Komplexität der Matrizen-Multiplikation

Anselm Haak

16.08.2013

Erklärung

Hiermit versichere ich, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

_____

Inhaltsverzeichnis

1 Einleitung

Die Matrizen-Multiplikation (im Folgenden MM) ist sicherlich eines der ältesten und grundlegendsten Probleme der Computational Complexity. Eine der wichtigsten Fragen in Bezug auf dieses Problem ist, wie viele arithmetische Operationen für die Multiplikation zweier beliebig großer $n \times n$-Matrizen benötigt werden. Interessant ist diese Frage, da die Anzahl der arithmetischen Operationen direkt den Rechenaufwand ergibt. Weiterhin ist die MM nicht nur unmittelbar von Interesse: Es gibt effiziente Reduktionen anderer Probleme auf die MM, weshalb effiziente Algorithmen für die MM noch weitläufiger von Bedeutung sind. Zu nennen sind dabei einerseits andere Probleme im Umfeld der Matrizen - die Bestimmung der Inversen einer Matrix, die Berechnung der Determinante einer Matrix, das Lösen linearer Gleichungssysteme -, aber auch zum Beispiel die Bestimmung der transitiven Hülle eines Graphen.
Erst kürzlich, genauer gesagt 2010, veröffentlichten Andrew Stothers in [Sto10] und Virginia Vassilevska Williams in [Wi10] neue Ergebnisse zum Problem der MM, was den Anstoß dazu gab, meine Arbeit diesem Thema zu widmen.
Es gibt bereits viele Publikationen auf diesem Gebiet und auch einige Zusammenfassungen bisheriger Ergebnisse. So veröffentlichte Victor Ya. Pan mit [Pan81] und [Pan84] zwei Zusammenfassungen, wobei erstere deutlich ausführlicher ausfiel und viele Resultate bewiesen wurden, während letztere eher einen guten Überblick gibt. Auch Stothers und Williams haben ihren neuen Publikationen Zusammenfassungen der alten Ergebnisse vorangestellt. Dazu kam noch eine Weitere von Williams, die sie auf einer Konferenz 2012 vorstellte, siehe [Wi12].
In Abgrenzung zu diesen Publikationen soll in dieser Arbeit besonderer Fokus auf die Verständlichkeit gelegt werden: Es sollen alle nötigen Grundlagen erläutert und erst dann auf die Ergebnisse, die daraus resultieren, eingegangen werden. Gerade bei den späteren Resultaten wird dabei kein Platz für detaillierte Beweise sein, es wird aber dennoch zumindest der grobe Ablauf des Beweises beschrieben und auf wichtige Ideen eingegangen. Die Publikationen, in denen weitere Details und insbesondere detaillierte Beweise zu finden sind, werden selbstverständlich referenziert.
Bei alledem beschränkt sich diese Arbeit größtenteils auf den Hauptstrang der Entwicklungen auf diesem Gebiet, also diejenigen Ergebnisse, die einzeln oder durch Kombination zu irgendeinem Zeitpunkt neue obere Schranken für die mindestens benötigten arithmetischen Operationen lieferten und die auch wichtig für das Verständnis der jüngsten Ergebnisse sind.
Daneben gibt es auch andere Ansätze und weitere Betrachtungen, auf die aber nur am Rande eingegangen werden kann. Für Interessierte werden aber entsprechende Referenzen angegeben. Auch Resultate in Bezug auf untere Schranken für ω finden nur kurz in (Abschnitt 4) Erwähnung.
In Bezug auf die Praxistauglichkeit sei noch gesagt, dass der Großteil der Verfahren einen sehr großen Overhead erzeugt. Zudem sind die Verfahren häufig für kleine Matrixgrößen gar nicht anwendbar. Dies geht soweit, dass meines Wissens nur der erste nicht-triviale Algorithmus tatsächlich angewendet wird - alle anderen sind für realistische Eingabegrößen nicht geeignet. Dennoch sind die Entwicklungen auf diesem Gebiet auch praktisch von Bedeutung, da sie immer tiefere Einblicke in die Struktur des Problems und auch verwandter Probleme liefern und damit schlussendlich unter Umständen auch zu besseren, anwendbaren Algorithmen führen könnten.
Im folgenden Abschnitt werden einige Grundlagen erläutert, begonnen wird mit der Definition des betrachteten Problems.

2 Grundlagen

In diesem Abschnitt werden Begriffe und Konzepte eingeführt und definiert, die für das Verständnis der anschließend vorgestellten Resultate benötigt werden oder zumindest hilfreich sind. Auf Tensoren könnte auch verzichtet werden, allerdings würden dadurch viele Formulierungen und Darstellungen aufwendiger. Da zudem in den bisherigen Publikationen Grundkenntnisse in Bezug auf Tensoren fast ausnahmslos vorausgesetzt werden, sollte dieser Abschnitt auch die Arbeit mit diesen Publikationen erleichtern. Neben den grundlegenden Definitionen wird dabei jeweils auch die Verbindung zur MM hergestellt.

2.1 Matrizen-Multiplikation und Exponent der Matrizen-Multiplikation

Das Problem der MM ist wie folgt definiert:

Definition 1. *Das Problem der MM ist für feste m,n,p wie folgt definiert:*
***Eingabe**: $m \times n$-Matrix A und $n \times p$-Matrix B über einem Körper K, $A = (a_{ij}), B = (b_{jk})$, $i \in \{1,\dots,m\}, j \in \{1,\dots,n\}, k \in \{1,\dots,p\}$.*
***Ausgabe**: Das Produkt $C = AB, C = (c_{ik})$, wobei die c_{ik} gegeben sind durch:*

$$c_{ik} = \sum_{j=1}^{n} a_{ij} \cdot b_{jk} \ \forall i,k \text{ wie oben.}$$

Dieses Problem bezeichnen wir im Folgenden mit $\underline{(m,n,p)}$.

Für diese Berechnung entscheidend ist die Anzahl der arithmetischen Operationen. Deshalb definieren wir den Exponenten der MM wie folgt:

Definition 2. *Sei* $\omega := inf\left\{ \gamma \;\middle|\; \begin{array}{c} \text{Es gibt Algorithmen für das Problem } (n,n,n), \\ \text{die } O(n^\gamma) \text{ arithmetische Operationen benötigen.} \end{array} \right\}$.
ω heißt <u>Exponent der Matrizen-Multiplikation</u>.

Dabei ist zu beachten, dass wir nicht fordern, dass der Exponent exakt erreicht wird, sondern nur, dass Algorithmen existieren, die beliebig nah an den Exponenten herankommen.
Das Ziel ist damit die Bestimmung von ω. Auf dem Weg zu diesem Ziel werden untere und obere Schranken für ω gesucht, wobei besonders die Suche nach Letzteren in der Vergangenheit sehr fruchtbar war: Im Verlauf der Zeit wurde der Exponent von 3 vor dem Jahr 1968 (für den trivialen Algorithmus) bzw. 2,807 im Jahr 1968 durch Strassen bis auf 2,3727 durch Williams im Jahr 2010 gesenkt.
Eine erste Vereinfachung bei der Suche nach besseren Algorithmen ist die Folgende: Wie Volker Strassen in [Str73] zeigt, müssen keine Algorithmen betrachtet werden, die Divisionen verwenden. Somit bleiben Multiplikationen und Additionen als arithmetische Operationen.

2.2 Multilineare Abbildungen und Multilinearformen

Eine multilineare Abbildung ist gewissermaßen die Verallgemeinerung einer linearen Abbildung auf mehrere Argumente, d.h. es ist eine Abbildung in mehreren Argumenten aus verschiedenen Vektorräumen über demselben Körper, die sich in jedem Argument linear verhält, wenn alle anderen Argumente festgehalten werden.

Definition 3. *Sei K ein Körper, $V_1,\dots,V_n,X$ K-Vektorräume. Eine Abbildung $\varphi: V_1\times\cdots\times V_n\to X$ heißt multilinear, wenn gilt:*
Seien $v_i,w_i\in V_i$ für alle $i\in\{1,\dots,n\}$.

$$f.a.\ i\in\{1,\dots,n\}:\varphi(v_1,\dots,v_i+w_i,\dots,v_n)=\varphi(v_1,\dots,v_i,\dots,v_n)+\varphi(v_1,\dots,w_i,\dots,v_n)$$

und für v_i wie oben und $a\in K$:

$$\forall i\in\{1,\dots,n\}:\varphi(v_1,\dots,a\cdot v_i,\dots,v_n)=a\cdot\varphi(v_1,\dots,v_i,\dots,v_n).$$

Eine Multilinearform ist eine multilineare Abbildung, die nicht in einen beliebigen Vektorraum X, sondern in den zugrunde liegenden Körper K abbildet.
Für die Spezialfälle von zwei und drei Argumenten werden die Begriffe bilineare Abbildung, Bilinearform, trilineare Abbildung und Trilinearform verwendet. Diese sind für uns im Weiteren von Bedeutung.
Eine Menge von verschiedenen Multilinearformen, deren Argumente aus den gleichen Vektorräumen stammen, lässt sich auch als eine multilineare Abbildung in einen Vektorraum auffassen, dessen Dimension genau der Anzahl der Multilinearformen entspricht und umgekehrt. Es wird also jede Komponente des Bild-Vektors (bzgl. einer Basis) einzeln betrachtet, sodass es sich um einzelne Abbildungen in den zugrunde liegenden Körper handelt.
Die MM, wie sie oben definiert wurde, ist damit eine bilineare Abbildung, da die Matrizen jeweils Vektoren aus Vektorräumen über demselben Körper K sind. Die Bilinearität lässt sich dabei leicht nachrechnen. Alternativ lässt sich diese bilineare Abbildung (s.o.) auch als Menge von Bilinearformen interpretieren, was für uns nützlicher ist. Auf diese Art lässt sich aber immer nur ein festes Problem (m,n,p) (über einem festen Körper) beschreiben.
Für die Nutzung von Bilinearformen und Trilinearformen für die MM werden folgende Begriffe von Bedeutung sein.

Definition 4. *Sei $\{\varphi_k \mid k\in\{1,\dots,p\}\}$ für ein $p\in\mathbb{N}$ eine Menge von Bilinearformen mit $\varphi_k: U\times V\to K$, K Körper, U und V K-Vektorräume von Dimensionen n und m.*
Seien $i\in\{1,\dots,n\},j\in\{1,\dots,m\},k\in\{1,\dots,p\}$ und $u=(u_i)\in U,v=(v_j)\in V$.
Seien nun $L_q^1=\sum_i\alpha_{i,q}u_i, L_q^2=\sum_j\beta_{j,q}v_j$ Linearkombinationen und $\alpha_{i,q},\beta_{j,q},\gamma_{k,q}\in K$ konstante Koeffizienten. Dann heißt die Darstellung

$$\varphi_k(u,v)=\sum_{q=1}^{r}\gamma_{k,q}L_q^1L_q^2\ \forall k$$

bilinearer Algorithmus der Menge von Bilinearformen und r heißt Rang oder auch Länge dieses Algorithmus.

Definition 5. *Sei $\varphi: U\times V\times W\to K$ eine Trilinearform, K,U,V wie oben und sei W K-Vektorraum von Dimension p.*
Seien i,j,k,u,v wie oben und $w=(w_k)\in W$.

Seien auch L_q^1, L_q^2 wie oben und $L_q^3 = \sum_k \gamma_{k,q} w_k$ Linearkombination mit konstanten Koeffizienten $\gamma_{k,q} \in K$. Dann heißt die Darstellung

$$\varphi(u,v,w) = \sum_{q=1}^{r} L_q^1 L_q^2 L_q^3$$

trilinearer Algorithmus der Trilinearform φ und r heißt Rang oder Länge dieses Algorithmus.

Zur besseren Übersichtlichkeit gerade in Hinblick auf die korrespondierenden Begriffe für Tensoren verwenden wir für bilineare und trilineare Algorithmen den Begriff Länge, nicht Rang.

Definition 6. *Die kleinste Länge r eines beliebigen bilinearen Algorithmus für eine Menge von Bilinearformen oder auch eines beliebigen trilinearen Algorithmus für eine Trilinearform heißt minimaler Rang oder kurz Rang dieser Menge von Bilinearformen bzw. dieser Trilinearform.*

2.3 Bilineare und trilineare Algorithmen für die Matrizen-Multiplikation und ihr Rang

Um die Interpretation der MM als Menge von Bilinearformen für die Suche nach Schranken für ω nutzbar zu machen, verwenden wir obige bilineare und trilineare Algorithmen. Allerdings schreiben wir diese passend zu den üblichen Basen für Vektorräume der Matrizen bestimmter Größe, also mit passenden Multi-Indizes.
Wir schreiben einen bilinearen Algorithmus für das Problem (m,n,p) folgendermaßen und definieren einen passenden Begriff:

Definition 7. *Seien $A = (a_{ij}), B = (b_{jk}), C = A \cdot B = (c_{ik})$ Matrizen entsprechender Größen, $\alpha_{ij,q}, \beta_{jk,q}, \gamma_{ik,q} \in K$ konstante Koeffizienten und $L_q^1 = \sum_{i,j} \alpha_{ij,q} a_{ij}, L_q^2 = \sum_{j,k} \beta_{jk,q} b_{jk}$ Linearkombinationen. Dann ist*

$$c_{ik} = \sum_{q=1}^{r} \gamma_{ik,q} L_q^1 L_q^2 \ \forall i,k$$

ein bilinearer Algorithmus für (m,n,p), also ein bilinearer Algorithmus der MM.

Damit wird ein Algorithmus für ein festes Problem (m,n,p) beschrieben.
Wir wollen nun zu einer Darstellung als eine Trilinearform übergehen. Anschließend werden wir sehen, dass sich trilineare Algorithmen dieser Trilinearform und bilineare Algorithmen der MM ineinander überführen lassen.
Die folgende Trilinearform beschreibt die Spur $Tr(ABC)$ des Matrixproduktes ABC, also die Summe aller Diagonaleinträge dieses Matrixproduktes, wobei A, B, C Matrizen geeigneter Größen über einem gemeinsamen Körper K sind:

$$Tr(ABC) = \sum_{i,j,k} a_{ij} \cdot b_{jk} \cdot c_{ki}$$

Die Korrektheit dieser Gleichung lässt sich wieder leicht nachrechnen.
Aufgrund der anschließend gezeigten Äquivalenz der Rangbegriffe sagen wir auch, dass diese Trilinearform für Matrizen der Größen $m \times n$, $n \times p$ und $p \times m$ zum Problem (m,n,p) korrespondiert.
Wir schreiben trilineare Algorithmen dieser Trilinearform (für beliebige m,n,p) ebenfalls angepasst und definieren auch hier einen passenden Begriff:

Definition 8. *Seien $A=(a_{ij}), B=(b_{jk}), C=(c_{ik})$ Matrizen entsprechender Größen, $\gamma_{ik,q} \in K$ konstante Koeffizienten. Seien L_q^1, L_q^2 wie in (Definition 7) und $L_q^3 = \sum_{k,i} \gamma_{ki,q} c_{ki}$ Linearkombinationen. Dann ist*

$$\sum_{i,j,k} a_{ij} b_{jk} c_{ki} = \sum_{q=1}^{r} L_q^1 L_q^2 L_q^3$$

ein trilinearer Algorithmus der zu (m,n,p) korrespondierenden Trilinearform, also ein trilinearer Algorithmus der MM.

Beachte, dass zwischen den beiden Definitionen die Indizes der c's und deshalb auch die Indizes der γ's ihre Rollen getauscht haben. Abgesehen davon ist der Unterschied, dass nun die $\gamma_{ij,q}$ als konstante Koeffizienten von Linearkombinationen der Einträge von C auftauchen und diese Linearkombinationen statt der konstanten $\gamma_{ij,q}$ für einzelne i,k auftreten. Außerdem wird über alle i,k summiert. Nun kommen wir zur Äquivalenz der beiden Rang-Begriffe.

Satz 1. *Es gibt einen bilinearen Algorithmus für das Problem (m,n,p) von Länge r genau dann, wenn es einen trilinearen Algorithmus für die Berechnung von $Tr(ABC)$ für Matrizen A,B,C der Größen $m \times n$, $n \times p$, $p \times m$ von Länge r gibt.*
Die kleinste solche Länge nennen wir den vorherigen Definitionen entsprechend den Rang des Problems (m,n,p).

Beweis. "$\Rightarrow$": Es existiert also ein bilinearer Algorithmus von Länge r für Matrizen obiger Größen. Wir nennen die Einträge der Matrix AB hier, um eine Unterscheidung zu den Einträgen der Matrix C für die Trilinearform zu haben, $d_{ik} := \sum_j a_{ij} b_{jk}$. Der bilineare Algorithmus ist also gegeben durch $d_{ik} = \sum_{q=1}^{r} \gamma_{ik,q} L_q^1 L_q^2 \ \forall i,k$, wobei wie oben $\gamma_{ik,q}$ Konstanten und L_q^1, L_q^2 Linearkombinationen der a_{ij} bzw. b_{jk} sind. Gesucht ist nun ein Algorithmus von gleicher Länge r, der $\sum_{i,j,k} a_{ij} b_{jk} c_{ki}$ berechnet, also ein trilinearer Algorithmus.
Schreibe dazu $\sum_{i,j,k} a_{ij} b_{jk} c_{ki} = \sum_{i,k} (\sum_j a_{ij} b_{jk}) c_{ki} = \sum_{i,k} d_{ik} c_{ki}$.
Damit erhalten wir:

$$\begin{aligned} d_{ik} c_{ki} &= (\sum_{q=1}^{r} \gamma_{ik,q} L_q^1 L_q^2) \cdot c_{ki} \\ &= \sum_{q=1}^{r} L_q^1 L_q^2 \cdot \gamma_{ik,q} c_{ki} \end{aligned}$$

und durch Summieren:

$$\begin{aligned} \sum_{i,k} d_{ik} c_{ki} &= \sum_{i,k} \sum_{q=1}^{r} L_q^1 L_q^2 \gamma_{ik,q} c_{ki} \\ &= \sum_{q=1}^{r} L_q^1 L_q^2 (\sum_{i,k} \gamma_{ik,q} c_{ki}) \\ &= \sum_{q=1}^{r} L_q^1 L_q^2 L_q^3 \text{ mit } L_q^3 = \sum_{i,k} \gamma_{ik,q} c_{ki} \end{aligned}$$

Zu beachten ist, dass die $\gamma_{ik,q}$ und die c_{ki} für die gleichen i,k definiert sind.

"$\Leftarrow$": Für die Rückrichtung existiert ein trilinearer Algorithmus von Länge r für Matrizen obiger Größen. Dieser ist gegeben durch $\sum_{i,j,k} a_{ij} b_{jk} c_{ki} = \sum_{q=1}^{r} L_q^1 L_q^2 L_q^3$, wobei wie oben L_q^1, L_q^2, L_q^3 Linearkombinationen der a_{ij}, b_{jk} bzw. c_{ki} sind.

Durch Umstellen erhalten wir

$$\sum_{i,j,k} a_{ij}b_{jk}c_{ki} = \sum_{q=1}^{r} L_q^1 L_q^2 L_q^3$$
$$\sum_{k,i}(\sum_{j} a_{ij}b_{jk})c_{ki} = \sum_{k,i}(\sum_{q=1}^{r} L_q^1 L_q^2 \gamma_{ki,q})c_{ki}$$

Auch hier ist zu beachten: $\gamma_{ki,q}$ ist definiert für alle i,k, die von der Summe durchlaufen werden. Damit können für den gesuchten bilinearen Algorithmus die L_q^1 und L_q^2 direkt übernommen werden und die $\gamma_{ki,q}$ ergeben sich durch Koeffizientenvergleich für die einzelnen c_{ki}. □

2.4 Tensorprodukt und Tensoren

Das Tensorprodukt ist ein Konstrukt, das sich bei dem Versuch ergibt, multilineare Abbildungen auf endliche Weise zu beschreiben, ähnlich zur Beschreibung linearer Abbildungen durch Matrizen. Der Übersichtlichkeit halber beschränke ich mich auf den Fall bilinearer Abbildungen statt allgemeiner multilinearer Abbildungen. Für multilineare Abbildungen kann aber völlig analog argumentiert werden.
In diesem Abschnitt soll das Konzept des Tensorproduktes vor allem verständlich und anschaulich gemacht werden[1], Beweise werden größtenteils weggelassen, da es sich um ein eigenes Teilgebiet der linearen Algebra handelt und zudem um absolute Grundlagen.
Die Idee bei der Konstruktion einer solchen endlichen Darstellung ist, wie bei der Darstellung linearer Abbildungen durch Matrizen, nur die Funktionswerte einiger weniger Eingaben zu beschreiben, allerdings sicherzustellen, dass sich durch die Bilinearität alle weiteren Funktionswerte daraus ergeben. Das heißt, es müssen die Funktionswerte gewisser Tupel (v,w) mit $v \in V, w \in W$ angegeben werden.
Wir beschränken uns nun auf Bilinearformen statt allgemeiner bilinearer Abbildungen, da bilineare Abbildungen als Menge von Bilinearformen aufgefasst werden können und insofern nicht separat behandelt werden müssen.
Dabei ist die Idee, die sich letztlich als richtig herausstellen wird, relativ leicht nachvollziehbar: Für ein festes $w \in W$ ist die Abbildung eine lineare Abbildung auf V, wir können diese also durch die Funktionswerte aller Basisvektoren von V beschreiben. Selbiges gilt umgekehrt als lineare Abbildung auf W für ein festes $v \in V$. Entsprechend ist es intuitiv nachvollziehbar, dass wir Funktionswerte für alle Kombinationen von je einem Basisvektor von V und einem Basisvektor von W festlegen müssen. Wir beschreiben diese Idee nun formal und ergänzen sie um weitere Beschreibungen des daraus resultierenden Vektorraumes.
Sei im Folgenden $\varphi : V \times W \to K$ eine Bilinearform. Wir interpretieren alle Tupel (v,w) mit $v \in V, w \in W$ als Funktionen auf der Menge der Bilinearformen, die wir hier $\mathcal{B}$ nennen wollen, in den zugehörigen Körper K. Das heißt:

$$\begin{aligned}(v,w)\colon \mathcal{B} &\to K\\ f &\mapsto f(v,w) \text{ für } f \in \mathcal{B}\end{aligned}$$

Die Bilinearformen bilden einen Vektorraum und damit bilden auch die durch Tupel (v,w) definierten Funktionen einen Vektorraum. Letzteren nennen wir hier T. Nun sind aber zwei Elemente

[1] für eine Heranführung an Tensoren in ähnlicher, aber etwas ausführlicherer Form siehe [Tens]

$(v_1, w_1), (v_2, w_2)$ dieses Vektorraumes genau dann gleich, wenn ihre Funktionswerte für alle $f \in \mathcal{B}$ gleich sind, also wenn gilt:

$$\begin{aligned} (v_1, w_1)(f) &= (v_2, w_2)(f) \quad \forall f \in \mathcal{B}, \\ \text{also} \quad f(v_1, w_1) &= f(v_2, w_2) \quad \forall f \in \mathcal{B} \end{aligned}$$

Zwei Elemente sind also gleich, wenn sie sich unter allen bilinearen Abbildungen gleich verhalten. Damit gilt, dass Vektoren aus diesem Vektorraum genau dann linear abhängig sind, wenn sich einer der Vektoren unter allen bilinearen Abbildungen genauso verhält wie eine Linearkombination der anderen Vektoren. Das heißt aber, dass eine Basis dieses Vektorraumes eine Menge von Vektoren enthält, die sozusagen unter der Bilinearität unabhängig sind. Andererseits lassen sich alle Vektoren, die unter den bilinearen Abbildungen verschieden sind (also Eingaben für eine Bilinearform aus $\mathcal{B}$) als Linearkombinationen daraus bilden. Damit enthält eine solche Basis genau eine Menge von Tupeln, für die wir Funktionswerte definieren müssen, um Bilinearformen eindeutig zu beschreiben. Wenn wir nun den Vektorraum T zusammen mit der bilinearen Abbildung

$$\begin{aligned} \otimes : V \times W &\to T \\ (v, w) &\mapsto (v, w) \text{ als Funktion auf } \mathcal{B} \end{aligned}$$

betrachten, so lässt sich zeigen, dass er folgende Eigenschaft hat:
Für jede bilineare Abbildung $\varphi : V \times W \to X$, wobei X ein beliebiger Vektorraum ist, gibt es eine eindeutig bestimmte lineare Abbildung $\psi : T \to X$ mit $\varphi(v, w) = \psi(v \otimes w)\ \forall v \in V, w \in W$. Es lässt sich weiterhin zeigen, dass obige Eigenschaft - auch als universelle Eigenschaft bezeichnet - den Vektorraum bereits bis auf Isomorphie eindeutig bestimmt, es gibt also bis auf Isomorphie nur einen Vektorraum, der diese universelle Eigenschaft hat. Entsprechend wird das Tensorprodukt nur anhand dieser Eigenschaft definiert.

Definition 9. *Das Tensorprodukt $V \otimes W$ ist ein Vektorraum mit einer bilinearen Abbildung $\otimes : V \times W \to V \otimes W$ mit folgender Eigenschaft:*
Für jede bilineare Abbildung $\varphi : V \times W \to X$, wobei X ein beliebiger Vektorraum ist, gibt es eine eindeutig bestimmte lineare Abbildung $\psi : V \otimes W \to X$ mit $\varphi(v, w) = \psi(v \otimes w)\ \forall v \in V, w \in W$.

Bemerkung 1. *Das Tensorprodukt existiert für beliebige Vektorräume V, W. Es ist außerdem (wie bereits erwähnt) bis auf Isomorphie eindeutig bestimmt.*

Eine andere Darstellung, die in der Mathematik sehr üblich ist und uns außerdem erleichtert eine Basis des Tensorproduktes anzugeben, ist die folgende:
Seien $e_{v,w}$ für beliebige $v \in V, w \in W$ neue Elemente. Sei weiterhin X der von der Menge $\{e_{v,w} \mid v \in V, w \in W\}$ aufgespannte Vektorraum. Die $e_{v,w}$ sind dabei - da nicht anders definiert - zunächst linear unabhängig. X enthält also für jedes Tupel (v, w) mit $v \in V, w \in W$ einen Basisvektor $e_{v,w}$. Damit würde es reichen auf diesen Basisvektoren Funktionswerte zu definieren, um eine Bilinearform eindeutig zu beschreiben. Allerdings handelt es sich immer noch um eine vollständige Beschreibung der Funktionswerte aller Elemente. Deshalb müssen wir dafür sorgen, dass Elemente gleich sind, sofern sie sich unter allen bilinearen Abbildungen gleich verhalten (siehe obige Herleitung von T). Dazu definieren wir den Untervektorraum, der von folgender Menge aufgespannt wird:

$$\left\{ \begin{array}{l} e_{v_1+v_2,w} - e_{v_1,w} - e_{v_2,w}, \quad e_{a \cdot v,w} - a \cdot e_{v,w}, \\ e_{v,w_1+w_2} - e_{v,w_1} - e_{v,w_2}, \quad e_{v,a \cdot w} - a \cdot e_{v,w} \end{array} \middle| \; v, v_1, v_2 \in V, w, w_1, w_2 \in W, a \in K \right\}$$

Diesen nennen wir E.
Bilden wir nun den Quotientenraum X/E, also den Vektorraum der Äquivalenzklassen von Elementen aus X, in dem jeweils Elemente von X genau dann in einer Äquivalenzklasse liegen, wenn

sie sich nur um Linearkombinationen der Elemente aus E voneinander unterscheiden, so haben wir das Ziel erreicht. So sind beispielsweise durch den ersten Basisvektor von E die Elemente $e_{v_1+v_2,w}$ und $e_{v_1,w}+e_{v_2,w}$ im Quotientenraum X/E gleich und diese Gleichheiten setzen sich linear fort.
Auch dieser Vektorraum erfüllt zusammen mit der bilinearen Abbildung $\otimes(v,w) := e_{v,w}$ die universelle Eigenschaft, ist also ein Tensorprodukt und isomorph zu T.

2.5 Eigenschaften des Tensorproduktes und damit verbundene Begriffe

Ist $dim(V) = n$, $dim(W) = m$ und sind Basen von V und W gegeben durch $\{e_i \mid i \in \{1,\dots,n\}\}, \{f_i \mid i \in \{1,\dots,m\}\}$, so lässt sich eine Basis des auf die zweite Art konstruierten Tensorproduktes $V \otimes W$ angeben durch $\{e_i \otimes f_j \mid i \in \{1,\dots,n\}, j \in \{1,\dots,m\}\}$. Damit ergibt sich gleichzeitig auch $dim(V \otimes W) = dim(V) \cdot dim(W)$.
Die obige Menge ist eine Basis, denn: Zum einen erzeugen die Elemente das Tensorprodukt, da aufgrund der Bilinearität von $\otimes$ aus Linearkombinationen der $e_i \otimes f_j$ beliebige Elemente $v \otimes w$ für $v \in V, w \in W$ gebildet werden können. Zum anderen sind die Basisvektoren von V und W jeweils linear unabhängig und zwei Elemente in dem auf die beschriebene Weise konstruierten Tensorprodukt sind nur dann gleich, wenn sich dies aus der Bilinearität ergibt. Deshalb ist die beschriebene Menge linear unabhängig. Tensoren können somit als mehrdimensionales Array angegeben werden, wobei die Anzahl der Vektorräume, deren Tensorprodukt gebildet wird, angibt, wieviele Indizes verwendet werden. Die Abbildung $\otimes : V \times W \to V \otimes W$ ist dann gegeben durch $(v_i) \otimes (w_j) \mapsto (t_{ij})$ mit $t_{ij} = v_i \cdot w_j$. Hier lässt sich leicht nachrechnen, dass dies der Fortsetzung von $\otimes$ für e_i, f_j auf beliebige Tupel entspricht.
Wir wollen nun das tensorielle Produkt zweier dreidimensionaler Tensoren, $t = t' \otimes t''$, betrachten. Da das Tensorprodukt ein Vektorraum ist, lässt sich dies mit Hilfe obiger Beobachtungen beschreiben. Obwohl wir die Tensoren nun als Vektoren betrachten, schreiben wir der Übersichtlichkeit halber weiterhin $(t'_{i_1,j_1,k_1}), (t''_{i_2,j_2,k_2})$. Damit wäre nach obigen Beobachtungen

$$t_{(i_1,j_1,k_1),(i_2,j_2,k_2)} = t'_{i_1,j_1,k_1} \cdot t''_{i_2,j_2,k_2}$$

Durch geeignete Umsortierung der Indizes (also durch die Wahl einer geeigneten Basis) erhalten wir eine Darstellung von t mit Multi-Indizes ijk mit $i = (i_1,i_2), j = (j_1,j_2), k = (k_1,k_2)$ und erhalten

$$t_{ijk} = t'_{i_1,j_1,k_1} \cdot t''_{i_2,j_2,k_2}. \tag{2.1}$$

Durch geeignete Wahl der Basis erhalten wir als tensorielles Produkt zweier Tensoren also wieder einen Tensor der gleichen Ordnung (mit gleicher Anzahl von Indizes), wobei die Größe sich folgendermaßen ändert: Waren t' und t'' Tensoren der Größen $M' \times N' \times K'$ und $M'' \times N'' \times K''$, so ist t ein Tensor der Größe $M'M'' \times N'N'' \times K'K''$, da jeweils beide Indizes alle möglichen Werte durchlaufen.
Für mehr als zwei Tensoren lässt sich analog argumentieren.

