3 Die Grenze zwischen P und NPC

In diesem Abschnitt beziehen wir uns auf Handzettel, die von der Webseite zur Vorlesung runtergeladen werden können.

Eine alte Binsenweisheit besagt, dass Glück und Leid dicht beieinander liegen. Wenn wir unter Leid die *NP*-Härte eines zu lösenden Berechnungsproblems und unter Glück seine Lösbarkeit in Polynomialzeit verstehen, dann trifft die obige Feststellung auch auf den Algorithmenentwurf zu. Hnadzettel 2 (entnommen aus *Garey & Johnson*) zeigt, wie dicht Probleme aus P und NPC beieinander liegen können:

- Der **kürzeste** Pfad zwischen zwei Knoten in einem Graphen mit positiven Kantengewichten kann (zum Beispiel mit dem Algorithmus von Dijkstra) effizient gefunden werden. Der **längste** Pfad hingegen vermutlich nicht. Das korrespondierende Entscheidungsproblem, LONGEST PATH BETWEEN TWO VERTICES, ist nämlich (sogar eingeschränkt auf Einheitsgewichte für Kanten) *NP*-vollständig.[1]

- VERTEX COVER ist, wie wir bereits wissen, *NP*-vollständig. Es gibt also vermutlich keinen effizienten Algorithmus, der zu einem gegebenen Graphen $G = (V, E)$ die kleinste Knotenmenge $U \subseteq V$ findet, die von jeder Kante mindestens einen Randknoten enthält. Das duale EDGE COVER Problem, das nach einer kleinsten Kantenmenge fragt, die jeden Knoten überdeckt, ist hingegen mit Hilfe von MAXIMUM MATCHING effizient lösbar.

- Das Problem MINIUMUM EQUIVALENT DIGRAPH fragt nach einer kleinsten Teilmenge $A' \subseteq A$ von Kanten eines gegebenen Digraphen $G = (V, A)$, so dass zwei beliebige $u, v \in V$ genau dann in G durch einen Pfad (von u nach v) verbunden werden können, wenn dies in $G' = (V, A')$ möglich ist. TRANSITIVE REDUCTION unterscheidet sich von MINIUMUM EQUIVALENT DIGRAPH nur dadurch, dass $A' \subseteq V \times V$, d.h., A' muss keine Teilmenge von A sein. TRANSITIVE REDUCTION kann (mit Hilfe von TRANSITIVE CLOSURE und Berechnung von starken Zusammenhangskomponenten) effizient gelöst werden. Das Entscheidungsproblem zu MINIMUM EQUIVALENT DIGRAPH hingegen enthält DIRECTED HAMILTONIAN CIRCUIT[2] als Teilproblem und ist daher *NP*-vollständig.

- Eine Eingabeinstanz zum Problem SCHEDULING WITH INDIVIDUAL DEADLINES besteht aus einer Menge T von Aufgaben (tasks), einer partiellen Ordnung $\prec$ auf T, einem individuellen Schlusstermin (deadline) $d(t)$ für jede Aufgabe $t \in T$ und einer Anzahl m von Prozessoren. Jede Aufgabe t kann in einer Zeiteinheit auf einem Prozessor ausgeführt werden. $t \prec t'$ bedeutet, dass t ausgeführt sein muss, bevor

[1] Dies kann (relativ leicht) mit einer Kette von Karp-Reduktionen nachgewiesen werden, die von HAMILTONIAN CIRCUIT über HAMILTONIAN PATH (auf die offensichtiche Weise definiert) nach LONGEST PATH BETWEEN TWO VERTICES führt.

[2] genauer: die Einschränkung von DIRECTED HAMILTONIAN CIRCUIT auf stark zusammenhängende Digraphen, die auch *NP*-vollständig ist (gleiche Reduktion wie in der Vorlesung)

die Ausführung von t' begonnen wird. Die Frage ist, ob man die Aufgaben aus T den m Prozessoren so zuordnen kann, dass alle Schlusstermine eingehalten werden. Eine partielle Ordnung auf T heisst INTREE oder auch *Montagebaum*, wenn jedes $t \in T$ maximal einen unmittelbaren Nachfolger hat. Analog spricht man von einem OUTTREE oder auch *Sortierbaum*, wenn jedes $t \in T$ maximal einen unmittelbaren Vorgänger hat. INTREE SCHEDULING (bzw. OUTTREE SCHEDULING) ist die Spezialisierung von SCHEDULING WITH INDIVIDUAL DEADLINES auf partielle Ordnungen der Form INTREE (bzw. OUTTREE). Wie Brucker, Garey und Johnson 1977 gezeigt haben, kann INTREE SCHEDULING effizient gelöst werden, wohingegen OUTTREE Scheduling *NP*-vollständig ist (Reduktion von VERTEX COVER).

Es gehört zur Grundbildung der Informatik, das Grenzgebiet zwischen P und NPC gut zu kennen. Der obere Teil von Handzettel 3 zeigt den Grenzverlauf. Die untere Grenzlinie von NPC wird von den am weitesten eingeschränkten *NP*-vollständigen Teilproblemen gebildet. Jede weitere Spezialisierung eines dieser Probleme führt aus NPC heraus. Die obere Grenzlinie von P wird von den allgemeinsten Teilproblemen gebildet, die noch in Polynomialzeit gelöst werden können. Jede weitere Generalisierung eines dieser Probleme führt aus P heraus. Je dichter diese beiden Grenzlinien aneinanderliegen, desto kleiner ist das unbekannte Terrain der Probleme mit einem offenen Status. Die voranschreitende Forschung schiebt die Grenzlinien tendenziell aufeinander zu, da sowohl der Fundus der *NP*-vollständigen Probleme als auch der Fundus der effizienten Algorithmen ständig erweitert wird. Anschaulich gesprochen gibt es *kleine Engelchen*, die das bekannte Territorium von P auf immer allgemeinere Berechnungsprobleme ausdehnen, und *kleine Teufelchen*, die von immer spezielleren Problemen die *NP*-Vollständigkeit nachweisen. Es ist aus der strukturellen Komplexitätstheorie bekannt, dass (unter der Vorausetzung $P \neq NP$) die Klasse $NPC \setminus P$ nicht leer ist. So sehr also Engelchen und Teufelchen sich mühen, sie werden sich niemals auf einer scharfen Grenzlinie begegnen.

