Parametrisierung von Pollards Rho-Methode

Finn Rudolph
22.04.2024

Erarbeitungsort: Hauptstraße 28, 96178 Pommersfelden
Fachgebiet: Mathematik / Informatik
Wettbewerbssparte: Jugend forscht
Bundesland: Bayern
Wettbewerbsjahr: 2024

Zusammenfassung

Pollards Rho-Methode ist einer der schnellsten Algorithmen, um kleine Primfaktoren einer Zahl zu finden. In dieser Arbeit soll untersucht werden, wie der Parameter k der Funktion $x \mapsto x^{2k} + 1$ in der Rho-Methode bestmöglich gewählt werden kann, um eine geringe Laufzeit zu erzielen. k hat unter Umständen großen Einfluss auf die Laufzeit — sowohl im positiven als auch im negativen Sinn. Da es kein wirklich effizientes Verfahren gibt, die Rho-Methode auf mehreren Maschinen parallel auszuführen, ist insbesondere von Interesse, wie k-Werte bei mehreren Maschinen optimal gewählt werden. Die Wahl von k geschieht nämlich für jede Maschine separat. Um die Frage anzugehen, habe ich eine Formel für die Laufzeit in Abhängigkeit von k hergeleitet. Mit dieser war es möglich zu zeigen, dass $k = 1$ im Fall einer Maschine optimal ist. Es wurde ebenfalls nachgewiesen, dass bei zwei Maschinen die Wahl von $k = 1$ für beide günstiger ist als die Wahl von zwei Primzahlen oder zwei gleichen Zahlen größer als 1. Diese Ergebnisse und die Korrektheit der Formel wurden von Laufzeitmessungen bestätigt. Mithilfe der Formel kann die Laufzeit des parallelen Rho-Algorithmus in Abhängigkeit von k nun besser verstanden werden.

Inhaltsverzeichnis

1 Motivation und Hintergrund

Pollards Rho-Methode bleibt trotz der Existenz asymptotisch schnellerer Algorithmen einer der meist verwendeten Algorithmen, um kleine Primfaktoren in Zahlen zu erkennen. Gleichzeitig werden Leistungssteigerungen bei modernen Computern häufig durch größere Nebenläufigkeit (z. B. mehr Prozessorkerne) erzielt. Um die Rho-Methode schnellstmöglich zu implementieren, ist es also nötig zu untersuchen, wie sie am besten parallel ausgeführt werden kann. Die Wahl von k geschieht für jede Maschine separat und hat unter Umständen großen Einfluss auf die Laufzeit, weshalb sie ein wichtiger Ansatzpunkt zur Weiterentwicklung der parallelen Rho-Methode ist.

Der Parameter k bestimmt in der Rho-Methode den Exponenten der Funktion $f : x \mapsto x^{2k}+1$. Diese Funktion wird wiederholt auf einen zufällig gewählten Anfangswert x_0 angewandt. Die Rho-Methode findet einen Faktor, wenn $f^{(m)}(x_0) = f^{(2m)}(x_0)$, wobei $f^{(m)}$ die m-malige Verkettung von f ist.

k zu variieren, wurde bereits in der ersten Veröffentlichung der Rho-Methode (Pollard, 1975) in Betracht gezogen. In Brent und Pollard, 1981 gelang durch eine Veränderung von k (allerdings nur auf einer Maschine) die erstmalige Faktorisierung der achten Fermatzahl $F_8 = 2^{2^8} + 1$. Wenn Kongruenzen der Primfaktoren bekannt sind, wie im Fall von Fermatzahlen, kann k gezielt gewählt werden, um die Laufzeit zu verringern. In dieser Arbeit wird allerdings der Fall allgemeiner Zahlen betrachtet, das heißt, keine Kongruenzen der Primfaktoren sind bekannt.

Bei einer direkten Parallelisierung der Rho-Methode auf M Maschinen ergibt sich eine Verringerung der Laufzeit um einen Faktor $\sqrt{M}$. In Crandall, 1999 wurde ein Verfahren vorgestellt, mit dem sich eine theoretische Verringerung um einen Faktor $(\log^2 M)/M$ erzielen lässt. Eine Methode, um bei M Maschinen eine Laufzeitreduktion um einen Faktor M zu erhalten, ist nicht bekannt (im Gegensatz zu dem verwandten Problem der Bestimmung diskreter Logarithmen, siehe van Oorschot und Wiener, 1999).

In dieser Arbeit soll es also um folgende Frage gehen: *Gegeben M Maschinen und eine Zahl, über deren Primfaktoren nichts besonderes bekannt ist. Wie wählt man den Parameter k für jede Maschine optimal, um eine möglichst geringe Laufzeit zu erzielen?* Wo genau die Problematik und Komplexität dieser Frage liegen, kann erst deutlich werden, nachdem die Rho-Methode und ihre parallelisierte Variante vorgestellt wurden.

2 Erläuterung der Rho-Methode

Sei n die zu faktorisierende Zahl. Es wird angenommen, dass n ungerade und keine Potenz einer natürlichen Zahl ist, da sonst einfach ein Faktor gefunden werden kann. Sei $g : \mathbb{Z}/n\mathbb{Z} \to \mathbb{Z}/n\mathbb{Z}$ mit $g : x \mapsto x^{2k} + 1$ für einen Parameter $1 \leq k \in \mathbb{N}$. Man wähle einen zufälligen Anfangswert $x_0 \in \mathbb{Z}/n\mathbb{Z}$ und betrachte die Folge $(x_i)_{i\in\mathbb{N}}$, definiert durch $x_i = g(x_{i-1})$. Da $(x_i)_{i\in\mathbb{N}}$ über der endlichen Menge $\mathbb{Z}/n\mathbb{Z}$ definiert ist, ist die Folge ab einem bestimmten Punkt periodisch. Sei p ein Primfaktor von n und $\pi : \mathbb{Z}/n\mathbb{Z} \to \mathbb{Z}/p\mathbb{Z}$ die natürliche Projektion. Die Idee der Rho-Methode ist, zwei Folgenglieder $x_i, x_j \in \mathbb{Z}/n\mathbb{Z}$ zu finden, sodass $x_i \neq x_j$ aber $\pi(x_i) = \pi(x_j)$. Dann ist nämlich $\gcd(n, x_i - x_j)$ ein echter Faktor von n. Nimmt man an, dass die Periodenlänge von $(x_i)_{i\in\mathbb{N}}$ in $\mathbb{Z}/n\mathbb{Z}$ deutlich länger als die Periodenlänge in $\mathbb{Z}/p\mathbb{Z}$ ist, reicht es aus, x_i, x_j mit $i \neq j$ zu finden, die kongruent modulo p sind. Die Annahme ist plausibel, weil für den kleinsten Primfaktor $p \leq \sqrt{n}$ gilt, es also deutlich weniger mögliche Werte für $\pi(x_i)$ als x_i gibt. Im

Folgenden ist p immer der kleinste Primfaktor von n. Das Ereignis, dass eine Folge einen Wert zweimal annimmt, wird eine *Kollision* in dieser Folge genannt. Es wird außerdem angenommen, dass die anderen Primfaktoren von n so viel größer als p sind, dass die Wahrscheinlichkeit einer Kollision modulo eines anderen Primfaktors vernachlässigbar gering ist.

2.1 Betrachtung der Folge als ein Weg in einem funktionalen Graphen. Zum Finden oben genannter x_i, x_j ist es hilfreich, den funktionalen Graphen von g zu betrachten.

Definition 1 (Funktionaler Graph)**.** Sei X eine endliche Menge und $f : X \to X$ eine Abbildung. Der funktionale Graph von f, geschrieben $\gamma(f)$, ist der gerichtete Graph mit Knotenmenge X und Kantenmenge E, wobei die Kante $(x, y) \in X \times X$ genau dann in E liegt, wenn $f(x) = y$.

Ein Beispiel eines funktionalen Graphen ist in Abbildung 1 zu sehen. Es ist leicht zu zeigen, dass jede Zusammenhangskomponente eines funktionalen Graphen aus einem Zyklus und an den Zyklusknoten gewurzelten Bäumen besteht. In $\gamma(g)$ ist x_0 also ein Knoten in einem der Bäume, der durch seinen Pfad zur Wurzel mit dem Zyklus seiner Zusammenhangskomponente verbunden ist. Die Folge $(x_i)_{i\in\mathbb{N}}$ startet bei x_0 und „läuft“ durch den Graphen, wobei immer die eindeutige von einem Knoten ausgehende Kante entlanggegangen wird. An der Wurzel des Baums von x_0 wird der Zyklus in der Zusammenhangskomponente von x_0 betreten, und ab genau diesem Punkt ist $(x_i)_{i\in\mathbb{N}}$ periodisch. Bei Auftritt der ersten Kollision bildet der von $(x_i)_{i\in\mathbb{N}}$ abgelaufene Pfad die Form eines „ρ“ in $\gamma(g)$, was dem Algorithmus seinen Namen gibt.