Definition 10. *Ein Tensor $t \in V \otimes W$ heißt zerlegbar, wenn er ein Element $v \otimes w$ für $v \in V, w \in W$ ist, also direkt als tensorielles Produkt zweier Vektoren entsteht. Nicht alle Tensoren sind zerlegbar. Die zerlegbaren Tensoren bilden ein Erzeugendensystem des Tensorproduktes, aber keine Basis.*

Lemma 1. *Seien $t_1,\dots,t_k$ zerlegbare Tensoren. Dann ist auch $t = t_1 \otimes \dots \otimes t_k$ zerlegbar.*

Beweis. Wir zeigen die Aussage für $k = 2$ und 3-dimensionale Tensoren, für die übrigen Fälle lässt sich aber analog argumentieren[2]. Sei $t' := t_1, t'' := t_2$. Nach Voraussetzung gibt es Zerlegungen

[2]Eine analoge Argumentation ist auf jeden Fall möglich, allerdings lässt sich die Aussage für $k > 2$ hier nicht direkt induktiv folgern, da ich nicht explizit auf Tensorprodukte von mehr als zwei Vektorräumen eingegangen bin - insbesondere die Assoziativität des Tensorprodukts wäre hier nötig

$t' = u_1 \otimes v_1 \otimes w_1, t'' = u_2 \otimes v_2 \otimes w_2$, wobei die Vektoren x_l für $x \in \{u,v,w\}, l \in \{1,2\}$ gegeben sind durch $x_l = (x_l^j)_j$. Es gilt also

$$t'_{i_1,j_1,k_1} = u_1^{i_1} v_1^{j_1} w_1^{k_1}, \quad t''_{i_2,j_2,k_2} = u_2^{i_2} v_2^{j_2} w_2^{k_2}$$

Damit gilt nach obigen Beobachtungen

$$t_{(i_1,i_2),(j_1,j_2),(k_1,k_2)} = t'_{i_1,j_1,k_1} \cdot t''_{i_2,j_2,k_2} = u_1^{i_1} v_1^{j_1} w_1^{k_1} \cdot u_2^{i_2} v_2^{j_2} w_2^{k_2} = u^{(i_1,i_2)} v^{(j_1,j_2)} w^{(k_1,k_2)}$$

für Vektoren u,v,w deren Einträge die Produkte der Einträge von u_1 und u_2, v_1 und v_2 bzw. w_1 und w_2 (passend sortiert) sind. Damit ist t zerlegbar. □

Definition 11. *Der Rang eines Tensors $t \in V \otimes W$, geschrieben $r(t)$, ist die kleinste natürliche Zahl r, sodass für $v_i \in V, w_i \in W, i \in \{1,\dots,r\}$ gilt:*

$$\sum_{i=1}^{r} v_i \otimes w_i = t,$$

also die minimale Anzahl zerlegbarer Tensoren, deren Summe den Tensor ergibt.
Für eine einzelne Zerlegung (die nicht optimal sein muss) nennen wir das r in obiger Darstellung die Länge dieser Zerlegung.

Damit erhalten wir für den Rang des tensoriellen Produktes zweier Tensoren folgendes

Lemma 2. *Der tensorielle Rang verhält sich submultiplikativ. Es gilt also für zwei Tensoren t, t':*

$$r(t \otimes t') \leq r(t) \cdot r(t')$$

Beweis. Nach Definition des Ranges existieren Zerlegungen von t und t', sodass

$$t = \sum_{l=1}^{r(t)} u_l \otimes v_l \otimes w_l, \quad t' = \sum_{l=1}^{r(t')} u'_l \otimes v'_l \otimes w'_l$$

Die hochgestellten Indizes geben dabei die Komponenten der Vektoren $u_l, v_l, w_l, u'_l, v'_l, w'_l$ an.
Damit gilt aufgrund der Distributivgesetze im zugrunde liegenden Körper:

$$t \otimes t' = \left(\sum_{i=1}^{r(t)} u_i \otimes v_i \otimes w_i \right) \otimes \left(\sum_{j=1}^{r(t')} u'_j \otimes v'_j \otimes w'_j \right) = \sum_{i=1}^{r(t)} \sum_{j=1}^{r(t')} (u_i \otimes v_i \otimes w_i) \otimes (u'_j \otimes v'_j \otimes w'_j)$$

Die Summanden sind tensorielle Produkte zerlegbarer Tensoren, also nach (Lemma 1) selbst zerlegbar, die Anzahl der Summanden ist $r(t) \cdot r(t')$. Damit ist $r(t \otimes t') \leq r(t) \cdot r(t')$. □

2.6 Tensoren zur Beschreibung der Matrizen-Multiplikation

Hierfür sollten wir uns zunächst betrachten, was die vorherigen Überlegungen für die Darstellung einer einzelnen Bilinearform bedeuten. Wir haben uns überlegt, dass Funktionswerte für alle Basisvektoren des Tensorproduktes festgelegt werden müssen. Dabei verwenden wir die oben besprochene, übliche Basis. Diese besteht aus Basisvektoren $e_i \otimes f_j$ für jede Kombination aus je einem Basisvektor e_i von V und einem Basisvektor f_j von W. So kann eine solche Bilinearform durch eine Matrix angegeben werden. Sind $\{e_1,\dots,e_n\}$ und $\{f_1,\dots,f_m\}$ Basen von V bzw. W, ist φ

die betrachtete Bilinearform und $A = (a_{ij})$ die Darstellung als Matrix, so steht in dieser Matrix an Position ij jeweils der Funktionswert $\varphi(e_i, f_j)$ (Diese Darstellung ist zunächst nicht eindeutig, aber aufgrund der folgenden Eigenschaft eine natürliche Darstellung). Daraus ergibt sich mit $x = (x_i) \in V, y = (y_j) \in W$ und $dim(V) = n, dim(W) = m$:

$$\begin{aligned}
x^T Ay &= x_1 a_{11} y_1 + \cdots + x_n a_{n1} y_1 + \cdots + x_1 a_{1m} y_m + \cdots + x_n a_{nm} y_m \\
&= \sum_{j=1}^{m} (\sum_{i=1}^{n} a_{ij} x_i) y_j \\
&= \sum_{i,j} x_i a_{ij} y_j \\
(\text{Definition } a_{ij}) &= \sum_{i,j} x_i \varphi(e_i, f_j) y_j \\
(\text{Bilinearität } \varphi) &= \sum_{i,j} \varphi(x_i, y_j) \\
(\text{Bilinearität } \varphi) &= \varphi(x, y)
\end{aligned}$$

Eine Menge von Bilinearformen lässt sich entsprechend als eine Menge solcher Matrizen, also Tensoren von Ordnung zwei, beschreiben. Diese lassen sich alternativ als ein einziger Tensor von Ordnung drei auffassen, wobei die entsprechende Bilinearform durch den dritten Index ausgewählt wird. Wir erhalten also ein 3-dimensionales Array als Darstellung einer Menge von Bilinearformen.
Das Problem (m, n, p) ist die Berechnung von $m \cdot p$ Produkten der Form $c_{ik} = \sum_j a_{ij} b_{jk}$. Wir verwenden für die Darstellung als Tensor zu Gunsten der Übersichtlichkeit Multi-Indizes $(i_2 j_1), (j_2 k_1), (k_2, i_1)$. Wir wollen diese MM mit Hilfe des Tensors $t = (t_{(i_2 j_1)(j_2 k_1)(k_2 i_1)})$ beschreiben, sodass gilt:

$$c_{i_1 k_2} = \sum_{i_2, j_1, j_2, k_1} t_{(i_2, j_1), (j_2, k_1), (k_2, i_1)} a_{i_2, j_1} b_{j_2, k_1}$$

Dabei passen die Multi-Indizes also jeweils zu den beiden Indizes der Matrizen, sodass eine Umrechnung auf einen einzelnen Index entfällt. t enthält dann nur Nullen und Einsen und lässt sich mit dem Kronecker-Symbol δ_{ij} beschreiben, das gegeben ist durch

$$\delta_{ij} = \begin{cases} 1 & , \text{ falls } i = j \\ 0 & , \text{ sonst} \end{cases}$$

Dabei stehen in $t_{(i_1, j_1), (j_2, k_1), (k_2, i_2)}$ genau an denjenigen Stellen Einsen, an denen $j_1 = j_2, k_1 = k_2, i_1 = i_2$ gilt, da damit genau die richtige Summe für den richtigen Eintrag c_{i_1, k_2} ausgewählt wird. Dementsprechend gilt

$$t_{(i_2, j_1), (j_2, k_1), (k_2, i_1)} = \delta_{i_1, i_2} \delta_{j_1, j_2} \delta_{k_1, k_2}.$$

Nun sieht man aber leicht, dass der Tensor t nicht nur mit der Multiplikation zweier Matrizen A, B assoziiert ist, sondern auch mit der Berechnung von $Tr(ABC)$, also der Spur des Matrixproduktes dreier Matrizen A, B, C, denn es gilt (siehe (Abschnitt 2.3)):

$$Tr(ABC) = \sum_{i,j,k} a_{ij} b_{jk} c_{ki}$$

und mit obiger Darstellung von t:

$$Tr(ABC) = \sum_{i_1, i_2, j_1, j_2, k_1, k_2} t_{(i_2, j_1), (j_2, k_1), (k_2, i_1)} a_{i_2, j_1} b_{j_2, k_1} c_{k_2, i_1}.$$

Definition 12. *Wir bezeichnen den mit dem Problem (m,n,p) assoziierten Tensor - in diesem Abschnitt t - mit $\underline{\langle m,n,p\rangle}$.*

Wenn wir nun also zeigen, dass der Rang einer Trilinearform gleich dem Rang von $\langle m,n,p\rangle$ ist, so ist damit sofort gezeigt, dass alle drei bisherigen Rang-Begriffe für die MM äquivalent sind - also der Rang einer Menge von Bilinearformen, der Rang der korrespondierenden Trilinearform und der Rang des zu diesen beiden assoziierten Tensors.

Satz 2. *Der Rang des Problems (m,n,p), der Rang der korrespondierenden Trilinearform und der Rang des damit assoziierten Tensors $\langle m,n,p\rangle$ sind gleich.*

Beweis. Wir haben bereits gezeigt, dass der Rang der Menge von Bilineaformen, mit der die Berechnung der c_{ik} beschrieben wird, und der Rang der einzelnen Trilinearform, mit der die Berechnung von $Tr(ABC)$ beschrieben wird, gleich sind. Es genügt also zu zeigen, dass letzterer Rang gleich dem Rang des assoziierten Tensors, also $\langle m,n,p\rangle$, ist.
Dazu zeigen wir, dass die Gleichheit für zerlegbare Tensoren und trilineare Algorithmen von Rang 1 gilt, denn daraus folgt: Haben wir eine tensorielle Zerlegung des Tensors, so können wir für jeden zerlegbaren Tensor dieser Zerlegung einen trilinearen Algorithmus von Länge 1 konstruieren und erhalten durch Summieren einen trilinearen Algorithmus, dessen Länge gleich der Länge der tensoriellen Zerlegung ist. Damit ist der Rang der Trilinearform kleiner oder gleich dem Rang des Tensors. Umgekehrt können wir bei gegebenem trilinearen Algorithmus für jedes Produkt der Form $L_q^1 L_q^2 L_q^3$ einen zerlegbaren Tensor angeben und damit den Tensor der Trilinearform als Summe dieser zerlegbaren Tensoren angeben, erhalten also eine tensorielle Zerlegung, deren Länge gleich der Länge des Algorithmus ist. Somit ist der Rang des Tensors kleinergleich dem Rang der Trilinearform und damit sind beide Ränge gleich. Angemerkt sei, dass es sich bei den zerlegbaren Tensoren dann nicht mehr um Tensoren handelt, die mit der MM assoziiert sind. Genauso beschreiben die einzelnen Summanden des trilinearen Algorithmus keine vollständige MM.
Es bleibt zu zeigen: Der Tensor t ist zerlegbar $\Leftrightarrow$ Die mit t assoziierte Trilinearform hat Rang 1.
Wir zeigen zwar beide Richtungen auf einmal, allerdings gehen wir für eine übersichtlichere Notation von der Seite des Tensors t aus. Seien $u=(u_i), v=(v_j), w=(w_k)$ Vektoren, sodass $t=u\otimes v\otimes w$. Seien $L_1=\sum_i \alpha_i a_i, L_2=\sum_j \beta_j b_j, L_3=\sum_k \gamma_k c_k$ Linearkombinationen. Seien $\alpha=(\alpha_i), \beta=(\beta_j), \gamma=(\gamma_k)$ die Koeffizienten als Vektoren aufgefasst.
Wir setzen nun $\alpha := u, \beta := v, \gamma := w$. Damit folgt die Aussage sofort aus folgenden Gleichheiten:

$$\begin{aligned} t_{ijk} &= u\otimes v\otimes w \\ \sum_{i,j,k} t_{ijk} a_i b_j c_k &= \sum_{i,j,k} (u\otimes v\otimes w)_{ijk} a_i b_j c_k \\ &= \sum_{i,j,k} (u_i v_j w_k) a_i b_j c_k \\ &= \sum_{i,j,k} (\alpha_i \beta_j \gamma_k) a_i b_j c_k \\ &= \sum_{i,j,k} (\alpha_i a_i)(\beta_j b_j)(\gamma_k c_k) \\ &= (\sum_i \alpha_i a_i)(\sum_j \beta_j b_j)(\sum_k \gamma_k c_k) \end{aligned}$$

□

Nun zeigen wir noch einen sehr wichtigen aber grundlegenden Satz, nämlich die Dualität des Problems (m,n,p), dass wir also bei beliebiger Permutation der m,n,p immer Probleme gleichen Ranges erhalten.

Lemma 3. *Die Probleme (m,n,p) und $(\sigma(m),\sigma(n),\sigma(p))$ haben denselben Rang für eine beliebige Permutation σ von m,n und p. Dies ist nach dem vorherigen Satz äquivalent dazu, dass die Tensoren $\langle m,n,p\rangle$ und $\langle\sigma(m),\sigma(n),\sigma(p)\rangle$ denselben Rang haben.*

Beweis. Für die Tensoren lässt sich dieser Satz sehr leicht zeigen. So lässt sich, wie wir oben gesehen haben, jeder Tensor $t := \langle m,n,p\rangle$ schreiben als

$$t = (t_{(i_1,i_2),(j_1,j_2),(k_1,k_2)}).$$

Werden die m,n,p nun permutiert, so tauschen nur die Multi-Indizes von t ihre Rollen. Damit reicht es aus, entsprechend der Permutation auch die Vektoren u_l, v_l, w_l der Zerlegung von t zu permutieren, sodass sich aus einer Zerlegung des Tensors $\langle m,n,p\rangle$ von Rang r immer eine Zerlegung von $\langle\sigma(m),\sigma(n),\sigma(p)\rangle$ gleichen Ranges ergibt. □

3 Obere Schranken für ω

In diesem Abschnitt folgt der Hauptteil dieser Arbeit. Seit 1968 sind viele Resultate erschienen, die obere Schranken für ω ergaben. Die Hauptresultate lieferten dabei vor allem jeweils neue Möglichkeiten, den Suchraum für Algorithmen zu vergrößern, also die Voraussetzungen zu reduzieren, die ein Algorithmus erfüllen muss, um daraus eine Schranke für ω abzuleiten. Es werden hier chronologisch die verschiedenen Resultate präsentiert, wobei Beweise teilweise aufgrund des Umfanges weniger detailliert gegeben werden. Detaillierte Beweise können in den zugehörigen, referenzierten Publikationen gefunden werden.

3.1 Strassens Algorithmus (1968)

Volker Strassen lieferte 1968 in [Str68] als erster eine verbesserte obere Schranke für ω und stieß damit eine ganze Welle weiterer Ansätze und daraus resultierender oberer Schranken an.
Die zugrunde liegende Idee ist einerseits noch sehr leicht verständlich, andererseits ermöglichte sie erst die weiteren Überlegungen und ist gewissermaßen Grundlage zumindest fast jedes weiteren Beweises für eine obere Schranke von ω. Strassen führte die Rekursion in die Algorithmen für die MM ein. Damit war es nicht mehr nötig einen allgemeinen Algorithmus für Matrizen beliebiger Größe anzugeben. Stattdessen war nur ein effizienter Grundalgorithmus - mit gewissen Einschränkungen - für die MM einer beliebigen festen Größe nötig, aus dem sich dann ein Algorithmus für MMen beliebiger Größen rekursiv ableiten lässt. Strassen formuliert dazu einen Algorithmus für das Problem $(2,2,2)$. Dazu seien $A=(a_{ij}), B=(b_{ij}), C=AB=(c_{ij}), i,j \in \{1,2\}$. Es werden dann zunächst die folgenden 7 Terme festgelegt:

$$\begin{aligned}
\mathrm{I} &= (a_{11}+a_{22})(b_{11}+b_{22}) \\
\mathrm{II} &= (a_{21}+a_{22})b_{11} \\
\mathrm{III} &= a_{11}(b_{12}-b_{22}) \\
\mathrm{IV} &= a_{22}(-b_{11}+b_{21}) \\
\mathrm{V} &= (a_{11}+a_{12})b_{22} \\
\mathrm{VI} &= (-a_{11}+a_{21})(b_{11}+b_{12}) \\
\mathrm{VII} &= (a_{12}-a_{22})(b_{21}+b_{22})
\end{aligned}$$

Damit ergibt sich ein Algorithmus für C durch:

$$\begin{aligned}
c_{11} &= \mathrm{I}+\mathrm{IV}-\mathrm{V}+\mathrm{VII} \\
c_{21} &= \mathrm{II}+\mathrm{IV} \\
c_{12} &= \mathrm{III}+\mathrm{V} \\
c_{22} &= \mathrm{I}+\mathrm{III}-\mathrm{II}+\mathrm{VI}
\end{aligned}$$

Auch wenn es zum Beweis der daraus resultierenden Schranke nicht nötig sein wird, sollen hier noch die Darstellung als bilinearer Algorithmus sowie die korrespondierende Zerlegung des Tensors $\langle 2,2,2\rangle$ angegeben werden. Diese dienen gleichzeitig als praktische Beispiele der verschiedenen

Darstellungen. Die Linearkombinationen, die sich für die Darstellung als bilinearer Algorithmus ergeben, sind:

$$\begin{aligned}
L_1^1 &= a_{11} + a_{22} \quad , \quad L_1^2 = b_{11} + b_{22} \\
L_2^1 &= a_{21} + a_{22} \quad , \quad L_2^2 = b_{11} \\
L_3^1 &= a_{11} \quad , \quad L_3^2 = b_{12} - b_{22} \\
L_4^1 &= a_{22} \quad , \quad L_4^2 = -b_{11} + b_{21} \\
L_5^1 &= a_{11} + a_{12} \quad , \quad L_5^2 = b_{22} \\
L_6^1 &= -a_{11} + a_{21} \quad , \quad L_6^2 = b_{11} + b_{12} \\
L_7^1 &= a_{12} - a_{22} \quad , \quad L_7^2 = b_{21} + b_{22}
\end{aligned}$$

Damit ergeben sich die $\gamma_{ik,q}$ aus folgenden Gleichungen:

$$\begin{aligned}
c_{11} &= L_1^1 L_1^2 + L_4^1 L_4^2 - L_5^1 L_5^2 + L_7^1 L_7^2 \\
c_{21} &= L_2^1 L_2^2 + L_4^1 L_4^2 \\
c_{12} &= L_3^1 L_3^2 + L_5^1 L_5^2 \\
c_{22} &= L_1^1 L_1^2 + L_3^1 L_3^2 - L_2^1 L_2^2 + L_6^1 L_6^2
\end{aligned}$$

In diesem Fall sind also alle Koeffizienten $\alpha_{ij,q}, \beta_{jk,q}, \gamma_{ik,q} \in \{0, 1, -1\}$. Nun folgt noch die korrespondierende tensorielle Zerlegung. Dabei nummerieren wir mit den hochgestellten Indizes die Vektoren durch, die tiefgestellten geben die jeweilige Komponente des Vektors an. Wir wollen hier keine Multi-Indizes verwenden, da diese verwirrend sein können. Stattdessen identifizieren wir die Multi-Indizes folgendermaßen mit einzelnen Indizes:

$$(11) \leftrightarrow 1, (12) \leftrightarrow 2$$

$$(21) \leftrightarrow 3, (22) \leftrightarrow 4$$

Wir erhalten damit (beachte den Tausch der beiden Indizes der γ_{ik} beim Übergang zur Trilinearform und damit auch zur tensoriellen Zerlegung) folgende tensorielle Zerlegung[1]:

$$\begin{aligned}
u_1 &= (1,0,0,1),\ v_1 = (1,0,0,1),\ w_1 = (1,0,0,1) \\
u_2 &= (0,1,0,1),\ v_2 = (1,0,0,0),\ w_2 = (0,1,0,-1) \\
u_3 &= (1,0,0,0),\ v_3 = (0,0,1,-1),\ w_3 = (0,0,1,1) \\
u_4 &= (0,0,0,1),\ v_4 = (-1,1,0,0),\ w_4 = (1,1,0,0) \\
u_5 &= (1,0,1,0),\ v_5 = (0,0,0,1),\ w_5 = (-1,0,1,0) \\
u_6 &= (-1,1,0,0),\ v_6 = (1,0,1,0),\ w_6 = (0,0,0,1) \\
u_7 &= (0,0,1,-1),\ v_7 = (0,1,0,1),\ w_7 = (1,0,0,0),
\end{aligned}$$

sodass:

$$t_{ijk} = \sum_{l=1}^{7} u_l^i v_l^j w_l^k$$

Lemma 4. *Der oben beschriebene Algorithmus ist ein korrekter Algorithmus für das Problem* $(2,2,2)$.

[1] Der Einfachheit halber sind die Vektoren hier als Zeilenvektoren angegeben

Beweis. Der Beweis für die Korrektheit wird vollständig durch Nachrechnen erbracht. □

Wir haben damit einen Algorithmus für das Problem $(2,2,2)$, der 7 Multiplikationen benötigt. Im Gegensatz dazu benötigt der triviale Algorithmus $2^3 = 8$ Multiplikationen. Strassens Idee war nun, diesen Algorithmus in der Art rekursiv anzuwenden, dass wir eine beliebige $2^n \times 2^n$-MM als 2×2-MM betrachten, wobei die Einträge Matrizen der Größe 2^{n-1} sind. Es lassen sich weiterhin Matrizen beliebiger Größe in $2^n \times 2^n$-Matrizen für die nächsthöhere Zweierpotenz 2^n einbetten, sodass wir einen Algorithmus für Matrizen beliebiger Größe erhalten. Dabei ist zu beachten, dass für diese Rekursion ein nicht-kommutativer Algorithmus benötigt wird - denn offensichtlich wird mit Matrizen als Einträgen gerechnet und für diese gilt bekanntlich keine Kommutativität. Allerdings verwendet Strassens Algorithmus, wie auch beliebige andere bilineare bzw. trilineare Algorithmen, keine Kommutativität, diese lassen sich also auf die beschriebene Art rekursiv anwenden. Aus diesem Grund bietet es sich auch an, die Anzahl der arithmetischen Operationen, die ein auf diese Art rekursiv angewendeter Algorithmus benötigt (also die daraus resultierende obere Schranke für ω), allgemein zu beweisen und den Exponenten von Strassen als Korollar zu gewinnen.

Satz 3. *Sei ein nicht-kommutativer Algorithmus, der nur Additionen, Subtraktionen und Multiplikationen verwendet, - beispielsweise ein bilinearer oder trilinearer Algorithmus der MM - gegeben, der das Produkt zweier $k \times k$-Matrizen mit M Multiplikationen berechnet.*
Dann gilt:

- *(i) Bei rekursiver Anwendung ist die Zahl der skalaren Multiplikationen[2] für die Multiplikation zweier $k^h \times k^h$-Matrizen M^h.*
- *(ii) Bei rekursiver Anwendung ist die Zahl der skalaren Additionen[3] für die Multiplikation zweier $k^h \times k^h$-Matrizen in $O((k^h)^2)$.*
- *(iii) $\omega \leq \log_k(M)$, sofern $M \geq k^2$ ([4]).*

Beweis. (i): Beweis durch Induktion.
IV: Aussage von (i).
IA $h = 0$: Es soll das Produkt zweier $k^0 \times k^0$-Matrizen, also 1×1-Matrizen, also zweier Skalare berechnet werden. Diese Berechnung benötigt offensichtlich eine skalare Multiplikation.
IS $h \to h+1$: Durch rekursive Anwendung des Algorithmus werden für die Multiplikation zweier $k^{h+1} \times k^{h+1}$-Matrizen M Multiplikationen von $k^h \times k^h$-Matrizen benötigt. Diese brauchen nach IV M^h Multiplikationen, also benötigt die Multiplikation zweier $k^{h+1} \times k^{h+1}$-Matrizen $M \cdot M^h = M^{h+1}$ Multiplikationen.

(ii): Sei l die Anzahl der Additionen in dem Algorithmus für die Multplikation zweier $k \times k$-Matrizen.
Wir unterscheiden hier die Fälle $k = 1$ und $k \geq 2$.
Fall $k = 1$: Die $k^h \times k^h$-MM bleibt für beliebige h immer eine Multiplikation zweier Skalare, benötigt also für beliebige h genau $l \in O(1)$ Additionen.
Fall $k \geq 2$: Der Algorithmus berechnet rekursiv absteigend gewisse Produkte von $k^i \times k^i$-Matrizen, wobei die Einträge, die dabei addiert werden, jeweils k^{i-1}-Matrizen sind. Damit ist die Anzahl der

[2] Multiplikation zweier Skalare
[3] Addition zweier Skalare
[4] Auf die Einschränkung kann meines Wissens verzichtet werden, sofern es sich um einen bilinearen oder trilinearen Algorithmus handelt

Additionen insgesamt:

$$\begin{aligned}\sum_{i=1}^{h} l \cdot (k^{h-i})^2 &= l \cdot \sum_{i=0}^{h-1} (k^i)^2 \\ &= \sum_{i=0}^{h-1} (k^2)^i \\ &< l \cdot (k^2)^h \\ &= l \cdot (k^h)^2 \\ &\in O((k^h)^2)\end{aligned}$$

Dabei sieht man leicht, das die Ungleichung zwischen der zweiten und dritten Zeile gilt, wenn man die Summanden als Stellen einer Zahl zur Basis k^2 betrachtet. Offensichtlich sind dann nämlich für $k^2 \geq 2$, also $k \geq 2$, die summierten Stellenwerte der 0-ten bis $(h-1)$-ten Stelle kleiner als der Stellenwert der h-ten Stelle.

(*iii*): Da die Anzahl der Additionen in einem solchen rekursiv konstruierten Algorithmus in $O((k^h)^2)$ liegt und nach Voraussetzung der Grund-Algorithmus mindestens k^2, der rekursiv konstruierte Algorithmus für die h-te Potenz also $(k^h)^2$, Multiplikationen benötigt, ist in der O-Notation die Anzahl von Additionen nicht von Bedeutung. Damit genügt es, die Anzahl der Multiplikationen zu betrachten. Es werden also $O(M^h)$ arithmetische Operationen für die Multiplikation zweier $k^h \times k^h$-Matrizen benötigt. Für die Multiplikation zweier Matrizen von Größe $n := k^h$, deren Größen also Potenzen von k sind, benötigen wir

$$M^h = M^{\log_k(n)} = (n^{\log_n(M)})^{\log_k(n)} = n^{\log_n(M)\log_k(n)}$$

Multiplikationen. Dabei gilt nach Definition des Logarithmus:

$$k^{\log_n(M)\log_k(n)} = k^{\log_k(n)\log_n(M)} = (k^{\log_k(n)})^{\log_n(M)} = n^{\log_n(M)} = M$$

Es ist also $\log_n(M)\log_k(n) = \log_k(M)$. Für Matrizen, deren Größe eine Potenz von k ist, ist der Satz damit bewiesen.
Für Matrizen beliebiger Größe n gilt jedoch, dass sie sich auf jeden Fall in Matrizen der Größe $n' < n \cdot k$ einbetten lassen mit $n' = k^{h+1}$ für $k^h < n \leq k^{h+1}$ und somit werden dann höchstens M^{h+1} Multiplikationen für das Problem (n,n,n) benötigt. Damit erhalten wir (analog zu obigen Gleichungen):

$$M^{h+1} = M^{\log_k(n')} < M^{\log_k(nk)} = M^{1+\log_k(n)} = M \cdot n^{\log_n(M)\log_k(n)} = M \cdot n^{\log_k(M)}$$

Da M konstant ist, liegt damit die Anzahl der benötigten arithmetischen Operationen zur Multiplikation zweier $n \times n$-Matrizen in $O(n^{\log_k(M)})$ und es gilt:

$$\omega \leq \log_k(M).$$

□

Korollar 1. *Strassens Algorithmus liefert* $\omega \leq \log_2(7) \approx 2,807$.

An dieser Stelle ist noch erwähnenswert, dass Strassens Algorithmus für das Problem $(2,2,2)$ optimal ist: Es kann keinen Algorithmus mit 6 oder weniger Multiplikationen geben. Es wurde sogar gezeigt - siehe [HM73] -, dass Strassens Algorithmus bis auf lineare Transformationen eindeutig ist. Allerdings heißt das nicht, dass die rekursive Anwendung für beliebige Matrixgrößen optimal ist.

Beispiel 1. Um die Idee der rekursiven Anwendung zu veranschaulichen wird nun an einigen Termen beispielhaft gezeigt, wie mit Strassens Algorithmus das Produkt zweier 4×4-Matrizen zu berechnen ist. Seien

$$A = (a_{ij}),\ B = (b_{ij}),\ i,j \in \{1,\dots,4\}$$

Wir unterteilen dafür die Matrizen A und B in Blöcke der halben Größe dieser Matrizen, also in 2×2-Blöcke. Seien diese Blöcke gegeben durch

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix},$$

also:

$$A_{11} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}, A_{12} = \begin{pmatrix} a_{13} & a_{14} \\ a_{23} & a_{24} \end{pmatrix}, A_{21} = \begin{pmatrix} a_{31} & a_{32} \\ a_{41} & a_{42} \end{pmatrix}, A_{22} = \begin{pmatrix} a_{33} & a_{34} \\ a_{43} & a_{44} \end{pmatrix},$$

B_{ij} für $i,j \in \{1,2\}$ analog.

Damit lässt sich der Algorithmus von Strassen direkt anwenden. Die äußere Multiplikation wird berechnet durch Anwendung von Strassens Algorithmus auf die Einträge A_{ij}, B_{ij} für $i,j \in \{1,2\}$, also indem in den entsprechenden Gleichungen einfach Kleinbuchstaben durch Großbuchstaben ersetzt werden. Die Multiplikationen in diesen Gleichungen sind Multiplikationen von 2×2-Matrizen, also wird für diese wieder Strassens Algorithmus angewendet. Dies soll hier beispielhaft für Term I durchgeführt werden:

$$\begin{aligned} \mathrm{I}_{außen} &= (A_{11} + A_{22})(B_{11} + B_{22}) \\ &= \begin{pmatrix} a_{11}+a_{33} & a_{12}+a_{34} \\ a_{21}+a_{43} & a_{22}+a_{44} \end{pmatrix} \cdot \begin{pmatrix} b_{11}+b_{33} & b_{12}+b_{34} \\ b_{21}+b_{43} & b_{22}+b_{44} \end{pmatrix} \end{aligned}$$

Nun kann Strassens Algorithmus auf diese Multiplikation zweier 2×2-Matrizen angewendet werden. Damit erhalten wir zum Beispiel für den Term I dieser inneren MM:

$$\mathrm{I}_{innen} = ((a_{11}+a_{33}) + (a_{22}+a_{44}))((b_{11}+b_{33}) + (b_{22}+b_{44}))$$

Analog dazu ergeben sich die übrigen Produkte auf beiden Ebenen.

3.2 Übertragung von Strassens Idee auf Tensoren

Die rekursive Anwendung eines Algorithmus für die MM einer festen Größe lässt sich auf eine entsprechende Schachtelung von Tensoren der MM übertragen. Damit lassen sich dann aufgrund der Submultiplikativität implizit auch tensorielle Zerlegungen rekursiv schachteln und entsprechende Schranken für ω folgern. Diese Überlegung geht nicht auf dieselbe Publikation zurück wie obiger Algorithmus, sie taucht bei verschiedenen Autoren auf. Deshalb wird sie hier separat vorgestellt.
Für Tensoren der MM ergibt sich folgender

Satz 4. *Es gilt für das tensorielle Produkt zweier Tensoren (bis auf Permutation der Einträge durch Permutation der Index-Werte):*
$\langle M', N', K' \rangle \otimes \langle M'', N'', K'' \rangle = \langle M'M'', N'N'', K'K'' \rangle$.

