Karlsruher Institut für Technologie

# Algorithmen II
# Vorlesung am 10.01.2013

## Approximierende Algorithmen: APAS für Bin Packing

INSTITUT FÜR THEORETISCHE INFORMATIK · PROF. DR. DOROTHEA WAGNER

Karlsruher Institut für Technologie

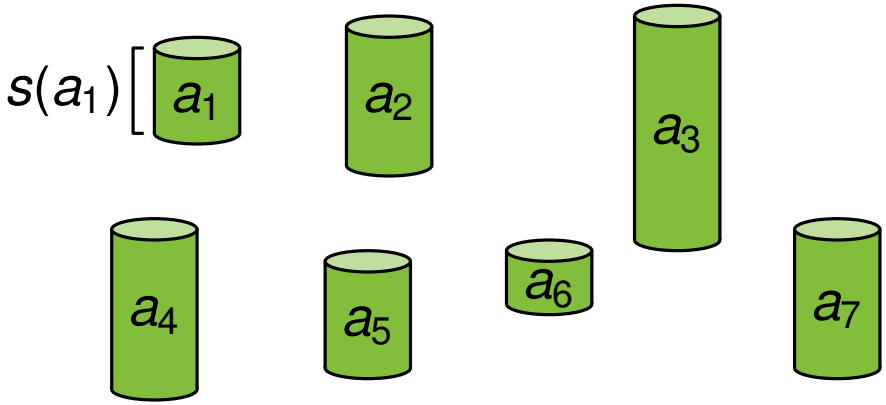
# APAS für Bin Packing

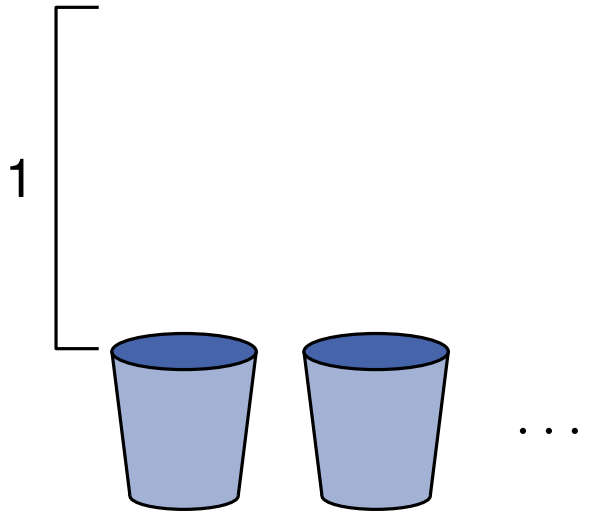


Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

# Wiederholung: Bin Packing – Definition

endliche Menge $M = \{a_1, \ldots, a_n\}$
mit Gewichtsfunktion $s\colon M \longrightarrow (0, 1]$




Eimer (Bins) mit Fassungsvermögen 1

Schreibe für $s(a_i)$ auch kurz $s_i$.

**Problem: BIN PACKING**
Weise die Elemente in $M$ einer minimalen Anzahl an Bins $B_1, \ldots, B_m$ zu, sodass für jeden Bin $B$ gilt:

$$\sum_{a_i \in B} s(a_i) \leq 1$$

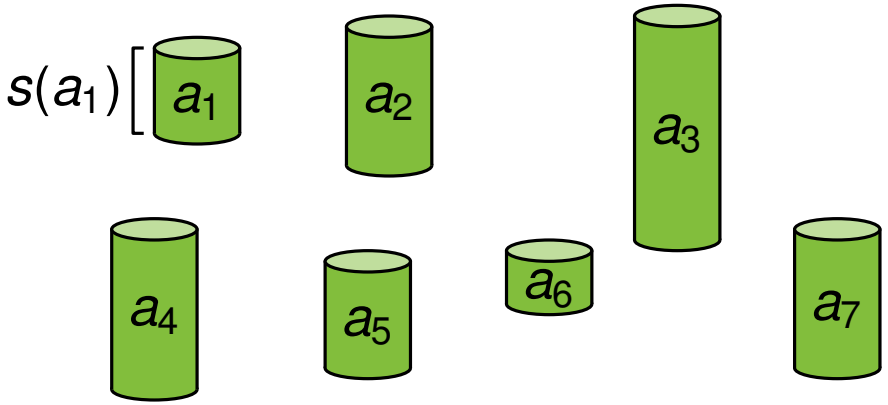
BIN PACKING ist $\mathcal{NP}$-schwer.