Folgende Begriffe werden für weitere Erklärungen und die Analyse des Algorithmus nötig sein:

Definition 2. Sei $f : X \to X$ eine Abbildung über einer endlichen Menge X und $x \in X$. Wir nennen

1. $\mu(f, x)$ die Höhe von x in seinem Baum in $\gamma(f)$.
2. $\lambda(f, x)$ die Länge des Zyklus in der Zusammenhangskomponente von x in $\gamma(f)$.
3. $\rho(f, x) = \mu(f, x) + \lambda(f, x)$ die Rho-Länge von x.

Als Beispiel betrachte man f aus Abbildung 1. Für $x = 13$ gilt $\mu(f, 13) = 3, \lambda(f, 13) = 6$ und $\rho(f, 13) = 9$. Für $x = 15$ gilt $\mu(f, 15) = 1, \lambda(f, 15) = 6$ und $\rho(f, 15) = 7$. Hier tritt also nach siebenmaliger Anwendung von f mit Anfangswert 15 erstmals eine Kollision auf.

2.2 Kollisionserkennung im funktionalen Graphen modulo *p*. Sei f die Abbildung g, betrachtet in $\mathbb{Z}/p\mathbb{Z}$ und $x'_i = \pi(x_i)$ für alle $i \in \mathbb{N}$. Die Folge $(x'_i)_{i\in\mathbb{N}}$ „läuft“ analog durch $\gamma(f)$, beginnend von x'_0. Unser Ziel, das Finden einer Kollision von $(x'_i)_{i\in\mathbb{N}}$, ist also äquivalent dazu, einen Knoten in $\gamma(f)$ zu finden, der zweimal in dem von $(x'_i)_{i\in\mathbb{N}}$ abgelaufenen Pfad erscheint. Dafür kann Floyds Algorithmus zur Zykluserkennung in $\gamma(f)$ verwendet werden. Floyds Algorithmus macht sich zunutze, dass es ein $1 \leq r \in \mathbb{N}$ mit $x'_r = x'_{2r}$ geben muss (Knuth, 1998, S. 7). Für das minimale solcher r gilt $r \leq \rho(f, x'_0)$ (Knuth, 1998, S. 7). Als Beispiel betrachte man wieder f aus Abbildung 1 und $x_0 = 15$. Das minimale r mit $x_r = x_{2r}$ ist $r = 6$, und es gilt $x_r = 2$.

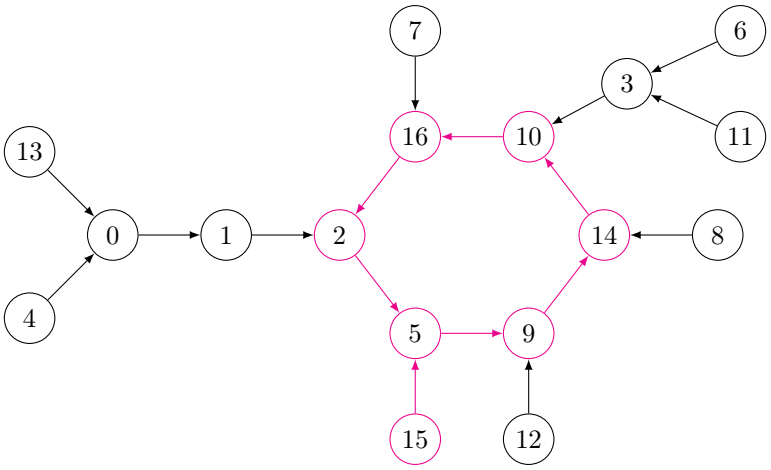

Abbildung 1: Der funktionale Graph von $f : \mathbb{Z}/17\mathbb{Z} \to \mathbb{Z}/17\mathbb{Z}$ mit $f : x \mapsto x^2 + 1$. Der Weg, den die Folge $x_0, f(x_0), f(f(x_0)), f(f(f(x_0))), \ldots$ mit $x_0 = 15$ beschreibt, ist farbig markiert. Konkret lautet die Folge $15, 5, 9, 14, 10, 16, 2, 5, 9, 14, 10, 16, 2, \ldots$

Es also genügt also, die Folgen $(x'_i)_{i\in\mathbb{N}}$ und $(x'_{2i})_{n\in\mathbb{N}}$ gleichzeitig Glied für Glied zu berechnen und in jedem Schritt zu überprüfen, ob $x'_i = x'_{2i}$. Das kann natürlich nicht explizit geschehen, da p unbekannt ist. Stattdessen werden $(x_i)_{i\in\mathbb{N}}$ und $(x_{2i})_{i\in\mathbb{N}}$ Glied für Glied berechnet. Das Überprüfen, ob $x'_i = x'_{2i}$ geschieht durch Berechnung von $\gcd(n, x_i - x_{2i})$. So erhält man nach maximal $\rho(f, x'_0)$ Schritten die gewünschte Kollision. Die Rho-Methode ist in Algorithmus 1 dargestellt. Nach der i-ten Iteration speichert x das i-te Glied der Folge $(x_i)_{i\in\mathbb{N}}$ und y das i-te Glied von $(x_{2i})_{i\in\mathbb{N}}$.

Algorithmus 1 : Pollards Rho-Methode

$x \leftarrow$ zufällige natürliche Zahl zwischen 0 und $n - 1$
$y \leftarrow x$
while TRUE **do**
  $x \leftarrow x^{2k} + 1 \mod n$
  $y \leftarrow (y^{2k} + 1)^{2k} + 1 \mod n$
  $g \leftarrow \gcd(n, x - y)$
  if $g \neq 1$ **and** $g \neq n$ **then**
    return g
  end
end

2.3 Analyse der Rho-Methode. Die Analyse der Rho-Methode erweist sich als schwierig, es ist keine rigorose Laufzeitanalyse bekannt. Unter heuristischen Annahmen lässt sich die Laufzeit allerdings gut abschätzen. Als erste Vereinfachung wird statt der mittleren Anzahl an Iterationen von Floyds Algorithmus die mittlere Rho-Länge analysiert. Eine weitere Annahme dreht sich um die Verteilung der Rho-Längen in $\gamma(f)$, für deren Formulierung der Begriff einer asymptotischen Näherung benötigt wird. Asymptotische Näherungen werden häufig bei der Analyse von Algorithmen oder in der Analysis eingesetzt („Asymptotic analysis — Wikipedia“, 2024).

Definition 3 (Asymptotische Näherung)**.** Eine Funktion $f : \mathbb{R} \to \mathbb{R}$ heißt genau dann asympto-

tische Näherung von einer Funktion $g : \mathbb{R} \to \mathbb{R}$, oder asymptotisch zu g, wenn

$$\lim_{x \to \infty} \frac{f(x)}{g(x)} = 1$$

In diesem Fall schreiben wir $f \sim g$.

Sei $A(n)$ die Menge der Abbildungen $\mathbb{Z}/n\mathbb{Z} \to \mathbb{Z}/n\mathbb{Z}$ für $n \in \mathbb{N}$. Über die Verteilung der Rho-Längen wird folgende Annahme getroffen, die auch *Random Mapping Assumption* (RMA) genannt wird (Brent und Pollard, 1981).

Annahme (Random Mapping Assumption)**.** *Sei $f : \mathbb{Z}/p\mathbb{Z} \to \mathbb{Z}/p\mathbb{Z}$ mit $f : x \mapsto x^{2k} + 1$ und $d = \gcd(p - 1, 2k)$. Seien $x_0 \in \mathbb{Z}/p\mathbb{Z}$ und $y_0 \in \mathbb{Z}/((p-1)/d)\mathbb{Z}$ zufällig und $g \in A((p-1)/d)$ zufällig. Dann gilt $\mathbb{P}(\rho(f, x_0) = m) \sim \mathbb{P}(\rho(g, y_0) = m)$ für $p \to \infty$.*

In anderen Worten sagt die Random Mapping Assumption, dass sich die Verteilung der Rho-Längen von $f : x \mapsto x^{2k} + 1$ wie bei einer zufälligen Funktion aus $A((p-1)/d)$ verhält. Insbesondere verhält sich $x \mapsto x^2 + 1$ bezüglich der Rho-Längen wie ein zufällige Funktion $\mathbb{Z}/(p-1)\mathbb{Z} \to \mathbb{Z}/(p-1)\mathbb{Z}$. Brent und Pollard, 1981 geben eine Begründung für RMA. Im Folgenden wird statt $(p-1)/d$ einfach p/d verwendet, da $p \sim p - 1$ für $p \to \infty$.

