Institut für Formale Methoden der Informatik
Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 3259

Der Algorithmus von Howgrave-Graham und Joux zur Lösung von Rucksackproblemen

Michael Ludwig

Studiengang:	Informatik
Prüfer:	Prof. Dr. Volker Diekert
Betreuer:	Prof. Dr. Volker Diekert
begonnen am:	24. Oktober 2011
beendet am:	24. April 2012
CR-Klassifikation:	F.2.2, G.2.1

Inhaltsverzeichnis

Abbildungsverzeichnis

Verzeichnis der Algorithmen

1. Einleitung

Das Rucksackproblem ist eines der prominentesten NP-vollständigen Probleme, nicht zuletzt weil es unter den 21 klassischen Problemen Karp's zu finden ist [Kar72]. Die Aufgabenstellung hierbei ist leicht: Gegeben ist eine Menge von Zahlen, auch *Gewichte* genannt, sowie ein Zielgewicht; ist es möglich, eine Teilmenge an Zahlen zu finden, die aufaddiert das Zielgewicht ergibt? Das Rucksackproblem hat kombinatorischen Charakter und im allgemeinen Fall sind natürlich nur Exponentialzeitalgorithmen bekannt. Es gibt auch Instanzen, die schneller lösbar sind. Gegenstand dieser Arbeit sind jedoch gerade die schwer lösbaren Instanzen und die zugehörigen Algorithmen. Gute Algorithmen zu finden bedeutet hier zum einen, den Faktor im Exponent der Laufzeit- bzw. Platzkomplexitäten zu reduzieren, zum anderen gute Tradeoffs zwischen Speicher und Zeit zu finden. Der 2010 entdeckte Algorithmus von Howgrave-Graham und Joux steht im Mittelpunkt der vorliegenden Arbeit. Sie ist wie folgt gegliedert:

1. **Einleitung**
2. **Das Rucksackproblem**. Der Definition des Problems schließt sich eine grobe Klassifizierung schwer zu lösender Probleme an.
3. **Eine Heuristik**. Alle neuen hier behandelten Algorithmen sind, im Gegensatz zu den älteren, randomisiert und verwenden eine Heuristik, die in diesem Abschnitt erklärt und bewiesen wird.
4. **Der Algorithmus von Horowitz und Sahni**. Dies ist der erste Ansatz, der besser ist, als die vollständige Lösungsraumsuche. Er bildet die Grundlage für alle weiteren Algorithmen.
5. **Der Algorithmus von Schroeppel und Shamir**. Wenige Jahre, nachdem der erste Algorithmus veröffentlicht wurde, konnte in einem verfeinerten Ansatz der Speicherverbrauch reduziert werden. Dieser Algorithmus ist, wie sein Vorgänger, deterministisch. Es wird außerdem eine randomisierte Version dieses Algorithmus angegeben, die auf Howgrave-Graham und Joux zurückgeht.
6. **Der Algorithmus von Howgrave-Graham und Joux**. Dies ist der neue Algorithmus aus dem Jahr 2010 und der Hauptgegenstand dieser Arbeit. Er baut auf der randomisierten Version des Schroeppel-Shamir-Algorithmus auf.
7. **Erweiterung des Tradeoffs**. Eines der beiden Hauptziele bei der Suche nach guten Algorithmen ist es, gute Tradeoffs zu finden. Im Rahmen dieser Arbeit konnte hierbei eine Verbesserung erzielt werden.
8. **Übersicht**. Hier werden die Komplexitäten der zahlreichen Algorithmen zusammengetragen und gegenübergestellt.

9. **Fazit**. Es wird eine Zusammenfassung der Arbeit gegeben und offene Probleme besprochen.

Im Anhang finden sich unter anderem einige Grundlagen zu Begrifflichkeiten der Algorithmik und zur Stochastik.
Das Rucksackproblem hat unzählige Anwendungen. Je nach Formulierung stellt es eine typische Optimierungsaufgabe, wie sie in vielen Feldern der Praxis auftritt, dar, oder, wie im hier behandelten Fall, eine kombinatorische Aufgabe. So ist Subset Sum vor allem für die Kryptographie interessant. Ein bekanntes Beispiel ist das Merkle-Hellman-Kryptosystem [MMI+78], welches attraktiv erschien, da es auf einem NP-vollständigen Problem basierte: Subset Sum. Doch Shamir konnte dieses Kryptosystem in Polynomialzeit brechen [Sha84]. In der modernen Kryptographie findet sich das Rucksackproblem immer noch wieder. Das NTRU-Kryptosystem [HPS98] zieht in der Kryptoanalyse zum Beispiel Rucksackprobleme nach sich [HG07]. Die SWIFFT-Haschfunktion ist ein weiterer Anwendungsbereich [LMPR08].

2. Das Rucksackproblem

Das klassische Rucksackproblem[1] ist ein kombinatorisches Optimierungsproblem. Hierbei ist eine Menge von Objekten gegeben. Jedes Objekt besitzt einen Kosten- und einen Nutzenkoeffizient. Des weiteren ist eine Grenze für die Kosten gegeben. Ziel ist es, eine möglichst große Menge an Objekten zu wählen, sodass die erlaubten Kosten genau ausgeschöpft werden und der Nutzen maximiert wird. Dieses Grundschema ist vielfältig modifizierbar; man kann also von der *Klasse* der Rucksackprobleme sprechen. Sie alle haben gemein, dass sie in polynomieller Zeit aufeinander reduzierbar sind.
Die Variante, die die Algorithmen, die in dieser Arbeit behandelt werden, lösen, heißt **Subset Sum**. In dieser fällt der Nutzenkoeffizient der Objekte weg. In der Entscheidungsvariante ist das Problem wie folgt definiert:

Definition 2.1 (Subset Sum) *Gegeben:* $A \subset \mathbb{Z}$ *mit* $|A| = n$ *und* $c \in \mathbb{Z}$
Gefragt: Existiert ein $X \subseteq A$ *mit* $\sum_{x \in X} x = c$

In der Suchvariante sind alle Lösungen X anzugeben. Es ist möglich, Subset Sum auf syntaktisch andere Art zu definieren.

Definition 2.2 (Subset Sum (alternativ)) *Gegeben:* $\mathbf{a} \in \mathbb{Z}^n$ *und* $c \in \mathbb{Z}$
Gefragt: Existiert ein $\mathbf{x} \in \mathbb{B}^n$ *mit* $\sum_{i=1}^{n} a_i x_i = \mathbf{ax} = c$

Je nach Kontext können unterschiedliche Definitionen handlicher sein. Es ist immer offensichtlich, welche verwendet wird. $\mathbf{a}$ beziehungsweise das Paar $(\mathbf{a}, c)$, bestehend aus den *Gewichten* $\mathbf{a}$ und dem *Zielgewicht* c, heißt *Instanz*.

Beispiel 2.3 ($n = 8$) *Es sei folgende Instanz gegeben:* $\mathbf{a} = (7, -78, 40, 251, -3, 111, 98, -249)$ *und* $c = 70$.
Eine Lösung ist $\mathbf{x} = (0, 1, 1, 0, 1, 1, 0, 0)$, *da*

$$0 \cdot 7 + 1 \cdot -78 + 1 \cdot 40 + 0 \cdot 251 + 1 \cdot -3 + 1 \cdot 111 + 0 \cdot 98 + 0 \cdot -249 = 70.$$

Subset Sum gehört zu Karp's 21 klassischen NP-vollständigen Problemen. Der Beweis zur NP-Vollständigkeit ist in Anhang A.4 zu finden. Es sind also keine allgemein gültigen Subexponentialzeitalgorithmen zu erwarten. Eine einfache Möglichkeit, kombinatorische Probleme zu lösen, besteht im Absuchen des gesamten Lösungsraumes; in diesem Fall ist dies $\mathbb{B}^n$. So hat ein

[1] engl.: Knapsack problem

solcher **naiver Algorithmus** eine Zeitkomplexität in $\mathcal{O}(n2^n)$ und eine Platzkomplexität in $\mathcal{O}(n)$.
Es gibt auch eine **modulare Variante von Subset Sum**, die später benötigt werden wird. $M \in \mathbb{Z}$ ist hierbei stets der Modulus.

Definition 2.4 (Subset Sum (modular)) *Gegeben:* $\mathbf{a} \in \mathbb{Z}^n$ *und* $c \in \mathbb{Z}$
Gefragt: Existiert ein $\mathbf{x} \in \mathbb{B}^n$ *mit* $\mathbf{ax} \equiv c \pmod{M}$

Auch hier kann es handlich sein, in der Definiton flexibel zu sein; es ist möglich, als Grundmenge für die Instanzen $\mathbb{Z}_M$ heranzuziehen[2].
Nun sollen die modulare und die nicht-modulare Version algorithmisch in Beziehung gesetzt werden. Ein Algorithmus für die modulare Version kann direkt zur Lösung nicht-modularer Instanzen verwendet werden, indem M auf $\sum_{i=1}^{n} a_i + 1$ gesetzt wird. Offensichtlich kann das Bilden des Teilungsrestes modulo M das Ergebnis nicht verändern. Umgekehrt können wir nicht-modulare Algorithmen verwenden, um modulare Instanzen zu lösen. Der gegebene Algorithmus soll als Blackbox verwendet und nicht selbst verändert werden. Zunächst wird jede Zahl der Instanz jeweils auf Teilungsrest modulo M gesetzt. Nun gilt für eine Lösung $\mathbf{x}$, dass aus $\mathbf{ax} \equiv c \pmod{M}$ folgt, dass es ein i gibt mit $0 \leq i < M$ und $\mathbf{ax} = c + iM$. Der Algorithmus muss also n mal mit verschiedenen Zielgewichten in Abhängigkeit von i aufgerufen werden. Angenommen, dieser Algorithmus hat eine Laufzeit von $\mathcal{O}(f(n))$, so hat die modulare Version hiervon eine Laufzeit von $\mathcal{O}(nf(n))$.
Hat f die Form $f(n) = n2^{\gamma n}$ mit $\gamma \in \mathbb{R}$, so motiviert diese Beobachtung die Verwendung modifizierter Landausymbole. In dem vorliegenden Fall ist es praktisch, den linearen Term n unter den Tisch fallen lassen zu können, schließlich gilt für alle $\epsilon > 0$, dass $n2^{\gamma n} \in \mathcal{O}(2^{(\gamma+\epsilon)n})$. Dies wird mit der sogenannten **Soft-$\mathcal{O}$-Notation** erreicht. $f \in \tilde{\mathcal{O}}(g)$ ist hierbei eine Abkürzung dafür, dass ein $k \in \mathbb{N}$ existiert mit $f \in \mathcal{O}(g \log^k g)$. Für die vorangehende algorithmische Konstruktion gilt also die Laufzeit von $\tilde{\mathcal{O}}(f(x))$, wenn f eine exponentielle Funktion ist.

Angenommen, zum Berechnen steht a priori Wissen zur Verfügung, wie viele Elemente die Lösung enthalten wird, also die **Balance**; kann hieraus eine Verbesserung erzielt werden? Wir definieren:

$$\alpha := \frac{|\mathbf{x}|}{n}$$

Dabei ist $|\mathbf{x}| := \sum_{i=1}^{n} x_i$. Ziehen wir den naiven Algorithmus heran, so lässt sich dieser beschleunigen, wenn α bekannt ist. Es müssen nur potentielle Lösungen betrachtet werden, die αn Gewichte involvieren. Die Größe des Suchraums wird damit auf $\binom{n}{\alpha n}$ eingeschränkt. Das asymptotische Verhalten dieses Ausdrucks wird im Anhang A.5 untersucht. Dort findet sich auch eine Tabelle für einige Werte für α. Wie man sieht, verändert sich für $\alpha = 1/2$ die Laufzeit nicht. Für kleinere Werte nimmt die Laufzeit exponentiell ab. Für größere gilt dies auch, jedoch ist es bequem anzunehmen, dass $\alpha \leq 1/2$. Durch Hinzuziehen des Komplementärproblemes ist dies einsichtig. Hat eine Instanz eine Lösung, die mehr als die Hälfte der Gewichte einbezieht, so hat die komplementäre Instanz $(\mathbf{a}, \sum_{i=1}^{n} a_i - c)$ die invertierte Lösung.

[2] $\mathbb{Z}_M := \mathbb{Z}/M\mathbb{Z}$

Es ist keine Einschränkung, das Wissen um α für die Algorithmen vorauszusetzen, da es auch hier möglich ist, einen solchen Algorithmus in einen größeren einzubetten, der ihn für alle sinnvollen[3] Werte von α aufruft. Diese Konstruktion hat keine größere Laufzeit, wie anhand des naiven Algorithmus einsichtig: $\tilde{\mathcal{O}}(\sum_{i=1}^{n/2} \binom{n}{i}) \subseteq \tilde{\mathcal{O}}(2^n)$. Auch bei Algorithmen mit exponentiellem Platzverbrauch dominiert der Fall $\alpha = 1/2$. Im Folgenden werden die Algorithmen für den Worst-Case $\alpha = 1/2$ entworfen und analysiert. Für kleinere Werte ergibt sich jeweils eine exponentielle Beschleunigung (analog für Platz). Sei zum Beispiel $\alpha = 1/8$, so wird die Komplexität für den naiven Algorithmus von $\tilde{\mathcal{O}}(2^n)$ auf $\tilde{\mathcal{O}}(2^{0{,}544n})$ verringert.

Wie in der Einleitung bereits erwähnt, lassen sich Rucksackinstanzen danach klassifizieren, wie schwer sie zu lösen sind. Hierfür definieren wir eine Kennzahl von Rucksackinstanzen, die sogenannte **Dichte**:

$$d := \frac{n}{\log_2 \max \bigcup_{i=1}^{n} \{a_i\}}$$

Die Dichte setzt die Anzahl der Gewichte mit der Größe des größten Gewichts in Relation. Es hat sich herausgestellt, dass diese Kennzahl sich eignet, eine Obermenge der wirklich schwer lösbaren Instanzen zu bestimmen. Wir sprechen von einer schweren Instanz, wenn $0{,}94 < d \leq 1$, ansonsten von einer leichten Instanz. Hierbei ist auf die Terminologie zu achten: Die Klasse der schweren Instanzen beinhaltet alle Instanzen, die nicht effizient lösbar sind, aber nicht jede schwere Instanz ist tatsächlich schwer zu lösen. Für leichte Instanzen gilt generell, dass sie wesentlich effizienter gelöst werden können [CJL$^+$92].

Schwere Instanzen. Für Instanzen mit $0{,}94 < d \leq 1$ sind im Wesentlichen nur Algorithmen bekannt, wie sie in dieser Arbeit behandelt werden. $d = 1$ ist dabei bewiesenermaßen der schwerste Fall [IN96]. Da es im Allgemeinen also sehr schwer ist, Instanzen dieser Klasse zu lösen, hat dieser Umstand Ansätze hervorgebracht, die Subset Sum in der Kryptographie einsetzen; zum Beispiel das Kryptosystem von Merkle und Hellman. Auch in der Kryptoanalyse, zum Beispiel bei NTRU, treten schwere Rucksackprobleme auf. Schnellere Algorithmen könnten längere Schlüssel für gewisse Kryptosysteme erforderlich machen.
Es sei noch einmal darauf hingewiesen, dass nicht jede Instanz mit $d = 1$ schwer zu lösen ist.

Beispiel 2.5 *Wir betrachten eine stark wachsende Folge. Eine (sortierte) Folge $(a)_i$ heißt stark wachsend, wenn für alle k gilt, dass $a_k > \sum_{i=1}^{k-1} a_i$. Wählen wir eine Folge, bei der die n-te Zahl n Bits lang ist und interpretieren die ersten n Zahlen dieser Folge als Instanz, so hat diese die Dichte $d = 1$ und kann trotzdem in Linearzeit gelöst werden. Hierzu wird* **a** *absteigend durchlaufen und wenn immer das betrachtete Gewicht in das verbleibende Zielgewicht passt, so wird es zur Lösung hinzugenommen und vom verbleibenden Zielgewicht abgezogen. Es kann hierbei nicht auftreten, dass das betrachtete Gewicht nicht zur Lösung gehört, da alle restlichen Gewichte zusammen das verbleibende Zielgewicht nicht auffüllen können.*

[3] Die Schritte sollten so groß sein, dass sich αn jeweils genau um 1 vergrößert.

Die Länge der Eingabe ist normalerweise der Parameter, in deren Abhängigkeit Komplexitäten angegeben werden. Ist $d = 1$, so ist die Länge der Eingabe quadratisch in n. Die Komplexitäten beziehen sich jedoch weiterhin auf n. Der lineare Faktor, der durch die Anzahl der Bits der Gewichte entsteht, kann durch Soft-$\mathcal{O}$-Notation jedoch vernachlässigt werden.

3. Eine Heuristik

Bis auf die beiden ältesten Algorithmen verwenden alle in dieser Arbeit behandelten eine heuristische Annahme, die sich für die meisten Fälle auch beweisen lässt. Die Beweise sollen in diesem Kapitel dargestellt werden. Bei der Analyse der Algorithmen wird mehrfach auf diesen Abschnitt verwiesen werden.
Informell lässt sich die Aussage der Heuristik so darstellen: Gegeben sei eine endliche Menge von ganzen Zahlen, A, und ein Modulus M. Wenn A_R die Teilmenge von A ist, die alle Elemente, die kongruent zu R modulo M sind, enthält, so gehen wir davon aus, dass für alle R gilt, dass $|A_R| \approx \frac{|A|}{M}$. Offensichtlich gilt $\sum_{R=0}^{M-1} |A_R| = |A|$. Die Heuristik besagt nun, dass sich die Elemente von A ungefähr gleichmäßig über alle R verteilen.
Diese leicht verständliche Aussage wollen wir auf das Rucksackproblem anpassen. Im Weiteren Verlauf sei $\mathbf{a} \in \mathbb{Z}_M^n$ und $c \in \mathbb{Z}_M^\delta$. Zunächst wenden wir uns dem Fall $\delta = 1$ zu, womit $c \in \mathbb{Z}_M$ ist.

3.1. Erste Variante

Die hier vorgestellten Beweise basieren auf [NSS00] und [HGJ10].
Wenden wir ein $\mathbf{x} \in \mathcal{B} \subseteq \mathbb{Z}_M^n$ auf $\mathbf{a}$ an[1], so ist $\mathbf{ax} \in \mathbb{Z}_M$. Man kann alle $\mathbf{x}$, die dasselbe Ergebnis liefern, zusammenfassen und zählen:

$$N_{\mathbf{a}}(c, \mathcal{B}) := |\{\mathbf{x} \in \mathcal{B} | \mathbf{ax} = c\}|$$

Teilt man nun durch $|\mathcal{B}|$, so erhält man die Wahrscheinlichkeit dafür, dass ein zufälliges $\mathbf{x} \in \mathcal{B}$ gilt, dass $\mathbf{ax} = c$.

$$P_{\mathbf{a}}(c, \mathcal{B}) := \frac{N_{\mathbf{a}}(c, \mathcal{B})}{|\mathcal{B}|}$$

[1] Typischerweise wird $\mathcal{B} \subseteq \mathbb{B}^n$ sein, jedoch werden die Beweise allgemein durchgeführt.

Das Problem soll also stochastisch angegangen werden. Stochastische Grundlagen werden im Anhang A.3 beschrieben.
Es sei $(\Omega, \mathcal{F}, P)$ ein Wahrscheinlichkeitsraum mit

$$\begin{aligned}\Omega &:= \{P_{\mathbf{a}}(c, \mathcal{B}) \mid \mathbf{a} \in \mathbb{Z}_M^n, c \in \mathbb{Z}_M\} \\ \mathcal{F} &:= 2^\Omega \\ P\colon \mathcal{F} &\to [0,1] \text{ mit} \\ \{\omega\} &\mapsto \frac{1}{M^n} \sum_{\mathbf{a} \in \mathbb{Z}_M^n} \frac{|\{c \in \mathbb{Z}_M | P_{\mathbf{a}}(c, \mathcal{B}) = \omega\}|}{M}\end{aligned}$$

Diese Wahrscheinlichkeitsverteilung gibt für ein ω die Wahrscheinlichkeit dafür an, dass für zufälliges $\mathbf{a}$ und c der Anteil der $\mathbf{x}$, für die $\mathbf{ax} = c$ gilt, gleich ω ist. Als Zufallsvariable wählen wir

$$X := id_\Omega.$$

Wir interessieren uns in der Anwendung der Heuristik für Ausdrücke wie $P(X > \lambda)$; das ist die Wahrscheinlichkeit für zufälliges $\mathbf{a}$ und c, dass es mehr als $\lambda|\mathcal{B}|$ Elemente $\mathbf{x}$ gibt, die $\mathbf{ax} = c$ erfüllen. Auf dem Weg dorthin beginnen wir mit dem Erwartungswert von X.

Lemma 3.1 *Es gilt* $\mathbb{E}(X) = 1/M$.

Beweis.

$$\mathbb{E}(X) = \sum_{\omega \in X(\Omega)} \omega P(\omega) = \sum_{\omega \in \Omega} \omega P(\omega)$$

Um über alle Ereignisse zu summieren, kann über alle $\mathbf{a}$ und c summiert werden, wobei zu beachten ist, dass dadurch Ereignisse mehrfach auftreten können:

$$\sum_{\omega \in \Omega} \omega P(\omega) = \sum_{\mathbf{a} \in \mathbb{Z}_M^n} \sum_{c \in \mathbb{Z}_M} \frac{P_{\mathbf{a}}(c, \mathcal{B}) P(P_{\mathbf{a}}(c, \mathcal{B}))}{|\{(\mathbf{a}', c') \in \mathbb{Z}_M^n \times \mathbb{Z}_M | P_{\mathbf{a}}(c, \mathcal{B}) = P_{\mathbf{a}'}(c', \mathcal{B})\}|}$$

Es gilt, dass

$$|\{(\mathbf{a}', c') \in \mathbb{Z}_M^n \times \mathbb{Z}_M | P_{\mathbf{a}}(c, \mathcal{B}) = P_{\mathbf{a}'}(c', \mathcal{B})\}| = M^n M P(P_{\mathbf{a}}(c, \mathcal{B})),$$

womit folgt

$$\mathbb{E}(X) = \frac{1}{M^n} \frac{1}{M} \sum_{\mathbf{a} \in \mathbb{Z}_M^n} \sum_{c \in \mathbb{Z}_M} P_{\mathbf{a}}(c, \mathcal{B}) = \frac{1}{M^n} \frac{1}{M} \sum_{\mathbf{a} \in \mathbb{Z}_M^n} 1 = \frac{1}{M}$$

□

Als nächstes soll die Varianz von X berechnet werden. Im Ansatz können wir die Aufsummierung über alle Ereignisse analog bewerkstelligen wie beim Erwartungswert:

$$\begin{aligned}\mathbb{V}(X) &= \sum_{\omega \in X(\Omega)} (\omega - \mathbb{E}(X))^2 P(\omega) \\ &= \sum_{\omega \in \Omega} (\omega - 1/M)^2 P(\omega) \\ &= \frac{1}{M^n} \frac{1}{M} \sum_{\mathbf{a} \in \mathbb{Z}_M^n} \sum_{c \in \mathbb{Z}_M} (P_{\mathbf{a}}(c, \mathcal{B}) - 1/M)^2\end{aligned}$$

In [NSS00] wird die Berechnung dieses Ausdrucks gezeigt; hier soll diese Rechnung ausführlich dargestellt werden. Hierfür ist Vorarbeit nötig:
Wir definieren eine Abbildung e_M, die uns M-te Einheitswurzeln liefert:

$$\begin{aligned} e_M \colon \mathbb{Z} &\to \mathbb{C} \\ z &\mapsto \mathrm{e}^{\frac{2\pi i z}{M}} \end{aligned}$$

Da für alle $k, z \in \mathbb{Z}$ gilt, dass $e_M(z) = e_M(z + kM)$, können wir e_M erweitern auf $e_M \colon \mathbb{Z}_M \to \mathbb{C}$.