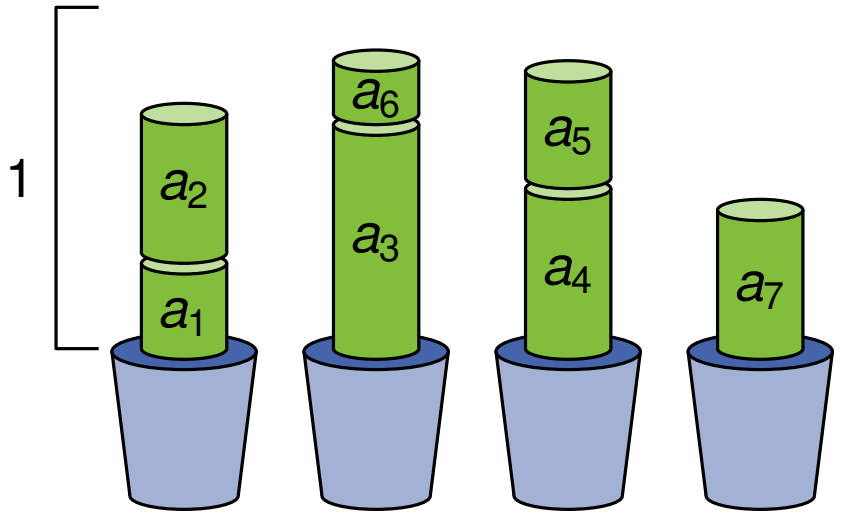
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

# Wiederholung: Bin Packing – Definition

endliche Menge $M = \{a_1, \ldots, a_n\}$
mit Gewichtsfunktion $s\colon M \longrightarrow (0, 1]$




Schreibe für $s(a_i)$ auch kurz $s_i$.




Eimer (Bins) mit Fassungsvermögen 1

4 Bins

**Problem: BIN PACKING**
Weise die Elemente in $M$ einer minimalen Anzahl an Bins $B_1, \ldots, B_m$ zu, sodass für jeden Bin $B$ gilt:

$$\sum_{a_i \in B} s(a_i) \leq 1$$

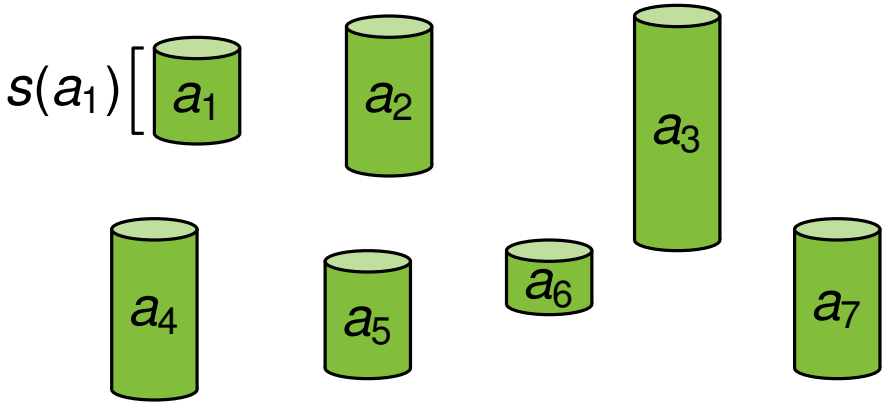
BIN PACKING ist $\mathcal{NP}$-schwer.



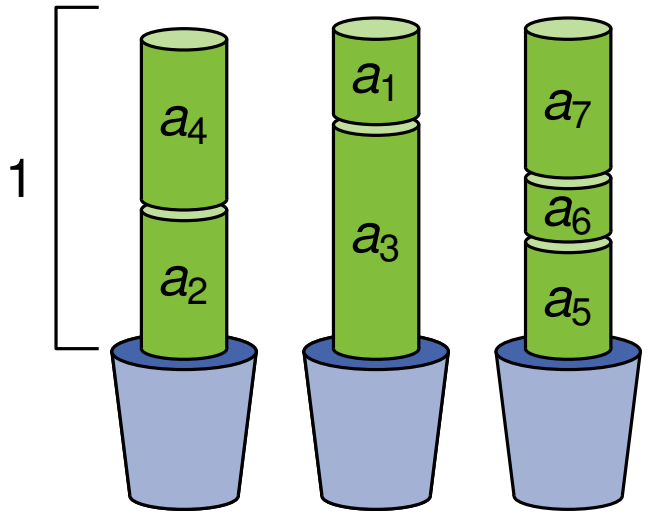
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

# Wiederholung: Bin Packing – Definition

endliche Menge $M = \{a_1, \ldots, a_n\}$
mit Gewichtsfunktion $s\colon M \longrightarrow (0, 1]$




Schreibe für $s(a_i)$ auch kurz $s_i$.