Wenden wir doch dieses allgemeine Schema einmal auf eine konkrete Problemhierachie an. PRECEDENCE CONSTRAINED SCHEDULING ist der Spezialfall von SCHEDULING WITH INDIVIDUAL DEADLINES bei dem die individuellen Schlusstermine alle identisch sind zu einem gemeinsamen Schlusstermin D. Mit einer von CLIQUE ausgehenden Karp-Reduktion kann man die *NP*-Vollständigkeit von PRECEDENCE CONSTRAINED SCHEDULING nachweisen. Aus dem unteren Teil von Handzettel 3 geht hervor, dass die Spezialfälle, bei denen die partielle Ordnung auf T leer oder ein Baum (Montage- oder Sortierbaum)[3] ist, in Polynomialzeit gelöst werden kann. Ebenso für allgemeine partielle Ordnung auf T und eine feste Anzahl 1, 2 oder 3 von Prozessoren (fleissige Engelchen). Für

[3]Für Montagebäume folgt dies aus dem oben zitierten allgemeineren Resultat für INTREE SCHEDULING (individuelle Schlusstermine). Für Sortierbäume kann man im Falle eines einheitlichen Schlusstermines D den Algorithmus für Montagebäume zusammen mit einem Symmetrieargument verwenden: Erstens, kehre im Sortierbaum SB die Orientierung aller Kanten um und erhalte einen Montagebaum MB. Zweitens, wende das Verfahren für Montagebäume an und erhalte einen optimalen Plan, der alle Aufgaben auf m Prozessoren in $T \leq D$ Zeiteinheiten $1, \ldots, T$ unter Einhaltung der (spiegelverkehrten) partiellen Ordnung MB ausführt. Drittens, spiegele den Plan, d.h., durchlaufe ihn in der zeitlichen Reihenfolge $T, \ldots, 1$. Dann wird die partielle Ordnung SB respektiert.

eine feste Anzahl von 4 oder mehr Prozessoren ist der Status des Problems offen.

Die Exploration des Grenzgebietes zwischen *NPC* und *P* geschieht in der Praxis meist für eine konkrete Problemhierarchie (wie zum Beispiel PRECEDENCE CONSTRAINED SCHEDULING).

Bei Graphenproblemen ergibt sich eine Problemhierarchie auf natürliche Weise, indem man von der Klasse aller (endlicher) Graphen zu speziellen Graphklassen übergeht. Zum Beispiel:

- Bäume
- planare Graphen
- Graphen mit beschränktem Knotengrad

Eine weitere Spezialisierung ergibt sich, indem man Variablen der Eingabe zu Konstanten "gefriert". Wir werden in Abschnitt 4 exemplarisch die Problemhierarchie zum Graphenfärbungsproblem diskutieren.

Für ein Zahlenproblem erhalten wir ein natürliches Teilproblem, indem wir verlangen, dass der Maximalbetrag eines Zahlparameters der Eingabe polynomiell durch die Eingabelänge beschränkt ist. Dies führt zu der Frage, ob ein Zahlenproblem nur darum nicht zu *P* gehört, weil man mit wenigen Bits "riesige Zahlen" spezifizieren kann, oder ob es auch schon für "moderate Zahlen" *NP*-vollständig ist. Probleme der ersten Kategorie heißen *pseudopolynomiell lösbar*, Probleme der letzteren Kategorie heißen *stark NP-vollständig*. Wir werden später sehen, dass zum Beispiel PARTITION pseudopolynomiell lösbar, aber eine Verallgemeinerung davon (3-PARTITION) stark *NP*-vollständig ist.

4 Analyse von eingeschränkten Graphproblemen

Eine *k-Färbung* eines Graphen $G = (V, E)$ ist eine Färbung der Knoten, die monochromatische Kanten vermeidet, d.h., eine Abbildung $f : V \to \{1, \ldots, k\}$ mit $f(v) \neq f(w)$ für alle $\{v, w\} \in E$. Zu gegebener Färbung heisst die Menge aller Knoten der gleichen Farbe i die *Farbklasse* zu i. Die *chromatische Zahl* $\chi(G)$ ist die kleinste Anzahl k von Farben, die eine legale Färbung von G ermöglicht. Das *Graphenfärbungsproblem* (als Optimierungsproblem) besteht darin, zu einem gegebenen Graphen G eine $\chi(G)$-Färbung anzugeben. COLORABILITY sei das Problem zu gegebenem Graphen G und gegebener Kostenschranke k zu entscheiden, ob G k-färbbar ist. k-COLORABILITY sei das Teilproblem, bei dem (die Konstante) k nicht Teil der Eingabe ist (Gefrieren einer Variable zu einer Konstanten).

Ein natürliches Anwendungsszenario für Graphenfärbung ist die Durchführung von Prozessen mit gemeinsamen Ressourcen. Zwei Prozesse stehen im Konflikt zueinander, wenn sie die gleiche Ressource benötigen. Zwei solche Prozesse können nicht im gleichen Zeitabschnitt ausgeführt werden. Prozesse und ihre Ressourcenkonflikte können durch einen Konfliktgraphen modelliert werden, dessen Knoten Prozesse und dessen Kanten Ressourcenkonflikte repräsentieren. Die chromatische Zahl des Konfliktgraphen gibt die minimale Anzahl von Zeitabschnitten an, die eine konfliktfreie Ausführung zulässt.

Es ist bekannt, dass 2-COLORABILTY zu P gehört. Wir skizzieren kurz die Grundidee. Eine Teilmenge U der Knotenmenge V eines Graphen $G = (V, E)$ heisst *unabhängig (Independent Set)*, wenn die Knoten aus U paarweise in G nicht benachbart sind. Man beachte, dass jede Farbklasse eine unabhängige Menge ist. G heisst *bipartit*, wenn V sich in zwei unabhängige Mengen zerlegen lässt. Das folgende Resultat ist nicht schwer zu zeigen:

Lemma 4.1 *Die folgenden Aussagen sind äquivalent:*

1. *G ist 2-färbbar.*
2. *G ist bipartit.*
3. *G enthält keine Kreise ungerader Länge.*

Da man im Rahmen einer systematischen Graphexploration (wie DFS oder BFS) die Existenz von Kreisen ungerader Länge leicht testen kann, erhalten wir die

Folgerung 4.2 *2-COLORABILITY$\in P$.*

Es bezeichne $\Delta(G)$ den maximalen Grad eines Knotens in G. Ein naives Verfahren zum Färben von G liefert der folgende gierige Algorithmus:
Sei $\Delta := \Delta(G)$. Färbe einen Knoten nach dem anderen mit Farben aus $C = \{1, \ldots, \Delta + 1\}$ gemäß folgender Strategie: der aktuelle Knoten v erhält die kleinste Farbe i aus C, die noch nicht für einen seiner Nachbarn verwendet wurde.
Dieses naive Verfahren ist effizient durchführbar und kommt mit $1+\Delta(G)$ Farben aus. Somit gilt $\chi(G) \leq \Delta(G) + 1$. Für die $(\Delta + 1)$-Clique ist diese Schranke scharf. Darüberhinaus gilt:

Lemma 4.3 (Satz von Brooks) — *ohne Beweis* —
Es sei G ein zusammenhängender Graph und $\Delta = \Delta(G)$. Dann ist $\chi(G) \leq \Delta$ sofern G nicht eine Clique der Größe $\Delta + 1$ bildet.

