2. Runde

des

42. Bundeswettbewerbs Informatik

Aufgabe 2: Stilvolle Paeckchen

Bearbeitet

von

Florian Bange
Teilnahme-ID: 71639

15. April 2024

Inhaltsverzeichnis

1. Einleitung

Dieses Dokument beinhaltet meine Dokumentation der zweiten Aufgabe[1] der zweiten Runde des 42. Bundeswettbewerbs Informatik[2] in den Jahren 2023 und 2024.

In dieser Aufgabe ist eine Menge an Kleidungsstuecken gegeben, wobei jedes dieser genau einer Sorte (z.B. »Hose«) und einer Stilrichtung (z.B. »sportlich«) zugeordnet ist. Dafuer wird fuer jede Kombinationen die Anzahl a_{ij} der Kleidungsstuecke der i-ten Sorte mit der j-ten Stilrichtung angegeben. Weiter existieren Einschraenkung fuer moegliche Kombinationen verschiedener Stilrichtungen in der Form »x ist kombinierbar mit y«. Fuer die Kleidungsstuecke soll eine Verteilung auf beliebig viele Boxen gefunden, sodass die insgesamt Anzahl der uebrigen Kleidungsstuecke minimiert wird. Dabei soll gelten, dass jede Box mindestens eine und maximal drei Kleidungsstuecke pro Sorte enthaelt und die Stilrichtungen der Kleidungsstuecke in der Box paarweise miteinander kombinierbar sind.

Dieses Optimierungsproblem wird in Kapitel 2 und Kapitel 3 formalisiert. Dazu werden Matrizen verwendet, deren Eintraege in der i-ten Zeile und j-ten Spalte die Anzahl der i-ten Sorte in der j-ten Stilrichtung beinhalten, sodass Boxen vollstaendigen durch Matrizen dargestellt werden koennen. Analog ist die Insegsamtanzahl der Kleidungsstuecke pro Paar an Sorte und Stilrichtung durch die Matrix (a_{ij}) gegeben. Dadurch laesst sich das Problem als Gleichung von Matrizen mit bestimmten Einschraenkung fuer die Boxen formalisieren. Insebesondere muessen dann die Zeilen der Matrizen, die die Boxen darstellen in Summe zwischen 1 und 3 (inklusive) liegen. Weiter muessen die Eintraege der Matrizen der Boxen zu der Kombinierbarkeit passen, welche durch eine Relation auf der Menge der Stilrichtungen dargestellt wird. Die Beschraenkung der Summe der Zeilen der Boxen wird dabei in der Erweiterung I sofort auf beliebige Schranken verallgemeinert. Daraufhin wird Erweiterung II definiert, bei welcher zusaetzlich zu Sorte und Stil auch die Groesze der Kleidung beachtet wird. Zuletzt wird Erweiterung III definiert. Bei dieser geht es darum, die Anzahl der genutzten Boxen zu minimieren, waehrend weiterhin die Minimalanzahl an uebrigen Kleidungsstuecken gewaehrleistet wird und eine Maximalanzahl an Boxen zu gewaehrleisten.

Diese Formalisierung ermoeglicht eine Betrachtung der Komplexitaet des Problems in Kapitel 4. Dort wird gezeigt, dass es sich um ein NP-schweres Problem handelt. Diese Tatsache folgt dann auch sofort fuer alle Erweiterungen des Problems.
Anschlieszend werden in Kapitel 5 Loesungsvorschlaege des Problem gegeben. Als erstes wird ein Ganzzahliges-Lineares-Programm (ILP) als Loesung vorgestellt, welches das Problem der Erweiterung I optimal loesen soll. Danach werden maximale Cliquen in der Kombinierbarkeitsrelation disskutiert. Dabei handelt es sich um Teilmengen der Stilrichtungen, sodass alle Stilrichtung der Teilmenge sich paarweise kombinieren lassen. Hat man diese bereits berechnet so, laesst sich die Loesung des Problem sehr stark vereinfachen. Mit Hilfe dieser Vereinfachung werden anschlieszend ein Loesungsverfahren vorgestellt, welches das als erstes behandelte ILP deutlich verbessert. Inbsesondere die Beispieleingaben der BwInf-Website werden durch das letzte Loesungsverfahren sehr schnell optimal geloest. Weiter wird ein itereatives Greedy-Verfahren vorgestellt, das keine exakten Loesung ermoeglicht, dafuer aber bezueglich der Platz- und Laufzeitkomplexitaet sehr gut ist. Fuer alle vorgestellten Loesungen, sowie fuer die Berechnung der Cliquen wird anschlieszend eine Analyse der Zeitkomplexitaet vorgenommen.

In den darauffolgenden Kapiteln werden die in Kapitel 3 definierten Erweiterungen geloest. In Kapitel 6 wird die zusaetzliche Einschraenkung der Kombinierbarkeit in Form von Groeszen der Kleidung behandelt. Dabei wird ebenfalls eine ILP-Loesung und eine Loesung nach dem Greedy-Verfahren vorgestellt, welche je analog zu denen des Kapitels 5 funktionieren. Daraufhin wird in Kapitel 7 die Erweiterung III behandelt. Diese laesst sich sehr einfach durch die vorherigen Kapitel loesen.//
In Kapitel 8 wird kurz auf die konkreten implementierungen der Loesungsvorschlaege eingegangen. Diese wurden in C++ 20 und Python 3 implementiert. Die wichtigsten Teile des dazugehoerigen Quellcodes werden in Kapitel 10 angegeben und erklaert. Weiter werden in Kapitel 9 die Beispiele der BwInf-Website zu den vorgestellten Problemen geloest. Leider konnte die Erweiterung II nicht mehr rechtzeitig implementiert werden, weshalb es dazu keinen Quellcode und keine Beispiele gibt.

[1] Siehe `https://bwinf.de/fileadmin/bundeswettbewerb/42/aufgaben422.pdf`
[2] Siehe `https://bwinf.de/bundeswettbewerb/42/2/`

2. Mathematische Grundlagen

Im Folgenden sollen die mathematischen Grundlagen gegeben werden, sodass im Anschluss das gegebene Problem definiert werden kann. Dazu werden einige Definitionen zu Mengen, Relationen und Matrizen gegeben.

Definition 1: Mengenoperationen

Es seien $A,\ B$ Mengen nach Cantor. Dann wird fuer den mengentheoretische Schnitt $A \cap B$, fuer die Vereinigung $A \cup B$ und fuer die Differenz $A \setminus B$ geschrieben.

Definition 2: Kardinalitaet

Die Kardinalitaet oder Maechtigkeit einer Menge X wird mit $\#(X)$ bezeichnet.

Definition 3: Relationen

Gegeben sei eine Menge X. Dann ist eine Relation R auf X eine Teilmenge des kartesischen Produkts

$$X \times X \coloneqq \{(x,\ y) : x, y \in X\}$$

D.h., $R \subseteq X \times X$. Dabei wird $(x,\ y)$ als Tupel bezeichnet.

Definition 4: Eigenschaften von Relationen

Es sei $R \subseteq X \times X$ eine Relation auf einer Menge X.

1. Reflexivitaet: Fuer alle $x \in X$ gilt: $(x,\ x) \in R$
2. Symetrie: Fuer alle $x,\ y \in X$ gilt: $(x,\ y) \in R \implies (y,\ x) \in R$
3. Transitivitaet: Fuer alle $x,\ y,\ z \in X$ gilt: $(x,\ y) \in R$ und $(y,\ z) \in R \implies (x,\ z) \in R$

Eine Relation $R \subseteq X \times X$ auf einer Menge X wird Aequivalenzrelation genannt, wenn sie reflexiv, symetrisch und transitiv ist.

Definition 5: Matrizen

Fuer $m, n \in \mathbb{N}$ und eine Menge X bezeichnet $\mathrm{Mat}(m,\ n;\ X)$ die Menge der Matrizen mit m Zeilen und n Spalten und Eintraegen aus der Menge X. Fuer eine Matrix $A \in \mathrm{Mat}(m,\ n;\ X)$ wird der Eintrag in der i-ten Zeile und j-ten Spalte fuer $i \in \{1,\ \dots\ ,\ m\}$ und $j \in \{1,\ \dots\ ,\ n\}$ mit $a_{ij} \in X$ bezeichnet. Dann schreibt man auch $A = (a_{ij})$. Weiter schreibt man

$$A = \begin{pmatrix} a_{1\ 1} & a_{1\ 2} & \cdots & a_{1\ n-1} & a_{1\ n} \\ \vdots & \vdots & & \vdots & \vdots \\ a_{m\ 1} & a_{m\ 2} & \cdots & a_{m\ n-1} & a_{m\ n} \end{pmatrix}$$

Zuletzt wird mit $\#(A)$ die Summe der gesamten Eintrage der Matrix bezeichnet werden. Also

$$\#(A) \coloneqq \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}$$

Definition 6: Zeilen und Spalten von Matrizen

Fuer eine Matrix $A \in \mathrm{Mat}(m,\ n;\ X)$ wird die i-te Zeile durch A^i bezeichnet. Dabei handelt es sich um ein n-Tupel. Analog wird die j-te Spalte durch A_j bezeichnet. Wobei es sich um einen Spaltenvektor mit m Eintraegen handelt. Zuletzt soll mit $\#(A^i)$ und $\#(A_j)$ die Summe der Eintraege der i-ten Zeile, bzw. der j-ten Spalte bezeichnet werden. D.h.,

$$\#(A^i) \coloneqq \sum_{j=1}^{n} a_{ij} \text{ und } \#(A_j) \coloneqq \sum_{i=1}^{m} a_{ij}$$

Definition 7: Matrizoperationen

Fuer Matrizen $A,\ B \in \mathrm{Mat}(m,\ n;\ X)$ wird die Summe $A + B \in \mathrm{Mat}(m,\ n;\ X)$ und die Differenz $A - B \in \mathrm{Mat}(m,\ n;\ X)$ elementweise definiert. Dabei ist zu bemerken, dass $(X,\ +)$ dazu eine Gruppe bilden muesste, damit die Differenz und Addition wohldefiniert ist. Im Folgenden wird die Gruppe $(\mathbb{Z},\ +)$ verwendet. Weiter sind die Relationen $>,\ <,\ \geq,\ \leq$ und $=$ auch elementweise zu verstehen.

3. Definition des Problems und der Erweiterungen I - III

3.1. Originales Problem

Mit Hilfe der soeben gegebenen mathematischen Grundlagen soll nun das gegebe Problem durch Matrizen und Relationen formal definiert werden. Dazu wird zunaechst die Probleminstanz durch die Menge der Sorten, die Menge der Stilrichtungen, einer Relation auf den Stilrichtungen und einer Matrix, die die verfuegbaren Kleidungsstuecke darstellt, definiert. Daraufhin wird das Problem mit Hilfe einer Matrizengleichung definiert, wobei auch die Boxen als Matrizen dargestellt werden, die bestimmte Eigenschaften erfuellen sollen.
Anschlieszend soll sofort die erste Erweiterung eingefuehrt werden. Bei dieser wird die Einschraenkung der Anzahl der Kleidungsstuecke in einer Box als Teil der Eingabe definiert.

Definition 8: Probleminstanz

Eine Probleminstanz (fuer das Problem der Aufgabenstellung) besteht aus $m,\ n \in \mathbb{N}_0$, den Mengen $S = \{s_1,\ s_2,\ \dots\ ,\ s_m\}$ der m Sorten und $R = \{r_1,\ r_2,\ \dots\ ,\ r_n\}$ der n Stilrichtungen, wobei $\#(S) = m$ und $\#(R) = n$; weiter einer reflexiven und symetrischen Relation $K \subseteq R \times R$, sodass

$$(x,\ y) \in K \iff x \text{ und } y \text{ sind kombinierbar}$$

und zuletzt einer Matrix $A = (a_{ij}) \in \mathrm{Mat}(m,\ n;\ \mathbb{Z})$ mit $A \geq 0$.

Bemerkung 9:

Die Matrix A stellt die Anzahl der Kleidungsstuecke der Sorten und der Stilrichtungen dar, sodass der Eintrag a_{ij} die Anzahl der Kleidungsstuecke der Sorte s_i in Stilrichtungen r_j angibt.

Bemerkung 10:

Die Relation $K \subseteq R \times R$ ist reflexiv und symetisch, weil

1. jede Stilrichtung $r \in R$ mit sich selbst kombinierbar sein soll, und
2. fuer alle $x,\ y \in R$ gelten soll, dass wenn x mit y kombinierbar ist, auch y mit x kombinierbar ist.

Definition 11: Problem 0 (P-0)

Fuer eine gegebene Probleminstanz nach Definition 8 werden $k \in \mathbb{N}$ und $X_1,\ \dots\ ,\ X_k \in \mathrm{Mat}(m,\ n;\ \mathbb{Z})$ mit $X_i \geq 0$ fuer alle i gesucht, sodass fuer

$$L \coloneqq A - (X_1 + \cdots + X_k)$$

$L \geq 0$ gilt und die Summe $\#(L)$ der Eintrage der Matrix L minimal ist. Dabei soll jede Matrix X der k Matrizen $X_1,\ \dots\ ,\ X_k$ diese Eigenschaften erfuellen:

(B1) Fuer jedes $i \in \{1,\ \dots\ ,\ m\}$ gilt, dass $1 \leq \#(X^i) \leq 3$

(B2) Fuer die Menge $I \coloneqq \left\{ j \in \{1,\ \dots\ ,\ n\} : \#(X_j) \geq 1 \right\}$ gilt, dass $\forall i, j \in I\ : (r_i,\ r_j) \in K$.

Bemerkung 12: Boxen

Analog zur Matrix A wird eine Box durch eine Matrix $X \in \mathrm{Mat}(m,\ n;\ \mathbb{Z})$ mit $X \geq 0$ gegeben, wobei x_{ij} die Anzahl der Kleidungsstuecke der Sorte s_i in Stilrichtungen r_j in der Box darstellt.
Weiter beschreibt die Eigenschaft (B1), die Minimal- und Maximalanzahl der Kleidungsstuecke pro Sorte. Denn in der Matrixdarstellung der Boxen beinhaltet die i-te Zeile X^i der Matrix die Kleidungsstuecke der i-ten Sorte. Somit soll fuer jede der m Zeilen die Summe $\#(X^i)$ der Eintrage dieser Zeile die Ungleichung $1 \leq \#(X^i) \leq 3$ erfuellen. D.h., dass die Summe der Anzahl der Kleidungsstuecke der i-ten Sorte mindestens 1 und maximal 3 ist.
Weiter beschreibt die Eigenschaft (B2) die Kombinierbarkeit der benutzen Stilrichtungen einer Box. Dazu wird die Menge I der Indizes j definiert, sodass $\#(X_j) \geq 1$. D.h., dass die Summe der Elemente der j-ten Spalte groeszer oder gleich 1 ist. Da die Spalten der Matrixdarstellung der Boxen die benutzten Stilrichtungen beschreiben, sind dies die Indizes der Stilrichtungen, die in der Box verwendet wurden. Somit soll dann gelten, dass alle Stilrichtung r_j eines Indizes $j \in I$ paarweise kombinierbar sind. Also $(r_i,\ r_j) \in K$ fuer alle $i,\ j \in I$.

3.2. Erweiterung I: Variable Mindest- und Maximalanzahl

Marius, Emily und Merle haben nach einigen Anlaufen erkannt, dass es etwas langweilig ist, wenn pro Box nur minimal ein und maximal drei Kleidungsstuecke pro Sorte vorhanden sind. Deswegen beschlieszen sie, diese Schranken zu veraendern. Nun stellt sich allerdings die Frage, welche Schranken genommen werden sollen. Da sie sich noch nicht einigen konnten, soll nun das bereits behandelte Problem mit beliebige Schranken geloest werden. Im Folgenden seien diese $B,\ C \in \mathbb{N}$ mit $1 \leq B \leq C$.
Die Probleminstanz nach Definition 8 wird dadurch um B und C erweitert. Weiter muss die Defintion des Problems aus Definition 11 ausschlieszlich in der Eigenschaft (B1) der Boxen $X_1,\ \dots\ ,\ X_k$ geaendert werden:

Definition 13: Problem I (P-I)

Fuer eine gegebene Probleminstanz nach Definition 8 und $B,\ C \in \mathbb{N}$ mit $1 \leq B \leq C$ werden $k \in \mathbb{N}$ und $X_1,\ \dots\ ,\ X_k \in \mathrm{Mat}(m,\ n;\ \mathbb{Z})$ mit $X_i \geq 0$ fuer alle i gesucht, sodass fuer

$$L \coloneqq A - (X_1 + \cdots + X_k)$$

$L \geq 0$ gilt und die Summe $\#(L)$ der Eintrage der Matrix L minimal ist. Dabei soll jede Matrix X der k Matrizen $X_1,\ \dots\ ,\ X_k$ diese Eigenschaften erfuellen:

(B1) Fuer jedes $i \in \{1,\ \dots\ ,\ m\}$ gilt, dass $B \leq \#(X^i) \leq C$

(B2) Fuer die Menge $I \coloneqq \left\{j \in \{1,\ \dots\ ,\ n\} : \#(X_j) \geq 1\right\}$ gilt, dass $\forall i, j \in I\ : (r_i,\ r_j) \in K$.

Beispiel 14:

Es sei eine Probleminstanz gegeben durch $m = 2$ und $n = 3$ mit $S = \{s_1,\ s_2\}$ und $R = \{r_1,\ r_2,\ r_3\}$, sowie

$$A = \begin{pmatrix} 2 & 4 & 2 \\ 3 & 2 & 1 \end{pmatrix}$$

D.h., es existieren zwei Sorten an Kleidungsstuecken und drei Stilrichtung und beispielsweise vier Kleidungsstuecke der Sorte s_1 in Stilrichtung r_2. Weiter sei die Relation K gegeben durch

$$K = \left\{(r_1,\ r_1),\ (r_2,\ r_2),\ (r_3,\ r_3),\ (r_1,\ r_2),\ (r_2,\ r_1),\ (r_1,\ r_3), (r_3,\ r_1)\right\}$$

Demnach sind die Stilrichtungen r_1, r_2 und r_3 je mit sich selbst kombinierbar und je r_1 und r_2, sowie r_1 und r_3 sind miteinander kombinierbar. Dann laesst sich das Problem mit der gegebenen Probleminstanz optimal loesen. Denn fuer $k = 3$ und die Matrizen $X_1,\ X_2,\ X_3 \in \mathrm{Mat}(2,\ 3;\ \mathbb{Z})$ mit

$$X_1 = \begin{pmatrix} 1 & 2 & 0 \\ 2 & 1 & 0 \end{pmatrix},\ X_2 = \begin{pmatrix} 0 & 2 & 0 \\ 0 & 1 & 0 \end{pmatrix},\ \text{und } X_3 = \begin{pmatrix} 1 & 0 & 2 \\ 1 & 0 & 1 \end{pmatrix}$$

gilt

$$X_1 + X_2 + X_3 = \begin{pmatrix} 1 & 2 & 0 \\ 2 & 1 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 2 & 0 \\ 0 & 1 & 0 \end{pmatrix} + \begin{pmatrix} 1 & 0 & 2 \\ 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 4 & 2 \\ 3 & 2 & 1 \end{pmatrix},$$

sodass

$$L = A - (X_1 + X_2 + X_3) = \begin{pmatrix} 2 & 4 & 2 \\ 3 & 2 & 1 \end{pmatrix} - \begin{pmatrix} 2 & 4 & 2 \\ 3 & 2 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Da stets $L \geq 0$ gelten soll, ist L hier minimal. Weiter erfuellen die Matrizen X_1, X_2 und X_3 die Bedingungen (B1) und (B2), denn die Summe der Zeilen ist stets 1, 2 oder 3 und die benutzten Spalten sind stets kombinierbar. Weiter gilt $X_1,\ X_2,\ X_3 \geq 0$.

