin{aligned}
\text{für } 0 \leqslant \delta \leqslant 2e-1: &\qquad \mathbf{P}(X \geqslant (1+\delta)\mu) \leqslant \left(\frac{e^\delta}{(1+\delta)^{(1+\delta)}}\right)^\mu \leqslant e^{-\delta^2\mu/5} \\
\text{für } 0 \leqslant \delta \leqslant 1: &\qquad \mathbf{P}(X \geqslant (1+\delta)\mu) \leqslant \left(\frac{e^\delta}{(1+\delta)^{(1+\delta)}}\right)^\mu \leqslant e^{-\delta^2\mu/3} \\
\text{und für } 1 > \delta \geqslant 0: &\qquad \mathbf{P}(X \leqslant (1-\delta)\mu) \leqslant \left(\frac{e^{-\delta}}{(1-\delta)^{(1-\delta)}}\right)^\mu \leqslant e^{-\delta^2\mu/2} \leqslant \left(\frac{e^\delta}{(1+\delta)^{(1+\delta)}}\right)^\mu .
\end{aligned}$$

4.22 Beweis. Die nachzuweisenden Ungleichungen lassen sich auf die aus Lemma 4.19 zurückführen (siehe Bemerkung 4.18). ■

Das folgende Korollar liefert weitere noch einfachere Abschätzungen für Abweichungen nach oben. An ihnen wird auch klar, warum in Lemma 4.21 dem Bereich $0 < \delta < 2e-1$ besondere Aufmerksamkeit gewidmet wurde.

4.23 Korollar. Es ist

$$\begin{aligned}
\text{für } 0 \leqslant \delta: &\qquad \mathbf{P}(X \geqslant (1+\delta)\mu) \leqslant \left(\frac{e}{1+\delta}\right)^{(1+\delta)\mu} \\
\text{für } 2e-1 \leqslant \delta: &\qquad \mathbf{P}(X \geqslant (1+\delta)\mu) \leqslant 2^{-(1+\delta)\mu}\,.
\end{aligned}$$

4.24 Beweis. Nach Satz 4.15 ist

$$\mathbf{P}(X \geqslant (1+\delta)\mu) \leqslant \left(\frac{e^\delta}{(1+\delta)^{(1+\delta)}}\right)^\mu \leqslant \left(\frac{e\cdot e^\delta}{(1+\delta)^{(1+\delta)}}\right)^\mu = \left(\frac{e}{1+\delta}\right)^{(1+\delta)\mu}\,.$$

Für $\delta \geqslant 2e-1$ ergibt sich:

$$\mathbf{P}(X \geqslant (1+\delta)\mu) \leqslant \left(\frac{e}{1+\delta}\right)^{(1+\delta)\mu} \leqslant \left(\frac{e}{1+2e-1}\right)^{(1+\delta)\mu} = \left(\frac{1}{2}\right)^{(1+\delta)\mu}\,.$$

■

4.4 Erster randomisierter Algorithmus

Wir wollen nun einen ersten randomisierten Algorithmus für das betrachtete Routingproblem beschreiben und analysieren.

Die Vorgehensweise ist sehr einfach:

4.25 Algorithmus.

1. Für jedes Paket b_x in Startknoten x wird unabhängig und gleichverteilt zufällig ein Zwischenknoten z_x gewählt.
2. Unter Verwendung des Bit-Fixing-Algorithmus wird jedes b_x von x nach z_x transportiert.
3. Unter Verwendung des Bit-Fixing-Algorithmus wird jedes b_x von z_x zu Zielknoten $f(x)$ transportiert.

Kommt es zwischendurch zu Staus, so mögen die Pakete jeweils nach der FIFO-Strategie behandelt werden. Kommen mehrere Pakete gleichzeitig in einem Knoten an und müssen weitergeschickt werden, so mögen sie in einer beliebigen Reihenfolge eingeordnet werden.

Man beachte, dass es vorkommen kann, dass Pakete von mehreren Startknoten zum gleichen Zwischenknoten transportiert werden müssen. Wie sich als Nebenprodukt aus der anschließend durchgeführten Analyse ergeben wird, ist es aber sehr unwahrscheinlich, dass es dadurch zu großen Staus kommt.