Lemma 3.2 *Für $u \in \mathbb{Z}$ gilt*

$$\sum_{\lambda=0}^{M-1} e_M(\lambda u) = \begin{cases} 0, & \text{wenn } u \not\equiv 0 \pmod{M} \\ M, & \text{wenn } u \equiv 0 \pmod{M} \end{cases}$$

Beweis. Wenn $u \equiv 0 \pmod{M}$, dann sei $u = kM$.

$$\sum_{\lambda=0}^{M-1} e_M(\lambda u) = \sum_{\lambda=0}^{M-1} e_M(\lambda k M) = \sum_{\lambda=0}^{M-1} \mathrm{e}^{\frac{2\pi i \lambda k M}{M}} = \sum_{\lambda=0}^{M-1} \mathrm{e}^{2\pi i \lambda k} = \sum_{\lambda=0}^{M-1} 1^{\lambda k} = M$$

Wenn $u \not\equiv 0 \pmod{M}$, dann sei $u = u' + kM$ mit $0 < u' < M$.

$$\sum_{\lambda=0}^{M-1} e_M(\lambda u) = \sum_{\lambda=0}^{M-1} e_M(\lambda(u' + kM)) = \sum_{\lambda=0}^{M-1} \mathrm{e}^{\frac{2\pi i \lambda (u' + kM)}{M}} = \sum_{\lambda=0}^{M-1} \mathrm{e}^{\frac{2\pi i \lambda u'}{M}} = \mathrm{e}^{\frac{u'}{M}} \sum_{\lambda=0}^{M-1} \mathrm{e}^{\frac{2\pi i \lambda}{M}} = 0$$

Damit sind beide Fälle gezeigt. □

Dieses Lemma sowie das folgende wird für die Berechnung der Varianz benötigt.

Lemma 3.3 *Für alle $\lambda \not\equiv 0 \pmod{M}$ gilt*

$$\sum_{\mathbf{a} \in \mathbb{Z}_M^n} \left| \sum_{\mathbf{x} \in \mathcal{B}} e_M(\lambda \mathbf{a} \mathbf{x}) \right|^2 = M^n |\mathcal{B}|$$

Beweis. Von links ausgehend formen wir um:

$$\begin{aligned}
&\sum_{\mathbf{a}\in\mathbb{Z}_M^n}\left|\sum_{\mathbf{x}\in\mathcal{B}} e_M(\lambda\mathbf{ax})\right|^2 \\
&\quad= \sum_{\mathbf{a}\in\mathbb{Z}_M^n}\left(\mathrm{Re}\left(\sum_{\mathbf{x}\in\mathcal{B}} e_M(\lambda\mathbf{ax})\right)^2 + \mathrm{Im}\left(\sum_{\mathbf{x}\in\mathcal{B}} e_M(\lambda\mathbf{ax})\right)^2\right) \\
&\quad= \sum_{\mathbf{a}\in\mathbb{Z}_M^n}\left(\left(\sum_{\mathbf{x}\in\mathcal{B}}\mathrm{Re}(e_M(\lambda\mathbf{ax}))\right)^2 + \left(\sum_{\mathbf{x}\in\mathcal{B}}\mathrm{Im}(e_M(\lambda\mathbf{ax}))\right)^2\right) \\
&\quad= \sum_{\mathbf{a}\in\mathbb{Z}_M^n}\left(\left(\sum_{\mathbf{x}\in\mathcal{B}}\sum_{\mathbf{y}\in\mathcal{B}}\mathrm{Re}(e_M(\lambda(\mathbf{ax}+\mathbf{ay})))\right) + \left(\sum_{\mathbf{x}\in\mathcal{B}}\sum_{\mathbf{y}\in\mathcal{B}}\mathrm{Im}(e_M(\lambda(\mathbf{ax}+\mathbf{ay})))\right)\right) \\
&\quad= \sum_{\mathbf{a}\in\mathbb{Z}_M^n}\sum_{\mathbf{x}\in\mathcal{B}}\sum_{\mathbf{y}\in\mathcal{B}}(\mathrm{Re}(e_M(\lambda(\mathbf{ax}+\mathbf{ay}))) + \mathrm{Im}(e_M(\lambda(\mathbf{ax}+\mathbf{ay}))))
\end{aligned}$$

Betrachtet man $\sum_{\mathbf{a}\in\mathbb{Z}_M^n} e_M(\lambda(\mathbf{ax}+\mathbf{ay}))$, so fällt auf, dass keine Zahlen mit komplexem Anteil entstehen können:

$$\sum_{\mathbf{a}\in\mathbb{Z}_M^n}\sum_{\mathbf{x}\in\mathcal{B}}\sum_{\mathbf{y}\in\mathcal{B}}(\mathrm{Re}(e_M(\lambda(\mathbf{ax}+\mathbf{ay}))) + \mathrm{Im}(e_M(\lambda(\mathbf{ax}+\mathbf{ay})))) = \sum_{\mathbf{a}\in\mathbb{Z}_M^n}\sum_{\mathbf{x}\in\mathcal{B}}\sum_{\mathbf{y}\in\mathcal{B}} e_M(\lambda(\mathbf{ax}+\mathbf{ay}))$$

Da in Einheitswurzeln modulo M gerechnet werden kann, kann in der Summe über alle $\mathbf{a}$ gefolgert werden dass:

$$\sum_{\mathbf{x}\in\mathcal{B}}\sum_{\mathbf{y}\in\mathcal{B}}\sum_{\mathbf{a}\in\mathbb{Z}_M^n} e_M(\lambda(\mathbf{ax}+\mathbf{ay})) = \sum_{\mathbf{x}\in\mathcal{B}}\sum_{\mathbf{y}\in\mathcal{B}}\sum_{\mathbf{a}\in\mathbb{Z}_M^n} e_M(\lambda(\mathbf{ax}-\mathbf{ay}))$$

Betrachtet man nun die innerste Summe, so gibt es, abhängig von den beiden vorhergehenden Summen, zwei Fälle.

Erster Fall: $\mathbf{x} \neq \mathbf{y}$
In diesem Fall sei ohne Einschränkung $x_n \neq y_n$. Dann gilt

$$\sum_{\mathbf{a}\in\mathbb{Z}_M^n} e_M(\lambda(\mathbf{ax}-\mathbf{ay})) = \sum_{\mathbf{a}\in\mathbb{Z}_M^{n-1}}\sum_{a_n=0}^{M-1} e_M\left(\lambda\left(\sum_{i=1}^{n-1} a_i x_i\right) - \left(\sum_{i=1}^{n-1} a_i y_i\right)\right) e_M(\lambda a_n(x_n - y_n))$$

Nach Hineinziehen der Summe über a_n und da $\lambda(x_n - y_n) \not\equiv 0 \pmod{M}$ und mit Lemma 3.2 gilt nun

$$\sum_{\mathbf{a}\in\mathbb{Z}_M^{n-1}} e_M\left(\lambda\left(\sum_{i=1}^{n-1} a_i x_i\right) - \left(\sum_{i=1}^{n-1} a_i y_i\right)\right) \cdot \sum_{a_n=0}^{M-1} e_M(\lambda a_n(x_n - y_n)) = 0$$

Das heißt also, dass in jedem Fall, in dem $\mathbf{x} \neq \mathbf{y}$ gilt, dieser nichts zur Gesamtsumme beiträgt.

Zweiter Fall: $\mathbf{x} = \mathbf{y}$
Hier gilt

$$\sum_{\mathbf{a}\in\mathbb{Z}_M^n} e_M(0) = M^n$$

Da es $|\mathcal{B}|$ viele Fälle gibt, in denen $\mathbf{x} = \mathbf{y}$ gilt, folgt das Ergebnis:

$$\sum_{\mathbf{x}\in\mathcal{B}}\sum_{\mathbf{y}\in\mathcal{B}}\sum_{\mathbf{a}\in\mathbb{Z}_M^n} e_M(\lambda(\mathbf{ax}-\mathbf{ay})) = M^n|\mathcal{B}|$$

□

Die Varianz lässt sich jetzt berechnen.

Satz 3.4 *Es gilt*

$$\mathbb{V}(X) = \frac{1}{M^n}\frac{1}{M}\sum_{\mathbf{a}\in\mathbb{Z}_M^n}\sum_{c\in\mathbb{Z}_M}(P_{\mathbf{a}}(c,\mathcal{B}) - 1/M)^2 = \frac{M-1}{M^2|\mathcal{B}|}$$

Beweis. Die erste Gleichheit wurde bereits gezeigt. Mit Lemma 3.2 gilt:

$$\begin{aligned} N_{\mathbf{a}}(c,\mathcal{B}) &= \frac{1}{M}\sum_{\mathbf{x}\in\mathcal{B}}\sum_{\lambda=0}^{M-1} e_M(\lambda(\mathbf{ax}-c)) \\ &= \frac{1}{M}\sum_{\mathbf{x}\in\mathcal{B}}\sum_{\lambda=1}^{M-1} e_M(\lambda(\mathbf{ax}-c)) + \frac{|\mathcal{B}|}{M} \end{aligned}$$

Zunächst wollen wir diesen Ausdruck berechnen:

$$\sum_{c\in\mathbb{Z}_M}\left(N_{\mathbf{a}}(c,\mathcal{B}) - \frac{|\mathcal{B}|}{M}\right)^2$$

Durch Anwenden der vorhergehenden Gleichung erhält man

$$\begin{aligned} \sum_{c\in\mathbb{Z}_M}\left(N_{\mathbf{a}}(c,\mathcal{B}) - \frac{|\mathcal{B}|}{M}\right)^2 &= \sum_{c\in\mathbb{Z}_M}\left(\frac{1}{M}\sum_{\mathbf{x}\in\mathcal{B}}\sum_{\lambda=1}^{M-1} e_M(\lambda(\mathbf{ax}-c))\right)^2 \\ &= \sum_{c\in\mathbb{Z}_M}\left(\frac{1}{M}\sum_{\lambda=1}^{M-1} e_M(-\lambda c)\sum_{\mathbf{x}\in\mathcal{B}} e_M(\lambda\mathbf{ax})\right)^2 \end{aligned}$$

Durch die Fallunterscheidung, wie sie im vorhergehenden Beweis verwendet wurde, lässt sich weiter umformen:

$$\begin{aligned}
&\sum_{c\in\mathbb{Z}_M}\left(\frac{1}{M}\sum_{\lambda=1}^{M-1} e_M(-\lambda c)\sum_{\mathbf{x}\in\mathcal{B}} e_M(\lambda\mathbf{a}\mathbf{x})\right)^2\\
&= \frac{1}{M^2}\sum_{c\in\mathbb{Z}_M}\sum_{\lambda,\eta=1}^{M-1} e_M(-c(\eta+\lambda))\sum_{\mathbf{x},\mathbf{y}\in\mathcal{B}} e_M(\lambda\mathbf{a}\mathbf{x}+\eta\mathbf{a}\mathbf{y})\\
&= \frac{1}{M^2}\sum_{\lambda,\eta=1}^{M-1}\left(\sum_{\mathbf{x},\mathbf{y}\in\mathcal{B}} e_M(\lambda\mathbf{a}\mathbf{x}+\eta\mathbf{a}\mathbf{y})\sum_{c\in\mathbb{Z}_M} e_M(-\eta c-\lambda c)\right)
\end{aligned}$$

Für die innere Summe gibt es nun nach Lemma 3.2 zwei Fälle. Ist $\lambda+\eta\not\equiv 0 \pmod M$, so ist $\sum_{c\in\mathbb{Z}_M} e_M(-\eta c-\lambda c)=0$. Ist $\lambda+\eta\equiv 0 \pmod M$, so ist $\sum_{c\in\mathbb{Z}_M} e_M(-\eta c-\lambda c)=M$. Hieraus ergibt sich für den Ausdruck

$$\frac{1}{M^2}\sum_{\lambda,\eta=1}^{M-1}\left(\sum_{\mathbf{x},\mathbf{y}\in\mathcal{B}} e_M(\lambda\mathbf{a}\mathbf{x}+\eta\mathbf{a}\mathbf{y})\sum_{c\in\mathbb{Z}_M} e_M(-\eta c-\lambda c)\right)=\frac{1}{M}\sum_{\lambda=1}^{M-1}\sum_{\mathbf{x},\mathbf{y}\in\mathcal{B}} e_M(\lambda(\mathbf{a}\mathbf{x}-\mathbf{a}\mathbf{y}))$$

Dieser Ausdruck kann rückwärts analog wie in Lemma 3.3 umgeformt werden:

$$\sum_{c\in\mathbb{Z}_M}\left(N_{\mathbf{a}}(c,\mathcal{B})-\frac{|\mathcal{B}|}{M}\right)^2=\frac{1}{M}\sum_{\lambda=1}^{M-1}\left|\sum_{\mathbf{x}\in\mathcal{B}} e_M(\lambda\mathbf{a}\mathbf{x})\right|^2.$$

Jetzt kann der Ursprungsausdruck für die Varianz berechnet werden.

$$\begin{aligned}
\frac{1}{M^n}\frac{1}{M}\sum_{\mathbf{a}\in\mathbb{Z}_M^n}\sum_{c\in\mathbb{Z}_M}\left(P_{\mathbf{a}}(c,\mathcal{B})-\frac{1}{M}\right)^2 &= \frac{1}{M^n}\frac{1}{M}\frac{1}{|\mathcal{B}|^2}\sum_{\mathbf{a}\in\mathbb{Z}_M^n}\sum_{c\in\mathbb{Z}_M}\left(N_{\mathbf{a}}(c,\mathcal{B})-\frac{|\mathcal{B}|}{M}\right)^2\\
&= \frac{1}{M^n}\frac{1}{M}\frac{1}{|\mathcal{B}|^2}\sum_{\mathbf{a}\in\mathbb{Z}_M^n}\frac{1}{M}\sum_{\lambda=1}^{M-1}\left|\sum_{\mathbf{x}\in\mathcal{B}} e_M(\lambda\mathbf{a}\mathbf{x})\right|^2
\end{aligned}$$

Nach Lemma 3.3 kann der Ausdruck zu diesem umgeformt werden:

$$\frac{1}{M^n}\frac{1}{M}\frac{1}{|\mathcal{B}|^2}\frac{1}{M}\sum_{\lambda=1}^{M-1} M^n|\mathcal{B}|=\frac{M-1}{M^2|\mathcal{B}|}=\mathbb{V}(X)$$

□

Mit der Varianz können nun stochastische Aussagen bewiesen werden, wie sie zur Analyse der Algorithmen benötigt werden.

Satz 3.5 *Für $\mu>0$ gilt*

$$P\left(\left|X-\frac{1}{M}\right|\geq\frac{\mu}{M}\right)\leq\frac{M}{|\mathcal{B}|\mu^2}.$$

Beweis. Es lässt sich die Tschebyschow-Ungleichung anwenden:

$$P\left(\left|X - \frac{1}{M}\right| \geq \frac{\mu}{M}\right) \leq \frac{\mathbb{V}(X)}{\left(\frac{\mu}{M}\right)^2} \leq \frac{M}{|\mathcal{B}|\mu^2}.$$

□

Der Ausdruck $P\left(\left|X - \frac{1}{M}\right| \geq \frac{\mu}{M}\right)$ gibt für zufälliges $\mathbf{a} \in \mathbb{Z}_M^n$ und $c \in \mathbb{Z}_M$ die Wahrscheinlichkeit an, dass $|P_{\mathbf{a}}(c, \mathcal{B}) - 1/M| \geq \mu/M$. Der vorhergehende Satz liefert eine obere Schranke für diese Wahrscheinlichkeit.

Satz 3.6 *Die Wahrscheinlichkeit, dass es für ein zufälliges $\mathbf{a} \in \mathbb{Z}_M^n$ mindestens ein $c \in \mathbb{Z}_M$ gibt, das $|P_{\mathbf{a}}(c, \mathcal{B}) - 1/M| \geq \mu/M$ erfüllt, beträgt höchstens $\frac{M^2}{|\mathcal{B}|\mu^2}$.*

Dieser Satz lässt sich noch allgemeiner fassen. Setzt man im folgenden Satz $\lambda = 1/M$, so erhält man den vorhergehenden.

Satz 3.7 *Es sind $\mu > 0$ und $0 < \lambda < 1$. Die Wahrscheinlichkeit, dass für ein zufälliges $\mathbf{a} \in \mathbb{Z}_M^n$ es mindestens λM viele $c \in \mathbb{Z}_M$ gibt, die $|P_{\mathbf{a}}(c, \mathcal{B}) - 1/M| \geq \mu/M$ erfüllen, beträgt höchstens $\frac{M}{\lambda|\mathcal{B}|\mu^2}$.*

Beweis. Es ist

$$P\left(\left|X - \frac{1}{M}\right| \geq \frac{\mu}{M}\right) = \frac{1}{M}\frac{1}{M^n}\left|\left\{(\mathbf{a}, c) \in \mathbb{Z}_M^n \times \mathbb{Z}_M \middle| \left|P_{\mathbf{a}}(c, \mathcal{B}) - \frac{1}{M}\right| \geq \frac{\mu}{M}\right\}\right|$$

Es sei $\chi\colon \mathbb{Z}_M^n \times \mathbb{Z}_M \to \mathbb{B}$ die charakteristische Funktion der Menge $\left\{(\mathbf{a}, c) \in \mathbb{Z}_M^n \times \mathbb{Z}_M \middle| \left|P_{\mathbf{a}}(c, \mathcal{B}) - \frac{1}{M}\right| \geq \frac{\mu}{M}\right\}$. Damit ist

$$P\left(\left|X - \frac{1}{M}\right| \geq \frac{\mu}{M}\right) = \frac{1}{M}\sum_{c \in \mathbb{Z}_M}\sum_{\mathbf{a} \in \mathbb{Z}_M^n}\frac{\chi(\mathbf{a}, c)}{M^n} = \frac{1}{M}\left(\sum_{\mathbf{a} \in \mathbb{Z}_M^n}\frac{\chi(\mathbf{a}, \pi(1))}{M^n} + \ldots + \sum_{\mathbf{a} \in \mathbb{Z}_M^n}\frac{\chi(\mathbf{a}, \pi(M))}{M^n}\right).$$

Dabei ist π eine Permutation auf $\mathbb{Z}_M$, die eine Sortierung anhand des Termes $\sum_{\mathbf{a} \in \mathbb{Z}_M^n}\frac{\chi(\mathbf{a}, \pi(k))}{M^n}$ liefert. Es gilt also für alle k, dass

$$\sum_{\mathbf{a} \in \mathbb{Z}_M^n}\frac{\chi(\mathbf{a}, \pi(k))}{M^n} \geq \sum_{\mathbf{a} \in \mathbb{Z}_M^n}\frac{\chi(\mathbf{a}, \pi(k+1))}{M^n}.$$

Die Wahrscheinlichkeit, für ein zufälliges $\mathbf{a} \in \mathbb{Z}_M^n$ mindestens λM viele $c \in \mathbb{Z}_M$ zu finden, ist

$$\begin{aligned}\sum_{\mathbf{a}\in\mathbb{Z}_M^n} \frac{\chi(\mathbf{a},\pi(\lambda M))}{M^n} &\leq \frac{\sum_{\mathbf{a}\in\mathbb{Z}_M^n} \frac{\chi(\mathbf{a},\pi(1))}{M^n}}{\lambda M} + \ldots + \frac{\sum_{\mathbf{a}\in\mathbb{Z}_M^n} \frac{\chi(\mathbf{a},\pi(\lambda M))}{M^n}}{\lambda M} \\ &\leq \sum_{k=1}^{M} \frac{\sum_{\mathbf{a}\in\mathbb{Z}_M^n} \frac{\chi(\mathbf{a},\pi(k))}{M^n}}{\lambda M} \\ &= \lambda P\left(\left|X - \frac{1}{M}\right| \geq \frac{\mu}{M}\right) \\ &\leq \lambda \frac{M}{|\mathcal{B}|\mu^2} = \frac{M}{\lambda|\mathcal{B}|\mu^2}\end{aligned}$$

Dies ist die gewünschte obere Schranke. □

3.2. Zweite Variante

Im weiteren Verlauf wird eine Variante der bisherigen Ergebnisse benötigt. Bisher war $\mathcal{B}$ eine beliebige Teilmenge aus $\mathbb{Z}_M^n$, jetzt soll jedoch $\mathcal{B} \subseteq (\mathbb{B}^n)^\delta$ sein. Wir stellen weitere Bedingungen an $\mathcal{B}$: für jedes $\mathbf{x} = (\mathbf{x}_i, \ldots, \mathbf{x}_\delta) \in \mathcal{B}$ soll gelten, dass für alle $i \in \{1, \ldots, \delta\}$ gilt, dass $|\mathbf{x}_i| = n/(\delta+1)$ ist. Außerdem soll für alle $t \in \{1, \ldots, n\}$ und $i, j \in \{1, \ldots, \delta\}$ gelten, dass $x_{i,t} \neq x_{j,t}$. Wie man sieht, soll $\mathcal{B}$ also alle möglichen Aufteilungen von $\mathbb{B}^n$ in $\delta+1$ Teile enthalten, wobei das letzte Teil ausgelassen wird, da es sich implizit ergibt. Die Teile sind alle gleich groß und überlappungsfrei. Es ist $|\mathcal{B}| = \binom{n}{n/4}\binom{3n/4}{n/4}\binom{n/2}{n/4} \approx 2^{2n}$. In [HGJ10] wird der Beweis für den Spezialfall $\delta = 3$ skizziert. Wir wollen hier nun so weit als möglich unabhängig von δ argumentieren. Der Aufbau dieses Abschnitts ähnelt stark dem des letzten. Die Beweise verlaufen analog, sind jedoch in der einfacheren Version des letzten Abschnitts verständlicher. Aus diesem Grund werden beide Fälle ausführlich behandelt.
Zunächst definieren wir wieder den Wahrscheinlichkeitsraum. Dazu wird

$$N_{\mathbf{a}}(\mathbf{c}, \mathcal{B}) := \left|\left\{\mathbf{x} \in \mathcal{B} \,\middle|\, \bigwedge_{i=1}^{\delta} \mathbf{a}\mathbf{x}_i = c_i\right\}\right|$$

und

$$P_{\mathbf{a}}(\mathbf{c}, \mathcal{B}) := \frac{N_{\mathbf{a}}(\mathbf{c}, \mathcal{B})}{|\mathcal{B}|}$$

benötigt. $(\Omega, \mathcal{F}, P)$ sei der Wahrscheinlichkeitsraum mit

$$\begin{aligned}\Omega &:= \{P_{\mathbf{a}}(\mathbf{c}, \mathcal{B}) \mid \mathbf{a} \in \mathbb{Z}_M^n, \mathbf{c} \in \mathbb{Z}_M^\delta\} \\ \mathcal{F} &:= 2^\Omega \\ P\colon \mathcal{F} &\to [0,1] \text{ mit} \\ \{\omega\} &\mapsto \frac{1}{M^n} \sum_{\mathbf{a}\in\mathbb{Z}_M^n} \frac{|\{\mathbf{c} \in \mathbb{Z}_M^\delta | P_{\mathbf{a}}(\mathbf{c}, \mathcal{B}) = \omega\}|}{M^\delta}\end{aligned}$$

Diese Wahrscheinlichkeitsverteilung gibt - analog zum vorherigen Abschnitt - für ein ω die Wahrscheinlichkeit dafür an, dass für zufälliges $\mathbf{a}$ und $\mathbf{c}$ der Anteil der $\mathbf{x}$, für die für alle i gilt, dass $\mathbf{a}\mathbf{x}_i = c_i$ gleich ω ist. Als Zufallsvariable wählen wir

$$X := id_\Omega.$$

Es wird wieder der Erwartungswert von X benötigt.

Lemma 3.8 *Es gilt* $\mathbb{E}(X) = 1/M^\delta$.