Eimer (Bins) mit Fassungsvermögen 1

3 Bins

**Problem: BIN PACKING**
Weise die Elemente in $M$ einer minimalen Anzahl an Bins $B_1, \ldots, B_m$ zu, sodass für jeden Bin $B$ gilt:

$$\sum_{a_i \in B} s(a_i) \leq 1$$

BIN PACKING ist $\mathcal{NP}$-schwer.



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

# Asymptotisches PAS

**Definition: Asymptotische Approximation**

Sei $\mathcal{A}$ ein Algorithmus. Definiere die *asymptotische Gütegarantie* $\mathcal{R}_{\mathcal{A}}^{\infty}$ wie folgt:

$$\mathcal{R}_{\mathcal{A}}^{\infty} := \inf \{ r \geq 1 \mid \text{ Es gibt } N > 0, \text{ sodass } \mathcal{R}_{\mathcal{A}}(I) \leq r \text{ für alle } I \text{ mit } \textsc{Opt}(I) \geq N \}$$

Falls $\mathcal{R}_{\mathcal{A}}^{\infty} \leq 1 + \varepsilon$, so ist $\mathcal{A}$ ein *asymptotisch $\varepsilon$-approximativer* Algorithmus.

**Beispiel:**

Sei $\mathcal{A}$ ein Algorithmus mit $\mathcal{A}(I) \leq \frac{3}{2} \cdot \textsc{Opt}(I) + 1$ für alle Instanzen $I$.
$\Rightarrow \mathcal{A}$ hat die asymptotische Gütegarantie $\mathcal{R}_{\mathcal{A}}^{\infty} = \frac{3}{2}$.

# Asymptotisches PAS

**Definition: Asymptotische Approximation**

Sei $\mathcal{A}$ ein Algorithmus. Definiere die *asymptotische Gütegarantie* $\mathcal{R}_{\mathcal{A}}^{\infty}$ wie folgt:

$$\mathcal{R}_{\mathcal{A}}^{\infty} := \inf \{ r \geq 1 \mid \text{ Es gibt } N > 0, \text{ sodass } \mathcal{R}_{\mathcal{A}}(I) \leq r \text{ für alle } I \text{ mit } \textsc{Opt}(I) \geq N \}$$

Falls $\mathcal{R}_{\mathcal{A}}^{\infty} \leq 1 + \varepsilon$, so ist $\mathcal{A}$ ein *asymptotisch $\varepsilon$-approximativer* Algorithmus.

**Beispiel:**

Sei $\mathcal{A}$ ein Algorithmus mit $\mathcal{A}(I) \leq \frac{3}{2} \cdot \textsc{Opt}(I) + 1$ für alle Instanzen $I$.
$\Rightarrow \mathcal{A}$ hat die asymptotische Gütegarantie $\mathcal{R}_{\mathcal{A}}^{\infty} = \frac{3}{2}$.

**Definition: Asymptotisches PAS (APAS)** **(Definition 7.16)**

Ein *asymptotisches PAS (APAS)* ist eine Familie von Algorithmen $\{\mathcal{A}_\varepsilon \mid \varepsilon > 0\}$, sodass $\mathcal{A}_\varepsilon$ ein asymptotisch $\varepsilon$-approximativer Algorithmus ist.
$\{\mathcal{A}_\varepsilon \mid \varepsilon > 0\}$ ist ein *asymptotisches vollpolynomielles PAS* (AFPAS), wenn die Laufzeit von $\mathcal{A}_\varepsilon$ polynomiell in $\frac{1}{\varepsilon}$ ist.

# APAS für Bin Packing – Überblick

**Unser Ziel:**
Finde APAS $\{\mathcal{A}_\varepsilon \mid \varepsilon > 0\}$, sodass für alle Instanzen $I$ von BIN PACKING gilt:

$$\mathcal{A}_\varepsilon(I) \leq (1 + \varepsilon) \cdot \text{OPT}(I) + 1$$

Damit kommt man beliebig nah an die optimale Lösung heran, bis auf einen zusätzlichen Bin (daher ein APAS und kein PAS).