4.26 Satz.

1. *Die Wahrscheinlichkeit, dass jedes Paket seinen Zwischenknoten nach spätestens* $7d$ *Schritten erreicht hat, ist mindestens* $1 - 2^{-5d}$.
2. *Die Wahrscheinlichkeit, dass jedes Paket sein Ziel nach spätestens* $14d$ *Schritten erreicht hat, ist mindestens* $1 - \frac{2}{N^5}$.
3. *Für* $d \geqslant 3$ *ist der Erwartungswert für die Laufzeit von Algorithmus 4.25 kleiner oder gleich* $14d + 1$.

Wesentliche Teile des Beweises handeln wir in vorangestellten Lemmata ab. Dazu sei für jeden Startknoten x zufällig ein z_x gewählt und ρ_x bezeichne den Pfad von x nach z_x gemäß dem Bit-Fixing-Algorithmus.

4.27 Lemma. Es sei x beliebig aber fest und $\rho_x = (e_1, e_2, \ldots, e_k)$. Es sei $S_x = \{b_y \mid y \neq x \wedge \rho_y \cap \rho_x \neq \emptyset\}$, wobei die unsaubere Schreibweise $\rho_y \cap \rho_x$ zu interpretieren sei als die Menge der Kanten, die in beiden Pfaden irgendwo vorkommen. Bezeichnet t den tatsächlichen Ankuftszeitpunkt von b_x in z_x, so ist die dann aufgelaufene „Verspätung“ $\ell_x = t - k$ kleiner oder gleich $|S_x|$.

4.28 Beweis.

1. Haben zwei Pfade ρ_x und ρ_y eine oder mehrere Kanten gemeinsam, so gilt: Sobald sich die Pfade das erste Mal wieder getrennt haben, führen sie nicht wieder zusammen.

   Man betrachte die letzte gemeinsame Kante vor der ersten Trennung und den Knoten v, zu dem sie hinführt. Es gibt zwei Fälle:

- Bei beiden Pfaden folgt noch mindestens eine weitere Kante. Da es sich um verschiedene Kanten handelt, werden verschiedene Bits der Adresse v geändert. Es seien j_1 und $j_2 > j_1$ die Positionen dieser Bits. Also wird auf dem einen Pfad das j_1-te Bit geändert, auf dem anderen aber nicht. Die Pfade ρ_x und ρ_y führen also in die disjunkten Hyperwürfel, die durch eine 0 bzw. 1 als j_1-ten Adressbit gekennzeichnet sind, und verlassen sie nie wieder, da bei beiden nur noch andere Adressbits angepasst werden.
- Nur ein Pfad führt weiter, der andere endet in v. In diesem Fall kann man analog argumentieren.

2. Wir vereinbaren zunächst folgende Sprechweisen:

   - Ein Paket $b_y \in S_x$ *verlasse* ρ_x, wenn es zum letzten Mal eine Kante von ρ_x benutzt. Achtung: Man lese den letzten Satz noch einmal und beachte die etwas merkwürdige Wortwahl!

     Wegen der Aussage im ersten Punkt ist der Zeitpunkt des „Verlassens“ für jedes Paket b_y eindeutig.
   - Das Paket b_x bzw. ein Paket $b_y \in S_x$ habe beim Transport über Kante e_i von ρ_x *Verspätung* ℓ, falls es erst in Schritt $t = i + \ell$ darüber transportiert wird.

     Für b_x ist die so definierte Verspätung beim Transport über Kante e_i wirklich die Zeitdifferenz zwischen dem frühest möglichen Zeitpunkt $t = i$ der Ankunft am Endpunkt von e_i und dem tatsächlichen Ankunftszeitpunkt $t = i + \ell$. Auf die Pakete $b_y \in S_x$ trifft dies nicht zu. Für sie handelt es sich im allgemeinen nur um „irgendeine“ Zahl.

3. Wir zeigen nun, dass jedes Mal, wenn sich die Verspätung von Paket b_x von ℓ auf $\ell + 1$ erhöht, es ein Paket in S_x gibt, das ρ_x mit Verspätung ℓ verlässt. Wegen der Aussage des ersten Punktes kann dies für jedes Paket in S_x nur ein einziges Mal passieren. Also muss tatsächlich $\ell_x \leqslant |S_x|$ sein.