3.3. Erweiterung II: Betrachtung von Kleidergroeszen

Nach kurzer Zeit ist den Dreien [3] aufgefallen, dass es sinnvoll ist, die Groesze der Kleidungsstueck zu beachten. Denn unpraktischer Weise koennen die meisten Erwachsenen nur sehr wenig mit einer Kinderhose in Groesze 140 anfangen. Somit beschlieszen sie, dass die Groeszen der, in einer Box genutzten Kleidungsstuecke, zusammen passen sollten. Dazu wird jedem Kleidungsstueck neben Stilrichtung und Typ auch eine Groesze zu geordnet. Dabei sei $T \in \mathbb{N}$ die maximale Groesze, sodass jedes Kleidungsstueck eine Grosze $t \in \{1,\ 2,\ \dots\ T\}$ zugeordnet werden kann. Weiter sei $r \in \mathbb{N}$ mit $r \leq T$ der Radius der Groszen in den Boxen, sodass fuer jede Box die benutzten Kleidungsstuecke eine absolute Differenz kleiner oder gleich r haben.
Fuer eine mathematische Defintion des Problems wird die bekannte Probleminstanz nach Definition 8 um $T \in \mathbb{N}$ ergaenzt. Weiter werden statt einer Matrix $A \in \mathrm{Mat}(m,\ n;\ \mathbb{Z})$, T Matrizen $A_1, A_2,\ \dots\ ,\ A_T \in \mathrm{Mat}(m,\ n;\ \mathbb{Z})$ angegeben, wobei A_t fuer $t \in \{1,\ 2,\ \dots\ T\}$ die Anzahl der vorhandenen Kleidungsstuecke der Groesze t nach Stilrichtung und Sorte angibt:

Definition 15: Problem II (P-II)

Gegeben sei nun eine Probleminstanz bestehend aus $m,\ n \in \mathbb{N}_0$, den Mengen $S = \{s_1,\ s_2,\ \dots\ ,\ s_m\}$ der m Sorten und $R = \{r_1,\ r_2,\ \dots\ ,\ r_n\}$ der n Stilrichtungen, wobei $\#(S) = m$ und $\#(R) = n$; weiter eine reflexive und symetrische Relation $K \subseteq R \times R$, sodass

$$(x,\ y) \in K \iff x \text{ und } y \text{ sind kombinierbar}$$

und zuletzt $B,\ C \in \mathbb{N}$ mit $1 \leq B \leq C$, die maximale Groesze $T \in \mathbb{N}$ der Radius $r \in \mathbb{N}$ mit $r \leq T$ und Matrizen $A_1,\ \dots\ ,\ A_T \in \mathrm{Mat}(m,\ n;\ \mathbb{Z})$ mit $A_t \geq 0$ fuer alle $t \in \{1,\ 2,\ \dots\ T\}$. Nun werden $k \in \mathbb{N}$ Boxen der Form $(Z_1^{(i)},\ \dots\ ,\ Z_T^{(i)})$ fuer $i \in \{1,\ \dots\ ,\ k\}$ mit $Z_1^{(i)},\ \dots\ ,\ Z_T^{(i)} \in \mathrm{Mat}(m,\ n;\ \mathbb{Z})$ und $Z_t^{(i)} \geq 0$ fuer alle $t \in \{1,\ 2,\ \dots\ T\}$ gesucht. Dabei sei

$$Z_t \coloneqq Z_t^{(1)} + \dots + Z_t^{(k)}$$

fuer alle $t \in \{1,\ 2,\ \dots\ T\}$. Dann soll fuer alle $t \in \{1,\ 2,\ \dots\ T\}$ mit

$$L_t \coloneqq A_t - Z_t$$

gelten, dass $L_t \geq 0$ und fuer

$$L \coloneqq L_1 + \dots + L_T$$

soll $L \geq 0$ minimiert werden. Weiter sei

$$X_i \coloneqq Z_1^{(i)} + \dots + Z_T^{(i)}$$

fuer alle $i \in \{1,\ \dots\ ,\ k\}$. Dann soll jede Matrix X der k Matrizen $X_1,\ \dots\ ,\ X_k$ diese Eigenschaften erfuellen:

(B1) Fuer jedes $i \in \{1,\ \dots\ ,\ m\}$ gilt, dass $B \leq \#(X^i) \leq C$

(B2) Fuer die Menge $I \coloneqq \left\{j \in \{1,\ \dots\ ,\ n\} : \#(X_j) \geq 1\right\}$ gilt, dass $\forall i, j \in I\ : (r_i,\ r_j) \in K$.

Als letzte Bedingung soll fuer alle $i \in \{1,\ \dots\ ,\ k\}$ gelten:

(B3) Es sind maximal r aufeinanderfolgende Matrizen der Matrizen $Z_1^{(i)},\ \dots\ ,\ Z_T^{(i)}$ nicht null:
Es existiert ein Index $j\ \in \{1,\ 2,\ \dots\ T\}$, mit

$$Z_1^i = \dots = Z_{j-1}^i = Z_{j+r}^i = \dots = Z_T^i = 0$$

3.4. Erweiterung III: Minimierung der verwendeten Boxen

Nachdem Marius, Emily und Merle nun fuer ihre verschiedenen Probleme Loesungen erhalten haben, ist ihnen ein weiteres Problem aufgefallen. Denn fuer sie ist es sehr wichtig, in jeder Hinsicht nachhaltig zu handeln. Deswegen beschlieszen sie, moeglichst wenige Boxen zu verwenden. Dabei soll dennoch, wie bisher, die Anzahl der uebrigen Kleidungsstuecke minimal sein.
Formal ausgedrueckt, soll nun fuer die bekannten Problemdefinitionen (Definition 11, Definition 19, Definition 20) folgendes ergaenzt werden: Fuer die minimale Summe $L^* \coloneqq \#(L)$ fuer beliebige $k \in \mathbb{N}$ soll das kleinste k gefunden werden, sodass Boxen $X_1,\ \dots\ ,\ X_k$ existieren mit $L^* = \#(L)$.
Da sie allerdings bis jetzt nur eine kleinere Anzahl an Boxen zu Verfuegung haben, stoszen sie auf ein weiteres Problem: Mit einer Maximalanzahl an Boxen D soll die Anzahl der uebrigen Kleidungsstuecke minimiert werden.

[3] Bei den Dreien handelt es sich leider noch immer um Marius, Emily und Merle und nicht um die Drei Fragezeichen Justus Jonas, Peter Shaw und Bob Andrews.

4. Komplexitaet des Problems

Nachdem die mathematischen Definitionen des Problems gegeben wurden, werden nun Ueberlegungen zur Komplexitaet des Problems angestellt. Dabei wird im Folgenden gezeigt, dass das Problem NP-schwer ist. Diese Tatsache, zusammen mit der von den meisten Wissenschaftlern angenommen Vermutung, dass $P \neq NP$, hat zur Folge, dass es fuer das soeben definierte Problem kein effizientes Loesungsverfahren gibt. Das bedeutet konkret, dass kein Algorithmus mit einer polynomiellen Laufzeitkomplexitaet existiert, der das Problem exakt loesen kann.
Dementspraechend werden im Folgenden Algorithmen mit polynomieller Laufzeitkomplexitaet angegeben, die das Problem nur moeglichst gut aber mit nichten perfekt loesen. Auszerdem werden Verfahren vorgestellt, die das Prolbem mit einer exponentiellen Laufzeit eindeutig loesen koennen und zumindest fuer die Beispieleingaben der BwInf-Website sehr schnell sind. Genaueres zu diesen Loesungen folgt im naechsten Kapitel.
Nun zum mathematischen Beweis der NP-schwere:

Satz 16: NP-Schwere des Problems P-I

Das Problem P-I ist NP-schwer.

Beweis:
Im Folgneden soll 3-Sat in Polynomialzeit auf das Problem P-I reduziert werden. Da 3-Sat seit [5] als NP-vollstaendig bekannt ist, folgt daraus die NP-schwere des Problems P-I.

Nun sei eine Instanz von 3-Sat gegeben durch eine aussagenlogische Formel F in konjungtiver Normalform mit genau 3 Literalen pro Klausel. Dabei seien Variablen $x_1,\ x_2,\ \dots\ x_n$ gegeben, sodass genau $2n$ Literale der Form x_i, bzw. $\neg x_i$ vorkommen koennen. Weiter seien m Klauseln der Form $L_j = (\alpha \vee \beta \vee \gamma)$, wobei $\alpha,\ \beta,\ \gamma$ Literale, wie oben beschrieben sind, sodass $F = L_1 \wedge \dots \wedge L_m$.

Nun soll eine Probleminstanz von P-I definiert werden. Dabei wird wird die Formel F dargestellt durch die Matrix $A = (a_{ij}) \in \mathrm{Mat}(m,\ 2n;\ \mathbb{Z})$, mit $a_{ij} = 1$ genau dann, wenn Literal r_i in Klausel j vorkommt, und sonst $a_{ij} = 0$, dabei ist $R \coloneqq \{r_1,\ r_2,\ \dots\ r_{2n}\}$ die Menge der Stilrichtungen und $S \coloneqq \{s_1,\ s_2,\ \dots\ s_m\}$ die Menge der Sorten, wobei $r_{2i-1} = x_i$ und $r_{2i} = \neg x_i$ fuer alle $i \in \{1,\ 2,\ \dots\ n\}$, sowie $s_j \coloneqq L_j$. Somit wurden die Literale mit Stilrichtungen und die Klauseln mit Kleidungssorten assoziert. Weiter wird nun die Relation K definiert durch

$$K \coloneqq R \times R \setminus \Big\{(r_{2i-1}, r_{2i}) = (x_i,\ \neg x_i)\ :\ i \in \{1,\ 2,\ \dots\ n\}\Big\}.$$

D.h., dass alle Stilrichtungen, also Literale, paarweise kombinierbar sind, auszer $r_{2i-1} = x_i$ mit $r_{2i} = \neg x_i$ fuer alle $i \in \{1,\ 2,\ \dots\ n\}$. Zuletzt wird $B \coloneqq 1$ und $C \coloneqq 3$ definiert.
Nun bleibt zu zeigen, dass F genau dann erfuellbar ist, wenn die Loesung des Problems P-I mindestens eine Box enthaelt:

1. Teil: Wenn die Loesung mindestens eine Box enthaelt, ist F erfuellbar.
Es sei $Z = (z_{ij})$ eine Box der Loesung des definierten Problems. Dann enthaelt jede Zeile $i \in \{1,\ 2,\ \dots\ m\}$ mindestens eine 1 an einem Eintrag z_{ij_i} mit $j_i \in \{1,\ 2,\ \dots\ 2n\}$. Dann setzt man das Literal $r_{j_i} \in R$ als **wahr** in der gesuchten Belegung der Variablen, sodass $x_{j_i} =$ **wahr**, falls r_{j_i} keine negierte Variable ist, bzw. $x_{j_i} =$ **falsch**, falls r_{j_i} eine negierte Variable ist.
Per Definition von A ist dann in jeder Klausel mindestens ein Literal **wahr**, sodass F erfuellt ist. Auszerdem werden, per Definition von K, niemals gleichzeit x_i und $\neg x_i$ auf **wahr** gesetzt.

2. Teil: Wenn F erfuellbar ist, enthaelt die Loesung mindestens eine Box.
Es sei F erfuellbar. Dann exisitert eine Belegung der Variablen, sodass alle Klauseln L_i mindestens ein Literal r_{j_i} mit dem Wert **wahr** beinhalten. Dann laesst sich eine Box $Z = (z_{ij})$ konstruieren, die die Nebenbedinungen erfuellt, indem $z_{ij} = 1$ gesetzt wird, wenn $j = j_i$ ist. Die restlichen Eintrage sind 0. Dann existiert in jeder Zeile mindestens ein Eintrag Somit existiert eine Loesung des Problems in Form ebendieser Box, die die Nebenbedinungen erfuellt, sodass inbsesondere auch eine optimale Loesung existiert.

Daraus folgt der Satz!
Aus diesem lassen sich einige Aussagen, ueber die anderen vorgestellten Probleme treffen:

Korollar 17:

Das Problem P- ist NP-schwer. Dies folgt sofort, da im Beweis $B = 1$ und $C = 3$ verwendet wurde.

Korollar 18:

Das Problem P-II ist NP-schwer. Um dies zu zeigen reduziert man P-I auf P-II, indem man $T = 1$, $r = 1$, sowie $A_1 = A$ waehlt.

Korollar 19:

Die Varianten des Problems der Erweiterung III sind NP-schwer. Es lassen sich trivialerweise P-I und P-II je auf den entsprechenden Teil 1 dieser Erweiterung reduzieren. Weiter lassen sich P-I und P-II auf den Teil 2 reduzieren, indem man D ausreichend grosz waehlt. Beispielsweise durch $D = \#(A)$.

Bemerkung 20:

Eine interessante Bemerkung ist, dass hier nur gezeigt wurde, dass das Problem P-I (mit allgemeinen Schranken B und C) NP-schwer ist, und dass das Problem P- NP-schwer ist ($B = 1$, und $C = 3$). Weiter laesst sich der Beweis fuer jede Variante des Problems mit $B = 1$ und C fest aber beliebig verwenden. Varianten mit $B \geq 2$ koennte allerdings eine andere Komplexitaet aufweisen.

5. Loesungsvorschlaege

Nun werden Loesungsvorschlaege fuer das Problem vor gestellt. Zuerst wird dazu ein Verfahren vorgestellt, welches das Problem als Ganzzahliges Optimierungsproblem formuliert. Diese Loesung wird daraufhin durch Betrachtung der maimalen Cliquen der Relation K deutlich verbessert. Anschlieszend wird eine weitere Loesung vorgestellt, welche ebenfalls diese Cliquen nutzt und nach dem Greedy-Verfahren Boxen erstellt. Fuer die letzten beiden Loesungsvorschlaege werden ebenfalls Optimierungen angegeben. Zuletzt werden alle Verfahren bezueglich ihrer Zeitkomplexitaet analysiert. Eine Evaluation der Verfahren wiederum geschieht in Kapitel 9.

5.1. Loesungsvorschlag I: Ganzzahliges Lineares Programm

Der folgende Loesungsvorschlag loesst das Problem, indem dieses als ganzzahliges lineares Programm (engl. integer linear program, kurz ILP oder IP) definiert wird. Dabei wird eine ausreichend grosze Anzahl k der Boxen definiert (s. unten). Dann wird fuer jede der k Boxen $X(l)$ fuer $l \in \{1,\ 2,\ \dots\ k\}$ die Matrix $X(l) = (x_{ij}^{(l)})$ mit unbekannten $x_{ij}^{(l)} \in \mathbb{N}_0$ definiert.

Satz 21:

Die Anzahl der Boxen in einer Loesungsinstanz des Problems (P-I) ist nach oben beschraenkt durch

$$\frac{1}{B} \cdot \min\Big\{\#(A^i)\ :\ i \in \{1,\ 2,\ \dots\ m\}\Big\}$$

Beweis:
Fuer die Matrix A, sei i_0 die Zeile mit minimaler Summe, sodass

$$g \coloneqq \frac{1}{B} \cdot \min\Big\{\#(A^i)\ :\ i \in \{1,\ 2,\ \dots\ m\}\Big\} = \frac{1}{B} \cdot \#(A^{i_0}).$$

Fuer einen Widerspruch werde angenommen, dass in einer Loesung des Problems P-I mehr als g Boxen verwendet werden. Also $k > g$. Nun wird gezeigt, dass die Zeilen X^{i_0} der Boxen, die in der Loesung verwwendet werden, in Summe groeszer sind als $\#(A^{i_0})$. Was zu einem Widerspruch fuehrt. Dazu sei

$$X \coloneqq X_1 + \ \dots\ X_k,$$

sodass per Definition gilt, dass $L = A - X \geq 0$. Wegen $L \geq 0$ und $X_i \geq 0$ fuer all i folgt, dass inbsesondere auch

$$\#(A^{i_0}) - \#(X^{i_0}) \geq 0 \iff \#(A^{i_0}) \geq \#(X^{i_0})$$

gilt. Weiter gilt fuer jede Boxen X_l fuer $l \in \{1,\ 2,\ \dots\ k\}$ per Definition, dass $\#(X_l^{i_0}) \geq B$. Daraus folgt

$$\#(X^{i_0}) = \#(X_1^{i_0}) + \dots + \#(X_k^{i_0}) \geq k \cdot B.$$

Wegen $k > g$ folgt daraus

$$\#(X^{i_0}) > g \cdot B = \#(A^{i_0}).$$

Somit besteht ein Widerspruch zu

$$\#(A^{i_0}) \geq \#(X^{i_0}).$$

Definition 22: Ganzaliges lineares Programm I (ILP-I)

Nun wird das ganzzahligen, lineare Programm abhaenig von einer Probleminstanz fuer das Problem P-I durch seine Variablen, Zielfunktion und Ungleichungen definiert. Dabei sei

$$k \coloneqq \left\lceil \frac{1}{B} \cdot \min\Big\{\#(A^i)\ :\ i \in \{1,\ 2,\ \dots\ m\}\Big\} \right\rceil$$

1. **Variablen**:
   (a) Fuer alle $l \in \{1,\ 2,\ \dots\ k\}$ fuer alle $i \in \{1,\ 2,\ \dots\ m\}$ und fuer alle $j \in \{1,\ 2,\ \dots\ n\}$: Die ganzzahligen Variablen $x_{ij}^{(l)} \in \mathbb{N}_0$, die die Eintraege der Matrix X_l darstellen sollen.
   (b) Fuer alle $l \in \{1,\ 2,\ \dots\ k\}$: Die binaeren (Werte 0 und 1) Variablen $\lambda_j^{(l)}$ fuer $j \in \{1,\ 2,\ \dots\ n\}$, wobei $\lambda_j^{(l)} = 1$ bedeuten wird, dass die j-te Spalte der Matrix X_l benutzt wird (d.h., nicht nur aus Nullen besteht);

(c) Fuer alle $l \in \{1,\ 2,\ \dots\ k\}$: Die bineare Variable $e^{(l)}$, wobei $e^{(l)} = 0$ bedeuten wird, dass die Box X_l vollstaendigen unbenutzt ist, d.h. nur Nullen enthaelt.

2. **Zielfunktion**:
Es soll die Summe der benutzten Variablen maximiert x_{ij} werden: Es sei

$$X \coloneqq X_1 + \cdots + X_k.$$

Dann soll

$$f(X) \coloneqq \#(X)$$

maximiert werden.

3. **Einschraenkung** (Lineare Ungleichungen):

(a) Fuer alle $i \in \{1,\ 2,\ \dots\ m\}$ und fuer alle $j \in \{1,\ 2,\ \dots\ n\}$:

$$\sum_{l=1}^{k} x_{ij}^{(l)} \leq a_{ij}$$

(b) Fuer alle $(r_a,\ r_b) \notin K$ und fuer alle $l \in \{1,\ 2,\ \dots\ k\}$:

$$\lambda_a^{(l)} + \lambda_b^{(l)} \leq 1$$

(c) Fuer alle $l \in \{1,\ 2,\ \dots\ k\}$ und fuer alle $j \in \{1,\ 2,\ \dots\ n\}$:

$$\#((X_l)_j) - D \cdot \lambda_j^{(i)} \leq 0$$

mit $D \coloneqq \#(A)$.