Für $k = 1$ ist unter RMA die erwartete Anzahl an Iterationen der while-Schleife asymptotisch zu $\sqrt{\pi p/2}$ (Knuth, 1998, S. 8). Da die Berechnung des größten gemeinsamen Teilers $O(\ln n)$ Schritte benötigt, ist die erwartete Laufzeit des Algorithmus $O(\sqrt{p} \ln n)$. Durch eine einfache Modifikation kann die Dauer des gcd amortisiert werden, sodass sich die Laufzeit auf $O(\sqrt{p})$ verringert (Brent, 1980). Damit ist pro Iteration also nur noch die Zeit zur Berechnung der $2k$-ten Potenzen von x und y relevant, was durchschnittlich in $c \lg 2k$ Schritten möglich ist, wobei c eine für den Vergleich verschiedener Werte von k unwichtige Konstante ist. Mit $\lg x$ wird der Logarithmus zur Basis 2 bezeichnet.

3 Parallelisierung der Rho-Methode

Während die vorangehende Analyse bekannt ist, wurde die Analyse bei einer beliebigen Anzahl an Maschinen und beliebigen Werten von k selbst entwickelt. Diese wird in den Abschnitten 3 und 4 vorgestellt.

Sei M die Anzahl verfügbarer Maschinen. Eine *Maschine* meint hier nicht zwingend einen Computer, sondern eine Ressource, auf der ein sequentielles Programm ausgeführt werden kann, was beispielsweise auch ein Prozessorthread sein kann. Die Rho-Methode lässt sich parallelisieren, indem M Anfangswerte (x_0 in Abschnitt 2) zufällig und unabhängig voneinander gewählt werden und der Algorithmus auf jeder der M Maschinen ausgeführt wird, bis eine der Maschinen einen Faktor findet. Die Frage nach der optimalen Wahl von k für diese M Maschinen ist damit nicht klar: Durch ein größeres k ist möglicherweise $\gcd(p - 1, 2k)$ groß, sodass die Zahl an Iterationen um einen Faktor $\sqrt{\gcd(p - 1, 2k) - 1}$ sinkt. Allerdings steigt die Dauer einer Iteration um einen Faktor $\lg 2k$ wegen der Berechnung von x^{2k}. Da es sich bei den Veränderungen in der Laufzeit durch Veränderung von k um konstante Faktoren handelt, wird für den Vergleich der Laufzeit nicht die O-Notation verwendet, sondern eine asymptotische Näherung für die erwartete Zahl an Zeiteinheiten bestimmt, wenn $p \to \infty$. Wir definieren eine *Zeiteinheit* als die Dauer einer Iteration für $k = 1$, bei einer Maschine mit Parameter k dauert eine Iteration also $\lg 2k$ Zeiteinheiten. Im

Gegensatz zur O-Notation kann zwischen zwei Funktionen, die asymptotisch zueinander sind, für große n kein konstanter Faktor liegen, sodass sich Veränderungen um konstante Faktoren sinnvoll vergleichen lassen.

Sei $k_1, \ldots, k_M, 1 \leq k_i \in \mathbb{N}$ eine Zuordnung von k-Werten für M Maschinen. Mit $L_{k_1,\ldots,k_M}(p)$ (oder kurz $L_{(k_i)}(p)$) wird die erwartete Laufzeit des parallelen Rho-Algorithmus mit entsprechenden k-Werten bezeichnet. Sei $h_i = p/(\gcd(p-1, 2k_i)-1)$ und Z_i die gleichverteilte Zufallsvariable über $A(h_i) \times \mathbb{Z}/h_i\mathbb{Z}$ mit $Z_i(f, x_0) = \rho(f, x_0)$ für $f \in A(h_i), x_0 \in \mathbb{Z}/h_i\mathbb{Z}$. Unter RMA gilt

$$L_{(k_i)}(p) = \mathbb{E}\left(\min_{i=1}^{M} Z_i \lg 2k_i \right) \tag{1}$$

In $L_{(k_i)}(p)$ ist $\gcd(p-1, 2k_i)$ für alle $1 \leq i \leq M$ noch fixiert, der Erwartungswert über alle Möglichkeiten von $\gcd(p-1, 2k_i)$ wird erst in Abschnitt 5 behandelt.

Wenn $k_i = k_j$, sind Z_i und Z_j nicht stochastisch unabhängig, weil sich die i-te und j-te Maschine dann im gleichen funktionalen Graphen bewegen. Es kann beispielsweise sein, dass die i-te Maschine der j-ten „hinterherläuft“, wenn der Anfangswert der i-ten in einem Teilbaum des Anfangswerts der j-ten liegt. Diese Abhängigkeit sorgt für eine höhere Laufzeit und erschwert die Analyse des Algorithmus. Das Problem kann umgangen werden, indem die Funktion $f(x) = x^{2k_i} + c_i$ statt $f(x) = x^{2k_i} + 1$ für Maschine i verwendet wird (Crandall, 1999, S. 6). Dabei wird $c_i \in \mathbb{Z}/n\mathbb{Z}$ für jede Maschine unabhängig gewählt, sodass Z_i und Z_j $(i \neq j)$ immer stochastisch unabhängig sind, auch wenn $k_i = k_j$. Für folgende Berechnungen wird daher immer angenommen, dass Z_i und Z_j für $i \neq j$ unabhängig sind.

Anmerkung. In der Version dieser Arbeit zum Regionalwettbewerb war mir das Paper von Crandall, 1999 noch nicht bekannt. Daher war mir nicht klar, dass sich die Abhängigkeit von Maschinen mit gleichem k durch unabhängige Wahl von c_i umgehen lässt, sondern es wurde nur $c_i = 1$ für alle i in Betracht gezogen. Folglich wurde der Fall unabhängiger Maschinen ($\Longleftrightarrow$ paarweise verschiedene k_i) vom Fall abhängiger Maschinen unterschieden. Für den Fall unabhängiger Maschinen wurde die Formel aus Abschnitt 4 hergeleitet, die durch zufällige Wahl von c_i nun allgemein gilt. Im Fall abhängiger Maschinen konnte eine Formel für $M = 2$ bestimmt werden. Sie ist hier für weitere Berechnungen nicht mehr relevant. Weil der zentrale Satz in der Herleitung aber auch unabhängig von der Anwendung auf die Rho-Methode interessant ist, soll dieser vorgestellt werden. Im Fall zweier abhängiger Maschinen bewegen sich beide Maschinen im gleichen funktionalen Graphen, beginnend von unabhängig gewählten Anfangsknoten. Die erwartete Anzahl an Iterationen ist also der Erwartungswert des Minimums der beiden Rho-Längen der Anfangsknoten. Diese Zahl wird unter RMA von Satz 1 beschrieben. Der Vorfaktor in der Definition von τ_n ist $1/n^{n+2}$, da es n^n Abbildungen $\mathbb{Z}/n\mathbb{Z} \to \mathbb{Z}/n\mathbb{Z}$ gibt und für jede von diesen n^2 Paare an Anfangswerten.

Satz 1. *Sei $n \in \mathbb{N}$ und $A(n)$ die Menge der Abbildungen $\mathbb{Z}/n\mathbb{Z} \to \mathbb{Z}/n\mathbb{Z}$. Wir bezeichnen mit*

$$\tau_n = \frac{1}{n^{n+2}} \sum_{g \in A(n)} \sum_{a \in \mathbb{Z}/n\mathbb{Z}} \sum_{b \in \mathbb{Z}/n\mathbb{Z}} \min\{\rho(g, a), \rho(g, b)\}$$

die erwartete minimale Rho-Länge zweier zufälliger Knoten in einem zufälligen funktionalen

Graphen von Größe n. Es gilt

$$\tau_n \sim \frac{25}{32}\sqrt{\pi n/2}$$

In dem Beweis wurde eine erzeugende Funktion für τ_n bestimmt, für deren Koeffizienten mithilfe analytischer Methoden aus Flajolet und Odlyzko, 1990 eine asymptotische Näherung bestimmt werden konnte. *Ende der Anmerkung.*

4 Herleitung einer Formel für die erwartete Laufzeit

In diesem Abschnitt wird eine Formel für $L_{(k_i)}(p)$ hergeleitet. Sei $h_i = p/(\gcd(p-1, 2k_i) - 1)$. Mit (1) und der Definition des Erwartungswerts gilt

$$L_{(k_i)}(p) = \sum_{z_1=1}^{h_1} \mathbb{P}(Z_1 = z_1) \sum_{z_2=1}^{h_2} \mathbb{P}(Z_2 = z_2) \cdots \sum_{z_M=1}^{h_M} \mathbb{P}(Z_M = z_M) \min_{i=1}^{M}(z_i \lg 2k_i) \tag{2}$$

wobei durch „$\cdots$“ M ineinander verschachtelte Summen über alle möglichen z_i für jedes $1 \leq i \leq M$ angedeutet werden. Zunächst soll $\mathbb{P}(Z_i = z_i)$ bestimmt werden.