Folgerung 4.4 *COLORABILITY eingeschränkt auf Graphen vom Maximalgrad 3 gehört zu P.*

Beweis Wir zeigen, dass $\chi(G)$ effizient berechenbar ist. Da man die Zusammenhangskomponenten eines Graphen unabhängig voneinander färben kann, dürfen wir oBdA annehmen, dass der gegebene Graph zusammenhängend ist. Falls G nicht aus einem einzigen Knoten besteht, benötigen wir mindestens zwei Farben. Wie oben erwähnt können wir in Polynomialzeit testen, ob G 2-färbbar ist. Nehmen wir zu unseren Ungunsten an dass wir mindestens 3 Farben brauchen. Da $\Delta(G) \leq 3$, brauchen wir maximal 4 Farben. Falls G eine 4-Clique ist, benötigen wir genau 4 Farben. Andernfalls garantiert der Satz von Brooks, dass drei Farben ausreichen. **qed.**

Insgesamt hat sich also gezeigt, dass COLORABILTY eingeschränkt auf zwei Farben oder auf maximalen Knotengrad 3 zu P gehört. Soweit die Nachrichten aus den himmlischen Sphären. Hören wir uns an, was die bocksbeinigen Kerlchen aus der unteren Etage zu dem Thema zu sagen haben.

Theorem 4.5 3-*COLORABILITY ist NP-vollständig. Dies gilt sogar dann, wenn wir die zulässigen Eingaben auf planare Graphen vom maximalen Knotengrad* 4 *einschränken.*

Beweis Mitgliedschaft zu *NP* ist offensichtlich. Wir haben die *NP*-Härte nachzuweisen. Der Beweis vollzieht sich in drei Stufen.

Stufe 1 3-SAT$\leq_{pol}$ 3-COLORABILTY.

Die Reduktion erfolgt mit der "'Methode der verbundenen Komponenten"'. Abbildung 1 zeigt eine Übersicht, Das Herzstück der Reduktion ist die Klauselkomponente.

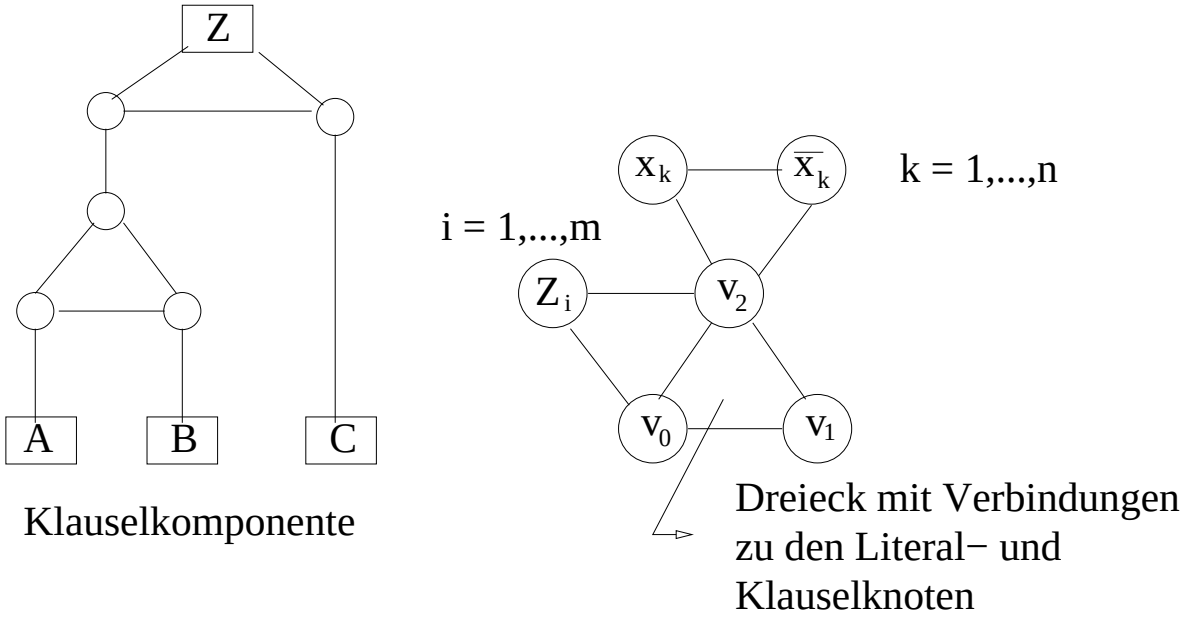

Figure 1: Ein Ausschnitt aus der Belegungskomponente im Falle $m = 3$: die Tripel aus T_k sind schraffiert, die Tripel aus F_k unschraffiert.

Sie ist so kunstvoll entworfen, dass legale 3-Färbungen (mit den Farben $0, 1, 2$) der Klauselkomponente implizit die korrespondierende Klausel "korrekt auswerten", wobei die Farben $0, 1$ die Booleschen Wahrheitswerte repräsentieren. Genauer gesagt gilt folgendes unter der Voraussetzung, dass für A, B, C nur die Farben $0, 1$ zur Verfügung stehen:

- Wenn A, B, C Farbe 0 haben, dann muss auch Z mit 0 gefärbt sein.
- Wenn einer der Knoten A, B, C die Farbe 1 hat, dann kann auch Z mit 1 gefärbt werden.

Die weiteren Kernbeobachtungen zu der Reduktion sind wie folgt:

- Das von v_0, v_1, v_2 gebildete Dreieck sorgt dafür, dass die Knoten v_0, v_1, v_2 drei verschiedene Farben erhalten: oBdA die Farben $0, 1, 2$.

- Das von $v_2, x_i, \bar{x}_i$ gebildete Dreieck sorgt dafür, dass die Farben von $x_i, \bar{x}_i$ entweder $0, 1$ oder $1, 0$ sind. Die Farben von $x_i, \bar{x}_i$ entsprechen somit einer Belegung der Booleschen Variablen x_i mit entweder 0 oder 1.
- Das von v_0, v_2, Z_i gebildete Dreieck sorgt dafür, dass Z_i mit 1 gefärbt werden muss. Intuitiv steht Z_i für den Wahrheitswert der i-ten Klausel.
- Zu jeder Klausel der Form $C_i = z_{i1} \vee z_{i2} \vee z_{i3}$ mit $z_{ij} \in \{x_1, \bar{x}_1, \ldots, x_n, \bar{x}_n\}$ konstruieren wir eine Klauselkomponente mit z_{i1}, z_{i2}, z_{i3} in den Rollen von A, B, C und Z_i in der Rolle von Z.

Aus diesen Beobachtungen ergibt sich leicht, dass gegebene 3-Klauseln $C_1, \ldots, C_m$ genau dann erfüllbar sind, wenn der zugehörige Graph 3-färbbar ist. Ergebnis von Stufe 1: 3-COLORABILITY ist *NP*-hart.