Beweis.

$$\mathbb{E}(X) = \sum_{\omega \in X(\Omega)} \omega P(\omega) = \sum_{\omega \in \Omega} \omega P(\omega)$$

Um über alle Ereignisse zu summieren, kann über alle $\mathbf{a}$ und $\mathbf{c}$ summiert werden:

$$\sum_{\omega \in \Omega} \omega P(\omega) = \sum_{\mathbf{a} \in \mathbb{Z}_M^n} \sum_{\mathbf{c} \in \mathbb{Z}_M^\delta} \frac{P_\mathbf{a}(\mathbf{c}, \mathcal{B}) P(P_\mathbf{a}(\mathbf{c}, \mathcal{B}))}{|\{(\mathbf{a}', \mathbf{c}') \in \mathbb{Z}_M^n \times \mathbb{Z}_M^\delta | P_\mathbf{a}(\mathbf{c}, \mathcal{B}) = P_{\mathbf{a}'}(\mathbf{c}', \mathcal{B})\}|}$$

Es gilt, dass

$$|\{(\mathbf{a}', \mathbf{c}') \in \mathbb{Z}_M^n \times \mathbb{Z}_M^\delta | P_\mathbf{a}(\mathbf{c}, \mathcal{B}) = P_{\mathbf{a}'}(\mathbf{c}', \mathcal{B})\}| = M^n M^\delta P(P_\mathbf{a}(\mathbf{c}, \mathcal{B})),$$

woraus folgt

$$\mathbb{E}(X) = \frac{1}{M^n} \frac{1}{M^\delta} \sum_{\mathbf{a} \in \mathbb{Z}_M^n} \sum_{\mathbf{c} \in \mathbb{Z}_M^\delta} P_\mathbf{a}(\mathbf{c}, \mathcal{B}) = \frac{1}{M^n} \frac{1}{M^\delta} \sum_{\mathbf{a} \in \mathbb{Z}_M^n} 1 = \frac{1}{M^\delta}$$

□

Neben dem Erwartungswert wird auch die Varianz von X benötigt.

$$\begin{aligned}
\mathbb{V}(X) &= \sum_{\omega \in X(\Omega)} (\omega - \mathbb{E}(X))^2 P(\omega) \\
&= \sum_{\omega \in \Omega} (\omega - 1/M^\delta)^2 P(\omega) \\
&= \frac{1}{M^n} \frac{1}{M^\delta} \sum_{\mathbf{a} \in \mathbb{Z}_M^n} \sum_{\mathbf{c} \in \mathbb{Z}_M^\delta} (P_\mathbf{a}(\mathbf{c}, \mathcal{B}) - 1/M^\delta)^2
\end{aligned}$$

Um diesen Ausdruck zu berechnen, benötigen wir wieder ein Lemma. Dieses verwendet folgende Definition:

Definition 3.9 *Es sind* $\lambda, \lambda' \in \mathbb{Z}_M^\delta$

- $\lambda \sim \lambda' :\Leftrightarrow$ *Es existiert eine Permutation* $\sigma \colon \mathbb{Z}_M \to \mathbb{Z}_M$ *mit* $\sigma(0) = 0$ *und eine Permutation* $\tau \colon [1, \delta] \to [1, \delta]$, *sodass für alle* $i \in [1, \delta]$ *gilt:* $\lambda_i = \sigma(\lambda'_{\tau(i)})$
- $\mathbb{Z}_M^\delta / \sim$ *ist die Menge der Äquivalenzklassen von* $\sim$.

- *Die Funktion* $Y\colon \mathbb{Z}_M^\delta \to \mathbb{N}$ *definiert sich so:*

$$\lambda \mapsto \left| \left\{ y \in \mathcal{B} \middle| \sum_{\mathbf{a}\in\mathbb{Z}_M^n} e_M\left(\sum_{i=1}^{\delta} \lambda_i(\mathbf{a}\mathbf{x}_i - \mathbf{a}\mathbf{y}_i)\right) = M^n \right\} \right|,$$

  wobei $\mathbf{x}$ *beliebig aber fest ist.*

Wir erweitern Y zu $Y\colon \mathbb{Z}_M^\delta/\sim \to \mathbb{N}$ mit $\Lambda \mapsto Y(\lambda \in \Lambda)$. Da die Äquivalenzrelation so gewählt ist, dass für alle Elemente einer Klasse Y das gleiche Ergebnis liefert, ist Y auf $\mathbb{Z}_M^\delta/\sim$ wohldefiniert.

Lemma 3.10 *Für alle* $(\lambda_1, \ldots, \lambda_\delta) = \lambda \neq \mathbf{0}$ *gilt*

$$\sum_{\mathbf{a}\in\mathbb{Z}_M^n} \left| \sum_{\mathbf{x}\in\mathcal{B}} e_M\left(\sum_{i=1}^{\delta} \lambda_i \mathbf{a}\mathbf{x}_i\right)\right|^2 = M^n |\mathcal{B}| Y(\lambda)$$

Beweis. Von links ausgehend formen wir um:

$$\begin{aligned}
&\sum_{\mathbf{a}\in\mathbb{Z}_M^n} \left| \sum_{\mathbf{x}\in\mathcal{B}} e_M\left(\sum_{i=1}^{\delta} \lambda_i \mathbf{a}\mathbf{x}_i\right)\right|^2 \\
&\quad= \sum_{\mathbf{a}\in\mathbb{Z}_M^n} \left(\mathrm{Re}\left(\sum_{\mathbf{x}\in\mathcal{B}} e_M\left(\sum_{i=1}^{\delta} \lambda_i \mathbf{a}\mathbf{x}_i\right)\right)^2 + \mathrm{Im}\left(\sum_{\mathbf{x}\in\mathcal{B}} e_M\left(\sum_{i=1}^{\delta} \lambda_i \mathbf{a}\mathbf{x}_i\right)\right)^2 \right) \\
&\quad= \sum_{\mathbf{a}\in\mathbb{Z}_M^n} \left(\left(\sum_{\mathbf{x}\in\mathcal{B}} \mathrm{Re}\left(e_M\left(\sum_{i=1}^{\delta} \lambda_i \mathbf{a}\mathbf{x}_i\right)\right)\right)^2 + \left(\sum_{\mathbf{x}\in\mathcal{B}} \mathrm{Im}\left(e_M\left(\sum_{i=1}^{\delta} \lambda_i \mathbf{a}\mathbf{x}_i\right)\right)\right)^2 \right) \\
&\quad= \sum_{\mathbf{a}\in\mathbb{Z}_M^n} \left(\sum_{\mathbf{x}\in\mathcal{B}}\sum_{\mathbf{y}\in\mathcal{B}} \mathrm{Re}\left(e_M\left(\sum_{i=1}^{\delta} \lambda_i (\mathbf{a}\mathbf{x}_i + \mathbf{a}\mathbf{y}_i)\right)\right)\right) \\
&\qquad+ \sum_{\mathbf{a}\in\mathbb{Z}_M^n} \left(\sum_{\mathbf{x}\in\mathcal{B}}\sum_{\mathbf{y}\in\mathcal{B}} \mathrm{Im}\left(e_M\left(\sum_{i=1}^{\delta} \lambda_i (\mathbf{a}\mathbf{x}_i + \mathbf{a}\mathbf{y}_i)\right)\right)\right) \\
&\quad= \sum_{\mathbf{a}\in\mathbb{Z}_M^n} \sum_{\mathbf{x}\in\mathcal{B}} \sum_{\mathbf{y}\in\mathcal{B}} \left(\mathrm{Re}\left(e_M\left(\sum_{i=1}^{\delta} \lambda_i (\mathbf{a}\mathbf{x}_i + \mathbf{a}\mathbf{y}_i)\right)\right) + \mathrm{Im}\left(e_M\left(\sum_{i=1}^{\delta} \lambda_i (\mathbf{a}\mathbf{x}_i + \mathbf{a}\mathbf{y}_i)\right)\right)\right) \\
&\quad= \sum_{\mathbf{x}\in\mathcal{B}} \sum_{\mathbf{y}\in\mathcal{B}} \sum_{\mathbf{a}\in\mathbb{Z}_M^n} e_M\left(\sum_{i=1}^{\delta} \lambda_i (\mathbf{a}\mathbf{x}_i + \mathbf{a}\mathbf{y}_i)\right) \\
&\quad= \sum_{\mathbf{x}\in\mathcal{B}} \sum_{\mathbf{y}\in\mathcal{B}} \sum_{\mathbf{a}\in\mathbb{Z}_M^n} e_M\left(\sum_{i=1}^{\delta} \lambda_i (\mathbf{a}\mathbf{x}_i - \mathbf{a}\mathbf{y}_i)\right)
\end{aligned}$$

Aufgrund der Beschaffenheit von $\mathcal{B}$ lässt sich über die innere Summe folgende Aussage treffen:

$$\left(\exists_{t\in[1,n]}\forall_{i,j\in[1,\delta]}\colon x_{i,t}=1=y_{j,t}\Rightarrow\lambda_i\neq\lambda_j\right)\Leftrightarrow\sum_{\mathbf{a}\in\mathbb{Z}_M^n}e_M\left(\sum_{i=1}^{\delta}\lambda_i(\mathbf{a}\mathbf{x}_i-\mathbf{a}\mathbf{y}_i)\right)=0$$

Dies lässt sich anhand folgender Umformung einsehen. Dabei sei hier ohne Einschränkung $t=n$ die Stelle, die den Existenzquantor erfüllt.

$$\sum_{\mathbf{x}\in\mathcal{B}}\sum_{\mathbf{y}\in\mathcal{B}}\sum_{\mathbf{a}\in\mathbb{Z}_M^n}e_M\left(\sum_{i=1}^{\delta}\lambda_i(\mathbf{a}\mathbf{x}_i-\mathbf{a}\mathbf{y}_i)\right)$$

$$=\sum_{\mathbf{x}\in\mathcal{B}}\sum_{\mathbf{y}\in\mathcal{B}}\sum_{\mathbf{a}\in\mathbb{Z}_M^{n-1}}e_M\left(\sum_{i=1}^{\delta}\lambda_i\sum_{t=1}^{n-1}(a_kx_{i,k}-a_ky_{i,k})\right)\cdot\sum_{a_n=1}^{M-1}e_M\left(a_n\sum_{i=1}^{\delta}\lambda_i(x_{i,n}-y_{i,n})\right)$$

Ist $\sum_{i=1}^{\delta}\lambda_i(x_{i,n}-y_{i,n})=0$, so ergibt die Summe über a_n die Zahl M. Für jedes $1\leq k\leq n$ gibt es höchstens ein i, sodass $x_{i,k}=1$. Ist $\sum_{i=1}^{\delta}\lambda_i(x_{i,n}-y_{i,n})\not\equiv 0 \pmod M$, so ergibt die Summe über a_n gleich 0, womit die Gesamtsumme über $\mathbf{a}$ gleich 0 ergibt. Die obige Aussage fasst die Begebenheiten zusammen, in denen die Summe über alle $\mathbf{a}$ verschwindet. Verschwindet sie nicht, so ist sie gleich M^n.
Somit ist

$$\sum_{\mathbf{a}\in\mathbb{Z}_M^n}\left|\sum_{\mathbf{x}\in\mathcal{B}}e_M\left(\sum_{i=1}^{\delta}\lambda_i\mathbf{a}\mathbf{x}_i\right)\right|^2=\sum_{\mathbf{x}\in\mathcal{B}}\sum_{\mathbf{y}\in\mathcal{B}}\sum_{\mathbf{a}\in\mathbb{Z}_M^n}e_M\left(\sum_{i=1}^{\delta}\lambda_i(\mathbf{a}\mathbf{x}_i-\mathbf{a}\mathbf{y}_i)\right)=M^n|\mathcal{B}|Y([\lambda])$$

□

Satz 3.11 *Es gilt*

$$\mathbb{V}(X)=\frac{1}{M^n}\frac{1}{M^\delta}\sum_{\mathbf{a}\in\mathbb{Z}_M^n}\sum_{\mathbf{c}\in\mathbb{Z}_M^\delta}(P_{\mathbf{a}}(\mathbf{c},\mathcal{B})-1/M^\delta)^2=\frac{1}{M^{2\delta}}\frac{1}{\mathcal{B}}\sum_{\Lambda\in\mathbb{Z}_M^\delta/\sim}Y(\Lambda)\cdot|\Lambda|-\frac{1}{M^{2\delta}}$$

Beweis. Die erste Gleichheit wurde bereits gezeigt. Mit Lemma 3.2 gilt:

$$\begin{aligned}N_{\mathbf{a}}(\mathbf{c},\mathcal{B})&=\frac{1}{M^\delta}\sum_{\mathbf{x}\in\mathcal{B}}\sum_{\lambda\in\mathbb{Z}_M^\delta}e_M(\lambda_1(\mathbf{a}\mathbf{x}_1-c_1)+\ldots+\lambda_\delta(\mathbf{a}\mathbf{x}_\delta-c_\delta))\\&=\frac{1}{M^\delta}\sum_{\mathbf{x}\in\mathcal{B}}\sum_{\lambda\neq\mathbf{0}}e_M(\lambda_1(\mathbf{a}\mathbf{x}_1-c_1)+\ldots+\lambda_\delta(\mathbf{a}\mathbf{x}_\delta-c_\delta))+\frac{|\mathcal{B}|}{M^\delta}\end{aligned}$$

Zunächst wollen wir diesen Ausdruck berechnen:

$$\sum_{\mathbf{c}\in\mathbb{Z}_M^\delta}\left(N_{\mathbf{a}}(\mathbf{c},\mathcal{B})-\frac{|\mathcal{B}|}{M^\delta}\right)^2$$

Durch Anwenden der vorhergehenden Gleichung erhält man

$$\begin{aligned}
&\sum_{\mathbf{c}\in\mathbb{Z}_M^\delta} \left(N_{\mathbf{a}}(\mathbf{c},\mathcal{B}) - \frac{|\mathcal{B}|}{M^\delta} \right)^2 \\
&\quad = \sum_{\mathbf{c}\in\mathbb{Z}_M^\delta} \left(\frac{1}{M^\delta} \sum_{\mathbf{x}\in\mathcal{B}} \sum_{\lambda\neq\mathbf{0}} e_M(\lambda_1(\mathbf{a}\mathbf{x}_1 - c_1) + \ldots + \lambda_\delta(\mathbf{a}\mathbf{x}_\delta - c_\delta)) \right)^2 \\
&\quad = \sum_{\mathbf{c}\in\mathbb{Z}_M^\delta} \left(\frac{1}{M^\delta} \sum_{\lambda\neq\mathbf{0}} e_M(-\lambda_1 c_1 - \ldots - \lambda_\delta c_\delta) \sum_{\mathbf{x}\in\mathcal{B}} e_M(\lambda_1\mathbf{a}\mathbf{x}_1 + \ldots + \lambda_\delta\mathbf{a}\mathbf{x}_\delta) \right)^2
\end{aligned}$$

Durch die Fallunterscheidung, wie sie im vorhergehenden Beweis verwendet wurde, lässt sich weiter umformen:

$$\begin{aligned}
&\sum_{\mathbf{c}\in\mathbb{Z}_M^\delta} \left(\frac{1}{M^\delta} \sum_{\lambda\neq\mathbf{0}} e_M(-\lambda_1 c_1 - \ldots - \lambda_\delta c_\delta) \sum_{\mathbf{x}\in\mathcal{B}} e_M(\lambda_1\mathbf{a}\mathbf{x}_1 + \ldots + \lambda_\delta\mathbf{a}\mathbf{x}_\delta) \right)^2 \\
&= \frac{1}{M^{2\delta}} \sum_{\mathbf{c}\in\mathbb{Z}_M^\delta} \sum_{\lambda\neq\mathbf{0}} \sum_{\eta\neq\mathbf{0}} e_M(-(\eta_1+\lambda_1)c_1 - \ldots - (\eta_\delta+\lambda_\delta)c_\delta) \\
&\qquad \cdot \sum_{\mathbf{x},\mathbf{y}\in\mathcal{B}} e_M(\lambda_1\mathbf{a}\mathbf{x}_1 + \eta_1\mathbf{a}\mathbf{y}_1 + \ldots + \lambda_\delta\mathbf{a}\mathbf{x}_\delta + \eta_\delta\mathbf{a}\mathbf{y}_\delta) \\
&= \frac{1}{M^{2\delta}} \sum_{\lambda\neq\mathbf{0}} \sum_{\eta\neq\mathbf{0}} \left(\sum_{\mathbf{x},\mathbf{y}\in\mathcal{B}} e_M\left(\sum_{i=1}^{\delta} \lambda_i\mathbf{a}\mathbf{x}_i + \eta_i\mathbf{a}\mathbf{y}_i \right) \sum_{\mathbf{c}\in\mathbb{Z}_M^\delta} e_M\left(\sum_{i=1}^{\delta} (-\eta_i c_i - \lambda_i c_i) \right) \right) \\
&= \frac{1}{M^\delta} \sum_{\lambda\neq\mathbf{0}} \sum_{\mathbf{x},\mathbf{y}\in\mathcal{B}} e_M\left(\sum_{i=1}^{\delta} \lambda_i(\mathbf{a}\mathbf{x}_i - \mathbf{a}\mathbf{y}_i) \right)
\end{aligned}$$

Dieser Ausdruck kann analog wie in Lemma 3.10 rückwärts umgeformt werden, womit sich ergibt:

$$\sum_{\mathbf{c}\in\mathbb{Z}_M^\delta} \left(N_{\mathbf{a}}(\mathbf{c},\mathcal{B}) - \frac{|\mathcal{B}|}{M^\delta} \right)^2 = \frac{1}{M^\delta} \sum_{\lambda\neq\mathbf{0}} \left| \sum_{\mathbf{x}\in\mathcal{B}} e_M(\lambda_1\mathbf{a}\mathbf{x}_1 + \ldots + \lambda_\delta\mathbf{a}\mathbf{x}_\delta) \right|^2 .$$

Jetzt kann der Ursprungsausdruck für die Varianz berechnet werden:

$$\begin{aligned}
&\frac{1}{M^n}\frac{1}{M^\delta} \sum_{\mathbf{a}\in\mathbb{Z}_M^n} \sum_{\mathbf{c}\in\mathbb{Z}_M^\delta} \left(P_{\mathbf{a}}(\mathbf{c},\mathcal{B}) - \frac{1}{M^\delta} \right)^2 \\
&\qquad = \frac{1}{M^n}\frac{1}{M^\delta}\frac{1}{|\mathcal{B}|^2} \sum_{\mathbf{a}\in\mathbb{Z}_M^n} \sum_{\mathbf{c}\in\mathbb{Z}_M^\delta} \left(N_{\mathbf{a}}(c,\mathcal{B}) - \frac{|\mathcal{B}|}{M^\delta} \right)^2 \\
&\qquad = \frac{1}{M^n}\frac{1}{M^\delta}\frac{1}{|\mathcal{B}|^2} \sum_{\mathbf{a}\in\mathbb{Z}_M^n} \frac{1}{M^\delta} \sum_{\lambda\neq\mathbf{0}} \left| \sum_{\mathbf{x}\in\mathcal{B}} e_M(\lambda_1\mathbf{a}\mathbf{x}_1 + \ldots + \lambda_\delta\mathbf{a}\mathbf{x}_\delta) \right|^2
\end{aligned}$$

Nach Lemma 3.10 kann der Ausdruck zu folgendem umgeformt werden:

$$\frac{1}{M^n}\frac{1}{M^\delta}\frac{1}{|\mathcal{B}|^2}\frac{1}{M^\delta}\sum_{\lambda\neq\mathbf{0}} M^n|\mathcal{B}|Y([\lambda]) = \frac{1}{M^{2\delta}}\left(\frac{1}{|\mathcal{B}|}\sum_{\Lambda\in\mathbb{Z}_M^\delta/\sim} Y(\Lambda)\cdot|\Lambda| - 1\right) = \mathbb{V}(X)$$

□

Nun können wir auf diese Weise ebenso zur Aussage von Satz 3.4 kommen.

Korollar 3.12 *Ist $\delta = 1$, so ist $\mathbb{V}(X) = \frac{M-1}{M^2|\mathcal{B}|}$.*

Wir interessieren uns für den Fall $\delta = 3$.

Korollar 3.13 *Ist $\delta = 3$, so ist*

$$\begin{aligned}\mathbb{V}(X) =& \frac{4(M-1)\binom{3n/4}{n/4}\binom{2n/4}{n/4}\binom{n/4}{n/4}}{M^6|\mathcal{B}|}\\ &+ \frac{3(M-1)\binom{2n/4}{n/4}\binom{2n/4}{n/4}}{M^6|\mathcal{B}|}\\ &+ \frac{6(M-1)(M-2)\binom{2n/4}{n/4}}{M^6|\mathcal{B}|}\\ &+ \frac{(M-1)(M-2)(M-3)}{M^6|\mathcal{B}|}\end{aligned}$$

Beweis. Für $\delta = 3$ ist $\Lambda = \{[(0,0,0)],[(0,0,1)],[(0,1,1)],[(0,1,2)],[(1,1,1)],[(1,1,2)],[(1,2,3)]\}$.

- $|[(0,0,0)]| = 1$ und $Y([(0,0,0)]) = |\mathcal{B}|$
- $|[(0,0,1)]| = 3(M-1)$ und $Y([(0,0,1)]) = \binom{3n/4}{n/4}\binom{2n/4}{n/4}\binom{n/4}{n/4}$
- $|[(0,1,1)]| = 3(M-1)$ und $Y([(0,1,1)]) = \binom{2n/4}{n/4}\binom{2n/4}{n/4}$
- $|[(0,1,2)]| = 3(M-1)(M-2)$ und $Y([(0,1,2)]) = \binom{2n/4}{n/4}$
- $|[(1,1,1)]| = M-1$ und $Y([(1,1,1)]) = \binom{3n/4}{n/4}\binom{2n/4}{n/4}\binom{n/4}{n/4}$
- $|[(1,1,2)]| = 3(M-1)(M-2)$ und $Y([(1,1,2)]) = \binom{2n/4}{n/4}$
- $|[(1,2,3)]| = (M-1)(M-2)(M-3)$ und $Y([(1,2,3)]) = 1$

□

Satz 3.14 *Für $\delta = 3$, $\mu > 0$, große n und $M \geq 2^{0,595n}$ gilt*

$$P\left(\left|X - \frac{1}{M^\delta}\right| \geq \frac{\mu}{M^\delta}\right) \leq \frac{M^3}{|\mathcal{B}|\mu^2}.$$

Beweis. Durch die Tschebyschow-Ungleichung erhält man:

$$P\left(\left|X - \frac{1}{M^\delta}\right| \geq \frac{\mu}{M^\delta}\right) \leq \frac{\mathbb{V}(X)}{\left(\frac{\mu}{M^\delta}\right)^2} \leq \frac{1}{|\mathcal{B}|\mu^2}\left(M^3 + 6M^2 2^{n/2} + 3M2^n + 4M2^{1,189n}\right)$$

Ist $M \geq 2^{0,595n}$, so ist der Summand M^3 dominant, womit die Aussage folgt. □

Der folgende Satz ist auf gleiche Weise beweisbar:

Satz 3.15 *Für $\delta = 3$, $\mu > 0$, große n und $M < 2^{0,595n}$ gilt*

$$P\left(\left|X - \frac{1}{M^\delta}\right| \geq \frac{\mu}{M^\delta}\right) \leq \frac{4M2^{1,189n}}{|\mathcal{B}|\mu^2}.$$

Satz 3.16 *Es ist $\mu > 0$ und $0 < \lambda < 1$. Wenn $M \geq 2^{0,595n}$, dann beträgt die Wahrscheinlichkeit, dass für ein zufälliges $\mathbf{a} \in \mathbb{Z}_M^n$ es mindestens λM^3 viele $\mathbf{c} \in \mathbb{Z}_M^3$ gibt, die $|P_{\mathbf{a}}(\mathbf{c}, \mathcal{B}) - 1/M^3| \geq \mu/M^3$ erfüllen, höchstens $\frac{M^3}{\lambda|\mathcal{B}|\mu^2}$.*

Beweis. Dieser Satz beweist sich völlig analog zu Satz 3.7. □

Dieser Satz ist somit auch direkt klar:

Satz 3.17 *Es ist $\mu > 0$ und $0 < \lambda < 1$. Wenn $M < 2^{0,595n}$, dann beträgt die Wahrscheinlichkeit, dass für ein zufälliges $\mathbf{a} \in \mathbb{Z}_M^n$ es mindestens λM^3 viele $\mathbf{c} \in \mathbb{Z}_M^3$ gibt, die $|P_{\mathbf{a}}(\mathbf{c}, \mathcal{B}) - 1/M^3| \geq \mu/M^3$ erfüllen, höchstens $\frac{4M2^{1,189n}}{\lambda|\mathcal{B}|\mu^2}$.*

4. Der Algorithmus von Horowitz und Sahni

Der erste Algorithmus, der den naiven Ansatz verbesserte, wurde 1974 von E. Horowitz und S. Sahni entdeckt [HS74]. Dieser einfache Algorithmus bildet den Ausgangspunkt für alle weiteren Algorithmen. Die Hauptidee lässt sich durch Umschreiben der Rucksackgleichung zeigen:

$$\mathbf{ax} = c \Leftrightarrow \underbrace{\sum_{i=1}^{n/2} a_i x_i}_{\sigma_1 \in L_1} = \underbrace{c - \sum_{i=n/2+1}^{n} a_i x_i}_{\sigma_2 \in L_2}$$

Es ist also nicht nötig, den gesamten Lösungsraum $\mathbb{B}^n$ zu durchsuchen, sondern es genügt, die beiden Hälften getrennt zu betrachten und dann nach einer Kollision zu suchen. Wir führen dies nun genauer aus.
Zunächst definieren wir zwei Listen, L_1 und L_2 (und ohne Beschränkung der Allgemeinheit sei n gerade):

$$L_1 = \{\mathbf{av} \,|\, \mathbf{v} \in \mathbb{B}^n, \forall_{\frac{n}{2} < i \leq n} : v_i = 0\}$$
$$L_2 = \{c - \mathbf{av} \,|\, \mathbf{v} \in \mathbb{B}^n, \forall_{1 \leq i \leq \frac{n}{2}} : v_i = 0\}$$

Die Listen enthalten also alle Summen, die durch die erste bzw. zweite Hälfte des Gewichtsvektors darstellbar sind. Im nächsten Schritt werden die beiden Listen sortiert. Die Lösung lässt sich jetzt finden, indem die beiden Listen parallel durchlaufen werden. Wird ein Element σ (sprich eine *Kollision*) gefunden, das beide Listen enthalten, so hat die Instanz eine Lösung. Wurden für alle Listenelemente die zugehörigen Teilvektoren aus $\mathbb{B}^n$ mitprotokolliert, wovon wir im Weiteren bei derartigen Listen immer ausgehen, so lässt sich die Lösung angeben. Algorithmus 4.1 stellt den Algorithmus in Pseudocode dar und Bild 4.1 stellt ihn grafisch dar. Das Sortieren der Listen kann mit einem beliebigen Sortieralgorithmus geschehen, der eine Laufzeit von $\mathcal{O}(m \log m)$ besitzt, wobei m die Länge der Eingabe ist.
Das Finden einer Kollision erfolgt mithilfe von zwei Variablen i und j in der while-Schleife, die L_1 bzw. L_2 beim ersten Element beginnend durchläuft. Ist zu einem Zeitpunkt $L_1[i] = L_2[j]$, so wurde eine Lösung gefunden. Ist $L_1[i] < L_2[j]$, so wird i inkrementiert und wenn $L_1[i] > L_2[j]$, so wird j inkrementiert. Die Kollisionssuche endet, wenn i und j am Ende der Listen angelangt sind.
Korrektheit und Vollständigkeit. Gibt der Algorithmus eine Lösung $\mathbf{x}$ aus, so erfüllt diese offensichtlich die Rucksackgleichung $\mathbf{ax} = c$. Umgekehrt wird auf diese Art auch jede Lösung entdeckt.
Die Verwendung von i_{max} und j_{max} trägt in der Suchvariante dafür Sorge, dass auch alle Lösungen gefunden werden, die innerhalb der beiden Hälften die gleiche Summe ergeben. In der Entscheidungsvariante kann dies entfallen.