Lemma 1. *Man betrachte eine Maschine mit Parameter k und $h = p/(\gcd(p-1, 2k) - 1)$. Sei Z die gleichverteilte Zufallsvariable über $A(h) \times \mathbb{Z}/h\mathbb{Z}$ mit $Z(f, x_0) = \rho(f, x_0)$. Es gilt*

$$\mathbb{P}(Z = z) = \frac{z}{h} \prod_{j=1}^{z-1} \left(1 - \frac{j}{h}\right)$$

Beweis. Für eine zufällige Funktion $f \in A(h)$ ist die Wahrscheinlichkeit einer Kollision im j-ten Schritt j/h, wenn in den ersten $j-1$ Schritten keine Kollision aufgetreten ist, da jeder der h möglichen Werte gleich wahrscheinlich ist und j von ihnen zu einer Kollision führen. Das Produkt ist also die Wahrscheinlichkeit, dass in den ersten $z-1$ Schritten keine Kollision auftritt, und der Faktor z/h ist die Wahrscheinlichkeit, dass im z-ten Schritt eine Kollision auftritt. □

Sei im Folgenden $Q(z, h) = (z/h)e^{-z^2/(2h)}$ und

$$F = \sum_{z_1=1}^{h_1} Q(z_1, h_1) \sum_{z_2=1}^{h_2} Q(z_2, h_2) \cdots \sum_{z_M=1}^{h_M} Q(z_M, h_M) \min_{i=1}^{M}(z_i \lg 2k_i) \tag{3}$$

Indem man die Näherung $1 + x \approx e^x$ für kleine x verwendet, sieht man, dass $\mathbb{P}(Z_i = z) \approx (z/h_i) \prod_{j=1}^{z-1} e^{-j/h_i} \approx Q(z, h_i)$. Damit gilt auch $F \approx L_{(k_i)}(p)$. „$\approx$“ hat im Gegensatz zu „$\sim$“ keine formale Bedeutung, sondern dient lediglich der Motivation der folgenden Schritte. Das Ziel ist nun, eine einfache asymptotische Näherung für F zu finden und zu zeigen, dass $L_{(k_i)}(p) \sim F$.

Lemma 2. *Sei $Q(z, h) = (z/h)e^{-z^2/(2h)}$ und $\delta > 0$. Dann gilt für $h \to \infty$*

$$\sum_{z=h^{1/2+\delta}}^{h} Q(z, h) \leq he^{-h^{2\delta}/2}$$

Beweis. Da $e^{-z^2/(2h)}$ für $z \geq 0$ monoton fällt, gilt

$$\sum_{z=h^{1/2+\delta}}^{h} \frac{z}{h} e^{-z^2/(2h)} \leq \sum_{z=h^{1/2+\delta}}^{h} e^{-h^{2\delta}/2} \leq h e^{-h^{2\delta}/2} \qquad \square$$

Lemma 3. *Sei $0 \leq \delta < 1/6$. Mit der Notation von Lemma 1 gilt für $p \to \infty$ und $z \leq h^{1/2+\delta}$*

$$|Q(z,h) - \mathbb{P}(Z = z)| = O\left(\frac{1}{h^{1-4\delta}}\right)$$

Anmerkung. z wird hier als Funktion von h verstanden. Denn um Lemma 3 später auf (2) für $p \to \infty$ anzuwenden, reicht eine Abschätzung für konstante z_i nicht aus, da in (2) über alle $1 \leq z_i \leq h_i$ summiert wird und $h_i = p/(\gcd(p-1, 2k_i) - 1)$.

Aus Lemma 3 folgt $\lim_{p\to\infty} \mathbb{P}(Z = z) = Q(z,h)$ für $z \leq h^{1/2+\delta}$. Mit einer einfachen Rechnung lässt sich das auch für $z > h^{1/2+\delta}$ zeigen. Daraus folgt aber noch nicht $L_{(k_i)}(p) \sim F$, da die Anzahl an Summanden in (2) von p abhängt. Das δ in Lemma 3 wird später hilfreich sein, um zu zeigen, dass wirklich die gesamte Summe in (2) asymptotisch zu F ist.

Beweis von Lemma 3. Durch Anwenden der Restgliedabschätzung $e^x = 1 + x + O(x^2)$ für $|x| \leq 1$ auf $\mathbb{P}(Z = z)$ erhalten wir

$$\mathbb{P}(Z = z) = \frac{z}{h} \prod_{j=1}^{z-1} \left(e^{-j/h} - O\left(\frac{j^2}{h^2}\right)\right)$$

Daraus folgt wegen $Q(z,h) = (z/h)e^{-z^2/(2h)}$

$$\begin{aligned}
|Q(z,h) - \mathbb{P}(Z = z)| &= \frac{z}{h}\left| e^{-z^2/(2h)} - \prod_{j=1}^{z-1} \left(e^{-j/h} - O\left(\frac{j^2}{h^2}\right)\right)\right| \\
&\leq \frac{z}{h}\left| e^{-z^2/(2h)} - \prod_{j=1}^{z-1} e^{-j/h}\right| \\
&\quad + \frac{z}{h}\left| \sum_{k=1}^{z-1} O\left(\frac{k^2}{h^2}\right) \prod_{j=1, j\neq k}^{z-1} e^{-j/h} - \sum_{k=1}^{z-1}\sum_{l=k+1}^{z-1} O\left(\frac{k^2 l^2}{h^4}\right) \prod_{j=1, j\neq k,l}^{z-1} e^{-j/h} + \cdots \right|
\end{aligned}$$

Die Terme des ausmultiplizierten Produkts wurden nach der Zahl an $O(j^2/h^2)$-Faktoren gruppiert. Außerdem wurde die Dreiecksungleichung angewandt. Für den ersten Summanden gilt

$$\begin{aligned}
\frac{z}{h}\left| e^{-z^2/(2h)} - \prod_{j=1}^{z-1} e^{-j/h}\right| &= \frac{z}{h}\left| e^{-z^2/(2h)} - e^{-\sum_{j=1}^{z-1} j/h}\right| \\
&= \frac{z}{h}\left| e^{-z^2/(2h)} - e^{-z(z-1)/(2h)}\right| \\
&= \frac{z}{h} e^{-z^2/(2h)}\left| 1 - e^{-z/(2h)}\right| \\
&\leq \frac{z^2}{h^2} \\
&= O\left(\frac{1}{h^{1-2\delta}}\right)
\end{aligned}$$

Von der dritten zur vierten Zeile wurde die Restgliedabschätzung in der Form $|1-e^x| \le 2|x|$ für $|x| \le 1$ verwendet. Für den zweiten Summanden gilt

$$\begin{aligned}
&\frac{z}{h}\left|\sum_{k=1}^{z-1} O\Big(\frac{k^2}{h^2}\Big) \prod_{j=1,j\neq k}^{z-1} e^{-j/h} - \sum_{k=1}^{z-1}\sum_{l=k+1}^{z-1} O\Big(\frac{k^2l^2}{h^4}\Big) \prod_{j=1,j\neq k,l}^{z-1} e^{-j/h} + \cdots\right| \\
&\le \frac{z}{h}\left(\sum_{k=1}^{z-1} O\Big(\frac{k^2}{h^2}\Big) + \sum_{k=1}^{z-1}\sum_{l=k+1}^{z-1} O\Big(\frac{k^2l^2}{h^4}\Big) + \sum_{k=1}^{z-1}\sum_{l=k+1}^{z-1}\sum_{m=l+1}^{z-1} O\Big(\frac{k^2l^2m^2}{h^6}\Big) + \cdots\right) \\
&\le \frac{z}{h}\left(z\,O\Big(\frac{z^2}{h^2}\Big) + z^2\,O\Big(\frac{z^4}{h^4}\Big) + z^3\,O\Big(\frac{z^6}{h^6}\Big) + \cdots\right) \\
&= O\left(\frac{1}{h^{1/2-\delta}}\left(\frac{1}{h^{1/2-3\delta}} + \frac{1}{h^{1-6\delta}} + \frac{1}{h^{3/2-9\delta}} + \cdots\right)\right) \\
&= O\left(\frac{1}{h^{1-4\delta}}\right)
\end{aligned}$$