(d) Fuer alle $l \in \{1,\ 2,\ \dots\ k\}$:

$$n \cdot e^{(l)} \geq \sum_{j=1}^{n} \lambda_j^{(l)},$$

sowie

$$n \cdot e^{(l)} \leq n - 1 + \sum_{j=1}^{n} \lambda_j^{(l)},$$

(e) Fuer alle $l \in \{1,\ 2,\ \dots\ k\}$ und fuer alle $j \in \{1,\ 2,\ \dots\ n\}$:

$$\#(X_l^i) \leq C$$

sowie

$$\#(X_l^i) + D \cdot (1 - e^{(l)}) \geq B$$

mit $D \coloneqq B$.

Satz 23:

Es sei eine Loesung des soeben definierten ILP's in Form der Variablen $x_{ij}^{(l)}$, mit $i \in \{1,\ 2,\ \dots\ m\}$, $j \in \{1,\ 2,\ \dots\ n\}$ und $l \in \{1,\ 2,\ \dots\ k\}$. gegeben. Dann stellen die Matrizen $X_l \coloneqq (x_{ij}^{(l)})$ fuer $l \in \{1,\ 2,\ \dots\ k\}$ mit $X_l \neq 0$ eine Loesung des Problems P-I dar.

(ohne Beweis)
Hier fuer muessen die Eigenschaften einer Loesung nach der Definition von Problem I einzeln geprueft werden. Dabei werden nur diejenigen Matrizen betrachtet die nicht vollstaendig Nullen als Eintrage haben.

5.2. Cliquen in der Relation K

Fuer eine Box $X \in \mathrm{Mat}(m,\ n;\ \mathbb{Z})$ muss fuer die Menge $R_X = \{r_i\ :\ i \in I\}$ der benutzten Stilrichtungen mit

$$I = \left\{ j \in \{1,\ \ldots\ ,\ n\} : \#(X_j) \geq 1 \right\}$$

wie in (B2) gelten, dass diese paarweise kombinierbar sind. D.h., $(x,\ y) \in K$ fuer alle $x,\ y \in R_X$. Daraus ergibt sich die Frage, ob es moeglichst grosze Teilmengen T von R gibt, sodass fuer alle $a,\ b \in T$ gilt, dass $(a,\ b) \in K$. Dieser Sachverhalt soll nun formal definiert werden. Anschlieszend werden Verfahren zum Erhalt dieser Teilmengen (in maximlaer Form) vorgestellt. Diese sogenannten maximalen Cliquen werden einen wichtigen Bestandteil der weiteren Loesungsvorschlaege sein.

Definition 24: Clique einer Relation

Es sei $R \subseteq X \times X$ eine Relation auf X. Dann ist eine Clique C in der Relation R eine Teilmenge von X, sodass fuer alle $a,\ b \in C$ gilt, dass $(a,\ b) \in R$.

Definition 25: Maximale Clique einer Relation

Es sei $C \subseteq X$ eine Clique in der Relation $R \subseteq X \times X$. Dann nennt man C maximal genau dann, wenn es kein $x \in X \setminus C$ gibt, sodass $C \cup \{x\}$ eine Clique ist.

Bemerkung 26: Clique in Graphen

Der Begriff der Clique ist hauptsaechlich bekannt aus dem Bereich der Graphentheorie. Ist $G = (V,\ E)$ ein ungerichteter Graph, bei dem V die Menge der Knoten und E die Menge der Kanten $(u,\ v)$ ist. Wobei insbesondere $(v,\ v)$ fuer alle $v \in V$ (jeder Knoten hat eine Kante zu sich selbst) und stets $(u,\ v) \in E \implies (v,\ u)$ gilt (entweder eine Kante existiert sowohl von v nach u und von u nach v, oder sie existiert nicht).
Diese Eigenschaften entspraechen genau der Reflexivitaet und Symetrie der Relation K der kombinierbaren Stilrichtungen. Somit laesst sich diese Relation als Mengen der Kanten im ungerichteten Graphen $G = (R,\ K)$ interpretieren, sodass Cliquen C in der Relation K sich als Cliquen in dem Graphen G auffassen lassen.

Algorithmus 27: Naive Bestimmung aller maximalen Cliquen in der Relation K

Nun wird ein simpler, naiver Algorithmus zur Bestimmung aller maximalen Cliquen der Relation $K \subseteq R \times R$ vorgestellt. Dieser Algorithmus baut mehrere Cliquen iterativ auf. Dabei wird angefangen mit den trivialen Cliquen, die aus einem Element bestehen. Diese Cliquen werden in einer Warteschlagenstruktur Q gespeichert. Fuer alle bisher berechneten Cliquen $C \subseteq R$ in Q wird jeweils geprueft, ob es sich um eine maximale Clique handelt. In diesem Fall kann sie zur Menge der Maximalen Cliquen hinzugefuegt werden. Ist sie hingegen keine maximale Clique, so wird fuer jedes $v \in \mathbb{R} \setminus C$, sodass $C \cup \{v\}$ eine Clique ist, die neue, groeszere Clique zu Q hinzugefuegt. Dabei ist insbesondere zu beachten, dass Cliquen mehrmals gefunden werden koennen. Siehe Algorihmus 1.

Bemerkung 28: Weitere Verfahren

Das Problem, alle Maximalen Cliquen in einen Graphen zu bestimmen ist bereits ausreichend untersucht[4]. Neben dem soeben vorgestellten Verfahren gibt es viele weitere [1–3]. Darunter beispielsweise der Bron–Kerbosch Algorithmus[5] [2], welcher eine Laufzeit von $\mathcal{O}\left(3^{n/3}\right)$ besitzt. Wie sich gleich zeigen wird, ist dies die optimale Worstcase-Laufzeit. Weiter wurde in [2] ein Verfahren vorgestellt, welches eine Laufzeit von $\mathcal{O}\left(n \cdot m \cdot C\right)$ hat, wenn C die Anzahl der maximalen Cliquen ist. Dieses Verfahren ist somit deutlich effizienter, wenn die Anzahl der Cliquen klein es. Auszerdem betrachtet es eigentlich keine Cliquen, sondern maximale, unabhaenige Mengen. Dies ist aequivalent zum Betrachtung von Cliquen, da zwischen ihnen eine bijektive Abbildung vom Graphen in den inversen Graphen besteht. Ein inverser Graph ist dabei ein Graph, wobei zwei Knoten genau dann verbunden sind, wenn sie es im urspruenglichen Graphen nicht sind. Ein weiteres Verfahren besteht darin, die maximalen Cliquen in lexikographisch aufsteigender Reihenfolge nacheinander auszugeben, sodass zwischen einer Ausgabe und der naechsten nur polynomiell viel Zeit gebraucht wird [3]. Konkret ist die gebrauchte Zeit, das sogenannte Delay hier $\mathcal{O}\left(n^3\right)$. Die Platzkomplexitaet ist allerdings $\mathcal{O}\left(b \cdot C\right)$, wobei C erneut Anzahl der maximalen Cliquen ist, also exponentiell.

[4] Siehe auch `https://en.wikipedia.org/wiki/Clique_problem#Listing_all_maximal_cliques`
[5] Siehe auch `https://en.wikipedia.org/wiki/Bron-Kerbosch_algorithm`

Algorithmus 1 : Bestimme alle maximalen Cliquen

Input : Menge R und Relation $K \subseteq R \times R$
Output : Menge M aller maximalen Cliquen $C \subseteq R$ der Relation K

```
1  M ← ∅
2  Q ← ∅ // Warteschlage Q
3  foreach r ∈ R do
4  |  if (r, r) ∈ K then
5  |  |  Q.push({r})
6  |  end if
7  end foreach
8  while Q is not empty do
9  |  C ← Q.top() // Erhalte erstes Element der Warteschlage Q und entferne es
10 |  foreach x ∈ R \ C do
11 |  |  if C ∪ {x} is a Clique then
12 |  |  |  Q.append(C ∪ {x})
13 |  |  end if
14 |  end foreach
15 |  if C is maximal Clique then
16 |  |  M.append(C)
17 |  end if
18 end while
19 return M
```

Bemerkung 29: Berechnung einer bestimmten Anzahl an maximalen Cliquen

Inbsesondere ist die Erkenntniss relevant, dass die Anzahl der maximalen Cliquen in einem Graphen mit n Knoten nur durch $3^{n/3}$ nach oben beschraenkt ist. Fuer $n = 3k$ wird ein solcher Graph durch den inversen Graphen eines Graphen mit k diskunkten 3-Cliquen erzeugt, sodass die maximalen Cliquen genau durch 3^k k-Cliquen gegeben sind [4]. Aufgrund dieser exponentiellen Anzahl, wird sich im Folgenden zeigen, dass es sinnvoll sein kann, nur eine bestimmte Anzahl an maximalen Cliquen zu generieren.
Dazu kann der bereits erwaehnte Algorihmus aus [3] verwendet werden. Dieser gibt alle maximalen Cliquen eines Graphen in lexikographisch Reihenfolge mit einem Delay von $\mathcal{O}\left(n^3\right)$ aus. Dieser Algorithmus ist leicht so zu veraendern, dass nach einer bestimmte Anzahl an Cliquen terminiert wird, da er itereativ mit einer Warteschlagenstruktur funktioniert.

5.3. Loesungsvorschlag II: Verbessertes Ganzzahliges Lineares Programm

Dieser Loesungsvorschlag soll mit Hilfe der soeben eingefuehrten maximalen Cliquen den vorherigen verbessern. Dabei ist zu bemerken, dass sehr viele der Varialen einer Matrix X_l in der vorherigen Loesung 0 sein werden, um die Kombinierbarkeitseigenschaft zu erfuellen. Insbesondere existiert fuer jede Box eine maximale Clique C, sodass die Menge der benutzten Spalten eine Teilmenge von C ist. Diese Erkenntniss motiviert die folgende Loesungsidee. Fuer jede maximale Clique wird eine aureichend grosze Anzahl k_C an Boxen eingefuehrt, welche nur Eintrage in den der Clique entsprechenden Spalten haben werden. Dabei ist fuer eine Clique $C = (c_1,\ c_2,\ \dots\ c_p)$

$$k_C \coloneqq \frac{1}{B} \cdot \min\Big\{\sum_{s=1}^{p} a_{ic_s}\ :\ i \in \{1,\ 2,\ \dots\ m\}\Big\},$$

d.h., das Minimum ueber alle Zeilen i der Summen der Eintraege in dieser Zeile der Spalten, die der Clique entspraechen.

Satz 30:

Es sei eine Loesung des Problems P-I gegeben durch die Boxen $X_1,\ X_2,\ \dots\ X_k$. Dann sei $R_X = \{r_i\ :\ i \in I\}$ die Menge der benutzten Stilrichtungen der Box X mit

$$I = \Big\{j \in \{1,\ \dots\ ,\ n\} : \#(X_j) \geq 1\Big\}.$$

Weiter seien $C_1,\ C_2,\ \dots C_p$ die maximalen Cliquen der Relation K. Dann existiert eine Verteilung der Boxen in Mengen $M_1,\ M_2,\ \dots M_p$, sodass fuer alle $f \in \{1,\ 2,\ \dots\ p\}$ gilt, dass

1. Fuer alle $X \in M_f$ gilt: $R_X \subseteq C_f$, und
2. $\#(M_f) \leq k_{C_f}$

Dies legitimiert das folgende ILP, da nun bekannt ist, dass sich alle Boxen einer Loesung so auf die Cliquen verteilen lassen, dass die Anzahl der Boxen pro Clique kleiner oder gleich k_C ist.

(ohne Beweis)

Definition 31: Ganzaliges lineares Programm II (ILP-II)

Nun wird erneut das ganzzahligen, lineare Programm abhaenig von einer Probleminstanz fuer das Problem P-I durch seine Variablen, Zielfunktion und Ungleichungen definiert. Dabei seien erneut $C_1,\ C_2,\ \dots\ C_p$ die maximalen Cliquen der Relation K. Nun sollen fuer jede maximale Clique C der Relation K genau k_C Boxen modelliert werden. Bei diesen sind allerdings alle Eintrage 0, auszer in den Spalten, die der Clique C entspraechen. Diese Spalten sollen je aus m Variablen bestehen. Um die Beschreibung des ILPs zu vereinfachen werden alle Boxen von 1 bis $k_{C_1} + \dots + k_{C_p} =: k$ durchnummeriert. Dann sei $\varphi(l)$ der Index der Clique der Box X_l.

1. **Variablen**:
   (a) Fuer alle $l \in \{1,\ 2,\ \dots\ k\}$ und fuer alle $i \in \{1,\ 2,\ \dots\ m\}$ und fuer alle $j \in C_{\varphi(l)}$: Die ganzzahligen Variablen $x_{ij}^{(l)} \in \mathbb{N}_0$, die die Eintraege der Matrix X_l darstellen sollen.
   (b) Fuer alle $l \in \{1,\ 2,\ \dots\ k\}$: Die bineare Variable $e^{(l)}$, wobei $e^{(l)} = 1$ bedeuten wird, dass die Box X_l vollstaendigen unbenutzt ist, d.h. nur Nullen enthaelt.
2. **Zielfunktion**:
   Es soll die Summe der benutzten Variablen maximiert x_{ij} werden: Es soll
   $$f \coloneqq \sum_{l=1}^{l} \sum_{i=1}^{m} \sum_{j \in C_f} x_{ij}^{(l)}$$
   maximiert werden. Dabei gilt
   $$f = \sum_{l=1}^{k} \#(X_l),$$
   mit der Definition, dass $x_{ij}^{(l)} = 0$ fuer $j \notin C_{\varphi(l)}$.
3. **Einschraenkung** (Lineare Ungleichungen):
   (a) Fuer alle $i \in \{1,\ 2,\ \dots\ m\}$ und fuer alle $j \in \{1,\ 2,\ \dots\ n\}$:
   $$\sum_{l=1}^{k} x_{ij}^{(l)} \leq a_{ij}$$
   mit der Definition, dass $x_{ij}^{(l)} = 0$ fuer $j \notin C_{\varphi(l)}$.
   (b) Fuer alle $l \in \{1,\ 2,\ \dots\ k\}$ und fuer alle $i \in \{1,\ 2,\ \dots\ m\}$:
   $$\#(X_l^i) \leq C$$
   sowie
   $$\#(X_l^i) + D \cdot e^{(l)} \geq B$$
   mit $D \coloneqq B$. Erneut mit der Definition, dass $x_{ij}^{(l)} = 0$ fuer $j \notin C_{\varphi(l)}$.

(c) (Sicherstellen, dass X_l leer gdw. $e^{(l)} = 1$.) Fuer alle $l \in \{1,\ 2,\ \dots\ k\}$:

$$\#(X_l) - D \cdot (1 - e^{(l)}) \leq 0$$

Wieder mit der Definition, dass $x_{ij}^{(l)} = 0$ fuer $j \notin C_{\varphi(l)}$.

Satz 32:
Es sei eine Loesung des soeben definierten ILP's in Form der Variablen $x_{ij}^{(l)}$, mit $i \in \{1,\ 2,\ \dots\ m\}$, $j \in \{1,\ 2,\ \dots\ n\}$ und $l \in \{1,\ 2,\ \dots\ k\}$. gegeben. Dann stellen die Matrizen $X_l \coloneqq (x_{ij}^{(l)})$ fuer $l \in \{1,\ 2,\ \dots\ k\}$ mit $X_l \neq 0$ eine Loesung des Problems P-I dar.

(ohne Beweis)
Auch hier muessen die Eigenschaften einer Loesung nach der Definition geprueft werden.

Optimierungen des Loesungsvorschlags II

In [4] wird gezeigt, dass die obere Schranke der Anzahl der Cliquen in einem Graphen mit n Knoten $3^{n/3}$ ist. Dementspraechend ist es moeglich, dass die Anzahl der Cliquen exponentiell im Verhaeltnis zur Anzahl der Stilrichtung waechst. In Loesungsvorschlag II ist es allerdings moeglich, statt der gesamten Menge an Cliquen, nur eine Teilmenge dieser zu verwenden. Dies fuehrt dazu, dass nicht mehr zwingend eine exakte Loesung bestimmt wird, aber ermoeglicht eine Verbesserung der Laufzeitkomplexitaet, da nicht zunaechst exponentiell viele Cliquen generiert werden muessen und auch nicht fuer exponentiell viele Cliquen Boxen erstellt werden muessen. Wie in Kapitel 5.2 bereits erwaehnt wurde, ist es moeglich nur eine bestimmte Anzahl an Cliquen zu generieren. Beispielsweise polynomiell viele in Abhaenigkeit von der Anzahl der Stilrichtungen, o.ae.

Eine weitere Optimierung fuer Probleminstanzen mit einer (gleichmaeszig verteilten) sehr hohen Anzahl an Kleidungsstuecken ist die Folgende:
Die gegebene Matrix A soll als Summe mehrere Matrizen geschrieben werden:

$$A = A_1 + \cdots + A_g$$

fuer ein $g \in \mathbb{N}$, sodass anschlieszend das vorgestellte Verfahren auf die einzelnen Matrizen A_i angewendet werden kann. Dabei ist allerdings zu bemerken, dass die optimalen Loesungen der einzelnen Matrizen zusammen nicht zwingend eine optimale Loesung fuer die Matriz A liefern. Dennoch erfuellt eine solche aus Teilloesungen zusammengesetzte Loesung die Nebenbedinungen. Somit handelt es sich um ein Approximationsverfahren.
Fuer die Aufteilung der Matrix kann man beispielsweise ein $g \in \mathbb{N}$ waehlen, sodass $\frac{a_{ij}}{g} \geq B$ fuer alle Eintrage der Matrix. und dann Matrizen der Form $\frac{1}{g} \cdot A$ erstellen, wobei auf Ganzzahlen gerundet wird und der Rest auf die Matrizen verteilt wird.

5.4. Loesungsvorschlag III: Greedy-Ansatz

Wenngleich Loesungsvorschlag II bereits ohne die Optimierungen alle Beispieleingaben der BwInf-Website loesen kann (s. Kapitel 9), hat dieses Verfahren bei groszen Eingaben, wegen seiner exponentiellen Laufzeit (s.u.), eine zu lange Laufzeit. Deswegen wird nun ein letztes Loesungsverfahren vorgestellt, welches eine deutliche bessere Laufzeit besitzt erstellt iterativ Boxen generiert, bis keine mehr erzeugt werden koennen.

Algorithmus 33: Greedy-Verfahren

Wie bereits erlaeutert, soll dieser Algorithmus solange neue Boxen erstellen, bis keine weitere Box mehr erstellt werden kann. Dazu wird die Funktion **next** verwendet, welche wiederum unterschiedliche Implementierung haben kann. Dazu im naechsten Absatz mehr. Der Algorithmus initialisiert zunaechst eine Liste $\vec{X}$ welche die erstellten Boxen beinhalten soll, die Matrix A' mit A, die in Folgenden die restlichen Kleidungsstuecke darstellen soll, sowie den Boolean b, welcher angeben wird, ob der Algorihmus weiter laufen soll.
Solange b den Wert **True** hat, wird der Rumpf der While-Schleife ausgefuehrt. Dort wird zunaechst mit Hilfe der eben genannten Funktion **next** abhaenig der restlichen Kleidungsstuecke A' die naechste Box X erstellt. Falls diese der Nullmatrix 0 entspricht, wird b auf **False** gesetzt, sodass die While-Schleife im naechsten Durchgang terminiert. Denn in diesem Fall konnte keine naechste Box erstellt werden. Andernfalls wird X zu den Boxen $\vec{X}$ hinzugefuegt und A' durch $A' - X$ akutalisiert. Den dies entspricht den uebrigen Kleidungsstuecken, sofern X eine valide Box bezueglich der uebrigen Kleidungsstuecke A' ist.