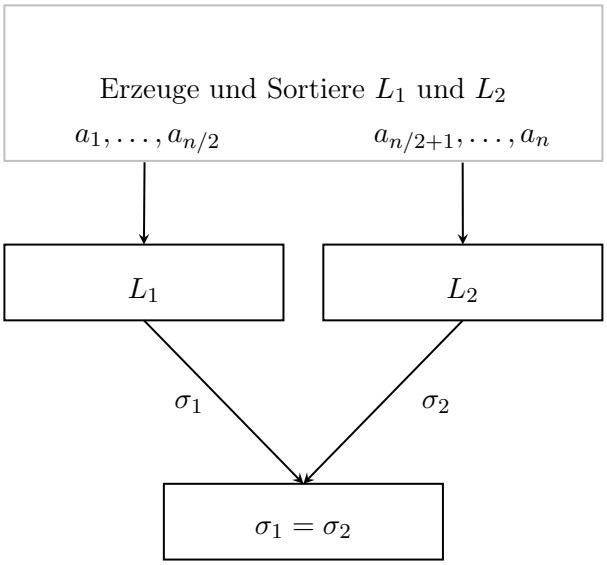

Abbildung 4.1.: Horowitz-Sahni-Algorithmus

Algorithmus 4.1 Der Algorithmus von Horowitz und Sahni

Erstelle L_1 und L_2
Sortiere L_1 und L_2
$i \leftarrow j \leftarrow 1$
while $i \leq 2^{n/2}$ und $j \leq 2^{n/2}$ **do**
  if $L_1[i] = L_2[j]$ **then**
    $i_{max} = \max\{k | L_1[k] = L_1[i]\}$
    $j_{max} = \max\{k | L_2[k] = L_2[j]\}$
    Erstelle alle möglichen Lösungen aus $L_1[\{i, \ldots, i_{max}\}]$ und $L_2[\{j, \ldots, j_{max}\}]$
    $i \leftarrow i_{max} + 1$
    $j \leftarrow j_{max} + 1$
  else if $L_1[i] < L_2[j]$ **then**
    $i \leftarrow i + 1$
  else
    $j \leftarrow j + 1$
  end if
end while

Platzkomplexität. Die beiden Listen haben eine Größe von $\tilde{\mathcal{O}}(2^{n/2})$. Das Sortieren und die Kollisionsfindung benötigen keinen zusätzlichen Platz.
Zeitkomplexität. Das Erstellen der Listen benötigt $\tilde{\mathcal{O}}(2^{n/2})$. Das Sortieren benötigt $\tilde{\mathcal{O}}(n2^{n/2})$, was jedoch gleich $\tilde{\mathcal{O}}(2^{n/2})$ ist. Die Schleife für die Kollisionsfindung läuft beide Listen ab und besucht jedes Element genau einmal, sodass auch hier die Zeitschranke $\tilde{\mathcal{O}}(2^{n/2})$ gilt.
Wir halten fest, dass die Gesamtkomplexität für Platz und Zeit in

$$\tilde{\mathcal{O}}(2^{n/2})$$

liegt.
Balance α. Dieser Algorithmus ist bisher unabhängig von α. Er ist leicht so zu modifizieren, dass er nur Lösungen sucht, die eine Länge von αn haben und so für $\alpha < 1/2$ beschleunigt wird. Für unbekanntes α können wir die Konstruktion verwenden wie sie im Kapitel 2 dargestellt wurde. Die Modifikation betrifft die Listen, welche wir nun so definieren:

$$L_1 = \left\{ \mathbf{av} \,\middle|\, \mathbf{v} \in \mathbb{B}^n, \forall_{\frac{n}{2}<i\leq n} : v_i = 0, |\mathbf{v}| = \frac{\alpha n}{2} \right\}$$
$$L_2 = \left\{ c - \mathbf{av} \,\middle|\, \mathbf{v} \in \mathbb{B}^n, \forall_{1\leq i\leq \frac{n}{2}} : v_i = 0, |\mathbf{v}| = \frac{\alpha n}{2} \right\}$$

Offensichtlich werden die Listen kleiner, wenn $\alpha < 1/2$ (vgl. Anhang A.5). Implizit ist hier jedoch die Annahme, dass genau die eine Hälfte der Lösungsgewichte in L_1 und die andere in L_2 liegt. Wir nehmen auch an, dass $|\mathbf{x}|$ gerade ist, was jedoch keine Einschränkung ist und die Darstellung an dieser Stelle vereinfacht.
Für folgenden Satz definieren wir $r(\mathbf{x}, k)$ als die k-te Rotation von $\mathbf{x}$. Beispielsweise ist $r((1,2,3,4),2) = (4,3,1,2)$. Außerdem soll an dieser Stelle $\mathbf{x}^1$ die erste Hälfte und $\mathbf{x}^2$ die zweite Hälfte eines Vektors $\mathbf{x}$ sein. Beispiel: $r((1,2,3,4),2)^2 = (1,2)$.

Lemma 4.1 *Für alle* $\mathbf{x} \in \mathbb{B}^n$ *gibt es ein* k *mit* $|r(\mathbf{x},k)^1| = |r(\mathbf{x},k)^2|$.

Beweis. Ist $|r(\mathbf{x},0)^1| = |r(\mathbf{x},0)^2|$, so sind wir fertig. Andernfalls sei ohne Einschränkung $|r(\mathbf{x},0)^1| < |r(\mathbf{x},0)^2|$. Offensichtlich gilt dann $|r(\mathbf{x},n/2)^1| > |r(\mathbf{x},n/2)^2|$. Des weiteren gilt für alle i, dass $|r(\mathbf{x},i)^1| - |r(\mathbf{x},i+1)^1| \in \{-1,0,1\}$ ist; denn sind das hinzukommende und das weggehende Element gleich, so ist die Veränderung gleich 0. Andernfalls ist sie im Betrag höchstens 1. Da bei Rotationen um einen Schritt sich der Betrag in den Hälften maximal um 1 verändern kann und nach $n/2$ Rotationen die Beträge der Hälften getauscht haben, muss es ein k geben zwischen 1 und $n/2-1$, das die geforderte Eigenschaft besitzt. □

Nach Lemma 4.1 genügt es also, die Instanz sukzessive zu rotieren; ab einer gewissen Rotation sind die Lösungshälften genau gleich auf die beiden Listen verteilt. Es gibt n Rotationen; nach n Rotationen ist der Ausgangszustand wieder erreicht. Das bedeutet, dass man den Algorithmus n mal auf die verschiedenen Rotationen ansetzen muss; dies ergibt einen zeitkomplexitätsmäßig vernachlässigbaren linearen Faktor. Für unbekanntes α bleibt die Komplexität also erhalten und für kleinere α verringert sie sich. Für $\alpha = 1/8$ ergibt sich zum Beispiel eine Platz- und Zeitkomplexität von $\tilde{\mathcal{O}}(2^{0{,}272n})$.

5. Der Algorithmus von Schroeppel und Shamir

Der Algorithmus von Horowitz und Sahni hat für Platz und Zeit eine Komplexität in $\tilde{\mathcal{O}}(2^{n/2})$. 1981 konnten R. Schroeppel und A. Shamir diesen Ansatz verbessern. Ihr Algorithmus hat einen verringerten Platzbedarf. Zunächst behandeln wir diese Grundversion. Im Anschluss wird eine randomisierte Version, die auf Howgrave-Graham und Joux zurückgeht, vorgestellt. Sie hat ähnliche Eigenschaften, ist jedoch der Grundversion unterlegen, da die heuristische Annahme, auf die sie sich stützt, nicht für ausnahmslos jede Instanz gilt. Die Daseinsberechtigung erhält die randomisierte Version dadurch, dass sie einen Zwischenschritt darstellt auf dem Weg zum aktuell effizientesten Algorithmus.

5.1. Grundversion

Die Grundversion wird in der Entscheidungsvariante in [SS81] präsentiert; wir interessieren uns für die Suchversion. Die Hauptidee besteht darin, im Gegensatz zum Horowitz-Sahni-Algorithmus, nicht zwei, sondern vier Listen zu verwenden. Wir nehmen $n \equiv 0 \pmod{M}$ an und können die Rucksackgleichung umschreiben:

$$\mathbf{ax} = c \Leftrightarrow \underbrace{\sum_{i=1}^{n/4} a_i x_i}_{\sigma_1 \in L_1} + \underbrace{\sum_{i=n/4+1}^{n/2} a_i x_i}_{\sigma_2 \in L_2} = \underbrace{c - \sum_{i=n/2+1}^{3n/4} a_i x_i}_{\sigma_3 \in L_3} \underbrace{- \sum_{i=3n/4+1}^{n} a_i x_i}_{\sigma_4 \in L_4}$$

Hieraus ergeben sich vier Listen L_1 bis L_4. Finden wir aus ihnen Elemente mit $\sigma_1 + \sigma_2 = \sigma_3 + \sigma_4$, so ist eine Lösung gefunden. Die vier Listen sind so gegeben:

$$\begin{aligned}
L_1 &= \{\mathbf{av} \,|\, \mathbf{v} \in \mathbb{B}^n, \forall_{\frac{n}{4} < i \leq n} : v_i = 0\} \\
L_2 &= \{\mathbf{av} \,|\, \mathbf{v} \in \mathbb{B}^n, \forall_{1 \leq i \leq \frac{n}{4}} : v_i = 0, \forall_{\frac{n}{2} < i \leq n} : v_i = 0\} \\
L_3 &= \{c - \mathbf{av} \,|\, \mathbf{v} \in \mathbb{B}^n, \forall_{1 \leq i \leq \frac{n}{2}} : v_i = 0, \forall_{\frac{3n}{4} < i \leq n} : v_i = 0\} \\
L_4 &= \{-\mathbf{av} \,|\, \mathbf{v} \in \mathbb{B}^n, \forall_{1 \leq i \leq \frac{3n}{4}} : v_i = 0\}
\end{aligned}$$

Würde man nun $L_1 + L_2$ und $L_3 + L_4$ explizit berechnen[1], so würde dieser Ansatz zum Horowitz-Sahni-Algorithmus degenerieren. Die sortierten Listen $L_1 + L_2$ und $L_3 + L_4$ müssen vollständig durchlaufen werden, jedoch muss vollständiges Auflisten der beiden Hälften

[1] Für zwei Listen S und T ist $S + T := \{s + t | s \in S, t \in T\}$. Somit ist $|S + T| = |S| \cdot |T|$, da jedes Element durch den erzeugenden Vektor aus $\mathbb{B}^n$, der mitprotokolliert wird, eindeutig ist.

vermieden werden, wenn Platz gespart werden soll. Hierfür ziehen Schroeppel und Shamir eine Prioritätswarteschlange heran. Sie nutzen dabei die Tatsache aus, dass auf $L_1 + L_2$ bzw. $L_3 + L_4$ keine Möglichkeit zum wahlfreien Zugriff gegeben sein muss, sonders es lediglich nötig ist, die beiden Listen in sortierter Reihenfolge zu durchlaufen. Dass dies möglich ist, zeigt der folgende Satz.

Satz 5.1 *Gegeben sind zwei sortierte Listen S und T der Größe s. Um alle Elemente der Liste $S + T$ in sortierter Reihenfolge aufzuzählen, wird $\tilde{\mathcal{O}}(s^2)$ Zeit und $\mathcal{O}(s)$ Platz benötigt*

Beweis. Wir definieren eine Prioritätswarteschlange (vgl. Anhang A.1) $Q \subseteq \{1, \ldots, n\} \times \{1, \ldots, n\}$, eine Ordnung auf den Elementen und die Zugriffsoperation auf ihr.

- Initialisiert wird Q wie folgt:

$$Q := \{(1, i) \mid i \in \{1, \ldots, n\}\}$$

- Die Ordnung ist so definiert:

$$(i_1, i_2) \leq (j_1, j_2) :\Leftrightarrow S[i_1] + T[i_2] \leq S[j_1] + T[j_2]$$

- Der Zugriff auf die Schlange erfolgt durch Entnahme des ersten (und damit kleinsten) Elementes; dieses Element sei (k_1, k_2). Gleichzeitig wird nun das Element $(k_1 + 1, k_2)$ direkt wieder in die Liste eingefügt, falls $k_1 + 1 \leq s$.

Durch s^2-maliges Entnehmen des ersten Elementes ergibt sich nun die gewünschte Aufzählung. Hierfür sind zwei Dinge zu zeigen: Es müssen alle Elemente von $S + T$ ausgegeben werden und dies muss in sortierter Reihenfolge geschehen.
Ersteres ist leicht zu sehen. Es müssen alle Elemente aus $\{1, \ldots, n\} \times \{1, \ldots, n\}$ ausgegeben werden; diese Menge kann als Tabelle angesehen werden. Zur Initialisierung werden alle Elemente der ersten Spalte vorgehalten. Bei jeder Zugriffsoperation rückt die Spalte eins nach rechts. Die Aufzählung ist erst beendet, wenn alle Zeilen ganz durchlaufen wurden. So wird kein Element vergessen.
Es bleibt zu zeigen, dass die Ausgabe sortiert erfolgt. Nach der Initialisierung steht das insgesamt kleinste Element direkt an der ersten Stelle. Nehmen wir nun induktiv an, dass die ersten i Entnahmeoperationen die ersten i Elemente korrekt sortiert ausgegeben haben. Es ist zu zeigen, dass nach $i + 1$ Schritten das nächstgrößte Element an erster Stelle der Schlange steht. Sei $P \subseteq \{1, \ldots, n\} \times \{1, \ldots, n\}$ die Menge der noch nicht ausgegebenen Elemente nach der i-ten Entnahme. Wir wissen, dass kein Element aus $\{1, \ldots, n\} \times \{1, \ldots, n\} \backslash P$ größer ist als eines aus P. Q ist also komplett in P enthalten. Das Element, das als $i + 1$-tes ausgegeben werden muss, befindet sich in P; es muss das kleinste bzw. ein kleinstes Element in P sein. Q speichert zu diesem Zeitpunkt die kleinsten Elemente jeder Zeile, also muss das insgesamt kleinste ebenfalls in Q zu finden sein und es muss an erster Stelle stehen.
Die Größe von Q ist zur Initialisierung maximal; sie hat hier n Elemente, woraus sich der Platzverbrauch $\mathcal{O}(s)$ ergibt. Es gibt s^2 Elemente, die ausgegeben werden müssen, also folgt die Laufzeit $\tilde{\mathcal{O}}(s^2)$. □

Algorithmus 5.1 Der Algorithmus von Schroeppel und Shamir

```
Erstelle L_1 bis L_4
Sortiere L_1 bis L_4
Q_1 := {(1, i) | i ∈ {1, ..., 2^{n/4}}}
Q_2 := {(1, i) | i ∈ {1, ..., 2^{n/4}}}
while Q_1 und Q_2 nicht leer do
    if Q_1[1] = Q_2[1] then
        Listen P_1 ← P_2 ← ∅
        while Q_1[1] = Q_2[1] do
            P_1 ← P_1 ∪ {Q_1[1]}
            Inkrementiere Q_1
        end while
        while P_1[1] = Q_2[1] do
            P_2 ← P_2 ∪ {Q_2[1]}
            Inkrementiere Q_2
        end while
        Konstruiere Lösungen aus P_1 + P_2 und gib diese aus
    else if Q_1[1] < Q_2[1] then
        Inkrementiere Q_1
    else if Q_1[1] > Q_1[1] then
        Inkrementiere Q_2
    end if
end while
```

Der Beweis dieses Satzes ist konstruktiv und liefert direkt den Algorithmus 5.1; Bild 5.1 veranschaulicht ihn. Q_1 und Q_2 sind dabei Prioritätswarteschlangen mit der Ordnung, wie sie im Beweis gegeben ist. Um den Algorithmus lesbarer zu machen, kürzen wir die Aktion auf den Warteschlangen mit "Inkrementiere Q" ab, die das erste Element, nennen wir es (k_1, k_2), entnimmt und, falls $k_1 + 1 \leq 2^{n/4}$ ist, $(k_1 + 1, k_2)$ wieder hinzufügt.

Korrektheit. Jede ausgegebene Lösung erfüllt offensichtlich die Rucksackgleichung.
Vollständigkeit. Da wir uns hier mit der Suchvariante des Algorithmus beschäftigen, müssen alle Lösungen gefunden werden. Durch die Konstruktion im "=-Fall" wird dafür Sorge getragen, dass verschiedene Lösungen, die jedoch innerhalb der beiden Hälften der Instanz die gleiche Summe ergeben, gefunden werden. An dieser Stelle ist anzumerken, dass in der Veröffentlichung von Schroeppel und Shamir [SS81] nur der Algorithmus für die Entscheidungsvariante angegeben wurde. Howgrave-Graham und Joux haben in ihrer Veröffentlichung [HGJ10] den Algorithmus in der Suchvariante dargestellt, es jedoch versäumt, dabei die eben beschriebene notwendige Konstruktion einzufügen. Ihre Version ist also nicht vollständig. Dies ist jedoch für ihren neuen Algorithmus nötig, da dieser seinerseits auf die Suchvariante des Schroeppel-Shamir-Algorithmus zurückgreift.
Platzkomplexität. Die Listen wie auch die Prioritätswarteschlangen benötigen den selben Platz $\tilde{\mathcal{O}}(2^{n/4})$.

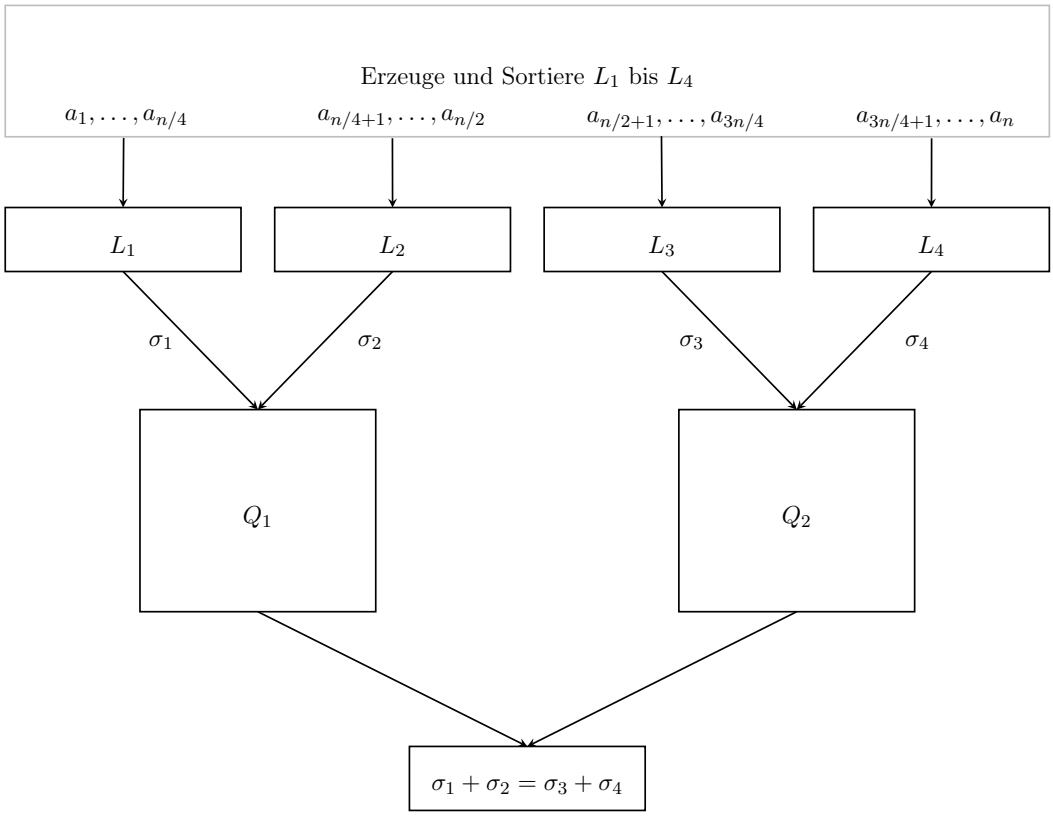

Abbildung 5.1.: Schroeppel-Shamir-Algorithmus

Zeitkomplexität. Es werden nach wie vor alle Elemente aus $L_1 + L_2$ bzw $L_3 + L_4$ betrachtet, wodurch die Zeitschranke $\tilde{\mathcal{O}}(2^{n/2})$ erhalten bleibt. Durch das Verwenden der Prioritätswarteschlange entsteht für jedes Element ein im schlechtesten Fall logarithmischer Aufwand, womit die Zeitkomplexität insgesamt mit $\tilde{\mathcal{O}}(2^{n/2})$ gültig bleibt.

Version für unbekannte Balance α. Analog zum Horowitz-Sahni-Algorithmus sind wir an einer Version interessiert, die apriori-Wissen über α verwenden kann. Wir fügen also den Listen L_1 bis L_4 die Bedingung hinzu, dass je genau $\frac{\alpha n}{4}$ Gewichte summiert werden. Die Annahme hierbei ist, dass sich die Lösung auf alle vier Listen gleichmäßig verteilt. Mit Lemma 4.1 wissen wir, dass es eine Rotation gibt, sodass sich die Lösung gleichmäßig auf die beiden Hälften der Instanz verteilt. Durch Rotieren innerhalb der Hälften kann hier wiederum die richtige Rotierung gefunden werden, sodass sich die Lösungsgewichte gleichmäßig auf die vier Viertel verteilen. Der Mehraufwand für die äußere Rotation ist ein zeitlicher Faktor n. Für eine innere Rotation in einer Hälfte ist er $n/2$. Da jede Rotation auf der linken Seite mit jeder auf der rechten kombiniert werden muss, ergibt sich ein insgesamt vernachlässigbarer Mehrkostenfaktor von $\mathcal{O}(n^3)$.