   Man betrachte nun eine Kante e_i, die von Paket b_x zu einem Zeitpunkt t benutzt werden möchte aber nicht benutzt werden kann, weil ein anderes Paket b_y sie benutzt. Die Verspätung von b_x erhöht sich also von $\ell = t - i$ auf $\ell + 1$, und das ist auch der einzige Fall, in dem das passieren kann. Dann ist die „Verspätung“ von b_y bei Benutzung dieser Kante $t - i = \ell$.

   Sei nun t' der letzte Zeitpunkt, zu dem ein Paket aus S_x Verspätung ℓ hat. Es sei b dieses Paket und $e_{j'}$ die Kante, die b benutzen „will“; also ist $t' - j' = \ell$.

   Dann gibt es auch ein Paket in S_x, das ρ_x zu diesem Zeitpunkt t' verlässt: Da b Kante $e_{j'}$ benutzen „will“, wird es selbst oder ein anderes Paket diese Kante auch tatsächlich benutzen. Es sei b' dieses Paket. Es hat offensichtlich Verzögerung $t' - j' = \ell$. Würde b' den Pfad ρ_x *nicht* verlassen, dann gäbe es ein Paket, das Kante $e_{j'+1}$ zum Zeitpunkt $t' + 1$ mit Verzögerung $t' + 1 - (j' + 1) = \ell$ benutzen würde. Dies stünde im Widerspruch zur Wahl von t': es sollte der letzte Zeitpunkt sein, zu dem ein Paket Verspätung ℓ hat. Also verlässt b' Pfad ρ_x zum Zeitpunkt t'.

   Wir schreiben daher nun b' zu, bei Paket b_x die Erhöhung der Verspätung von ℓ auf $\ell + 1$ verursacht zu haben. Da b' den Pfad ρ_x verlässt und nie wieder betritt, wird auf diese Weise keinem Paket aus S_x zweimal eine Verspätungserhöhung zugeschrieben. Also ist $\ell_x \leqslant |S_x|$.

■

4.29 Lemma. Es bezeichne H_{xy} die Zufallsvariable mit $H_{xy} = \begin{cases} 1 & \text{falls } \rho_x \cap \rho_y \neq \emptyset \\ 0 & \text{sonst} \end{cases}$. Dann gilt:

1. Die Gesamtverspätung von b_x beim Eintreffen in z_x ist $\ell_x \leqslant \sum_{y \neq x} H_{xy}$.
2. $\mathbf{E}\left[\sum_{y \neq x} H_{xy}\right] \leqslant d/2$.
3. $\mathbf{P}(\ell_x \geqslant 6d) \leqslant \mathbf{P}\left(\sum_{y \neq x} H_{xy} \geqslant 6d\right) \leqslant 2^{-6d}$.

4.30 Beweis.

1. Dies ist eine einfache Umformulierung von Lemma 4.27.
2. Es sei nach wie vor $\rho_x = (e_1, \ldots, e_k)$ mit $k \leqslant d$ irgendein Pfad. Die Zufallsvariable $T(e)$ gebe die Anzahl der Pfade ρ_y mit $y \neq x$ an, die über eine Kante e führen. Dann ist $\sum_{y \neq x} H_{xy} \leqslant \sum_{i=1}^{k} T(e_i)$ und folglich $\mathbf{E}\left[\sum_{y \neq x} H_{xy}\right] \leqslant \sum_{i=1}^{k} \mathbf{E}[T(e_i)]$. Wir zeigen nun noch, dass $\mathbf{E}[T(e_i)] \leqslant 1/2$ ist, so dass die Behauptung aus $k \leqslant d$ folgt.

   Die Kante e_i führe o. B. d. A. von einem Knoten mit Adresse $(x_1 \cdots x_r 0 x_{r+2} \cdots x_d)$ zu Knoten $(x_1 \cdots x_r 1 x_{r+2} \cdots x_d)$. Damit ein Pfad von einem Knoten y nach z_y über e_i führt, muss (da der Bit-Fixing-Algorithmus benutzt wird) y von der Form $y = u_1 \cdots u_r 0 x_{r+2} \cdots x_d$ sein und z_y mit dem Präfix $x_1 \cdots x_r 1$ beginnen. Solche Knoten $y \neq x$ gibt es $2^r - 1$. Da die Zwischenknoten alle zufällig gleichverteilt und unabhängig gewählt werden, ist für jedes y die Wahrscheinlichkeit für das Präfix $x_1 \cdots x_r 1$ in z_y stets $2^{-(r+1)}$.