Algorithmus 2 : Iterativer Algorithmus zum erzeugen von Boxen

Input : Matirx $A \in \mathrm{Mat}(m,\ n; \mathbb{Z})$
Output : Liste $\vec{X}$ an Boxen

```
1  X⃗ ← ∅
2  A' ← A
3  b ← True
4  while b do
5  |  X ← next(A')
6  |  if X = 0 then
7  |  |  b ← False
8  |  end if
9  |  else
10 |  |  X⃗.append(X)
11 |  |  A' ← A' − X
12 |  end if
13 end while
14 return X⃗
```

Fuer das konkrete Erzeugen der naechsten Boxen, sollen nun verschiedene Moeglichkeiten vorgestellt werden. Der erste Algorithmus probiert iterativ jede der gegebenen Cliquen, um eine Box zu erstellen, welche nur Spalten (Stilrichtungen) dieser Clique enthaelt und benutzt die erste fuer die dies moeglich ist. Dabei wird stets eine moeglichst grosze Box erzeugt. Fuer eine gegebene Clique exisitert genau dann eine valide Box fuer die gegebe Matrix A', falls in jeder Zeile (Sorte) die Summe der zur Clique gehoerenden Spalten groeszer oder gleich B ist. Dementspraechend wird durch alle Cliquen Q iterriert und jeweils eine Box X erzeugt (initialisiert als die Nullmatrix) und ein boolescher Wert b mit **True** initialisiert, welcher aussagt, ob fuer die aktuelle Clique eine Box gefunden wurde. Um X zu erzeugen, wird durch die m Zeilen interiert. Fuer die Zeile i wird die Anzahl s der (in dieser Zeile) breits verwendeten Kleidungsstueck gespeichert (und mit 0 initialisiert). Fuer jede Spalten der Clique wird dann die Zahl v durch $\min(a'_{iq},\ C - s)$ initialisiert. Dies entspricht dem Maximalen Wert den die Box am aktuellen Eintrag annehmen kann, ohne die Maximalanzahl C der Zeilen zu verletzten. Dann wird der aktuelle Eintrag x_{iq} der Box X auf v gesaetzt und die Summe s um v erhoeht. Fall s nach Iteration durch alle Spalten kleiner ist als B, so wird b auf **False** gesetzt. Denn dies bedeutet, dass in der aktuellen Zeile nicht ausreichend Kleidungsstuecke in Spalten der Clique sind, um eine valide Box zu erstellen. Nach dem X erzeugt wurde, wird diese zurueckgegeben (und dadurch die Funktion beendet), fall b noch den Wert **True** hat. Wurde nach Iteration durch alle Cliquen keine Box zurueckgegeben, wird die Nullmatrix 0 zurueckgegeben, die aussagt, dass keine Box gefunden werden konnte (siehe Algorihmus 3).
Eine simple Optimierung fuer diesen Algorithmus ist es, global zu speichern, welche Cliquen keine Box erstellen konnten, da dies im naechsten Aufruf der Funktion *next* ebenfalls nicht moeglich sein wird.

Algorithmus 3 : Funktion zum Erzeugen der naechsten Box (I)

```
   Input  : B, C ∈ ℕ und Cliquen Q⃗
 1 Function next(A' ∈ Mat(m, n;ℤ)):
 2     foreach Q ∈ Q⃗ do
 3         X ← 0 // Nullmatrix
 4         b ← True
 5         // Iteriere durch alle Zeilen
 6         for i ← 1 to m do
 7             s ← 0
 8             // Iteriere durch alle Spalten der Clique Q
 9             foreach q ∈ Q do
10                 v ← min(a'_iq, C − s)
11                 x_iq ← v
12                 s ← s + v
13             end foreach
14             if s < B then
15                 b ← False
16             end if
17         end for
18         if b = True then
19             return X
20         end if
21     end foreach
22     return 0
```

Im Folgenden soll dieses Verfahren zur Erzeugnung der naechsten Box so veraendert werden, dass A' moeglichst gleichmaeszig verteilt ist. Das intuitive Verfahren bei dem versucht wird, stets die Kleidungsstuecke (nach Sorte und Stilrichtung) zu verwenden von denen noch am meisten vorhanden ist und weniger von den Kleidungsstuecken zu waehlen, von welchen weniger vorhanden ist. Eine gleichmaeszig Verteilung der Matrix soll nun formalisiert werden, indem sie so durch einen ganzzahligen Wert ausgedrueckt wird, dass ein kleinerer Wert einer gleichmaeszigeren Verteilung entspricht als ein groeszerer Wert. Genauer soll hier die Gleichmaeszigkeit $\mathrm{ms}(A)$ definiert sein, als die Summe der euklidischen Distanzen

$$\|A^i - A^j\|_2 \coloneqq \sqrt{(a_{i1} - a_{j1})^2 + \cdots + (a_{in} - a_{jn})^2}$$

aller Paare i, j. Dann gilt $\mathrm{ms}(A) = 0$ wenn alle Zeilen der Matrix gleich sind. Um eine moeglichst gute Gleichmaeszigkeit der Matrix $A' - X$ zu erreichen, iterriert die Funktion *next* ebenfalls durch alle Cliquen Q und geht nach dem Greedy-Prinzip vor:
Vor Beginn der Berechnung fuer jede Clique wird fuer jede Spalte j ein Wert bestimmt, welcher anschlieszend in jeder Zeile in dieser Spalte moeglichst genau erzielt werden soll. Dieser Wert sei

$$y_j \coloneqq \frac{1}{n} \cdot (a_{1j} + \cdots + a_{mj})$$

also der Durchschnitt der Eintrage der Matrix A' in der j-ten Spalte.
Fuer eine gegebene Clique Q soll nun eine valide Box X berechnet werden, die erzielt, dass in der Matrix $A' - X$ in jeder Zeile i die Werte a_ij nah an y_j liegen. Wobei die Grenzen B und C fuer die Summe der Zeilen der Box beachtet werden muessen. Zum berechnen der Zeile i der Box wird wie folgt vorgegangen:

Es wird eine Prioritaetswarteschlange (engl. priority queue) verwendet, welche die Indizes $j \in Q$ speichert, und anhand des Wertes $a_{ij} - x_{ij} - y_j$ so sortiert, dass der groeszte Wert *vorne* ist. Dabei ist x_{ij} ein Eintrag der Box X die mit 0 initialisiert wird, sodass der Wert $a_{ij} - x_{ij} - y_j$ betragsmaeszig dem Abstand des aktuellen Wertes von $A' - X$ im Eintrag ij und dem Zielwert y_j entspricht. Dementspraechend soll der Index j mit dem groeszten Wert einen kleineren Wert bekommen, indem x_{ij} um 1 erhoeht wird. Den Index der aktuell des groeszten Wert hat erhaelt man durch die Prioritaetswarteschlange. Dann wird der Index j mit dem groeszten Wert zunaechst aus der Warteschlage entfernt, x_{ij} erhoeht, und anschlieszend wird j wieder eingefuegt. Die

letzten beiden Schritte duerfen allerdings nur geschehen, wenn $a_{ij} > 0$ ist, da ansonsten keine Kleidungsstuecke der entspraechenden Stilrichtung und Sorte vorhanden sind.

Dies wird so lange wiederholt, bis die maximale Summe C der aktuellen Zeile X^i erreicht ist, oder die Warteschlage leer ist. Dabei ist zu beachten, dass es moeglich ist das die minimale Summe B der akutellen Zeile nicht erreicht wurde. In diesem Fall wird die berechnete Box verworfen und mit der naechsten Clique fortgefahren. Dieses Verfahren erzielt allerdings nicht zwingend den optimal Wert von $\mathrm{ms}(A' - X)$ unter der gegebenen Clique Q, stellt aber eine gute Heuristik dar.

Nachdem fuer alle Cliquen versucht wurde, eine Box zu berechnen, wird die Box zurueckgegeben, die den kleinsten Wert $\mathrm{ms}(A' - X)$ liefert. Konnte keine Box berechnet werden, wird die Nullmatrix 0 zurueckgegeben. Es sei angemerkt, dass das vorgestellte Verfahren nur eine Moeglichkeit ist, das anschaulich einfach verstaendliche Verfahren, Boxen so zu erstellen, dass eine *gleichmaeszige* Matriz entsteht, formal umzusetzen.

Optimierung des Loesungsvorschlags III

Wie auch im vorherigen Loesungsvorschlag ist es moeglich statt der gesamten Menge der maximalen Cliquen nur eine Teilmenge dieser zu verwenden, sodass nicht zunaechst exponentiell viele Cliquen genereiert werden muessen. Dadurch koennte potenziell auch die Loesung des Loesungsvorschlags III verschlechtert, bzw. veraendert werden. Allerdings ermoelgicht dies inbsesondere bei Loesungsvorschlag III eine deutliche Verbesserung der Laufzeitkomplexitaet.

5.5. Analyse der Zeitkomplexitaet der Loesungsvorschlaege

Nun sollen die vorgestellten Loesungen bezueglich ihrere Zeitkomplexitaet theoretisch analysiert werden. Eine Evaluation der Loesungsvorschlaege durch ihre Ergebnisse und tatsaechlichen Laufzeiten erfolgt in Kapitel 9.

Bestimmung aller maximaler Cliquen

Das vorgestellt, naive Verfahren besitzt eine exponentielle Laufzeit. Im allgemeinen wurde bereits eingefuehrt, dass die Maximalanzahl an maximalen Cliquen in $\mathcal{O}\left(3^{n/3}\right)$ ist. Das erwaehnte Verfahren, um nur eine besimmte Anzahl an Cliquen zu generieren, hat wiederum eine Laufzeit von $\mathcal{O}\left(n^3 \cdot Q\right)$, wobei Q die Anzahl der Cliquen ist.

Loesungsvorschlag II: Verbessertes ganzzahliges Optimierungsproblem

Die Laufzeitkomplexitaet des ILP Solvers ist schwer zu erfassen. Diese ist in Worst-Case exponentiell im Verhaeltnis zur Anzahl der Variablen und Bedingungen, ist aber so optimiert, dass eine theoretische Analyse nicht viele Informationen ueber die tatsaechliche Laufzeit gibt. Deswegen soll nun nur die Anzahl der Variablen und Bedingungen analysiert werden.

An diesem Zeitpunkt war leider die Zeit zum Schreiben der Doku endgueltig vorbei. Anders als erwartet ziemlich ganau gegen Mitternacht :(

Loesungsvorschlag III: Greedy-Ansatz

Fuer den letzten Loesungsvorschlag soll nun die tatsaechliche Laufzeit analysiert werden.
Die erste Variante der next-Funktion besitzt die Laufzeitkomplexitaet $\mathcal{O}\left(\#(\vec{Q}) \cdot m \cdot n\right)$, weil jede maximale Clique maximal n Element besitzt.
Die Laufzeit des Algorithmus 2 ist dann abhaenig von der Anzahl $f(A)$ der maximalen Anzahl an Boxen, welche sich durch die minimale Zeile der Matrix A mit Summe a' abschaetzen laesst. Die Laufzeit ist dann

$$\mathcal{O}\left(\frac{a'}{B} \cdot \#(\vec{Q}) \cdot m \cdot n\right)$$

6. Erweiterung II: Betrachtung von Kleidergroeszen

In diesem Kapitel soll das bereits formal definierte Problem P-II behdanelt werden. Dazu werden aufbauend auf den Loesungen der Erweiterung I aus Kapitel 5 zwei Loesungsvorschlaege vorgestellt, und anschlieszend bezueglich ihrere Zeitkomplexitaet analysiert.
Das erste Verfahren ist ein Ganzaliges lineares Programm analog zu dem aus Kapitel 5.3. Es werden ebenfalls die maximalen Cliquen, bzw. eine Teilmenge dieser verwendet, um ein ILP zu definieren, welches das Problem optimal loest. Das zweite Verfahren basiert analog zu Kapitel 5.4 auf dem Greedy-Prinzip und berechnet iterativ neue Boxen, bis dies nicht mehr moeglich ist.

6.1. Loesungsvorschlag I: Eine Groesze pro Box

Der erste und einfachste Loesungsvorschlag reduziert das Problem P-II auf das zuvor behandelte Problem P-I. Dazu wird festgelegt, dass jede Box nur eine Kleidungsgroesze verwenden soll. Dann laesst sich eine Loesung des Problems P-II als zusammengesetzte Loesung der Loesungen fuer das Problem P-I berechnen. Dazu werden die Matirzen $A_1,\ A_2,\ \dots\ A_T$ als Eingaben fuer P-I verwendet. Somit werden T Mengen an Boxen berechnet, welche je einer Kleidungsgroesze entspraechen. Dabei ist zu bemerken, dass man alle zuvor vorgestellten Loesungen verwenden kann, um die die einzelnen Loesungen zu berechnen. So erzeugte Loesungen sind allerdings nicht umbedingt optimal, und liefern inbsesondere keine besonders interessante Boxen, sodass nun weitere Loesungsvorschlaege gegeben werden sollen.

6.2. Loesungsvorschlag II: Ganzzahliges lineares Programm

Es soll nun ein Ganzzahliges lineares Programm definiert werden, welches analog zu dem aus Kapitel 5.3 funktioniert. Dabei werden erneut die maximalen Cliquen verwendet, um Boxen zu modellieren, welche nur Spalten aus einer bestimmten Clique verwenden. Zusaetzlich wird den Boxen ein Intervall $\{s,\ s+1\ \dots\ s+r-1\}$ fuer alle $s \in \{1,\ 2,\ \dots\ T-r+1\}$ zugeordnet. Dementspraechend soll nun eine obere Schranke fuer die Anzahl der Boxen die einer bestimmten Clique $Q = (q_1,\ q_2,\ \dots\ q_p)$ und einem bestimmten solchen Intervall fuer ein $s \in \{1,\ 2,\ \dots\ T-r+1\}$ gefunden werden:

Satz 34:

Analog zu Satz 30 soll dieser Satz die Anzahl der Boxen im folgenden ILP rechtfertigen:
Fuer

$$k_{Q,s} \coloneqq \frac{1}{B} \cdot \sum_{t=s}^{s+r-1} \min\Big\{\sum_{j=1}^{p} a_{ic_j}^{(t)}\ :\ i \in \{1,\ 2,\ \dots m\}\Big\}$$

(die Summe der oberen Schranken in den Matrizen $A_s,\ A_{s+1},\ \dots A_{s+r-1}$ bezueglich der Clique Q), ist $k_{Q,s}$ eine obere Schranke der Boxen, die zur Clique Q gehoren und Groszen mit dem Intervallanfang s beinhalten.
(ohne Beweis)

Definition 35: Ganzzahliges lineares Programm III (ILP-III)

Im Folgenden werden fuer jedes Paar an Clique Q und Intervallanfang $s \in \{1,\ 2,\ \dots\ T-r+1\}$ genau $k_{Q,s}$ Boxen modelliert. Der Einfachheit halber werden die Boxen X von 1 bis k fuer

$$k \coloneqq \sum_{g=1}^{p} \sum_{s=1}^{T-r+1} k_{Q_g,s},$$

wobei p Cliquen $Q_1,\ Q_2,\ \dots\ ,Q_p$ gegeben sind. Dann sei $\varphi(l)$ der Index der Clique der Box X_l, und $\phi(l)$ der Intervallanfang s dieser Box. Nun besteht eine Box X_l aus r Matrizen $Z_{\phi(l)}^{(l)},\ \dots\ Z_{\phi(l)+r-1}^{(l)}$, wobei im Folgenden nur die relevanten Eintrage (d.h. die Eintrage in Spalten der Clique entspraechend) als Variablen betrachtet werden. Die restlichen Eintrage sind im Folgenden als 0 zu verstehen.

1. **Variablen**:

   (a) Fuer alle $l \in \{1,\ 2,\ \dots\ k\}$ und fuer alle $t \in \{\phi(l),\ \dots\ \phi(l)+r-1\}$ werden die Eintraege der Matrix $Z_t^{(l)}$ durch die ganzzahligen Variablen $z_{ij}^{(l,\ t)} \in \mathbb{N}_0$, fuer alle $i \in \{1,\ 2,\ \dots\ m\}$ und fuer alle $j \in C_{\varphi(l)}$ dargestellt.

(b) Fuer alle $l \in \{1,\ 2,\ \dots\ k\}$: Die bineare Variable $e^{(l)}$, wobei $e^{(l)} = 1$ bedeuten wird, dass

$$X_l \coloneqq Z^{(l)}_{\phi(l)} + \cdots + Z^{(l)}_{\phi(l)+r-1}$$

nur Nullen enthaelt. D.h., dass die Matrizen $Z^{(l)}_{\phi(l)},\ \dots\ Z^{(l)}_{\phi(l)+r-1}$ alle gleich der Nullmatrix sind.

2. **Zielfunktion**:
Es soll die Summe der benutzten Variablen maximiert $z_{ij}^{(l,\ t)}$ werden: Es soll

$$f = \sum_{t=1}^{k} \#(X_l),$$

mit der Notation wie oben, maximiert werden, wobei die nicht definierten Variablen als 0 interpretiert werden.

3. **Einschraenkung** (Lineare Ungleichungen):

(a) Fuer alle $t \in \{1,\ 2,\ \dots\ T\}$, fuer alle $i \in \{1,\ 2,\ \dots\ m\}$ und fuer alle $j \in \{1,\ 2,\ \dots\ n\}$:

$$\sum_{l=1}^{k} z_{ij}^{(l,\ t)} \leq a_{ij}^{(t)},$$

wobei $A_t = (a_{ij}^{(t)})$. Dabei wird $z_{ij}^{(l,\ t)}$ als 0 interpretiert, falls diese Variable nicht existiert. Dies ist der Fall wenn $t \notin \{\phi(l),\ \dots\ \phi(l) + r - 1\}$.

(b) Fuer alle $l \in \{1,\ 2,\ \dots\ k\}$ und fuer alle $i \in \{1,\ 2,\ \dots\ m\}$:

$$\#(X_l^i) \leq C$$

sowie

$$\#(X_l^i) + D \cdot e^{(l)} \geq B$$

mit $D \coloneqq B$ und der Definition von X_l wie oben.

(c) (Sicherstellen, dass X_l leer gdw. $e^{(l)} = 1$.) Fuer alle $l \in \{1,\ 2,\ \dots\ k\}$:

$$\#(X_l) - D \cdot (1 - e^{(l)}) \leq 0$$

Wieder mit der Definition von X_l wie oben und einem ausreichend groeszen D. Beispielsweise $D \coloneqq \#(A_1 + \cdots + A_T)$.