Algorithmus 5.2 Der Algorithmus von Schroeppel und Shamir (randomisierte Version)

Erstelle L_1 bis L_4
Sortiere L_1 bis L_4
$M \leftarrow$ Primzahl nahe $2^{n/4}$
for $R = 0$ **to** $M - 1$ **do**
    Erstelle $L_{1,2}^R$ und $L_{3,4}^R$
    Sortiere $L_{1,2}^R$ und $L_{3,4}^R$
    Finde Kollisionen zwischen $L_{1,2}^R$ und $L_{3,4}^R$ (wie bei Horowitz und Sahni)
end for

5.2. Randomisierte Version

Wie eingangs bereits erwähnt, verfolgt die randomisierte Version des Schroeppel-Shamir-Algorithmus, die auf Howgrave-Graham und Joux zurückgeht [HGJ10] das Ziel, den neuen Algorithmus vorzubereiten. Die Veränderung betrifft die in der Grundversion verwendete Prioritätswarteschlange. Diese aufwändige Konstruktion wird in dieser Version ersetzt.
Zunächst werden, wie gehabt, die Listen L_1 bis L_4 erzeugt und sortiert. Nun sind $\sigma_1 \in L_1$ bis $\sigma_4 \in L_4$ mit $\sigma_1 + \sigma_2 = \sigma_3 + \sigma_4$ gesucht. Eine triviale Beobachtung ist, dass bezüglich eines Modulus M ein R existiert mit

$$\sigma_1 + \sigma_2 = \sigma_3 + \sigma_4 \equiv R \pmod{M}.$$

Wir definieren nun zwei Listen, die die beiden Prioritätswarteschlangen ersetzen.

$$L_{1,2}^R := \{\sigma \in L_1 + L_2 \,|\, \sigma \equiv R \pmod{M}\}$$
$$L_{3,4}^R := \{\sigma \in L_3 + L_4 \,|\, \sigma \equiv R \pmod{M}\}$$

Erzeugen wir diese Listen für das richtige R und finden eine Kollision, so haben wir offensichtlich eine Lösung gefunden, falls diese Kollision die Rucksackgleichung erfüllt. Die Erzeugung dieser Listen geschieht durch paralleles Durchlaufen und benötigt keinen zusätzlichen Platz oder Zeit. Da wir das richtige R nicht kennen, werden alle Möglichkeiten durchprobiert. R läuft also in einer Schleife von 0 bis $M - 1$. Die Wahl von M ist dabei entscheidend. Je größer M gewählt wird, desto kleiner werden die Listen und desto länger wird die Laufzeit in diesem Schritt. Jedoch gibt es nur eine sinnvolle Möglichkeit, M zu wählen, nämlich als Primzahl nahe $2^{n/4}$. Die Primzahleigenschaft wird in Beweis zur Heuristik benötigt. Der resultierende Algorithmus ist in 5.2 gegeben; siehe auch Bild 5.2. Die Kollisionsfindung zwischen $L_{1,2}^R$ und $L_{3,4}^R$ läuft analog wie bei Horowitz und Sahni und wird übersichtlichkeitshalber hier abgekürzt.

Korrektheit und Vollständigkeit. Jede ausgegebene Lösung erfüllt offensichtlich die Rucksackgleichung und jede Lösung für eine Instanz ist mit einer bestimmten Belegung für R auffindbar.

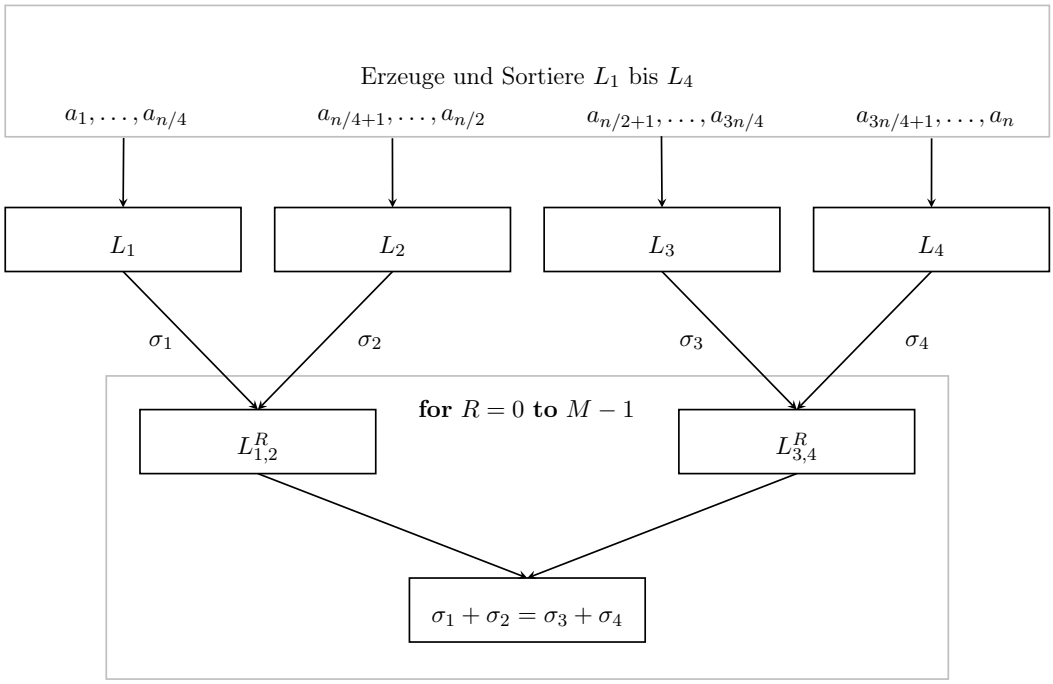

Abbildung 5.2.: Heuristische Version des Schroeppel-Shamir-Algorithmus

Zeitkomplexität. Die Komplexität für die Erstellung der sortierten Listen L_1 bis L_4 ist bekannt. Für die Schleife gilt:

$$\begin{aligned}\tilde{\mathcal{O}}\left(\sum_{R=0}^{M-1}\left(|L_{1,2}^R| + |L_{3,4}^R|\right)\right) &= \tilde{\mathcal{O}}\left(\sum_{R=0}^{M-1}|L_{1,2}^R| + \sum_{R=0}^{M-1}|L_{3,4}^R|\right)\\ &= \tilde{\mathcal{O}}\left(|L_1| \cdot |L_2| + |L_3| \cdot |L_4|\right)\\ &= \tilde{\mathcal{O}}(2^{n/2})\end{aligned}$$

Somit bleibt die Zeitkomplexität erhalten.
Platzkomplexität. Auch hier betrachten wir nur die Schleife. Die Komplexität ist

$$\tilde{\mathcal{O}}\left(\max \bigcup_{R=0}^{M-1}\{|L_{1,2}^R|, |L_{3,4}^R|\}\right).$$

Zunächst können wir eine Schranke für den Worst Case angeben. Sind alle Elemente der Listen (z.B. L_1 und L_2) kongruent zu einem bestimmten R_0, so sind alle Listen außer $L_{1,2}^{R_0}$ leer, wobei $|L_{1,2}^{R_0}| = 2^{n/2}$. Somit hat dieser Algorithmus im schlechtesten Fall die gleiche Komplexität wie der Horowitz-Sahni-Algorithmus. Betrachtet man jedoch die Menge aller möglichen Instanzen, so ist zu erwarten, dass die meisten nicht so beschaffen sind, dass überproportional viele Summen auf dieselbe Äquivalenzklasse modulo M zusammenfallen. Dies ist die grundlegende heuristische Annahme, die auch allen weiteren Algorithmen zu Grunde liegt. Sie wurde in

Kapitel 3 hergeleitet.

Satz 5.2 *Für $\mu > 0$ wird mit einer Wahrscheinlichkeit von $1 - \frac{1}{\mu^2}$ die Platzschranke (Anzahl der Elemente) $(\mu + 1)2^{n/4}$ für die Listen $L^R_{1,2}$ bzw. $L^R_{3,4}$ nicht verletzt.*

Beweis. Wir verwenden hierfür Satz 3.6 und wenden ihn je auf die beiden Hälften $(a_1, \ldots, a_{n/2})$ und $(a_{n/2+1}, \ldots, a_n)$ an; wir führen die Rechnung für den linken Teil aus. $\mathcal{B}$ ist dabei jeweils $\mathbb{B}^{n/2}$ und korrespondiert zu den Elementen in $L_1 + L_2$ und außerdem ist $M \approx 2^{n/4}$.
Der Ausdruck $|P_{\mathbf{a}}(R, \mathcal{B}) - 1/M| \geq \mu/M$ beschreibt, falls $\mu > 1$, das Ereignis, dass Elemente aus $L_1 + L_2$ mit einer Wahrscheinlichkeit von mehr als $(\mu + 1)/M$ in die Liste $L^R_{1,2}$ fallen. Anders gesagt bedeutet dies, dass

$$|L^R_{1,2}| > \frac{(\mu + 1)|\mathcal{B}|}{M} \approx (\mu + 1)2^{n/4}.$$

Die Wahrscheinlichkeit, dass es für ein $\mathbf{a}$ ein R gibt, das dies erfüllt, ist höchstens

$$\frac{M^2}{|\mathcal{B}|\mu^2} = \frac{1}{\mu^2}.$$

Negiert man diese Aussage, so ergibt sich der Satz. □

Wie man sieht geht, wenn man $\mu = n$ setzt, die Wahrscheinlichkeit für große n gegen 1. Dies bedeutet, dass man mit einem linearen Faktor die Wahrscheinlichkeit, dass eine Instanz diese Platzschranke nicht verletzt, beliebig nahe an 1 annähern kann. Daraus ergibt sich die Schlussfolgerung, dass für einen beliebig großen Anteil aller Instanzen die Platzschranke

$$\tilde{\mathcal{O}}(2^{n/4})$$

für die Listen L_1 bis L_4 und $L^R_{1,2}$ und $L^R_{3,4}$ gilt.

6. Der Algorithmus von Howgrave-Graham und Joux

Dieses Kapitel stellt den Hauptteil der Arbeit dar und behandelt den aktuellen Algorithmus von N. Howgrave-Graham und A. Joux aus dem Jahr 2010 [HGJ10]. Er baut auf der randomisierten Version des Schroeppel-Shamir-Algorithmus auf. Von diesem ausgehend sind noch zwei Modifikationen zu tätigen, um den neuen Algorithmus zu erhalten. Zu beachten ist im folgenden, dass der neue Algorithmus zunächst in der Entscheidungsvariante zur Verfügung stehen wird im Gegensatz zu den vorhergehenden, die in der Suchvariante vorliegen.

Erste Modifikation. Die randomisierte Version des Schroeppel-Shamir-Algorithmus beinhaltet eine Schleife für R von 0 bis $M-1$, um alle Möglichkeiten für R durchzuprobieren. Diese Schleife ersetzen wir nun. Es sollen Werte für R zufällig gewählt werden, bis eine Lösung gefunden wurde; siehe Algorithmus 6.1.

Wir wollen nun untersuchen, wie oft im Schnitt R geraten werden muss, bis *die*[1] Lösung gefunden wurde.

Die Wahrscheinlichkeit, dass ein zufälliges R die Lösung nicht liefert, ist $1-2^{-n/4}$. Es sei k die Anzahl der Schleifendurchläufe in denen R zufällig gewählt wird.

$$1-\left(1-\frac{1}{2^{n/4}}\right)^k$$

Algorithmus 6.1 Der Algorithmus von Schroeppel und Shamir (randomisierte Version 2)

```
Erstelle L_1 bis L_4
Sortiere L_1 bis L_4
M ← Primzahl nahe 2^{n/4}
while keine Lösung gefunden do
    R ← random(0, ..., M − 1)
    Erstelle L^R_{1,2} und L^R_{3,4}
    Sortiere L^R_{1,2} und L^R_{3,4}
    Finde Kollisionen zwischen L^R_{1,2} und L^R_{3,4} (wie bei Horowitz und Sahni)
end while
```

[1] Wir rechnen nicht damit, dass es mehr als eine Lösung gibt.

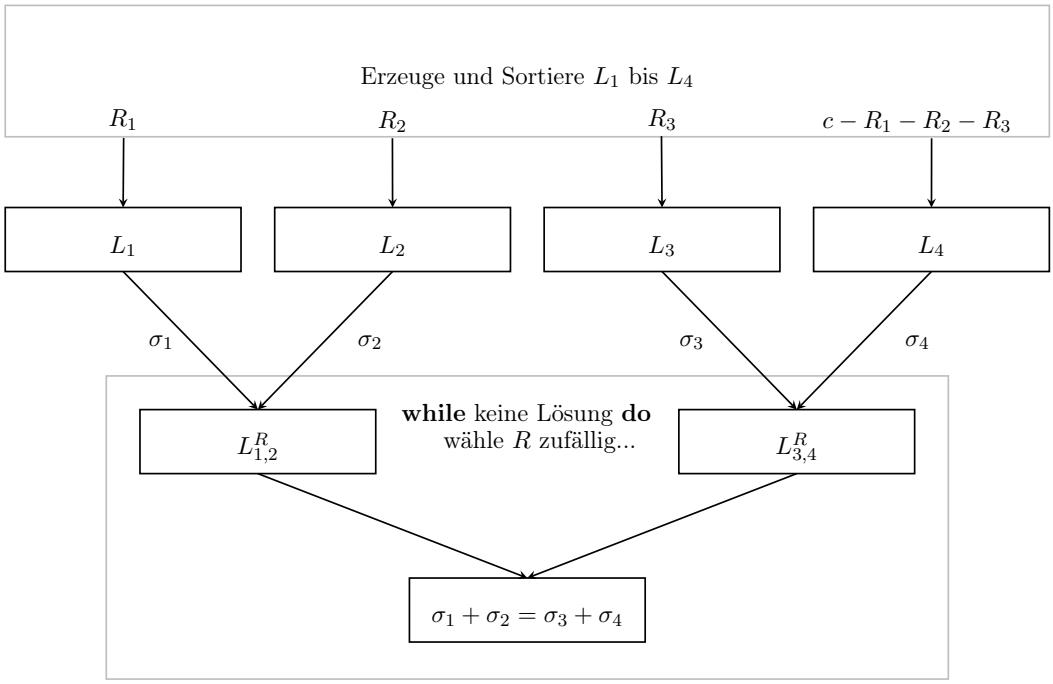

Abbildung 6.1.: Algorithmus von Howgrave-Graham und Joux

ist dann die Wahrscheinlichkeit, bei diesen k Durchläufen mindestens einmal das richtige R mit der Lösung zu treffen. Es gilt der Grenzwert

$$\lim_{n\to\infty} \left(1 + \frac{m}{n}\right)^n = e^m.$$

Nach $k = 2^{n/4}$ Schleifendurchläufen ist die Wahrscheinlichkeit, die Lösung gefunden zu haben, somit bei $1 - 1/e$ und für $k = n2^{n/4}$ und große n bei 1. Das bedeutet, dass die Anzahl der Schleifendurchläufe sich im Vergleich zum Durchlaufen für R von 0 bis $M - 1$ nicht negativ verändert und die Laufzeit bleibt mit beliebig großer Wahrscheinlichkeit bei $\tilde{\mathcal{O}}(2^{n/2})$.
Angenommen eine Instanz hat viele Lösungen: Je mehr es davon gibt, desto höher ist die Wahrscheinlichkeit, in einem Schleifendurchlauf eine zu finden und desto weniger Durchläufe werden benötigt. Dass eine Instanz nur eine Lösung besitzt, ist unabänderbar, jedoch lässt sich eine Lösung vervielfältigen, was die Idee hinter der zweiten Modifikation ist, die somit den endgültigen Algorithmus von Howgrave-Graham und Joux darstellt.

Zweite Modifikation. Bisher sind die Listen L_1 bis L_4 den vier Vierteln der Instanz zuge-

Algorithmus 6.2 Der Algorithmus von Howgrave-Graham und Joux

$M \leftarrow$ Primzahl nahe $2^{\beta n}$
$R_1, R_2, R_3 \leftarrow random(0, \ldots, M-1)$
Erstelle L_1 bis L_4 mithilfe des Schroeppel-Shamir-Algorithmus mit Balance $\alpha/4$
Sortiere L_1 bis L_4
$M' \leftarrow$ Primzahl nahe $2^{(0{,}544-\beta)n}$
while keine Lösung gefunden **do**
  $R \leftarrow random(0, \ldots, M'-1)$
  Erstelle $L_{1,2}^R$ und $L_{3,4}^R$ (mit Konsistenzprüfung)
  Sortiere $L_{1,2}^R$ und $L_{3,4}^R$
  Finde konsistente Kollisionen zwischen $L_{1,2}^R$ und $L_{3,4}^R$ (wie bei Horowitz und Sahni)
end while

ordnet. Im Folgenden werden diese umdefiniert, was auch schon die wesentliche Veränderung ist, durch die man den neuen Algorithmus bekommt:

$$L_1 = \left\{ \mathbf{av} \;\middle|\; \mathbf{v} \in \mathbb{B}^n,\ |\mathbf{v}| = \frac{\alpha n}{4},\ \mathbf{av} \equiv R_1 \pmod{M} \right\}$$
$$L_2 = \left\{ \mathbf{av} \;\middle|\; \mathbf{v} \in \mathbb{B}^n,\ |\mathbf{v}| = \frac{\alpha n}{4},\ \mathbf{av} \equiv R_2 \pmod{M} \right\}$$
$$L_3 = \left\{ \mathbf{av} \;\middle|\; \mathbf{v} \in \mathbb{B}^n,\ |\mathbf{v}| = \frac{\alpha n}{4},\ \mathbf{av} \equiv R_3 \pmod{M} \right\}$$
$$L_4 = \left\{ \mathbf{av} \;\middle|\; \mathbf{v} \in \mathbb{B}^n,\ |\mathbf{v}| = \frac{\alpha n}{4},\ \mathbf{av} \equiv c - R_1 - R_2 - R_3 \pmod{M} \right\}$$

Dabei ist M eine Primzahl nahe $2^{\beta n}$, die Variable β dient zum Einstellen des Speicher-Zeit-Tradeoffs und R_1 bis R_3 liegen zufällig zwischen 0 und $M-1$.
Wählt man die richtigen Werte für R_1 bis R_3, so verteilt sich die Lösung auf L_1 bis L_4. Es wird dabei keine Variable R_4 benötigt, da für die Lösung nur $c - R_1 - R_2 - R_3$ übrig bleiben kann. Offensichtlich gibt es viele Möglichkeiten, wie die Lösung in vier Teile partitioniert werden kann. Jede Partition lässt sich unter anderen Werten für R_1 bis R_3 finden. Die Anzahl der Partitionen - im Folgenden sprechen wir von *Repräsentationen* - für den Worst Case $\alpha = 1/2$ ist

$$\binom{4n/8}{n/8}\binom{3n/8}{n/8}\binom{2n/8}{n/8}\binom{n/8}{n/8} \approx 2^n.$$

Im Gegenzug gibt es $2^{3\beta}$ Möglichkeiten R_1 bis R_3 zu wählen. Unter der Annahme, dass die Repräsentationen sich gleichmäßig verteilen, ist die Anzahl der Repräsentationen, die für eine Wahl von R_1 bis R_3 getroffen werden

$$2^{(1-3\beta)n}.$$

Das Ausnützen dieser Tatsache wird auch als **Repräsentationstechnik** bezeichnet. Es kann nun der Howgrave-Graham-Joux-Algorithmus angegeben werden; siehe Algorithmus 6.2 und Bild 6.1.

Das Erstellen von L_1 bis L_4 ist ein modulares Rucksackproblem mit Balance $\alpha/4$; es kann hierfür der Schroeppel-Shamir-Algorithmus verwendet werden. Für die Erzeugung der Listen

$L_{1,2}^R$ bzw. $L_{3,4}^R$ wird M' statt M verwendet. Bei der Erstellung und Kollisionsfindung zwischen $L_{1,2}^R$ und $L_{3,4}^R$ muss beachtet werden, dass Teillösungen sich überlappen können. Um eine Lösung zu erhalten, darf dies natürlich nicht der Fall sein; die Lösung muss *konsistent* sein.
Korrektheit. Eine gefundene Lösung ist ursprünglich überlappungsfrei auf $L_{1,2}^R$ und $L_{3,4}^R$ bzw. L_1 bis L_4 verteilt. Die Rucksackgleichung wird offensichtlich ebenfalls erfüllt.
Vollständigkeit. Da es sich um die Entscheidungsvariante handelt, genügt es, lediglich *eine* Lösung finden. Wie bereits erwähnt, werden nach der heuristischen Annahme für eine Belegung von R_1 bis R_3 eine Anzahl von $2^{(1-3\beta)n}$ Repräsentationen gefunden. Dies soll nun bewiesen werden. Die Menge der Repräsentationen ist als eine Teilmenge von $(\mathbb{B}^n)^4$ darstellbar. Da wir uns jedoch an dieser Stelle nur für Gewichte interessieren, die Teil der Lösung sind und da sich das vierte Teilstück der Lösung implizit ergibt, können wir die Menge der Repräsentationen definieren als $\mathcal{B} \subset (\mathbb{B}^{\alpha n})^3$, wobei hierzu ein Gewichtsvektor $\mathbf{a}' \in \mathbb{Z}^{\alpha n}$ korrespondiert, der nur die Lösungsgewichte enthält. Ein $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) \in \mathcal{B}$ erfüllt dabei, dass $|\mathbf{x}_1| = |\mathbf{x}_2| = |\mathbf{x}_3| = \alpha n/4$ und dass $\mathbf{x}_1$ bis $\mathbf{x}_3$ je paarweise überlappungsfrei sind.

Satz 6.1 *Für große n und $\beta > 0{,}2972$ beinhalten die Listen L_1 bis L_4 (welche anhand R_1 bis R_3 erzeugt werden) $\tilde{\mathcal{O}}(2^{(1-3\beta)n})$ Repräsentationen.*

Beweis. Satz 3.16 liefert eine obere Schranke für den Anteil der Instanzen, die nicht die gewünschte Eigenschaft haben; das heißt, eine obere Schranke für den Anteil der Instanzen, für die mehr als λM^3 Belegungen für R_1 bis R_3 eine zu geringe Anzahl an Repräsentationen trifft; dabei ist $0 < \lambda < 1$. Es ist zu beachten, dass bei der Anwendung des Satzes n mit αn(im Worst Case also $n/2$) substituiert werden muss. Hieraus folgt die Voraussetzung des zu beweisenden Satzes, nämlich dass $\beta > 0{,}2972$. Im Weiteren ist $\mathcal{B}$ wie oben definiert und $N_{\mathbf{a}'}((R_1, R_2, R_3), \mathcal{B})$ die Anzahl der Repräsentationen, die durch R_1 bis R_3 betroffen werden.
Nach Satz 3.16 ist die Wahrscheinlichkeit für ein *schlechtes* $\mathbf{a}$ höchstens $\frac{M^3}{\lambda|\mathcal{B}|\mu^2}$, wobei ein $\mathbf{a}$ schlecht ist, wenn mehr als λM^3 viele Belegungen für R_1 bis R_3 zu wenig Repräsentationen beinhalten, also

$$N_{\mathbf{a}'}((R_1, R_2, R_3), \mathcal{B}) \leq \frac{(1-\mu)|\mathcal{B}|}{M^3}.$$

Für große n können wir λ und μ klein wählen und erhalten immer noch eine gegen 0 gehende Wahrscheinlichkeit. Insbesondere $\lambda = \frac{M^3}{|\mathcal{B}|2^{\epsilon n}}$ gewählt werden für ein $\epsilon > 0$. Somit kann für $(1-\lambda)$ viele Belegungen für R_1 bis R_3 auf hinreichend viele Repräsentationen gezählt werden und $(1-\lambda)$ geht dabei gegen 1. □

Wir können diesen Beweis auch auf kleinere Werte für β erweitern.

Satz 6.2 *Für große n und $\beta \leq 0{,}2972$ beinhalten die Listen L_1 bis L_4 (welche anhand R_1 bis R_3 erzeugt werden) $\tilde{\mathcal{O}}(2^{(1-3\beta)n})$ Repräsentationen.*

Beweis. Die Argumentation ist gleich wie im vorhergehenden Satz. Jedoch wird zum Beweis statt Satz 3.16 der Satz 3.17 herangezogen. Die Wahrscheinlichkeit für ein schlechtes $\mathbf{a}$ ist dann $\frac{4M2^{0{,}595n}}{\lambda|\mathcal{B}|\mu^2}$. Wir wählen in diesem Fall $\lambda = \frac{4M2^{0{,}595n}}{|\mathcal{B}|2^{\epsilon n}}$ mit $\epsilon > 0$ und erhalten für $\beta < 0{,}406$ das gleiche Ergebnis. □

Für $\beta < 1/3$ wird also die Lösung gefunden. Für große n werden hier schon exponentiell viele Repräsentationen getroffen. Ab $\beta = 1/3$ ist dies nicht mehr der Fall.
Platzkomplexität. Zunächst benötigt der Aufruf des Schroeppel-Shamir-Algorithmus zur Erzeugung der Listen L_1 bis L_4 Platz. Da er mit der Balance $\alpha/4$ aufgerufen wird, ergibt sich der Platzverbrauch $\tilde{\mathcal{O}}(2^{0{,}544\cdot\frac{1}{4}n}) = \tilde{\mathcal{O}}(2^{0{,}136n})$.
Die Grundmenge der Listen L_1 bis L_4 hat eine Größe von $\binom{n}{n/8} \approx 2^{0{,}544n}$. Heuristisch gilt also, dass die Listen eine Größe haben, die in $\tilde{\mathcal{O}}(\binom{n}{n/8}/M) = \tilde{\mathcal{O}}(2^{(0{,}544-\beta)n})$ liegt.