Beweis. Sei $t' = \langle M', N', K' \rangle, t'' = \langle M'', N'', K'' \rangle$. Wir übertragen direkt die Idee der rekursiven Anwendung auf die Tensoren der MM. Dafür bilden wir das tensorielle Produkt $t' \otimes t''$. Betrachte

dazu (Gleichung (2.1)). Da wir Tensoren der MM mit Multi-Indizes angeben (siehe (Abschnitt 2.6)), also

$$t' = (t_{(i_{11},i_{12}),(j_{11},j_{12}),(k_{11},k_{12})}), t'' = (t_{(i_{21},i_{22}),(j_{21},j_{22}),(k_{21},k_{22})}),$$

erhalten wir hier

$$t = (t_{(i_{11},i_{12},i_{21},i_{22}),(j_{11},j_{12},j_{21},j_{22}),(k_{11},k_{12},k_{21},k_{22})}) = t't''$$

Wir können die Tupel von je 4 Indizes derartig auf jeweils 2 Indizes umrechnen, dass wir einen Tensor von Blöcken erhalten, wobei jeder Block das Produkt eines Eintrages von t' mit dem gesamten Tensor t'' ist. Dies entspricht dann genau der Konstruktion von Strassen.
Der eine Tensor beschreibt die MM, bei der die Einträge Blöcke sind und der andere Tensor die Multiplikation dieser Blöcke. Besonders anschaulich wird dies, wenn wir die Darstellung der Tensoren als Produkte von Kronecker-Symbolen verwenden: Es werden diejenigen Produkte 1, für die sowohl der Eintrag des inneren als auch der Eintrag des äußeren Tensors eine 1 ist. Wenn wir z.B. das Problem $(4,4,4)$ betrachten und die Matrizen in Blöcke der Größe 2×2 aufteilen, so gibt die äußere MM vor, welche Blöcke für die einzelnen Blöcke des Ergebnisses multipliziert werden müssen, die innere gibt dazu vor, wie sich die einzelnen Einträge eines Ergebnis-Blocks aus den Einträgen dieser Blöcke berechnen.
Formal lässt sich diese Konstruktion wie folgt beschreiben. Es soll gelten:

$$\langle M'M'', N'N'', K'K''\rangle = t = (t_{(i_1,i_2),(j_1,j_2),(k_1,k_2)}) = (t_{(i_{11},i_{12},i_{21},i_{22}),(j_{11},j_{12},j_{21},j_{22}),(k_{11},k_{12},k_{21},k_{22})}) = t't''$$

Dazu bilden wir für alle $x \in \{i,j,k\}$ die Vier-Tupel von Indizes wie folgt auf Zwei-Tupel von Indizes ab:

$$x_1 = (x_{11}-1)X'' + x_{12}, \quad x_2 = (x_{21}-1)X'' + x_{22}$$

mit

$$X = \begin{cases} M & \text{, falls } x = i \\ N & \text{, falls } x = j \\ K & \text{, falls } x = k \end{cases}$$

x_{11}, x_{21} wählen also die jeweiligen Blöcke aus, x_{12}, x_{22} die Einträge in diesen Blöcken. Es ist aufgrund der Analogie zur Konstruktion für bilineare Algorithmen leicht nachvollziehbar, dass der erhaltene Tensor tatsächlich $\langle M'M'', N'N'', K'K''\rangle$ ist. □

Definition 13. *Sei die Tensorpotenz eines Tensors t definiert durch* $t^{\otimes s} = \underbrace{t \otimes \cdots \otimes t}_{s \text{ mal}}$

Korollar 2. *Es gilt* $\langle M,N,K\rangle^{\otimes s} = \langle M^s, N^s, K^s\rangle$.

Beweis. Beweis durch induktive Anwendung von (Satz 4), beginnend mit $M' = M'', N' = N'', K' = K''$ und anschließend durch Multiplikation mit jeweils einem weiteren Tensor $\langle M', N', K'\rangle$. □

Korollar 3 (Symmetrisierung)**.** *Aus einem gegebenen Tensor $\langle M,N,K\rangle$ lässt sich ein Tensor, der mit dem Problem (MNK, MNK, MNK), also einer symmetrischen[5] MM, assoziiert ist, konstruieren als*

$$\langle MNK, MNK, MNK\rangle = \langle M,N,K\rangle \otimes \langle K,M,N\rangle \otimes \langle N,K,M\rangle$$

Damit ergibt sich (aufgrund der Submultiplikativität) für die Ränge:

$$r(\langle MNK, MNK, MNK\rangle) \leq r(\langle M,N,K\rangle)^3$$

[5] alle Matrixgrößen sind identisch

3.3 Victor Pan: Trilinear Aggregating, Uniting und Canceling (1978)

Nach dem Durchbruch durch Strassens Algorithmus blieb die daraus resultierende Schranke für ω recht lange ungeschlagen. Erst 1978 gelang Pan in [Pan78] eine erste Verbesserung[6]. Er verwendet dazu die trilineare Darstellung und sucht nach einem effizienten Grundalgorithmus wie dem von Strassen. Allerdings konstruiert er einen Algorithmus für MMs zweier Matrizen der Größen $n \times n$ für beliebige feste n und sucht dann dasjenige n, für das der Algorithmus am effizientesten ist. Anschließend wendet er auf diesen Algorithmus als Grundalgorithmus (Satz 3) an.
Bei der Konstruktion eines allgemeinen Algorithmus verwendet Pan vornehmlich drei wichtige Techniken, die in den folgenden drei Unterabschnitten vorgestellt werden.

3.3.1 Trilinear Aggregating

Trilinear Aggregating meint die Technik, mehrere der Summanden der Trilinearform der MM (siehe (Abschnitt 2.3)) gleichzeitig zu berechnen. Dazu werden mehrere Summanden ausgewählt, alle darin auftretenden Variablen a_{ij}, b_{jk}, c_{ki} in jeweils einer Linearkombination zusammengefasst und diese Linearkombinationen anschließend multipliziert. Beispielsweise würde aus

$$a_{ij}b_{jk}c_{ki} + a_{i+1,j+1}b_{j+1,k+1}c_{k+1,i+1}$$

durch Aggregierung

$$(a_{ij} + a_{i+1,j+1}) \cdot (b_{jk} + b_{j+1,k+1}) \cdot (c_{ki} + c_{k+1,i+1}).$$

Dadurch wird aus den 2 Produkten des obigen Terms ein einziges Produkt. Allerdings werden dadurch mehr Terme erhalten als gewünscht.

Definition 14. *Wir nennen eine derartige Zusammenfassung von (beliebig vielen) Termen Aggregat.*

Bei der Konstruktion von Algorithmen durch Aggregierung ist darauf zu achten, dass die Indizes der zusammengefassten Terme derartig gewählt werden, dass anschließend über eine Teilmenge aller Tripel von Indizes summiert werden kann und auf diese Art alle Tripel von Indizes in den Aggregaten in dieser Summe auftreten. Im obigen Beispiel wurden die Indizes als $(i,j),(j,k),(k,i),(i+1,j+1),(j+1,k+1),(k+1,i+1)$ gewählt; dies ermöglicht uns für gerade Matrixgrößen über die Menge aller Tripel von Indizes, die in Summe eine gerade Zahl ergeben, zu summieren. Die Indizes müssen dann modulo der Größe der Matrizen gelesen werden und die Tripel $(i+1,j+1,k+1)$ ergeben genau die fehlenden Tripel - diejenigen, bei denen die Summe der Einträge ungerade ist. Sei

$$S = \{(i,j,k) \mid i+j+k \text{ gerade}\}.$$

Wir erhalten also

$$\sum_{(i,j,k)\in S} (a_{ij} + a_{i+1,j+1})(b_{jk} + b_{j+1,k+1})(c_{ki}c_{k+1,i+1})$$

und in dieser Summe treten alle Terme $a_{ij}b_{jk}c_{ki}$ für beliebige Tripel (i,j,k) auf.

[6] für eine aufbereitete Version der Publikation siehe [Pan80]

3.3.2 Trilinear Uniting

Wie oben bereits erwähnt (und wie leicht zu sehen ist) erhalten wir durch Aggregierung deutlich mehr Summanden als eigentlich benötigt werden. Für obiges Beispiel erhalten wir so neben den gewünschten Termen $a_{ij}b_{jk}c_{ki}$ und $a_{i+1,j+1}b_{j+1,k+1}c_{k+1,i+1}$ auch beliebige andere Produkte aus je einer der beiden Variablen a_{ij} und $a_{i+1,j+1}$, einer der Variablen b_{jk} und $b_{j+1,k+1}$ und einer der Variablen c_{ki} und $c_{k+1,i+1}$. Um einen Algorithmus für die zur MM korrespondierende Trilinearform zu erhalten, müssen also diese Terme wieder subtrahiert werden. Allerdings hätte dies keinen Nutzen, wenn alle Terme einzeln subtrahiert würden. Stattdessen wird versucht, diese Terme über alle Summanden zusammenzufassen. Für obiges Beispiel erhalten wir als falsche Terme:

$$a_{i+1,j+1}b_{jk}c_{ki},\ a_{ij}b_{j+1,k+1}c_{ki},\ a_{ij}b_{jk}c_{k+1,i+1},$$

$$a_{i+1,j+1}b_{j+1,k+1}c_{ki},\ a_{i+1,j+1}b_{jk}c_{k+1,i+1},\ a_{ij}b_{j+1,k+1}c_{k+1,i+1}$$

Diese müssen für einen funktionierenden Algorithmus nun subtrahiert werden und zwar möglichst über alle Tripel von Indizes, deren Summe gerade ist, zusammen. Die zu subtrahierenden Terme lassen sich folgendermaßen zusammenfassen:

$$\sum_{(i,j,k)\in S}(a_{ij}+a_{i+1,j+1})b_{jk}c_{k+1,i+1},$$

$$\sum_{(i,j,k)\in S}a_{i+1,j+1}(b_{jk}+b_{j+1,k+1})c_{ki},$$

$$\sum_{(i,j,k)\in S}a_{ij}b_{j+1,k+1}(c_{ki}+c_{k+1,i+1})$$

Diese Terme sollen nun möglichst effizient berechnet werden, also mit möglichst wenigen Multiplikationen. Das Ziel dabei ist, die Summe nur über zwei der Indizes laufen zu lassen und über den dritten Index nur noch Variablen einer der Matrizen summieren zu müssen. Dadurch würde dieser eine Index nur in eine Linearkombination der Einträge einer der Matrizen einfließen und die gesamte Summe würde nur noch quadratisch viele Multiplikationen benötigen. Das Problem dabei ist, dass sich in den obigen Summen keiner der Indizes derartig isolieren lässt: jeweils zwei der Linearkombinationen von Variablen einer Matrix legen bereits alle drei Indizes fest. Allerdings sehen wir, dass jeweils die beiden einzeln stehenden (nicht geklammerten) Variablen aus den beiden verschiedenen Summanden des Aggregats stammen - also eine dieser Variablen zwei der Indizes i,j,k und die andere zwei der Indizes $i+1,j+1,k+1$ hat. Dies sollte aber auch nicht überraschen: Würden diese beiden Variablen zu dem gleichen Summanden des Aggregats gehören, (hätten also entweder beide Variablen Indizes mit „+1“ oder beide Variablen Indizes ohne „+1“), so würden wir zusammen mit einem der beiden Summanden in der Klammer einen erwünschten Term erhalten - und erwünschte Terme sollen nicht subtrahiert werden. Da die beiden Variablen aus verschiedenen Summanden des Aggregats stammen, können wir diese durch Veränderung des Aggregats in einem gewissen Rahmen unabhängig voneinander variieren. Dabei muss nur gewährleistet sein, dass sich die beiden Summanden des Aggregats nicht in der Gesamtsumme überschneiden, also dass Terme nicht doppelt berechnet werden. In unserer Konstruktion ist dies zum Beispiel durch Permutation der Multi-Indizes an den Variablen möglich, da dies nichts an der Zugehörigkeit von Tripeln zu S ändert. Wir permutieren diese also so, dass an den beiden einzeln stehenden Variablen in obigen Gleichungen jeweils die gleichen Indizes aus $\{i,j,k\}$ auftauchen (in ihrer Variante mit oder ohne „+1“). Das Aggregat ergibt sich damit zu

$$a_{ij}b_{jk}c_{ki}+a_{k+1,i+1}b_{i+1,j+1}c_{j+1,k+1}$$

und wir erhalten

$$(a_{ij}+a_{k+1,i+1})(b_{jk}+b_{i+1,j+1})(c_{ki}+c_{j+1,k+1})$$

und als zu subtrahierende Terme

$$\sum_{(i,j,k)\in S}(a_{ij}+a_{k+1,i+1})b_{jk}c_{j+1,k+1}=\sum_{j,k=0}^{n-1}(\sum_{i:(i,j,k)\in S}a_{ij}+a_{k+1,i+1})b_{jk}c_{j+1,k+1},$$

$$\sum_{(i,j,k)\in S}a_{k+1,i+1}(b_{jk}+b_{i+1,j+1})c_{ki}=\sum_{i,k=0}^{n-1}a_{k+1,i+1}(\sum_{j:(i,j,k)\in S}b_{jk}+b_{i+1,j+1})c_{ki},$$

$$\sum_{(i,j,k)\in S}a_{ij}b_{i+1,j+1}(c_{ki}+c_{j+1,k+1})=\sum_{i,j=0}^{n-1}a_{ij}b_{i+1,j+1}(\sum_{k:(i,j,k)\in S}c_{ki}+c_{j+1,k+1}).$$

n ist dabei die Größe der zu multiplizierenden Matrizen. Wir haben also einen trilinearen Algorithmus der MM, der $\frac{n^3}{2}+3n^2$ Multiplikationen verwendet. Damit können wir für verschiedene n Basisalgorithmen angeben und aus diesen mit (Satz 3) eine Schranke für ω ableiten, nämlich

$$\omega\leq\log_n\left(\frac{n^3}{2}+3n^2\right).$$

Durch Minimierung erhalten wir für $n=34$ die Schranke $\omega<2,84953$. Dieser Exponent ist zwar schlechter als derjenige, der sich aus Strassens Algorithmus ergibt, aber wir sehen bereits, dass diese Technik nicht-triviale Exponenten liefert.

3.3.3 Trilinear Canceling

Um die Technik des trilinearen Canceling zu erläutern, benötigen wir ein etwas komplexeres Beispiel einer Konstruktion, die durch Aggregierung entstanden ist. Aufgrund des großen Schreibaufwandes und der Tatsache, dass sämtliche Gleichheiten durch Nachrechnen beweisbar sind, sparen wir uns hier Beweise. Es soll vielmehr die Idee des trilinearen Canceling vermittelt werden. Wir wählen dafür einen Algorithmus von Pan, der jeweils drei Summanden folgender Form aggregiert:

$$T_{i,j,k}=a_{ij}b_{jk}c_{ki}+a_{jk}b_{ki}c_{ij}+a_{ki}b_{ij}c_{jk} \tag{3.1}$$

Wir bilden hier also das Produkt

$$(a_{ij}+a_{jk}+a_{ki})(b_{jk}+b_{ki}+b_{ij})(c_{ki}+c_{ij}+c_{jk}).$$

Zur besseren Übersichtlichkeit fassen wir hier für jedes Tripel (i,j,k) die zu subtrahierenden Terme wie folgt zusammen:

$$\begin{aligned}
T^0_{i,j,k}&=a_{ij}b_{ij}(c_{ki}+c_{ij}+c_{jk})+a_{jk}b_{jk}(c_{ij}+c_{jk}+c_{ki})+a_{ki}b_{ki}(c_{jk}+c_{ki}+c_{ij})\\
T^1_{i,j,k}&=a_{ij}(b_{jk}+b_{ki})c_{ij}+a_{jk}(b_{ki}+b_{ij})c_{jk}+a_{ki}(b_{ij}b_{jk})c_{ki}\\
T^2_{i,j,k}&=(a_{ki}+a_{ij})b_{jk}c_{jk}+(a_{ij}+a_{jk})b_{ki}c_{ki}+(a_{jk}+a_{ki})b_{ij}c_{ij}\\
T^3_{i,j,k}&=a_{ij}b_{ki}c_{jk}-a_{jk}b_{ij}c_{ki}-a_{ki}b_{jk}c_{ij}
\end{aligned}$$

Damit lässt sich das Aggregat $T_{i,j,k}$ übersichtlich schreiben als

$$T_{i,j,k}=(a_{ij}+a_{jk}+a_{ki})(b_{jk}+b_{ki}+b_{ij})(c_{ki}+c_{ij}+c_{jk})-T^0_{i,j,k}-T^1_{i,j,k}-T^2_{i,j,k}-T^3_{i,j,k}.$$

Es ist leicht zu sehen, dass sich die Summe der T^0_{ijk}, die Summe der T^1_{ijk} und die Summe der T^2_{ijk} wieder durch Uniting mit einer effizienten Anzahl von Multiplikationen berechnen lassen, denn hier tauchen wieder die Variablen aus je zwei Matrizen mit den gleichen Indizes auf und so kann der dritte Index innerhalb einer Linearkombination der Einträge der dritten Matrix durchlaufen werden. Um allerdings überhaupt einen Algorithmus zu erhalten, benötigen wir noch eine Menge von Indizes über die wir summieren können, sodass keine Terme doppelt auftreten (aber alle Terme abgedeckt sind). Die genaue Vorgehensweise ist hier nicht wichtig, im Groben geht Pan wie folgt vor: Die Menge aller Tripel (i,j,k) wird zunächst unterteilt in Tripel (i,i,i) und Tripel, für die nicht $i=j=k$ gilt. Anschließend werden letztere in drei gleich große Teilmengen zerlegt, sodass es genügt eine von ihnen zu durchlaufen, um mit den Aggregaten alle Terme außer die (i,i,i) zu erhalten. Außerdem gibt es nur n Terme der Form (i,i,i) und diese lassen sich mit den ohnehin vorhandenen Termen, die subtrahiert werden, zusammenfassen, sodass sie keine zusätzlichen Multiplikationen ergeben. Damit bleibt einzig die Summe der T^3_{ijk} übrig, die nicht effizient berechnet wird. Das Ergebnis ist ein Algorithmus von Rang $\frac{n^3-n}{3}+3n^2+X$ wobei X die Anzahl der Multiplikationen zur Berechnung der Summe der T^3_{ijk} ist. Bei einzelner Berechnung aller Summanden wären dies $X=n^3$, wodurch der Algorithmus nicht mehr effizient wäre.
Die Idee des trilinearen Canceling ist nun die folgende: Wir bilden mehrere verschiedene Versionen von (Gleichung (3.1)), wobei die Indizes zwischen diesen Versionen abgewandelt werden (zum Beispiel für manche Versionen nur Indizes $< \frac{n}{2}$ und für manche nur Indizes $\geq \frac{n}{2}$) und die Variablen geschickt mit Vorzeichen versehen werden. Dabei soll folgendes erreicht werden:

1. Die Terme lassen sich derartig summieren, dass dadurch nicht mehr Aggregate als zuvor berechnet werden müssen. Es wird also letztlich über eine noch kleinere Teilmenge der Tripel (i,j,k) summiert, aber für jedes dieser Tripel werden alle verschiedenen Versionen von (Gleichung (3.1)) summiert, sodass insgesamt wieder die gesamte Trilinearform berechnet wird.
2. Die Vorzeichen werden so verteilt, dass die Terme, die auch in der zu berechnenden Trilinearform auftreten, ihre positiven Vorzeichen behalten.
3. Die Vorzeichen werden so verteilt, dass die Terme, die nicht durch trilineares Uniting effizient berechnet werden können (bzw. für die keine effiziente Berechnung bekannt ist) - hier die Terme T^3_{ijk} - in der Summe über die verschiedenen Versionen eines Aggregats 0 ergeben und damit komplett entfallen.

Die ersten beiden Punkte sind dabei offenbar notwendig, um die Effizienz des zuvor konstruierten Algorithmus aufrecht zu erhalten, der letzte erspart uns die Berechnung der Terme T^3_{ijk}.

Bemerkung 2. *Die Zerlegung der Menge von Tripeln in geeignete Teilmengen wird z.B. dadurch erreicht, dass die Menge anhand der Ordnung der drei Indizes (z.B. $i<j<k$) oder anhand der Eigenschaft, wie viele Indizes oberhalb von $\frac{n}{2}$ liegen, zerlegt wird.*

Aus obigem Algorithmus gewinnt Pan auf diese Weise einen Algorithmus von Rang $\frac{n^3-4n}{3}+6n^2$, wir erhalten also mit (Satz 3): $\omega \leq \log_n(\frac{n^3-4n}{3}+6n^2)$. Durch Minimierung erhalten wir (da auch hier erforderlich ist, dass n gerade ist) für $n=70$:

$$\omega \leq \log_{70}\left(\frac{70^3-4\cdot 70}{3}+6\cdot 70^2\right) = \log_{70}(143.640) \approx 2,795.$$

3.3.4 Sonstiges

Pan entwarf auf diese Art immer wieder neue Algorithmen. Vor allem in Kombination mit den in den nächsten Abschnitten vorgestellten Methoden ergaben sich durch seine Konstruktionen mehrfach

neue kleinste obere Schranken. Dabei musste auch das Verfahren teilweise noch verfeinert werden. So konnten in weiteren Konstruktionen beispielsweise teilweise nicht alle Terme durch Uniting und Canceling behandelt werden und es blieben unerwünschte Terme übrig, die sich nicht effizient berechnen ließen. Diese fasste Pan dann in einem weiteren Aggregierungsschritt zusammen. Dabei entstanden erneut unerwünschte Terme und diese konnten noch einmal aggregiert werden. Doch wie schon an obigem Beispiel zu sehen ist, werden all diese Algorithmen sehr unübersichtlich und benötigen sehr viel Platz. Zudem ist der Erkenntnisgewinn durch solch einen Algorithmus begrenzt - auf den ersten Blick ist die Korrektheit nicht erkennbar und der Beweis beschränkt sich auf Nachrechnen. Wir kommen deshalb nun zu den bald darauf entwickelten Techniken von Dario Bini et al.

3.4 Bini: Arbitrary Precision Approximation Algorithmen für die Matrizen-Multiplikation (1979)

Der Begriff der Arbitrary Precision Approximation Algorithmen (im Folgenden APA-Algorithmen) wurde 1979 von Dario Bini, Grazia Lotti und Franceso Romani in [Bi79-2] eingeführt und in [Bi80] von Bini genauer untersucht. Er ist sehr wichtig für alle weiteren Resultate. Dieser erlaubt statt Konstanten in bilinearen und trilinearen Algorithmen Polynome in einer Variablen ε. Auch wenn dieser Begriff von Schönhage in [Schö81] leicht abgewandelt eingeführt wurde, was allgemeinere Resultate ermöglichte[7], halten wir uns hier an die eher einleuchtende Variante von Bini et al. Auch wenn der Begriff des APA-Algorithmus sich auf bilineare bzw. trilineare Algorithmen der MM bezieht, wird er für die tensorielle Zerlegung definiert, da dies einige Argumente vereinfacht. Zur besseren Abgrenzung verwenden wir ab jetzt die Bezeichnungen „approximative Zerlegung“ (für tensorielle Zerlegungen) und „approximativer Algorithmus“ (für bilineare und trilineare Algorithmen). Diese Begriffe verwendet auch Schönhage für seine abgewandelte Form von APA-Algorithmen.

Definition 15. *Sind $u_l(\varepsilon), v_l(\varepsilon), w_l(\varepsilon)$ Vektoren mit Polynomausdrücken in ε über K, dann nennen wir*

$$\sum_{l=1}^{r} u_l(\varepsilon) \otimes v_l(\varepsilon) \otimes w_l(\varepsilon) = t + E(\varepsilon),$$

wobei die Einträge von $E(\varepsilon)$ konstante Terme $= 0$ haben, eine approximative Zerlegung von Länge r. Das kleinste solche r, geschrieben $br(t)$, heißt auch border rank von t.

Bemerkung 3. *Wie bereits angedeutet kann dieser Begriff passend auf bilineare und trilineare Algorithmen übertragen werden, indem Polynome über ε als Koeffizienten in den Algorithmen zugelassen werden. Wir nennen derartige Algorithmen dann approximative bilineare/trilineare Algorithmen und den zugehörigen Rangbegriff ebenfalls border rank einer Menge von Bilinearformen bzw. einer Trilinearform.*

Bemerkung 4. *$E(\varepsilon)$ ist bei einer solchen approximativen Zerlegung ein Fehlerterm, genauer ein Tensor aus dem gleichen Tensorprodukt wie t mit Einträgen, die Polynome über ε sind. Der Begriff des APA-Algorithmus und das Adjektiv „approximativ“ für diese Algorithmen und Zerlegungen stammen daher, dass für $\varepsilon \to 0$ der Fehlerterm $E(\varepsilon)$ beliebig klein wird.*

[7] Schönhage zeigt ein allgemeineres Resultate, sodass die Eigenschaften für den border rank (siehe Definition 15) als Korollar folgen. Daneben erhält er auf diese Art noch die Aussage, dass Algorithmen über beliebigen algebraischen Körpererweiterungen keinen geringeren Rang haben können. Der Begriff der Körpererweiterung soll hier aber nicht behandelt werden

Approximative Algorithmen bzw. Zerlegungen sind besonders deshalb interessant, weil sie tatsächlich bessere Resultate erzielen können als die besten nicht-approximativen Algorithmen bzw. Zerlegungen, d.h. der border rank eines Tensors kann echt kleiner sein als sein Rang (Bini et al verwenden für approximative Algorithmen bzw. Zerlegungen, deren Länge unterhalb des Ranges der zugehörigen Trilinearform liegt, den Begriff „superoptimal“). Dies wurde vor allem anhand einfacher Tensoren gezeigt. Allerdings ist das Bestimmen des border ranks eines Tensors sehr schwierig und es ist damit schwierig, zu beurteilen, ob gewisse Zerlegungen bereits optimal sind. So wurde selbst für das Problem $(2,2,2)$ bzw. den Tensor $\langle 2,2,2\rangle$ erst 2006 von Joseph M. Landsberg - siehe [Land06] - gezeigt, dass Strassens Algorithmus auch unter Berücksichtigung approximativer Algorithmen bzw. Zerlegungen optimal ist.

Lemma 5. *Folgende Resultate übertragen sich direkt vom tensoriellen Rang auf den border rank:*

1. *Der border rank des Problems (m,n,p), der korrespondierenden Trilinearform und des damit assoziierten Tensors sind gleich (vgl. (Satz 2))*
2. *Das tensorielle Produkt zweier zerlegbarer Tensoren t',t'', deren Einträge Polynome in ε sind, ist ein zerlegbarer Tensor t, dessen Einträge Polynome in ε sind. Ist d' der höchste Grad eines Polynoms in t' und d'' der höchste Grad eines Polynoms in t'', so ist der höchste Grad eines Polynoms in t höchstens $d'+d''$ (vgl. (Lemma 1))*
3. *Submultiplikativität (vgl. (Lemma 2))*
4. *Bei Tensoren $\langle m,n,p\rangle$: Gleicher border rank wie für Permutationen der m,n,p (vgl. (Lemma 3))*
5. *Bei Tensoren $\langle m,n,p\rangle$: Symmetrisierung (vgl. (Korollar 3)). Betrachten wir einen bestimmten approximativen Algorithmus und ist bei diesem der Grad der Polynome in den Einträgen des Fehlerterms $\leq d$, so entsteht für $\langle mnp,mnp,mnp\rangle$ eine Zerlegung, in der der Grad der Polynome in den Einträgen des Fehlerterms $\leq 3d$ ist.*

Beweis. Einzig der Grad der Einträge des Fehlerterms für das Produkt zweier Tensoren und bei Symmetrisierung ist hier nicht durch die Beweise der Eigenschaften des (normalen) Ranges abgedeckt. Die Einträge des tensorielle Produktes $t = t' \otimes t''$ sind Produkte der Einträge von t' und t'', die höchste Potenz, die vorkommen kann ist damit $\varepsilon^{d'} \cdot \varepsilon^{d''} = \varepsilon^{d'+d''}$.
Analog dazu ergibt sich bei Symmetrisierung als höchste Potenz, die in der konstruierten Zerlegung vorkommen kann, $\varepsilon^d \cdot \varepsilon^d \cdot \varepsilon^d = \varepsilon^{3d}$. □

Auf diese Weise könnte nun also durch $\varepsilon \to 0$ der Tensor t (oder eine Menge von Bilinearformen bzw. eine Trilinearform) beliebig genau approximiert werden. Hierfür betrachten Bini in [Bi80] und Romani in [Rom80] die numerischen Eigenschaften dieser Approximation. Allerdings interessiert uns vielmehr die Beziehung zu ω, mit der sich Bini et al ebenfalls beschäftigten. Wir kommen nun zu einem Resultat, das es ermöglicht aus approximativen Zerlegungen oder Algorithmen Schranken für ω zu folgern.

Lemma 6. *Sei eine approximative Zerlegung des Tensors t von Länge r und mit Polynomausdrücken von Grad $\leq d$ in $E(\varepsilon)$ gegeben. Dann existiert eine (nicht-approximative) Zerlegung für t von Rang $(1+d)\cdot r$, sofern der zugrunde liegende Körper K ausreichend viele Elemente hat.*

Beweis. Das Ziel ist es, aus der approximativen Zerlegung eine nicht-approximative Zerlegung von t zu konstruieren. Hierfür werden wir für ε verschiedene Werte einsetzen und eine Zerlegung durch Linearkombination der erhaltenen Zerlegungen von $t+E(\varepsilon)$ für die verschiedenen ε gewinnen. Sei hierfür die gegebene approximative Zerlegung

$$t+E(\varepsilon) = \sum_{l=1}^{r} u_l(\varepsilon)v_l(\varepsilon)w_l(\varepsilon)$$

Wähle nun[8] $\varepsilon_1, \ldots, \varepsilon_{d+1} \in K$ mit $\varepsilon_i \neq \varepsilon_j$ für $i \neq j$, $\varepsilon_i \neq 0 \; \forall i$ und betrachte das Vandermonde-System[9]

$$\begin{pmatrix} 1 & 1 & \cdots & 1 \\ \varepsilon_1 & \varepsilon_2 & \cdots & \varepsilon_{d+1} \\ \vdots & \vdots & \ddots & \vdots \\ \varepsilon_1^d & \varepsilon_2^d & \cdots & \varepsilon_{d+1}^d \end{pmatrix} \cdot \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{d+1} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Dieses Gleichungssystem hat eine Lösung, denn die Spalten der Matrix sind offenbar linear unabhängig für $\varepsilon_i \neq \varepsilon_j$. Wir konstruieren nun mit Hilfe der Einträge α_i einer Lösung des Vandermonde-Systems eine Zerlegung von t:

$$\sum_{i=1}^{d+1} \alpha_i \sum_{l=1}^{r} u_l(\varepsilon) \otimes v_l(\varepsilon) \otimes w_l(\varepsilon)$$

Beachte, dass die konstanten Terme der gegebenen (approximativen) Zerlegung genau die Einträge des Tensors t ergeben und dass sämtliche Terme, die eine Potenz von ε enthalten, in $E(\varepsilon)$ stehen. Nach Wahl der α_i ergibt sich damit, wenn wir die Einträge des Tensors $\sum_{l=1}^{r} u_l(\varepsilon) \otimes v_l(\varepsilon) \otimes w_l(\varepsilon)$ einzeln betrachten:
Für einen beliebigen Term a in t:

$$\sum_{i=1}^{d+1} \alpha_i a = (\sum_{i=1}^{d+1} \alpha_i) a = 1 \cdot a = a$$

aufgrund der ersten Zeile des Vandermonde-Systems.
Für ein beliebiges Monom $b\varepsilon^j, j = 1, \ldots, d$ in einem Eintrag von $E(\varepsilon)$ gilt analog:

$$\sum_{i=1}^{d+1} \alpha_i b \varepsilon_i^j = (\sum_{i=1}^{d+1} \alpha_i \varepsilon_i^j) b = 0 \cdot b = 0$$

nach der j-ten Zeile des Vandermonde-Systems. Damit werden auch beliebige Summen von Monomen, die innerhalb des Fehlerterms auftreten, Null und wir erhalten

$$t = \sum_{i=1}^{d+1} \sum_{l=1}^{r} \alpha_i \cdot (u_l(\varepsilon) \otimes v_l(\varepsilon) \otimes w_l(\varepsilon)),$$

also eine (nicht-approximative) Zerlegung von t von Länge $(1+d) \cdot r$. □

Bemerkung 5. *Arnold Schönhage zeigt in [Schö81] ein ähnliches Resultat für beliebige Körper K, also auch für Körper mit $|K| < d+2$ (vgl. (Fußnote 8)).*

Dieses Ergebnis sieht schon relativ gut aus, da es sich bei $1+d$ um einen konstanten Faktor handelt. Wir erheben nun t in Tensorpotenzen und erhalten aus der approximativen Zerlegung von t approximative Zerlegungen der Tensorpotenzen von t mit dem Ziel, dass der konstante Faktor durch Bilden genügend hoher Tensorpotenzen entfällt.