Von der ersten zur zweiten Zeile wurden mit der Dreiecksungleichung alle negativen Vorzeichen entfernt. Anschließend wurden die Produkte von $e^{-j/h}$ weggelassen, da sie ≤ 1 sind. In der letzten Zeile wurde die geometrische Summenformel verwendet. □

Lemma 4. *Sei F wie in (3) definiert. Es gilt*

$$F \sim \sqrt{\pi p/2}\left(\sum_{i=1}^{M} \frac{\gcd(p-1, 2k_i) - 1}{\lg^2 2k_i}\right)^{-1/2}$$

Beweis. Die Summen in (3) werden durch Integrale angenähert.

$$F \sim \int_0^{h_1} Q(z_1,h_1) \int_0^{h_2} Q(z_2,h_2) \cdots \int_0^{h_M} Q(z_M,h_M) \min_{i=1}^{M}(z_i \lg 2k_i)\, dz_M \cdots dz_2 dz_1$$

Für beliebiges $z_1 \in \mathbb{R}$ mit $z_1 \ge 0$ gilt wegen $Q(z_i,h_i) \le 1$ und $h_i \le p$

$$\int_0^{h_2} Q(z_2,h_2) \int_0^{h_3} Q(z_3,h_3) \cdots \int_0^{h_M} Q(z_M,h_M) \min_{i=1}^{M}(z_i \lg 2k_i)\, dz_M \cdots dz_3 dz_2 = O\big(p^M\big)$$

Folglich gilt

$$\begin{aligned}
&\int_{h_1}^{\infty} Q(z_1,h_1) \int_0^{h_2} Q(z_2,h_2) \cdots \int_0^{h_M} Q(z_M,h_M) \min_{i=1}^{M}(z_i \lg 2k_i)\, dz_M \cdots dz_2 dz_1 \\
&= O\big(p^M\big) \int_{h_1}^{\infty} \frac{z_1}{h_1} e^{-z_1^2/(2h_1)} dz_1 \\
&= O\big(p^M\big) e^{-h_1/2} \to 0 \quad (p \to \infty)
\end{aligned}$$

Indem dieses Argument wiederholt angewandt wird, folgt

$$F \sim \int_0^{\infty} \frac{z_1 e^{-z_1^2/(2h_1)}}{h_1} \cdots \int_0^{\infty} \frac{z_M e^{-z_M^2/(2h_M)}}{h_M} \min_{i=1}^{M}(z_i \lg 2k_i)\, dz_M \cdots dz_1$$

Nun wird ein Variablenwechsel $y_i = z_i \lg 2k_i$ durchgeführt. Damit gilt

$$F \sim \left(\prod_{i=1}^{M} \frac{1}{\lg 2k_i}\right) \int_0^\infty \frac{y_1 e^{-y_1^2/(2h_1 \lg^2 2k_1)}}{h_1 \lg 2k_1} \cdots \int_0^\infty \frac{y_M e^{-y_M^2/(2h_M \lg^2 2k_M)}}{h_M \lg 2k_M} \min_{i=1}^{M}(y_i)\, dy_M \cdots dy_1$$

Anstatt über alle y_i von 0 bis ∞ zu integrieren, wird nun unterschieden, welches der y_i das Minimum ist. Wenn y_j das Minimum ist, wird über y_j von 0 bis ∞ integriert und über y_i $(i \neq j)$ von y_j bis ∞. Dadurch ist es möglich, $\min_{i=1}^{M}(y_i)$ nach vorne zu ziehen, sodass die Integrale für $i \neq j$ unabhängig voneinander ausgewertet werden können. Nun kann jedes der y_i das Minimum sein, das heißt es wird über alle möglichen j aufsummiert, wobei y_j immer das Minimum ist. Es gilt also

$$F \sim \left(\prod_{i=1}^{M} \frac{1}{\lg 2k_i}\right) \sum_{j=1}^{M} \int_0^\infty \frac{y_j^2 e^{-y_j^2/(2h_j \lg^2 2k_j)}}{h_j \lg 2k_j} \prod_{i=1, i\neq j}^{M} \int_{y_j}^\infty \frac{y_i e^{-y_i^2/(2h_i \lg^2 2k_i)}}{h_i \lg 2k_i} dy_i\, dy_j$$

Für eine einfachere Notation werden die Integrationsvariablen von y_j zu y und von y_i zu x umbenannt. Zunächst werden die Integrale im Produkt rechts ausgewertet. Es gilt

$$\begin{aligned}\int_y^\infty \frac{x e^{-x^2/(2h_i \lg^2 2k_i)}}{h_i \lg 2k_i} dx &= e^{-x^2/(2h_i \lg^2 2k_i)}(-\lg 2k_i)\Big|_y^\infty \\ &= e^{-y^2/(2h_i \lg^2 2k_i)} \lg 2k_i\end{aligned}$$

Also gilt

$$\begin{aligned}F &\sim \left(\prod_{i=1}^{M} \frac{1}{\lg 2k_i}\right) \sum_{j=1}^{M} \int_0^\infty \frac{y^2 e^{-y^2/(2h_j \lg^2 2k_j)}}{h_j \lg 2k_j} \prod_{i=1, i\neq j}^{M} e^{-y^2/(2h_i \lg^2 2k_i)} \lg 2k_i\, dy \\ &= \left(\prod_{i=1}^{M} \frac{1}{\lg 2k_i}\right) \sum_{j=1}^{M} \int_0^\infty \frac{y^2}{h_j \lg^2 2k_j} \prod_{i=1}^{M} e^{-y^2/(2h_i \lg^2 2k_i)} \lg 2k_i\, dy \\ &= \sum_{j=1}^{M} \int_0^\infty \frac{y^2}{h_j \lg^2 2k_j} \prod_{i=1}^{M} e^{-y^2/(2h_i \lg^2 2k_i)}\, dy \\ &= \left(\sum_{j=1}^{M} \frac{1}{h_j \lg^2 2k_j}\right) \int_0^\infty y^2\, e^{-\sum_{i=1}^{M} y^2/(2h_i \lg^2 2k_i)}\, dy\end{aligned}$$

Um das Integral auszuwerten, wurde die Tabelle in „Gaussian Integral — Wikipedia“, 2023 verwendet. Damit erhält man schließlich

$$\begin{aligned}F &\sim \left(\sum_{i=1}^{M} \frac{1}{h_i \lg^2 2k_i}\right) \sqrt{\pi/2} \left(\sum_{i=1}^{M} \frac{1}{h_i \lg^2 2k_i}\right)^{-3/2} \\ &= \sqrt{\pi p/2} \left(\sum_{i=1}^{M} \frac{\gcd(p-1, 2k_i) - 1}{\lg^2 2k_i}\right)^{-1/2}\end{aligned}$$

□

Um $L_{(k_i)}(p) \sim F$ zu zeigen, teile man die erste Summe in (3) bei $h_1^{9/16}$

$$F = \sum_{z_1=1}^{h_1^{9/16}} Q(z_1,h_1) \sum_{z_2=1}^{h_2} Q(z_2,h_2) \cdots \sum_{z_M=1}^{h_M} Q(z_M,h_M) \min_{i=1}^{M}(z_i \lg 2k_i)$$
$$+ \sum_{z_1=h_1^{9/16}}^{h_1} Q(z_1,h_1) \sum_{z_2=1}^{h_2} Q(z_2,h_2) \cdots \sum_{z_M=1}^{h_M} Q(z_M,h_M) \min_{i=1}^{M}(z_i \lg 2k_i)$$

Da die Summen über z_2 bis z_M durch ein Polynom in p beschränkt sind, geht die zweite Zeile nach Lemma 2 gegen 0. Indem auf die erste Summe Lemma 3 mit $\delta = 1/16$ angewandt wird, folgt

$$F \sim \sum_{z_1=1}^{h_1^{9/16}} \left(\mathbb{P}(Z_1 = z_1) + O\left(h_1^{-3/4}\right)\right) \sum_{z_2=1}^{h_2} Q(z_2,h_2) \cdots \sum_{z_M=1}^{h_M} Q(z_M,h_M) \min_{i=1}^{M}(z_i \lg 2k_i)$$

Nach Lemma 4 gilt $\sum_{z_2=1}^{h_2} Q(z_2,h_2) \cdots \sum_{z_M=1}^{h_M} Q(z_M,h_M) \min_{i=1}^{M}(z_i \lg 2k_i) = O(\sqrt{p})$. Folglich ist die Summe aller Terme mit einem $O\big(h_1^{-3/4}\big)$-Faktor durch $O(p^{5/16})$ beschränkt. Für eine asymptotische Näherung von F können sie also weggelassen werden. Indem dieses Argument für $Q(z_2,h_2)\ldots,Q(z_M,h_M)$ wiederholt wird, können alle $Q(z_i,h_i)$ durch $\mathbb{P}(Z_i = z_i)$ ersetzt werden. Da $\mathbb{P}(z_i,h_i) = O(Q(z_i,h_i))$, können die oberen Grenzen der Summen von $h_i^{9/16}$ durch eine erneute Anwendung von Lemma 2 zurück zu h_i geändert werden. Daraus folgt $L_{(k_i)}(p) \sim F$.