Satz 36:

Es sei eine Loesung des soeben definierten ILP's in Form der Variablen $z_{ij}^{(l,\ t)}$ mit $l \in \{1,\ 2,\ \dots\ k\}$, $i \in \{1,\ 2,\ \dots\ m\}$, $j \in C_{\varphi(l)}$, sowie $t \in \{\phi(l),\ \dots\ \phi(l) + r - 1\}$ gegeben. Dann stellen die Matrizen $Z^{(l)}_{\phi(l)},\ \dots\ Z^{(l)}_{\phi(l)+r-1}$ mit $Z_t^{(l)} \coloneqq (z_{ij}^{(l,\ t)})$ fuer $l \in \{1,\ 2,\ \dots\ k\}$ mit $X_l \neq 0$ eine Loesung des Problems P-II dar, wobei erneut die nicht definierten Variablen als 0 interpretiert werden.

(ohne Beweis)

Optimierungen des Loesungsvorschlags II

Da wie gehabt potenziell exponentiell viele maximale Cliquen exisiteren, ist es auch in diesem Loesungsvorschlag sinnvoll, nicht die gesamte Menge der maximalen Cliquen zu berechnen, sondern eine Maximalanzahl dieser festzulegen. Diese kann beispielsweise polynomiell von der Anzahl der Stilrichtungen abhaengen, o.ae.
Ebenfalls laesst sich bei diesem Loesungsvorschlag die Optimierung anwenden, die die gegebene Probleminstanz in viele kleinere Instanzen zerlegt, sodass die Menge der Loesungen der einzelnen Instanzen eine Loesung des Problems erzeugen. Diese ist allerdings nicht umbedingt eine optimale Loesung. Dazu werden die Matrizen A_t als Summe mehrere Matrizen dargestellt:

$$A_t = A_t^{(1)} + \cdots + A_t^{(g)}$$

fuer alle $t \in \{1,\ 2,\ \dots T\}$ und ein festes $g \in \mathbb{N}$. Dann stellen die Matrizen $A_1^{(i)}, \dots\ A_T^{(i)}$ fuer alle $i \in \{1,\ 2,\ \dots\ g\}$ die kleineren Probleminstanzen dar.

6.3. Loesungsvorschlag III: Greedy-Verfahren

Wie in Kapitel 5.4 soll nun ein Verfahren vorgestellt werden, welches nach dem Greedy-Prinzip eine nicht optimale Loesung berechnet, aber eine deutlich bessere Laufzeitkomplexitaet hat. Dieses erstellt ebenfalls itereativ Boxen, bis dies nicht mehr moeglich ist. Dazu wird ein Verfahren analog zum zweiten Teil des Kapitels 5.4 verwendet, wobei die Box X nun in Abhaenigkeit von einer Clique Q und einem Intervallanfang $s \in \{1,\ 2,\ \dots\ T - r + 1\}$ berechnet und die Gleichmaeszigkeit einer Box als die Summe von $\mathrm{ms}(Z_s - A_s)$, bis $\mathrm{ms}(Z_{s+r-1} - A_{s+r-1})$ definiert wird, wobei $\mathrm{ms}(A)$ definiert ist wie in Kapitel 5.4.
Auch in diesem Verfahren soll die Box zeilenweise erzeugt werden, wobei in einer gegebenen Zeile stets die Spalte und die Matrix $A \in \{A_s,\ \dots A_{s+r-1}\}$ gewaehlt wird, die den groeszten Wert $a_{ij}^{(t)} - z_{ij}^{((t))} - y_j^{(t)}$ besitzt.

Optimierung des Loesungsvorschlags III

Auch hier ist folgende Optimierung zu erwaehnen: Aufgrund der bereits mehrfach eingefuehrten potenziell exponentiellen Anzahl der maximalen Cliquen im Verhaeltnis zur Anzahl der Stilrichtungen, ist es sinnvoll, nicht alle maximalen Cliquen zu berechnen, indem das in Kapitel 5.2 vorgestellte Verfahren verwendet wird. Dadurch kann das vorgestellte Verfahren moeglicherweise schlechtere, bzw. andere Ergebnisse erzeugen, hat aber eine deutlich bessere Laufzeit.

7. Erweiterung III: Minimierung der verwendeten Boxen

In diesem Kapitel sollen zwei Probleme behandelt werden, welche die Anzahl der verwendeten Boxen betreffen. Im ersten Teil soll fuer die Minimalanzahl n_0 an uebrigen Kleidungsstuecken eine Loesung gefunden werden, welche die Anzahl der verwendeten Boxen minimiert.
Im zweiten Teil soll eine Maximalanzahl $D \in \mathbb{N}$ an Boxen gegeben werden unter welcher das urspruengliche Problem geloest wird.
Dieses Probleme sollen sowohl fuer Erweiterung I als auch fuer Erweiterung II geloest werden. Dazu werden jeweils die bereits vorgestellten Greedy- und ILP-Loesungen angepasst.

7.1. Erweiterung I

1. Teil
Zunaechst sei n_0 die Minimalanzahl an uebrigen Kleidungsstuecken, welche mit Hilfe der Verfahren aus Kapitel 5 exakt bestimmt werden kann. Nun soll das Verfahren aus Loesungsvorschlag II des Kapitels 5 so angepasst werden, dass es die Anzahl der verwendeten Boxen minimiert: Dazu wird eine neue Zielfunktion definiert durch

$$f \coloneqq \sum_{l=1}^{k} e^{(l)},$$

welche maximiert werden soll. Dabei ist $e^{(l)} \in \{0,\ 1\}$ genau dann 1, wenn die Box X_l vollstaendig leer, d.h. unbenutzt ist. Somit wird die Anzahl der verwendeten Boxen, wie gefordert, minimiert.
Allerdings muss nun die vorherige Zielfunktion durch eine Bedingung (lineare Ungleichung) ersetzt werden:

$$\sum_{l=1}^{k} \#(X_l) \geq \#(A) - n_0$$

Dadurch, dass n_0 das Minimum der uebrigen Kleidungsstuecke ist, ist $\#(A) - n_0$ das Maximum der benutzten Kleidungsstuecke. Durch diese Bedingung wird somit implizit die Anzahl der benutzten Kleidungsstuecke auf ihr Maximum $\#(A) - n_0$ gesetzt.

2. Teil
Hier muss Loesungsvorschlag II des Kapitels 5 ausschlieszlich um eine Bedingung ergaenzt werden:

$$\sum_{l=1}^{k} 1 - e^{(l)} = k - \sum_{l=1}^{k} e^{(l)} \leq D$$

Da $e^{(l)} \in \{0,\ 1\}$ genau dann 1 ist, wenn die Box X_l vollstaendig leer, d.h. unbenutzt ist, wird dadruch die Anzahl an Boxen durch D nach oben beschraenkt.

Eine weitere Loesung dieses Problems besteht in einer Anpassung des Greedy-Verfahrens aus Kapitel 5.4. Denn fuer dieses laesst sich die Anzahl der verwendeten Boxen einfach beschraenken, indem der Algorithmus nach dieser Anzahl an Boxen terminiert, falls er dies nicht vorher getan hat. Dies kann durch eine einfache Zaehlvariable umgesetzt werden.

7.2. Erweiterung II

1. Teil
Analog zum Verfahren fuer Erweiterung I wird das ILP-Verfahren aus Kapitel 6 so angepasst, wobei die Minimalanzahl der uebrigen Kleidungsstuecke n_0 bereits bekannt ist. Dazu wird zunaechst eine neue Zielfunktion definiert durch

$$f \coloneqq \sum_{l=1}^{k} e^{(l)},$$

welche maximiert werden soll, denn dadurch wird, wie zuvor, die Anzahl der leeren Boxen maximiert, sodass die Anzahl der benutzten Boxen minimiert wird.
Weiter wird eine Bedingung hinzugefuegt, die gewaehrleistet, dass die bereits bekannte Minimalanzahl an uebrigen Kleidungsstuecken n_0 erreicht wird:

$$\sum_{l=1}^{k} \#(X_l) \geq \#(A) - n_0,$$

wobei

$$X_l \coloneqq Z^{(l)}_{\phi(l)} + \cdots + Z^{(l)}_{\phi(l)+r-1}.$$

2. Teil
Auch hier muss Loesungsvorschlag II des Kapitels 6 ausschlieszlich um die Bedingung

$$\sum_{l=1}^{k} 1 - e^{(l)} = k - \sum_{l=1}^{k} e^{(l)} \leq D$$

ergaenzt werden, da hier $e^{(l)} \in \{0,\ 1\}$ genau dann 1 ist, wenn

$$X_l \coloneqq Z^{(l)}_{\phi(l)} + \cdots + Z^{(l)}_{\phi(l)+r-1}$$

vollstaendig leer, d.h. unbenutzt ist, wird dadurch die Anzahl an Boxen durch D nach oben beschraenkt.

Auch fuer diese Variante besteht eine weitere Loesung in der Anpassung des Greedy-Verfahrens aus Kapitel 6.3. Dazu wird ebenfalls das Programm spaetestens terminiert, wenn die vorgegebene Anzahl an Boxen erreicht wurde.

Eine Analyse der Zeitkomplexitaet wird fuer dieses Kapitel nicht durchgefuehrt, da die Loesungen sich kaum von denen der vorherigen Kapiteln unterscheiden.

8. Implementierung

Die vorgestellte Loesungen wurden in C++ 17 und Python 3 implementiert.

In Python 3 wurden die Loesungen implementiert, die ein ILP beinhalten. Der Code dazu besteht je aus einer Python-Datei, welche in zwei bis drei Teile unterteilt ist. Dies sind (1) das Einlesen, Aufrufen der anderen Teile und Ausgeben, (2) die Berechnung aller maximaler Cliquen, und (3) das ganzzahligen lineare Programm. Konkret wurden die ganzzahligen linearen Programme in Python mit Hilfe des Python Pakets »PuLP« modelliert und geloest. Genauer wurde intern »CBC« (»COIN-OR branch and cut«) als Solver verwendet.[6] Dieses Python Paket ermoeglicht es Variablen, Bedingungen (in Form der Linearen Ungleichungen) und die Zielfunktion, die maximiert oder minimiert wird, zu definieren und anschlieszend mit einem bestimmtem Solver (hier CBC) das Ganzzahlige Optimierungsproblem optimal zu loesen. Fuer sehr grosze Instanzen kann es allerdings sinnvoll sein, ein Zeitlimit festzulegen, sodass das Programm nach einer gegebenen Zeit beendet wird und ggf. die bis dahin beste Loesung ausgegeben wird. Dabei besteht auch die Moeglichkeit das in der vorgegebenen Zeit keine Loesung gefunden werden kann.

Die Loesungsvorschlaege, die nach dem Greedy-Verfahren funktionieren, wurden in C++ 20 implementiert. Dabei wurde ebenfalls erneut die Berechnung der maximalen Cliquen in C++ implementiert. Die C++ Programme wurde mit Verwendung von »CMake«(cross-platform make) [7] erstellt und sind in mehrere Datein und Ordner unterteilt. Darunter die Berechnung aller maximaler Cliquen in einem Graphen, eine Klasse, welche Matrizen in einem eindimensionalen vector modelliert, und in der Datei, die die main-Funktion enthaelt, die geriege Berechnung der Boxen.

Genauer wurden alle vorgestellten Loesungsverfahren einzeln implementiert. Leider konnten Erweiterung II, sowie die zeite Variante der next-Funktion aus dem Greedy-Verfahren der ersten Erweiterung, nicht implementiert werden :(

1. Erweiterung I
   (a) 1. ILP
   (b) 2. ILP
   (c) Greedy-Verfahren
2. Erweiterung III
   (a) Erweiterung I
      i. 1. Teil (ILP)

Diese Struktur findet sich auch im Ordner «Quellcode», welcher alle Implementierungen in C++ und Python vollstaendig enthaelt (beigefuegt ist fuer die C++ Programme stets auch ein ausfuehbares Programm), sowie im Kapitel 10, welches die wichtigsten Teile des Quellcodes beinhaltet.

[6] Bei beiden handelt es sich um Open-Source Projekte der »Computational Infrastructure for Operations Research«
[7] Bei CMake handelt es sich um Open-Source Software Build System fuer C und C++

9. Beispiele und Evaluation der Loesungsvorschlaege

Nun sollen zu den vorgestellten Problemen Beispiele betrachtet werden. Dazu werden die Beispieleingaben der BwInf-Website genutzt. Fuer diese Beispiele werden die gefundenen Loesungen vollstaendig angegeben.
Genauer werden zu allen Beispieleingaben der BwInf-Website die optimale Loesung angegeben. Im anschliesszenden Kapitel werden die Ergebnisse des Greedy-Verfahrens angegben aber nicht vollstaendig dargestellt. Die vollstaendigen Loesungen dazu sind in den Datein zu finden. Ebenfalls wurden die Ergbenisse zum 1. Teil der Erweiterung III angegben aber nicht vollstaendig dargestellt.

9.1. Erweiterung I: ILP-Verfahren

Beispiele der BwInf-Website

In diesem Kapitel sollen die Loesungen der Beispieleingaben der BwInf-Website behandelt wurden. Das verbesserte ILP konnte diese alle vollstaendig loesen.
paeckchen0.txt:
Die Matrix A sieht in diesem Beispiel wie folgt aus:

$$A = \begin{pmatrix} 3 & 2 \\ 0 & 3 \\ 0 & 3 \end{pmatrix}$$

In diesem Fall ist die Menge der maximalen Cliquen gegeben durch $\{\{0,\ 1\}\}$ Das erste ILP-Verfahren liefert nach 0.008s diese Loesung bestehend aus 2 Boxen:

$$\begin{pmatrix} 0 & 2 \\ 0 & 1 \\ 0 & 1 \end{pmatrix},\ \begin{pmatrix} 3 & 0 \\ 0 & 2 \\ 0 & 2 \end{pmatrix}$$

Das zweite ILP-Verfahren liefert nach 0.007s diese Loesung bestehend aus 3 Boxen:

$$\begin{pmatrix} 0 & 2 \\ 0 & 1 \\ 0 & 1 \end{pmatrix},\ \begin{pmatrix} 2 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix},\ \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix}$$

Beide Verfahren liefern also eine Loesung mit 0 uebrigen Kleidungsstuecken!

Der erste Teil der Erweiterung III liefert die folgende Loesung:

paeckchen1.txt:
In diesem Beispiel sieht A wie folgt aus:

$$\begin{pmatrix} 48 & 55 & 76 & 27 \\ 39 & 46 & 29 & 37 \\ 57 & 38 & 28 & 19 \end{pmatrix}$$

Die Menge der maximalen Cliquen sieht in diesem Beispiel wie folgt aus:

$$\Big\{\{2,\ 3\}, \{1,\ 2\}, \{0,\ 1\}\Big\}$$

Das 1. ILP-Verfahren liefert nach 2.067s die folgende Loesung mit 73 Boxen:

$$\begin{pmatrix}0&3&0&0\\0&1&0&0\\3&0&0&0\end{pmatrix},\begin{pmatrix}0&3&0&0\\1&0&0&0\\0&3&0&0\end{pmatrix},\begin{pmatrix}3&0&0&0\\3&0&0&0\\3&0&0&0\end{pmatrix},\begin{pmatrix}0&0&0&3\\0&0&0&1\\0&0&0&3\end{pmatrix},\begin{pmatrix}0&3&0&0\\0&3&0&0\\0&0&3&0\end{pmatrix},\begin{pmatrix}0&0&3&0\\0&0&0&2\\0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&3&0\\0&0&0&3\\0&0&1&0\end{pmatrix},$$
$$\begin{pmatrix}0&3&0&0\\3&0&0&0\\3&0&0&0\end{pmatrix},\begin{pmatrix}3&0&0&0\\0&3&0&0\\3&0&0&0\end{pmatrix},\begin{pmatrix}0&0&0&3\\0&0&3&0\\0&0&1&0\end{pmatrix},\begin{pmatrix}3&0&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}0&0&0&3\\0&0&0&3\\0&0&0&1\end{pmatrix},\begin{pmatrix}0&3&0&0\\0&1&0&0\\0&3&0&0\end{pmatrix},\begin{pmatrix}0&0&3&0\\0&2&0&0\\0&0&1&0\end{pmatrix},$$
$$\begin{pmatrix}0&0&3&0\\0&0&0&3\\0&0&0&1\end{pmatrix},\begin{pmatrix}3&0&0&0\\3&0&0&0\\3&0&0&0\end{pmatrix},\begin{pmatrix}0&0&3&0\\0&0&1&0\\0&0&0&1\end{pmatrix},\begin{pmatrix}0&3&0&0\\0&0&1&0\\0&0&1&0\end{pmatrix},\begin{pmatrix}0&0&3&0\\0&0&1&0\\0&1&0&0\end{pmatrix},\begin{pmatrix}0&0&0&3\\0&0&0&3\\0&0&3&0\end{pmatrix},\begin{pmatrix}3&0&0&0\\0&3&0&0\\3&0&0&0\end{pmatrix},$$
$$\begin{pmatrix}3&0&0&0\\1&0&0&0\\0&3&0&0\end{pmatrix},\begin{pmatrix}0&3&0&0\\0&0&1&0\\0&0&1&0\end{pmatrix},\begin{pmatrix}3&0&0&0\\1&0&0&0\\3&0&0&0\end{pmatrix},\begin{pmatrix}0&3&0&0\\3&0&0&0\\0&1&0&0\end{pmatrix},\begin{pmatrix}0&0&3&0\\0&3&0&0\\0&0&3&0\end{pmatrix},\begin{pmatrix}0&0&1&0\\0&0&3&0\\0&3&0&0\end{pmatrix},\begin{pmatrix}0&3&0&0\\1&0&0&0\\2&0&0&0\end{pmatrix},$$
$$\begin{pmatrix}3&0&0&0\\0&3&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}0&0&3&0\\0&1&0&0\\0&3&0&0\end{pmatrix},\begin{pmatrix}0&3&0&0\\2&0&0&0\\0&1&0&0\end{pmatrix},\begin{pmatrix}0&0&0&3\\0&0&0&1\\0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&3&0\\0&0&2&0\\0&0&0&1\end{pmatrix},\begin{pmatrix}3&0&0&0\\0&3&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}1&0&0&0\\1&0&0&0\\2&1&0&0\end{pmatrix},$$
$$\begin{pmatrix}0&0&3&0\\0&0&0&3\\0&0&0&1\end{pmatrix},\begin{pmatrix}0&3&0&0\\3&0&0&0\\0&3&0&0\end{pmatrix},\begin{pmatrix}0&0&3&0\\0&0&3&0\\0&1&0&0\end{pmatrix},\begin{pmatrix}1&0&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}0&0&3&0\\0&3&0&0\\0&0&1&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\1&0&0&0\\3&0&0&0\end{pmatrix},\begin{pmatrix}0&0&3&0\\0&0&0&3\\0&0&0&1\end{pmatrix},$$
$$\begin{pmatrix}0&3&0&0\\3&0&0&0\\3&0&0&0\end{pmatrix},\begin{pmatrix}0&0&0&3\\0&0&0&3\\0&0&1&0\end{pmatrix},\begin{pmatrix}3&0&0&0\\0&3&0&0\\3&0&0&0\end{pmatrix},\begin{pmatrix}0&0&3&0\\0&0&1&0\\0&0&3&0\end{pmatrix},\begin{pmatrix}0&0&3&0\\0&3&0&0\\0&1&0&0\end{pmatrix},\begin{pmatrix}0&0&0&3\\0&0&0&3\\0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&1&0\\0&0&0&1\\0&0&0&1\end{pmatrix},$$
$$\begin{pmatrix}0&0&3&0\\0&0&0&3\\0&0&0&3\end{pmatrix},\begin{pmatrix}0&3&0&0\\0&0&1&0\\0&1&0&0\end{pmatrix},\begin{pmatrix}0&0&3&0\\0&0&0&3\\0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&2&0\\0&3&0&0\\0&0&1&0\end{pmatrix},\begin{pmatrix}0&3&0&0\\0&0&1&0\\0&0&1&0\end{pmatrix},\begin{pmatrix}0&0&3&0\\0&0&1&0\\0&1&0&0\end{pmatrix},\begin{pmatrix}0&3&0&0\\1&0&0&0\\3&0&0&0\end{pmatrix},$$
$$\begin{pmatrix}0&0&3&0\\0&0&0&2\\0&0&1&0\end{pmatrix},\begin{pmatrix}0&0&3&0\\0&0&1&0\\0&0&3&0\end{pmatrix},\begin{pmatrix}3&0&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}3&0&0&0\\1&0&0&0\\0&1&0&0\end{pmatrix},\begin{pmatrix}0&0&3&0\\0&3&0&0\\0&3&0&0\end{pmatrix},\begin{pmatrix}0&0&0&3\\0&0&3&0\\0&0&3&0\end{pmatrix},\begin{pmatrix}1&0&0&0\\3&0&0&0\\0&3&0&0\end{pmatrix},$$
$$\begin{pmatrix}0&3&0&0\\3&0&0&0\\0&3&0&0\end{pmatrix},\begin{pmatrix}0&3&0&0\\0&3&0&0\\3&0&0&0\end{pmatrix},\begin{pmatrix}0&0&0&3\\0&0&1&0\\3&0&0&0\end{pmatrix},\begin{pmatrix}0&3&0&0\\1&0&0&0\\3&0&0&0\end{pmatrix},\begin{pmatrix}3&0&0&0\\0&3&0&0\\3&0&0&0\end{pmatrix},\begin{pmatrix}3&0&0&0\\0&2&0&0\\3&0&0&0\end{pmatrix},\begin{pmatrix}0&0&3&0\\0&0&3&0\\0&1&0&0\end{pmatrix},$$
$$\begin{pmatrix}0&0&3&0\\0&0&1&0\\0&1&0&0\end{pmatrix},\begin{pmatrix}0&0&3&0\\0&0&1&0\\0&0&0&1\end{pmatrix},\begin{pmatrix}3&0&0&0\\1&0&0&0\\3&0&0&0\end{pmatrix}$$