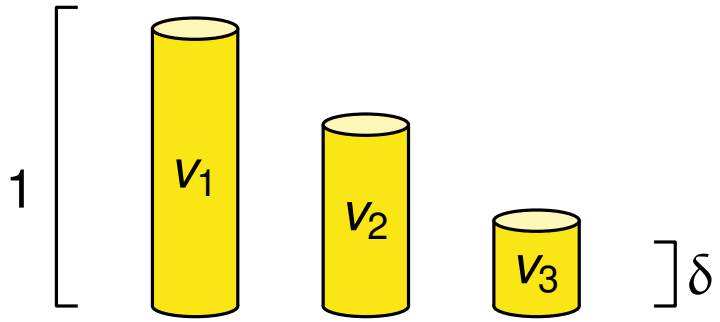
**Strategie:** Benutze folgende Techniken

- Restriktion von BIN PACKING
- Entfernen kleiner Elemente
- Lineares Gruppieren

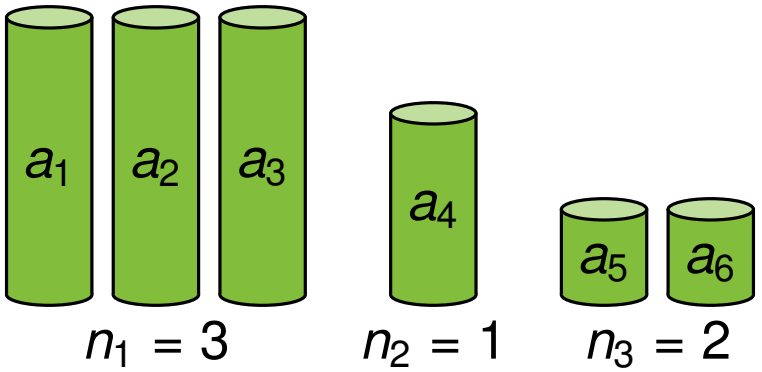
# Restricted Bin Packing (RBP)

**Zwei Einschränkungen:**

- Die Elementgrößen $s_1, \ldots, s_n$ sind auf $m$ erlaubte Größen $V = \{v_1, \ldots, v_m\}$ eingeschränkt ($m < n$).
  - $v_1 \geq \cdots \geq v_m$
- Für jede erlaubte Größe $v_i$ gilt $v_i \geq \delta$.




Instanz von RBP[$\delta$, $m$] ist gegeben durch Anzahl $n_i$ jeder Größe $v_i$.



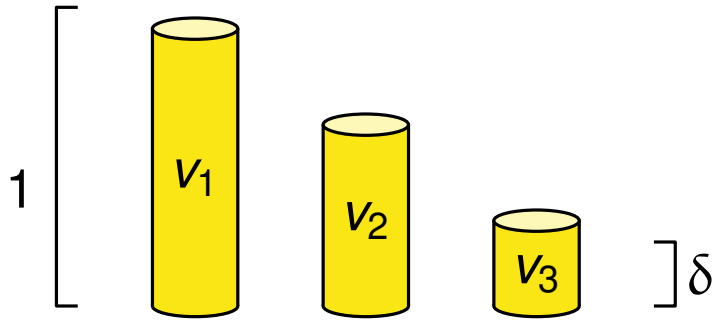



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

# Restricted Bin Packing (RBP)

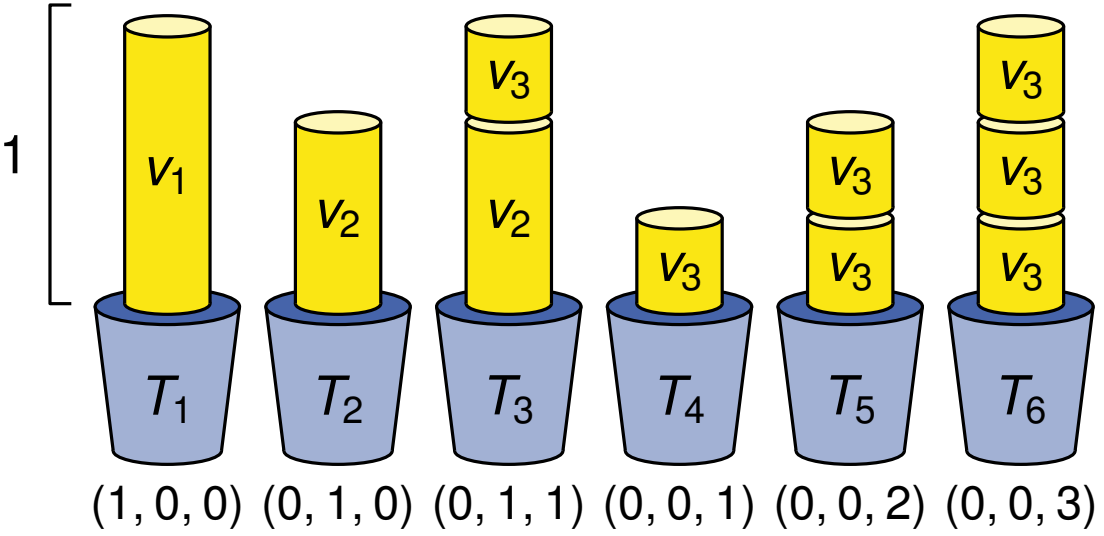
## Zwei Einschränkungen:

- Die Elementgrößen $s_1, \ldots, s_n$ sind auf $m$ erlaubte Größen $V = \{v_1, \ldots, v_m\}$ eingeschränkt ($m < n$).
  - $v_1 \geq \cdots \geq v_m$
- Für jede erlaubte Größe $v_i$ gilt $v_i \geq \delta$.




## Bin-Typen

- Eine Zuweisung von Elementen auf ein Bin ist durch ein $m$-Tupel $T_t = (T_{t_1}, \ldots, T_{t_m})$ gegeben.
- Ein Tupel $T_t$ heißt *Bin-Typ*, falls $\sum_{j=1}^{m} T_{t_j} \cdot v_j \leq 1$

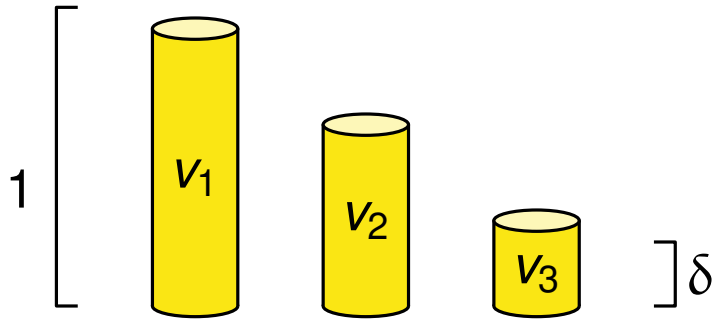



**Wie viele Bin-Typen kann es geben?**

# Restricted Bin Packing (RBP)

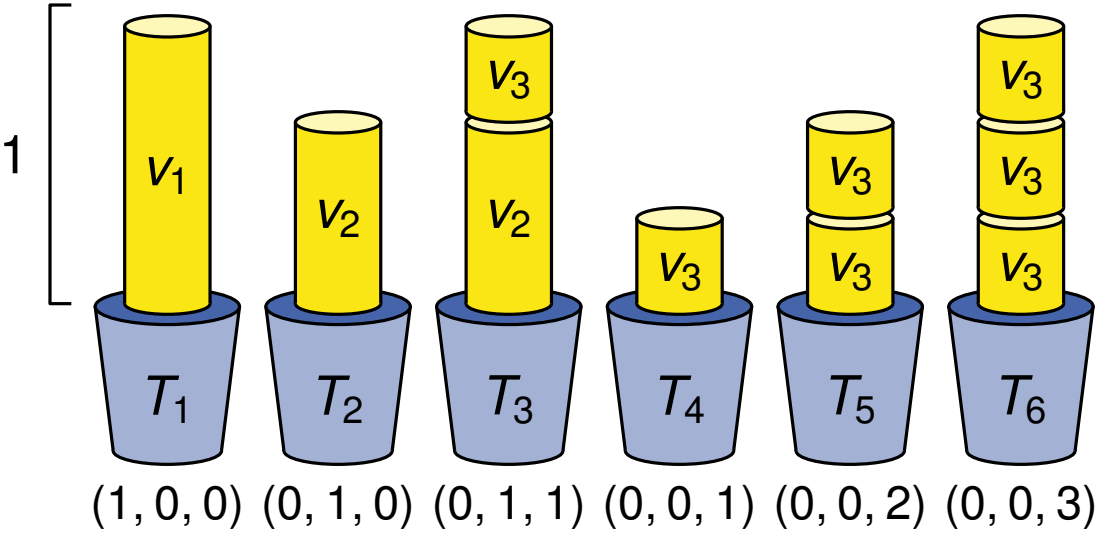
## Zwei Einschränkungen:

- Die Elementgrößen $s_1, \ldots, s_n$ sind auf $m$ erlaubte Größen $V = \{v_1, \ldots, v_m\}$ eingeschränkt ($m < n$).
  - $v_1 \geq \cdots \geq v_m$
- Für jede erlaubte Größe $v_i$ gilt $v_i \geq \delta$.