Aus Lemma 4 und $L_{(k_i)}(p) \sim F$ folgt nun eine asymptotische Näherung für die Laufzeit der Rho-Methode.

Satz 2. *Unter der Random Mapping Assumption gilt für die erwartete Laufzeit der Rho-Methode auf M Maschinen mit k-Werten $k_1,\ldots,k_M$*

$$L_{k_1,\ldots,k_M}(p) \sim \sqrt{\pi p/2}\left(\sum_{i=1}^{M} \frac{\gcd(p-1,2k_i)-1}{\lg^2 2k_i}\right)^{-1/2} \tag{4}$$

Als Beispiel kann man $M = 1$ und $k_1 = 1$ betrachten. In diesem Fall vereinfacht sich (4) zu $\sqrt{\pi p/2}$, was genau das Ergebnis in Pollard, 1975, S. 332 ist.

5 Bestimmung optimaler Exponenten für die Rho-Methode

In diesem Abschnitt wird die Frage behandelt, wie der Parameter k bei M Maschinen bestmöglich gewählt werden kann. Mit Satz 2 konnten Ergebnisse in den Fällen $M = 1$ und $M = 2$ erzielt werden. Die grundlegende Strategie ist, den Erwartungswert von $L_{k_1,\ldots,k_M}(p)$ über alle Möglichkeiten von $\gcd(p-1,2k_i)$ für alle $1 \le i \le M$ zu bilden und so einen Wert für die erwartete Laufzeit in Abhängigkeit der k_i zu erhalten. Da $p-1$ gerade ist, gilt $\gcd(p-1,2k_i) = 2\gcd((p-1)/2,k_i)$. Weitere Kongruenzen von $p-1$ sind im Allgemeinen nicht bekannt, weshalb angenommen wird, dass jeder Rest von $(p-1)/2$ modulo k_i gleich wahrscheinlich ist.

Satz 3. *Sei $L_k(p)$ wie in (1) definiert. Der Erwartungswert $\mathbb{E}(L_k(p))$ über alle möglichen $\gcd((p-1)/2,k)$ nimmt ein globales Minimum für $k = 1$ an.*

Beweis. Durch Einsetzen von $M = 1$ in (4) erhalten wir

$$L_k(p) \sim \sqrt{\pi p/2}\,\frac{\lg 2k}{\sqrt{\gcd(p-1,2k)-1}}$$

Die erwartete Laufzeit im Fall $k = 1$ ist folglich $\sqrt{\pi p/2}$. Es wird also gezeigt, dass $\mathbb{E}(L_k(p)) > \sqrt{\pi p/2}$ für $k > 1$. Mit φ wird die eulersche Phifunktion bezeichnet. Dann gilt

$$\mathbb{E}(L_k(p)) \sim \sqrt{\pi p/2}\,\lg(2k)\sum_{d|k}\frac{\mathbb{P}(\gcd((p-1)/2,k)=d)}{\sqrt{2d-1}} \geq \sqrt{\pi p/2}\,\lg(2k)\,\frac{\varphi(k)}{k}$$

Für die Ungleichung wurde statt der Summe über alle Teiler nur $d = 1$ betrachtet. Da es $\varphi(k)$ teilerfremde Zahlen kleiner k gibt, ist $\mathbb{P}(\gcd((p-1)/2,k) = 1) = \varphi(k)/k$. Nach Rosser und Schoenfeld, 1962, Theorem 15 gilt $\varphi(k)/k > 1/(e^\gamma \ln(\ln(k))+2.51/\ln(\ln(k)))$ für $k \geq 3$, wobei $\gamma \approx 0.5772$ die Euler-Mascheroni-Konstante ist. Für $k \geq e^e$ folgt daraus $\varphi(k)/k > 1/(e^\gamma \ln(\ln(k)) + 2.51)$. Indem nun gezeigt wird, dass $\lg(2k)/(e^\gamma \ln(\ln(k)) + 2.51) > 1$ für $k \geq 16$ wird der Satz im Fall $k \geq 16$ bewiesen. Sei $f(x) = \lg(2x)/(e^\gamma \ln(\ln(x)) + 2.51)$. Es gilt $f(16) \approx 1.1557$ und

$$f'(x) = \frac{\ln(x)(\ln(\ln(x)) + 1.51) - \ln 2}{x \ln(2) \ln(x)(\ln(\ln(x)) + 2.51)^2}$$

Für $x \geq 16$ ist der Nenner von f' positiv, denn $x > 0$ und $\ln x > 0$, und weil $\ln\ln x > 1$ für $x \geq 16$ ist der Term unter dem Quadrat positiv. Der Zähler ist ebenfalls positiv, da $\ln x \geq 1$ und $\ln\ln x \geq 1$, woraus $\ln(x)(\ln(\ln(x)) + 1.51) \geq 1 \cdot (1 + 1.51) = 2.51 > \ln 2$ folgt. Also ist f streng monoton steigend für $x \geq 16$, und da bereits $f(16) > 1$ gezeigt wurde, folgt $f(x) > 1$ für $x \geq 16$. Der Fall $k < 16$ wurde durch Ausrechnen von (4) für $2 \leq k \leq 15$ überprüft. □

Nun sehen wir uns den Fall von zwei Maschinen an.

Satz 4. *Sei $L_{k_1,k_2}(p)$ wie in (1) definiert. Man nehme RMA an und bilde den Erwartungswert $\mathbb{E}(L_{k_1,k_2})$ über alle möglichen $\gcd((p-1)/2,k_i)$ für $i = 1, 2$.*

1. *Wenn k_1, k_2 Primzahlen sind, gilt $\mathbb{E}(L_{1,1}(p)) < \mathbb{E}(L_{k_1,k_2}(p))$.*
2. *Wenn $1 < k \in \mathbb{N}$, gilt $\mathbb{E}(L_{1,1}(p)) < \mathbb{E}(L_{k,k}(p))$.*

Beweis. Nach Satz 2 gilt

$$L_{k_1,k_2}(p) \sim \sqrt{\pi p/2}\left(\frac{\gcd(p-1,2k_1)-1}{\lg^2 2k_1} + \frac{\gcd(p-1,2k_2)-1}{\lg^2 2k_2}\right)^{-1/2}$$

Durch Einsetzen von $k_1 = k_2 = 1$ erhalten wir $L_{1,1}(p) \sim \sqrt{\pi p/4}$. Es gilt also in beiden Teilen zu zeigen, dass der Erwartungswert größer ist. Zunächst wird Teil 1 bewiesen.