Satz 6.3 *Mit an Sicherheit grenzender Wahrscheinlichkeit wird die Platzschranke für die Größe der Listen L_1 bis L_4 mit $2 \cdot 2^{(0{,}544-\beta)n}$ nicht verletzt.*

Beweis. Das beweisen wir mit Satz 3.7. Wir wählen $\mu = 1$, womit sich die Platzschranke ergibt. Die Wahrscheinlichkeit, dass diese für λM viele R_1 (bzw. R_2 oder R_3) nicht eingehalten wird, ist höchstens $\frac{M}{\mu^2|\mathcal{B}|\lambda}$. Wählen wir λ nahe $M/|\mathcal{B}|$, so geht diese Wahrscheinlichkeit gegen 0. Die Wahrscheinlichkeit vier mal die Platzschranke nicht zu verletzen ist damit $(1-\lambda)^4$, was gegen 1 geht.□

Für die Listen $L_{1,2}^R$ und $L_{3,4}^R$ gilt das gleiche: Die Grundmenge ist $L_1 + L_2$ bzw. $L_3 + L_4$. Der Platzverbrauch liegt heuristisch in $\tilde{\mathcal{O}}(2^{2(0{,}544-\beta)n}/M') = \tilde{\mathcal{O}}(2^{(0{,}544-\beta)n})$ und kann wiederum bewiesen werden analog zu Satz 5.2. Zur Erinnerung: die Platzschranken für die Listen können mit einem linearen Term beliebig wahrscheinlich eingehalten werden.
Dies alles ergibt einen Gesamtplatzverbauch von

$$\tilde{\mathcal{O}}(\max\{2^{0{,}136n}, 2^{(0{,}544-\beta)n}\}).$$

Zeitkomplexität. Zunächst wird offensichtlich mindestens so viel Zeit wie Platz verwendet. Das Erstellen der Listen L_1 bis L_4 mit dem Schroeppel-Shamir-Algorithmus benötigt $\tilde{\mathcal{O}}(2^{0{,}544\cdot\frac{1}{2}n}) = \tilde{\mathcal{O}}(2^{0{,}272n})$ Zeit.
Als nächstes wollen wir wissen, wie oft R mindestens zufällig gewählt werden muss, um sehr sicher sein zu können, dass eine Lösung gefunden wird. Nach Satz 6.1 gibt es ungefähr $2^{(1-3\beta)n}$ zugängliche Repräsentationen in L_1 bis L_4. Jedoch kann nicht a priori geschlossen werden, dass es $2^{(1-3\beta)n}$ viele Werte für R gibt, sodass eine Repräsentation getroffen wird, denn es gibt eine gewisse Varianz für die Größe der Listen $L_{1,2}^R$ bzw. $L_{3,4}^R$. Größere Listen werden auch eine proportional höhere Wahrscheinlichkeit haben, eine Repräsentation zu beinhalten, als kleinere. Der nächste Satz zeigt, dass wir jedoch davon ausgehen können, dass keine zwei Repräsentationen unter derselben Belegung für R zu finden sind.

Satz 6.4 *Für $\beta > \frac{1{,}466}{7} = 0{,}209$ geht die Wahrscheinlichkeit, dass alle Repräsentationen unter verschiedenen Belegungen für R zu finden sind, für große n gegen 1.*

Analog zu Satz 5.2 gehen wir davon aus, dass die Listen $L_{1,2}^R$ und $L_{3,4}^R$ höchstens $(1+\mu)2^{(0{,}544-\beta)n}$ Einträge haben. Im für die zu beweisende Aussage schlechtesten Fall ist die Anzahl der leeren Listen maximal. Wir rechnen weiter, in dem wir nur eine der beiden Listen betrachten, da sich bei der richtigen Wahl für R die andere Hälfte von selbst ergibt. Wir nehmen also an, dass für

$p := \frac{|L_1+L_2|}{(1+\mu)2^{(0,544-\beta)n}} = \frac{2^{(0,544-\beta)n}}{1+\mu}$ viele Belegungen für R gilt, dass $L_{1,2}^R$ die maximale Größe hat. Im Gegenzug sind in $\frac{\mu}{1+\mu}2^{(0,544-\beta)n}$ der Fälle die Listen leer. Weiter setzen wir $q := 2^{(1-3\beta)n}$. Wir schätzen nun ab, wie groß die Wahrscheinlichkeit ist, dass sich die q Repräsentationen in q unterschiedlichen Listen finden. Sie ist

$$\frac{p}{p} \cdot \frac{p-1}{p} \cdot \ldots \cdot \frac{p-q-1}{p} = \prod_{i=0}^{q-1} \frac{p-i}{p} = \prod_{i=0}^{q-1} \left(1 - \frac{i}{p}\right).$$

Es gilt $1 + x \geq 2^x$ für $0 \leq x \leq 1$. Also:

$$\prod_{i=0}^{q-1} \left(1 - \frac{i}{p}\right) \geq \prod_{i=0}^{q-1} 2^{-\frac{i}{p}} = \frac{1}{2^{\frac{1}{p}\sum_{i=0}^{q-1} i}} = \frac{1}{2^{\frac{q^2-q}{2p}}}$$

Es sei ρ die erstrebte Wahrscheinlichkeit. Nun soll also gelten

$$\frac{1}{2^{(q^2-q)/2p}} > \rho.$$

Dies formen wir um:

$$\begin{aligned}
&\frac{1}{2^{(q^2-q)/2p}} > \rho \\
\Leftrightarrow\ & 1 > \rho 2^{(q^2-q)/2p} > \rho 2^{q^2/2p} \\
\Rightarrow\ & \frac{1}{\rho} > 2^{q^2/2p} \\
\Leftrightarrow\ & \log \frac{1}{\rho} > \frac{q^2}{2p} \\
\Leftrightarrow\ & -\log \rho > \frac{2^{2(1-3\beta)n}}{2\frac{1}{1+\mu}2^{(0,544-\beta)n}} > \frac{2^{2(1-3\beta)n}}{2 \cdot 2^{(0,544-\beta)n}} = 2^{(2(1-3\beta)-(0,544-\beta))n-1} \\
\Rightarrow\ & \frac{\log(-2\log\rho)}{n} > 1{,}466 - 7\beta \\
\Leftrightarrow\ & \frac{1{,}466}{7} - \frac{\log(-2\log\rho)}{7n} < \beta
\end{aligned}$$

Wie man sieht, kann für große n die Wahrscheinlichkeit ρ beliebig nahe bei 1 gewählt werden, wenn $\beta > \frac{1{,}466}{7} = 0{,}209$ ist. □

Wir können also damit rechnen, dass ungefähr $2^{(1-3\beta)n}$ Belegungen für R zu einer Lösung führen. Es sei wieder k die Anzahl der Schleifendurchläufe.

$$\left(1 - \frac{1}{M'}\right)^{2^{(1-3\beta)n}} = \left(1 - \frac{1}{2^{(0,544-\beta)n}}\right)^{2^{(1-3\beta)n}}$$

ist die Wahrscheinlichkeit, dass durch eine Wahl von R keine der Repräsentationen gefunden wird.

$$1 - \left(\left(1 - \frac{1}{2^{(0,544-\beta)n}}\right)^{2^{(1-3\beta)n}}\right)^k$$

ist dann die Wahrscheinlichkeit, dass nach k Durchläufen mindestens eine Repräsentation gefunden wurde. Setzt man nun

$$k = n\frac{2^{(0,544-\beta)n}}{2^{(1-3\beta)n}} = n2^{(2\beta-0,457)n},$$

so geht die betrachtete Wahrscheinlichkeit für große n gegen 1.
In jedem Schleifendurchlauf müssen die Listen $L_{1,2}^R$ und $L_{3,4}^R$ erzeugt werden, was durch paralleles Durchlaufen der Listen L_1 und L_2 bzw. L_3 und L_4 geschieht. Um eine Kollision zwischen $L_{1,2}^R$ und $L_{3,4}^R$ zu finden, werden diese sortiert und wiederum parallel durchlaufen. Ein Schleifendurchlauf hat also eine Laufzeit in $\tilde{\mathcal{O}}(2^{(0,544-\beta)n})$.
Setzt man alles zusammen, erhält man folgende Laufzeit:

$$\tilde{\mathcal{O}}(\max\{2^{(0,544-\beta)n}, 2^{(\beta+0,087)n}\})$$

In Bezug auf die Laufzeit sei noch bemerkt, dass die Analyse des Algorithmus in [HGJ10] fehlerhaft war; dieser Fehler wurde in [BCJ11] erläutert.
Tradeoffs. Durch die Variable β hat man die Möglichkeit, Speicher und Zeit gegeneinander abzuwägen. Dabei bestehen für β Grenzen, außerhalb derer kein sinnvoller Tradeoff mehr möglich ist. Zunächst muss $\beta < 1/3$ sein, damit davon ausgegangen werden kann, dass überhaupt eine Lösung gefunden wird. Des weiteren sollte $\beta \geq 0{,}2285$ sein, da für noch kleinere β kein weiterer Zeitvorteil erreicht werden kann, da in diesem Grenzfall im Schnitt schon ein einziger Schleifendurchlauf genügt, um eine Lösung zu finden. Der Satz 6.1 gilt nur für $\beta > 0{,}2972$, sodass Howgrave-Graham und Joux das als eine Schranke ansehen. Für kleinere Werte treffen sie die Annahme, dass Satz 6.1 auch dann noch gilt, jedoch schränken sie diese gleichzeitig wieder auf Werte für $\beta > 1/4$ ein. Satz 6.2 untermauert, dass generell diese Vermutung für alle $\beta < 0{,}2972$ gilt. 0,2285 ist damit *die* untere Schranke.
Des Weiteren ist zu beachten, dass es in der Praxis vorkommen könnte, dass für ein β nahe $1/3$ keine Lösung gefunden wird. Die Aussage, dass für $\beta < 1/3$ eine Lösung gefunden wird, gilt schließlich nur für große n.
Bild 6.2 stellt eine Übersicht über die bisher behandelten Algorithmen dar.
Suchvariante. Der Algorithmus, wie er bisher vorliegt, stoppt, sobald eine Lösung gefunden wurde. Die erwartete Laufzeit dafür wurde berechnet. Es ist jedoch auch möglich, den Algorithmus so zu modifizieren, dass er mit hoher Wahrscheinlichkeit alle Lösungen ausgibt. Sei $\mathcal{L}$ die Anzahl der Lösungen einer Instanz. Es wurde bereits etabliert, dass

$$1 - \left(\left(1 - \frac{1}{2^{(0,544-\beta)n}}\right)^{2^{(1-3\beta)n}}\right)^k$$

die Wahrscheinlichkeit ist, durch eine Wahl von R eine bestimmte Lösung zu finden. Die Wahrscheinlichkeit, nach k Durchläufen alle $\mathcal{L}$ Lösungen zu finden, ist also

$$\left(1 - \left(\left(1 - \frac{1}{2^{(0,544-\beta)n}}\right)^{2^{(1-3\beta)n}}\right)^{k\cdot p}\right)^{\mathcal{L}},$$

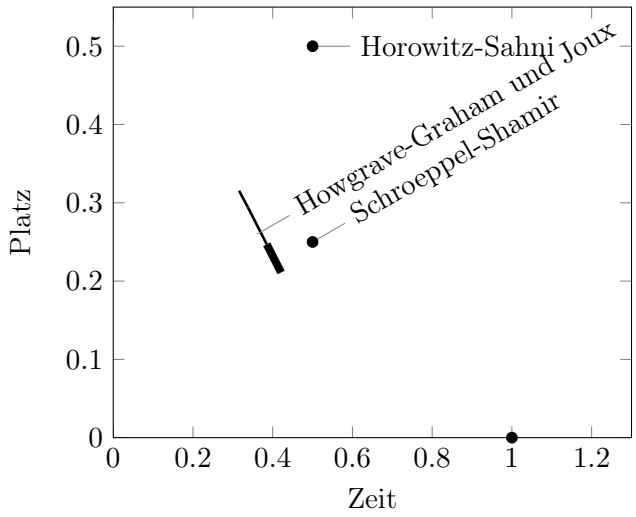

Abbildung 6.2.: Übersicht über die Komplexitäten der bisher behandelten Algorithmen. Die Achsen geben jeweils den Faktor im Exponenten an. Der dick gezeichnete Bereich ist der durch Satz 6.1 abgedeckte Bereich; der dünne entspricht dem hier durch Satz 6.2 bewiesenen Bereich.

wobei p der Faktor sein soll, um den die Schleife öfter durchlaufen werden muss. k wird gleich gewählt wie bisher, nämlich mit $\frac{2^{(0,544-\beta)n}}{2^{(1-3\beta)n}}$. Die Wahrscheinlichkeit soll nun gegen 1 gehen. Dies gilt, wenn p folgendes erfüllt:

$$\begin{aligned} n\mathcal{L} &\leq \frac{1}{\left(\left(1-\frac{1}{2^{(0,544-\beta)n}}\right)^{2^{(1-3\beta)n}}\right)^{k\cdot p}} \\ &= \frac{1}{\left(1-\frac{p}{2^{(0,544-\beta)n}}\right)^{k2^{(1-3\beta)n}}} \end{aligned}$$

Für große n gilt also mit Einsetzen von k und Anwenden des Grenzwertes, dass

$$n\mathcal{L} \leq \frac{1}{e^{-p}} \Leftrightarrow \ln(n\mathcal{L}) \leq p.$$

Die Schleife muss also $\ln(n\mathcal{L})$ mal öfter durchlaufen werden, um mit an Sicherheit grenzender Wahrscheinlichkeit alle Lösungen gefunden zu haben. Die Laufzeit bleibt damit unter Vernachlässigung polylogarithmischer Faktoren gleich.
Hat eine Instanz viele Lösungen, so beschleunigt dies die Entscheidungsvariante, denn es muss nur eine einzige Lösung gefunden werden. Gibt es $\mathcal{L}$ viele Lösungen, so gibt es auch um den Faktor $\mathcal{L}$ mehr Repräsentationen einer Lösung. Die Anzahl der Wiederholungen k wird also um den Faktor $\mathcal{L}$ kleiner.

7. Erweiterung des Tradeoffs

Der Algorithmus von Howgrave-Graham und Joux erlaubt für die Tradeoff-Variable β nur Werte kleiner 1/3. Da Platz teuer ist, ist es erstrebenswert, eine Möglichkeit zu haben, den Tradeoff für $\beta \geq 1/3$ fortzusetzen, was zu einer größeren Laufzeit und geringerem Platzverbrauch führt. In diesem Abschnitt soll dies bewerkstelligt werden.

7.1. Erste Erweiterung

Betrachtet man den Grenzfall $\beta = 1/3$, so gibt es nach Satz 6.1 nur eine konstante, nicht von n abhängige Anzahl an Repräsentationen, die mit einer zufälligen Wahl von R_1 bis R_3 getroffen wird. Aufgrund von Varianz wird es jedoch auch Situationen geben, in denen keine Repräsentation getroffen wird, was der Grund ist, dass $\beta < 1/3$ gefordert wurde. Den Fall $\beta = 1/3$ kann man jedoch behandeln, indem man gegebenenfalls bei Misserfolg neue Werte für R_1 bis R_3 wählt. Im bisherigen Algorithmus ist ein Misserfolg aber nicht detektierbar, da die Listen in zweiter Ebene ($L^R_{1,2}$ und $L^R_{3,4}$) zufällig erzeugt werden und er deshalb nicht terminieren würde. Da wir im besten Fall jedoch ohnehin nur von einer Lösung ausgehen, kann hier der gesamte Bereich für R abgesucht werden. Jedoch ist es eine noch bessere Möglichkeit, die Listen L_1 bis L_4 wie bei der Grundversion des Schroeppel-Shamir-Algorithmus mithilfe der Prioritätsschlangenkonstruktion deterministisch nach einer Lösung abzusuchen. Dies umgesetzt führt zu einem neuen Algorithmus 7.1; siehe auch Bild 7.1. Diese Konstruktion ist offensichtlich anwendbar für $\beta \geq 1/3$. Nun sollen neue Grenzen für β gefunden werden.
Korrektheit und Vollständigkeit. Jede ausgegebene Lösung erfüllt die Rucksackgleichung. Außerdem wird nach hinreichend langer Laufzeit mit an Sicherheit grenzender Wahrscheinlichkeit eine Repräsentation gefunden.

Satz 7.1 *Nach $\tilde{\mathcal{O}}(M^3/2^n)$ Schleifendurchläufen wurde mit hoher Wahrscheinlichkeit eine Lösung gefunden.*

Beweis. Zunächst gilt es abzuschätzen, für wie viele verschiedene Belegungen für R_1 bis R_3 eine Repräsentation getroffen wird. Wir zeigen, dass mehr als $2^n/2$ Belegungen für R_1 bis R_3 dies gewährleisten. Wir setzen $p := 2^n$ als die Anzahl der Repräsentationen und $q := M^3 = 2^{3\beta n}$. Für den Grenzfall $\beta = 1/3$ ist $p = q$. Wir argumentieren anhand des Urnenmodells. Es werden p mal Elemente aus $\{1, \ldots, q\}$ mit Zurücklegen gezogen. Es sei $P \in \{1, \ldots p\}^q$ ein Vektor, der speichert, wie oft jedes Element gezogen wurde. Da wir die untere Schranke 1/2 zeigen wollen,

Algorithmus 7.1 Algorithmus mit erweitertem Tradeoff

```
M ← Primzahl nahe 2^{βn}
while keine Lösung gefunden do
    R_1, R_2, R_3 ← random(0, ..., M − 1)
    Erstelle L_1 bis L_4 mithilfe des Schroeppel-Shamir-Algorithmus mit Balance α/4
    Sortiere L_1 bis L_4
    //ab hier analog zur Grundversion des Schroeppel-Shamir-Algorithmus
    Q_1 := {(1, i) | i ∈ {1, ..., |L_2|}}
    Q_2 := {(1, i) | i ∈ {1, ..., |L_4|}}
    while Q_1 und Q_2 nicht leer do
        if Q_1[1] = Q_2[1] then
            Listen P_1 ← P_2 ← ∅
            while Q_1[1] = Q_2[1] do
                P_1 ← P_1 ∪ {Q_1[1]}
                Inkrementiere Q_1
            end while
            while P_1[1] = Q_2[1] do
                P_2 ← P_2 ∪ {Q_2[1]}
                Inkrementiere Q_2
            end while
            Konstruiere konsistente Lösungen aus P_1 + P_2 und gib diese aus
        else if Q_1[1] < Q_2[1] then
            Inkrementiere Q_1
        else
            Inkrementiere Q_2
        end if
    end while
end while
```

nehmen wir an, dass P an der Hälfte der Stellen 0 ist. Betrachte folgende Aktion: Zunächst wird ein beliebiges Element aus der Auswahl gelöscht, das heißt eine von 0 verschiedene Stelle in P wird dekrementiert. Danach wird eine neue Zahl gezogen, was bedeutet, dass eine zufällige Stelle in P inkrementiert wird. Mit einer Wahrscheinlichkeit kleiner $1/2$ erhöht sich die Zahl der 0-Stellen in P, nämlich genau dann, wenn eine Position mit dem Eintrag 1 gewählt wurde. Hingegen erhöht sich die Zahl der von 0 verschiedenen Stellen mit Wahrscheinlichkeit von genau $1/2$, da die Hälfte der Stellen 0 sind. Wird diese Aktion iteriert, wird sich der Anteil erhöhen, wodurch gezeigt wurde, dass $1/2$ eine untere Schranke für den Anteil der von 0 verschiedenen Stellen ist. Das heißt, für $\beta = 1/3$ können wir annehmen, dass für $p/2$ aller Belegungen von R_1 bis R_3 eine Repräsentation gefunden wird. Für $\beta > 1/3$ steigt dieser Anteil noch weiter.
Diesen Faktor $1/2$ können wir im Weiteren für große n vernachlässigen. Die Wahrscheinlichkeit, nach einmaligem Wählen von R_1 bis R_3 eine Repräsentation zu finden, ist also

$$1 - \frac{1}{2^{(3\beta-1)n}}.$$

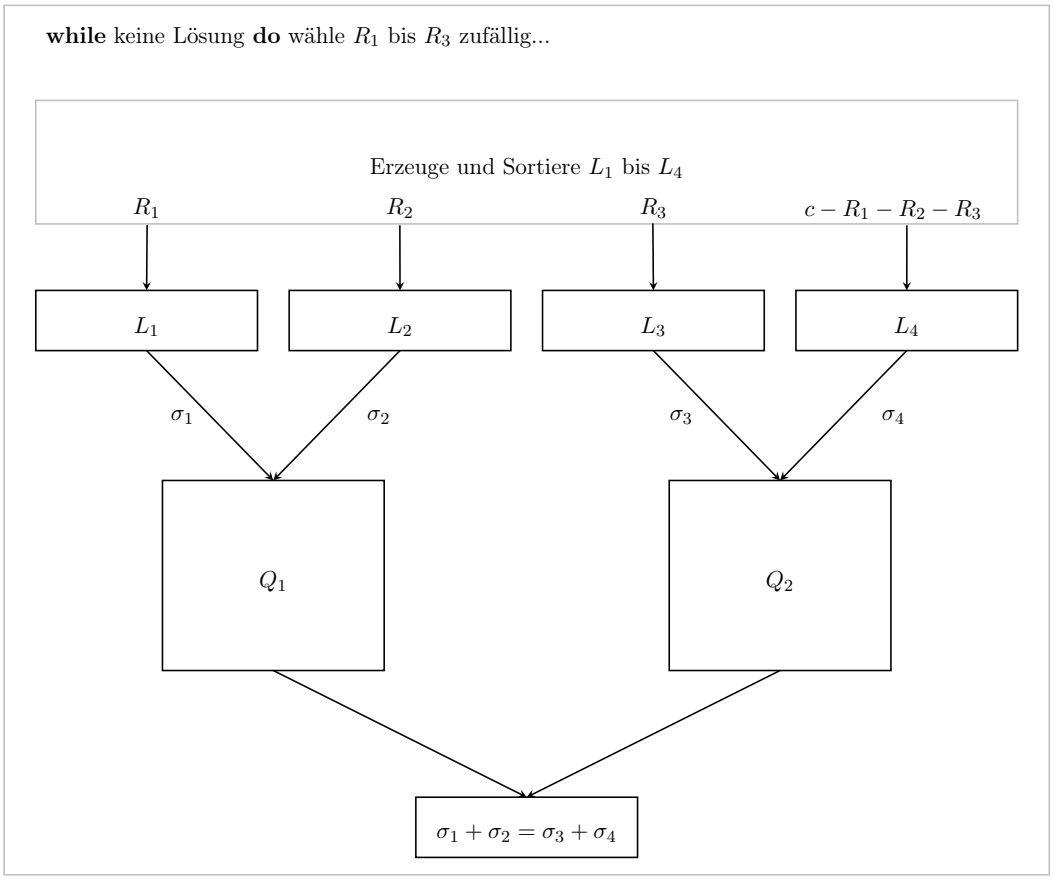

Abbildung 7.1.: Algorithmus mit erweitertem Tradeoff

Es sei k die Anzahl der Schleifendurchläufe. Die Wahrscheinlichkeit, mindestens eine Repräsentation zu finden ist

$$1 - \left(1 - \frac{1}{2^{(3\beta-1)n}}\right)^k.$$

Ist $k \geq n2^{(3\beta-1)n}$, so wird mit großer Sicherheit die Lösung gefunden. □

Platzkomplexität. Die Prioritätswarteschlangen Q_1 und Q_2 sind höchstens so groß wie L_2 bzw. L_4. Für die Größe der Listen muss Satz 6.3 verfeinert werden, da angenommen wurde, dass L_1 bis L_4 nur einmalig erzeugt werden.

Satz 7.2 *Für große n wird die Platzschranke für die Größe der Listen L_1 bis L_4 mit $2 \cdot 2^{(0,544-\beta)n}$ bei k-maligem Ziehen nicht verletzt, wobei $k \in o(|\mathcal{B}|/M) = o(2^{(0,544-\beta)n})$.*

Beweis. Wir ziehen wieder Satz 3.7 heran und wählen $\mu = 1$, womit sich die Platzschranke ergibt. Die Wahrscheinlichkeit, dass diese für λM viele R_1 (bzw. R_2 oder R_3) nicht eingehalten wird, ist höchstens $\frac{M}{\mu^2|\mathcal{B}|\lambda}$. Wählen wir λ nahe $M/|\mathcal{B}|$, so geht diese Wahrscheinlichkeit gegen 0.