## Bin-Typen

- Eine Zuweisung von Elementen auf ein Bin ist durch ein $m$-Tupel $T_t = (T_{t_1}, \ldots, T_{t_m})$ gegeben.
- Ein Tupel $T_t$ heißt *Bin-Typ*, falls
$$\sum_{j=1}^{m} T_{t_j} \cdot v_j \leq 1$$




**Lemma: Anzahl Bin-Typen** **(Lemma 7.18)**

Sei $k = \left\lfloor \frac{1}{\delta} \right\rfloor$. Dann gilt für die Anzahl $q(\delta, m)$ an Bin-Typen: $q(\delta, m) \leq \binom{m+k}{k}$

# Formulierung als ILP

## Darstellung einer Lösung

- Eine Lösung einer Instanz von RBP[$\delta, m$] kann dadurch charakterisiert werden, wie viele Bins von jedem der $q = q(\delta, m)$ Bin-Typen vorkommen.
- $\Rightarrow$ Lösung ist durch ein $q$-Tupel $X = (x_1, \ldots, x_q)$ gegeben, wobei $x_t$ die Anzahl der Bins vom Typ $T_t$ ist.

# Formulierung als ILP

## Darstellung einer Lösung

- Eine Lösung einer Instanz von RBP[$\delta, m$] kann dadurch charakterisiert werden, wie viele Bins von jedem der $q = q(\delta, m)$ Bin-Typen vorkommen.
- $\Rightarrow$ Lösung ist durch ein $q$-Tupel $X = (x_1, \ldots, x_q)$ gegeben, wobei $x_t$ die Anzahl der Bins vom Typ $T_t$ ist.

## Formuliere RBP$[\delta, m]$-Instanz $I$ als ganzzahliges lineare Programm (ILP)

- Benutze die $x_i$ im Tupel $X$ als Variablen.
- Für jede erlaubte Größe $v_j$ (für $j \in \{1, \ldots, m\}$) stelle folgende Bedingung auf:

$$\sum_{t=1}^{q} x_t \cdot T_{t_j} = n_j$$

$x_t$: Anzahl der Bins von Typ $T_t$

$T_{t_j}$: Anzahl Elemente der Größe $v_j$ in Bin-Typ $T_t$

Linke Seite: Anzahl Elemente der Größe $v_j$ für $X = (x_1, \ldots, x_q)$

$n_j$: Anzahl Elemente der Größe $v_j$ in $I$.

- Die Anzahl Bins in der Lösung $X$ ist gerade $\sum_{i=1}^{q} x_i$ (das soll minimiert werden)

# Formulierung als ILP

**Wie groß ist das ILP?**

- Anzahl Variablen: $q(\delta, m)$.
- Anzahl Nebenbedingungen: $m$.

$\rightarrow$ $q(\delta, m)$ ist exponentiell in $m$ und $\frac{1}{\delta}$ (Erinnerung: $q(\delta, m) \leq \binom{m+k}{k}$, mit $k = \lfloor \frac{1}{\delta} \rfloor$)

Die Größe des ILP ist unabhängig von $n$ und damit konstant, falls $m$ und $\delta$ konstant.

**Satz: Algorithmus für RBP** **(Lemma 7.21)**

Eine Instanz $I$ von RBP$[\delta, m]$ kann in $O(n + f(\delta, m))$ gelöst werden, wobei $f(\delta, m)$ eine Konstante ist, die nur von $\delta$ und $m$ abhängt.

# Formulierung als ILP

**Wie groß ist das ILP?**

- Anzahl Variablen: $q(\delta, m)$.
- Anzahl Nebenbedingungen: $m$.

$\rightarrow$ $q(\delta, m)$ ist exponentiell in $m$ und $\frac{1}{\delta}$ (Erinnerung: $q(\delta, m) \leq \binom{m+k}{k}$, mit $k = \lfloor \frac{1}{\delta} \rfloor$)

Die Größe des ILP ist unabhängig von $n$ und damit konstant, falls $m$ und $\delta$ konstant.