Stufe 2 Entwurf des "Crossover-Gadget"

Wir erinnern daran, dass ein Graph planar ist, wenn man die Knoten und Kanten so in die Ebene einbetten kann, dass Kanten sich nur an gemeinsamen Randpunkten berühren. Die in Stufe 1 benutzte Reduktion erzeugt nicht notwendig einen planaren Graphen. Wir können ihn aber in einen planaren Graphen transformieren, indem wir bei sich kreuzenden Kanten $\{x, x'\}$ und $\{y, y'\}$ ein "Crossover-Gadget" (s. oberen Teil von Handzettel 4) einbauen. Die Kreuzung wird dabei durch einen kreuzungsfreien Untergraphen ersetzt. Das Gadget ist so kunstvoll konstruiert, dass es beim Färben des Ursprungsgraphen keine neuen Freiheitsgrade einbaut und keine alten Freiheitsgrade vernichtet. Wie im mittleren Teil von Handzettel 4 zu sehen, kann das Crossover-Gadget benutzt werden, um die alle Kantenkreuzungen eines nichtplanaren ' Graphen sukzessive zu eliminieren. Ergebnis von Stufe 2: 3-COLORABILITY eingeschränkt auf planare Graphen ist *NP*-hart.

Stufe 3 Entwurf des "Portal-Gadgets"

Die in Stufen 1 und 2 skizzierte Reduktion erzeugt einen Graphen mit evtl. hohem maximalen Knotengrad. Wir setzen die Reduktion so fort, dass sie den maximalen Knotengrad auf 4 reduziert. Dies kann mit Hilfe einer lokalen Ersetzung geschehen (s. den unteren Teil von Handzettel 4). Dabei wird ein Knoten v mit k Nachbarn durch ein Portal-Gadget H_k ersetzt. H_k ist ein Untergraph mit inneren Knoten (die nicht durch Kanten mit Knoten ausserhalb H_k verbunden sind) und k Portalen. Jedes der Portale ist über eine Kante mit genau einem Nachbarn von v verbunden. Die Portale haben innerhalb H_k den Grad 3, also insgesamt den Grad 4. Die inneren Knoten von H_k haben Grad 4. H_k ist so kunstvoll konstruiert, dass die Portale alle die gleiche Farbe haben können und müssen. Man hat also beim Färben des neuen Graphen die gleichen Freiheitsgrade wie zuvor.

Stufe 3 schließt den Beweis des Theorems ab. **qed.**

Die Grenze zwischen P und NP hat sich beim Graphenfärbungsproblem somit (einigermaßen) präzise bestimmen lassen.

5 Starke NP-Vollständigkeit

5.1 Beispiele für pseudopolynomielle Algorithmen

Anna und Bert lassen sich durch die neuesten Nachrichten zur NP-Vollständigkeit nicht entmutigen. Anna hat eine Idee, um das PARTITION-Problem zu lösen. Seien $w_1, \ldots, w_n \geq 0$ die gegebenen Zahlen und $S = w_1 + \cdots + w_n$ ihre Summe. Nehmen wir oBdA an, dass S eine gerade Zahl ist. Anna will eine $n \times (1 + S/2)$-Tabelle $T = (t_{r,s})_{1 \leq r \leq n, 0 \leq s \leq S/2}$ mit den Booleschen Einträgen

$$t_{r,s} = \left(\exists I \subseteq \{1, \ldots, r\} : \sum_{i \in I} w_i = s \right) \tag{1}$$

berechnen. In Worten: die Indikatorvariable $t_{r,s}$ hat genau dann den Wahrheitswert 1, wenn sich aus den Zahlen $w_1, \ldots, w_r$ eine Auswahl mit Summenwert s treffen lässt. Die erste Zeile der Tabelle ist leicht auszurechnen:

$$t_{1,s} = ((s = 0) \vee (w_1 = s)) \,. \tag{2}$$

Gegeben Zeile $r-1$, dann lässt sich auch Zeile r mit Hilfe einer einfachen Vorschrift ausfüllen:

$$t_{r,s} = (t_{r-1,s} \vee ((s \geq w_r) \wedge t_{r-1,s-w_r})) \,.$$

Denn Summenwert s lässt sich genau dann mit einer Auswahl aus $w_1, \ldots, w_r$ erhalten, wenn er sich durch eine solche Auswahl **ohne** Teilnahme von w_r oder durch eine solche Auswahl **mit** Teilnahme von w_r erhalten lässt. Der erste Fall liegt genau dann vor, wenn man Summenwert S bereits durch eine Auswahl der Zahlen $w_1, \ldots, w_{r-1}$ erhalten kann. Der zweite Fall liegt genau dann vor, wenn $s \geq w_r$ und man Summenwert $s - w_r$ durch eine Auswahl der Zahlen $w_1, \ldots, w_{r-1}$ erhalten kann.
Anna kann die Rechenvorschriften (1) und (2) benutzen, um die Tabelle T zeilenweise auszufüllen. Offensichtlich gehört die Eingabe $w_1, \ldots, w_n$ genau dann zur Sprache PARTITION, wenn $t_{n,S/2}$ den Wahrheitswert 1 hat. Der Eintrag in der rechten unteren Ecke von Tabelle T verrät uns also die Lösung.
Das ist nicht übel, sagt Bert. *Ich habe übrigens eine ähnliche Methode zum Lösen von KNAPSACK (KP). Meine Tabelle ist auch 2-dimensional und hat Einträge $t_{r,s} \in \mathbb{N}_0$, die den maximalen Nutzen angeben, der sich mit den Objekten $1, \ldots, r$ unter Beachtung von Gewichtsschranke s erzielen lässt. Ich kann diese Tabelle mit einer ähnlich einfachen Rechenvorschrift systematisch ausfüllen wie Du Deine. Schon gut*, sagt Anna. *Ich kann mir vorstellen wie Du das machst.*[4] *Wir haben also beide ein NP-vollständiges Problem gelöst.*

Um die vorangehenden Überlegungen zu klaren Aussagen zusammenzufassen, benötigen wir die folgende

Definition 5.1 *Bei einem Zahlenproblem (formale Sprache) L assoziieren wir zu einer Eingabeinstanz I neben der Eingabelänge $n(I)$ die maximale Größe $M(I)$ einer der in I*

[4] die Hörer und Hörerinnen der Vorlesung hoffentlich ebenfalls (s. Übung)

gegebenen Zahlparameter. Zu einem festen Polynom p bezeichne L_p die Einschränkung von L auf Eingaben I mit $M(I) \leq p(n(I))$. L heißt pseudopolynomiell entscheidbar*, wenn $L_p \in P$ für jedes Polynom*[5] *p. Ein* pseudopolynomieller *Algorithmus für L ist eine DTM, die das Mitgliedschaftsproblem für L löst und deren Laufzeit polynomiell in $n(I)$ und $M(I)$ beschränkt ist.*

Aus der Existenz eines pseudopolynomiellen Algorithmus für L folgt offensichtlich, dass L pseudopolynomiell entscheidbar ist. Anna's und Bert's Verfahren ergeben also folgendes

Theorem 5.2 *PARTITION und KP sind mit pseudopolynomiellen Algorithmen lösbar.*

Da SUBSET SUM ein Teilproblem von KP ist, überträgt sich dieses Resultat auf SUBSET SUM.