Das 2. ILP-Verfahren liefert nach 0.0988s eine Loesung mit 142 Boxen:

$$\begin{pmatrix}0&0&1&0\\0&0&1&0\\0&0&1&0\end{pmatrix},\begin{pmatrix}0&0&1&0\\0&0&0&1\\0&0&1&0\end{pmatrix},\begin{pmatrix}0&0&0&1\\0&0&0&1\\0&0&1&0\end{pmatrix},\begin{pmatrix}0&0&0&1\\0&0&1&0\\0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&3&0\\0&0&0&1\\0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&1\\0&0&0&1\\0&0&1&0\end{pmatrix},\begin{pmatrix}0&0&0&3\\0&0&1&0\\0&0&1&0\end{pmatrix},$$
$$\begin{pmatrix}0&0&1&0\\0&0&0&1\\0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&1\\0&0&0&1\\0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&1&0\\0&0&0&1\\0&0&1&0\end{pmatrix},\begin{pmatrix}0&0&1&0\\0&0&0&1\\0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&1\\0&0&0&1\\0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&1&0\\0&0&0&3\\0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&1&0\\0&0&0&3\\0&0&0&1\end{pmatrix},$$
$$\begin{pmatrix}0&0&0&3\\0&0&0&1\\0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&1&0\\0&0&0&1\\0&0&1&0\end{pmatrix},\begin{pmatrix}0&0&1&0\\0&0&0&1\\0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&1&0\\0&0&0&1\\0&0&1&0\end{pmatrix},\begin{pmatrix}0&0&1&0\\0&0&0&1\\0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&1\\0&0&0&1\\0&0&1&0\end{pmatrix},\begin{pmatrix}0&0&1&0\\0&0&1&0\\0&0&1&0\end{pmatrix},$$
$$\begin{pmatrix}0&0&0&3\\0&0&0&1\\0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&1\\0&0&0&1\\0&0&1&0\end{pmatrix},\begin{pmatrix}0&0&1&0\\0&0&0&1\\0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&3\\0&0&0&3\\0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&1\\0&0&0&1\\0&0&1&0\end{pmatrix},\begin{pmatrix}0&0&0&3\\0&0&0&1\\0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&1&0\\0&0&1&0\\0&0&1&0\end{pmatrix},$$
$$\begin{pmatrix}0&0&1&0\\0&0&0&1\\0&0&1&0\end{pmatrix},\begin{pmatrix}0&0&0&1\\0&0&0&1\\0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&1&0\\0&0&0&1\\0&0&1&0\end{pmatrix},\begin{pmatrix}0&0&1&0\\0&0&1&0\\0&0&1&0\end{pmatrix},\begin{pmatrix}0&0&3&0\\0&0&0&1\\0&0&1&0\end{pmatrix},\begin{pmatrix}0&0&1&0\\0&0&0&2\\0&0&0&1\end{pmatrix},\begin{pmatrix}0&1&0&0\\0&1&0&0\\0&1&0&0\end{pmatrix},$$
$$\begin{pmatrix}0&0&1&0\\0&1&0&0\\0&1&0&0\end{pmatrix},\begin{pmatrix}0&0&3&0\\0&0&1&0\\0&1&0&0\end{pmatrix},\begin{pmatrix}0&0&1&0\\0&0&1&0\\0&1&0&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\0&1&0&0\\0&1&0&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\0&0&1&0\\0&1&0&0\end{pmatrix},\begin{pmatrix}0&0&3&0\\0&1&0&0\\0&1&0&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\0&0&1&0\\0&1&0&0\end{pmatrix},$$
$$\begin{pmatrix}0&1&0&0\\0&1&0&0\\0&0&1&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\0&1&0&0\\0&0&1&0\end{pmatrix},\begin{pmatrix}0&0&3&0\\0&1&0&0\\0&0&1&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\0&0&1&0\\0&1&0&0\end{pmatrix},\begin{pmatrix}0&0&1&0\\0&1&0&0\\0&1&0&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\0&0&1&0\\0&1&0&0\end{pmatrix},\begin{pmatrix}0&0&3&0\\0&0&1&0\\0&1&0&0\end{pmatrix},$$
$$\begin{pmatrix}0&1&0&0\\0&1&0&0\\0&0&1&0\end{pmatrix},\begin{pmatrix}0&0&1&0\\0&0&1&0\\0&1&0&0\end{pmatrix},\begin{pmatrix}0&0&1&0\\0&0&1&0\\0&1&0&0\end{pmatrix},\begin{pmatrix}0&0&1&0\\0&1&0&0\\0&0&1&0\end{pmatrix},\begin{pmatrix}0&0&1&0\\0&0&1&0\\0&1&0&0\end{pmatrix},\begin{pmatrix}0&0&1&0\\0&1&0&0\\0&0&1&0\end{pmatrix},\begin{pmatrix}0&0&3&0\\0&0&1&0\\0&1&0&0\end{pmatrix},$$
$$\begin{pmatrix}0&1&0&0\\0&1&0&0\\0&0&1&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\0&0&1&0\\0&0&1&0\end{pmatrix},\begin{pmatrix}0&0&1&0\\0&1&0&0\\0&1&0&0\end{pmatrix},\begin{pmatrix}0&0&3&0\\0&1&0&0\\0&1&0&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\0&0&1&0\\0&1&0&0\end{pmatrix},\begin{pmatrix}0&3&0&0\\0&0&1&0\\0&1&0&0\end{pmatrix},\begin{pmatrix}0&0&3&0\\0&0&1&0\\0&1&0&0\end{pmatrix},$$
$$\begin{pmatrix}0&1&0&0\\0&0&1&0\\0&1&0&0\end{pmatrix},\begin{pmatrix}0&0&3&0\\0&1&0&0\\0&0&1&0\end{pmatrix},\begin{pmatrix}0&0&3&0\\0&1&0&0\\0&1&0&0\end{pmatrix},\begin{pmatrix}0&0&3&0\\0&0&1&0\\0&1&0&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\0&1&0&0\\0&0&1&0\end{pmatrix},\begin{pmatrix}0&0&2&0\\0&0&1&0\\0&1&0&0\end{pmatrix},\begin{pmatrix}0&0&3&0\\0&0&1&0\\0&0&1&0\end{pmatrix},$$
$$\begin{pmatrix}1&0&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}3&0&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}0&3&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}1&0&0&0\\0&1&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}1&0&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\0&1&0&0\\1&0&0&0\end{pmatrix},$$
$$\begin{pmatrix}0&1&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}1&0&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}1&0&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}1&0&0&0\\0&1&0&0\\1&0&0&0\end{pmatrix},$$
$$\begin{pmatrix}1&0&0&0\\0&1&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\0&1&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\1&0&0&0\\0&1&0&0\end{pmatrix},\begin{pmatrix}1&0&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}1&0&0&0\\0&1&0&0\\1&0&0&0\end{pmatrix},$$
$$\begin{pmatrix}1&0&0&0\\0&1&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}2&0&0&0\\0&1&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\0&1&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}3&0&0&0\\1&0&0&0\\0&1&0&0\end{pmatrix},$$
$$\begin{pmatrix}0&1&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}3&0&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}1&0&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}3&0&0&0\\0&1&0&0\\1&0&0&0\end{pmatrix},$$
$$\begin{pmatrix}3&0&0&0\\0&1&0&0\\0&1&0&0\end{pmatrix},\begin{pmatrix}1&0&0&0\\0&1&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}3&0&0&0\\0&1&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}1&0&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}1&0&0&0\\0&1&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}1&0&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}1&0&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},$$
$$\begin{pmatrix}1&0&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}1&0&0&0\\0&1&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\1&0&0&0\\0&1&0&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\0&1&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},$$

$$\begin{pmatrix}1&0&0&0\\0&1&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}1&0&0&0\\0&1&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\0&1&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}1&0&0&0\\0&1&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\0&1&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},$$
$$\begin{pmatrix}0&1&0&0\\0&1&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}1&0&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}1&0&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\0&1&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}1&0&0&0\\0&1&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},$$
$$\begin{pmatrix}1&0&0&0\\0&1&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}1&0&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}1&0&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}1&0&0&0\\0&1&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},$$
$$\begin{pmatrix}3&0&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix},\begin{pmatrix}0&1&0&0\\1&0&0&0\\1&0&0&0\end{pmatrix}$$

Beide Loesungen erreichen 499 Kleidungsstueckem, bzw. 0 uebrige Kleidungsstuecke.

paeckchen2.txt:
In diesem Beispiel ist

$$A=\begin{pmatrix}1&1&1&0&0&1&2&0&0\\0&1&0&2&2&3&4&1&0\\0&3&0&0&3&0&1&2&4\end{pmatrix}$$

Die Menge der maximalen Cliquen ist gegeben durch

$$Q=\Big\{\{8,1,0,7\},\{8,2,6\},\{8,1,2,0\},\{0,1,5,7\},\{0,3,4\},\{0,1,2,4\}\Big\}$$

Das erste ILP erzeugt nach 0.066s eine Loesung mit 6 Boxen:

$$\begin{pmatrix}0&1&0&0&0&0&0&0&0\\0&0&0&0&0&3&0&0&0\\0&3&0&0&0&0&0&0&0\end{pmatrix},\begin{pmatrix}0&0&0&0&0&0&1&0&0\\0&0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&0&3\end{pmatrix},\begin{pmatrix}0&0&1&0&0&0&0&0&0\\0&0&0&0&1&0&0&0&0\\0&0&0&0&2&0&0&0&0\end{pmatrix},$$
$$\begin{pmatrix}0&0&0&0&0&1&0&0&0\\0&1&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&2&0\end{pmatrix},\begin{pmatrix}0&0&0&0&0&0&1&0&0\\0&0&0&0&0&0&3&0&0\\0&0&0&0&0&0&1&0&1\end{pmatrix},\begin{pmatrix}1&0&0&0&0&0&0&0&0\\0&0&0&2&1&0&0&0&0\\0&0&0&0&1&0&0&0&0\end{pmatrix}$$

Das zweite ILP erzeugt nach einer Laufzeit von 0.028s das folgende Ergebnis mit ebenfalls 6 Boxen:

$$\begin{pmatrix}0&1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&1&0\\0&2&0&0&0&0&0&0&0\end{pmatrix},\begin{pmatrix}0&0&0&0&0&0&1&0&0\\0&0&0&0&0&0&1&0&0\\0&0&0&0&0&0&1&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&0&1&0&0\\0&0&0&0&0&0&3&0&0\\0&0&0&0&0&0&0&0&3\end{pmatrix},$$
$$\begin{pmatrix}0&0&0&0&0&1&0&0&0\\0&0&0&0&0&3&0&0&0\\0&0&0&0&0&0&0&2&0\end{pmatrix},\begin{pmatrix}1&0&0&0&0&0&0&0&0\\0&0&0&2&0&0&0&0&0\\0&0&0&0&3&0&0&0&0\end{pmatrix},\begin{pmatrix}0&0&1&0&0&0&0&0&0\\0&0&0&0&2&0&0&0&0\\0&1&0&0&0&0&0&0&0\end{pmatrix}$$

Erneut sorgen beide Loesungen fuer 0 uebrige Kleidungsstuecke, bzw. 32 genutzte Kleidungsstuecke.

paeckchen3.txt:
In diesem Beispiel gilt

$$A=\begin{pmatrix}4&2&0&3&0&0&3&4&2&0\\4&4&0&0&11&1&1&2&1&1\\2&1&2&6&0&2&0&0&2&1\\2&6&1&0&6&1&0&0&4&0\\0&1&1&5&0&3&0&4&2&1\end{pmatrix}$$

Weiter gilt

$$Q=\Big\{\{5,6,7,8,9\},\{8,2,6\},\{1,5,6,7\},\{0,1,5,7\},\{1,2,6\},\{3,4\}\Big\}$$

Fuer dieses und alle weiteren Beispieleingaben benoetigt das erste ILP Verfahren zu viel Zeit, sodass hier nur die Ergebnisse des verbesserten ILP Verfahrens aufgefuehrt werden. Dieses hat fuer diesen Beispiel nach 0.0343s

eine Loesung mit 13 Boxen geliefert.

$$\begin{pmatrix} 0&0&0&0&0&0&0&3&0&0\\ 0&0&0&0&0&1&0&1&0&0\\ 0&0&0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&0&1&1 \end{pmatrix}, \begin{pmatrix} 0&0&0&0&0&0&1&0&1&0\\ 0&0&0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&1&0&0&0&0 \end{pmatrix}, \begin{pmatrix} 0&0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&1&0&0&0&0 \end{pmatrix},$$

$$\begin{pmatrix} 0&0&0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0&0&1&0\\ 0&0&1&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&0&1&0 \end{pmatrix}, \begin{pmatrix} 0&0&0&0&0&0&1&0&0&0\\ 0&2&0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0&0&0\\ 0&2&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&1&0&0&0&0 \end{pmatrix}, \begin{pmatrix} 2&1&0&0&0&0&0&0&0&0\\ 1&1&0&0&0&0&0&0&0&0\\ 1&0&0&0&0&0&0&0&0&0\\ 1&1&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&1&0&0 \end{pmatrix},$$

$$\begin{pmatrix} 0&0&0&0&0&0&0&1&0&0\\ 1&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&0&1&0&0 \end{pmatrix}, \begin{pmatrix} 1&0&0&0&0&0&0&0&0&0\\ 1&0&0&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0&0&0\\ 1&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&1&0&0 \end{pmatrix}, \begin{pmatrix} 1&0&0&0&0&0&0&0&0&0\\ 1&0&0&0&0&0&0&0&0&0\\ 1&0&0&0&0&0&0&0&0&0\\ 0&3&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&1&0&0 \end{pmatrix},$$

$$\begin{pmatrix} 0&1&0&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0&0&0\\ 0&1&1&0&0&0&0&0&0&0 \end{pmatrix}, \begin{pmatrix} 0&0&0&1&0&0&0&0&0&0\\ 0&0&0&0&3&0&0&0&0&0\\ 0&0&0&1&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0&0&0 \end{pmatrix}, \begin{pmatrix} 0&0&0&1&0&0&0&0&0&0\\ 0&0&0&0&3&0&0&0&0&0\\ 0&0&0&3&0&0&0&0&0&0\\ 0&0&0&0&3&0&0&0&0&0\\ 0&0&0&1&0&0&0&0&0&0 \end{pmatrix},$$

$$\begin{pmatrix} 0&0&0&1&0&0&0&0&0&0\\ 0&0&0&0&3&0&0&0&0&0\\ 0&0&0&2&0&0&0&0&0&0\\ 0&0&0&0&2&0&0&0&0&0\\ 0&0&0&3&0&0&0&0&0&0 \end{pmatrix}$$

Anders als die vorherigen Loesungen bleiben hier nicht 0, sondern 2 von 96 Kleidungsstuecken uebrig. Die Matriz L sieht wie folgt aus:

$$L = \begin{pmatrix} 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&2&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0 \end{pmatrix}$$

paeckchen4.txt:
Nun gilt

$$A = \begin{pmatrix} 0&0&1&0&0&3&0&0&0&23&0&6&0&0&1&0&9&4&2&0&0&0&0&10\\ 0&0&0&10&7&0&0&0&0&11&2&6&9&0&0&7&0&0&0&0&0&4&0&0\\ 0&0&0&11&8&0&0&0&0&0&3&5&9&0&0&6&0&0&0&0&15&5&0&0\\ 0&0&0&9&8&0&0&0&0&0&3&6&12&0&0&9&0&0&0&0&11&6&0&0\\ 0&0&0&8&7&0&0&0&0&0&2&2&11&0&0&5&0&0&0&0&0&6&20&0\\ 0&0&0&10&7&0&0&0&0&0&3&3&11&0&0&6&0&0&0&0&16&4&0&0\\ 12&3&0&0&0&0&18&2&3&0&0&0&0&1&0&0&0&2&0&14&0&0&20&0 \end{pmatrix}$$

sowie

$$Q = \Big\{\{0,23,22,15\},\{23,4,6,22\},\{9,20,22\},\{8,2,22,15\},\{9,11,21,7\},\{11,18,19,20\},\\ \{17,19,20,12\},\{17,10,20,14\},\{9,11,20\},\{16,9,3,1\},\{9,13,3,5\}\Big\}$$

Mit einer Laufzeit von 0.0545s erreicht das Loesungsverfahren eine Loesung mit 0 uebrigen Kleidungsstuecken bestehend aus 35 Boxen:

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&2&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
3&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{pmatrix},$$

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
3&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{pmatrix},$$

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
3&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{pmatrix},$$

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
3&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{pmatrix},$$

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{pmatrix},$$

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{pmatrix},$$

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&2&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{pmatrix},$$

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{pmatrix}$$

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&0&2&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{pmatrix},$$

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&2&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{pmatrix},$$

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&3&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&3&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&3&0
\end{pmatrix},$$

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0
\end{pmatrix},$$

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&3&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&3&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&2&0
\end{pmatrix},$$

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0
\end{pmatrix},$$

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&3&0
\end{pmatrix},$$

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&3&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0
\end{pmatrix}$$