Die Wahrscheinlichkeit, $4k$ mal die Platzschranke nicht zu verletzen ist damit $(1-\lambda)^{4k}$, was für $k \in o(1/\lambda)$ gegen 1 geht. □

Durch Satz 7.1 wissen wir, dass $k = 2^{(3\beta-1)n}$. Um die Voraussetzung des Satzes zu erfüllen, fordern wir $3\beta - 1 < 0{,}544 - \beta$, woraus folgt, dass $\beta < 1{,}544/4 = 0{,}386$. Für größere Werte ist es notwendig, den Listen mehr Platz zuzugestehen. Im Rahmen des obigen Beweises würde die Variable μ den erhöhten Platzbedarf abfangen. Folge ist, dass für $\beta > 0{,}386$ der Gesamtplatzbedarf wieder steigen würde. Für die Listen ist er dann $2^{(\beta-0{,}228)n}$ und damit nicht mehr zielführend. Informell kann der Sachverhalt so erklärt werden, dass, je kleiner die Listen werden, mehr Wiederholungen nötig werden. Mehr Wiederholungen wiederum erhöhen die Chance auf Ausreißer, sodass für hinreichend kleine β keine beliebig kleinen Listen garantiert werden können. Diese allgemeine Beobachtung könnte es schwierig machen, die Lücke zwischen den hier behandelten und den Subexponentialplatz-Algorithmen (vgl. [BCJ11]) zu schließen.
Wir fahren fort unter der Annahme, dass auch für $\beta > 0{,}386$ die Listen die Platzschranke nicht verletzen.
Insgesamt erhalten wir eine Platzkomplexität von

$$\tilde{\mathcal{O}}(\max\{2^{0{,}136n}, 2^{(0{,}544-\beta)n}\}).$$

Zeitkomplexität. Ein Schleifendurchlauf, in dem L_1 bis L_4 erzeugt und nach einer Lösung durchsucht wird, benötigt $|L_1|^2$ Zeit, was $\tilde{\mathcal{O}}(2^{2(0{,}544-\beta)n})$ entspricht. Außerdem müssen die Listen erzeugt werden, was $\tilde{\mathcal{O}}(2^{0{,}272n})$ benötigt. Zusammen mit Satz 7.1 erhalten wir die Gesamtlaufzeit

$$\tilde{\mathcal{O}}(\max\{2^{(\beta+0{,}087)n}, 2^{(3\beta-0{,}728)n}\}).$$

Betrachtet man die Platzkomplexität, so kann man hieran ablesen, dass $\beta < 0{,}544 - 0{,}136 = 0{,}408$ sein sollte, da ab diesem Wert der Platzverbrauch des Aufrufs des Schroeppel-Shamir-Algorithmus dominant ist.
Bild 7.2 stellt den neuen Tradeoff dar.

7.2. Zweite Erweiterung

Die Erweiterung des Tradeoffs kann noch fortgesetzt werden. In Anhang B wird ein Tradeoff für den Schroeppel-Shamir-Algorithmus wiedergegeben, der an dieser Stelle verwendet wird. Um den Platzverbrauch weiter zu reduzieren, muss jedoch überproportional mehr Zeit investiert werden. Der neue Tradeoff lohnt, bis sich die Komplexität mit der des Schroeppel-Shamir-Tradeoffs selbst schneidet.
Algorithmus 7.2 stellt den abermals erweiterten Tradeoff dar. Die Kollisionsfindung zwischen $L_1 + L_2$ und $L_3 + L_4$ wurde dabei abgekürzt. Der Algorithmus soll für den Bereich $\beta \geq 0{,}408$ gelten.
Der Schroeppel-Shamir-Tradeoff hat für Balance 1/8, wie sie im Worst Case auftritt, eine Laufzeit von

$$\tilde{\mathcal{O}}(2^{(0{,}544/2+\beta')n})$$

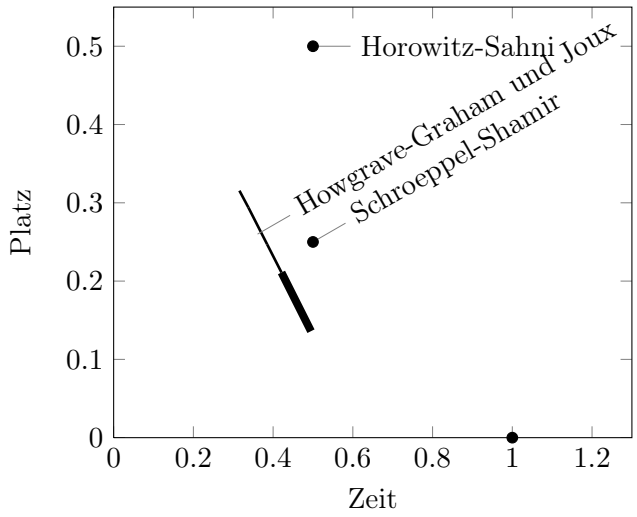

Abbildung 7.2.: Übersicht über die Komplexitäten der bisher behandelten Algorithmen. Die Achsen geben jeweils den Faktor im Exponenten an. Der dünne Bereich korrespondiert zum Algorithmus von Howgrave-Graham und Joux und der dicke zum neuen Tradeoff.

Algorithmus 7.2 Algorithmus mit erweitertem Tradeoff (2)

$M \leftarrow$ Primzahl nahe $2^{\beta n}$
while keine Lösung gefunden **do**
  $R_1, R_2, R_3 \leftarrow random(0, \ldots, M-1)$
  Erstelle L_1 bis L_4 (Schroeppel-Shamir-Algo.; Balance $\alpha/4$; Tradeoff $\beta' = \beta - 0{,}408$)
  Sortiere L_1 bis L_4
  Finde konsistente Kollision zwischen $L_1 + L_2$ und $L_3 + L_4$
end while

und einen Platzverbrauch von

$$\tilde{\mathcal{O}}(\max\{2^{0{,}544/16n}, 2^{(0{,}544/4-\beta')n}\}).$$

Das ist eine bessere Komplexität, als der Algorithmus von Howgrave-Graham und Joux für $\alpha = 1/2$ hätte. Der sinnvolle Bereich für β' ist $0 \leq \beta' \leq 0{,}544 \cdot 3/16$ und damit ergibt sich für β ein Bereich von $0{,}408 \leq \beta \leq 0{,}408 + 0{,}544 \cdot 3/16 = 0{,}510$.
Platzkomplexität. Die Listen L_1 bis L_4 haben, wie gehabt, die Größe $\tilde{\mathcal{O}}(2^{(0{,}544-\beta)n})$. Die Berechnung der Listen benötigt nach Wahl von β' den Platz $\tilde{\mathcal{O}}(2^{(0{,}544/4-\beta+0{,}408)n}) = \tilde{\mathcal{O}}(2^{(0{,}544-\beta)n})$. Wir erhalten also

$$\tilde{\mathcal{O}}(\max\{2^{0{,}034n}, 2^{(0{,}544-\beta)n}\}).$$

Zeitkomplexität. Ein Schleifendurchlauf, in dem L_1 bis L_4 erzeugt und nach einer Lösung durchsucht wird, benötigt $|L_1|^2$ Zeit, was $\tilde{\mathcal{O}}(2^{2(0{,}544-\beta)n})$ entspricht. Außerdem müssen die

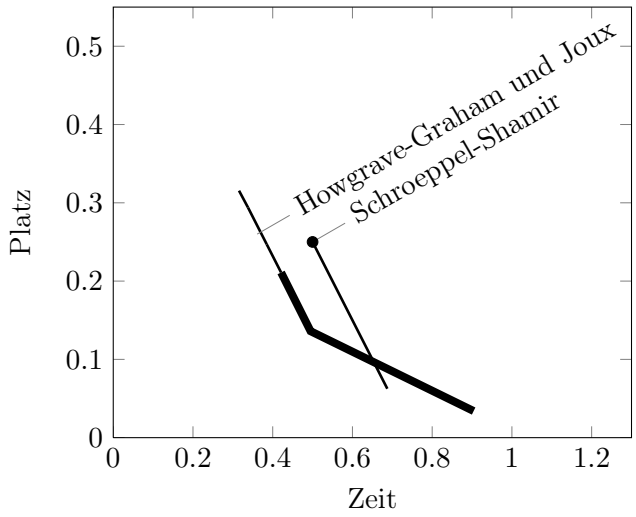

Abbildung 7.3.: Schroeppel-Shamir-Tradeoff und neuer Tradeoff für Howgrave-Graham und Joux (dicke Linie)

Listen erzeugt werden, was $\tilde{\mathcal{O}}(2^{(0{,}272+\beta-0{,}408)n}) = \tilde{\mathcal{O}}(2^{(\beta-0{,}136)n})$ benötigt. Nach Satz 7.1 sind $\tilde{\mathcal{O}}(2^{-(1-3\beta)n})$ Schleifendurchläufe nötig. Zusammen ergibt das

$$\tilde{\mathcal{O}}(\max\{2^{(\beta+0{,}087)n}, 2^{(4\beta-1{,}136)n}\})$$

In Bild 7.3 ist der neue Tradeoff eingezeichnet. Das Ende der Geraden ist mit $\beta = 0{,}510$ erreicht. Der neue Tradeoff ist sinnvoll bis zum Schnittpunkt mit dem Schroeppel-Shamir-Tradeoff. Dieser liegt für den neuen Tradeoff bei $\beta = 0,447$.

8. Übersicht

In der Arbeit wurden zahlreiche Algorithmen behandelt. In diesem Abschnitt sollen ihre Eckdaten gegenübergestellt werden.

Komplexitäten der Algorithmen ohne Tradeoff für $\alpha = 1/2$:

	Zeitkomplexität	Platzkomplexität
naiver Algorithmus	$\tilde{\mathcal{O}}(2^n)$	$\tilde{\mathcal{O}}(n)$
Horowitz-Sahni	$\tilde{\mathcal{O}}(2^{n/2})$	$\tilde{\mathcal{O}}(2^{n/2})$
Schroeppel-Shamir Grundversion (gilt auch für randomisierte Ver.)	$\tilde{\mathcal{O}}(2^{n/2})$	$\tilde{\mathcal{O}}(2^{n/4})$

Komplexitäten der Algorithmen ohne Tradeoff in Abhängigkeit von α (für ψ-Funktion vgl. Anhang A.5):

	Zeitkomplexität	Platzkomplexität
naiver Algorithmus	$\tilde{\mathcal{O}}(2^{\psi(\alpha)n})$	$\tilde{\mathcal{O}}(n)$
Horowitz-Sahni	$\tilde{\mathcal{O}}(2^{\psi(\alpha)n/2})$	$\tilde{\mathcal{O}}(2^{\psi(\alpha)n/2})$
Schroeppel-Shamir Grundversion (gilt auch für randomisierte Ver.)	$\tilde{\mathcal{O}}(2^{\psi(\alpha)n/2})$	$\tilde{\mathcal{O}}(2^{\psi(\alpha)n/4})$

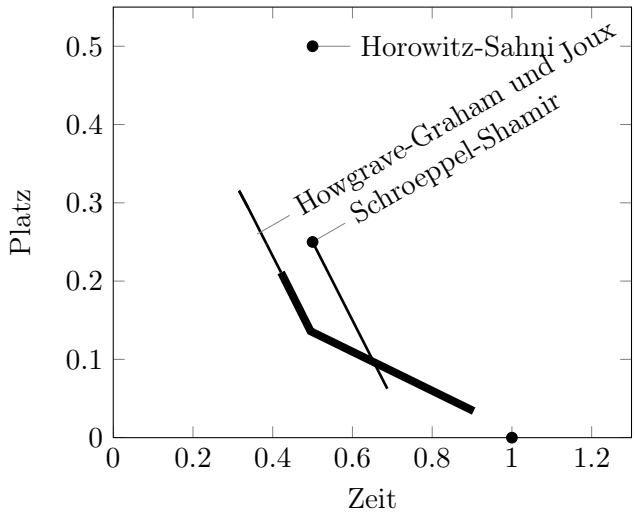

Abbildung 8.1.: Übersicht über die Komplexitäten der behandelten Algorithmen für $\alpha = 1/2$

Komplexitäten für Tradeoff-Algorithmen mit $\alpha = 1/2$:

	Zeitkomplexität	Platzkomplexität
Schroeppel-Shamir mit Tradeoff	$\tilde{\mathcal{O}}(2^{(1/2+\beta)n})$	$\tilde{\mathcal{O}}(2^{(1/4-\beta)n})$
Howgrave-Graham und Joux (gilt auch für Algorithmus 7.1)	$\tilde{\mathcal{O}}(\max\{2^{(0,544-\beta)n}, 2^{(0,087+\beta)n}\})$	$\tilde{\mathcal{O}}(\max\{2^{0,136n}, 2^{(0,544-\beta)n}\})$
Algorithmus 7.2	$\tilde{\mathcal{O}}(\max\{2^{(4\beta-1,136)n}, 2^{(0,087+\beta)n}\})$	$\tilde{\mathcal{O}}(\max\{2^{0,034n}, 2^{(0,544-\beta)n}\})$

Die Algorithmen 7.1 und 7.2 sind die mit dem erweiterten Tradeoff aus Kapitel 7. Für die Tradeoff-Algorithmen sind die Grenzen für die Tradeoff-Variable wichtig. Der Schroeppel-Shamir-Tradeoff operiert innerhalb der Grenze $0 \leq \beta \leq 3/16$.
Wir behandeln den Algorithmus von Howgrave-Graham und die beiden neuen Algorithmen als einen durchgängigen Tradeoff. Die Tabellen auf den beiden folgenden Seiten verdeutlichen die Bereiche.
Bild 8.1 stellt die Komplexitäten der Algorithmen innerhalb ihrer Tradeoffbereiche für $\alpha = 1/2$ dar.

Bereich für β	Erläuterung
$\beta < 0{,}2285$	nicht sinnvoll, da Komplexität sich sowohl für Platz, als auch Zeit verschlechtert
$0{,}2285 \leq \beta < 1/3$	gesamter Bereich, auf den der Algorithmus von Howgrave-Graham und Joux potentiell angewendet werden kann
$0{,}2972 \leq \beta < 1/3$	Bereich, für den in [HGJ10] die Heuristik bewiesen wurde (vgl. Satz 6.1)
$1/4 < \beta < 0{,}2972$	Bereich, für den Howgrave-Graham und Joux annehmen, dass Heuristik nach wie vor gilt
$0{,}2285 \leq \beta < 0{,}2972$	durch Satz 6.2 beweisbarer Bereich
$1/3 \leq \beta < 0{,}408$	In diesem Bereich gilt die erste Tradeofferweiterung; siehe Algorithmus 7.1.
$0{,}402 \leq \beta \leq 0{,}510$	In diesem Bereich gilt die zweite Tradeofferweiterung; siehe Algorithmus 7.2; ab $\beta = 0{,}447$ ist jedoch Schroeppel-Shamir-Tradeoff besser (vgl. Anhang B).
$1/3 \leq \beta < 0{,}386$	Für diesen Bereich ist für die Tradeofferweiterung die Heuristik beweisbar.
$0{,}510 < \beta$	nicht sinnvoll, da Platzverbrauch nicht weiter reduziert wird

β	Zeit	Platz	Erläuterung
$\beta = 0{,}2285$	$\tilde{\mathcal{O}}(2^{0{,}3155n})$	$\tilde{\mathcal{O}}(2^{0{,}3155n})$	beste Laufzeit; Heuristik beweisbar
$\beta = 1/4$	$\tilde{\mathcal{O}}(2^{0{,}337n})$	$\tilde{\mathcal{O}}(2^{0{,}294n})$	nach Howgrave-Graham und Joux schnellster Tradeoff
$\beta = 0{,}2972$	$\tilde{\mathcal{O}}(2^{0{,}3842n})$	$\tilde{\mathcal{O}}(2^{0{,}247n})$	schnellster in [HGJ10] heuristisch bewiesener Tradeoff
$\beta = 1/3$	$\tilde{\mathcal{O}}(2^{0{,}420n})$	$\tilde{\mathcal{O}}(2^{0{,}211n})$	geringster Platzverbrauch für Algorithmus von Howgrave-Graham und Joux
$\beta = 0{,}386$	$\tilde{\mathcal{O}}(2^{0{,}473n})$	$\tilde{\mathcal{O}}(2^{0{,}158n})$	geringster Platzverbrauch in der Tradeoff-Erweiterung, für den Heuristik beweisbar
$\beta = 0{,}408$	$\tilde{\mathcal{O}}(2^{0{,}495n})$	$\tilde{\mathcal{O}}(2^{0{,}142n})$	geringster Platzverbrauch der ersten Erweiterung
$\beta = 0{,}447$	$\tilde{\mathcal{O}}(2^{0{,}652n})$	$\tilde{\mathcal{O}}(2^{0{,}097n})$	gleiche Komplexität mit Schroeppel-Shamir-Tradeoff erreichbar
$\beta = 0{,}510$	$\tilde{\mathcal{O}}(2^{0{,}904n})$	$\tilde{\mathcal{O}}(2^{0{,}034n})$	geringster Platzverbrauch

9. Fazit

9.1. Zusammenfassung

Aufgabenstellung dieser Arbeit ist es, eine in sich geschlossene Darstellung des Schroeppel-Shamir-Algorithmus, sowie des neuen Algorithmus von Howgrave-Graham und Joux zu geben. Diese Algorithmen wurden ausführlich behandelt und - ausgehend vom Horowitz-Sahni-Algorithmus - hergeleitet, wobei klar wurde, wie die Algorithmen aufeinander aufbauen. Zu einer geschlossenen Darstellung gehören auch die Beweise zur heuristischen Annahme, welche hier ebenso vollständig erbracht wurden. Hierzu mussten viele Auslassungen in den Quellen [HGJ10] und [NSS00] aufgefüllt werden.
Neben den lückenlosen Beweisen besteht die Eigenleistung dieser Arbeit in der Erweiterung des Algorithmus von Howgrave-Graham und Joux. Die beiden hier neu vorgestellten Algorithmen erweitern den Tradeoff zugunsten des Speicherverbrauchs und konnten auch für einen bestimmten Bereich bewiesen werden. Auf der anderen Seite des Tradeoffs konnte die Heuristik für $\beta < 0{,}2972$ bewiesen werden. Zusammenfassend wurde ein beweisbarer Tradeoff etabliert, der sich für β von 0,2285 bis 0,386 erstreckt. Zum Vergleich: Der beweisbare Bereich in [HGJ10] beginnt bei 0,2972 und endet vor 1/3.

9.2. Ausblick und offene Probleme

Ein generelles Ziel ist es natürlich, Algorithmen mit geringerer Komplexität zu finden. Auch neue Tradeoffs sind gefragt. In [BCJ11] haben Becker, Coron und Joux eine leicht verbesserte Version des Algorithmus von Howgrave-Graham und Joux vorgestellt. Die Idee hierbei ist, dass Lösungsteile nicht auf Vektoren über $\{0, 1\}$ beschränkt bleiben, sondern sie auf $\{-1, 0, 1\}$ zu erweitern, wodurch mehr Repräsentationen entstehen. Die Zeit- und Platzkomplexität ist nun bei $\tilde{\mathcal{O}}(2^{0{,}291n})$. Dieser Algorithmus könnte verbessert werden, indem der Platzverbrauch reduziert wird oder ein Tradeoff integriert wird.
In derselben Veröffentlichung sind ebenso zwei Algorithmen zu finden, die subexponentiellen Platzaufwand haben. Der eine basiert auf einem Schroeppel-Shamir-inspirierten Ansatz und hat eine Laufzeit von $\tilde{\mathcal{O}}(2^{0{,}75n})$ und der andere auf der Idee hinter dem Algorithmus von Howgrave-Graham und Joux; dieser hat eine Laufzeit von $\tilde{\mathcal{O}}(2^{0{,}72n})$. Beide Algorithmen verwenden Floyds Algorithmus zur Detektierung von Zyklen, vergleichbar zur Pollard-ρ-Methode.
Die neuen Algorithmen semi-entscheiden das Rucksackproblem lediglich. Hat eine Instanz keine Lösung, so terminieren sie nicht. Hier wäre das Ziel, einen schnellen Las-Vegas-Algorithmus zu finden, der zeigt, dass eine Instanz keine Lösung hat. Eng verwandt mit dieser Problemstellung ist die Frage, ob es bessere deterministische Algorithmen gibt. Bisher ist in diesem Bereich der

ursprüngliche Schroeppel-Shamir-Algorithmus nach wie vor unangefochten.
Da nicht zu erwarten ist, ohne Randomisierung gute Laufzeiten zu erreichen, besteht die Frage, wie viele ungünstige Instanzen[1] es gibt, bei denen die Komplexitätsschranken nicht eingehalten werden oder keine Lösung gefunden wird. Können solche Instanzen klassifiziert werden? Auch wäre es denkbar, das Problem gänzlich zu umgehen. Ein Beispiel für ungünstige Instanzen sind solche, deren Gewichte Vielfache von M sind. Wenn es nun immer so ist, dass ungünstige Instanzen von M abhängen, so wäre eine Lösung, den Algorithmus bei zu großem Ressourcenverbrauch mit einem anderen M neu zu starten. Wenn gezeigt werden kann, dass es nicht exponentiell viele M gibt, für die sich eine Instanz schlecht verhält, so kann dieses Problem eliminiert werden.
Da schwere Rucksackprobleme besonders für die Kryptographie interessant sind, stellt sich die Frage, welche Konsequenzen effizientere Algorithmen nach sich ziehen. Eine mögliche Folge wäre beispielsweise die Notwendigkeit, längere Schlüssel verwenden zu müssen. Für das prominente Beispiel NTRU wurden die Schlüssellängen jedoch konservativ genug gewählt, sodass der Algorithmus von Howgrave-Graham und Joux keine Verlängerung des Schlüssels nötig macht [HGJ10].

[1] engl.: *bad knapsacks*

A. Grundlagen

A.1. Datenstrukturen

In diesem und dem folgenden Abschnitt sollen einige Definitionen und Eigenschaften von und für Algorithmen und Datenstrukturen erläutert werden.
In der vorliegenden Arbeit treten zwei Datenstrukturen auf: Listen und Prioritätswarteschlangen.
Listen. Eine Liste ist eine Zusammenfassung von Elementen unter Berücksichtigung der Reihenfolge. Elemente können mehrfach vorkommen. Um auf das i-te Element einer Liste zuzugreifen, verwenden wir in der Notation eckige Klammern. Ist beispielsweise L eine Liste, so adressiert $L[5]$ das fünfte Element. Wir gehen davon aus, dass diese Listen statisch sind und der Zugriff auf Elemente in konstanter Zeit möglich ist. Das erinnert an das Konzept der Arrays, wie sie aus Programmiersprachen bekannt sind.
Prioritätswarteschlangen. Eine Warteschlange ist eine Datenstruktur, bei der stets nur auf das erste Element zugegriffen wird. Ist Q eine Warteschlange, so ist Zugriff also nur auf $Q[1]$ möglich. Durch Entnahme des ersten Elementes rücken alle weiteren um eine Position auf, wodurch sie sukzessive ebenso abrufbar werden. In regulären Warteschlangen gilt das FIFO-Prinzip. In Prioritätswarteschlangen ist das nicht so. Hier wird sichergestellt, dass zu jedem Zeitpunkt das kleinste (bzw. größte) vorgehaltene Element an der ersten Position gehalten wird. Bei entsprechender Implementierung sind Zugriffe effizient möglich. Zugriff auf das kleinste Element und Einfügen eines Elements ist in $\mathcal{O}(\log n)$ möglich(vgl. [CLRS01]).

A.2. Algorithmen

Für algorithmische Probleme gibt es drei Betrachtungsweisen. In der **Entscheidungsvariante** eines Problems ist nur nach der Existenz einer Lösung gefragt. In der **Suchvariante** ist die Lösung anzugeben. Im Fall von Subset Sum wollen wir darunter verstehen, dass in der Suchvariante alle Lösungen einer Instanz gefunden werden müssen. In der Entscheidungsvariante werden Algorithmen nach einer einzigen Lösung suchen, um zu zeigen, dass eine Instanz lösbar ist, jedoch nicht alle Lösungen aufzählen. Es gibt außerdem die **Optimierungsvariante**, die in dieser Arbeit allerdings keine Rolle spielt.
In der Entscheidungsvariante wird gefordert, dass ein Algorithmus entscheidet, ob eine Probleminstanz eine Lösung hat oder nicht; das heißt, sowohl die Ja- als auch die Nein-Antwort müssen ggf. gegeben werden. Diese Bedingung kann abgeschwächt werden, sodass nur die Ja-Antwort gefordert wird; im Nein-Fall muss der Algorithmus nicht terminieren.
Algorithmen können **deterministisch** oder **randomisiert** sein. Randomisierte Algorithmen

bedienen sich des Zufalls, um eine Lösung zu finden. Sie haben oft eine schlechte Worst-Case-Komplexität, doch gute Komplexitäten für den Average-Case.
Randomisierte Algorithmen lassen sich wiederum aufteilen in **Las-Vegas-Algorithmen** und **Monte-Carlo-Algorithmen**. Las-Vegas-Algorithmen geben stets ein korrektes Ergebnis aus. Die in dieser Arbeit behandelten randomisierten Algorithmen fallen hierunter. Monte-Carlo-Algorithmen dürfen auch ein falsches bzw. suboptimales Ergebnis liefern. Sie sind interessant für Optimierungsprobleme. Monte-Carlo-Algorithmen ohne abschätzbare Komplexität werden **heuristische Algorithmen** genannt. Es ist zu beachten, dass die hier behandelten Algorithmen zwar eine **Heuristik** (vgl. Kapitel 3) verwenden, jedoch nach dieser Definition keine heuristischen Algorithmen sind.