5.2 Erste Beispiele für stark NP-vollständige Probleme

Kann man vielleicht alle Zahlenprobleme pseudopolynomiell lösen? Zum Beispiel auch BP? Oder gibt es Zahlenprobleme, bei denen alle pseudopolynomiellen Lösungsversuche an unüberwindliche Barrieren stoßen?

Die folgende Definition liefert ein wichtiges Werkzeug zur Beantwortung dieser Frage:

Definition 5.3 *Ein Zahlenproblem (formale Sprache) L heißt* stark *NP*-hart*, wenn es ein Polynom p gibt, so dass L_p NP-hart ist. Gilt zusätzlich $L \in NP$, dann heißt L* stark *NP*-vollständig.

Wir merken an, dass rein kombinatorische Probleme (wie zum Beispiel SAT, 3-SAT, CLIQUE, DHC, HC) ohne (echte) Zahlparameter automatisch stark *NP*-vollständig sind, wenn sie *NP*-vollständig sind. Der neue Begriff macht nur Sinn bei echten Zahlenproblemen, deren Eingaben potenziell superpolynomiell in $n(I)$ große Zahlen enthalten können. Das folgende Resultat ist offensichtlich:

Theorem 5.4 *Falls $P \neq NP$, dann kann es zu einem stark NP-vollständigen Problem keinen pseudopolynomiellen Algorithmus geben.*

Ein erstes stark *NP*-vollständiges Zahlenproblem kennen wir schon:

Theorem 5.5 *TSP ist stark NP-vollständig.*

Beweis Die Reduktion von HC auf TSP, die wir zum Nachweis der *NP*-Vollständigkeit von TSP verwendet haben, benutzt in der Distanzmatrix nur Zahlenparameter mit den Werten 1 oder 2. Die Kostenschranke hatte den Wert n (Anzahl der Städte). Es treten also keine Zahlparameter auf, die superpolynomiell in der Länge der Eingabe für TSP sind. **qed.**

Mit TSP kennen wir ein stark *NP*-vollständiges Anordnungsproblem (mit Zahlparametern). Wir begeben uns nun auf die Jagd nach einem stark *NP*-vollständigen Zerlegungsproblem (mit Zahlparametern). Objekt unserer Begierde sind die folgenden Probleme:

[5] wie üblich mit Koeffizienten aus $\mathbb{N}$

3-DM Perfektes 3-Dimensional Matching

Eingabe Drei disjunkte gleichmächtige Mengen $X = \{x_1, \ldots, x_q\}$, $Y = \{y_1, \ldots, y_q\}$, $Z = \{z_1, \ldots, z_q\}$ sowie eine Menge $M \subseteq X \times Y \times Z$ von Tripeln. Hierbei reicht es, M in der Eingabe zu spezifizieren, da X, Y, Z implizit durch die Komponenten der Tripel aus M gegeben sind.

Frage Gibt es eine Auswahl $T \subseteq M$ von q Tripeln aus M, so dass jedes Element aus $X \cup Y \cup Z$ in genau einem Tripel vorkommt?

3-PARTITION Partition in m Tripel mit gleichen Teilsummen

Eingabe $n = 3m$ Zahlen $a_1, \ldots, a_n \in \mathbb{N}$ mit einer Gesamtsumme $B = a_1 + \cdots + a_n$ der Form $B = mb$, $b \in \mathbb{N}$. Für $i = 1, \ldots, n$ gelte $b/4 < a_i < b/2$.

Frage Kann man diese Zahlen in m gleich große Teilsummen zerlegen, d.h., existiert eine Zerlegung von $\{1, \ldots, n\}$ in Mengen $I_1, \ldots, I_m$ mit

$$\forall j = 1, \ldots, m : \sum_{i \in I_j} a_i = b? \tag{3}$$

Beachte, dass wegen $b/4 < a_i < b/2$ die Bedingung (3) höchstens dann erfüllbar ist, wenn jedes I_j dreielementig ist.

4-Partition Partition in m Quadrupel mit gleichen Teilsummen

Eingabe $n = 4m$ Zahlen $a_1, \ldots, a_n \in \mathbb{N}$ mit einer Gesamtsumme $B = a_1 + \cdots + a_n$ der Form $B = mb$, $b \in \mathbb{N}$. Für $i = 1, \ldots, n$ gelte $b/5 < a_i < b/3$.

Frage Kann man diese Zahlen in m gleich große Teilsummen zerlegen, d.h., existiert eine Zerlegung von $\{1, \ldots, n\}$ in Mengen $I_1, \ldots, I_m$ mit

$$\forall j = 1, \ldots, m : \sum_{i \in I_j} a_i = b? \tag{4}$$

Beachte, dass wegen $b/5 < a_i < b/3$ die Bedingung (4) höchstens dann erfüllbar ist, wenn jedes I_j vierelementig ist.

Wir merken kurz an, dass 2-DM (Perfektes 2-Dimensionales Matching) das zu 3-DM analoge Problem mit Paaren anstelle von Tripeln ist. Bei 2-DM geht es dann um die Frage, ob eine Auswahl $T \subseteq M$ von q Paaren aus M existiert, so dass jedes Element in $X \cup Y$ in genau einem Paar vorkommt. Da man sich die Elemente aus X als *Frauen*, die Elemente aus Y als *Männer*, die Paare aus M als *verträgliche Paarbildungen* und die Auswahl $T \subseteq M$ als ein *perfektes Heiratssystem* vorstellen kann, ist 2-DM auch unter dem Namen *Heiratsproblem* bekannt.[6] Mit Maximum Matching Techniken kann man zeigen (s. Vorlesung *Effiziente Algorithmen*):

[6] 3-DM ist das Heiratsproblem für eine Gesellschaft, in welcher sich neben *männlich* und *weiblich* ein dritter Sextypus etabliert hat. Es wird Traditionalisten zutiefst befriedigen, dass (wie sich im Verlaufe dieses Abschnittes noch zeigen wird) 3-DM *NP*-vollständig ist.

Lemma 5.6 *2-DM* $\in P$.

Wir beabsichtigen, die folgende Kette

$$\text{3-SAT} \leq_{pol} \text{3-DM} \leq_{pol} \text{4-PARTITION}_p \tag{5}$$

von Reduktionen zu entwerfen, wobei p ein Polynom ist. Damit hätten wir 4-PARTITION als stark *NP*-vollständiges Zahlenproblem etabliert.