**Satz: Algorithmus für RBP** **(Lemma 7.21)**

Eine Instanz $I$ von RBP$[\delta, m]$ kann in $O(n + f(\delta, m))$ gelöst werden, wobei $f(\delta, m)$ eine Konstante ist, die nur von $\delta$ und $m$ abhängt.

**Strategie:** Benutze folgende Techniken

- Restriktion von BIN PACKING $\leftarrow$ gerade gesehen
- Entfernen kleiner Elemente $\leftarrow$ damit $\delta$ nicht zu klein ist
- Lineares Gruppieren $\leftarrow$ damit $m$ nicht zu groß ist

# Entfernen kleiner Elemente

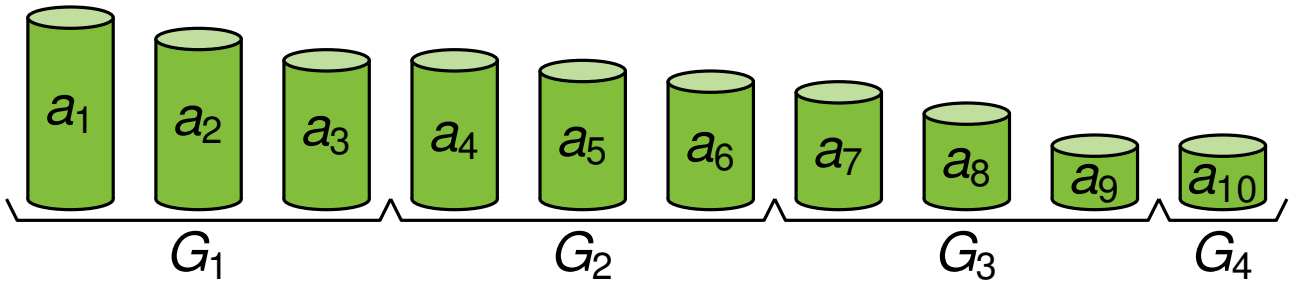
**Lemma: Kleine Elemente** **(Lemma 7.22)**

Sei $I$ eine Instanz von BIN PACKING mit einer Teillösung, die alle Elemente der Größe $s_i > \delta$ für $0 < \delta \leq \frac{1}{2}$ in $\beta$ Bins packt. Die Teillösung kann zu einer Lösung von $I$ erweitert werden, die maximal $\max\{\beta, (1 + 2 \cdot \delta) \cdot \text{OPT}(I) + 1\}$ Bins benötigt.

# Lineares Gruppieren

Instanz $I$ von BIN PACKING mit Elementen der Größe $1 \geq s_1 \geq \cdots \geq s_n \geq \delta > 0$
Für ein $k < n$ sei $m = \left\lfloor \frac{n}{k} \right\rfloor$. Definiere $m+1$ Gruppen $G_1, \ldots, G_{m+1}$, sodass $G_1$ die $k$ größten Elemente, $G_2$ die $k$ Nächstgrößten, usw. enthält.

$n = 10$
$k = 3$
$m = 3$



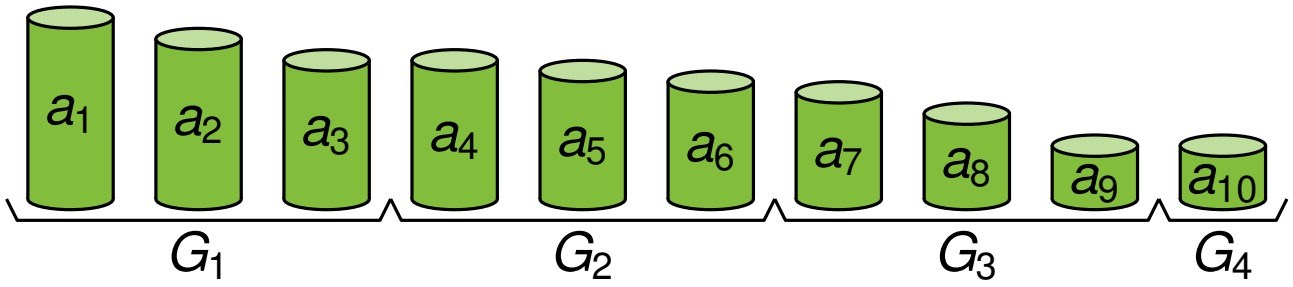



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

# Lineares Gruppieren

Instanz $I$ von BIN PACKING mit Elementen der Größe $1 \geq s_1 \geq \cdots \geq s_n \geq \delta > 0$
Für ein $k < n$ sei $m = \left\lfloor \frac{n}{k} \right\rfloor$. Definiere $m+1$ Gruppen $G_1, \ldots, G_{m+1}$, sodass $G_1$ die $k$ größten Elemente, $G_2$ die $k$ Nächstgrößten, usw. enthält.