Lemma 7. *Sei eine approximative Zerlegung für den Tensor t von Länge r und mit Polynomausdrücken von Grad $\leq d$ in $E(\varepsilon)$ gegeben. Dann existiert eine approximative Zerlegung für $t^{\otimes k}$ von Länge r^k und mit Polynomausdrücken von Grad $\leq dk$ in $E'(\varepsilon)$. Dabei ist $E'(\varepsilon)$ der Fehlerterm in besagter approximativer Zerlegung für $t^{\otimes k}$.*

[8] hier benötigen wir, dass der Körper K ausreichend viele Elemente hat, genauer gesagt die Null und $d+1$ weitere Elemente, also $|K| \geq d+2$

[9] d.h.: Ein lineares Gleichungssystem Ax = b, wobei A eine Vandermonde-Matrix ist

Beweis. Betrachte hierfür, wie sich die Zerlegung in der Tensorpotenz verhält. Sei dafür $F_l(\varepsilon) = u_l(\varepsilon) \otimes v_l(\varepsilon) \otimes w_l(\varepsilon)$:

$$(\sum_{l=1}^{r} F_l(\varepsilon))^{\otimes k} = \sum_{(s_1,\ldots,s_k)\in\{1,\ldots,r\}^k} F_{s_1}(\varepsilon) \otimes \cdots \otimes F_{s_k}(\varepsilon)$$

Die Summanden in dieser Summe sind tensorielle Produkte von zerlegbaren Tensoren, also nach (Lemma 5) wieder zerlegbar. Damit hat obige Zerlegung Länge

$$|\{1,\ldots,r\}^k| = r^k$$

und die Polynome im Fehlerterm sind von Grad $\leq dk$, da die Produkte, die auftreten, als höchste Monome $\underbrace{\varepsilon^k \cdots \varepsilon^k}_{d\text{-mal}}$ enthalten. □

Satz 5. *Sei eine approximative Zerlegung des Tensors t von Länge r und mit Polynomausdrücken von Grad $\leq d$ in $E(\varepsilon)$ gegeben. Dann existiert eine (nicht-approximative) Zerlegung für $t^{\otimes k}$ von Länge $(1+dk)\cdot r^k$.*

Beweis. Nach (Lemma 7) gibt es eine approximative Zerlegung für $t^{\otimes k}$ von Länge r^k und Polynomausdrücken von Grad $\leq dk$ im Fehlerterm dieser Zerlegung. Nach (Lemma 6) gibt es dann eine (nicht-approximative) Zerlegung für $t^{\otimes k}$ von Länge $(1+dk)\cdot r^k$. □

Korollar 4. *Sei für ein festes m eine approximative Zerlegung des Tensors $\langle m,m,m\rangle$ von Länge r gegeben (oder anders gesagt: Gelte $br(\langle m,m,m\rangle) \leq r$). Dann gilt*

$$\omega \leq \log_m(r).$$

Beweis. Nach (Satz 5) liefern die Voraussetzungen eine (nicht-approximative) Zerlegung für $\langle m^k,m^k,m^k\rangle$ von Länge $(1+dk)\cdot r^k$ für beliebige k. Mit $n := m^k$ erhalten wir

$$\begin{aligned}(1+dk)\cdot r^k &= (1+d\cdot\log_m(n))\cdot(m^{\log_m(r)})^k \\ &= (1+d\cdot\log_m(n))\cdot(m^k)^{\log_m(r)} \\ &= (1+d\cdot\log_m(n))\cdot n^{\log_m(r)}\end{aligned}$$

Wählen wir nun eine feste Größe n_0 (durch Wahl von k) und wenden die Äquivalenz der verschiedenen Rangbegriffe zusammen mit (Satz 3) an, so erhalten wir

$$\begin{aligned}\omega &\leq \log_{n_0}((1+d\cdot\log_m(n_0))(n_0^{\log_m(r)})) \\ &= \log_{n_0}(n_0^{\log_m(r)}) + \log_{n_0}(1+d\cdot\log_m(n_0)) \\ &= \log_m(r) + \frac{\log_m(1+d\cdot\log_m(n_0))}{\log_m(n_0)} \\ &= \log_m(r) + \delta(n_0)\end{aligned}$$

Dabei wird $\delta(n_0) \in O\left(\frac{\log(\log(n_0))}{\log(n_0)}\right)$ monoton kleiner, wenn wir n_0 vergrößern. Es gilt damit:
Für beliebige $\varepsilon > 0$ existiert ein n_0, sodass $\delta(n_0) < \varepsilon$ und somit wegen der Definition von ω als Infimum

$$\omega \leq \log_m(r).$$

□

Ein Beispiel für eine solche approximative Zerlegung wird in (Abschnitt 3.6) in Form der von Bini et al 1979 veröffentlichten Zerlegung gegeben.

3.5 Bini, Capovani, Romani, Lotti: Partielle Matrizen-Multiplikation (1979)

Die partielle MM tauchte zuerst bei Bini et al in [Bi79-1] auf, diese verwendeten sie zur Konstruktion einer approximativen Zerlegung, die eine neue Schranke für ω ergab. Partielle MM meint dabei, dass einige Einträge der beiden Matrizen A und B Null sind und somit sämtliche Produkte, die diese Einträge beinhalten, entfallen. Schönhage griff die Idee in [Schö81] auf, definierte den Begriff der partiellen MM klar und zeigte allgemein, dass sich gewisse Schranken für ω aus bilinearen bzw. trilinearen Algorithmen der partiellen MM bzw. Zerlegungen der assoziierten Tensoren folgern lassen. Hier soll zunächst die Definition von Schönhage angegeben werden, um anschließend besagten Algorithmus von Bini et al als Beispiel einer approximativen Zerlegung und gleichzeitig einer Zerlegung des Tensors einer partiellen MM anzugeben. Danach werden die allgemeinen Betrachtungen von Schönhage vorgestellt.

Definition 16. *Wir beziehen uns auf das Problem (m,n,p) und formulieren hierfür eine partielle Version: Seien A,B,C wieder die beteiligten Matrizen, es soll also $C = A \cdot B$ berechnet werden, wobei die Einträge der Matrizen A und B teilweise Null sind. Dann ist die partielle MM zum Problem (m,n,p) die Berechnung von C und gegeben durch die Zahlen m,n,p und die Indexmengen*

$$I \subseteq \{1,\dots,m\} \times \{1,\dots,n\}$$

$$J \subseteq \{1,\dots,n\} \times \{1,\dots,p\}.$$

Dabei geben die Elemente von I die Positionen in A an, die Variablen enthalten (die übrigen Einträge sind Null), ebenso die Elemente von J für B.
Zur besseren Lesbarkeit nennen wir dieses Problem (für beliebige I,J) auch $\underline{(m,n,p)_{part}}$.

Bemerkung 6. *Diese Definition überträgt sich analog auf die assoziierten Tensoren: Dort werden alle Einträge, die sich auf einen der Null-Einträge von A oder B beziehen, auf Null gesetzt. Wir können damit den Tensor, der mit dem Problem der partiellen MM, die durch (m,n,p) und I,J gegeben ist, angeben als:*

$$t_{(i_2,j_1),(j_2,k_1),(k_2,i_1)} = \delta_{i_1,i_2}\delta_{j_1,j_2}\delta_{k_1,k_2} \cdot \chi_I(i_2,j_1) \cdot \chi_J(j_2,k_1)$$

Dabei ist χ_A die charakteristische Funktion der Menge A, also

$$\chi_A(a) = \begin{cases} 1 & \text{, falls } a \in A \\ 0 & \text{, falls } a \notin A \end{cases}$$

Wir nennen derartige Tensoren (für beliebige I,J) auch $\underline{\langle m,n,p\rangle_{part}}$.

3.6 Bini, Capovani, Romani, Lotti: Beispiel für APA-Algorithmen und partielle Matrizen-Multiplikation

Da es sich bei diesem Beispiel aus [Bi79-1] um ein übergreifendes Beispiel für zwei Techniken handelt, erhält es einen eigenen Abschnitt. Die Schritte der Konstruktion sind dabei:

1. Konstruktion einer approximativen Zerlegung eines Tensors $\langle 2,2,2\rangle_{\text{part}}$
2. Konstruktion einer approximativen Zerlegung des Tensors $\langle 3,2,2\rangle$

3. Folgern einer Schranke für ω durch Symmetrisierung

Schritt 1: Wir betrachten das Problem $(2,2,2)_{\text{part}}$ mit

$$I = \{(1,1),(1,2),(2,2)\}, \quad J = \{1,2\} \times \{1,2\}.$$

Es handelt sich also um eine Multiplikation der Form

$$\begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ 0 & a_{22} \end{pmatrix} \cdot \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}.$$

Um die Zerlegung des damit assoziierten Tensors $\langle 2,2,2 \rangle_{\text{part}}$ leichter angeben zu können, verwenden wir mittels folgender Umbenennung einfache Indizes:

$$a_1 \leftrightarrow a_{11}, a_2 \leftrightarrow a_{12}, a_3 \leftrightarrow a_{22}$$

$$b_1 \leftrightarrow b_{11}, b_2 \leftrightarrow b_{21}, b_3 \leftrightarrow b_{12}, b_4 \leftrightarrow b_{22}$$

$$c_1 \leftrightarrow c_{11}, c_2 \leftrightarrow c_{21}, c_3 \leftrightarrow c_{12}, c_4 \leftrightarrow c_{22}$$

Wir geben nun eine approximative Zerlegung des Tensors an, die Spalten der Matrizen sind dabei jeweils die Vektoren u_l, v_l, w_l (sodass z.B. $u_1 = (u_{i,1})_i$ ist):

$$U = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & \varepsilon & \varepsilon \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$

$$V = \begin{pmatrix} \varepsilon & 0 & 0 & -\varepsilon & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \varepsilon \\ 1 & -1 & 1 & 0 & 1 \end{pmatrix}$$

$$W = \begin{pmatrix} \varepsilon^{-1} & \varepsilon^{-1} & -\varepsilon^{-1} & \varepsilon^{-1} & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -\varepsilon^{-1} & 0 & \varepsilon^{-1} \\ 1 & 0 & 0 & 0 & -1 \end{pmatrix}$$

Dabei handelt sich offenbar um eine approximative Zerlegung von Länge 5. Es lässt sich zeigen, dass dieser Wert kleiner ist als der Rang der beschriebenen partiellen MM. Damit ist dies ein Beispiel eines Tensors, für den der border rank echt kleiner als der Rang ist.

Schritt 2: Es lässt sich aus dieser Zerlegung eine approximative Zerlegung (gleicher Länge) des Tensors $\langle 2,2,2 \rangle_{\text{part}}$ der partiellen MM $(2,2,2)_{\text{part}}$ mit den Mengen $I = \{(1,1),(2,1),(2,2)\}, J = \{1,2\} \times \{1,2\}$ konstruieren. Dies sieht man wie folgt:
Letztere partielle MM hat die Form

$$\begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} = \begin{pmatrix} a_{11} & 0 \\ a_{21} & a_{22} \end{pmatrix} \cdot \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}.$$

Wir können die Einträge der Matrizen jeweils umbenennen, sodass jeweils die Einträge auf derselben Diagonalen tauschen und erhalten:

$$\begin{pmatrix} c_{22} & c_{21} \\ c_{12} & c_{11} \end{pmatrix} = \begin{pmatrix} a_{22} & 0 \\ a_{12} & a_{11} \end{pmatrix} \cdot \begin{pmatrix} b_{22} & b_{21} \\ b_{12} & b_{11} \end{pmatrix}$$

Damit erhalten wir

$$c_{22} = a_{22}b_{22}, \quad c_{21} = a_{22}b_{21},$$

$$c_{12} = a_{12}b_{22} + a_{11}b_{12}, \quad c_{11} = a_{12}b_{21} + a_{11}b_{11},$$

was genau den Gleichungen für die vorherige partielle MM entspricht, für die wir eine tensorielle Zerlegung angegeben haben. Die Umbenennung findet durch einen Tausch der Indizes statt und lässt sich somit durch Permutation der Indizes innerhalb der Zerlegung umsetzen. Es gibt also eine approximative Zerlegung gleicher Länge für diese zweite partielle MM. Nutzen können wir das nun, indem wir die beiden Zerlegungen zu einer approximativen Zerlegung des Tensors $\langle 3,2,2\rangle$ zusammensetzen. Dafür betrachte

$$\begin{pmatrix} a_{11} & a_{12} \\ x_{11} & a_{22} \\ x_{21} & x_{22} \end{pmatrix} \cdot \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

und dafür separat die Produkte

$$\begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ 0 & a_{22} \end{pmatrix} \cdot \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$
$$\begin{pmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{pmatrix} = \begin{pmatrix} x_{11} & 0 \\ x_{21} & x_{22} \end{pmatrix} \cdot \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

Für diese haben wir durch die angegebene approximative Zerlegung bereits gezeigt, dass ihr border rank jeweils ≤ 5 ist. Man sieht nun aber leicht, dass gilt:

$$\begin{pmatrix} c_{11} & c_{12} \\ c_{21}+d_{11} & c_{22}+d_{12} \\ d_{21} & d_{22} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ x_{11} & a_{22} \\ x_{21} & x_{22} \end{pmatrix} \cdot \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

Es lassen sich somit die beiden Zerlegungen zu einer approximativen Zerlegung für $\langle 3,2,2\rangle$ von Rang $5+5=10$ zusammensetzen. Im letzten Schritt nun wenden wir die Symmetrisierung aus (Lemma 5) an und erhalten

$$\omega \leq \log_{12}(1000) \approx 2,7799.$$

3.7 Partielle Matrizen-Multiplikation bei Schönhage (1981)

Wie bereits angedeutet zeigte Schönhage in [Schö81] ein allgemeines Resultat im Hinblick auf Schranken, die sich aus partiellen MMs folgern lassen. Dabei handelt es sich um den folgenden

Satz 6. *Sei t ein Tensor $\langle m,n,p\rangle_{part}$. t enthalte e Einsen und habe einen border rank $br(t) \leq r$. Dann gilt:*

$$\omega \leq 3 \cdot \frac{\ln(r)}{\ln(e)} = 3 \cdot \log_e(r)$$

Beweis. Der Beweis dieses Satzes gliedert sich in die folgenden 5 Schritte:

1. Erheben des Tensors in eine Tensorpotenz
2. Untersuchen der Verteilung von Einsen in dieser Tensorpotenz und Wegstreichen gewisser Zeilen und Spalten, sodass die übrigen Spalten und Zeilen eine gleiche Anzahl von Einsen enthalten
3. Kompression der Matrizen durch geeignete Multiplikation mit weiteren Matrizen. Dadurch entsteht eine vollständige MM, für deren Rang wir eine Schranke erhalten
4. Symmetrisierung nach (Lemma 5)

5. In den vorherigen Schritten sind Variablen aufgetaucht, die wir nun geeignet belegen, um die im Satz genannte Schranke für ω zu erhalten

Schritt 1:
Seien I, J die Mengen, die die Einspositionen im gegebenen Tensor $\langle m,n,p\rangle_{\text{part}}$ beschreiben. Der Tensor dieser MM ist - wie wir oben gesehen haben - gegeben durch

$$t_{(i_2,j_1),(j_2,k_1),(k_2,i_1)} = \delta_{i_1,i_2}\delta_{j_1,j_2}\delta_{k_1,k_2}\chi_I(i_1,j_1)\chi_J(j_1,k_1).$$

Zur Darstellung der Tensorpotenz $t^{\otimes s}$ verwenden wir Multi-Indizes $i_1, i_2, j_1, j_2, k_1, k_2$, die jeweils s-Tupel sind. Ihre Einträge geben wir durch hochgestellte Indizes an:

$$i_1 = (i_1^l), i_2 = (i_2^l), j_1 = (j_1^l), j_2 = (j_2^l), k_1 = (k_1^l), k_2 = (k_2^l).$$

Für die Tensorpotenz ergibt sich dann nach (Abschnitt 2.5):

$$\begin{aligned}
t^{\otimes s}_{(i_2,j_1),(j_2,k_1),(k_2,i_1)} &= \delta_{i_1,i_2}\delta_{j_1,j_2}\delta_{k_1,k_2}\cdot\chi_{I^{(s)}}(i_1,j_1)\chi_{J^{(s)}}(j_1,k_1)\\
&= \prod_{l=1}^{s}(\delta_{i_1^l,i_2^l}\delta_{j_1^l,j_2^l}\delta_{k_1^l,k_2^l})\prod_{l=1}^{s}(\chi_I(i_1^l,j_1^l)\chi_J(j_1^l,k_1^l))\\
&= \prod_{l=1}^{s}(\delta_{i_1^l,i_2^l}\delta_{j_1^l,j_2^l}\delta_{k_1^l,k_2^l}\chi_I(i_1^l,j_1^l)\chi_J(j_1^l,k_1^l))\\
&= \prod_{l=1}^{s} t_{(i_2^l,j_1^l),(j_2^l,k_1^l),(k_2^l,i_1^l)}
\end{aligned}$$

Beachte, dass hier das Kronecker-Delta für die erste Zeile passend auf Multi-Indizes erweitert werden muss: Es hat den Wert 1 genau dann, wenn die beiden Multi-Indizes in allen Komponenten übereinstimmen, sonst 0. Außerdem bezeichnen die Mengen $I^{(s)}, J^{(s)}$, wie unten definiert, die Mengen der Variablenpositionen für die Tensorpotenz $t^{\otimes s}$.
Aufgrund der Submultiplikativität, siehe (Lemma 5), erhalten wir außerdem $br(t^{\otimes s}) \leq br(t)^s$.

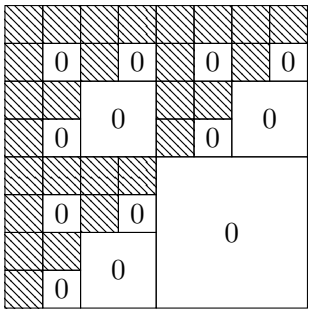
Assoziiert mit $t^{\otimes s}$ ist dann $(m^s, n^s, p^s)_{\text{part}}$ mit

$$I^{(s)} = \left\{ (i_2,j_1) \in \{1,\dots,m\}^s \times \{1,\dots,n\}^s \;\middle|\; \prod_{l=1}^{s}\chi_I(i_2^l,j_1^l) = 1 \right\},$$

$$J^{(s)} = \left\{ (j_2,k_1) \in \{1,\dots,n\}^s \times \{1,\dots,p\}^s \;\middle|\; \prod_{l=1}^{s}\chi_J(j_2^l,k_1^l) = 1 \right\}.$$

Dabei sind i_2, j_1, j_2, k_1 also wieder Multi-Indizes, wobei die gleichen Namen verwendet wurden, um den Bezug zur tensoriellen Darstellung herzustellen. Wir wollen zur Veranschaulichung dieser Schachtelung partieller MMs die Mengen I und $I^{(3)}$ für einen Tensor $\langle 2,2,2\rangle_{\text{part}}$ und seine dritte Tensorpotenz $\langle 2,2,2\rangle^{\otimes 3}_{\text{part}}$ visualisieren, wobei I wie im letzten Abschnitt (Beispiel von Bini et al) gewählt sei. Dazu zeigen wir die Matrix A, deren Aufbau I beschreibt, wobei Variablen-Positionen schraffiert und Null-Positionen mit Nullen markiert sind.
I lässt sich visualisieren als

und $I^{(3)}$ folgendermaßen:



Schritt 2:
Um die Verteilung der Einsen in $t^{\otimes s}$ untersuchen zu können, müssen wir zunächst eine geeignete Beschreibung der Positionen in t finden. Dazu betrachte die mit dem Tensor assozziierte partielle MM $C = AB$. Wir definieren m_j als Anzahl der Variablen-Positionen in der j-ten Spalte von A und p_j als Anzahl der Variablen-Positionen in der j-ten Zeile von B und erhalten für $j \in \{1,\dots,n\}$:

$$m_j = \sum_i \chi_I(i,j)$$
$$p_j = \sum_k \chi_J(j,k)$$

Es werden also einfach die Einsen mit Hilfe der charakteristischen Funktion der Mengen I und J einzeln gezählt. O.B.d.A. seien alle $m_j > 0, p_j > 0$. Würde ein Wert $= 0$ auftauchen, so könnten wir die entsprechende Zeile bzw. Spalte und die zugehörigen Einträge in I und J streichen und würden ein entsprechend angepasstes Problem ohne Nullen erhalten, das den gleichen Rang hat.
Mit Hilfe der m_j, p_j können wir nun die Anzahl der Einsen in t schreiben als

$$e = m_1 p_1 + \cdots + m_n p_n.$$

Dies liegt an folgender Beobachtung: Wir betrachten die MM als Multiplikation

$$C = \begin{pmatrix} a_1 & \cdots & a_n \end{pmatrix} \cdot \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix},$$

wobei die a_j die Spalten der linken Matrix und die b_j die Zeilen der rechten Matrix sind. Dann treten im Ergebnis C alle Einträge einer festen Zeile b_x mit allen Einträgen der zugehörigen festen Spalte a_x als Produkt auf (und zwar auf alle Einträge von C verteilt). Schreiben wir C wie üblich nach dem trivialen Algorithmus, so ist beispielsweise das Produkt von a_{ix} und b_{xk} im Eintrag c_{ik} als x-ter Summand zu finden. Das heißt aber auch, dass damit bereits alle auftretenden Produkte beschrieben werden, entsprechend werden die Einsen im assoziierten Tensor genau durch diese Produkte festgelegt. Nun geben aber die m_j, p_j die Anzahlen der Variablen in den einzelnen Spalten von A und den einzelnen Zeilen von B an, somit ist die Anzahl aller auftretender Produkte und damit aller Einsen im assoziierten Tensor, hier e, durch obige Gleichung gegeben.
Wir gehen nun in die (bereits in Schritt 1 betrachtete) Tensorpotenz $t^{\otimes s}$ für ein festes s über. Außerdem legen wir eine Partition

$$s = \sigma_1 + \cdots + \sigma_n \text{ mit } \sigma_i \geq 0$$

fest, die Anzahl der Summanden entspricht also der Anzahl von Spalten von A bzw. von Zeilen von B. In Schritt 5 werden wir die Partition $s = \sigma_1 + \cdots + \sigma_n$ passend wählen und den Fall

$s \to \infty$ betrachten, um die gewünschte Schranke für ω zu erhalten. Betrachten wir $t^{\otimes s}$ als mehrere ineinanander geschachtelte Tensoren t (s.o.), so sehen wir leicht, dass $t^{\otimes s}$

$$e^s = (m_1 p_1 + \cdots + m_n p_n)^s$$

Einsen enthält. Um unabhängig von der genauen Verteilung von Einsen in den einzelnen Spalten und Zeilen argumentieren zu können, definieren wir die Menge

$$\Phi = \{(\varphi_1, \ldots, \varphi_s) \mid \text{Die Anzahl der Einträge } \varphi_j = i \text{ ist } \sigma_i \text{ für alle } i \in \{1, \ldots, n\}\}$$

und erhalten

$$|\Phi| = \frac{s!}{\sigma_1! \cdots \sigma_n!}.$$

Diese Anzahl ergibt sich dadurch, dass durch die σ_i insgesamt die Anzahlen der einzelnen Werte i in $\varphi \in \Phi$ vorgegeben werden. Es gibt also $s!$ Möglichkeiten nacheinander die Positionen dieser Werte innerhalb von φ auszuwählen. Nun sind aber jeweils σ_i Werte gleich und innerhalb dieser σ_i Werte spielt die Reihenfolge jeweils keine Rolle. Deshalb teilen wir durch $\sigma_1! \cdots \sigma_m!$.
Die mit der i-ten Tensorpotenz $t^{\otimes i}$ assoziierte MM sei jeweils

$$C^{(i)} = A^{(i)} B^{(i)} \text{ für } i \in \{1, \ldots, s\}.$$

Wir wollen nun die Anzahl der Einsen in den Spalten der Matrix $A^{(s)}$ und den Zeilen der Matrix $B^{(s)}$ angeben. Dazu interpretieren wir die $\varphi \in \Phi$ als Multi-Indizes, sodass jedes φ eine Spalte von $A^{(s)}$ und eine Zeile von $B^{(s)}$ auswählt. Hierzu betrachte die Matrizen wieder als ineinander geschachtelte Matrizen, beispielsweise besteht mit dieser Darstellung $A^{(s)}$ aus Blockmatrizen $A^{(s-1)}$, diese wiederum aus Blockmatrizen $A^{(s-2)}$ und so weiter. Die Komponenten von φ wählen dann die Spalten in den verschiedenen Verschachtelungstiefen aus, also zunächst eine Spalte, deren Einträge Matrizen $A^{(s-1)}$ sind, dann darin eine Spalte, deren Einträge Matrizen $A^{(s-2)}$ sind und so weiter.
Nun ist für alle $\varphi \in \Phi$ die Anzahl M von Einsen in Spalte φ von $A^{(s)}$ jeweils gleich und auch die Anzahl P von Einsen in Zeile φ von $B^{(s)}$ jeweils gleich und wir können diese bestimmen.
Für Spalte φ von $A^{(s)}$:
Es wählen (o.E) die $\varphi_1, \ldots, \varphi_s$ der Reihe nach von außen nach innen die Spalten von Blockmatrizen. Dann wählt φ_1 eine Spalte, deren Einträge Matrizen $A^{(s-1)}$ sind, von denen jeweils genau m_{φ_1} ungleich Null sind. Für jede davon wählt φ_2 dieselbe Spalte von Matrizen $A^{(s-2)}$, von denen jeweils genau m_{φ_2} ungleich Null sind. Induktiv erhalten wir damit

$$\begin{aligned} M &= \prod_{i=1}^{s} m_{\varphi_i} \\ &= m_1^{\sigma_1} \cdots m_n^{\sigma_n}. \end{aligned}$$

Letzteres folgt wegen $\varphi \in \Phi$. Analog lässt sich auch P bestimmen und wir erhalten

$$\begin{aligned} M &= m_1^{\sigma_1} \cdots m_n^{\sigma_n} \\ P &= p_1^{\sigma_1} \cdots p_n^{\sigma_n}. \end{aligned}$$

Wir entfernen nun alle Spalten von $A^{(s)}$ und Zeilen von $B^{(s)}$, die nicht durch ein $\varphi \in \Phi$ ausgewählt werden, indem wir alle Einträge in den entsprechenden Zeilen und Spalten auf Null setzen. Durch Weglassen der auf diese Art gelöschten Zeilen und Spalten erhalten wir eine partielle MM $(m^s, |\Phi|, p^s)_{\text{part}}$ (mit passenden Mengen I, J), denn die Anzahl der Zeilen von $A^{(s)}$ und die Anzahl der Spalten von $B^{(s)}$ wurden nicht verändert und es werden offenbar $|\Phi|$ Spalten von $A^{(s)}$ und Zeilen von $B^{(s)}$ übrig gelassen.

Es ist zudem leicht nachvollziehbar, dass der Rang des assoziierten Tensors dieser MM nicht größer als der Rang des mit der vorherigen MM assoziierten Tensors sein kann (da nur Einträge gelöscht wurden). Schönhage verwendet hierfür den Begriff des Subtensors.

Schritt 3:
Wir haben nun also eine partielle MM $(m^s, |\Phi|, p^s)_{\text{part}}$ (mit passenden Mengen I, J). Sei diese MM zur Vereinfachung wieder gegeben als

$$C = AB.$$

Dann wissen wir, dass jede Spalte von A genau M Einsen enthält und jede Zeile von B genau P Einsen. Das Ziel ist nun, aus diesen Matrizen vollständige Matrizen der Größen $M \times |\Phi|$ und $|\Phi| \times P$ zu konstruieren, um daraus zu folgern, dass $br(\langle M, |\Phi|, P\rangle) \leq r^s$ gilt.
Wir wollen also eine (vollständige) MM $W = UV$, beschrieben durch das Problem $(M, |\Phi|, P)$, konstruieren. Dazu legen wir Matrizen G, H der Größen $M \times m^s$ und $p^s \times P$ mit konstanten Einträgen fest, sodass

$$U = GA,\ V = BH,\ W = GABH = GCH.$$

Wir zeigen nun, dass durch passende Wahl der Matrizen G und H erreicht wird, dass die Einträge von A sich als Linearkombinationen der Einträge von U und die Einträge von B als Linearkombinationen der Einträge von V ergeben. Außerdem ergeben sich die Einträge von W als Linearkombinationen der Einträge von C und somit lassen sich approximative Zerlegungen für den Tensor der partiellen MM $C = AB$ umformen zu approximativen Zerlegungen des Tensors $\langle M, |\Phi|, P\rangle$ der Multiplikation $W = UV$ gleicher Länge.
Wir bezeichnen nun die Spalten von A und U als a_j und u_j und die Zeilen von B und V als b_j und v_j. Dann gilt:

$$Ga_j = u_j$$
$$b_j H = v_j$$

Das Ziel ist nun, durch Invertieren von G und H und Multiplikation mit den Inversen die Einträge von A als Linearkombinationen der Einträge von U und die Einträge von B als Linearkombinationen der Einträge von V zu erhalten. Da es sich aber i.A. nicht um quadratische Matrizen handelt brauchen wir hierfür einen Trick.
Für $Ga_j = u_j$ gehen wir dabei wie folgt vor:
Wir wählen für eine beliebige, aber feste Spalte a_j die Zeilen $\tau_1 < \cdots < \tau_M$ aus, die Variablen (und keine Nullen) enthalten. Sei nun G' die Matrix, die aus G durch Wegstreichen aller Spalten entsteht, deren Index keinem der τ_i entspricht. G' hat dann die Größe $M \times M$. Sei a'_j der Vektor, der durch Wegstreichen aller Nullen aus a_j entsteht. Wir streichen also alle Einträge von G, die im Produkt Ga_j ohnehin nur auf Nullen treffen würden und streichen entsprechend auch die Nullen aus a_j. Damit gilt:

$$G'a'_j = u_j$$

Wir wählen nun G so, dass die Determinanten aller Matrizen, die durch Wegstreichen von Zeilen oder Spalten aus G entstehen, ungleich Null sind, alle diese Matrizen also invertierbar sind. Dies kann, wenn der betrachtete Körper ausreichend viele Elemente hat, erreicht werden, indem wir G als Vandermonde-Matrix (siehe Beweis zu (Lemma 6): Vandermonde-System) mit paarweise verschiedenen Elementen des Körpers wählen. Damit haben alle derartigen G' Inverse $(G')^{-1}$ und wir erhalten:

$$a'_j = (G')^{-1} u_j.$$

Somit haben wir also eine Darstellung der Einträge von A als Linearkombinationen der Einträge von U.
Analog erhalten wir die gleiche Aussage auch für die Einträge von B und V. Weiterhin ist $W = GCH$

und damit lassen sich die Einträge von W ohne weitere Umformungen offensichtlich als Linearkombinationen der Einträge von C angeben.
Nun können approximative Zerlegungen des Tensors der partiellen MM $C = AB$ zu approximativen Zerlegungen des Tensors von $W = UV$ gleicher Länge umgeformt werden, was wir leicht sehen, wenn wir die Zerlegungen in bilineare Algorithmen überführen: Wir können im bilinearen Algorithmus, der $C = AB$ berechnet, die Linearkombinationen der Einträge von A und B durch Linearkombinationen der Einträge von U und V ersetzen. Damit ergibt sich ein bilinearer Algorithmus, der C aus U und V berechnet. Nun können die Einträge von W als Linearkombinationen der Einträge von C berechnet werden. Damit erhalten wir einen bilinearen Algorithmus für $W = UV$ der gleichen Länge und können diesen wieder in eine tensorielle Zerlegung überführen. Wir erhalten somit eine approximative Zerlegung gleicher Länge und gleichen Grades für den Tensor $\langle M, |\Phi|, P\rangle$.
Nach Voraussetzung gibt es eine approximative Zerlegung des Tensors t von Länge r und mit Polynomen im Fehlerterm von Grad $\leq d$ für ein beliebiges festes d, also gibt es nach (Lemma 6) eine approximative Zerlegung des Tensors $t^{\otimes s}$ und damit auch des Tensors $\langle M, |\Phi|, P\rangle$ von Länge r^s und mit Polynomen im Fehlerterm von Grad $\leq ds$.