Lemma 5.7 *3-SAT*$\leq_{pol}$*3-DM.*

Beweis Sei $C = (C_0, \ldots, C_{m-1})$ mit $C_i = z_{i1} \vee z_{i2} \vee z_{i3}$ und $z_{ij} \in \{x_1, \bar{x}_1, \ldots, x_n, \bar{x}_n\}$ eine Eingabe für 3-SAT. Wir geben eine Eingabetransformation $C \mapsto M$ an, die C eine Tripelmenge M zuordnet. C soll genau dann erfüllbar sein, wenn wir für M ein perfektes 3-dimensionales Matching $T \subseteq M$ finden. Die Tripelmenge M besteht aus der *Belegungskomponente* M^B, der *Klauselerfüllungskomponente* M^K und der *Ergänzungskomponente* M^E:

Belegungskomponente Zu jeder Variable x_k, $k = 1, \ldots, n$, assoziieren wir die folgenden Tripel:

$$\begin{aligned} T_k &= \{(\bar{x}_{k,i}, a_{k,i}, b_{k,i}) |\ i = 0, \ldots, m-1\} \\ F_k &= \{(x_{k,i}, a_{k,i+1 \bmod m}, b_{k,i}) |\ i = 0, \ldots, m-1\} \end{aligned}$$

Wir setzen $M_k^B = T_k \cup F_k$ und $M^B = \cup_{k=1}^n M_k^B$. Abbildung 2 illustriert diese Konstruktion. Die Elemente $x_{k,i}, \bar{x}_{k,i}$ nennen wir *Literalelemente*. Wir werden sehen, dass sie auch noch in anderen Komponenten vorkommen. Die Elemente $a_{k,i}$ und $b_{k,i}$ sind *interne Elemente* von M_k^B, d.h., sie kommen in keinen Tripeln außerhalb M_k^B vor. Man überlegt sich leicht, dass es zu einer Zerlegung der internen Elemente von M_k^B in Tripel nur zwei Möglichkeiten gibt: **entweder** alle Tripel aus T_k **oder** alle Tripel aus F_k. Intuitiv verknüpfen wir mit der Entscheidung für T_k die Vorstellung, x_k mit 1 zu belegen, und mit der Entscheidung für F_k die Vorstellung, x_k mit 0 zu belegen.

Klauselerfüllungskomponente Zu jeder Klausel C_i, $i = 0, \ldots, m-1$, assoziieren wir die folgenden Tripel:

$$\begin{aligned} S_i^+ &= \{(x_{k,i}, c_i, d_i) |\ 1 \leq k \leq n, x_k \in C_i\} \\ S_i^- &= \{(\bar{x}_{k,i}, c_i, d_i) |\ 1 \leq k \leq n, \bar{x}_k \in C_i\} \end{aligned}$$

Wir setzen $M_i^K = S_i^+ \cup S_i^-$ und $M^K = \cup_{i=0}^{m-1} M_i^K$. Die Elemente c_i und d_i sind *interne Elemente* von M_i^K, d.h., sie kommen in keinen Tripeln außerhalb M_i^K vor. Um diese Elemente in die Tripelzerlegung mit einzubeziehen, sind wir gezwungen, ein Tripel aus M_i^K auszuwählen. Intuitiv verbinden wir damit die Auswahl eines C_i erfüllenden Literals.

Ergänzungskomponente Die Ergänzungskomponente wird dazu dienen, noch unüberdeckte Literalelemente in die Tripelzerlegung einzubeziehen. Die internen Elemente von M^B und M^K werden durch $nm + m$ Tripel exakt überdeckt werden. Damit sind auch bereits $nm + m$ Literalelemente überdeckt. Von den insgesamt $2nm$ Literalelementen wären dann noch $(n-1)m$ unüberdeckt. Dies motiviert die folgende Definition der Ergänzungskomponente:

$$M^E = \{(x_{k,i}, e_l, f_l), (\bar{x}_{k,i}, e_l, f_l | \ 1 \leq k \leq n, 0 \leq i \leq m-1, 1 \leq l \leq (n-1)m\} \ .$$

e_l, f_l sind die *internen Elemente* von M^E.

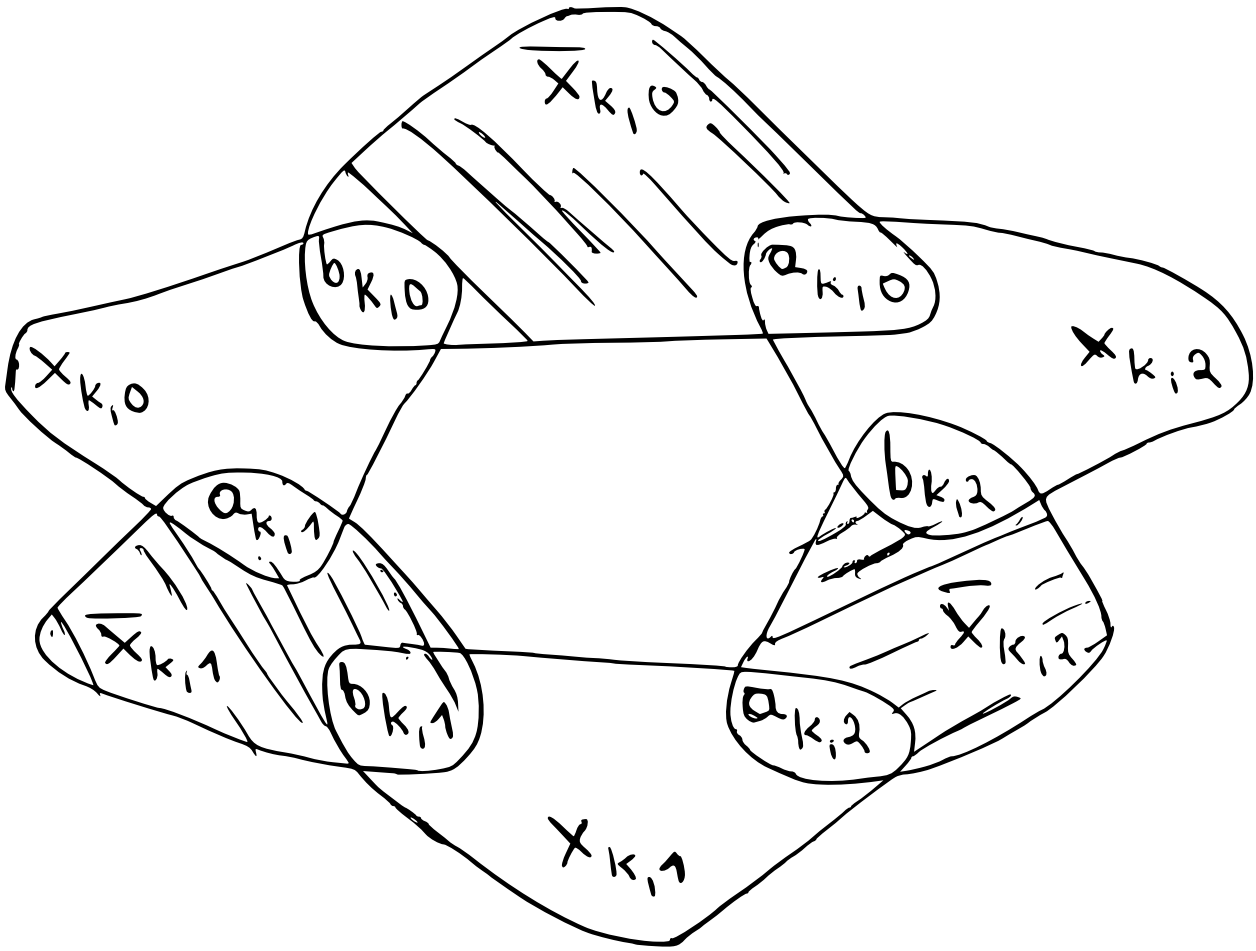

Figure 2: Ein Ausschnitt aus der Belegungskomponente im Falle $m = 3$: die Tripel aus T_k sind schraffiert, die Tripel aus F_k unschraffiert.