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&3&0
\end{pmatrix},$$

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0
\end{pmatrix},$$

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&3&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0
\end{pmatrix},$$

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&3&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0
\end{pmatrix},$$

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&3&0
\end{pmatrix},$$

$$\begin{pmatrix}
0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&2&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&2&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{pmatrix},$$

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&3&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&3&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&3&0&0\\
0&0&0&0&0&0&0&0&0&0&0&2&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{pmatrix},$$

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&3&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&2&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&3&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&3&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&3&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{pmatrix}$$

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&3&0&0&0&0
\end{pmatrix},$$

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&3&0&0&0&0
\end{pmatrix},$$

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&2&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&2&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&3&0&0&0&0
\end{pmatrix},$$

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&2&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0
\end{pmatrix},$$

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&3&0&0&0&0
\end{pmatrix},$$

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&2&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0
\end{pmatrix},$$

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&2&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&2&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&2&0&0&0&0&0&0&0&0
\end{pmatrix},$$

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0&0\\
0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&2&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{pmatrix},$$

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0\\
0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{pmatrix},$$

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{pmatrix},$$

$$\begin{pmatrix}
0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&2&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0
\end{pmatrix}$$

paeckchen5.txt:
Es gilt

$$A = \begin{pmatrix}
2&0&0&0&0&0&0&9&0&0&0&0&7&0&4&0&0&4&5&0&3\\
7&0&5&0&0&0&0&0&3&0&0&0&0&0&8&3&0&4&0&2&0\\
0&3&0&6&2&0&0&0&0&12&0&0&0&0&7&0&0&0&0&1&7\\
0&5&0&0&0&0&0&0&0&0&9&0&0&1&3&0&0&4&6&0&9\\
0&6&0&0&0&7&3&0&0&0&0&8&0&0&0&0&1&0&6&2&0
\end{pmatrix}$$

sowie

$$Q = \Big\{\{16,10,20,15\},\{19,12,13,7\},\{8,9,18,3\},\{17,11,4,5\},\{17,4,5,6\},\{10,11,14,7\},\{2,3,14,6\},\{0,1,2\}\Big\}$$

Das ILP-Verfahren erhaelt eine optimale Loesung mit 19 Boxen nach 0.026s:

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&3\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&3&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&3\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&3\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0
\end{pmatrix},$$

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&2&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&2&0
\end{pmatrix},$$

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&3&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&3&0&0
\end{pmatrix},$$

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&3&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&2&0&0
\end{pmatrix},$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix},$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\begin{pmatrix} 0&0&0&0&0&0&0&0&0&0&0&0&0&0&2&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&2&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\ 0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \end{pmatrix},$$

$$\begin{pmatrix} 0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&3&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\ 0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \end{pmatrix},$$

$$\begin{pmatrix} 1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 3&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&3&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&3&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \end{pmatrix},$$

$$\begin{pmatrix} 1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 3&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&2&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&2&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&3&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \end{pmatrix}$$

Bei dieser Loesung bleiben 19 von 174 Kleidungsstuecken uebrig. Fuer die uebrigen Kleidungsstuecke gilt dann

$$L = \begin{pmatrix} 0&0&0&0&0&0&0&0&0&0&0&0&4&0&0&0&0&0&0&0&0\\ 1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&3&0&0&0&0&0&0&0&0&0&0&4\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&6\\ 0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \end{pmatrix}$$

paeckchen6.txt:
In diesem Beispiel gilt:

$$A = \begin{pmatrix} 17&0&0&0&15&59&38&33&0&0\\ 0&0&49&32&20&0&12&0&37&0\\ 59&51&0&0&0&0&13&11&38&0\\ 33&12&58&50&0&0&0&0&0&13\\ 0&0&0&0&0&0&0&0&0&120 \end{pmatrix}$$

und

$$Q = \Big\{\{8,9,6\},\{0,9,3,7\},\{0,2,4,5,9\},\{0,1,2,3,9\}\Big\}$$

Das ILP-Verfahren hat nach 0.114s eine Loesung mit 100 Boxen geliefert:

$$\begin{pmatrix}0&0&0&0&0&0&3&0&0&0\\0&0&0&0&0&0&3&0&0&0\\0&0&0&0&0&0&0&0&3&0\\0&0&0&0&0&0&0&0&0&1\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&0&3&0&0&0\\0&0&0&0&0&0&0&0&3&0\\0&0&0&0&0&0&1&0&2&0\\0&0&0&0&0&0&0&0&0&1\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&0&3&0&0&0\\0&0&0&0&0&0&3&0&0&0\\0&0&0&0&0&0&0&0&3&0\\0&0&0&0&0&0&0&0&0&1\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},$$
$$\begin{pmatrix}0&0&0&0&0&0&3&0&0&0\\0&0&0&0&0&0&3&0&0&0\\0&0&0&0&0&0&3&0&0&0\\0&0&0&0&0&0&0&0&0&1\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&0&3&0&0&0\\0&0&0&0&0&0&0&0&3&0\\0&0&0&0&0&0&0&0&3&0\\0&0&0&0&0&0&0&0&0&1\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&0&3&0&0&0\\0&0&0&0&0&0&0&0&3&0\\0&0&0&0&0&0&0&0&3&0\\0&0&0&0&0&0&0&0&0&1\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},$$
$$\begin{pmatrix}0&0&0&0&0&0&3&0&0&0\\0&0&0&0&0&0&0&0&3&0\\0&0&0&0&0&0&0&0&3&0\\0&0&0&0&0&0&0&0&0&1\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&0&2&0&0&0\\0&0&0&0&0&0&0&0&3&0\\0&0&0&0&0&0&3&0&0&0\\0&0&0&0&0&0&0&0&0&1\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&0&3&0&0&0\\0&0&0&0&0&0&0&0&3&0\\0&0&0&0&0&0&0&0&3&0\\0&0&0&0&0&0&0&0&0&1\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},$$
$$\begin{pmatrix}0&0&0&0&0&0&3&0&0&0\\0&0&0&0&0&0&0&0&3&0\\0&0&0&0&0&0&3&0&0&0\\0&0&0&0&0&0&0&0&0&1\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&0&0&1&0&0\\0&0&0&1&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&0&0&1&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&1&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},$$
$$\begin{pmatrix}0&0&0&0&0&0&0&1&0&0\\0&0&0&1&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&3&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&0&0&1&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&1&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&0&0&1&0&0\\0&0&0&1&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},$$
$$\begin{pmatrix}0&0&0&0&0&0&0&1&0&0\\0&0&0&1&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&3&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&0&0&1&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&1&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&0&0&1&0&0\\0&0&0&1&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},$$
$$\begin{pmatrix}0&0&0&0&0&0&0&1&0&0\\0&0&0&1&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&0&0&1&0&0\\0&0&0&3&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&0&0&1&0&0\\0&0&0&1&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},$$
$$\begin{pmatrix}0&0&0&0&0&0&0&3&0&0\\0&0&0&1&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&0&0&1&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&1&0&0\\0&0&0&3&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&0&0&1&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&1&0&0\\0&0&0&3&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},$$
$$\begin{pmatrix}0&0&0&0&0&0&0&1&0&0\\0&0&0&1&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&0&0&1&0&0\\0&0&0&3&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&0&0&1&0&0\\0&0&0&1&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},$$
$$\begin{pmatrix}0&0&0&0&0&0&0&3&0&0\\0&0&0&1&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&0&0&1&0&0\\0&0&0&3&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&0&0&1&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&1&0&0\\0&0&0&3&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},$$
$$\begin{pmatrix}0&0&0&0&0&0&0&1&0&0\\0&0&0&1&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&0&0&1&0&0\\0&0&0&3&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&0&0&1&0&0\\0&0&0&1&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},$$
$$\begin{pmatrix}0&0&0&0&0&0&0&3&0&0\\0&0&0&1&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&3&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&0&0&1&0&0\\0&0&0&1&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&0&0&1&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&0&0&1\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},$$

$$\begin{pmatrix}0&0&0&0&0&0&0&1&0&0\\0&0&0&1&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&0&0&1&0&0\\0&0&0&1&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&0&0&1&0&0\\0&0&0&1&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\3&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},$$

$$\begin{pmatrix}0&0&0&0&0&0&0&1&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&1&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&3\end{pmatrix},\begin{pmatrix}0&0&0&0&0&0&0&2&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&1&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&0&0&3&0&0\\0&0&0&1&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&3&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},$$

$$\begin{pmatrix}0&0&0&0&0&1&0&0&0&0\\0&0&0&0&1&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&1&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&3&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&3&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},$$

$$\begin{pmatrix}0&0&0&0&0&1&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&3&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&3\end{pmatrix},\begin{pmatrix}0&0&0&0&0&3&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},$$

$$\begin{pmatrix}0&0&0&0&0&1&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&3&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&1&0&0&0&0\\0&0&0&0&1&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&3&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},$$

$$\begin{pmatrix}0&0&0&0&0&1&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&1&0&0&0&0\\0&0&0&0&0&1&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&3&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&3&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&3\end{pmatrix},$$

$$\begin{pmatrix}0&0&0&0&0&1&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\3&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&1&0&0&0&0\\0&0&0&0&0&1&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&3&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&3\end{pmatrix},\begin{pmatrix}0&0&0&0&0&1&0&0&0&0\\0&0&0&0&0&1&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},$$

$$\begin{pmatrix}0&0&0&0&0&3&0&0&0&0\\0&0&3&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&3&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&3&0&0&0&0\\0&0&0&0&0&1&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&1&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},$$

$$\begin{pmatrix}0&0&0&0&0&3&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&1&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&3\end{pmatrix},\begin{pmatrix}0&0&0&0&1&0&0&0&0&0\\0&0&0&0&1&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},$$

$$\begin{pmatrix}0&0&0&0&0&1&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&1&0&0&0&0\\0&0&0&0&0&1&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&1&0&0&0&0\\0&0&3&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},$$

$$\begin{pmatrix}0&0&0&0&0&1&0&0&0&0\\0&0&0&0&1&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&1&0&0&0&0\\0&0&0&0&1&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&1&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},$$

$$\begin{pmatrix}0&0&0&0&0&1&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&3\end{pmatrix},\begin{pmatrix}0&0&0&0&0&1&0&0&0&0\\0&0&0&0&0&1&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&0&1&0&0&0\\0&0&0&0&0&0&1&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},$$

$$\begin{pmatrix}0&0&0&0&0&1&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&3&0&0&0&0\\0&0&0&0&0&1&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&1&0&0&0&0&0\\0&0&0&0&1&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},$$

$$\begin{pmatrix}0&0&0&0&1&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&2&0&0&0&0&0\\0&0&0&0&1&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&1&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},$$

$$\begin{pmatrix}0&0&0&0&1&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&2&0&0&0&0\\0&0&0&0&0&1&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&3&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&3\end{pmatrix},\begin{pmatrix}0&0&0&0&0&3&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\2&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},$$

$$\begin{pmatrix}0&0&0&0&0&3&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&3&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&3&0&0&0&0\\0&0&0&0&0&1&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&3&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}1&0&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&3&0&0&0&0&0&0&0&0\\0&0&3&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},$$

$$\begin{pmatrix}1&0&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&3&0&0&0&0&0&0&0&0\\0&0&3&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}1&0&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&3&0&0&0&0&0&0&0&0\\0&3&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}1&0&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&3&0&0&0&0&0&0&0&0\\0&0&3&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},$$

$$\begin{pmatrix}1&0&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&3&0&0&0&0&0&0&0&0\\0&3&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}1&0&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&3&0&0&0&0&0&0&0&0\\0&0&0&3&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}1&0&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&3&0&0&0&0&0&0&0&0\\3&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},$$

$$\begin{pmatrix}1&0&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&3&0&0&0&0&0&0&0&0\\0&0&3&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}1&0&0&0&0&0&0&0&0&0\\0&0&0&2&0&0&0&0&0&0\\0&3&0&0&0&0&0&0&0&0\\0&3&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}1&0&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&3&0&0&0&0&0&0&0&0\\0&0&0&3&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},$$

$$\begin{pmatrix}1&0&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&3&0&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}1&0&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&3&0&0&0&0&0&0&0&0\\0&2&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}1&0&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&3&0&0&0&0&0&0&0&0\\0&0&0&3&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},$$

$$\begin{pmatrix}1&0&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&3&0&0&0&0&0&0&0&0\\0&0&0&3&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}1&0&0&0&0&0&0&0&0&0\\0&0&2&0&0&0&0&0&0&0\\0&3&0&0&0&0&0&0&0&0\\0&0&0&3&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}1&0&0&0&0&0&0&0&0&0\\0&0&3&0&0&0&0&0&0&0\\0&3&0&0&0&0&0&0&0&0\\0&0&3&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&3\end{pmatrix},$$

$$\begin{pmatrix}1&0&0&0&0&0&0&0&0&0\\0&0&3&0&0&0&0&0&0&0\\0&3&0&0&0&0&0&0&0&0\\0&0&0&3&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&3\end{pmatrix}$$

Diese Loesung verwendet 748 von 770 Kleidungsstuecken. Somit sind 22 uerbig:

$$L = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 12 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

paeckchen7.txt:
Hier gilt

$$A = \begin{pmatrix} 17 & 0 & 0 & 0 & 0 & 59 & 38 & 33 & 0 & 0 \\ 0 & 0 & 49 & 32 & 20 & 0 & 0 & 0 & 37 & 0 \\ 59 & 51 & 0 & 0 & 0 & 0 & 0 & 11 & 38 & 0 \\ 33 & 0 & 58 & 50 & 0 & 0 & 0 & 0 & 0 & 13 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 102 \end{pmatrix}$$

und

$$Q = \Big\{\{8,9,6\},\{0,9,1,7\},\{0,9,4,5\},\{0,1,2,3,9\}\Big\}$$

Das Programm hat eine optimale Loesung mit 50 Boxen innerhalb von 0.045s geliefert:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \end{pmatrix},$$
$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$
$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$
$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \end{pmatrix},$$
$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$
$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \end{pmatrix},$$
$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

$$\begin{pmatrix}0&0&0&0&0&3&0&0&0&0\\0&0&0&0&1&0&0&0&0&0\\3&0&0&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&3\end{pmatrix},\begin{pmatrix}0&0&0&0&0&3&0&0&0&0\\0&0&0&0&1&0&0&0&0&0\\3&0&0&0&0&0&0&0&0&0\\3&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&3\end{pmatrix},\begin{pmatrix}0&0&0&0&0&3&0&0&0&0\\0&0&0&0&1&0&0&0&0&0\\3&0&0&0&0&0&0&0&0&0\\3&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},$$
$$\begin{pmatrix}0&0&0&0&0&3&0&0&0&0\\0&0&0&0&1&0&0&0&0&0\\3&0&0&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&3\end{pmatrix},\begin{pmatrix}0&0&0&0&0&3&0&0&0&0\\0&0&0&0&1&0&0&0&0&0\\3&0&0&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&2&0&0&0&0\\0&0&0&0&1&0&0&0&0&0\\3&0&0&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},$$
$$\begin{pmatrix}0&0&0&0&0&3&0&0&0&0\\0&0&0&0&1&0&0&0&0&0\\3&0&0&0&0&0&0&0&0&0\\3&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&3\end{pmatrix},\begin{pmatrix}0&0&0&0&0&3&0&0&0&0\\0&0&0&0&1&0&0&0&0&0\\3&0&0&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&3&0&0&0&0\\0&0&0&0&1&0&0&0&0&0\\3&0&0&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},$$
$$\begin{pmatrix}0&0&0&0&0&3&0&0&0&0\\0&0&0&0&1&0&0&0&0&0\\3&0&0&0&0&0&0&0&0&0\\3&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&3\end{pmatrix},\begin{pmatrix}0&0&0&0&0&3&0&0&0&0\\0&0&0&0&1&0&0&0&0&0\\2&0&0&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}0&0&0&0&0&3&0&0&0&0\\0&0&0&0&1&0&0&0&0&0\\3&0&0&0&0&0&0&0&0&0\\3&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&3\end{pmatrix},$$
$$\begin{pmatrix}1&0&0&0&0&0&0&0&0&0\\0&0&0&3&0&0&0&0&0&0\\0&3&0&0&0&0&0&0&0&0\\0&0&3&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&3\end{pmatrix},\begin{pmatrix}1&0&0&0&0&0&0&0&0&0\\0&0&3&0&0&0&0&0&0&0\\0&3&0&0&0&0&0&0&0&0\\0&0&3&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&3\end{pmatrix},\begin{pmatrix}1&0&0&0&0&0&0&0&0&0\\0&0&3&0&0&0&0&0&0&0\\0&3&0&0&0&0&0&0&0&0\\0&0&3&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&3\end{pmatrix},$$
$$\begin{pmatrix}1&0&0&0&0&0&0&0&0&0\\0&0&0&3&0&0&0&0&0&0\\0&3&0&0&0&0&0&0&0&0\\0&0&3&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&3\end{pmatrix},\begin{pmatrix}1&0&0&0&0&0&0&0&0&0\\0&0&0&3&0&0&0&0&0&0\\0&3&0&0&0&0&0&0&0&0\\0&0&3&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}1&0&0&0&0&0&0&0&0&0\\0&0&0&3&0&0&0&0&0&0\\0&3&0&0&0&0&0&0&0&0\\0&0&3&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&3\end{pmatrix},$$
$$\begin{pmatrix}1&0&0&0&0&0&0&0&0&0\\0&0&3&0&0&0&0&0&0&0\\0&3&0&0&0&0&0&0&0&0\\0&0&3&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},\begin{pmatrix}1&0&0&0&0&0&0&0&0&0\\0&0&3&0&0&0&0&0&0&0\\0&3&0&0&0&0&0&0&0&0\\0&0&3&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&3\end{pmatrix},\begin{pmatrix}1&0&0&0&0&0&0&0&0&0\\0&0&0&3&0&0&0&0&0&0\\0&3&0&0&0&0&0&0&0&0\\0&0&3&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},$$
$$\begin{pmatrix}1&0&0&0&0&0&0&0&0&0\\0&0&3&0&0&0&0&0&0&0\\0&3&0&0&0&0&0&0&0&0\\0&0&3&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&3\end{pmatrix},\begin{pmatrix}1&0&0&0&0&0&0&0&0&0\\0&0&0&3&0&0&0&0&0&0\\0&3&0&0&0&0&0&0&0&0\\0&0&3&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&3\end{pmatrix},\begin{pmatrix}1&0&0&0&0&0&0&0&0&0\\0&0&3&0&0&0&0&0&0&0\\0&3&0&0&0&0&0&0&0&0\\0&0&3&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix},$$
$$\begin{pmatrix}1&0&0&0&0&0&0&0&0&0\\0&0&0&3&0&0&0&0&0&0\\0&3&0&0&0&0&0&0&0&0\\0&0&3&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&3\end{pmatrix},\begin{pmatrix}1&0&0&0&0&0&0&0&0&0\\0&0&3&0&0&0&0&0&0&0\\0&3&0&0&0&0&0&0&0&0\\0&0&3&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\end{pmatrix}$$

Diese Loesung erreicht 569 von 700 Kleidungsstuecken. Somit bleiben 131 Kleidungsstuecke uebrig:

$$L=\begin{pmatrix}0&0&0&0&0&0&0&33&0&0\\0&0&28&2&0&0&0&0&0&0\\0&0&0&0&0&0&0&11&0&0\\0&0&7&50&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0\end{pmatrix}$$