Durch Bilden des Erwartungswerts über alle möglichen $\gcd((p-1)/2,k_i)$ für $i = 1, 2$ erhält man

$$\mathbb{E}(L_{k_1,k_2}) \sim \sqrt{\pi p/2}\sum_{d_1|k_1}\mathbb{P}(\gcd((p-1)/2,k_1)=d_1)$$

$$\sum_{d_2|k_2} \mathbb{P}(\gcd((p-1)/2, k_2) = d_2) \left(\frac{2d_1 - 1}{\lg^2 2k_1} + \frac{2d_2 - 1}{\lg^2 2k_2} \right)^{-1/2}$$

$$\geq \sqrt{\pi p/2}\ \frac{\varphi(k_1)\varphi(k_2)}{k_1 k_2} \left(\frac{1}{\lg^2 2k_1} + \frac{1}{\lg^2 2k_2} \right)^{-1/2}$$

$$= \sqrt{\pi p/2}\ \frac{(k_1 - 1)(k_2 - 1)}{k_1 k_2} \sqrt{\frac{\lg^2(2k_1)\lg^2(2k_2)}{\lg^2(2k_1) + \lg^2(2k_2)}} \tag{5}$$

Wie bei $M = 1$ wurden die Summen über alle Teiler von k_1, k_2 durch den Wert bei $d_1 = d_2 = 1$ nach unten begrenzt. Die Terme $(k_i - 1)/k_i$ sind streng monoton steigend in k_i. Ebenso ist das Argument der nachfolgenden Wurzel in (5) streng monoton steigend in jedem der k_i. Für die Monotonie in k_1 bemerke man zunächst, dass lg streng monoton steigt und setze $x = \lg^2 2k_1, y = \lg^2 2k_2$. Sei $x' > x$ beliebig. Dann gilt

$$\frac{x'y}{x' + y} = \frac{x'y}{x' + y} \frac{x + y}{xy} \frac{xy}{x + y} = \frac{xx'y + x'y^2}{xx'y + xy^2} \frac{xy}{x + y} > \frac{xy}{x + y}$$

da $x, x', y > 0$ und $x < x'$. Der Term ist symmetrisch in x und y, womit er auch streng monoton steigend in y ist. Da die Wurzelfunktion streng monoton steigt und die Verkettung streng monoton steigender Funktionen streng monoton steigt, folgt, dass die gesamte Wurzel auf der rechten Seite von (5) streng monoton steigt. Weil nun jeder einzelne Faktor in (5) streng monoton steigt und positiv ist, ist ganz (5) streng monoton steigend in den k_i. Indem man $k_1 = k_2 = 3$ in (5) einsetzt, sieht man, dass $E(L_{3,3}(p)) \geq 0.8123\sqrt{\pi p/2} > \sqrt{\pi p/4}$. Weil (5) symmetrisch in k_1, k_2 ist, kann $k_1 < k_2$ angenommen werden. Dann folgt aus der Monotonie von (5), dass $E(L_{k_1,k_2}(p)) > \mathbb{E}(L_{1,1}(p))$, wenn $k_1 \geq 3$. Es bleibt also lediglich der Fall $k_1 = 2$. Durch Einsetzen von $k_1 = 2,\ k_2 = 7$ in (5) gilt $\mathbb{E}(L_{2,7}(p)) \geq 0.7588\sqrt{\pi p/2} > \sqrt{\pi p/4}$. Aus der Monotonie von (5) folgt $\mathbb{E}(L_{k_1,k_2}(p)) > \mathbb{E}(L_{1,1}(p))$ für $k_1 = 2$ und $k_2 \geq 7$. Die übrigen Fälle $k_1 = 2$ und $k_2 = 2, 3, 5$ wurden nachgerechnet.

Nun zum Beweis von Teil 2. Aus (4) folgt $L_{k,k}(p) = L_k(p)/\sqrt{2}$ für $1 \leq k \in \mathbb{N}$. Damit folgt Teil 2 des Satzes aus Satz 3. □

6 Experimentelle Ergebnisse

Um zu demonstrieren, dass Satz 2 das Laufzeitverhalten von Pollards Rho-Algorithmus gut beschreibt, wurden Laufzeitmessungen für $M = 1, M = 2$ und $M = 3$ durchgeführt. Als Testzahlen wurden 192-Bit-Zahlen verwendet, bei denen jeder Primfaktor größer als 2^{22} ist und ein Primfaktor kleiner als 2^{23} ist. Die Größe der Primfaktoren nach unten zu beschränken ist sinnvoll, da die Laufzeit der Rho-Methode sonst sehr kurz wäre und die Messergebnisse durch Dinge wie den Aufwand der Laufzeitmessung selbst verfälscht werden würden. Es ist auch sinnvoll, dass die Testzahlen einen kleinen Primfaktor enthalten, da die Laufzeit sonst sehr groß wäre. Es wurden jeweils $2^{14} = 16384$ Testzahlen verwendet. Die mittlere Laufzeit und Werte von (4) zum Vergleich sind in den Abbildungen 2, 3 und 4 dargestellt. Sowohl die Werte der Formel als auch die Messdaten wurden skaliert, sodass bei $k_i = 1$ für alle $1 \leq i \leq M$ der Wert 1 steht. Der Code für die Laufzeitmessungen ist unter `https://github.com/finn-rudolph/rhok` verfügbar.

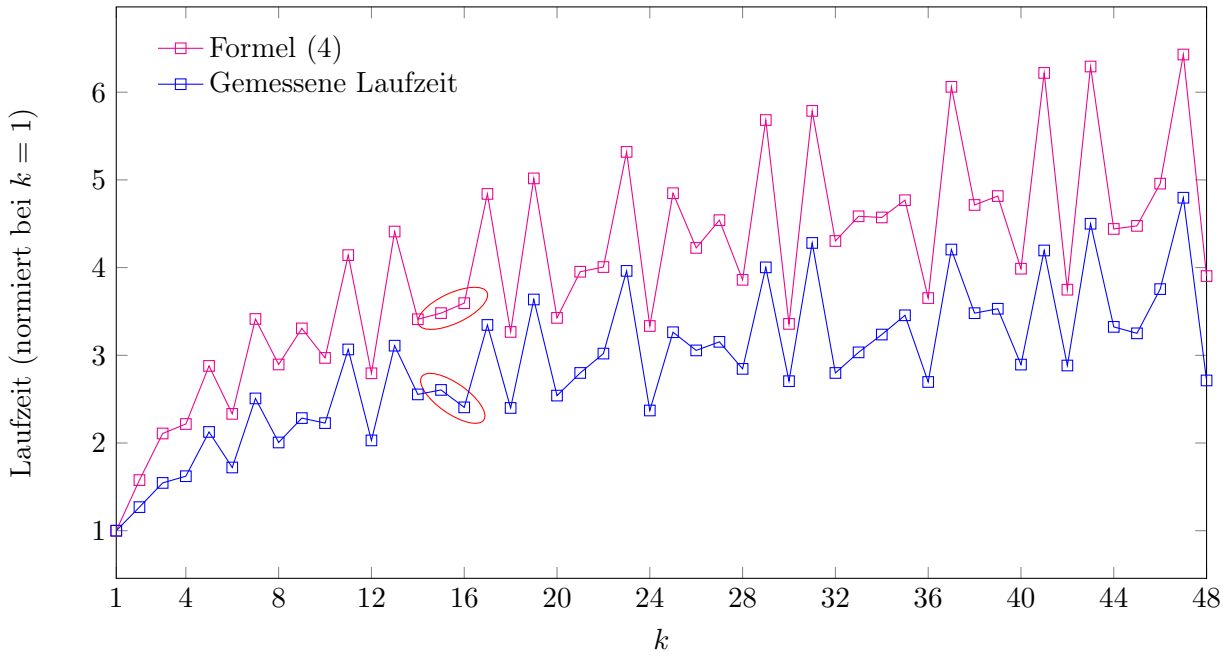

Abbildung 2: Die durchschnittliche gemessene Laufzeit des Rho-Algorithmus für $M = 1$ und Werte von Formel (4) für $1 \leq k \leq 48$.

$k_1 \backslash k_2$	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1.00	1.10	1.18	1.20	1.25	1.23	1.27	1.25	1.27	1.27	1.29	1.27	1.29	1.28
2		1.32	1.38	1.44	1.51	1.48	1.56	1.53	1.56	1.57	1.61	1.56	1.61	1.60
3			1.66	1.60	1.75	1.70	1.81	1.72	1.86	1.79	1.89	1.80	1.90	1.84
4				1.72	1.81	1.74	1.89	1.86	1.86	1.89	1.98	1.86	1.98	1.94
5					2.23	1.95	2.25	2.07	2.20	2.25	2.42	2.10	2.44	2.29
6						1.94	2.02	1.97	2.06	2.01	2.11	2.02	2.11	2.06
7							2.58	2.20	2.38	2.38	2.69	2.27	2.71	2.58
8								2.15	2.14	2.18	2.32	2.13	2.33	2.27
9									2.50	2.35	2.57	2.36	2.58	2.44
10										2.46	2.56	2.30	2.57	2.47
11											3.13	2.42	3.07	2.78
12												2.32	2.42	2.37
13													3.16	2.78
14														2.74