Die Transformation $C \mapsto M$ mit $M = M^B \cup M^K \cup M^E$ ist offensichtlich in Polynomialzeit berechenbar. Der Beweis wird abgeschlossen durch die

Behauptung C ist genau dann erfüllbar, wenn für M ein perfektes 3-dimensionales Matching existiert.

Nehmen wir zunächst an, dass eine C erfüllende Belegung gegeben ist. Wir bilden ein perfektes 3-dimensionales Matching T für M nach folgender Strategie:

1. Für $k = 1, \ldots, n$ nimm alle Tripel aus T_k in T auf, falls $x_k = 1$, und nimm alle Tripel aus F_k in T auf, falls $x_k = 0$.
   Effekt Alle internen Elemente von M^B sind exakt überdeckt; ebenso alle Literalelemente $x_{k,i}$ mit $x_k = 0$ bzw. alle Literalelemente $\bar{x}_{k,i}$ mit $x_k = 1$. Die anderen Literalelemente (die zu erfüllten Literalen korrespondieren) sind noch unüberdeckt.

2. Fixiere zu jeder Klausel C_i, $0 \leq i \leq m-1$, ein erfülltes Literal $z_{i,j}$. Es existiert ein k, so dass $z_{i,j} \in \{x_k, \bar{x}_k\}$. Falls $z_{i,j} = x_k$, dann nimm das Tripel $(x_{k,i}, c_i, d_i) \in S_i^+$ in T auf. Falls $z_{i,j} = \bar{x}_k$, dann nimm das Tripel $(\bar{x}_{k,i}, c_i, d_i) \in S_i^-$ in T auf. Beachte, dass in beiden Fällen ein noch unüberdecktes Literalelement überdeckt wurde.
   Effekt Alle internen Elemente von M^K sind exakt überdeckt. Insgesamt $nm + m$ Literalelemente sind nun exakt überdeckt. Somit sind $(n-1)m$ Literalelemente noch unüberdeckt.

3. Benutze — in der offensichtlichen Weise — $(n-1)m$ Tripel aus M^E, um die noch unüberdeckten Literalelemente sowie die internen Elemente von M^E exakt zu überdecken.

Nehmen wir nun umgekehrt an, dass ein perfektes 3-dimensionales Matching T für M gegeben ist. Wie bereits oben angemerkt, muss T für jedes k entweder alle Tripel aus T_k oder alle Tripel aus F_k enthalten. Im ersten Fall setzen wir $x_k = 1$, im zweiten Fall $x_k = 0$. Wir zeigen, dass die so erhaltene Belegung in jeder Klausel mindestens 1 Literal erfüllt. Die Tripel in $T \cap M^B$ lassen genau die Literalelemente unüberdeckt, die zu erfüllten Literalen korrespondieren. Da die Tripel aus $T \cap M^K$ alle internen Elemente von M^K überdecken, muss es zu jeder Klausel ein erfülltes Literal geben: die erste Komponente des c_i, d_i überdeckenden Tripels ist nämlich ein von $T \cap M^B$ noch unüberdecktes Literalelement, das zu einem C_i erfüllenden Literal korrespondiert. **qed.**

Lemma 5.8 *Es existiert ein Polynom p mit 3-DM$\leq_{pol}$4-PARTITION$_p$.*

Beweis Sei $M \subseteq X \times Y \times Z$ mit $X = \{x_1, \ldots, x_q\}$, $Y = \{y_1, \ldots, y_q\}$ und $Z = \{z_1, \ldots, z_q\}$ eine gegebene Tripelmenge. Zu jedem $w \in W = X \cup Y \cup Z$ bezeichne $L(w)$ die Anzahl der Tripel, in denen w vorkommt. Sei weiter $m = |M|$ und $r = 32q$. Wir können oBdA $m \geq q$ voraussetzen, da kein perfektes 3-dimensionale Matching für M mit weniger als q Tripeln existiert. Die zu M korrespondierende Eingabeinstanz für 4-PARTITION bestehe aus den folgenden Zahlen:

Elementzahlen Zu jedem $w \in W$ assoziieren wir die *Hauptzahl* $a_1(w)$ und die *Nebenzahlen*

$a_l(w)$ gemäß folgender Definition:

$$\begin{aligned}
a_1(x_i) &= 10r^4 + ir + 1,\ 1 \le i \le q \\
a_l(x_i) &= 11r^4 + ir + 1,\ 1 \le i \le q,\ 2 \le l \le L(x_i) \\
a_1(y_j) &= 10r^4 + jr^2 + 2,\ 1 \le j \le q \\
a_l(y_j) &= 11r^4 + jr^2 + 2,\ 1 \le j \le q,\ 2 \le l \le L(y_j) \\
a_1(z_k) &= 10r^4 + kr^3 + 4,\ 1 \le k \le q \\
a_l(z_k) &= 8r^4 + kr^3 + 4,\ 1 \le k \le q,\ 2 \le l \le L(z_k)
\end{aligned}$$

Tripelzahlen Zu jedem $(x_i, y_j, z_k) \in M$ assoziieren wir die Zahl

$$a(x_i, y_j, z_k) = 10r^4 - kr^3 - jr^2 - ir + 8.$$

Wir erhalten insgesamt eine Menge $\mathcal{A}$ bestehend aus $\sum_{w \in W} L(w) = 3m$ Elementzahlen und m Tripelzahlen. Es wird sich weiter unten implizit ergeben, dass die Gesamtsumme aller $4m$ Zahlen genau mb beträgt, wobei

$$b = 40r^4 + 15.$$

Man verifiziert leicht, dass jede Zahl aus $\mathcal{A}$ größer als $b/5$ und kleiner als $b/3$ ist. (Daher können nur Zahlquadrupel aus $\mathcal{A}$ die Teilsumme b ergeben.) Die Transformation $M \mapsto \mathcal{A}$ ist offensichtlich in Polynomialzeit berechenbar. Da alle Zahlen in $\mathcal{A}$ kleiner als $12r^4 = 12 \cdot (32q)^4 \le 12 \cdot (32m)^4 \le 12 \cdot (8|\mathcal{A}|)^4$ sind, ist der maximale Zahlparameter in $\mathcal{A}$ polynomiell in Eingabelänge $n(\mathcal{A}) \ge |\mathcal{A}|$ beschränkt. Der Beweis wird daher abgeschlossen durch folgende **Behauptung** Es existiert genau dann ein perfektes 3-dimensionales Matching für M, wenn $\mathcal{A}$ sich in m Quadrupel (vierelementige Teilmengen) zerlegen lässt, so dass jedes Quadrupel die Teilsumme b liefert.