Schritt 4: Durch Symmetrisierung nach (Lemma 5) erhalten wir damit eine approximative Zerlegung von Länge r^{3s} und Grad $3ds$ für den Tensor $\langle X, X, X\rangle$ mit

$$X = M \cdot |\Phi| \cdot P = \frac{s!}{\sigma_1! \cdots \sigma_n!}(m_1 p_1)^{\sigma_1} \cdots (m_n p_n)^{\sigma_n}$$

und nach (Satz 5)

$$r(\langle X, X, X\rangle) \leq (1 + 3ds) r^{3s}.$$

Schritt 5:
Wir entscheiden uns nun bei fest gewähltem s für eine passende Zerlegung

$$s = \sigma_1 + \cdots + \sigma_n.$$

Nach dem Ergebnis aus Schritt 4 ist die resultierende Schranke fest, einzig die Größe der MM lässt sich durch Wahl der σ_i variieren, entsprechend ist das Ziel X zu maximieren. Für die Aussage dieses Satzes genügt dabei folgende Abschätzung: X ist ein einzelner Summand aus der Potenz

$$e^s = (m_1 p_1 + \cdots + m_n p_n)^s = \sum_{(\sigma_1, \ldots, \sigma_n) \text{ mit } s = \sigma_1 + \cdots + \sigma_n} \frac{s!}{\sigma_1! \cdots \sigma_n!}(m_1 p_1)^{\sigma_1} \cdots (m_n p_n)^{\sigma_n}.$$

Die Argumentation ist dabei ähnlich zu der für die Anzahl von Elementen in Φ: Für jede Partition $s = \sigma_1 + \cdots + \sigma_n$ sind die Exponenten der einzelnen $m_i p_i$ durch σ_i vorgegeben. Damit bleibt noch die Reihenfolge zu wählen: e^s ist ein Produkt von s Summen $(m_1 p_1 + \cdots + m_n p_n)$. Wir wählen für jeden der vorgegebenen s Terme nacheinander aus, aus der wievielten dieser Summen er stammen soll. Dadurch ergibt sich $s!$. Es wird allerdings σ_i mal der Term $m_i p_i$ auf diese Art ausgewählt und unter diesen gleichen Termen können wir die jeweiligen Summen, aus denen diese Terme gewählt werden, vertauschen. Somit teilen wir durch $\sigma_1! \cdots \sigma_n!$.
Zählen wir nun die Anzahl dieser Summanden, also die Anzahl möglicher Zerlegungen $s = \sigma_1 + \cdots + \sigma_n$, so können wir X in jedem Fall so wählen, dass

$$X \geq \frac{e^s}{\text{Anzahl der Zerlegungen } s = \sigma_1 + \cdots + \sigma_n}.$$

Die Anzahl der Zerlegungen ist dabei der Binomialkoffizient

$$\binom{s + n - 1}{n - 1}.$$

Dies lässt sich kombinatorisch leicht zeigen: Wir schreiben die Zahl s als Zeichenkette aus s Einsen. Da wir sie in n Summanden zerlegen wollen, kommen noch $n-1$ Trennzeichen dazu, sodass zwischen den Trennzeichen jeweils (als Summe von Einsen) die einzelnen σ_i stehen. Es handelt sich also um eine Zeichenkette der Länge $s+n-1$, die aus s Einsen und $n-1$ Trennzeichen besteht. Nun ist die Anzahl der Zerlegungen von s in n Summanden gleich der Anzahl der Möglichkeiten, in dieser Zeichenkette die $n-1$ Zeichen auszuwählen, die Trennzeichen sein sollen, also genau der obige Binomialkoeffizient.
Damit erhalten wir

$$X \geq \frac{e^s}{\binom{s+n-1}{n-1}} = \frac{e^s}{\frac{(s+n-1)!}{s!(n-1)!}} \geq \frac{e^s}{\frac{(s+n-1)^{n-1}}{(n-1)!}} \geq \frac{e^s}{(s+n-1)^{n-1}}$$

und zusammen mit dem Ergebnis aus Schritt 4 unter Anwendung von (Satz 3)

$$\begin{aligned}
\omega &\leq \frac{\ln((1+3ds)r^{3s})}{\ln(X)} \\
&\leq \frac{\ln(1+3ds)+3s\ln(r)}{\ln\left(\frac{e^s}{(s+n-1)^{n-1}}\right)} \\
&= \frac{\ln(1+3ds)+3s\ln(r)}{\ln(e^s)-\ln((s+n-1)^{n-1})} \\
&= \frac{\ln(1+3ds)+3s\ln(r)}{s\ln(e)-(n-1)\ln(s+n-1)}.
\end{aligned}$$

Für $s \to \infty$ erhalten wir dann

$$\lim_{s\to\infty}\left(\frac{\ln(1+3ds)+3s\ln(r)}{s\ln(e)-(n-1)\ln(s+n-1)}\right) = 3\cdot\frac{\ln(r)}{\ln(e)} = 3\cdot\log_e(r).$$

Wir können also Exponenten finden, die beliebig nahe an diesen Wert herankommen, indem wir entsprechend hohe Tensorpotenzen bilden und da ω als Infimum definiert ist erhalten wir

$$\omega \leq 3\cdot\frac{\ln(r)}{\ln(e)} = 3\cdot\log_e(r).$$

□

(Satz 3) ergibt sich damit nun sogar als Spezialfall dieses Satzes für $e = mnp$ (beachte allerdings, dass (Satz 3) im Beweis verwendet wurde). Somit lässt sich beispielsweise auch der Exponent von Strassen aus diesem Satz folgern. Für die approximative Zerlegung von Bini et al von Länge 5 für den Tensor, der zur Multiplikation einer 2×2-Matrix mit 3 Einträgen und einer vollen 2×2-Matrix assoziiert ist und der 6 Einsen enthält, ergibt sich mit diesem Satz ohne weitere Konstruktion die Schranke $\omega \leq 3\cdot\log_6(5) \approx 2{,}695$, was bereits eine bessere Schranke ist, als diejenige von Bini et al, die durch manuelle Konstruktion aus der approximativen Zerlegung gewonnen wurde. Darüber hinaus gibt Schönhage noch einige Beispiele an, von denen eines hier ebenfalls beschrieben werden soll, da es sich in leicht abgewandelter Form anschließend auch als Beispiel für den folgenden Abschnitt eignet.

Beispiel 2. Wir beschreiben einen Algorithmus für eine partielle MM $(m,n,p)_{\text{part}}$. Dabei verwenden wir zwei Parameter x,y und wählen

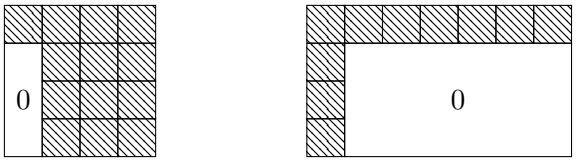
$$m = x, \quad n = y+1, \quad p = 1+(x-1)(y-1).$$

Die Mengen I, J, die die Variablenpositionen der partiellen MM festlegen, seien gegeben durch

$$I = \{(i,j) \mid i = 1 \text{ oder } j > 1\} \subseteq \{1,\dots,m\} \times \{1,\dots,n\},$$

$$J = \{(j,k) \mid j = 1 \text{ oder } k = 1\} \subseteq \{1,\dots,n\} \times \{1,\dots,p\}.$$

Zur Veranschaulichung hier eine visuelle Darstellung von I (links) und J (rechts) für $x = 4, y = 3$, also $m = 4, n = 4, p = 7$:



Nach (Gleichung (3.7)) aus Schritt 2 des vorherigen Beweises ist die Anzahl der Einsen im mit diesem Problem assoziierten Tensor

$$e = 1 + (x-1)(y-1) + xy = 1 + xy - x - y + 1 + xy = 2xy - x - y + 2$$

Die korrespondierende Trilinearform ist (dies lässt sich anhand des Bildes nachvollziehen):

$$T = \sum_{v=1}^{p} a_{1,1} b_{1,v} c_{v,1} + \sum_{i=1}^{m} \sum_{j=2}^{n} a_{ij} b_{j,1} c_{1,i}.$$

Für diese Trilinearform gibt Schönhage den folgenden Algorithmus an (beachte, dass der Algorithmus bei Schönhage aufgrund der leicht anderen Definition der approximativen Algorithmen und Zerlegungen leicht anders aussieht):

$$\begin{aligned}
T + E(\varepsilon) = {} & (a_{1,1} + \varepsilon^2 a_{1,2})(b_{2,1} + \varepsilon^2 b_{1,1})\varepsilon^{-2} c_{1,1} \\
& + \sum_{j=3}^{n} \left((a_{1,1} + \varepsilon^2 a_{1,j})\varepsilon^{-2} b_{j,1} \left(c_{1,1} - \sum_{i=3}^{m} \varepsilon c_{v,1} \right) \right) \\
& + \sum_{i=2}^{m} \left((a_{1,1} + \varepsilon^2 a_{i,2}) \left(b_{2,1} - \sum_{j=3}^{n} \varepsilon b_{1,v} \right) \varepsilon^{-2} c_{1,i} \right) \\
& + \sum_{i=2}^{m} \sum_{j=3}^{n} ((a_{1,1} + \varepsilon^2 a_{i,j})(b_{j,1} + \varepsilon b_{1,v})(\varepsilon^{-2} c_{1,i} + \varepsilon^{-1} c_{v,1})) \\
& - \varepsilon^{-2} a_{1,1} (b_{2,1} + b_{3,1} + \cdots + b_{n,1})(c_{1,1} + c_{1,2} + \cdots + c_{1,m})
\end{aligned}$$

Dabei soll der Index v durch eine Bijektion $\{2,\dots,m\} \times \{3,\dots,n\} \to \{2,\dots,p\}$ von den Indizes i, j festgelegt werden, z.B. durch $v = i + (j-3)(k-1)$. Die Korrektheit dieses Algorithmus ergibt sich durch Nachrechnen. Die Länge ist offenbar

$$\begin{aligned}
1 + (n-2) + (m-1) + (m-1)(n-2) + 1 &= (n-1) + (m-1)(n-1) + 1 \\
&= m(n-1) + 1 \\
&= xy + 1.
\end{aligned}$$

Damit ergibt sich mit (Satz 6)

$$\omega \leq 3 \frac{\ln(1+xy)}{\ln(2xy - x - y + 2)}$$

und durch Minimierung über x und y erhält Schönhage damit

$$\omega \leq 3 \frac{\ln(17)}{\ln(26)} \approx 2{,}6087.$$

Wie bereits erwähnt, werden wir dieses Beispiel in leicht abgewandelter Form auch für die im nächsten Abschnitt eingeführte disjunkte Matrizen-Multiplikation verwenden können. Die Schranke für ω, die sich daraus ergibt wird sogar besser sein also die hier gewonnene.

3.8 Schönhage: Disjunkte Matrizen-Multiplikation (1981)

Die nächste und noch bedeutendere Idee von Schönhage, ebenfalls in [Schö81] veröffentlicht, war es nun, nicht nur den Rang von Tensoren der MM oder der partiellen MM zu betrachten, sondern auch den Rang von Tensoren, die die Berechnung mehrerer disjunkter MMs beschreiben. Mit disjunkt ist dabei gemeint, dass die verschiedenen MMs sich in keiner Variablen überschneiden. Intuitiv klingt es wenig einleuchtend, dass die Berechnung mehrerer völlig unabhängiger Probleme einen Vorteil im Vergleich zur separaten Berechnung dieser Probleme bringen könnte. So formulierte auch Strassen eine berühmte Vermutung, die vor allem als Direct Sum Conjecture bekannt ist. Diese bezieht sich auf beliebige arithmetische Probleme und sagt aus, dass sich die Anzahl von arithmetischen Operationen für ein solches zusammengesetztes Problem immer aus den Anzahlen von arithmetischen Operationen in Algorithmen für die Teilprobleme summiert. Diese Vermutung wurde im allgemeinen Fall und auch für die MM weder bewiesen noch widerlegt, allerdings stellte sich heraus, dass sie zumindest für approximative Zerlegungen bzw. Algorithmen nicht gilt. Es ist also möglich, dass für die Berechnung mehrerer disjunkter Probleme weniger Operationen benötigt werden, als für die Berechnung der Teilprobleme einzeln, sofern wir approximative Algorithmen zulassen. Aufgrund dieser Beobachtung ist es besonders interessant, zu untersuchen, welche Schranken sich allgemein aus derartig zusammengesetzten Problemen folgern lassen.
Zunächst wollen wir nun definieren, wie wir solche zusammengesetzten Probleme formal fassen wollen.

Definition 17. *Seien t',t'' zwei Tensoren. Dann ist die disjunkte Summe dieser Tensoren, geschrieben $t = t' \oplus t''$, der Tensor, der die Berechnung der beiden Trilinearformen t',t'' auf einmal beschreibt.*

Sind t',t'' Tensoren von Ordnung drei (also zu Trilinearformen assoziiert) und stellen wir diese als 3-dimensionale Arrays dar, so ist $t' \oplus t''$ der Tensor, in dem beide Tensoren t' und t'' diagonal nebeneinander angeordnet sind. Es gibt also keine Einträge der beiden verschiedenen Tensoren, die irgendeinen Index gemeinsam haben (dies ist nur für verschiedene Einträge desselben Tensors möglich). Dadurch gelten für derartige disjunkte Summen und das tensorielle Produkt Distributivgesetze, auch als Produkt einer solchen disjunkten Summe mit einem anderen Tensor wird also wieder eine disjunkte Summe zweier Tensoren erhalten. Auch können wir eine Aussage über den Rang disjunkter Summen von Tensoren treffen.

Lemma 8. *Für $\otimes$ und $\oplus$ gelten die Distributivgesetze*

$$t \otimes (t' \oplus t'') = (t \otimes t') \oplus (t \otimes t'')$$
$$(t' \oplus t'') \otimes t = (t' \otimes t) \oplus (t'' \otimes t).$$

Zudem gilt für den Rang einer disjunkten Summe zweier Tensoren:

$$r(t' \oplus t'') \leq r(t') + r(t'')$$

Beweis. Dass die Distributivgesetze gelten, ist folgendermaßen leicht nachvollziehbar: Auf der linken Seite werden zunächst die beiden Tensoren t' und t'' unabhängig voneinander im entsprechend

größeren Tensor $t' \oplus t''$ angeordnet. Anschließend werden alle Einträge von t mit allen Einträgen von $t' \oplus t''$ multipliziert und in einem noch größeren Tensor angeordnet. Nach unseren bisherigen Beschreibungen von $\otimes$ werden dabei t und $t' \oplus t''$ gewissermaßen ineinander geschachtelt. Es entsteht also ein Tensor, der Blöcke enthält, die jeweils einzelne Einträge von t multipliziert mit $t' \oplus t''$ sind. Damit überlagern sich nie die Teile $t \cdot t'$ und $t \cdot t''$ dieser Blöcke (für beliebige Blöcke $t_{ijk} \cdot t'$, $t_{i'j'k'} \cdot t''$ gibt es keine Einträge aus den zwei verschiedenen Blöcken, für die einer der Indizes in $t \otimes (t' \oplus t'')$ übereinstimmt) und durch Permutation der Werte der Indizes erhalten wir genau $(t \otimes t') \oplus (t \otimes t'')$. Der zweite Fall folgt analog.
Die Aussage über die Ränge gilt, da wir aus beliebigen Zerlegungen für t' von Länge r_1 und t'' von Länge r_2 immer eine Zerlegung von $t' \oplus t''$ von Länge $r_1 + r_2$ zusammensetzen können - denn für $t' \oplus t''$ genügt es, unabhängig voneinander t' und t'' zu zerlegen (wie bereits erwähnt, muss allerdings - zumindest für approximative Algorithmen - nicht zwingend jede Zerlegung von $t' \oplus t''$ auf diese Art entstehen). □

Wir wollen nun aus Schranken für den border rank solcher disjunkter Summen von Tensoren partieller MMs Schranken für ω ableiten. Hierfür führen wir noch folgende Notation ein:

Definition 18. *Wir schreiben die n-fache Summe eines Tensors auch als*

$$n \odot t := \underbrace{t \oplus \cdots \oplus t}_{n\text{-mal}}.$$

Zudem wollen wir noch folgende Vereinfachung einführen:

Bemerkung 7. *Im Allgemeinen ist $t' \otimes t'' \neq t'' \otimes t'$, $\otimes$ ist also nicht kommutativ. Allerdings ist hierbei nur die Verschachtelung umgekehrt und durch eine geeignete Permutation der Werte der Indizes lassen sich die beiden Produkte ineinander überführen. Inbesondere haben die beiden Tensoren auch den gleichen Rang. Für unsere Betrachtungen genügen diese Eigenschaften und somit schreiben wir im Folgenden aufgrund dieser Isomorphie einfach $t' \otimes t'' = t'' \otimes t'$.*

Es ergibt sich in Bezug auf Schranken für ω, die sich aus Schranken für den border rank disjunkter Summen partieller MMs ableiten lassen, nach Schönhage folgender

Satz 7 (Schönhages τ-Theorem)**.** *Seien $t_1, \ldots, t_k$ Tensoren partieller MMs und enthalte t_i genau e_i Einsen für alle $i \in \{1, \ldots, k\}$. Sei der border rank ihrer disjunkten Summe $br(t_1 \oplus \cdots \oplus t_k) \leq r$. Dann gilt*

$$\omega \leq 3\alpha,$$

wobei α gegeben ist durch

$$e_1^\alpha + \cdots + e_k^\alpha = r. \tag{3.2}$$

Beweis. Wir beschränken uns hier auf den Fall einer disjunkten Summe von Tensoren der MM, also keiner partiellen MMs. Der Beweis für den Fall partieller MMs ist technisch relativ aufwendig, weshalb wir hier darauf verzichten wollen - die Ideen dafür finden sich ohnehin bereits im Beweis zu (Satz 6).
Wir gehen also davon aus, dass t_i Tensoren (vollständiger) MMs sind und schreiben

$$t_i = \langle m_i, n_i, p_i \rangle \text{ und}$$

$$e_i = m_i n_i p_i.$$

Wir können zudem folgende Sonderfälle ignorieren:
Ist $k = 1$, so entspricht die Aussage (Satz 6) (sogar für den partiellen Fall).

Ist $\alpha = 1$, so ist die Aussage des Satzes $\omega \leq 3$, was trivial gilt.
Da zudem $br(t_1 \oplus \cdots \oplus t_k) \geq k$ gilt, was intuitiv nachvollziehbar ist, erhalten wir

$$e_1 + \cdots + e_k = e_1^1 + \cdots + e_k^1 > e_1^\alpha + \cdots + e_k^\alpha = r \geq k \geq 2. \tag{3.3}$$

Der Beweis gliedert sich nun in folgende Schritte:

1. Definieren einer Folge, die gegen α konvergiert
2. Erheben des Tensors $t_1 \oplus \cdots \oplus t_p$ in die s-te Tensorpotenz und ableiten einer Schranke für den Rang eines gewissen darin enthaltenen (Sub-)Tensors
3. Unter Verwendung des Ergebnisses von (Schritt 2) induktiv zeigen, dass für alle Elemente x der Folge aus (Schritt 1) gilt: $\omega \leq 3x$

Schritt 1: In diesem Schritt wollen wir eine Folge definieren, die strikt gegen die Lösung α von (Gleichung (3.2)) konvergiert. Dazu betrachten wir die Funktion

$$\varphi(x) = \frac{\ln(\sum_{i=1}^k e_i^x)}{\ln(r)},$$

die für $x \in \{0, \ldots, 1\} \subseteq \mathbb{R}$ definiert ist. Offenbar gilt dann $\varphi(\alpha) = 1$. Wir zeigen nun zunächst, dass ein solches α existiert und eindeutig ist:
Aus (Gleichung (3.3)) wissen wir, dass $\sum_{i=1}^k e_i > k$ und somit gibt es ein $e_i > 1$. Das heißt aber, dass die Funktion $\varphi(x)$ strikt monoton steigt. Später brauchen wir noch folgende, präzisere Aussage über die Ableitung, die sich aus dem gleichen Grund ergibt:

$$\varphi'(x) \geq c_0 \text{ für eine Konstante } c_0 > 0 \tag{3.4}$$

Weiterhin sehen wir, dass

$$\varphi(0) = \frac{\ln(\sum_{i=1}^k e_i^0)}{\ln(r)} = \frac{\ln(k)}{\ln(r)} \leq 1 \quad \text{und}$$

$$\varphi(1) = \frac{\ln(\sum_{i=1}^k e_i^1)}{\ln(r)} > \frac{\ln(r)}{\ln(r)} = 1$$

und nach dem Zwischenwertsatz gibt es dann ein eindeutig bestimmtes $\alpha \in [0, 1)$, sodass $\varphi(\alpha) = 1$. Der Zwischenwertsatz liefert dabei zunächst $\alpha \in [0, 1]$, aber wegen $\varphi(1) > 1$ ergibt sich die offene Intervallgrenze.
Die Folge, die wir konstruieren wollen, soll sich von oben an α annähern. Wir machen dafür zunächst folgende Beobachtungen. Wählen wir ein beliebiges $x > \alpha$, so ist $\sum_{i=1}^k e_i^x > r$. Außerdem können wir für

$$(\sum_{i=1}^k e_i^x)^y = r \tag{3.5}$$

den Wert von y bestimmen als $y = \frac{1}{\varphi(x)}$. Um dies zu zeigen setzen wir $a = \sum_{i=1}^k e_i^x$ und erhalten

$$a^{(\frac{1}{\varphi(x)})} = a^{(\frac{\ln(r)}{\ln(a)})} = a^{\log_a(r)} = r.$$

Außerdem erhalten wir wegen $y < 1$ und $e_i^x \geq 1$ die Ungleichung

$$(\sum_{i=1}^k e_i^x)^y < \sum_{i=1}^k (e_i^{xy})$$

und zusammen mit (Gleichung (3.5)) folgt $r < \sum_{i=1}^{k} e_i^{xy}$ und wir erhalten

$$xy > \alpha.$$

Wir formulieren aufgrund dieser Beobachtungen nun folgenden iterativen Algorithmus, der α berechnet bzw. folgende Folge, die gegen α konvergiert:
Wir definieren die Folge rekursiv durch

$$x_0 = 1, \quad x_{j+1} = x_j y_j$$

$$\text{mit } y_j = \frac{1}{\varphi(x_j)} = \frac{\ln(r)}{\ln(\sum_{i=1}^{k} e_i^{x_j})}.$$

Es ist nun noch zu beweisen, dass diese Folge bzw. dieser iterative Algorithmus tatsächlich gegen α konvergiert. Wir nennen dafür den Fehlerterm in den einzelnen Schritten

$$\delta_j = x_j - \alpha$$

und wollen zeigen, dass $\lim_{j\to\infty}(\delta_j) = 0$.
Dazu betrachten wir das Verhältnis von δ_j zu δ_{j+1} und werden sehen, dass es eine Konstante $0 < c < 1$ gibt, sodass $\delta_{j+1} \leq c\delta_j$ für alle j. Damit ist $\delta_j \leq c^j \delta_0$ und wir erhalten die Konvergenz gegen 0.
Es bleibt also zu zeigen, dass es eine solche Konstante c gibt. Dazu betrachte ein festes $x_j = \alpha + \delta_j$ mit $\delta_j > 0$. Diese Bedingung wird von allen Folgengliedern erfüllt. Für die Fehlerterme gilt

$$\begin{aligned}
\delta_{j+1} = x_{j+1} - \alpha &= x_j y_j - (x_j - \delta_j) \\
&= x_j \left(\frac{1}{\varphi(x_j)} - 1 \right) + \delta_j \\
&= \delta_j - x_j \left(1 - \frac{1}{\varphi(x_j)} \right) \\
&= \delta_j - (\alpha + \delta_j) \left(1 - \frac{1}{\varphi(x_j)} \right).
\end{aligned}$$

Nun benutzen wir, dass $\varphi'(x) \geq c_0$ für eine Konstante $c_0 > 0$ ist. Wir erhalten damit

$$\varphi(x_j) = \varphi(\alpha + \delta_j) \geq \varphi(\alpha) + c_0\delta_j = 1 + c_0\delta_j$$

Damit können wir obige Gleichung umformen zu (beachte für das Ungleichheitszeichen, dass $\varphi(x_j)$ ein positives Vorzeichen hat):

$$\begin{aligned}
\delta_{j+1} = \delta_j - (\alpha + \delta_j) \left(1 - \frac{1}{\varphi(x_j)} \right) &\leq \delta_j - (\alpha + \delta_j) \left(\frac{1 + c_0\delta_j}{1 + c_0\delta_j} - \frac{1}{1 + c_0\delta_j} \right) \\
&= \delta_j - (\alpha + \delta_j) \left(\frac{c_0\delta_j}{1 + c_0\delta_j} \right) \\
&= \delta_j - \alpha\delta_j \frac{c_0}{1 + c_0\delta_j} - \delta_j^2 \frac{c_0}{1 + c_0\delta_j} \\
&= \delta_j (1 - \alpha \frac{c_0}{1 + c_0\delta_j}) - \delta_j^2 \frac{c_0}{1 + c_0\delta_j}.
\end{aligned}$$

Da alle beteiligten Konstanten $\alpha, c_0, \delta_j > 0$ sind, sehen wir leicht, dass

$$\delta_{j+1} < \delta_j,$$

die Fehlerterme nehmen also strikt monoton ab. Daraus können wir nun aber folgern, dass $\delta_j \leq \delta_0 = 1-\alpha$ und erhalten somit

$$\frac{c_0}{1+c_0\delta_j} \geq \frac{c_0}{1+c_0(1-\alpha)} =: c_1$$

für eine Konstante $c_1 > 0$ und durch Einsetzen in obige Gleichung schließlich

$$\begin{aligned}\delta_{j+1} &\leq \delta_j(1-\alpha\frac{c_0}{1+c_0\delta_j}) - \delta_j^2\frac{c_0}{1+c_0\delta_j} \\ &\leq \delta_j(1-\alpha c_1) - \delta_j^2 c_1.\end{aligned}$$

Wir haben also gezeigt, dass δ_{j+1} im Vergleich zu δ_j jeweils mindestens um den konstanten Faktor $(1-\alpha c_1)$ abnimmt (denn $\delta_j^2 c_1$ ist positiv) und somit gilt $\lim_{j\to\infty}(\delta_j) = 0$, die Folge der x_i konvergiert also gegen α.

Bemerkung 8. *Schönhage geht hierbei auch noch auf die numerischen Eigenschaften dieser Annäherung ein, um zu zeigen, dass es sich um ein praktikables Verfahren zur Berechnung von α handelt. Für diesen Satz benötigen wir diese Eigenschaft allerdings nicht.*

Schritt 2: Wir wollen nun also den Rang von Tensorpotenzen des Tensors der gegebenen disjunkten MM untersuchen. Dazu lassen sich zunächst die Voraussetzungen des Satzes zusammen mit der Voraussetzung, dass es sich um vollständige MMs handelt, schreiben als

$$br(\langle m_1,n_1,p_1\rangle \oplus \cdots \oplus \langle m_k,n_k,p_k\rangle) \leq r.$$

Wir schreiben die s-te Tensorpotenz dann als

$$(\langle m_1,n_1,p_1\rangle \oplus \cdots \oplus \langle m_k,n_k,p_k\rangle)^{\otimes s} = \bigoplus_{(s_1,\ldots,s_k):s_1+\cdots+s_k=s} \left(\frac{s!}{s_1!\cdots s_k!} \odot \langle \prod_i m_i^{s_i}, \prod_i n_i^{s_i}, \prod_i p_i^{s_i}\rangle\right).$$

Dies ergibt sich aus den Distributivgesetzen und (Bemerkung 7). Der Bruch ergibt sich wieder dadurch, dass alle s Tensoren, die aus den einzelnen Faktoren der Tensorpotenz gewählt werden, durch die Partition von s vorgegeben sind, die Positionen der gleichen Tensoren aber untereinander getauscht werden können.
Wir erhalten dann aufgrund der Submultiplikativität des border ranks (siehe (Lemma 5)):

$$br\left(\bigoplus_{(s_1,\ldots,s_k):s_1+\cdots+s_k=s} \left(\frac{s!}{s_1!\cdots s_k!} \odot \langle \prod_i m_i^{s_i}, \prod_i n_i^{s_i}, \prod_i p_i^{s_i}\rangle\right)\right) \leq r^s$$

Um die folgende Argumentation zu vereinfachen wählen wir eine feste Partition

$$s = s_1 + \cdots + s_k$$

(wie wir diese wählen können sehen wir in Schritt 3). Dadurch erhalten wir den Tensor

$$E \odot \langle M,N,P\rangle,$$

wobei

$$E = \frac{s!}{s_1!\cdots s_k!}, \quad M = \prod_i m_i^{s_i},\ N = \prod_i n_i^{s_i},\ P = \prod_i p_i^{s_i}.$$

Dieser Tensor ist einer der Summanden der obigen Tensorpotenz, entsteht also durch Weglassen gewisser Zeilen und Spalten daraus (es handelt sich nach Schönhages Definition wieder um einen Subtensor). Der border rank dieses Tensors ist somit ≤ dem border rank der Tensorpotenz:

$$br\left(E \odot \langle M,N,P\rangle\right) \leq r^s$$

Aufgrund der Distributivgesetze ergibt sich ähnlich zur Symmetrisierung (beachte (Bemerkung 7)):

$$\left(E \odot \langle M,N,P\rangle\right) \otimes \left(E \odot \langle N,P,M\rangle\right) \otimes \left(E \odot \langle P,M,N\rangle\right)$$

$$= \underbrace{\left(\langle M,N,P\rangle \oplus \cdots \oplus \langle M,N,P\rangle\right)}_{\text{E-mal}} \otimes \underbrace{\left(\langle N,P,M\rangle \oplus \cdots \oplus \langle N,P,M\rangle\right)}_{\text{E-mal}} \otimes \underbrace{\left(\langle P,M,N\rangle \oplus \cdots \oplus \langle P,M,N\rangle\right)}_{\text{E-mal}}$$

$$= E^3 \odot \langle MNP, MNP, MNP\rangle$$

Damit erhalten wir

$$br(E^3 \odot \langle X,X,X\rangle) \leq r^{3s}, \text{ wobei}$$

$$X = MNP = \prod_i m_i^{s_i} \cdot \prod_i n_i^{s_i} \cdot \prod_i p_i^{s_i} = \prod_i (m_i n_i p_i)^{s_i} = \prod_i e_i^{s_i}.$$

Wenn wir diese Argumentation auf eine einzelne approximative Zerlegung beziehen, deren Länge $\leq r$ ist und deren Einträge im Fehlerterm von Grad $\leq d$ sind, so erhalten wir auf diese Art eine approximative Zerlegung der Länge r^{3s}, in der die Einträge des Fehlerterms von Grad $\leq 3ds$ sind. Mit (Satz 5) erhalten wir schließlich[10]

$$r(E^3 \odot \langle X,X,X\rangle) \leq (1+3ds)r^{3s}.$$

Das bedeutet aber, dass wir E^3 verschiedene Probleme (X,X,X) mit nicht mehr als $(1+3ds)r^{3s}$ Multiplikationen berechnen können. Wir könnten dies nun nutzen, um das Problem (EX,EX,EX) zu lösen, da sich dieses Problem (durch Schachtelung) offenbar mit E^3 Multiplikationen von je 2 Matrizen der Größe $X \times X$ berechnen lässt, was genau dem Problem (X,X,X) entspricht.
Allerdings entsprechen die E^3 Multiplikationen nur dem trivialen Algorithmus, wir können stattdessen auch andere Schranken für die Anzahl von Multiplikationen, die für das Problem (q,q,q) (für ein beliebiges, aber festes q) nötig sind, verwenden.