   Also ist $\mathbf{E}[T(e_i)] = \sum_{y \neq x} \mathbf{P}(\rho_y \text{ benutzt } e_i) = \sum_{u_1 \cdots u_r \neq x_1 \cdots x_r} \mathbf{P}\left(\rho_{u 0 x_{r+2} \cdots x_d} \text{ benutzt } e_i\right) = (2^r - 1) \cdot 2^{-(r+1)} \leqslant 1/2$.
3. Wegen der ersten beiden Punkte ist $\mathbf{E}[\ell_x] \leqslant d/2$. Folglich gilt wegen des zweiten Teils von Korollar 4.23:

$$\begin{aligned} \mathbf{P}(\ell_x \geqslant 6d) \leqslant \mathbf{P}\left(\sum H_{xy} \geqslant 6d\right) &= \mathbf{P}\left(\sum H_{xy} \geqslant 12 \cdot d/2\right) \\ &\leqslant \mathbf{P}\left(\sum H_{xy} \geqslant (1+11)\mathbf{E}\left[\sum H_{xy}\right]\right) \leqslant 2^{-12d/2} = 2^{-6d} \end{aligned}$$

■

4.31 Beweis. (von Satz 4.26)

1. Nach Lemma 4.29 ist die Wahrscheinlichkeit, dass ein Paket um mehr als $6d$ Schritte verzögert wird, höchstens 2^{-6d}.

   Insgesamt werden $N = 2^d$ Pakete unabhängig voneinander transportiert. Die Wahrscheinlichkeit, dass wenigstens eines von ihnen um mehr als $6d$ Schritte verzögert wird, ist folglich höchstens $2^d \cdot 2^{-6d} = 2^{-5d}$.

   Da jedes Paket zusätzlich über maximal d Kanten transportiert werden muss, ist auch die Wahrscheinlichkeit, dass wenigstens ein Paket erst nach mehr als $7d$ Schritten am Ziel ist, höchstens 2^{-5d}. Also sind mit Wahrscheinlichkeit $1 - 2^{-5d}$ alle Pakete nach spätestens $7d$ Schritten am Zwischenknoten.

2. Die zweite Phase von Algorithmus 4.25 kann als Umkehrung der ersten Phase aufgefasst werden. Deshalb gilt getrennt hierfür auch die gleiche Analyse. Damit es bei der Nacheinanderausführung beider Phasen nicht zu Effekten kommt, die bei den obigen Abschätzungen nicht berücksichtigt wurden, erweitert man die erste Phase dahingehend, dass jedes Paket

nach seiner Ankunft im Zwischenknoten verharrt, bis insgesamt seit Beginn des Routing $7d$ Schritte vergangen sind.

Folglich ist die Wahrscheinlichkeit, dass alle Pakete nach $\leqslant 14d$ Schritten am Ziel sind, mindestens $(1-2^{-5d})(1-2^{-5d}) = 1-2/N^5+1/N^{10} \geqslant 1-2/N^5$.

3. Die schlimmste Laufzeit, die überhaupt auftreten kann, ist jedenfalls dadurch nach oben beschränkt, dass die Pakete nacheinander jedes einzeln geroutet werden. Der Zeitbedarf hierfür wäre kleiner gleich $2dN$.

   Für $d \geqslant 3$ ist $N \geqslant 8$, also $14d \leqslant 2dN$. Also gilt dann für den Erwartungswert, dass er kleiner oder gleich $(1-2/N^5)14d + (2/N^5)\cdot 2dN = 14d - 28d/N^5 + 4d/N^4 \leqslant 14d+1$ ist.

■

4.5 Die probabilistische Methode

Die Tragweite der sogenannten *probabilistischen Methode* wurde wohl zuerst von Erdős erkannt, dessen Name seitdem eng mit ihr verbunden ist. Das „Standardbuch“ über die probabilistische Methode trägt ebendiesen Titel und stammt von Alon und Spencer (1992).

Es gibt (unter anderen?) die beiden folgenden Varianten.

Die eine fußt auf der Beobachtung, dass jede Zufallsvariable X mindestens einen Wert annimmt, der nicht kleiner als $\mathbf{E}[X]$ ist; und ebenso einen Wert, der nicht größer als $\mathbf{E}[X]$ ist. Also *existieren* Ereignisse, für die X die jeweiligen Werte annimmt.