Nehmen wir zunächst an, dass ein perfektes 3-dimensionales Matching T für M vorliegt. Wir bilden für jedes der m Tripel aus M ein Zahlenquadrupel nach der folgenden Strategie:

Fall 1 Sei $(x_i, y_j, z_k) \in T$. Dann bildet $a(x_i, y_j, z_k)$ zusammen mit $a_1(x_i), a_1(y_j), a_1(z_k)$ ein Quadrupel. Die Summe dieser vier Zahlen ist offensichtlich b.

Fall 2 Sei $(x_i, y_j, z_k) \in M \setminus T$. Dann bildet $a(x_i, y_j, z_k)$ zusammen mit $a_{l_1}(x_i), a_{l_2}(y_j), a_{l_3}(z_k)$ ein Quadrupel. Hierbei werden $l_1, l_2, l_3 \ge 2$ so gewählt, dass keine Nebenzahl mehrfach verwendet wird. (Der Vorrat an Nebenzahlen ist genau passend bemessen.) Die Summe dieser vier Zahlen ist wiederum b.

Nehmen wir jetzt umgekehrt an, dass eine erfolgreiche Zerlegung von $\mathcal{A}$ in Zahlenquadrupel vorgegeben ist. Sei (A_1, A_2, A_3, A) eines dieser Quadrupel. Aus $A_1 + A_2 + A_3 + A = b$ folgt

$$(A_1 + A_2 + A_3 + A \bmod r) = (b \bmod r) = 15.$$

Diese Gleichung kann nur eingehalten werden, wenn Indizes $i, j, k, i', j', k', l_1, l_2, l_3$ existieren, so dass die Komponenten des Quadrupels die Form

$$A = a(x_i, y_j, z_k),\ A_1 = a_{l_1}(x_{i'}),\ A_2 = a_{l_2}(y_{j'}),\ A_3 = a_{l_3}(z_{k'})$$

haben. Die Rechnung modulo r^2 ergibt dann

$$(i'r + 1) + 2 + 4 + (-ir + 8) = (A_1 + A_2 + A_3 + A \bmod r^2) = (b \bmod r^2) = 15.$$

Hieraus können wir $i' = i$ ableiten. Die Rechnung modulo r^3 liefert

$$(ir + 1) + (j'r^2 + 2) + 4 + (-jr^2 - ir + 8) = (A_1 + A_2 + A_3 + A \bmod r^3) = (b \bmod r^3) = 15.$$

Hieraus können wir $j' = j$ ableiten. Die Rechnung modulo r^4 ergibt

$$\begin{aligned}(ir + 1) + (jr^2 + 2) + (k'r^3 + 4) + (-kr^3 - jr^2 - ir + 8) \quad &= \\ (A_1 + A_2 + A_3 + A \bmod r^3) = (b \bmod r^3) &= 15.\end{aligned}$$

Es folgt $k' = k$. Der Koeffizient von r^4 bei der Summenzahl $b = 40r^4 + 15$ hat den Wert 40. Die Summe von A_1, A_2, A_3, A muss daher auch Koeffizient 40 vor dem Term r^4 liefern. Eine Inspektion der Elementzahlen ergibt, dass A_1, A_2, A_3 **entweder** drei Hauptzahlen **oder** drei Nebenzahlen sein müssen. (Im ersten Fall ergibt sich Koeffizient 40 als eine Summe der Form $10 + 10 + 10 + 10$, im zweiten als Summe der Form $11 + 11 + 8 + 10$.) Man überzeugt sich nun leicht, dass die Hauptzahl-Quadrupel ein perfektes 3-dimensionales Matching für M repräsentieren. **qed.**

Folgerung 5.9 *4-PARTITION ist stark NP-vollständig.*

5.3 Weitere stark NP-vollständige Probleme

NP-Härte vererbt sich entlang von Karp-Reduktionsketten. Reduktionen, die starke *NP*-Härte erhalten, sind der Gegenstand der folgenden

Definition 5.10 *Seien L, L' Zahlenprobleme. Eine Karp-Reduktion von L nach L' mit Reduktionsabbildung f heißt* pseudopolynomiell, *falls ein Polynom q existiert mit $M(f(I)) \leq q(M(I))$. In Worten: wenn das Maximum der Zahlparameter in Eingabe I durch $M(I)$ beschränkt ist, dann ist das Maximum der Zahlparameter in der transformierten Eingabe $I' = f(I)$ durch $q(M(I))$ beschränkt.*

Das folgende Resultat ist offensichtlich:

Theorem 5.11 *Seien L und L' Zahlenprobleme und sei L pseudopolynomiell Karp-reduzierbar auf L'. Falls ein pseudopolynomieller Algorithmus für L' existiert, dann existiert auch ein pseudopolynomieller Algorithmus für L. Falls andererseits L stark NP-hart ist, dann ist auch L' stark NP-hart.*

Mit Hilfe pseudopolynomieller Reduktionen können wir ausgehend von 4-PARTITION weitere stark *NP*-vollständige Zahlenprobleme ausfindig machen:

Lemma 5.12 *(ohne Beweis)*
4-PARTITION ist pseudopolynomiell Karp-reduzierbar auf 3-PARTITION.

Lemma 5.13 *4-PARTITION und 3-PARTITION sind pseudopolynomiell Karp-reduzierbar auf BP.*

Beweis Wir führen den Beweis für 3-PARTITION. (Der Beweis für 4-PARTITION ergibt sich mit einem analogen Argument.) Es handelt sich um eine Transformation mit Spezialisierung: BP ist das Teilproblem von 3-PARTITION, bei welchem die Binkapazität auf b und die Kostenschranke (Schranke für die erlaubte Anzahl von Bins) auf m gesetzt wird. m Bins der Kapazität b reichen nämlich genau dann aus, wenn die gegebene Zahlenkollektion (mit Gesamtsumme mb) sich in m Tripel mit Teilsumme b zerlegen lässt. **qed.**

Folgerung 5.14 *3-PARTITION und BP sind stark NP-vollständig.*

Wir bemerken abschließend, dass 3-PARTITION für pseudopolynomielle Karp-Reduktionen ein ähnlich beliebtes Quellproblem ist wie 3-SAT für (gewöhnliche) Karp-Reduktionen.