Definition 19. *Wir wollen die Anzahl der nötigen Multiplikationen, um das Problem (q,q,q) zu lösen, mit $m^*(q)$ bezeichnen.*

Es gilt dann

$$m^*(q) \leq E^3 \Rightarrow m^*(qX) \leq (1+3ds)r^{3s}.$$

Können wir also eine Multiplikation zweier $q \times q$-Matrizen mit $\leq E^3$ Multiplikationen durchführen, so können wir die Einträge auch durch $X \times X$-Matrizen ersetzen und können die Multiplikation der entstehenden beiden $qX \times qX$-Matrizen mit höchstens $(1+3ds)r^{3s}$ Multiplikationen durchführen. Die dabei entstehenden Schranken können auf diese Art rekursiv immer weiter verbessert werden, um dann schließlich eine Schranke für ω abzuleiten.

Schritt 3: Das Ziel ist nun mit den Ergebnissen aus Schritt 2 zu zeigen, dass für alle Glieder x_j der in Schritt 1 definierten Folge $\omega \leq 3x_j$ ist. Aufgrund der Konvergenz dieser Folge gegen α ergibt sich damit $\omega \leq 3\alpha$, also die Aussage dieses Satzes. Da ω als Infimum definiert ist, genügt es dafür zu zeigen: Für alle x_j gilt, dass für alle $x > x_j$ das Problem (N,N,N) in $O(N^x)$ gelöst werden kann. N bezieht sich dabei nicht auf das N aus Schritt zwei, vielmehr sind allgemeine symmetrische MMs

[10] Beachte, dass Schönhage hier eine leicht andere Schranke erhält, da er die Abschätzung für beliebige Körper verwendet

gemeint. Aufgrund der O-Notation können wir unser Ziel damit auf folgende Weise formulieren:
Für alle $j \in \mathbb{N}$ gilt: Für jedes $x > x_j$ gibt es ein $N_0(x)$, sodass

$$m^*(N) \leq N^{3x} \quad \forall N \geq N_0(x).$$

Wir zeigen dies durch Induktion über j:
IV: Die Aussage ist erfüllt für x_j.
IA, $j = 0$: Da $m^*(N) \leq N^3$ $\forall N$ ist die Aussage trivial erfüllt.
IS, $j \to j+1$: Wir wählen zunächst ein festes aber beliebiges

$$x' > x_{j+1} = x_j y_j$$

Nun wählen wir zusätzlich ein festes x, sodass

$$x > x_j \text{ und } xy_j < x'.$$

Dies kann zum Beispiel erreicht werden durch $x := \frac{1}{2}(x_j + \frac{x'}{y_j})$. Das Ziel ist nun, zu zeigen, dass für jedes solche x' eine Schranke $N_0(x')$ existiert, sodass die IV für x_{j+1} erfüllt wird.
Wir betrachten dafür zunächst ein festes s und die Zerlegung $s = s_1 + \cdots + s_k$, für die der Term EX^{x_j} maximal wird. Nach Definition von E und X ist

$$EX^{x_j} = \frac{s!}{s_1! \cdots s_k!} \prod_i e_i^{x_j s_i}.$$

Damit ist EX^{x_j} einer der Summanden von

$$\left(\sum_i e_i^{x_j} \right)^s = \sum_{(s_1,\ldots,s_k): s = s_1 + \cdots + s_k} \frac{s!}{s_1! \cdots s_k!} \prod_i e_i^{x_j s_i}.$$

Nun gilt aber $EX^x > EX^{x_j}$ und zudem können wir, da EX^{x_j} unter den Summanden der obigen Summe maximal gewählt wurde, wie in Schritt 5 von (Satz 6) argumentieren und erhalten

$$EX^x > EX^{x_j} \geq \frac{\left(\sum_i e_i^{x_j} \right)^s}{\binom{s+k-1}{k-1}}.$$

Als nächstes wählen wir das maximale q mit

$$m^*(q) \leq E^3.$$

Dabei gilt

$$m^*(q) > \frac{1}{8} E^3.$$

Widerspruchsbeweis: Wäre diese Bedingung nicht erfüllt, so könnten wir den Algorithmus, der $m^*(q)$ Multiplikationen benötigt, mit dem trivialen Algorithmus für das Problem $(2,2,2)$ schachteln (vgl. (Satz 3)) und somit erhalten: $m^*(2q) \leq \frac{1}{8} E^3 \cdot 2^3 = E^3$, q wäre also nicht maximal.
Wir benötigen nun $q \geq N_0(x)$, um die IV für x einsetzen zu können. Um zu zeigen, dass q beliebig groß wird für entsprechend große s genügt es wegen $q^3 \geq m^*(q) > \frac{1}{8} E^3$ zu zeigen, dass E beliebig groß wird für entsprechend große s.
Hierfür betrachte für große s den Term EX^{x_j} und dabei getrennt die Auswirkungen von s auf E und auf X^{x_j}.
E: Für Partitionen, für die es ein s_i gibt mit $s_i = s$ und $s_j = 0$ für $j \neq i$, gilt $E = 1$. Für Partitionen,

in denen kein $s_i = s$ ist, gilt aber: Es gibt ein s_i mit $s_i \geq s_j \ \forall j$. Sei o.E. s_1 dieses maximale s_i. Wir erhalten

$$\begin{aligned} E = \frac{s!}{s_1! \cdots s_k!} &= \frac{s(s-1)\cdots(s_1+1)}{s_2! \cdots s_k!} \\ &\geq \frac{s \cdot s_2! \cdots s_k!}{s_2! \cdots s_k!} \\ &= s. \end{aligned}$$

Dabei ist die Abschätzung möglich, da alle Faktoren des Zählers $\geq$ den Faktoren des Nenners sind und die Anzahl der Faktoren gleich ist, denn $s - s_1 = s_2 + \cdots + s_k$. Nun wäre eigentlich ein Faktor zu wenig vorhanden, sodass s im Zähler verschwinden müsste, aber im Nenner ist mindestens ein $s_i \neq 0$ und somit kommt mindestens ein Faktor 1 vor, der ignoriert werden kann. Somit wird E beliebig groß für große s, sofern es in der Partition kein s_i gibt mit $s_i = s$.
X^{x_j}: Hier wirkt sich die Wahl der Partition nur auf die Verteilung der Exponenten der verschiedenen konstanten $e_i^{x_j}$ aus. Eine Veränderung der Partition von s um eine (von s unabhängige) konstante Summe verändert den Wert von X^{x_j} auch nur um einen konstanten Wert. Da E aber für Partitionen in mindestens zwei Summanden für große s beliebig groß wird, besteht eine optimale Partition (für große s) immer aus mehr als einem Summanden.
Insgesamt folgt also, dass durch entsprechend große Wahl von s der Wert von E im maximalen Term EX^{x_j} beliebig groß wird.
Damit können wir nun $q \geq N_0(x)$ wählen und die IV einsetzen:

$$\begin{aligned} q^{3x} \geq m^*(q) > \frac{1}{8}E^3 \Rightarrow q^x &> \frac{1}{2}E \\ \Leftrightarrow q^x X^x &> \frac{1}{2}EX^x \\ \Leftrightarrow (qX)^x &> \frac{1}{2}EX^x \end{aligned}$$

Nun können wir das Ergebnis von Schritt 2 verwenden:

$$m^*(qX) \leq (1+3ds)r^{3s}.$$

Dies ist also die Stelle, an der wir rekursiv das Ergebnis von Schritt 2 anwenden, denn in jedem Schritt der Induktion wird einmal das Ergebnis von Schritt 2 auf das vorherige Ergebnis angewandt. Wir erhalten damit die folgenden Gleichungen:

$$\frac{\ln(m^*(qX))}{\ln((qX)^x)} = \frac{\ln(m^*(qX))}{x\ln(qX)} = \frac{1}{x}\log_{(qX)}(m^*(qX)) \tag{3.6}$$

und

$$\begin{aligned} \frac{\ln(m^*(qX))}{\ln((qX)^x)} &< \frac{\ln((1+3ds)r^{3s})}{\ln(\frac{1}{2}EX^x)} \\ &< \frac{3s\ln(r)+\ln(1+3ds)}{\ln(\frac{1}{2}(\sum_i e_i^{x_j})^s/\binom{s+k-1}{k-1})} \\ &= \frac{3s\ln(r)+\ln(1+3ds)}{s\ln(\sum_i e_i^{x_j}) - \ln(2) - \ln(\binom{s+k-1}{k-1})} \end{aligned}$$

Wir können wie in Schritt 5 von (Satz 6) den Binomialkoeffizienten abschätzen durch

$$\binom{s+k-1}{k-1} \leq (s+k-1)^{k-1}$$

Damit können wir die Terme in obigem Bruch mit Hilfe der O-Notation vereinfacht darstellen, denn

$$\ln(1+3ds) \in O(\ln(s)) \text{ und}$$

$$\ln(2)+\ln\left(\binom{s+k-1}{k-1}\right) \leq \ln(2)+\ln((s+k-1)^{k-1}) \in O(\ln(s))$$

Wir erhalten zusammen mit (Gleichung (3.6)) und nach Multiplikation mit x:

$$\log_{(qX)}(m^*(qX)) \leq x\frac{3s\ln(r)+O(\ln(s))}{s\ln(\sum_i e_i^{x_j})-O(\ln(s))}$$

Für $s \to \infty$ erhalten wir damit

$$\lim_{s\to\infty}(\log_{(qX)}(m^*(qX))) \leq x\frac{3\ln(r)}{\ln(\sum_i e_i^{x_j})} = 3xy_j.$$

Nach Definition des Logarithmus bedeutet diese Gleichung einfach, dass für $s \to \infty$ gilt:

$$(qX)^{3xy_j} \geq m^*(qX)$$

Nun können wir endlich ausnutzen, dass wir x so gewählt haben, dass $xy_j < x'$, also $3xy_j < 3x'$, denn damit ergibt sich für $s \to \infty$

$$(qX)^{3x'} \geq m^*(qX)+\varepsilon \quad \text{für ein } \varepsilon > 0$$

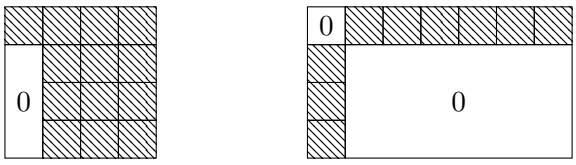
Dann lässt sich s aber auch ausreichend groß, aber fest wählen, sodass

$$(qX)^{3x'} \geq m^*(qX)$$

Da x' beliebig gewählt war, gibt es also für jedes $x' > x_{j+1}$ ein $N_0(x')$, sodass $m^*(N) \leq N^{3x'} \quad \forall N \geq N_0(x')$, was den IS beweist.

Aufgrund der Konvergenz der Folge gegen α erhalten wir $\omega \leq 3\alpha$, denn für jedes $\varepsilon > 0$ lässt sich ein Folgenglied $x_j < \alpha+\varepsilon$ finden. Dann gilt nach obigen Überlegungen aber auch, dass wir $x_j < x' < \alpha+\varepsilon$ wählen können, sodass für $N > N_0(x')$ gilt: $m^*(N) \leq N^{x'}$. □

Beispiel 3. Wir beginnen mit der Konstruktion aus (Beispiel 2). Allerdings setzen wir in der zweiten Matrix Position $(1,1)$ auf 0. Damit ergibt sich folgende Darstellung von I und J:



Diese Darstellung zeigt, dass wir diese partielle MM auch als zwei disjunkte, vollständige MMs interpretieren können: Der Eintrag $(1,1)$ der ersten Matrix wird mit der ersten Zeile der zweiten Matrix multipliziert, die übrigen Zeilen der ersten Matrix werden mit der ersten Spalte der zweiten Matrix multipliziert.
Die Anzahlen der Einsen der mit diesen MMs assoziierten Tensoren ergeben sich wieder aus (Gleichung (3.7)) zu

$$1\cdot(1+(x-1)(y-1)-1) = xy-x-y+1$$

und

$$\underbrace{(x\cdot 1+\cdots+x\cdot 1)}_{(y+1-1)-mal} = xy.$$

Außerdem haben wir aus (Beispiel 2) einen Algorithmus von Länge $xy+1$ und wir erhalten mit (Satz 7):

$$\omega \leq \alpha,$$

wobei α die Lösung von

$$(xy)^\alpha + (xy-x-y+1)^\alpha = xy+1 \quad \text{ist.}$$

Setzen wir hier $x = y = 4$, wie zuvor, so erhalten wir $\omega < 2,548$. Wir sehen also, dass (Satz 7) - zumindest in diesem Fall - auf Basis des gleichen Algorithmus eine deutlich bessere Schranke für ω liefert als (Satz 6).

3.9 Keine Matrizen-Multiplikation

Die nächste Idee geht noch einmal auf Strassen zurück. Die Idee ist, eine Trilinearform aufzustellen, die nicht direkt eine MM beschreibt, deren Struktur aber in gewisser Weise einer MM ähnelt. Strassen definiert dafür den Begriff des $\mathcal{C}$-Tensors, eine Art von Tensor, die auf ganz bestimmte Weise Ähnlichkeiten zu Tensoren der MM hat, siehe dafür [Str87-1] und [Str87-2]. Ohne auf Details einzugehen, lässt sich sagen, dass die geforderte Eigenschaft ist, dass der Tensor sich auf eine bestimmte Weise aus verschiedenen Tensoren der MM zusammensetzt. Strassen zeigt auch, dass sich aus Zerlegungen von $\mathcal{C}$-Tensoren Schranken für ω ableiten lassen.
Coppersmith und Winograd übernahmen dann die Idee, Tensoren zu verwenden, die keine MM mehr beschreiben, sondern nur zu gewissen Tensoren der MM ähnlich sind. Sie verwenden für ihren Algorithmus allerdings einen Tensor, der nicht einmal $\mathcal{C}$-Tensor ist.
Beiden Ansätzen gemeinsam ist folgendes Vorgehen: Die Trilinearform wird in mehrere kleinere Trilinearformen zerlegt. Dabei wird eine Zerlegung gewählt, deren kleinere Trilinearformen MMs entsprechen. Diese dürfen dabei zunächst gemeinsame Variablen enthalten. Strassen und CW verwenden dann verschiedene Ideen, um daraus eine gewisse Anzahl disjunkter MMs zu konstruieren und eine Schranke für ω zu folgern. Strassen erhält auf diese Weise $\omega < 2,4785$, Coppersmith und Winograd sogar $\omega < 2,376$.
Letztere Schranke ist dabei deutlich bedeutender und bekannter, da es sich hierbei um die bis 2010 ungeschlagene Schranke handelt. Aus diesem Grund und da Strassens Konstruktion aufgrund der starken Unterschiede zur Vorgehensweise von Coppersmith und Winograd nicht maßgeblich zum Verständnis dieser Vorgehensweise beiträgt, wird hier auf eine Präsentation von Strassens Konstruktion verzichtet[11]. Im nächsten Abschnitt folgt deshalb direkt eine genauere Erläuterung der Vorgehensweise von Coppersmith und Winograd.

3.10 Coppersmith und Winograd: Der Coppersmith-Winograd Algorithmus (1987)

Der Algorithmus von Coppersmith und Winograd (im Folgenden CW für die Personen und CW-Algorithmus für den Algorithmus), siehe [CW87], ist bis heute - in Form leichter Abwandlungen - der (asymptotisch) beste gefundene Algorithmus für die MM. Zwar wurden, wie bereits erwähnt, 2010 bessere Schranken für ω als die von CW gefunden, allerdings entstanden diese aus der Untersuchung höherer Tensorpotenzen desselben Algorithmus (d.h.: Es wurden Tensorpotenzen des Algorithmus als Grundalgorithmus verwendet und darauf weiter argumentiert). Wie im vorigen

[11] Hier bietet sich bei Interesse [CW87] an

Abschnitt erläutert, basiert das Verfahren auf einer Trilinearform, die sich in mehrere Trilinearformen zerlegen lässt, die wir als MMs auffassen können. Allerdings sind diese Trilinearformen nicht disjunkt, sodass insgesamt keine MM und auch keine Menge von disjunkten MMs vorliegt. Für diese Trilinearform wird dann ein approximativer Algorithmus beschrieben, der als Basis dient. Es wird anschließend auf einer Tensorpotenz (siehe Bemerkung 9) dieser Trilinearform argumentiert und dort werden geschickt Variablen nullgesetzt, bis eine disjunkte Menge von MMs entsteht. Für den border rank dieser Trilinearform erhalten wir eine Schranke. Außerdem lässt sich die Anzahl disjunkter MMs abschätzen, sodass wir mit (Satz 7) eine Schranke für ω folgern können.
CW wenden diese Vorgehensweise dabei auf eine einfachere und eine kompliziertere Trilinearform an - wir beschränken uns hier auf die einfachere. Zudem verfeinern sie für die Schranke $\omega \leq 2{,}376$ die Vorgehensweise noch mit einer Idee von Strassen. Da aber Williams dieselbe Idee für ihr allgemeineres Resultat verwendet, gehen wir erst im folgenden Abschnitt darauf ein.
Zunächst wollen wir folgende Vereinfachung einführen:

Bemerkung 9. *Seien T_1, T_2 zwei Trilinearformen. Dann definieren wir für diese das tensorielle Produkt $T_1 \otimes T_2$ und die direkte Summe $T_1 \oplus T_2$ anhand der Definitionen für Tensoren. Das heißt: Wir führen die Operationen auf den korrespondierenden Tensoren durch. Entsprechendes gilt für die Potenz $T_1^{\otimes s}$.*

Der Ausgangspunkt für die einfachere Konstruktion von CW ist die folgende Trilinearform:

$$T = \sum_{i=1}^{q} \left(x_0^{[0]} y_i^{[1]} z_i^{[1]} + x_i^{[1]} y_0^{[0]} z_i^{[1]} + x_i^{[1]} y_i^{[1]} z_0^{[0]} \right)$$

Die tiefgestellten Indizes sind dabei die tatsächlichen Indizes, die Einträge der gegebenen Vektoren auswählen. Die hochgestellten Indizes sind nur eine Hilfsdarstellung. Anhand dieser wird im Folgenden die Zerlegung in kleinere Trilinearformen vorgenommen. CW geben folgenden approximativen Algorithmus für T an:

$$\begin{aligned} T + O(\lambda) = \sum_{i=1}^{q} & \lambda^{-2} \left(x_0^{[0]} + \lambda x_i^{[1]} \right) \left(y_0^{[0]} + \lambda y_i^{[1]} \right) \left(z_0^{[0]} + \lambda z_i^{[1]} \right) \\ & - \lambda^{-3} \left(x_0^{[0]} + \lambda^2 \sum x_i^{[1]} \right) \left(y_0^{[0]} + \lambda^2 \sum y_i^{[1]} \right) \left(z_0^{[0]} + \lambda^2 \sum z_i^{[1]} \right) \\ & + (\lambda^{-3} - q\lambda^{-2})(x_0^{[0]})(y_0^{[0]})(z_0^{[0]}) \end{aligned}$$

Die Vorgehensweise, um aus diesem Algorithmus eine Schranke für ω zu folgern, gliedert sich in diese Schritte:

1. Bilden der Tensorpotenz $T^{\otimes 3s}$. Die Indizes werden damit zu Multi-Indizes, die die Auswahl der Variablen in den verschiedenen Schachtelungstiefen angeben. Zudem Auswählen enthaltener (kleinerer) Trilinearformen, die für ein festes s der Berechnung von MMs einer festen Größe entsprechen
2. Nullsetzen von Variablen aus $T^{\otimes 3s}$, sodass die kleineren Trilinearformen paarweise keine gemeinsame Variablen verwenden
3. Abschätzen der Anzahl übriger kleinerer Trilinearformen
4. Folgern einer Schranke für ω mit (Satz 7)

Um die Argumentation für die Anzahl übriger Trilinearformen in Schritt 3 zu vereinfachen bzw. überhaupt zu ermöglichen, benötigen wir den Begriff der Salem-Spencer-Menge und einen zugehörigen Satz von Salem und Spencer. Dazu benötigen wir zunächst den Begriff der arithmetischen Progression mit drei Termen.

Definition 20. *Eine arithmetische Progression mit drei Termen ist eine Folge $a<b<c$ mit $a+c=2b$, d.h. die Abstände sind gleich: $b-a=c-b$.*

Satz 8. [12] *Sei $\varepsilon>0$. Dann gibt es $M_\varepsilon \simeq 2^{c/\varepsilon^2}$, sodass für alle $M>M_\varepsilon$ eine Menge $B=\{b_1,\dots,b_{|B|}\}$ existiert mit den Eigenschaften:*

1. $|B|>M^{1-\varepsilon}$
2. $0<b_1<\cdots<b_{|B|}<\frac{M}{2}$
3. *B enthält keine arithmetische Progression mit drei Termen, also:*

$$\forall b_i,b_j,b_k\in B\colon\ b_i+b_j=2b_k \Leftrightarrow b_i=b_j=b_k$$

Definition 21. *Wir nennen eine Menge wie B in (Satz 8) Salem-Spencer-Menge.*

Beachte, dass eine solche Menge B auch modulo M keine arithmetische Progression mit drei Termen enthält, da die Elemente der Menge $<\frac{M}{2}$ sind und somit für alle $b_i,b_j,b_k\in B$ gilt: $b_i+b_j<M$, $2b_k<M$.
Wir beginnen nun mit der eigentlichen Konstruktion.

Blöcke: Zunächst wollen wir den Begriff des Blockes einführen. Mit Blöcken meinen wir die hochgestellten Indizes in der obigen Trilinearform. Entsprechend sprechen wir von Block-Tripeln, wenn wir die Blöcke an den drei Variablen eines Produktes $x_iy_jz_k$ meinen. Wir können dann die Zerlegung in kleinere Trilinearformen anhand der Block-Tripel vornehmen und erhalten so für jedes Block-Tripel (I,J,K) eine Trilinearform, die genau über die Produkte summiert, für die der x-Block I, der y-Block J und der z-Block K ist.

Schritt 1: Zunächst wählen wir ein festes $\varepsilon>0$. Wir bilden in diesem Schritt die $3s$-te Tensorpotenz $T^{\otimes 3s}$ der Trilinearform T. Bevor wir diese genauer betrachten, wählen wir $M=2\binom{2s}{s}+1$ und dann s derartig, dass $M>M_\varepsilon$, damit wir im Folgenden eine Salem-Spencer-Menge für dieses M bilden können.
Wir untersuchen nun die $3s$-te Tensorpotenz. Dazu veranschaulichen wir uns, wie die Potenzen sich verhalten, indem wir $T^{\otimes 2}$ vollständig aufschreiben. Beachte, dass auch die Blöcke zu Multi-Indizes werden, in denen jeder Eintrag durch den jeweiligen Eintrag des tiefgestellten Indizes bestimmt ist.

$$\begin{aligned}T^{\otimes 2}=\sum_{i_1=1}^{q}\sum_{i_2=1}^{q}\Big(&x_{0,0}^{[0,0]}y_{i_1,i_2}^{[1,1]}z_{i_1,i_2}^{[1,1]}+x_{0,i_2}^{[0,1]}y_{i_1,0}^{[1,0]}z_{i_1,i_2}^{[1,1]}+x_{0,i_2}^{[0,1]}y_{i_1,i_2}^{[1,1]}z_{i_1,0}^{[1,0]}\\&+x_{i_1,0}^{[1,0]}y_{0,i_2}^{[0,1]}z_{i_1,i_2}^{[1,1]}+x_{i_1,i_2}^{[1,1]}y_{0,0}^{[0,0]}z_{i_1,i_2}^{[1,1]}+x_{i_1,i_2}^{[1,1]}y_{0,i_2}^{[0,1]}z_{i_1,0}^{[1,0]}\\&+x_{i_1,0}^{[1,0]}y_{i_1,i_2}^{[1,1]}z_{0,i_2}^{[0,1]}+x_{i_1,i_2}^{[1,1]}y_{i_1,0}^{[1,0]}z_{0,i_2}^{[0,1]}+x_{i_1,i_2}^{[1,1]}y_{i_1,i_2}^{[1,1]}z_{0,0}^{[0,0]}\Big)\end{aligned}$$

Für unsere Argumentation überlegen wir uns nun, wie die Terme in der dritten Tensorpotenz aussehen. Dort erhalten wir Terme von der Form

$$x_{0,i_2,i_3}^{[0,1,1]}y_{i_1,0,i_3}^{[1,0,1]}z_{i_1,i_2,0}^{[1,1,0]}.$$

wobei die Blöcke und die tiefgestellten Indizes Multi-Indizes mit je drei Einträgen sind. Wir können $T^{\otimes 3}$ ebenso wie T als Summe von Trilinearformen T_{IJK} schreiben, wobei nun I,J,K Multi-Indizes sind. In diesen treten wie zuvor für T jeweils genau die Summanden mit I als x-, J als y- und K als z-Block auf. Dabei ist leicht zu sehen, dass Trilinearformen T_{IJK} zu Block-Tripeln (I,J,K), in

[12] Hier handelt es sich um die von CW verwendete Version, nicht um die Originalversion des Satzes. Allerdings folgt diese aus der ursprünglichen Aussage

denen jeder Block-Multi-Index genau eine Null aufweist, genau dem Problem (q,q,q) entsprechen. Ein Beispiel für eine solche Trilinearform ist

$$\sum_{i_1}\sum_{i_2}\sum_{i_3} x_{0,i_2,i_3} y_{i_1,0,i_3} z_{i_1,i_2,0}.$$

Wir können dabei, da jeweils einer der Indizes für x, y und z ungenutzt bleibt, jeweils zwei Indizes zusammenfassen: Für x fassen wir $(0,i_2)$ zu i_2 zusammen, für y $(i_1,0)$ zu i_1 und für z $(i_2,0)$ zu i_2. Anschließend tauschen wir noch die Reihenfolge der beiden Indizes an der Variable y und erhalten so

$$\sum_{i_1}\sum_{i_2}\sum_{i_3} x_{i_2,i_3} y_{i_3,i_1} z_{i_1,i_2},$$

was genau dem Problem (q,q,q) entspricht.
Beachte, dass die Nullen in den drei Blöcken an verschiedenen Stellen stehen müssen, da in der Basis-Trilinearform für jeden Summanden nur einer der drei Blöcke Null ist.
Diese Überlegungen lassen sich direkt auf die $3s$-te Tensorpotenz übertragen. Die Indizes sind dann Multi-Indizes mit $3s$ Einträgen und wir wählen die Trilinearformen T_{IJK} derjenigen Tripel (I,J,K) aus, für die I, J und K jeweils genau s Nullen enthalten. Analog zur abschließenden Bemerkung zur Tensorpotenz $T^{\otimes 3}$ ergibt sich hier, dass die Nullpositionen innerhalb eines Block-Tripels paarweise disjunkt sind, es tauchen also innerhalb eines Block-Tripels nie Nullen an der gleichen Position in verschiedenen Block-Multi-Indizes auf. Die Berechnung einer dieser Trilinearformen entspricht dann jeweils der Lösung des Problems (q^s,q^s,q^s).
Im Folgenden sprechen wir sinnvollerweise davon, einen Block nullzusetzen, wenn wir alle Variablen nullsetzen, die diesen Block aufweisen. Wir setzen alle Blöcke in $T^{\otimes 3s}$ Null, die nicht genau s Nullen enthalten, sodass die Trilinearformen zu allen übrigen Block-Tripeln dem Problem (q^s,q^s,q^s) entsprechen.

Schritt 2: Wir wollen nun Blöcke nullsetzen, sodass alle übrigen Trilinearformen T_{IJK} der in Schritt 1 beschriebenen Form disjunkt sind, damit wir diese als disjunkte MMs auffassen können und mit (Satz 7) eine Schranke für ω erhalten.
In der Tensorpotenz $T^{\otimes 3s}$ direkt Überschneidungen zu beseitigen und die Anzahl übriger Block-Tripel abzuschätzen wäre schwierig und würde nicht unbedingt zu sinnvollen Ergebnissen führen. Deshalb werden wir die Blöcke auf geschickte Art hashen, um nur noch in kleineren Mengen von Block-Tripeln nach Überschneidungen suchen zu müssen und außerdem mit Hilfe stochastischer Methoden die Anzahl übriger Block-Tripel abschätzen zu können.
Wir gehen dazu wie folgt vor: Wir bilden eine Salem-Spencer-Menge M_S für das M von oben. Dann bilden wir mit geschickt gewählten Hash-Funktionen die Block-Multi-Indizes auf Zahlen modulo M ab. Anschließend setzen wir all diejenigen Blöcke Null, deren Hash-Werte nicht in M_S liegen. Nach Wahl der Hash-Funktionen bilden die Hash-Werte der Block-Tripel immer eine arithmetische Progression mit drei Termen und da sie für die Tripel, die wir hier nicht nullsetzen, in einer Salem-Spencer-Menge liegen, folgt daraus direkt, dass die drei Blöcke in einem Tripel jeweils gleich sind. Damit ist jedem Block-Tripel eindeutig ein einzelner Hash-Wert zugeordnet. Wir bilden dann für jeden möglichen Hash-Wert einzeln - also für jedes Element der Salem-Spencer-Menge - die Menge der Block-Tripel, denen dieser Hash-Wert zugeordnet ist, und müssen nach Überschneidungen nur innerhalb der einzelnen Mengen suchen. Dieses Vorgehen soll nun im Detail beschrieben werden.
Wir benennen die Einträge der Block-Multi-Indizes mit $I = (I_j)_{j\in\{1,\dots,3s\}}$, $J = (J_j)_{j\in\{1,\dots,3s\}}$, $K = (K_j)_{j\in\{1,\dots,3s\}}$. Die Hash-Funktionen für die x-, die y- und die z-Blöcke geben wir in Abhängigkeit

von Zufallsvariablen $w_1, \ldots, w_{3s}$ an, wobei $0 \leq w_j < M \ \forall j \in \{0, \ldots, 3s\}$:

$$b_X(I) \equiv \sum_{j=1}^{3s} I_j w_j \mod M$$

$$b_Y(J) \equiv w_0 + \sum_{j=1}^{3s} J_j w_j \mod M$$

$$b_Z(K) \equiv \left(w_0 + \sum_{j=1}^{3s} (2 - K_j) w_j \right) /2 \mod M$$

Die Division durch 2 ist dabei wohldefiniert, denn M ist ungerade und damit ist die Multiplikation mit 2 bijektiv und die Division durch 2 ihre Umkehrfunktion. Im dritten Schritt können wir aufgrund der Zufallsvariablen w_i Methoden der Stochastik für die Argumentation verwenden.
Da für einen einzelnen Index j (also in jeder Verschachtelungstiefe der Tensorpotenz) jeweils genau einer der drei Indizes I_j, J_j, K_j Null ist und die anderen beiden Eins, gilt $I_j + J_j + K_j = 2$, also

$$2 - K_j = I_j + J_j.$$

Somit gilt für ein beliebiges Block-Tripel (I, J, K) der Trilinearform:

$$b_X(I) + b_Y(J) = 2 b_Z(K),$$

es handelt sich also um eine arithmetische Progression mit drei Termen. Wir setzen also nun alle diejenigen Blöcke Null, deren Multi-Index nicht in die Salem-Spencer-Menge M_S gehasht wird. Nun gilt also für alle übrigen Block-Tripel (I, J, K): $b_X(I), b_Y(J), b_Z(K) \in M_S$ und damit, wie bereits einleitend erwähnt:

$$b_X(I) = b_Y(J) = b_Z(K),$$

da es sich um eine arithmetische Progression mit drei Termen in einer Salem-Spencer-Menge handelt. Damit ist jedem Block-Tripel jetzt nur noch ein einzelner Hash-Wert zugeordnet (die Hash-Werte der drei Blöcke in diesem Tripel stimmen überein). Wir betrachten dann für jeden potentiellen Hash-Wert, also für jedes Element der Menge M_S, die Menge von Block-Tripeln, denen dieser Hash-Wert zugeordnet ist. Die Trilinearformen zu Block-Tripeln, die in verschiedenen Mengen liegen, können nie gleiche Variablen enthalten, da sie verschiedene Hash-Werte für alle drei Blöcke haben. Es können nun also für jede dieser Mengen einzeln Blöcke nullgesetzt werden, sodass nur noch Block-Tripel mit disjunkten zugehörigen Trilinearformen in den einzelnen Mengen - und damit auch insgesamt - übrig bleiben.
Eine Abschätzung für die Anzahl der nach diesen verschiedenen Schritten des Nullsetzens übrigen Block-Tripel, also auch Trilinearformen, ermitteln wir im letzten Schritt der Konstruktion.