Die andere Variante basiert auf der folgenden Tatsache. Wird aus einem Universum von Objekten zufällig eines ausgewählt und ist die Wahrscheinlichkeit dafür, dass es eine bestimmte Eigenschaft hat, echt größer als Null, dann muss im Universum ein Objekt *existieren*, das diese Eigenschaft hat. (Andernfalls muss die Wahrscheinlichkeit, ein solches Objekt auszuwählen ja Null sein.)

Für die erste Variante wird in einem späteren Kapitel angewendet werden. Für die zweite Variante werden wir im folgenden Abschnitt ein Beispiel kennenlernen.

4.6 Zweiter randomisierter Algorithmus

Um im folgenden kompakter formulieren zu können sprechen wir statt von einem randomisierten Algorithmus für Permutationsrouting in Hyperwürfeln kürzer von einem *RPH-Algorithmus*. Ein RPH-Algorithmus sei *schnell*, wenn seine erwartete Laufzeit in $O(d)$ liegt.

Algorithmus 4.25 benötigt $\Theta(Nd)$ Zufallsbits und ist schnell, hat also erwartete Laufzeit von $O(d)$. In Satz 4.7 haben wir gesehen, dass Algorithmen, die 0 Zufallsbits benutzen (also deterministische Algorithmen) im schlimmsten Fall Laufzeit $\Omega(\sqrt{N}/d)$ haben.

Es stellt sich daher die Frage, ob es RPH-Algorithmen gibt, die weniger als $\Theta(Nd)$ Zufallsbits benutzen und trotzdem schnell sind. Ziel dieses Abschnittes ist der Nachweis, dass $\Theta(d)$ Zufallsbits notwendig und hinreichend für RPH-Algorithmen sind, um das Problem schnell zu lösen.

Wir zeigen zunächst die Notwendigkeit. Anschließend benutzen wir die probabilistische Methode, um auch die Existenz eines entsprechenden Algorithmus nachzuweisen.

4.32 Satz. *Wenn ein RPH-Algorithmus in Würfeln mit* $N = 2^d$ *Knoten nur* k *Zufallsbits benutzt, dann ist seine erwartete Laufzeit in* $\Omega(2^{-k}\sqrt{N}/d)$.

4.33 Beweis. Ein entsprechender RPH-Algorithmus R kann als Wahrscheinlichkeitsverteilung über 2^k deterministischen Algorithmen aufgefasst werden. Dann gibt es mindestens einen, der mit Wahrscheinlichkeit 2^{-k} ausgewählt wird. Er werde mit A bezeichnet. Es sei x diejenige Eingabe für A, für die A Laufzeit $\Omega(\sqrt{N}/d)$ hat. Man betrachte die Bearbeitung von x durch R. Mit einer Wahrscheinlichkeit von mindestens 2^{-k} wird R wie A arbeiten. Also ist der Erwartungswert für die Laufzeit mindestens $\Omega(2^{-k}\sqrt{N}/d)$. ∎

4.34 Korollar. Jeder schnelle RPH-Algorithmus muss $\Omega(d)$ Zufallsbits verwenden.

4.35 Beweis. Damit für irgendeine positive Konstante c gilt, dass $2^{-k}\sqrt{N}/d \leqslant cd$ ist, muss $2^k \geqslant \sqrt{N}/(cd^2)$ sein, also $k \geqslant \log\sqrt{N} - O(\log d)$ bzw. $k \in \Omega(d)$. ∎

Nachdem klar ist, dass ein schneller RPH-Algorithmus $\Omega(d)$ Zufallsbits benötigt, wollen wir nun zeigen, dass es (jedenfalls in einem gewissen Sinne) auch tatsächlich einen solchen Algorithmus gibt.

4.36 Satz. *Für jedes* d *gibt es einen schnellen RPH-Algorithmus, der* $3d$ *Zufallsbits benötigt und erwartete Laufzeit* $22d$ *hat.*

Man beachte, dass hier *nicht* die Existenz eines RPH-Algorithmus für Hyperwürfel aller Größen zugesichert wird.

4.37 Beweis. Im folgenden bezeichne $\mathcal{A} = (A_1, \ldots, A_t)$ eine Liste deterministischer PH-Algorithmen. Jedes $\mathcal{A}$ legt einen randomisierten Algorithmus $R_{\mathcal{A}}$ fest, der zufällig gleichverteilt ein $A_i \in \mathcal{A}$ auswählt.