9.2. Erweiterung I: Greedy-Verfahren

Die folgenden Ergebnisse beziehen sich auf das Greedy-Verfahren fuer erste Erweiterung mit der ersten Varianten der next-Funktion. Die vollstaendigen Loesungen sind in den Datein angegeben:

paecken0.txt:
Die gefundene Loesung verwendet 9 von 11 Kleidungsstuecke in einer Box (beste Loesung: 11). Laufzeit des Programms: 8 Mikrosekunden

paecken1.txt:
Die gefundene Loesung verwendet von 427 von Kleidungsstuecke in 48 Boxen (beste Loesung: 499). Laufzeit des Programms: 166 Mikrosekunden

paecken2.txt:
Die gefundene Loesung verwendet 23 von 32 Kleidungsstuecke in 3 Boxen (beste Loesung: 32). Laufzeit des Programms: 35 Mikrosekunden

paecken3.txt:
Die gefundene Loesung verwendet 78 von 96 Kleidungsstuecke in (beste Loesung: 94). Laufzeit des Programms: 74 Mikrosekunden

paecken4.txt:
Die gefundene Loesung verwendet 242 von 437 Kleidungsstuecke in 12 Boxen (beste Loesung: 437). Laufzeit des Programms: 325 Mikrosekunden

paecken5.txt:
Die gefundene Loesung verwendet 121 von 174 Kleidungsstuecke in 9 Boxen (beste Loesung: 155). Laufzeit des Programms: 163 Mikrosekunden

paecken6.txt:
Die gefundene Loesung verwendet 430 von 770 Kleidungsstuecke in 23 Boxen (beste Loesung: 748). Laufzeit des Programms: 230 Mikrosekunden

paecken7.txt:
Die gefundene Loesung verwendet 239 von 700 Kleidungsstuecke in 16 Boxen (beste Loesung: 569). Laufzeit des Programms: 103 Mikrosekunden

9.3. Erweiterung III

Beispiele der BwInf-Website

Mit Hilfe der im ersten Unterkapitel bestimmten optimalen Anzahl der benutzten Kleidungsstuecke, kann nun die minimale Anzahl der Boxen bestimmten werden, die diese Anzhal an Kleidungsstuecken erreicht.
Die konkreten Loesungen sind jeweils in den Datein angegeben.

paecken0.txt:
Die minimale Anzahl der Boxen fuer 0 uebrige Kleidungsstuecke ist 2 (vorher: 2). Laufzeit des Programms: 0.008s.

paecken1.txt:
Die minimale Anzahl der Boxen fuer 0 uebrige Kleidungsstuecke ist 69 (vorher: 73). Laufzeit des Programms: 0.314s.

paecken2.txt:
Die minimale Anzahl der Boxen fuer 0 uebrige Kleidungsstuecke ist 6 (vorher: 6). Laufzeit des Programms: 0.083s.

paecken3.txt:
Die minimale Anzahl der Boxen fuer 2 uebrige Kleidungsstuecke ist 8 (vorher: 13). Laufzeit des Programms: 0.319s.

paecken4.txt:
Die minimale Anzahl der Boxen fuer 0 uebrige Kleidungsstuecke ist 31 (vorher: 35). Laufzeit des Programms: 0.107s.

paecken5.txt:
Die minimale Anzahl der Boxen fuer 19 uebrige Kleidungsstuecke ist 14 (vorher: 19). Laufzeit des Programms: 0.026s.

paecken6.txt:
Die minimale Anzahl der Boxen fuer 22 uebrige Kleidungsstuecke ist 66 (voher: 100). Laufzeit des Programms: 0.371s.

paecken7.txt:
Die minimale Anzahl der Boxen fuer 131 uebrige Kleidungsstuecke ist 50 (vorher: 50). Laufzeit des Programms: 0.083s.

10. Quellcode

Nun folgt der Quellcode der wichtigsten Teile der Loesungen in C++ und Python. Dabei wurden Erweiterung I und Erweiterung III implementiert.

10.1. Erweiterung I

1. ILP

Das 1. ILP zum loesen der Erweiterung I wurde in Pyton implementiert. Der Hauptteil des Quellcodes besteht in zwei Funktionen:

```
# Function computing the
def solve(types, styles, left, right, A, K):
    # Sum of all elements in A
    sum_a = 0
    for i in range(types):
        for j in range(styles):
            sum_a += A[i][j]

    # Minimum of all sums of rows of A
    k = -1
    for i in range(types):
        sum = 0
        for j in range(styles):
            sum += A[i][j]
        if k == -1 or sum < k:
            k = sum

    # Set k to ceiled k / left
    # Now, k is the upper limit for the amount of boxes!
    k = math.ceil(float(k) / float(left))

    print("----------␣Start␣computations␣----------")

    msg = 0
    timeLimit = None # time limit in seconds
    start = time.time()

    PROBLEM, boxes, empty, columns = ilp(A, K, types, styles, k, left, right, msg, timeLimit)

    end = time.time()
    print("Time:", end - start)

    print("----------␣Done␣computating␣----------")
    # Print stats
    print("----------␣Stats␣----------")
    print("Boxes:", k)
    print("constraints:", len(PROBLEM.constraints))
    print("Variables:", len(PROBLEM.variables()))
    print("Status:", LpStatus[PROBLEM.status], "(", PROBLEM.status, ")")
    print("Objective:", value(PROBLEM.objective))
    print("Sum␣of␣A:", sum_a)
    print("----------␣Stats␣----------")
    # Print Boxes to a file
    file_name = "results.txt"
    print("----------␣Starting␣to␣write␣results␣to", file_name, "----------")
    used_boxed = 0

    with open(file_name, "w") as f:
        f.truncate()

        for index in range(k):
            if empty[index].value() == 1:
                used_boxed += 1
                for i in range(types):
                    for j in range(styles):
                        f.write(str(boxes[index][i][j].value()) + "␣")
                    f.write("\n")
                f.write("\n")
    f.close()
    print("Used␣boxes:", used_boxed)
```

```
    print("----------␣Done␣writing␣results␣----------")



def ilp(A, K, types, styles, k, left, right, message=0, time=600):
    # Maximum element of A
    m = max(list(map(lambda l: max(l), A)))

    boxes = []
    columns = []
    empty = []

    # 1) Create Variables
    # 1.1) Boxes
    for index in range(k):
        box = [ [] for _ in range(types)]
        for i in range(types):
            for j in range(styles):
                name = "Box-" + str(index) + "-" + str(i) + "-" + str(j)
                x = LpVariable(name, lowBound=0, upBound=right, cat="Integer")
                box[i].append(x)
        boxes.append(box)

    # 1.2) Decistion Variables for cols and empty (k + k*styles)
    for index in range(k):
        empt = LpVariable("Empty-" + str(index), lowBound=0, cat="Binary")
        empty.append(empt)

        col = []
        for j in range(styles):
            name = "Column-" + str(index) + "-" + str(j)
            x = LpVariable(name, lowBound=0, cat="Binary")
            col.append(x)
        columns.append(col)

    # 2) Define Problem and setup objective
    PROBLEM = LpProblem("Problem-1", LpMaximize)
    objective = lpSum(boxes)
    PROBLEM += objective

    # 3) Add constraints
    # 3.1) Set columns that cannot be used together
    # Make sure to only use upper diag of matrix K
    for index in range(k):
        for i in range(styles):
            for j in range(i+1, styles):
                if K[i][j] == False:
                    PROBLEM += columns[index][i] + columns[index][j] <= 1

    # 3.2) Set sum over all boxes of x_ij <= a_ij
    for i in range(types):
        for j in range(styles):
            sum = 0
            for index in range(k):
                sum += boxes[index][i][j]
            PROBLEM += sum <= A[i][j]

    # 3.3) For each col: IF decison variable for col is 0 => column will be empty;
  # If its 1 we dont care if the column is used
    D1 = types * m + 1 # Set D1 such that D1 >= sum of column
    for index in range(k):
        for j in range(styles):
            sum = 0
            for i in range(types):
                sum += boxes[index][i][j]
            PROBLEM += sum - D1 * columns[index][j] <= 0

    # 3.4) If any column is used we want the row-constrains to be ensured
    D2 = styles * m + left + 1 # Set D2 such that D2 >= LEFT + sum of row

    for index in range(k):
        sum = 0 # Sum over descion Variables of current box
        for j in range(styles):
            sum += columns[index][j]
        # Define empty[index] as the or of the decison variables of current box
        # E.i., the box is empty if empty[index] is 0
        PROBLEM += styles * empty[index] - sum >= 0
```

```
        PROBLEM += styles * empty[index] - sum - styles + 1 <= 0

        # If the box is empty (empty[index] is 0) make sure LOWER BOUND constraint is always satisfied
        for i in range(types):
            row = 0
            for j in range(styles):
                row += boxes[index][i][j]

            # 1) Lower Bound (A)
            PROBLEM += row + D2 * (1 - empty[index]) >= left

            # 2) Upper bound (B)
            PROBLEM += row <= right

    # 4) Solve
    solver = pulp.PULP_CBC_CMD(msg=message, timeLimit=time)
    PROBLEM.setSolver(solver)
    PROBLEM.solve()

    return (PROBLEM, boxes, empty, columns)
```

Cliquen Berechnung

Das naive Verfahren zum berechnen der Cliquen wurde in Python und C++ implementiert:

```
def get_cliques(K, styles):
    list = []
    queue = []

    for i in range(styles):
        queue.append({i})

    while len(queue) != 0:
        clique = queue.pop()
        max = True

        for i in range(styles):
            if i in clique:
                continue
            # Try to add i to clique. Check if it remains a clique
            is_clique = True
            for x in clique:
                if K[i][x] == False:
                    is_clique = False
                    break

            if is_clique:
                max = False
                new_clique = set()
                for x in clique:
                    new_clique.add(x)
                new_clique.add(i)
                queue.append(new_clique)
        if max:
            sorted(clique)
            if not clique in list:
                list.append(clique)

    return list
```

In C++ ist die naive Implementierung fuer die Berechnung aller maximaler Cliquen wie folgt:

```
// 1. simple all max cliques algorithm
std::set<std::set<int>> max_cliques(int n, matrix_bool &mat) {
    std::set<std::set<int>> all;
    std::queue<std::set<int>> q;

    for (int i = 0; i < n; i++) {
        q.push({i});
    }

    while (q.size()) {
        // Current clique
        std::set<int> top = q.front();
        q.pop();

        bool max = true;

        for (int i = 0; i < n; i++) {
            // Node i already in clique
            if (top.find(i) != top.end()) continue;

            // Check if adding v results in a clique;
            bool clique = true;

            for (int v : top) {
                if (!mat(v, i)) {
                    clique = false;
                    break;
                }
            }

            // If new, bigger clique: Add to queue
            if (clique) {
                max = false; // top cannot be a max. clique
                std::set<int> new_clique = top;
                new_clique.insert(i);
                q.push(new_clique);
            }
        }

        // Add to all if it is a max. clique
        if (max) {
            all.insert(top);
        }
    }

    return all;
}
```

2. ILP

Das der Hautpteil des verbesserten ILPs besteht ebenfalls aus zwei Funktionen:

```
  def solve(types, styles, left_bound, right_bound, A, K):
    cliques = get_cliques(K, styles)

    s = 0
    for i in range(types):
      for j in range(styles):
        s += A[i][j]

    print("----------␣Start:␣" + path + "----------")
    print("Cliques:", cliques)

    msg = 0
    timeLimit = None
    start = time.time()

    PROBLEM, boxes, empty, k = ilp(A, cliques, types, styles, left_bound, right_bound, msg, timeLimit)

    end = time.time()
    print("Time:", end - start)

    print("Boxes:", k)

    print("constraints:", len(PROBLEM.constraints))
    print("Variables:", len(PROBLEM.variables()))

    print("Status:", LpStatus[PROBLEM.status])
    print("Status:", PROBLEM.status)
    print("Objective", value(PROBLEM.objective))
    print("Sum:", s)

    print("----------␣Stats␣----------")

    # Print Boxes to a file
    file_name = "results.txt"
    print("----------␣Starting␣to␣write␣results␣to", file_name, "----------")
    used_boxed = 0

    with open(file_name, "w") as f:
      f.truncate()

      for index in range(k):
        print(empty[index].value())
        if empty[index].value() == 0:
          used_boxed += 1
          for i in range(types):
            for j in range(styles):
              if boxes[index][i][j] is None:
                f.write("0␣")
              else:
                f.write(str(boxes[index][i][j].value()) + "␣")
          f.write("\n")
        f.write("\n")
    f.close()

    print("Used␣boxes:", used_boxed)
    print("----------␣Done␣writing␣results␣----------")


def ilp(A, cliques, types, styles, left, right, message=0, time=None):
    # --- Init ILP Solver ---
    # 1) Create Variables
    #    For each clique create some amount k of boxes only using that clique
    #    Also, create decision variables indicating of the box is empty
    # boxes = [] # 4D array: [clique index][box index][box row][box col]
    boxes = [] # 3D array: [box index][box row][box col]
    # empty = [] # 2D array: [clique index][box index]
    clique_index = [] # 1D array: Map box index to clique index
    empty = [] # 1D array: [box index]
    cnt = 0 # Box index over all boxes

    for ii in range(len(cliques)):
```

```
        clique = cliques[ii]
        # Define the amount of boxes only using current clique to be k
        # Where k = minimum over all rows (clique cols) of row sum
        k = 100000
        for i in range(types):
            sum = 0
            for j in clique:
                sum += A[i][j]
            k = min(k, sum)
        k = k // left

        # For every box:
        for index in range(k):
            clique_index.append(ii)

            # 1.1) Create box of current type
            box = [ [None for _ in range(styles)] for _ in range(types)]
            for i in range(types):
                for j in clique:
                    name = "Box-" + str(cnt) + "-" + str(i) + "-" + str(j)
                    x = LpVariable(name, lowBound=0, upBound=3, cat="Integer")
                    box[i][j] = x
            boxes.append(box)

            # 1.2) For each box create a variable indicating that that box is all zero
            name =  "Empty-" + str(cnt)
            x = LpVariable(name, lowBound=0, cat="Binary")
            empty.append(x)
            cnt += 1 # All box counter

    # 2) Define Problem and objective
    PROBLEM = LpProblem("Problem-1", LpMaximize)
    objective = lpSum(boxes)
    PROBLEM += objective

    # 3) Add constraints
    # 3.1) Sum per (type, style) over all boxes
    for i in range(types):
        for j in range(styles):
            sum = 0
            for index in range(cnt):
                if boxes[index][i][j] is None:
                    continue
                sum += boxes[index][i][j]
            PROBLEM += sum <= A[i][j]

    # 3.2) Row constraints considering empty boxes
    D2 = left + 1
    for index in range(cnt):
        for i in range(types):
            sum = 0
            for j in range(styles):
                if boxes[index][i][j] is None:
                    continue
                sum += boxes[index][i][j]
            PROBLEM += sum <= right
            PROBLEM += sum + D2 * (empty[index]) >= left

    # 3.3) Empty boxes
    D3 = 10000 # Make sure D3 > Sum of all element of A
    for index in range(cnt):
        sum = 0
        for i in range(types):
            for j in range(styles):
                if boxes[index][i][j] is None:
                    continue
                sum += boxes[index][i][j]
        PROBLEM += sum - D3 * (1 - empty[index]) <= 0

    # 4) Solve
    solver = pulp.PULP_CBC_CMD(msg=message, timeLimit=time)
    PROBLEM.setSolver(solver)
    PROBLEM.solve()

    return (PROBLEM, boxes, empty, cnt)
```

Greedy-Verfahren

Der Hauptteil des Greedy-Verfahrens sieht in C++ wie folgt aus:

```
std::vector<matrix_int> solve(int types, int styles, matrix_int &clothes,
                std::vector<std::set<int>> &cliques) {
    std::vector<matrix_int> boxes;
    matrix_int X(types, styles, 0);
    matrix_int clothes_left = clothes;
    bool b = true;

    while (b) {
        X = next(types, styles, clothes_left, cliques);

        if (X.get_cols() == 0) {
            b = false;
        } else {
            boxes.push_back(X);
            clothes_left = clothes_left - X;
        }
    }

    return boxes;
}
```

Die erste Variante der next-Funktion sieht so aus:

```
  // Find the next box or return empty box if there is none
  matrix_int next(int types, int styles, matrix_int &clothes, std::vector<std::set<int>> &cliques) {
    for (std::set<int> Q : cliques) {
      matrix_int box(types, styles, 0);
      bool b = true;

      for (int i = 0; i < types; i++) {
        int s = 0;
        for (int q : Q) {
          int a = clothes(i, q);
          int val = std::min(a, right - s);
          box(i, q) = val;
          s += val;
        }
        if (s < left) {
          b = false;
          break;
        }
      }

      if (b) {
        return box;
      }
    }

    matrix_int nul(0, 0, 0);
    return nul;
  }
```

10.2. Erweiterung III

Fuer die Erweiterung III muessen nur wenige Aenderungen im ILP-Verfahren von Erweiterung I vorgenommen werden.

Erweiterung I

Der Python-Code der Implementierung des 2. ILP's wird wie folgt geaendert:
1. Teil: (ILP)
Hier wurde eine neue Zielfunktion definiert:

```
# 2) Define Problem and objective
# The object is now given by the amount of empty boxes:s
PROBLEM = LpProblem("Problem-1", LpMaximize)
objective = lpSum(empty)
PROBLEM += objective
```

Auszerdem wurde eine Bedingung hinzugefuegt, die sicherstellt, dass die bereits bekannte Anzahl an Kleidugnsstueck erreicht wird. Dabei ist best amount diese Anzahl an benutzten Kleidungsstuecken (nicht die Anzahl der uebrigen Kleidugnsstuecke!):

```
# 3.4) New constaint for making sure the amount of boxes wanted will be achieved
all_sum = 0
for index in range(cnt):
    for i in range(types):
        for j in range(styles):
            if boxes[index][i][j] is None:
                continue
            all_sum += boxes[index][i][j]
PROBLEM += all_sum >= best_amount
```

2. Teil: (ILP)
Hier wurde eine Bedinung hinzugefuegt, die die Maximalanzahl an Boxen gewaehrleistet:

```
# 3.4) New Constaint: Make sure the amount of used boxes is <= maxi
sum = 0
for index in range(cnt):
    sum += 1 - empty[index]
PROBLEM += sum <= maxi
```

11. Literatur

[1] Bron, Coen; Kerbosch, Joep (1973), «Algorithm 457: finding all cliques of an undirected graph», Communications of the ACM, Vol. 16, No. 9, p. 575–577, 10.1145/362342.362367

[2] Tsukiyama, S.; Ide, M.; Ariyoshi, I.; Shirakawa, I. (1977), «A new algorithm for generating all the maximal independent sets», SIAM Journal on Computing, Vol. 6, Iss. 3, doi: 10.1137/0206036

[3] Johnson, D. S.; Yannakakis, M. (1988), «On generating all maximal independent sets«, Information Processing Letters, Vol. 27, Iss. 3, p. 119–123, doi: 10.1016/0020-0190(88)90065-8

[4] Moon, J. W.; Moser, L. (1965), «On cliques in graphs», Israel Journal of Mathematics, Vol. 3, p. 23–28, doi: 10.1007/BF02760024

[5] Karp, Richard M. (1972). «Reducibility Among Combinatorial Problems», Complexity of Computer Computations, pp. 85–103, doi: 10.1007/978-1-4684-2001-2