$k_1 \backslash k_2$	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1.00	1.17	1.24	1.24	1.29	1.25	1.32	1.28	1.31	1.29	1.34	1.27	1.35	1.31
2		1.58	1.67	1.80	1.84	1.78	1.92	1.93	1.87	1.89	2.00	1.87	2.02	1.95
3			2.11	1.97	2.22	2.16	2.36	2.18	2.50	2.21	2.51	2.27	2.55	2.32
4				2.22	2.27	2.17	2.42	2.48	2.35	2.39	2.58	2.36	2.63	2.51
5					2.88	2.32	2.88	2.60	2.78	2.85	3.13	2.51	3.21	2.82
6						2.33	2.48	2.44	2.62	2.41	2.66	2.51	2.72	2.54
7							3.41	2.83	3.05	2.87	3.50	2.72	3.60	3.32
8								2.90	2.73	2.77	3.08	2.73	3.16	2.96
9									3.31	2.77	3.36	2.86	3.46	2.99
10										2.97	3.13	2.65	3.21	2.95
11											4.14	2.96	4.05	3.42
12												2.79	3.03	2.83
13													4.41	3.52
14														3.41

Abbildung 3: Die durchschnittliche Laufzeit von Pollards Rho-Algorithmus (oben) und berechnete Werte für die erwartete Laufzeit (unten), für $M = 2$ und $1 \leq k_1 \leq k_2 \leq 14$.

k_1	k_2	k_3	Laufzeit	Formel (4)
1	1	1	1.00	1.00
1	1	2	1.07	1.10
1	1	3	1.10	1.14
1	1	4	1.10	1.14
1	2	2	1.16	1.25
1	2	3	1.20	1.29
1	2	4	1.22	1.31
1	3	3	1.29	1.38
1	3	4	1.29	1.36
1	4	4	1.31	1.39
2	2	2	1.34	1.58
2	2	3	1.39	1.62
2	2	4	1.41	1.71
2	3	3	1.48	1.75
2	3	4	1.50	1.77
2	4	4	1.53	1.91
3	3	3	1.74	2.11
3	3	4	1.72	1.97
3	4	4	1.72	2.00
4	4	4	1.81	2.22

Abbildung 4: Laufzeitmessungen und theoretische Werte von Formel (4) für $M = 3$ und $1 \leq k_1 \leq k_2 \leq k_3 \leq 4$.

6.1 Eine Maschine. Man sieht, dass der Verlauf der Laufzeit von Formel (4) widergespiegelt wird. Beispielsweise werden hohe Werte bei Primzahlen und niedrige Werte bei Zahlen mit vielen verschiedenen Primfaktoren (z.B. 24) angenommen.

Für $k > 1$ ist die Laufzeit im Verhältnis zur Laufzeit bei $k = 1$ allerdings geringer als erwartet. Eine Erklärung dafür ist, dass der Aufwand der Berechnung von $\gcd(n, x_i - x_j)$ in der Rho-Methode vernachlässigt wurde. Obwohl, wie in Abschnitt 2 beschreiben, die Optimierung aus Brent, 1980 verwendet wurde, um diesen Aufwand zu amortisieren, macht er immer noch einen relevanten Teil aus. Die Laufzeit ist also geringer als erwartet, was zunächst widersprüchlich erscheint, da der gcd in der Analyse vernachlässigt wurde. Das lässt sich dadurch erklären, dass die Laufzeit relativ zu $k = 1$ dargestellt ist und der größere Aufwand der Berechnung von x^{2k} für größere k in der Praxis weniger ins Gewicht fällt als in der Analyse.

An den Messergebnissen ist noch eine weitere Vereinfachung in der Analyse erkennbar. Wenn man die Laufzeit und Werte von (4) bei $k = 8, 16, 32$ mit nahe gelegenen Werten vergleicht, beobachtet man, dass die Laufzeit im Verhältnis kleiner ist. Beispielsweise ist die gemessene Laufzeit bei $k = 16$ geringer als bei $k = 15$, der theoretische Wert von (4) jedoch höher (siehe Markierung in Abbildung 2), und die Laufzeit bei $k = 32$ ist nahezu gleich der bei $k = 30$, wohingegen der Wert von (4) deutlich größer ist. Das liegt wahrscheinlich daran, dass Square-and-Multiply (Knuth, 1998, S. 461) zur Berechnung von der Aufwand zur Berechnung von x^{2k} verwendet wurde. Die Laufzeit dieses Algorithmus ist nicht genau $\lg 2k$, sondern hängt in der Praxis von der Anzahl an Einsen in der Binärdarstellung von k ab, und Zweierpotenzen haben eine minimale Anzahl an Einsen in ihrer Binärdarstellung.

6.2 Zwei und drei Maschinen. Auch für $M = 2$ und $M = 3$ beschreibt (4) das Laufzeitverhalten gut. Denn die Formel nimmt bei genau den Parameterwahlen hohe bzw. niedrige Werte an, bei denen auch die Laufzeit hoch bzw. niedrig ist. In Bezug auf $M = 2$ wird die Aussage

von Satz 4 bestätigt, und es gibt zumindest für $1 \leq k_1 \leq k_2 \leq 14$ kein Paar k_1, k_2 mit einer geringeren Laufzeit als $k_1 = k_2 = 1$. Da die Laufzeit für größere k_1, k_2 tendenziell zu steigen scheint, wird die Vermutung aufgestellt, dass $k_1 = k_2 = 1$ optimal ist. Auch bei $M = 3$ legen die Werte der Formel und die Laufzeitmessungen nahe, dass $k_1 = k_2 = k_3 = 1$ optimal ist (siehe Abbildung 4). Berechnet man für $M = 3$ Werte der Formel für noch größere k, sieht man, dass sich der tendenzielle Anstieg der Laufzeit fortsetzt.

7 Fazit

Der Parameter k der Rho-Methode wurde bei bekannten Kongruenzen der Primfaktoren schon erfolgreich zur Verringerung der Laufzeit eingesetzt. Allerdings war über die optimale Wahl bei allgemeinen Zahlen bisher wenig bekannt, insbesondere bei einer parallelen Ausführung auf mehreren Maschinen. Die in dieser Arbeit bestimmte Formel für die erwartete Laufzeit bei gegebenen k-Werten ermöglicht ein besseres Verständnis des Laufzeitverhaltens der parallelen Rho-Methode. Mithilfe der Formel war es möglich zu zeigen, dass für $M = 1$ am besten $k = 1$ gewählt wird, und für $M = 2$ die Wahl $k_1 = k_2 = 1$ besser ist als zwei Primzahlen und zwei gleiche Zahlen größer als 1. Dass $k_1 = k_2 = 1$ optimal ist, konnte anhand der Formel noch nicht formal bewiesen werden, Laufzeitmessungen und berechnete Werte der Formel lassen dies aber vermuten. Die Laufzeitmessungen zeigen ebenfalls, dass Formel (4) das Laufzeitverhalten der Rho-Methode gut beschreibt und daher zum Vergleich der Laufzeit bei verschiedenen Werten von k geeignet ist. Ein nächster Schritt in der Beantwortung der Frage nach der optimalen Parametrisierung könnte der Beweis sein, dass $k_1 = k_2 = 1$ für $M = 2$ optimal ist. Daneben könnte man versuchen, Teilergebnisse für $M \geq 3$ zu erzielen.

Literatur

Asymptotic analysis — Wikipedia. (2024). https://en.wikipedia.org/wiki/Asymptotic_analysis

Brent, R. P. (1980). An improved Monte Carlo factorization algorithm. *BIT Numerical Mathematics*, *20*(2), 176–184. https://doi.org/10.1007/BF01933190

Brent, R. P., & Pollard, J. M. (1981). Factorization of the eighth Fermat number. *Mathematics of Computation*, *36*, 627–630. https://doi.org/10.1090/S0025-5718-1981-0606520-5

Crandall, R. E. (1999). https://www.reed.edu/physics/faculty/crandall/papers/parrho.pdf

Flajolet, P., & Odlyzko, A. M. (1990). Random Mapping Statistics. In J.-J. Quisquater & J. Vandewalle (Hrsg.), *Advances in Cryptology — EUROCRYPT '89* (S. 329–354). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-46885-4_34

Gaussian Integral — Wikipedia. (2023). https://en.wikipedia.org/wiki/Gaussian_integral

Knuth, D. E. (1998). *The Art of Computer Programming - Volume 2 (Seminumerical Algorithms)*. Addison-Wesley.

Pollard, J. M. (1975). A monte carlo method for factorization. *BIT Numerical Mathematics*, *15*(3), 331–334. https://doi.org/10.1007/BF01933667

Rosser, J. B., & Schoenfeld, L. (1962). Approximate formulas for some functions of prime numbers. *Illinois Journal of Mathematics*, *6*(1), 64–94. https://doi.org/10.1215/ijm/1255631807

van Oorschot, P. C., & Wiener, M. J. (1999). Parallel Collision Search with Cryptanalytic Applications. *Journal of Cryptology*, *12*(1), 1–28. https://doi.org/10.1007/PL00003816