Wir nennen $\mathcal{A}$ ein *effizientes* N-*Schema*, falls für jede zu routende Permutation auf einem Hyperwürfel mit $N = 2^d$ Knoten gilt: Die erwartete Laufzeit von $R_{\mathcal{A}}$, ist höchstens $22d$.

Im folgenden wird gezeigt, dass es für jedes N ein effizientes N-Schema mit $t = N^3$ Algorithmen gibt. Folglich benötigt $R_{\mathcal{A}}$ nur $\log t \in O(\log N) = O(d)$ Zufallsbits (um einen der Algorithmen A_i auszuwählen).

Zum Nachweis der Existenz eines so kleinen effizienten N-Schemas werden wir die probabilistische Methode verwenden. Dazu betrachte man das folgende Zufallsexperiment: Den in Abschnitt 4.4 vorgestellten ersten RPH-Algorithmus kann man als Menge $\mathcal{B} = \{B_1, \ldots, B_{N^N}\}$ von N^N deterministischen PH-Algorithmen auffassen. Hieraus möge zufällig (mit Zurücklegen) eine Liste $\mathcal{A} = (A_1, \ldots, A_{N^3})$ von N^3 Algorithmen $A_i = B_{j_i}$ ausgewählt werden. Wir werden nun zeigen, dass die Wahrscheinlichkeit, dass $\mathcal{A}$ ein effizientes N-Schema ist, echt größer Null ist. Also existiert ein solches.

Es seien die π_i (mit $1 \leqslant i \leqslant N!$) alle $N!$ möglichen zu routenden Permutationen. Ein deterministischer PH-Algorithmus heiße *gut* für ein π_i, wenn diese Permutation in höchstens $14d$ Schritten durchführt wird, und andernfalls *schlecht*. Aus dem zweiten Teil von Satz 4.26 wissen wir, dass für jedes π_i höchstens ein Bruchteil von $1/N$ aller B_j schlecht für π_i ist.

Es sei zunächst ein beliebiges π_i fixiert. Aus dem eben Gesagten folgt, dass der Erwartungswert für die Anzahl der für π_i schlechten Algorithmen in $\mathcal{A}$ höchstens $N^3/N = N^2$ ist. Es bezeichne X_j die 0-1-Zufallsvariable, die genau dann 1 ist, falls A_j schlecht für π_i ist. Dann ist also $\mu = \mathbf{E}\left[\sum_{j=1}^{N^3} X_j\right] \leqslant N^2$. Die X_j sind unabhängige Zufallsvariablen. Wir wollen nun eine obere Schranke

für $\mathbf{P}\left(\sum_{j=1}^{N^3} X_j > 4N^2\right)$ beweisen. Dazu sei $c = N^2/\mu \geqslant 1$. Nach einigen Umformungen[1] kann man z. B. die Chernoff-Schranke in der Form aus Korollar 4.23 mit $\delta = 3$ benutzen:

$$\begin{aligned}
\mathbf{P}\left(\sum_{j=1}^{N^3} X_j > (1+3)N^2\right) &= \mathbf{P}\left(\sum_{j=1}^{N^3} X_j > (1+\delta)c\mu\right) = \mathbf{P}\left(\sum_{j=1}^{N^3} X_j > (c+c\delta)\mu\right) \\
&\leqslant \mathbf{P}\left(\sum_{j=1}^{N^3} X_j > (1+c\delta)\mu\right) \\
&\leqslant \left(\frac{e^{c\delta}}{(1+c\delta)^{1+c\delta}}\right)^{\mu} \leqslant \left(\frac{e^{c\delta}}{(1+c\delta)^{c\delta}}\right)^{\mu} \\
&= \left(\frac{e}{1+c\delta}\right)^{c\delta\mu} \leqslant \left(\frac{e}{1+3}\right)^{c\delta\mu} \\
&= \left(\frac{e}{4}\right)^{3N^2} \leqslant \left(\frac{1}{e}\right)^{N^2}
\end{aligned}$$

Es sei nun E_i das schlechte Ereignis, dass mehr als $4N^2$ Algorithmen in $\mathcal{A}$ schlecht für π_i sind. Dann ist also $\mathbf{P}(E_i) < e^{-N^2}$.