Schritt 3: Auf der Menge der Block-Tripel wurden in Schritt 1 und 2 folgende Einschränkungen vorgenommen:

1. I, J und K enthalten nie an denselben Stellen Nullen (ergibt sich aus der Basis-Trilinearform).
2. I, J und K enthalten jeweils genau s Nullen (damit die zugehörigen Trilinearformen MMs (q^s, q^s, q^s) entsprechen).
3. $b_X(I), b_Y(J), b_Z(K) \in M_S$ (daraus folgt $b_X(I) = b_Y(J) = b_Z(K)$)
4. Für jedes $b \in M_S$ Bestimmen der Menge von Block-Tripeln, denen dieses b als Hash-Wert zugeordnet ist (siehe Schritt 2) und in jeder dieser Mengen Nullsetzen von Blöcken, sodass sich die zugehörigen Trilinearformen paarweise keine Variablen teilen.

Wir ermitteln nun für eine zufällige Belegung der Variablen w_j den Erwartungswert für die Anzahl von übrigen Tripeln in jeder der Mengen aus (4.). Da es mindestens eine mögliche Belegung der Variablen w_j geben muss, für die dieser Erwartungswert erreicht (oder übertroffen) wird, ist dieser eine untere Schranke für die Anzahl disjunkter MMs bei optimaler Wahl der w_j (jedes übrig gebliebene Block-Tripel entspricht einer MM (q^s, q^s, q^s), wobei alle diese MMs disjunkt sind).
Wir beginnen mit der Anzahl von Tripeln, die die ersten beiden Bedingungen erfüllen, also die Anzahl der Elemente in

$$L := \left\{ (I,J,K) \;\middle|\; \begin{array}{c} I,J,K \in \{0,1\}^{3s}, \text{ wobei sie jeweils genau } s \text{ Nullen enthalten} \\ \text{und die Nullpositionen zweier Multi-Indizes paarweise disjunkt sind} \end{array} \right\}$$

Diese ist

$$|L| = \binom{3s}{s,s,s} = \frac{(3s)!}{s!s!s!},$$

wobei die Argumentation ähnlich zu der in Schritt 2 von (Satz 6) verläuft. Es werden nacheinander die Positionen der s Nullen in I, der s Nullen in J und der s Nullen in K ausgewählt. Da aber alle diese Positionen Nullen sind, sich also nicht unterscheiden, können sie jeweils innerhalb eines Multi-Indizes permutiert werden.
Wir bestimmen nun den Erwartungswert für die Anzahl von Tripeln in der Menge für ein festes $b \in M_S$. Der Ausgangspunkt hierfür ist obige Menge L. Nach (3.) bleiben diejenigen Tripel übrig, deren Block-Multi-Indizes auf den Wert b gehasht werden. Wegen $b_X(I) + b_Y(J) = 2b_Z(K)$ genügt es, wenn $b_X(I) = b_Y(J) = b$. Es folgt dann direkt $2b = b_X(I) + b_Y(J) = 2b_Z(K)$, also $b_X(I) = b_Y(J) = b_Z(K) = b$.
Wir betrachten nun also für unser festes b die Menge der Block-Tripel, die nach Schritt 3 übrig bleiben:

$$L' := L \backslash \{(I,J,K) \mid I,J,K \in \{0,1\}^{3s}, b_X(I) \neq b \text{ oder } b_Y(J) \neq b\}$$

Sei der Erwartungswert für die Anzahl der Elemente dieser Menge bei zufälliger Wahl der w_j bezeichnet mit $E(L')$.
Es lässt sich zeigen, dass für eine zufällige Wahl der w_j die beiden Ereignisse $b_X(I) = b$ und $b_Y(J) = b$ stochastisch unabhängig sind und dass die Wahrscheinlichkeit für die einzelnen Werte a mit $0 \leq a < M$ gleich ist.[13] Damit erhalten wir als Wahrscheinlichkeit für diese Ereignisse jeweils M^{-1} und als Wahrscheinlichkeit für $b_X(I) = b_Y(J) = b$ aufgrund der Unabhängigkeit $M^{-1} \cdot M^{-1} = M^{-2}$.
Der Erwartungswert für die Elemente in L' ist die Summe der Erwartungswerte für jedes einzelne Element (dafür müssen die Ereignisse nicht stochastisch unabhängig sein), also erhalten wir

$$E(|L'|) = \binom{3s}{s,s,s} M^{-2}.$$

Es fehlt nun noch der Erwartungswert für die Anzahl von Elementen in der Menge für b, nachdem in (4.) Blöcke nullgesetzt wurden, um disjunkte MMs zu erhalten. Dazu überlegen wir, wie wir beim Nullsetzen vorgehen wollen. Es bietet sich an, nacheinander die einzelnen Tripel durchzugehen und jeweils zu prüfen, ob das aktuelle Tripel gemeinsame Blöcke mit einem der vorherigen Tripel aufweist. Ist dies der Fall, so wird nach Möglichkeit einer der Blöcke nullgesetzt, der in keinem vorherigen Tripel vorkommt. Stimmt jeder der Blöcke mit einem Block eines vorherigen Tripels überein, so werden alle Tripel eliminiert, die den Block enthalten, den wir nullsetzen. Wir schätzen nun die Anzahl der eliminierten Tripel ab. Dazu betrachten wir alle ungeordneten Paare von Tripeln aus L', also alle Teilmengen der Größe zwei:

$$L_{\text{pair}} = \{\{(I,J,K),(I',J',K')\} \mid (I,J,K),(I',J',K') \in L'\}$$

[13] Stothers erläutert dies relativ ausführlich in [Sto10]

Wir können diese statt der Menge von Tripeln untersuchen, denn:
Tritt ein Block in zwei der Tripel auf, zum Beispiel der Block K in den Tripeln (I,J,K) und (I',J',K), dann werden wir einen der beteiligten Blöcke nullsetzen. Sei dies beispielhaft der Block J'. Es werden dadurch neben (I',J',K) auch alle weiteren Tripel eliminiert, die diesen Block ebenfalls enthalten. Sei die Anzahl dieser Tripel U. Dann werden mindestens alle Paare aus L_{pair} eliminiert, die genau aus zweien dieser Tripel bestehen, also $\binom{U}{2}$. Dazu kommt noch das Paar, das wir eingangs betrachtet haben (in diesem kann J nicht übereinstimmen, da schon K übereinstimmt und zwei der Blöcke ein Block-Tripel bereits eindeutig bestimmen). Infolgedessen werden mindestens

$$\binom{U}{2}+1 \geq U$$

Paare eliminiert, also mindestens so viele Paare aus L_{pair} wie Tripel aus L'.
Wir untersuchen nun die Mengen der ungeordneten Tripel-Paare, die jeweils mindestens einen gemeinsamen Block haben, also

$$L^K_{\text{pair}} = \{\{(I,J,K),(I',J',K)\} \mid (I,J,K),(I',J',K) \in L'\} \subseteq L_{\text{pair}}$$

und analog für I und J. Diese enthalten insgesamt alle Paare von Tripeln, die gemeinsame Blöcke haben, liefern also die Anzahl von Tripel-Paaren, die eliminiert werden.
Für die Erwartungswerte der Anzahlen von Elementen in diesen Mengen gilt offenbar $E(|L^I_{\text{pair}}|) = E(|L^J_{\text{pair}}|) = E(|L^K_{\text{pair}}|)$, da die Mengen völlig symmetrisch gebildet werden. Wir untersuchen deshalb nur L^K_{pair}. Um $E(|L^K_{\text{pair}}|)$ zu erhalten, bestimmen wir zunächst die Anzahl der geordneten Paare von Tripeln nach (2.), in denen das zweite Tripel den gleichen K-Block aufweist wie das erste Tripel. Dies ist die Anzahl aller Tripel, die nach (2.) übrig sind mal der Anzahl aller weiteren solcher Tripel, die dasselbe K aufweisen. Die Anzahl der Block-Tripel mit festem K in L ist $\binom{2s}{s,s} = \binom{2s}{s}$. Das liegt daran, dass bereits durch K s Nullpositionen feststehen und somit noch die s Nullpositionen in I (oder in J) ausgewählt werden können, um das Block-Tripel eindeutig festzulegen. Eines von diesen Tripeln ist genau das erste Tripel des Paares, weshalb 1 von der Anzahl subtrahiert wird. Um daraus die Anzahl ungeordneter Paare zu erhalten, müssen wir nur noch durch zwei teilen, da es jeweils zwei Möglichkeiten für die Reihenfolge innerhalb eines Paares gibt. Die Anzahl ungeordneter Paare mit gleichem K vor (3.) ist somit

$$\frac{1}{2}\binom{3s}{s,s,s}\left(\binom{2s}{s}-1\right).$$

Da in L^K_{pair} aber nur noch Paare von Tripeln aus L' (also nach (3.)) enthalten sind, müssen wir uns noch überlegen, wie groß der Erwartungswert für die Anzahl von Paaren ist, die zudem die Bedingung

$$b = b_Z(K) = b_Y(J) = b_Y(J')$$

erfüllen (wieder ergibt sich der jeweils dritte Hash-Wert - hier $b_X(I)$ bzw. $b_X(I')$ - bereits aus den beiden anderen Hashwerten). Da sich auch für diese drei Gleichheiten zeigen lässt, dass die Ereignisse stochastisch unabhängig sind, erhalten wir einen Erwartungswert von M^{-3} pro Paar von Tripeln und damit insgesamt (durch Summieren der Erwartungswerte für alle Paare)

$$E(|L^I_{\text{pair}}|) = E(|L^J_{\text{pair}}|) = E(|L^K_{\text{pair}}|) = \frac{1}{2}\binom{3s}{s,s,s}\left(\binom{2s}{s}-1\right)M^{-3}.$$

Der Erwartungswert für die Anzahl eliminierter Tripel von (3.) zu (4.) ist damit kleinergleich

$$3E(|L^K_{\text{pair}}|) = \frac{3}{2}\binom{3s}{s,s,s}\left(\binom{2s}{s}-1\right)M^{-3}.$$

Sei L'' die Anzahl der Tripel, die nach (4.) in der Menge von Tripeln für ein einzelnes b übrig sind, dann erhalten wir als Erwartungswert für die Anzahl von Elementen in L'':

$$\begin{aligned}
E(|L''|) &\geq E(|L'|) - 3E(|L^K_{\text{pair}}|) \\
&= \binom{3s}{s,s,s} M^{-2} - \frac{3}{2}\binom{3s}{s,s,s}\left(\binom{2s}{s} - 1\right) M^{-3} \\
&\geq \binom{3s}{s,s,s} M^{-2} - \frac{3}{2}\binom{3s}{s,s,s}\left(\binom{2s}{s} + 1\right)\left(2\binom{2s}{s} + 1\right)^{-1} M^{-2} \\
&= \binom{3s}{s,s,s} M^{-2} - \frac{3}{4}\binom{3s}{s,s,s} M^{-2} \\
&= \frac{1}{4}\binom{3s}{s,s,s} M^{-2}
\end{aligned}$$

Der Erwartungswert für die Anzahl aller übrigen Tripel nach (4.) ist dann die Summe der Erwartungswerte für die Anzahlen von Elementen in allen diesen Mengen zusammen (also summiert für alle b):

$$|M_S| \cdot E(|L''|).$$

Da der Erwartungswert nur zustande kommen kann, wenn er mindestens von einer Belegung auch erreicht oder überschritten wird, gibt es also eine Belegung der w_j, sodass wir mindestens $|M_S| \cdot E(|L''|)$ Tripel erhalten, was nach Konstruktion derselben Anzahl disjunkter MMs der Form (q^s, q^s, q^s) entspricht. Mit (Satz 7) erhalten CW durch Maximierung (für $q = 8$)

$$\omega \leq \log_8(4000/27) < 2{,}40364.$$

Bemerkung 10. *Wie Eingangs erwähnt liefern Coppersmith und Winograd noch eine kompliziertere Variante, mit der sie $\omega < 2{,}388$ erhalten. Zudem geben sie noch eine dritte Variante an, mit der sie schließlich die bekannte Schranke $\omega < 2{,}376$ erhalten. Da die Vorgehensweise aber dieselbe ist, mit der Williams später ihr allgemeineres Resultat erhielt, gehen wir nicht näher auf die Konstruktion von Coppersmith und Winograd ein.*

In Vorbereitung auf die Resultate von Williams kommen wir nun zum Begriff des Values einer Trilinearform. Dieser wurde von Strassen eingeführt und auf ihm basiert das Vorgehen von CW für die dritte Konstruktion ebenso wie die Vorgehensweise von Williams.

3.11 Der Value einer Trilinearform in Bezug auf die Matrizen-Multiplikation

Zuerst eingeführt wurde der Begriff des Values einer Trilinearform in Bezug auf die MM von Strassen in [Str87-1]. Diese Publikation beschäftigt sich auf sehr mathematische Weise mit Tensoren und der MM. Coppersmith und Winograd übernahmen die Idee des Values und beschrieben ihn durch seine Eigenschaften, statt ihn klar zu definieren, was ihnen den relativ komplizierten mathematischen Unterbau ersparte. Auf die gleiche Weise nutzte auch Williams 2010 den Begriff des Values, um eine bestimmte Art von Algorithmen, zu denen auch der CW-Algorithmus gehört, zu analysieren. Aus diesem Grund beschränken wir uns hier darauf, den Value nur vage zu definieren und anschließend durch seine Eigenschaften zu beschreiben.

Definition 22. *Der Value[14] einer Trilinearform T in Bezug auf die Matrizen-Multiplikation, kurz Value von T, trifft eine Aussage über Ähnlichkeiten der Trilinearform zu gewissen MMs. Durch Verbindung mit Schönhages τ-Theorem lassen sich aus dem Value Schranken für ω ableiten. Der Value ist eine (monoton wachsende) Funktion in τ, wir schreiben: $V_\tau(T)$.*

Nun beschreiben wir, wie sich der Value einer Trilinearform T bestimmen lässt. Dies geschieht mit folgenden Schranken: Kann die Berechnung mehrerer disjunkter MMs, $\bigoplus_{i=1}^{q} \langle m_i, n_i, p_i \rangle$, für ein s in die s-te Tensorpotenz von T, also $T^{\otimes s}$, eingebettet werden, dann erhalten wir die Schranke

$$V_\tau(T) \geq \left(\sum_{i=1}^{q} (k_i m_i n_i)^\tau \right)^{1/s}.$$

Damit wächst $V_\tau(T)$ monoton für wachsende τ und somit ergeben obere Schranken für den Value auch obere Schranken für τ. Darüber hinaus wird die Idee der Symmetrisierung auf den Value übertragen, was folgende Schranke ergibt:

$$(V_\tau(T \otimes \pi T \otimes \pi^2 T))^{1/3} \geq (V_\tau(T) V_\tau(\pi T) V_\tau(\pi^2 T))^{1/3}$$

Es lässt sich zudem zeigen, dass sich der Value von Trilinearformen supermultiplikativ und superadditiv verhält, also für zwei Trilinearformen T_1, T_2 gilt:

$$V_\tau(T_1 \otimes T_2) \geq V_\tau(T_1) V_\tau(T_2)$$

$$V_\tau(T_1 \oplus T_2) \geq V_\tau(T_1) + V_\tau(T_2)$$

Das entscheidende Resultat, um das Konzept des Values zu nutzen ist folgende Übetragung von Schönhages τ-Theorem auf den Value:

Satz 9. *Sei $T = \bigoplus_{i=1}^{q} T_i$ eine Trilinearform. Sei der Rang von T gegeben als $r > q$. Dann folgt für τ gegeben durch*

$$\sum_{i=1}^{q} V_\tau(T_i) = r$$

die Schranke $\omega \leq 3\tau$.

Dieses Resultat ermöglicht es, aus Schranken für den Rang gewisser Trilinearformen zusammen mit Aussagen über ihren Value, nicht nur Schranken für τ, sondern auch für ω, zu folgern. Es kann also der Value $V_\tau(T)$ für Trilinearformen T untersucht werden, anstatt direkt MMs in diese einzubetten.

3.12 Williams: (p, P)-uniforme Konstruktionen und daraus folgende Schranken für ω

Nachdem im Jahr 1987 CW ihren Algorithmus und die Schranke $\omega < 2{,}376$ veröffentlichten, blieb diese lange Zeit ungeschlagen. Es war jedoch bereits von CW selbst in ihrer Publikation darauf hingewiesen worden, dass höhere Tensorpotenzen ihrer Trilinearform untersucht werden könnten, da sie es für möglich hielten, dass diese bessere Resultate liefern. Mit der Untersuchung höherer Tensorpotenzen ist hierbei nicht wie üblich gemeint, dass die s-te Tensorpotenz gebildet und auf

[14] Der Begriff wurde bewusst nicht übersetzt, da der Begriff „Wert“ mit seiner gewöhnlichen Bedeutung relativ häufig vorkommt und dies zu Verwirrung führen könnte

gewisse Art untersucht wird und anschließend Resultate für $s \to \infty$ betrachtet werden. Stattdessen ist gemeint, dass als Basis eine Tensorpotenz der ursprünglichen Trilinearform gebildet wird, um diese geschickt in kleinere Trilinearformen zu zerlegen (wie in der Konstruktion von CW) und anschließend mit ähnlichen Betrachtungen wie denen für die einfache Konstruktion eine Schranke für ω abzuleiten. Bereits CW verwendeten für ihre beste Schranke die zweite Tensorpotenz einer ihrer Trilinearformen, allerdings stellte sich heraus, dass die dritte Tensorpotenz keine bessere Schranke liefert. Zudem nimmt die Komplexität in den höheren Tensorpotenzen stark zu. Vermutlich ist dies der Grund dafür, dass erst 2010 durch Untersuchen der vierten Tensorpotenz von Stothers in [Sto10] eine neue Schranke gefunden wurde. Zudem erschien im gleichen Jahr in [Wi10] auch das allgemeinere Resultat von Williams, aus dem sich neben diesem Resultat auch eine noch bessere Schranke aus der achten Tensorpotenz ableiten lässt. Doch warum ermöglichte das Resultat so schnell die Untersuchung so hoher Tensorpotenzen? Die Idee von Williams ist, die Argumentation von CW im allgemeinen Fall durchzuführen und auf diese Weise gewisse Gleichungssysteme zu bilden, die es ermöglichen, Schranken, die aus Konstruktionen wie der von CW folgen, direkt mit Computern berechnen zu lassen, anstatt diese manuell Schritt für Schritt herzuleiten. Da sowohl die Beweise als auch die Resultate sehr technisch sind, wird zwar schrittweise erläutert wie Williams dabei argumentiert, jedoch nicht auf die Korrektheit einzelner Gleichungen im Detail eingegangen. Auch wird darauf verzichtet, die unübersichtlichen Algorithmen, die sich ergeben, vollständig aufzuschreiben. Bei Interesse bietet sich deshalb die Lektüre der Arbeit [Wi10] von Williams an.
Zunächst benötigen wir den Begriff (p, P)-uniformer Trilinearformen und Konstruktionen, da sich die Resultate auf Trilinearformen bzw. Konstruktionen dieser besonderen Form beschränken.

Definition 23. *Sei p: $\{1, \dots, n\} \to \{1, \dots, N\}$ eine Funktion. Sei $P \in \{1, \dots, N\}$. Eine Trilinearform heißt (p, P)-uniform, wenn für den mit ihr assoziierten Tensor t gilt:*

$$\forall t_{ijk} \neq 0 : p(i) + p(j) + p(k) = P$$

Sei dafür n so gewählt, dass p für alle auftretenden Indizes von t definiert ist und sei $N \in \mathbb{N}$ beliebig. Auch eine Konstruktion, die eine solche Trilinearform nutzt, heißt (p, P)-uniform.

Die Funktion p lässt sich dabei als Zuweisung von Blöcken interpretieren und damit gilt für eine (p, P)-uniforme Trilinearform, dass die Summe der Blöcke eines jeden Summanden in der Trilinearform P ist. Insofern ist auch die Trilinearform aus der einfacheren Konstruktion von CW (p, P)-uniform, wobei hierbei p die angegebene Zuordnung von Blöcken und $P = 2$ ist.

Bemerkung 11. *Wir können die Trilinearform damit auch in diesem allgemeineren Fall als Summe von Trilinearformen für einzelne Block-Tripel (I, J, K) schreiben. Beispielsweise wäre dann*

$$T_{I,J,K} = \sum_{i,j,k : p(i)=I, p(j)=J, p(k)=K} x_i y_j z_k.$$

Zu einer solchen Trilinearform würden also alle Summanden der ursprünglichen Trilinearform gehören, deren Indizes auf die Blocknummern I, J und K abgebildet werden. Die ursprüngliche Trilinearform ist damit die Summe der Trilinearfomen $T_{I,J,K}$ über alle Tripel von Blöcken.

Williams beschreibt nun algorithmisch eine Vorgehensweise zur Analyse der q-ten Tensorpotenz (p, P)-uniformer Trilinearformen und daraus folgender Schranken für ω. Diese gliedert sich grundlegend in zwei Teile: Zuerst wird der Value aller Trilinearformen zu Block-Tripeln bestimmt, die sich in $T^{\otimes q}$ finden. Dafür müssen die Funktion p und die Blöcke geeignet auf Tensorpotenzen übertragen werden. Anschließend wird dann aufgrund der Values dieser Teile eine Abschätzung für den Value von Tensorpotenzen von $T^{\otimes q}$ gegeben. Damit ergibt sich dann mit (Satz 9) eine Schranke für ω. Wir gliedern unsere Beschreibung hier wie Williams und beginnen deshalb mit dem zweiten Schritt.

3.12.1 Folgern von Schranken für ω aus (p, P)-uniformen Konstruktionen

Folgern einer Schranke für ω in Abhängigkeit gewisser Variablen $\overline{a}_{IJK}$ und a_{IJK}:
Gegeben sei eine (p, P)-uniforme Trilinearform T von Rang $\leq r$ und außerdem für alle Block-Tripel (I, J, K) der Tensorpotenz $T^{\otimes q}$ der Value V_{IJK} der zugehörigen Trilinearform als monoton steigende Funktion in τ. Auf die folgende Art erhalten wir einen Algorithmus zur Folgerung einer Schranke für ω aus den Eigenschaften einer Tensorpotenz $T^{\otimes q}$ für beliebige $q \geq 2$ mit Hilfe der gegebenen Voraussetzungen.
Wir bilden zunächst die zu untersuchende Tensorpotenz $T^{\otimes q}$, deren Rang $\leq r^q$ ist. Die Variablen in dieser Tensorpotenz haben Multi-Indizes von Länge q. Da es sich um eine (p, P)-uniforme Trilinearform handelt, gilt für alle Summanden $x_I y_J z_K$[15] in $T^{\otimes q}$:

$$p(I_i) + p(J_i) + p(K_i) = P \ \forall i.$$

Definieren wir $\overline{p}(I) := \sum_i p(I_i)$, so gilt offenbar weiterhin

$$\overline{p}(I) + \overline{p}(J) + \overline{p}(K) = Pq,$$

somit ist $T^{\otimes q}$ eine $(\overline{p}, Pq)$-uniforme Trilinearform. Wir können $T^{\otimes q}$ also analog zur Zerlegung von T in Trilinearformen der Form

$$T_{IJK} = \sum_{i,j,k:\overline{p}(i)=I,\overline{p}(j)=J,\overline{p}(k)=K} x_i y_j z_k$$

aufteilen. Diese werden sich aber - wie auch in der Konstruktion von CW - Variablen teilen, weshalb wir ähnlich zum Vorgehen von CW in die s-te Tensorpotenz übergehen und Variablen nullsetzen, bis wir eine genügend große Anzahl disjunkter Trilinearformen erhalten. Anschließend kann mit (Satz 9) eine Schranke für ω gefolgert werden. Die gegebenen V_{IJK} sind die Values dieser T_{IJK}.
Die s-te Tensorpotenz $(T^{\otimes q})^{\otimes s}$ enthält Variablen mit Multi-Indizes der Länge s, deren Einträge einzelne Variablen innerhalb der verschiedenen ineinander geschachtelten Kopien von $T^{\otimes q}$ auswählen. Damit sind die Einträge Multi-Indizes der Länge q. Wir bilden nun Multi-Indizes in $(T^{\otimes q})^{\otimes s}$ folgendermaßen auf Blöcke ab:

$$(i_1, \ldots, i_s) \mapsto (\overline{p}(i_1), \ldots, \overline{p}(i_s))$$

Die Blöcke in $(T^{\otimes q})^{\otimes s}$ sind also nun Multi-Indizes von Blöcken in $T^{\otimes q}$.
Wir führen nun Variablen $A_I \in [0, 1] \subseteq \mathbb{Q}$ für alle Blöcke I ein. Diese sollen den Anteil der einzelnen Blöcke innerhalb eines Block-Multi-Indizes angeben. Da diese Anteile vollständig die Blöcke innerhalb der Block-Multi-Indizes beschreiben sollen, muss für die A_I gelten:

$$1 = \sum_I A_I$$

Für große s meint das in der s-ten Tensorpotenz, dass in den Multi-Indizes jeder Block I genau $s \cdot A_I$ mal auftritt, denn für große s ist $s \cdot A_I \in \mathbb{N}_0$, da $A_I \in \mathbb{Q}$. Dieser Aufbau der Block-Multi-Indizes muss zunächst nicht vorliegen, deshalb erzwingen wir ihn nun: Wir setzen alle Variablen Null, deren Multi-Index nicht auf einen Block-Multi-Index mit entsprechender Aufteilung der Blöcke abgebildet wird. Diese Vorgehensweise entspricht in der einfacheren Konstruktion von CW dem Nullsetzen aller Variablen, deren Block-Multi-Index nicht s Nullen und $2s$ Einsen enthält.
Es werden weiterhin Variablen a_{IJK} eingeführt. Diese verfeinern die Aussage der A_I in dem Sinne, dass sie die Anteile der einzelnen Block-Tripel in den Block-Tripeln von $(T^{\otimes q})^{\otimes s}$, anstatt nur die

[15] $I = (I_i), J = (J_i), K = (K_i)$ sind dabei wieder Multi-Indizes

Häufigkeit der Blöcke in einem einzelnen Block-Multi-Index, angeben. Beachte dazu, dass wir ein Block-Tripel nach den Positionen der drei Block-Multi-Indizes sortieren können. Damit erhalten wir für jede Position ein Block-Tripel von $T^{\otimes q}$, wobei dieses das Block-Tripel in der entsprechenden Verschachtelungstiefe der Tensorpotenz angibt (also in einer Kopie von $T^{\otimes q}$). Da $T^{\otimes q}$ eine $(\overline{p}, Pq)$-uniforme Trilinearform ist und sich die A_I auf alle drei Blöcke eines Block-Tripels gleichermaßen beziehen, gilt für die a_{IJK}:

$$A_I = \sum_J a_{IJ(Pq-I-J)} = \sum_J a_{JI(Pq-I-J)} = \sum_J a_{(Pq-I-J)JI}$$

Zudem beschreiben die a_{IJK} vollständig die Aufteilung der Block-Tripel, deshalb gilt:

$$1 = \sum_I A_I = \sum_{I,J,K} a_{IJK}$$

Wir setzen zur Verkürzung der folgenden Gleichungen

$$N := \sum_{[a_{IJK}]} \prod_I \binom{s \cdot A_I}{[s \cdot a_{IJK}]_J}, \quad N' := \prod_I \binom{s \cdot A_I}{[s \cdot a_{IJK}]_J}$$

Dabei meinen $\binom{s}{[s \cdot A_I]_I}$ und ähnliche Darstellungen die Multinomialkoffizienten, z.B.: $\binom{s}{[s \cdot A_I]} = \frac{s!}{\prod_I (s \cdot A_I)!}$.
Die Summe in N läuft über alle gültigen Wahlen der a_{IJK}, während für N' bereits eine feste Belegung der a_{IJK} gewählt wurde (s.u.). Wir erhalten - noch ohne eine feste Wahl der a_{IJK} - folgende Darstellung der Anzahl von Block-Tripeln, die unserer Wahl der A_I entsprechen:

$$\binom{s}{[s \cdot A_I]} \cdot N$$

Wir legen uns nun auf eine Wahl der a_{IJK} fest, setzen aber zunächst keine weiteren Blöcke Null. Betrachten wir nur die Block-Tripel in $(T^{\otimes q})^{\otimes s}$, die unserer Wahl der a_{IJK} entsprechen, erhalten wir für die zugehörigen Trilinearformen mindestens einen Value von

$$\prod_{I,J} V_{IJK}^{a_{IJK} \cdot s}.$$

Für die Anzahl von Block-Tripeln, die unserer Wahl der a_{IJK} entsprechen, ergibt sich:

$$N' \cdot \binom{s}{[s \cdot A_I]}$$

Würden diese Trilinearformen keine gemeinsamen Variablen enthalten, dann könnten wir bereits eine Schranke für ω folgern. Allerdings werden sie i.A. gemeinsame Variablen enthalten, weshalb nun mit Hilfe einer Salem-Spencer-Menge analog zum Vorgehen von CW in ihrer einfacheren Konstruktion durch Nullsetzen weiterer Variablen Überschneidungen zwischen den verschiedenen Trilinearformen entfernt werden.
Es wird also nun eine Primzahl $M \in \Theta(N)$ gewählt und dann eine Salem-Spencer-Menge S mit $|S| = M^{1-\varepsilon}$ gebildet (später lassen wir $\varepsilon \to 0$ laufen). Dies ist wieder möglich, da N durch entsprechende Wahl von s beliebig groß wird. Anschließend definieren wir Zufallsvariablen $w_0, \ldots, w_s \in \{0, \ldots, M-1\}$ und drei Hash-Funktionen $b_X(I), b_Y(J), b_Z(K)$ für Blöcke I, J, K der Tensorpotenz $(T^{\otimes q})^{\otimes s}$, die leichte Abwandlungen der Hash-Funktionen aus der einfacheren Konstruktion von CW sind. Wir wollen diese hier nicht explizit angeben, es soll genügen, dass sich diese so definieren lassen, dass sie (wie bereits bei der einfacheren Konstruktion von CW) folgende Eigenschaften haben:

1. Für alle Block-Tripel (I,J,K) in $(T^{\otimes q})^{\otimes s}$ gilt $b_X(I)+b_Y(J)-2b_Z(K)=0$.
2. Für jedes $b \in S$ (bzw. sogar $b \in \{1,\dots,M-1\}$) sind die Ereignisse $b_X(I)=b$, $b_Y(J)=b$ und $b_Z(K)=b$ stochastisch unabhängig.
3. Die Wahrscheinlichkeiten dafür, dass die einzelnen Werte aus M als Hash-Wert auftreten, sind gleichverteilt

Setzen wir also wie bei CW alle Blöcke Null, die nicht auf Elemente der Menge S abgebildet werden, so gilt aufgrund der Eigenschaften der Salem-Spencer-Menge für alle übrigen Block-Tripel (I,J,K):

$$b_X(I) = b_Y(J) = b_Z(K)$$

Aufgrund der Gleichverteilung und der stochastischen Unabhängigkeit ergibt sich der Erwartungswert für den Anteil übriger Block-Tripel nach diesem Schritt zu

$$\frac{M^{1-\varepsilon}}{M} \cdot \frac{1}{M} = \frac{1}{M^{1+\varepsilon}}$$

Auch bei Beschränkung auf die Block-Tripel, die unserer Wahl der a_{IJK} entsprechen, gilt dieser Erwartungswert für den Anteil übriger Tripel.
Es folgt, wie in der einfacheren Konstruktion von CW, ein Schritt, in dem für jedes Element von S die Menge der zugehörigen Block-Tripel gebildet wird und dann innerhalb dieser Mengen jeweils Überschneidungen beseitigt werden. Wie zuvor ist dies möglich, da jedem übrigen Block-Tripel nur noch ein Hash-Wert zugeordnet ist (da alle drei Blöcke des Tripels auf denselben Wert gehasht werden). Zudem kann es keine Überschneidungen zwischen Elementen verschiedener Mengen geben, da für verschiedene Hash-Werte auch die Blöcke des Tripels verschieden sind. Williams passt die Vorgehensweise dabei leicht an, um statt einer Abschätzung für den Erwartungswert der Anzahl übriger Tripel insgesamt eine Abschätzung für den Erwartungswert der Anzahl von Tripeln, die unserer Wahl der a_{IJK} entsprechen, zu erhalten. Sie erhält, dass dieser größergleich folgendem Term ist:

$$\frac{\binom{s}{[s\cdot A_I]} N'}{CM^{1+\varepsilon}}$$

Die w_i lassen sich so wählen, dass mindestens dieser Erwartungswert erreicht wird.
Sei $N_{\max}$ der größtmögliche Wert, den N' für eine Wahl der a_{IJK} annehmen kann. Es ergibt sich (intuitiv leicht nachvollziehbar):

$$N_{\max} \leq N \leq \operatorname{poly}(s) \cdot N_{\max}$$

Dabei ist $\operatorname{poly}(s)$ ein Polynomausdruck in s. Mit dieser Gleichung und $M \in \Theta(N)$ gilt:

$$\frac{\binom{s}{[s\cdot A_I]} N'}{CM^{1+\varepsilon}} \in \Omega\left(\binom{s}{[s\cdot A_I]} \frac{N'}{N_{\max}} \cdot \frac{1}{\operatorname{poly}(s)\cdot M^{\varepsilon}}\right)$$

Dies ist eine Abschätzung für den Erwartungswert der Anzahl von unserer Wahl der a_{IJK} entsprechenden Block-Tripeln, zwischen denen sich keine Variablen überschneiden und damit auch eine Abschätzung für die Anzahl disjunkter Trilinearformen, die mindestens Value

$$\prod_{I,J} V_{IJK}^{a_{IJK}\cdot s} \quad \text{haben.}$$

Aufgrund der Gleichheit des Ranges von Trilinearformen mit permutierten Variablen ergibt sich direkt auch, dass der Value der Trilinearform eines Block-Tripels unter Permutation der Blöcke gleich bleibt. Außerdem ergibt sich nach Williams, dass eine beste Wahl der a_{IJK} die a_{IJK} ebenfalls unter allen Permutationen der Blöcke gleich setzt. Deshalb definieren wir uns die Funktionen

sort(IJK) und perm(IJK), wobei sort die Blöcke lexikographisch sortiert zurückliefert und perm die Anzahl der echt verschiedenen Permutationen der Werte von I, J und K liefert. Damit erhalten wir mit den vorherigen Ergebnissen nun für die s-te Tensorpotenz:

$$r^{qs} \geq \underbrace{\binom{s}{[s \cdot A_I]} \frac{N'}{N_{\max}} \cdot \frac{1}{\text{poly}(s) \cdot M^{\varepsilon}}}_{\text{Anzahl der Trilinearformen}} \cdot \underbrace{\prod_{I \leq J \leq K} V_{IJK}^{\text{perm}(IJK) \cdot s \cdot a_{IJK}}}_{\text{Abschätzung des Values je Trilinearform}}$$

Seien $\overline{a}_{IJK}$ die Belegungen der a_{IJK}, für die genau $N' = N_{\max}$ erreicht wird. Dann können wir in der obigen Gleichung $N_{\max}$ entsprechend ersetzen. Durch relativ lange Umformungen, für die unter anderem die Stirling-Formel als Abschätzung für den Wert einer Fakultät genutzt wird, ergibt sich daraus

$$r^q \geq \left(\prod_{I \leq J \leq K} \left(\frac{\overline{a}_{IJK}^{\overline{a}_{IJK}}}{a_{IJK}^{a_{IJK}}} \right)^{\text{perm}(IJK)} \cdot V_{IJK}^{\text{perm}(IJK) \cdot a_{IJK}} \right) \Big/ \prod_I A_I^{A_I}$$

Mit der vereinfachenden Annahme[16] $a_{IJK} = \overline{a}_{IJK}$ - oder anders gesagt $N_{\max} = N'$ - wird daraus

$$r^q \geq \left(\prod_{I \leq J \leq K} V_{IJK}^{\text{perm}(IJK) \cdot a_{IJK}} \right) \Big/ \prod_I A_I^{A_I}$$

Da die Values monoton steigende Funktionen in τ sind, liefert diese Gleichung eine Schranke für τ und mit (Satz 9) auch für ω.

Bestimmen der Werte für die Variablen $\overline{a}_{IJK}$ und a_{IJK} von oben:
Es gilt für die Variablen nach Definition die Beschränkung

$$A_I = \sum_J a_{IJK} = \sum_J \overline{a}_{IJK}.$$

Außerdem gilt für die passende Wahl der $\overline{a}_{IJK}$ nach Williams $\overline{a}_{IJK} = \overline{a}_{\text{sort}(IJK)}$, weshalb sich obige Gleichung schreiben lässt als

$$1 = \sum_I A_I = \sum_{I \leq J \leq K} \text{perm}(IJK) \cdot \overline{a}_{IJK}.$$

Da die a_{IJk} nur noch in geringem Ausmaß variiert werden können, wenn die $\overline{a}_{IJK}$ festgehalten werden, beschränken wir uns auf die Bestimmung der $\overline{a}_{IJK}$. Offenbar ergibt $A_I = \sum_J \overline{a}_{IJK}$ ein lineares Gleichungssystem in den Variablen $\overline{a}_{IJK}$ mit Konstanten A_I. Hat dies vollen Rang, so können wir die $\overline{a}_{IJK}$ direkt aus den A_I bestimmen. Dadurch wären dann auch sofort die a_{IJK} bestimmt. Wenn nicht, gehen wir wie folgt vor:
Wir bilden eine minimale Menge X von Variablen $\overline{a}_{IJK}$, sodass das lineare Gleichungssystem vollen Rang hat, falls wir alle Variablen in der Menge X als Konstanten auffassen. Dann ergeben sich die Variablen außerhalb von X als Lösungen dieses Gleichungssystems, also als Linearkombinationen der A_I und der Variablen in X. Wir schreiben dann den zu optimierenden Term als Funktion (da die Wahl der Variablen $\overline{a}_{IJK}$ gesucht ist, für die dieser maximal wird):

$$G := \prod_I \binom{s \cdot A_I}{[s \cdot \overline{a}_{IJK}]_{\overline{a}_{IJK} \in X}, [s \cdot \overline{a}_{IJK}]_{\overline{a}_{IJK} \notin X}}$$

[16] Williams verwendet diese für ihre Anwendungen des Algorithmus, da sie die Berechnung vereinfachen und ihrer Erfahrung nach nicht deutlich schlechtere Ergebnisse liefern

G ist eine Funktion in den Variablen aus X. Mit analytischen Methoden lässt sich diese maximieren und wir erhalten schließlich mit[17]

$$y_{(I'J'K'IJK)} := \frac{\partial \overline{a}_{I'J'K'}}{\partial \overline{a}_{IJK}}$$

für jedes $\overline{a}_{IJK}$ eine Gleichung

$$\overline{a}_{IJK} \cdot \prod_{\overline{a}_{I'J'K'} \notin X, y_{(I'J'K'IJK)} > 0} (\overline{a}_{I'J'K'})^{y_{(I'J'K'IJK)}} = \prod_{\overline{a}_{I'J'K'} \notin X, y_{(I'J'K'IJK)} < 0} (\overline{a}_{I'J'K'})^{-y_{(I'J'K'IJK)}}$$

Damit haben wir die Möglichkeit, Werte für die Variablen, die nicht in X liegen, auszuprobieren und mit Hilfe dieser Gleichungen die Werte der Variablen in X zu bestimmen, wodurch sich auch die A_I ergeben. Für die Werte der A_I, die sich dabei ergeben, ist die vorliegende Wahl der Variablen $\overline{a}_{IJK}$ dabei jeweils optimal. Wird unter den gültigen Möglichkeiten, also den Belegungen der Variablen außerhalb von X, für die sich $\sum_I A_I = 1$ und nicht-negative Variablen in X ergeben, nach einem Maximum gesucht, so wird mit dieser Vorgehensweise eine Lösung erhalten und damit insgesamt auch eine Schranke für ω.
Die Vorgehensweise, die in diesem Abschnitt beschrieben wurde, fasst Williams als Ergebnis in dem ersten ihrer Algorithmen zusammen.

3.12.2 Analyse der Values V_{IJK} der Trilinearformen in $T^{\otimes q}$ für beliebige q

Die Idee hierfür ist, die Values der zu den einzelnen Block-Tripeln gehörigen Trilinearformen in einer gewissen Tensorpotenz $T^{\otimes q}$ rekursiv aus den Values der Trilinearformen in kleineren Tensorpotenzen zu berechnen. Dazu stellen wir fest, dass sich die Potenz folgendermaßen aufspalten lässt:

$$T^{\otimes q} = T^{\otimes q'} \otimes T^{\otimes (q-q')}$$

Wir werden deshalb beschreiben, wie unter Verwendung der Values der Trilinearformen in $T^{\otimes q'}$ und $T^{\otimes (q-q')}$ die Values der Trilinearformen in $T^{\otimes q}$ bestimmt werden können. Dafür definieren wir uns analog zu $\overline{p}$ oben die Funktionen p_q für beliebige Tensorpotenzen, es soll also jeweils für einen Multi-Index $I = (I_i)_{(i \in \{1,\ldots,q\})}$ (also einen Index in der q-ten Tensorpotenz) gelten:

$$p_q(I) = \sum_{i=1}^{q} p(I_i).$$

Damit lassen sich die Multi-Indizes der q-ten Tensorpotenz auffassen als Paare (l,m) aus einem Multi-Index l der q'-ten Tensorpotenz und einem Multi-Index m der $(q-q')$-ten Tensorpotenz und es gilt:

$$p_q(l,m) = p_{q'}(l) + p_{(q-q')}(m)$$

Wir beginnen jetzt, die Trilinearformen zu einzelnen Block-Tripeln zu untersuchen. Zu jedem Block-Tripel (I,J,K) in $T^{\otimes q}$ sei T_{IJK} die Trilinearform, die alle Summanden $x_i y_j z_k$ enthält, für die $p_q(i) = I, p_q(j) = J, p_q(k) = K$ gilt. Ausgehend von den einzelnen Block-Tripeln (I,J,K) in $T^{\otimes q}$ definieren wir nun Block-Tripel (i,j,k) der q'-ten Tensorpotenz als gute Tripel, wenn (i,j,k) ein gültiges Block-Tripel der q'-ten Tensorpotenz und außerdem $(I-i, J-j, K-k)$ ein gültiges Block-Tripel der $(q-q')$-ten Tensorpotenz ist. Damit erhalten wir

$$T_{IJK} = \sum_{\text{gute } (i,j,k)} T_{ijk} \otimes T_{I-i,J-j,K-k},$$

[17] ∂ ist das Zeichen für die partielle Integration

wobei dies keine direkte Summe ist, die einzelnen Summanden also keine disjunkten Trilinearformen sind. Der Value eines Summanden $T_{ijk} \otimes T_{I-i,J-j,K-k}$ ist dabei jeweils mindestens

$$V_{ijk} \cdot V_{I-i,J-j,K-k}.$$

Um disjunkte Trilinearformen zu erhalten, gehen wir nun wieder in die s-te Tensorpotenz über, wobei wir später wieder $s \to \infty$ laufen lassen. Wir wandeln diese Vorgehensweise hier allerdings leicht ab. So bilden wir für die drei Trilinearformen $T_{IJK}, T_{KIJ}, T_{JKI}$ jeweils die s-te Tensorpotenz und bilden dann ihr tensorielles Produkt, wodurch eine gewisse Symmetrie erreicht und damit das weitere Vorgehen erleichtert wird. Wir gehen dann für die s-te Tensorpotenz der drei Trilinearformen analog zueinander vor und beschreiben deshalb nur für $T_{IJK}^{\otimes s}$ das Vorgehen:
Analog zu den A_I aus dem vorherigen Abschnitt, definieren wir Variablen X_i, Y_j, Z_k, wobei i, j, k dabei alle Werte annehmen, die in guten Block-Tripeln für T_{IJK}, also Block-Tripeln von $T^{\otimes q'}$, vorkommen. Die X_i, Y_j, Z_k geben nun den Anteil der einzelnen Blöcke $(i, I-i), (J-j)$ bzw. $(K-k)$ innerhalb der Block-Multi-Indizes von $T_{IJK}^{\otimes s}$ an, wobei sich X_i auf die x-Variablen, Y_j auf die y-Variablen und Z_k auf die z-Variablen bezieht. Wir setzen dann alle x-Variablen Null, deren Block-Multi-Index nicht für alle i genau $X_i \cdot s$ Blöcke $(i, I-i)$ enthält. Analog setzen wir auch y- und z-Variablen aufgrund der Y_j und Z_k Null. Es bleiben damit folgende Anzahlen von Blöcken übrig:

$$\text{für x: } \binom{s}{[s \cdot X_i]_i}, \text{ für y: } \binom{s}{[s \cdot Y_j]_j}, \text{ für z: } \binom{s}{[s \cdot Z_k]_k}$$

Wenden wir diese Vorgehensweise entsprechend auf T_{KIJ} und T_{JKI} an, sodass sich jeweils die X_i auf den Block I, die Y_j auf den Block J und die Z_k auf den Block K beziehen, ist die Anzahl übriger Blöcke im tensoriellen Produkt $T_{IJK}^{\otimes s} \otimes T_{KIJ}^{\otimes s} \otimes T_{JKI}^{\otimes s}$ für x, y und z wieder gleich, nämlich:

$$\Gamma := \binom{s}{[s \cdot X_i]_i} \cdot \binom{s}{[s \cdot Y_j]_j} \cdot \binom{s}{[s \cdot Z_k]_k}$$

Wir definieren analog zu den a_{IJK} des vorherigen Abschnitts Variablen α_{ijk} für alle guten Tripel (i, j, k), wobei diese die Gleichungen $X_i = \sum_j \alpha_{ijk}$, $Y_j = \sum_i \alpha_{ijk}$ und $Z_k = \sum_i \alpha_{ijk}$ erfüllen. Die Bedeutung soll wieder sein, dass diese Variablen angeben, wie häufig die einzelnen Block-Tripel (i, j, k) innerhalb der Tripel von Block-Multi-Indizes vorkommen.
Die Anzahl der Block-Tripel, die einen festgelegten x-, y- oder z-Block enthalten und einer bestimmten Wahl der α_{ijk} entsprechen, ist damit

$$N := \prod_i \binom{s \cdot X_i}{[s \cdot \alpha_{ijk}]_j} \prod_j \binom{s \cdot Y_j}{[s \cdot \alpha_{ijk}]_i} \prod_k \binom{s \cdot Z_k}{[s \cdot \alpha_{ijk}]_i}.$$

Die Anzahl der Tripel, die einer bestimmten Wahl der α_{ijk} entsprechen, ist $\Gamma \cdot N$.
Nun folgt wieder die übliche Vorgehensweise unter Verwendung einer Salem-Spencer-Menge und Zufallsvariablen $w_0, \ldots, w_{3s} \in \{0, \ldots, M-1\}$. Die Besonderheit ist dieses Mal, dass die Hash-Funktionen für Block-Paare $(i, I-i)$ nur auf der vorderen Komponente i definiert werden. Ansonsten verläuft die Argumentation wie in den vorherigen Abschnitten, weshalb wir diese hier auslassen. Als Ergebnis erhalten wir, dass der Erwartungswert für die Anzahl von Block-Tripeln, die einer bestimmten Wahl der α_{ijk} entsprechen, nach Beseitigung der Überschneidungen in

$$\Omega(\Gamma / M^{\varepsilon})$$

liegt, es gibt also für $\varepsilon \to 0$ eine Belegung der w_i, für die ungefähr Γ Block-Tripel, die einer bestimmten Wahl der α_{ijk} entsprechen, übrig bleiben. Der Value der zu diesen Block-Tripeln gehörenden Trilinearformen ist dabei jeweils

$$\geq \prod_{\text{gute } ijk} (V_{ijk} \cdot V_{I-i,J-j,K-k})^{3s\alpha_{ijk}}$$

und wir erhalten insgesamt

$$V_{IJK}^{3s} \geq \prod_i \binom{s}{[s \cdot X_i]} \prod_j \binom{s}{[s \cdot Y_j]} \prod_k \binom{s}{[s \cdot Z_k]} \prod_{ijk} (V_{ijk} \cdot V_{I-i,J-j,K-k})^{3s\alpha_{ijk}}$$

Die Bedingungen, die von den Variablen α_{ijk} erfüllt werden müssen, liefern wieder ein Gleichungssystem. Wie im vorherigen Abschnitt muss dieses nicht vollen Rang haben, wir können aber eine minimal Menge Δ von Variablen α_{ijk} festlegen, sodass das Gleichungssystem vollen Rang hat, wenn wir diese als Konstanten betrachten. Da damit alle Variablen α_{ijk} Linearkombinationen der Variablen in Δ und der Variablen X_i, Y_j, Z_k sind, können wir für jedes α_{ijk} eine Darstellung folgender Form angeben:

$$\alpha_{ijk} = \sum_{y \in \Delta \cup \{X_{i'}, Y_{j'}, Z_{k'}\}_{i',j',k'}} y \frac{\partial \alpha_{ijk}}{\partial y}$$

Damit können wir in obiger Gleichung für den Value die α_{ijk} in Summanden zerlegen. Setzen wir

$$W_{ijk} := V_{ijk} \cdot V_{I-i,J-j,K-k},$$

$$nx_\ell := \prod_{ijk} W_{ijk}^{3\frac{\partial \alpha_{ijk}}{\partial X_\ell}}, \quad ny_\ell := \prod_{ijk} W_{ijk}^{3\frac{\partial \alpha_{ijk}}{\partial Y_\ell}}, \quad nz_\ell := \prod_{ijk} W_{ijk}^{3\frac{\partial \alpha_{ijk}}{\partial Z_\ell}} \quad \forall \ell,$$

so ergibt sich obige Gleichung zu:

$$V_{IJK}^{3s} \geq \binom{s}{[s \cdot X_i]} \prod_\ell nx_\ell^{sX_\ell} \binom{s}{[s \cdot Y_i]} \prod_\ell nz_\ell^{sY_\ell} \binom{s}{[s \cdot Z_i]} \prod_\ell nz_\ell^{sZ_\ell} \prod_{ijk} W_{ijk}^{3s\left(\sum_{y \in \Delta} y \frac{\partial \alpha_{ijk}}{\partial y}\right)}$$

Williams zeigt zudem, für welche Belegung der X_i Terme von der Form $\binom{s}{[s \cdot X_i]} \prod_\ell nx_\ell^{sX_\ell}$ maximiert werden und gibt als Abschätzung für den maximalen Wert eines solchen Terms $(\sum_\ell nx_\ell)^s / \mathrm{poly}(s)$ an. Damit ergibt sich durch Umformungen (Stirling-Formel, $s \to \infty$, ziehen der $3s$-ten Wurzel) schließlich

$$V_{IJK} \geq \left(\sum_\ell nx_\ell\right)^{1/3} \left(\sum_\ell ny_\ell\right)^{1/3} \left(\sum_\ell nz_\ell\right)^{1/3} \prod_{ijk} W_{ijk}^{3s\left(\sum_{y \in \Delta} y \frac{\partial \alpha_{ijk}}{\partial y}\right)}.$$

Dabei sind die X_ℓ, Y_ℓ, Z_ℓ bereits durch die vorhergehende Maximierung festgelegt. Dies ist für $\Delta = \varnothing$ bereits eine vollständige Formel für V_{IJK}. Falls $\Delta \neq \varnothing$, so werden einfach die Variablen in Δ, für die die Variablen außerhalb von Δ nicht-negative Werte annehmen, durchprobiert und so das Maximum bestimmt.
Auf diese Weise lassen sich also rekursiv die Values der Trilinearformen in der q-ten Tensorpotenz bestimmen. Williams fasst diese Vorgehensweise in ihrem zweiten Algorithmus zusammen. Zudem stellt sie fest, dass für die q-te Tensorpotenz der Algorithmus $O(q^2)$ Mal angewendet werden muss, wenn wir jeweils $q' = \lfloor q/2 \rfloor$ und somit $q - q' = \lceil q/2 \rceil$ setzen.

3.12.3 Sonstiges

Neben diesen beiden Hauptergebnissen zeigt Williams auch noch, inwiefern sich der zweite Algorithmus vereinfachen lässt, wenn als Exponent der Tensorpotenz eine Zweierpotenz betrachtet wird. Dass dies möglich ist, leuchtet auf den ersten Blick ein, da $\lfloor q/2 \rfloor$ und $\lceil q/2 \rceil$ damit für alle Stufen der Rekursion gleich werden und somit die Values für beide Teile auf einmal erhalten werden.
Außerdem wendet sie die Verfahren auf verschiedene Tensorpotenzen des Algorithmus von CW an und erhält damit für die 8te Tensorpotenz die derzeitig beste Schranke $\omega < 2{,}3727$.

4 Andere Resultate

Neben den vorgestellten Resultaten gibt es noch viele weitere. Dazu gehören zum einen teilweise andere Algorithmen, die die großen Resultate angewendet und damit bessere Schranken erreicht haben, als die in den Originalpublikationen erreichten. Zum anderen gehören auch Ideen dazu, die nicht zu einer besseren Schranke geführt haben, die aber auch vielversprechend aussehen, so zum Beispiel einige alternative Ideen von Coppersmith und Winograd, die sie in derselben Publikation veröffentlichten wie den CW-Algorithmus, also in [CW87].
Daneben wurde auch versucht, sich dem Problem der Bestimmung von ω von der anderen Seite aus zu nähern, also untere Schranken zu suchen. Allerdings gelang es dabei nicht, die triviale untere Schranke $\omega \geq 2$ zu übertreffen[1]. Zwar gab es auch in Bezug auf die Mindestanzahl von benötigten Operationen Fortschritte, aber diese hatten asymptotisch keine Verbesserung zur Folge.
Alle in dieser Arbeit vorgestellten Ergebnisse stützen sich auf die Beschreibung der MM als Trilinearform oder Menge von Bilinearformen bzw. mittels Tensoren. Neben diesem Ansatz gibt es jedoch auch noch andere. Zum einen gibt es eine gruppentheoretische Beschreibung der MM, siehe dafür [CU03] und [CKSU05]. Auf diese Art lassen sich ebenfalls Algorithmen konstruieren und Schranken für ω folgern. Obwohl dieser Ansatz bisher noch nicht die besten Schranken unterbieten konnte, ist er dennoch sehr aussichtsreich: In den wenigen Jahren, in denen dieser Ansatz verfolgt wurde, wurde bereits ein bester Exponent von $2,41$ erreicht, was schon sehr nahe an die besten Exponenten der anderen Verfahren herankommt. Zum anderen gibt es noch einen geometrischen Ansatz, der von Landsberg in [Land08] beschrieben wird.
Es gab zumindest auf Basis der ersten beiden Ansätze auch häufig Versuche, direkt $\omega = 2$ zu beweisen. Gerade auf Basis des gruppentheoretischen Ansatzes gibt es hier - relativ zur Anzahl der Publikationen insgesamt - zahlreiche Versuche. Diese Beweisideen wurden jeweils mit Voraussetzungen veröffentlicht, die sich bislang nicht beweisen ließen.

[1] Diese ergibt sich daraus, dass das Problem (n,n,n) $2n^2$ Eingaben und n^2 Ausgaben hat, was intuitiv die Schranke $\omega \geq 2$ nachvollziehbar macht

5 Zusammenfassung und Ausblick

Die Geschichte des Exponenten der Matrizen-Multiplikation, ω, ist eine lange und viele verschiedene Ideen haben immer wieder Verbesserungen herbeigeführt. Besonders die Verbindung der Ideen waren dabei sehr effektiv, zudem fanden auch immer wieder Resultate, die sich nicht direkt auf die Matrizen-Multipliation beziehen, Einzug in die Beweise - so zum Beispiel die Salem-Spencer-Mengen.
Die verschiedenen Resultate zwischen 1968 und 1987 - abgesehen vom trilinearen AUC von Pan - ermöglichten mit der Zeit die Konstruktion von Algorithmen für Probleme, die immer weniger mit der eigentlichen Matrizen-Multiplikation bzw. der Darstellung als Menge von Bilinearformen zu tun hatten, der Suchraum für Algorithmen wurde also immer größer. Dabei kommt es häufig vor, dass sich selbst aus vergleichsweise einfachen Algorithmen mit Hilfe der allgemeinen Resultate bereits sehr niedrige obere Schranken für ω ergeben. Als Beispiele sind zum einen eine Publikation von Romani, siehe [Rom82], und zum anderen Skizzen für Algorithmen am Ende von [Pan81] und [Schö81] zu nennen. Seit 1987 kam es dann für lange Zeit nicht mehr zu weiteren Verallgemeinerungen und auch nicht zu neuen und besseren Algorithmen. Erst 2010 gelang es Stothers und Williams die obere Schranke für ω, die 1987 von Coppersmith und Winograd gefunden wurde, zu unterbieten - aber auch diese Resultate fanden dafür keinen völlig neuen Algorithmus oder starke Verallgemeinerungen, sondern untersuchten lediglich weitergehend, welche Schranken sich auf Basis des CW-Algorithmus ableiten lassen. Dabei verallgemeinerte Williams ein Stück weit die Idee des CW-Algorithmus.
Aufgrund der bisher aus dem CW-Algorithmus abgeleiteten Schranken lässt sich vermuten, dass sich keine drastisch besseren Schranken mehr daraus ableiten lassen, sondern wenn überhaupt nur noch minimale Verbesserungen möglich sind. Da andererseits die unteren Schranken für ω immer noch sehr weit von den oberen Schranken entfernt sind, stellt sich damit die Frage, ob sich noch weitere Verallgemeinerungen - wie die der Suche nach Basisalgorithmen für eine rekursive Konstruktion, das Erlauben von approximativen Algorithmen, die Berechnung partieller Matrizen-Multiplikationen an Stelle von vollständigen, die Berechnung mehrerer disjunkter Matrizen-Multiplikationen durch einen einzelnen Algorithmus und schließlich die Verwendung von Trilinearformen, die gar nicht mehr direkt die Struktur von MMs aufweisen - ergeben werden. Speziell stellt sich für die Verwendung von Trilinearformen, die nicht mehr direkt die Struktur von MMs aufweisen (wie zum Beispiel im CW-Algorithmus zu sehen) die Frage, ob sich diese Idee noch weiter verallgemeinern lässt, als durch Williams geschehen.
Die Hauptfrage, die hinter und über all diesen Fragen steht, bleibt aber, wie groß der Exponent der Matrizen-Multiplikation, ω, tatsächlich ist.

Literaturverzeichnis

[Str68] Volker Strassen, 1968: Gaussian Elimination is not Optimal (Seiten: 21ff.)

[Str73] Volker Strassen, 1973: Vermeidung von Divisionen (Seiten: 9)

[Str87-1] Volker Strassen, 1987: The asymptotic spectrum of tensors (Seiten: 54, 61)

[Str87-2] Volker Strassen, 1987: Relative bilinear complexity and matrix multiplication (Seiten: 54)

[Pan78] Victor Ya. Pan, 1978: Strassen's Algorithm is not Optimal - Trilinear Technique of Aggregating, Uniting and Canceling for Constructing Fast Algorithms for Matrix Operations (Seiten: 27ff.)

[Pan80] Victor Ya. Pan, 1980: New Fast Algorithms for Matrix Multiplication (Seiten: 27ff.)

[Pan81] Victor Ya. Pan, 1981: New Combinations of Methods for the Acceleration of Matrix Multiplication (Seiten: 7, 73)

[Pan84] Victor Ya. Pan, 1984: How Can We Speed Up Matrix Multiplication? (Seiten: 7)

[Bi79-1] Dario Bini, Milvio Capovani, Francesco Romani, Grazia Lotti, 1979: $O(n^{2.7799})$ Complexity for $n \times n$ Approximate Matrix Multiplication (Seiten: 35ff.)

[Bi79-2] Dario Bini, Grazia Lotti, Francesco Romani, 1979: Approximate Solutions for the Bilinear Form Computatioal Problem (Seiten: 31ff.)

[Bi80] Dario Bini, 1980: Relations Between Exact And Approximate Bilinear Algorithms. Applications (Seiten: 31ff.)

[Rom80] Francesco Romani, 1980: Complexity Measures for Matrix Multiplication Algorithms (Seiten: 32)

[Rom82] Francesco Romani, 1982: Some Properties of Disjoint Sums of Tensors Related to Matrix Multiplication (Seiten: 73)

[Schö81] Arnold Schönhage, 1981: Partial and Total Matrix Multiplication (Seiten: 31, 33, 35, 37ff., 45ff., 73)

[CW87] Don Coppersmith, Shmuel Winograd, 1987: Matrix Multiplication via Arithmetic Progressions (Seiten: 54ff., 71)

[Wi10] Virginia Vassilevska Williams, 2010: Breaking the Coppersmith-Winograd barrier (Seiten: 7, 62ff.)

[Wi12] Virginia Vassilevska Williams, 2012: An overview of the recent progress on matrix multiplication (Seiten: 7)

[Sto10] Andrew James Stothers, 2010: On the Complexity of Matrix Multiplication (Seiten: 7, 59, 62)

[CU03] Henry Cohn, Christopher Umans, 2003: A Group-theoretic Approach to Fast Matrix Multiplication (Seiten: 71)

[CKSU05] Henry Cohn, Robert Kleinberg, Balázs Szegedy, Christopher Umans, 2005: Group-theoretic Algorithms for Matrix Multiplication (Seiten: 71)

[Land06] Joseph M. Landsberg, 2006: The Border Rank of the Multiplication of 2×2 Matrices is Seven (Seiten: 32)

[Land08] Joseph M. Landsberg, 2008: Geometry and the Complexity of Matrix Multiplication (Seiten: 71)

[Tens] How to lose your fear of tensor products:
`http://www.dpmms.cam.ac.uk/~wtg10/tensors3.html`
(Seiten: 13)

[HM73] John Edward Hopcroft, Jean E. Musinski, 1973: Duality Applied to the Complexity of Matrix Multiplications and Other Bilinear Forms (Seiten: 24)