A.3. Stochastik

In dieser Arbeit werden stochastische Aussagen bewiesen. An dieser Stellen sollen die dafür nötigen Definitionen festgehalten werden, welche sich an [Geo09] orientieren.
Zufallsexperimente werden durch **Wahrscheinlichkeitsräume** modelliert, welche 3-Tupel sind:

$$\mathcal{W} = (\Omega, \mathcal{F}, P).$$

Dabei ist Ω der **Ereignisraum**, $\mathcal{F}$ eine σ-Algebra und $P\colon \mathcal{F} \to [0,1]$ eine Abbildung mit

$$P\left(\bigcup_{i=1}^{n} A_i\right) = \sum_{i=1}^{n} P(A_i),$$

wobei $A_i \in \mathcal{F}$ und paarweise disjunkt. Außerdem ist $P(\Omega) = 1$. Ist Ω diskret, so kann $\mathcal{F} = 2^{\Omega}$ gesetzt werden.
Eine **Zufallsvariable** $X\colon \Omega \to \Omega'$ ist eine beliebige Abbildung zwischen zwei Ereignisräumen. Ist $\mathcal{W} = (\Omega, \mathcal{F}, P)$ der zu Ω zugehörige Wahrscheinlichkeitsraum und ist $\mathcal{F}'$ zugehörig zu Ω', so ergibt sich der Wahrscheinlichkeitsraum $\mathcal{W}' = (\Omega', \mathcal{F}', P')$ durch P' so: $P'(A') = P(X^{-1}(A')) =: P(X = A)$, wobei $A' \in \mathcal{F}'$.
Der **Erwartungswert** einer Zufallsvariablen X definiert sich wie folgt:

$$\mathbb{E}(X) := \sum_{x \in X(\Omega)} xP(X = x).$$

Hierbei ist $X\colon \Omega \to \mathbb{R}$.
Die **Varianz** einer Zufallsvariable X ist ein Maß für die Streuung der Ereignisse. Es ist

$$\mathbb{V}(X) := \mathbb{E}((X - \mathbb{E}(x))^2) = \sum_{x \in X(\Omega)} (x - \mathbb{E}(X))^2 P(X = x).$$

Hieraus ergibt sich die **Standardabweichung** $\sigma(X) := \sqrt{\mathbb{V}(X)}$.

Satz A.1 (Tschebyschow) *Ist X eine Zufallsvariable und $\epsilon > 0$, so gilt*

$$P(|X - \mathbb{E}(X)| \geq \epsilon) \leq \frac{\mathbb{V}(X)}{\epsilon^2}.$$

Eine im Rahmen stochastischer Berechnungen wiederkehrende Gleichung ist eine Verallgemeinerung der Definition der Euler'schen Zahl:

$$\lim_{n\to\infty} \left(1 + \frac{k}{n}\right)^n = e^k$$

A.4. NP-Vollständigkeit von Subset Sum

Grundlagen zur Komplexitätstheorie und zur NP-Vollständigkeit finden sich in [Pap94]. Die NP-Vollständigkeit wurde in [Kar72] gezeigt.

Satz A.2 *Subset Sum ist NP-vollständig.*

Beweis. Subset Sum liegt in NP, denn es kann eine Lösung geraten und in Linearzeit verifiziert werden.
Um die NP-Schwere zu zeigen, reduzieren wir das NP-vollständige Problem 3KNF-SAT in polynomieller Zeit auf Subset Sum. Eine 3KNF-SAT-Formel F hat die Gestalt

$$F = \bigwedge_{i=1}^{m} (x_{i,1} \vee x_{i,2} \vee x_{i,3}),$$

wobei es n Literale gibt. Wir stellen F alternativ als Klauselmenge dar: $F = \bigcup_{i=1}^{m} \{K_m\}$. Gesucht ist jetzt eine Abbildung f mit

$$f(F) = (\mathbf{a}, c)$$

und der Eigenschaft, dass genau dann, wenn F erfüllbar ist, die Instanz $(\mathbf{a}, c)$ eine Lösung besetzt. Um die Konstruktion anzugeben, ist es bequem, von der vektoriellen Variante von Subset Sum auszugehen, was keine Einschränkung darstellt. Die Dimension muss dabei $m + n$ sein. Es werden $2n + 2m$ Gewichte und das Zielgewicht c benötigt. Die ersten $2n$ Gewichte korrespondieren zu der Menge der positiven und negativen Literale; die $2m$ Gewichte ergeben sich daraus, dass jeder Klausel zwei Gewichte zugeordnet werden. Die Literal-Gewichte werden so definiert:

$$a_{2k} := (\underbrace{0, \ldots, 0}_{k-1}, 1, \underbrace{0, \ldots, 0}_{n-k}, \underbrace{\chi_{K_1}(x_k), \ldots, \chi_{K_m}(x_k)}_{m})$$
$$a_{2k+1} := (\underbrace{0, \ldots, 0}_{k-1}, 1, \underbrace{0, \ldots, 0}_{n-k}, \underbrace{\chi_{K_1}(\neg x_k), \ldots, \chi_{K_m}(\neg x_k)}_{m})$$

Für $k \in \{1, \ldots, n\}$. Dabei ist χ_M die charakteristische Funktion einer Menge M. Die Klausel-Gewichte definieren wir so:

$$a_{2n+2k} := (\underbrace{0, \ldots, 0}_{n}, \underbrace{0, \ldots, 0}_{k-1}, 1, \underbrace{0, \ldots, 0}_{m-k})$$
$$a_{2n+2k+1} := (\underbrace{0, \ldots, 0}_{n}, \underbrace{0, \ldots, 0}_{k-1}, 2, \underbrace{0, \ldots, 0}_{m-k})$$

Für $k \in \{1, \ldots, m\}$. Es bleibt c zu definieren:

$$c := (\underbrace{1, \ldots, 1}_{n}, \underbrace{4, \ldots, 4}_{m})$$

Ist F erfüllbar, so besitzt die Instanz $f(F)$ eine Lösung. Eine erfüllende Belegung für F macht genau die positive oder negative Variante eines Literals wahr. In der Instanz werden hierzu n der ersten $2n$ Gewichte benötigt. Für alle $k \in \{1, \ldots, n\}$ ist entweder a_{2k} oder a_{2k+1} Teil der Lösung. Damit eine Belegung erfüllend ist, muss in jeder Klausel mindestens ein Literal wahr sein. Dies wird in der Instanz wie folgt modelliert: In den Literal-Gewichten spiegelt der hintere Teil die Formel wieder. Betrachtet man die Spalte, die einer Klausel zugehörig ist, so muss diese sich für die Lösung insgesamt zu 4 addieren, wie in c gefordert. In einer erfüllten Formel hat jede Klausel ein, zwei oder drei wahre Literale. Im Fall eines wahren Literals wird das entsprechende Literal-Gewicht in die Lösung übernommen. Hinzukommen die beiden zugehörigen Klausel-Gewichte 1 und 2. Die Summe ist 4. Die Fälle 2 und 3 lassen sich analog zeigen.
Besitzt die Instanz $f(F)$ eine Lösung, so kann hieraus direkt eine erfüllende Belegung für F abgelesen werden. □

A.5. Abschätzungen über den Binomialkoeffizienten

Der Binomialkoeffizient $\binom{n}{k}$ ist definiert als $\frac{n!}{(n-k)!\cdot k!}$. Wir wollen Koeffizienten der Form $\binom{n}{\alpha n}$ abschätzen mit $0 \leq \alpha \leq 1$. Für Fakultäten gilt die Stirlingsche Formel:

$$\lim_{n\to\infty} \frac{n!}{\sqrt{2\pi n}\left(\frac{n}{e}\right)^n} = 1$$

Somit gilt für große n, dass

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n.$$

Weiter erhalten wir:

$$\begin{aligned}\binom{n}{\alpha n} &= \frac{n!}{(n-\alpha n)!(\alpha n)!} \\ &= \frac{n!}{((1-\alpha)n)!(\alpha n)!} \\ &\approx \frac{\sqrt{2\pi n}\left(\frac{n}{e}\right)^n}{\sqrt{2\pi(1-\alpha)n}\left(\frac{(1-\alpha)n}{e}\right)^{(1-\alpha)n} \cdot \sqrt{2\pi\alpha n}\left(\frac{\alpha n}{e}\right)^{\alpha n}}\end{aligned}$$

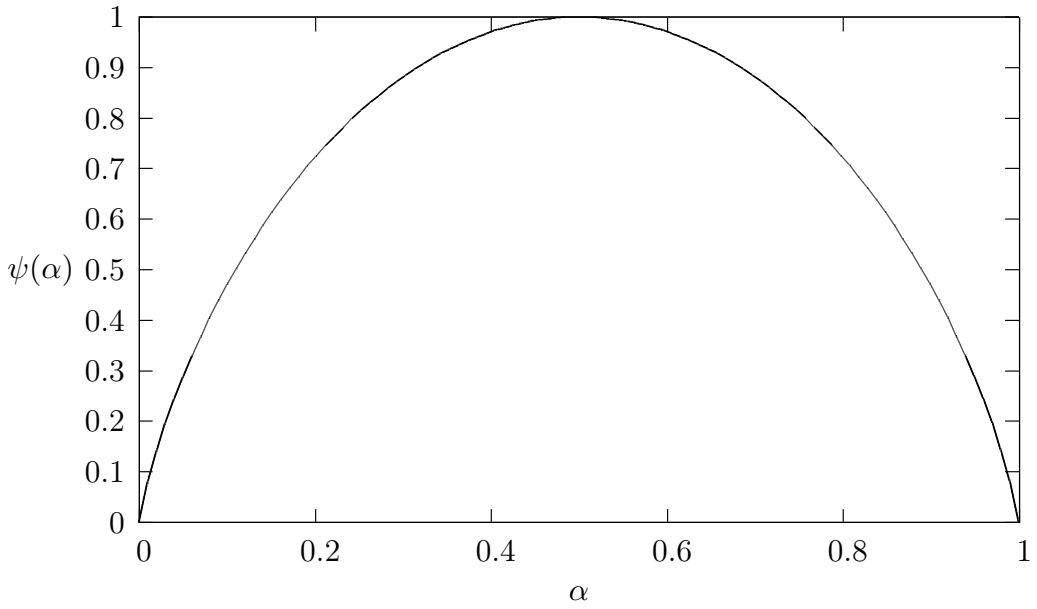

Abbildung A.1.: Graph der ψ-Funktion

Da wir in unserer Abschätzung polynomielle Faktoren unterschlagen wollen, können wir vereinfachen:

$$\begin{aligned}\frac{\left(\frac{n}{e}\right)^n}{\left(\frac{(1-\alpha)n}{e}\right)^{(1-\alpha)n} \cdot \left(\frac{\alpha n}{e}\right)^{\alpha n}} &= \frac{n^n}{((1-\alpha)n)^{(1-\alpha)n}(\alpha n)^{\alpha n}} \\ &= \frac{1}{\frac{(1-\alpha)^n}{(1-\alpha)^{\alpha n}}\alpha^{\alpha n}} \\ &= \left(\frac{(1-\alpha)^\alpha}{(1-\alpha)\alpha^\alpha}\right)^n\end{aligned}$$

Zusammenfasssend haben wir also

$$\binom{n}{\alpha n} \in \tilde{\mathcal{O}}\left(\left(\frac{1}{(1-\alpha)^{1-\alpha}\alpha^\alpha}\right)^n\right) = \tilde{\mathcal{O}}(2^{\psi(\alpha)n})$$

Die Komplexitäten werden zur Basis 2 angegeben. α nimmt oft typische wiederkehrende Werte an, weshalb einige Beispielwerte aufgelistet werden. Die Funktion $\psi\colon [0,1] \to [0,1]$ findet in dieser Arbeit an verschiedenen Stellen Verwendung. Explizit angegeben ist sie:

$$\alpha \mapsto \log_2\left(\frac{(1-\alpha)^{(\alpha-1)}}{\alpha^\alpha}\right)$$

Tabelle A.1 enthält einige relevante Beispielwerte und Bild A.1 ist der Graph der ψ-Funktion.

α	$\psi(\alpha)$
$1/2$	1
$1/3$	$0{,}91829583\ldots$
$1/4$	$0{,}81127812\ldots$
$1/5$	$0{,}72192809\ldots$
$1/6$	$0{,}65002242\ldots$
$1/7$	$0{,}59167277\ldots$
$1/8$	$0{,}54356444\ldots$
$1/16$	$0{,}36517884\ldots$
$1/32$	$0{,}20062232\ldots$

Tabelle A.1.: Beispielwerte für ψ

B. Tradeoff für den Schroeppel-Shamir-Algorithmus

2011 präsentierten Joux et al. einen Tradeoff für den Schroeppel-Shamir-Algorithmus [BCJ11]. Dieser soll nun hier wiedergegeben werden.
Der nicht-randomisierte Grundalgorithmus stellt dabei die Version mit dem größten Platzverbrauch dar. Durch den Tradeoff kann die Platzkomplexität zu Lasten der Laufzeit reduziert werden. Wir stellen den Grundalgorithmus noch einmal vereinfacht dar (Algorithmus B.1). Die Prioritätsschlangenkonstruktion ist hier nebensächlich. Dabei sind die Listen den vier Vierteln der Instanz zugeordnet:

$$
\begin{aligned}
L_1 &= \{\mathbf{av} \,|\, \mathbf{v} \in \mathbb{B}^n, \forall_{\frac{n}{4}<i\leq n} : v_i = 0\} \\
L_2 &= \{\mathbf{av} \,|\, \mathbf{v} \in \mathbb{B}^n, \forall_{1\leq i\leq \frac{n}{4}} : v_i = 0, \forall_{\frac{n}{4}<i\leq n} : v_i = 0\} \\
L_3 &= \{c - \mathbf{av} \,|\, \mathbf{v} \in \mathbb{B}^n, \forall_{1\leq i\leq \frac{n}{2}} : v_i = 0, \forall_{\frac{3n}{4}<i\leq n} : v_i = 0\} \\
L_4 &= \{-\mathbf{av} \,|\, \mathbf{v} \in \mathbb{B}^n, \forall_{1\leq i\leq \frac{3n}{4}} : v_i = 0\}
\end{aligned}
$$

Verkleinern wir nun die Listen durch die bekannte Heuristik, so verringert sich die Platz- und Zeitkomplexität, jedoch muss der Algorithmus exponentiell oft angewandt werden, bis die Lösung gefunden wurde.
Sei β die Stellvariable. Es ist $M = 2^{\beta n}$. R_1, R_2 und R_3 werden gewählt, um daraus die Listen zu berechnen; dabei ist $R_4 := c - R_1 - R_2 - R_3$. Wir definieren die neuen Listen:

$$
\begin{aligned}
L_1^{R_1} &= \{\mathbf{av} \,|\, \mathbf{v} \in \mathbb{B}^n, \forall_{\frac{n}{4}<i\leq n} : v_i = 0, \mathbf{av} \equiv R_1 \pmod M\} \\
L_2^{R_2} &= \{\mathbf{av} \,|\, \mathbf{v} \in \mathbb{B}^n, \forall_{1\leq i\leq \frac{n}{4}} : v_i = 0, \forall_{\frac{n}{4}<i\leq n} : v_i = 0, \mathbf{av} \equiv R_2 \pmod M\} \\
L_3^{R_3} &= \{c - \mathbf{av} \,|\, \mathbf{v} \in \mathbb{B}^n, \forall_{1\leq i\leq \frac{n}{2}} : v_i = 0, \forall_{\frac{3n}{4}<i\leq n} : v_i = 0, \mathbf{av} \equiv R_3 \pmod M\} \\
L_4^{R_4} &= \{-\mathbf{av} \,|\, \mathbf{v} \in \mathbb{B}^n, \forall_{1\leq i\leq \frac{3n}{4}} : v_i = 0, \mathbf{av} \equiv R_4 \pmod M\}
\end{aligned}
$$

Dies führt direkt zu Algorithmus B.2.
Korrektheit und Vollständigkeit sind offensichtlich gegeben.
Platzkomplexität. Um die Listen $L_1^{R_1}$ bis $L_4^{R_4}$ mit dem Schroeppel-Shamir-Algorithmus zu

Algorithmus B.1 Der Algorithmus von Schroeppel und Shamir (vereinfacht dargestellt)

Erstelle L_1 bis L_4
Sortiere L_1 bis L_4
Finde Kollision zwischen $L_1 + L_2$ und $L_3 + L_4$ mithilfe der Prioritätswarteschlange

Algorithmus B.2 Der Algorithmus von Schroeppel und Shamir mit Tradeoff

$M \leftarrow 2^{\beta n}$
for all $R_1, R_2, R_3 \in \{0, \ldots, M-1\}$ **do**
  Erstelle $L_1^{R_1}$ bis $L_4^{R_4}$ mithilfe der Grundversion des Schroeppel-Shamir-Algorithmus
  Sortiere $L_1^{R_1}$ bis $L_4^{R_4}$
  Finde Kollision zwischen $L_1^{R_1} + L_2^{R_2}$ und $L_3^{R_3} + L_4^{R_4}$ mithilfe der Prioritätswarteschlange
end for

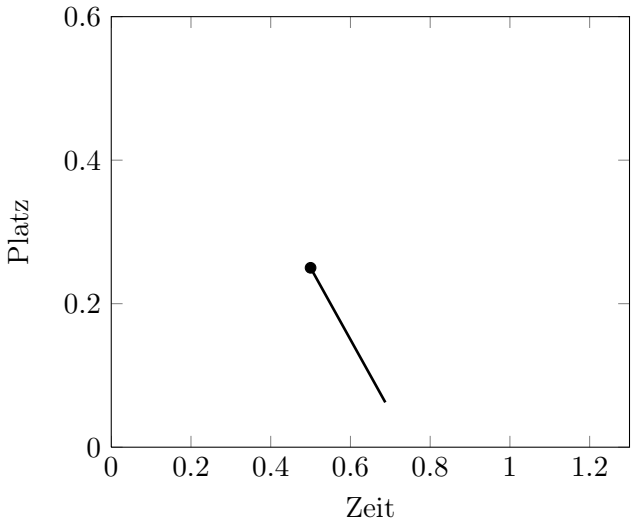

Abbildung B.1.: Tradeoff des Schroeppel-Shamir-Algorithmus

berechnen, benötigt dieser $\tilde{\mathcal{O}}(2^{n/16})$. Die Listen haben nach Satz 5.2 die Größe $\tilde{\mathcal{O}}(2^{(1/4-\beta)n})$. Die Kollisionsfindung durch die Prioritätswarteschlange funktioniert innerhalb dieser Platzgrenzen. Zusammenfassend ist die Platzkomplexität

$$\tilde{\mathcal{O}}(\max\{2^{n/16}, 2^{(1/4-\beta)n}\}).$$

Hieraus sieht man direkt, dass β sinnvollerweise kleiner $3/16$ ist.
Zeitkomplexität. Um die richtigen Werte für R_1 bis R_3 zu erlangen, werden alle durchprobiert. Das sind also $2^{3\beta n}$ Schleifendurchläufe. Je Schleifendurchlauf benötigt die Erstellung der Listen $\tilde{\mathcal{O}}(2^{1/8n})$ Zeit. Die Kollisionsfindung ist quadratisch in der Größe der Listen und benötigt damit $\tilde{\mathcal{O}}(2^{2(1/4-\beta)n})$ Zeit. Da $\beta \leq 3/16$ ist dies insgesamt

$$\tilde{\mathcal{O}}(2^{(1/2+\beta)n}).$$

Dieser Tradeoff ist in Bild B.1 dargestellt. Wie man sieht, ist es jetzt möglich, den Platzverbrauch bis auf $\tilde{\mathcal{O}}(2^{n/16})$ zu reduzieren.

Literaturverzeichnis

[BCJ11] A. Becker, J.-S. Coron, A. Joux. Improved Generic Algorithms for Hard Knapsacks. In K. G. Paterson, Herausgeber, *EUROCRYPT*, Band 6632 von *Lecture Notes in Computer Science*, S. 364–385. Springer, 2011. (Zitiert auf den Seiten 43, 48, 55 und 63)

[CJL+92] M. J. Coster, A. Joux, B. A. Lamacchia, A. M. Odlyzko, C.-P. Schnorr, J. Stern. Improved low-density subset sum algorithms. *Computational Complexity*, 2:111–128, 1992. (Zitiert auf Seite 9)

[CLRS01] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein. *Introduction to Algorithms*. The MIT Press, 2nd Auflage, 2001. (Zitiert auf Seite 57)

[Geo09] H.-O. Georgii. *Stochastik: Einführung in die Wahrscheinlichkeitstheorie und Statistik*. Walter de Gruyter, 4 Auflage, 2009. (Zitiert auf Seite 58)

[HG07] N. Howgrave-Graham. A Hybrid Lattice-Reduction and Meet-in-the-Middle Attack Against NTRU. In *CRYPTO*, S. 150–169. 2007. (Zitiert auf Seite 6)

[HGJ10] N. Howgrave-Graham, A. Joux. New Generic Algorithms for Hard Knapsacks. In H. Gilbert, Herausgeber, *EUROCRYPT*, Band 6110 von *Lecture Notes in Computer Science*, S. 235–256. Springer, 2010. (Zitiert auf den Seiten 11, 18, 31, 33, 37, 43, 53, 54, 55 und 56)

[HPS98] J. Hoffstein, J. Pipher, J. H. Silverman. NTRU: A Ring-Based Public Key Cryptosystem. In *ANTS*, S. 267–288. 1998. (Zitiert auf Seite 6)

[HS74] E. Horowitz, S. Sahni. Computing Partitions with Applications to the Knapsack Problem. *J. ACM*, 21:277–292, 1974. URL http://doi.acm.org/10.1145/321812.321823. (Zitiert auf Seite 25)

[IN96] R. Impagliazzo, M. Naor. Efficient Cryptographic Schemes Provably as Secure as Subset Sum. *J. Cryptology*, 9(4):199–216, 1996. (Zitiert auf Seite 9)

[Kar72] R. Karp. Reducibility among combinatorial problems. In R. Miller, J. Thatcher, Herausgeber, *Complexity of Computer Computations*, S. 85–103. Plenum Press, 1972. (Zitiert auf den Seiten 5 und 59)

[LMPR08] V. Lyubashevsky, D. Micciancio, C. Peikert, A. Rosen. SWIFFT: A Modest Proposal for FFT Hashing. In K. Nyberg, Herausgeber, *FSE*, Band 5086 von *Lecture Notes in Computer Science*, S. 54–72. Springer, 2008. (Zitiert auf Seite 6)

[MMI+78] R. C. Merkle, S. Member, Ieee, M. E. Hellman, S. Member. Hiding information and signatures in trapdoor knapsacks. *IEEE Transactions On Information Theory*, 24:525–530, 1978. (Zitiert auf Seite 6)

[NSS00] P. Q. Nguyen, I. E. Shparlinski, J. Stern. Distribution Of Modular Sums And The Security Of The Server Aided Exponentiation, 2000. (Zitiert auf den Seiten 11, 13 und 55)

[Pap94] C. H. Papadimitriou. *Computational complexity.* Addison-Wesley, 1994. (Zitiert auf Seite 59)

[Sha84] A. Shamir. A polynomial-time algorithm for breaking the basic Merkle-Hellman cryptosystem. *IEEE Transactions on Information Theory*, 30(5):699–704, 1984. (Zitiert auf Seite 6)

[SS81] R. Schroeppel, A. Shamir. A $T = O(2^{n/2})$, $S = O(2^{n/4})$ Algorithm for Certain NP-Complete Problems. *SIAM Journal on Computing*, 10(3):456–464, 1981. doi: 10.1137/0210033. URL http://link.aip.org/link/?SMJ/10/456/1. (Zitiert auf den Seiten 29 und 31)

Erklärung

Hiermit versichere ich, diese Arbeit selbständig verfasst und nur die angegebenen Quellen benutzt zu haben.

(Michael Ludwig)