Die Wahrscheinlichkeit, dass $\mathcal{A}$ für mindestens ein π_i schlecht ist, ist dann

$$\mathbf{P}\left(\bigcup_{i=1}^{N!} E_i\right) \leqslant \sum_{i=1}^{N!} \mathbf{P}(E_i) \leqslant N! \cdot e^{-N^2} < 1 \,.$$

Den Nachweis für die letzte Abschätzung werden wir im Anschluss an diesen Beweis skizzieren. Aus der Abschätzung folgt, dass die Wahrscheinlichkeit, dass von den Algorithmen in $\mathcal{A}$ für jede Permutation höchstens $4N^2$ schlecht sind, positiv ist. Also existiert ein solches $\mathcal{A}$.

Es bleibt nun noch zu zeigen, dass dieses $\mathcal{A}$ sogar ein effizientes N-Schema ist: Dazu sei π_i eine beliebige Permutation. Mit Wahrscheinlichkeit $1 - (4N^2/N^3) = 1 - 4/N$ wird $R_{\mathcal{A}}$ diese Permutation in höchstens $14d$ Schritten durchführen. Andernfalls werden höchstens $2dN$ Schritte benötigt. Der Erwartungswert für die Laufzeit ist daher höchstens

$$(1 - \frac{4}{N})14d + \frac{4}{N}2dN \leqslant 22d \,.$$

■

4.38 Es bleibt noch zu skizzieren, dass für hinreichend große N gilt: $N! \cdot e^{-N^2} < 1$.

Hierzu kann man sich der Stirlingschen Formel bedienen. Sie besagt:

$$N! = \sqrt{2\pi N}\frac{N^N}{e^N}(1 + h(N)) \text{ mit } h(N) = \frac{1}{12N} + \frac{1}{288N^2} - \frac{139}{5140N^3} \pm \cdots \in O(\frac{1}{N})$$

Mit anderen Worten ist

$$\lim_{N\to\infty} \frac{N! \cdot e^N}{\sqrt{2\pi N} \cdot N^N} = 1$$

woraus sofort die Behauptung folgt. Genaueres Nachrechnen zeigt, dass schon für $N \geqslant 4$ gilt: $N! \cdot e^{-N^2} < 1$.

[1] Dank an D. Hoske und S. Walzer für die Korrektur

4.39 Satz 4.36 behauptet nur die *Existenz* eines – noch dazu nichtuniformen – schnellen RPH-Algorithmus, der nur $O(d)$ Zufallsbits braucht. Sehr viel schwieriger scheint es zu sein, explizit RPH-Algorithmen anzugeben, die möglichst wenige Zufallsbits benötigen. Der beste bislang bekannte immerhin noch $\Theta(d^2)$ Zufallsbits, ist dafür allerdings uniform. Es ist eine nach wie vor ungelöste Aufgabe, diesen Wert zu senken.

Zusammenfassung

1. Beim Routing-Problem in Hyperwürfeln gibt es für jeden deterministischen Algorithmus Permutationen mit *sehr* langen Laufzeiten. Die kann man mit randomisierten Algorithmen im Erwartungswert vermeiden.
2. Mit Hilfe der probabilistischen Methode kann man zeigen, dass ein nichtuniformer RPH-Algorithmus existiert, der größenordnungsmäßig minimale Anzahl von Zufallbits benötigt, aber trotzdem kleine erwartete Laufzeit hat.

Literatur

Alon, Noga und J. Spencer (1992). *The Probabilistic Method*. Wiley Interscience (siehe S. 38).

Borodin, A. und J. E. Hopcroft (1985). „Routing, Merging, and Sorting on Parallel Models of Computation“. In: *Journal of Computer and System Sciences* 30, S. 130–145 (siehe S. 29).

Chernoff, Herman (1952). „A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations“. In: *Annals of Mathematical Statistics* 23, S. 493–507 (siehe S. 31).

Hagerup, T. und C. Rüb (1990). „A Guided Tour of Chernoff Bounds“. In: *Information Processing Letters* 33, S. 305–308 (siehe S. 31).

Kaklamanis, C., D. Krizanc und T. Tsantilas (1990). „Tight Bounds for Oblivious Routing in the Hypercube“. In: *Proceedings of the 2nd Annual ACM Symposium on Parallel Algorithms and Architectures*. S. 31–36 (siehe S. 29).

Leighton, Frank Thomson (1992). *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*. San Mateo, CA 94403: Morgan Kaufmann Publ. ISBN: 1-55860-117-1 (siehe S. 29).