$n = 10$
$k = 3$
$m = 3$




**Definition: Dominanz** **(Definition 7.23)**

Man definiert zu zwei Instanzen $I_1$ und $I_2$ von BIN PACKING mit

$I_1$ enthält Elemente der Größe $x_1 \geq x_2 \geq \cdots \geq x_n$

$I_2$ enthält Elemente der Größe $y_1 \geq y_2 \geq \cdots \geq y_n$

die Relation „$\geq$“ durch $I_1 \geq I_2 \Leftrightarrow x_i \geq y_i$ für $i \in \{1, \ldots, n\}$.
Man sagt dann $I_1$ *dominiert* $I_2$.

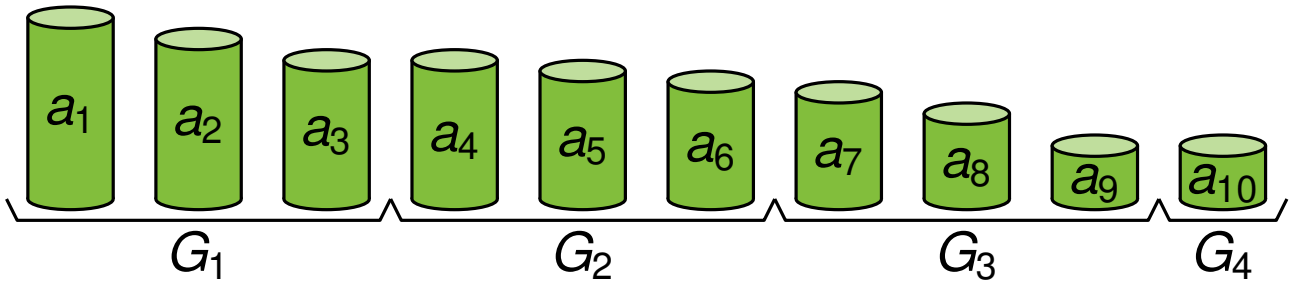


Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

# Lineares Gruppieren

Instanz $I$ von BIN PACKING mit Elementen der Größe $1 \geq s_1 \geq \cdots \geq s_n \geq \delta > 0$
Für ein $k < n$ sei $m = \left\lfloor \frac{n}{k} \right\rfloor$. Definiere $m+1$ Gruppen $G_1, \ldots, G_{m+1}$, sodass $G_1$ die $k$ größten Elemente, $G_2$ die $k$ Nächstgrößten, usw. enthält.

$n = 10$
$k = 3$
$m = 3$




**Definition: Dominanz** **(Definition 7.23)**

Man definiert zu zwei Instanzen $I_1$ und $I_2$ von BIN PACKING mit

$I_1$ enthält Elemente der Größe $x_1 \geq x_2 \geq \cdots \geq x_n$

$I_2$ enthält Elemente der Größe $y_1 \geq y_2 \geq \cdots \geq y_n$

die Relation „$\geq$“ durch $I_1 \geq I_2 \Leftrightarrow x_i \geq y_i$ für $i \in \{1, \ldots, n\}$.
Man sagt dann $I_1$ *dominiert* $I_2$.

**Folgerung:** **(Folgerung 7.24)**

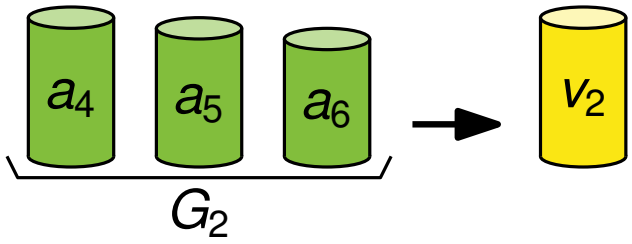
Falls $I_1 \geq I_2$, so gilt SIZE$(I_1) \geq$ SIZE$(I_2)$ und OPT$(I_1) \geq$ OPT$(I_2)$.
Offensichtlich: $G_1 \geq G_2 \geq \cdots \geq G_m$

(SIZE$(I)$ ist die Gesamtgröße der Elemente)



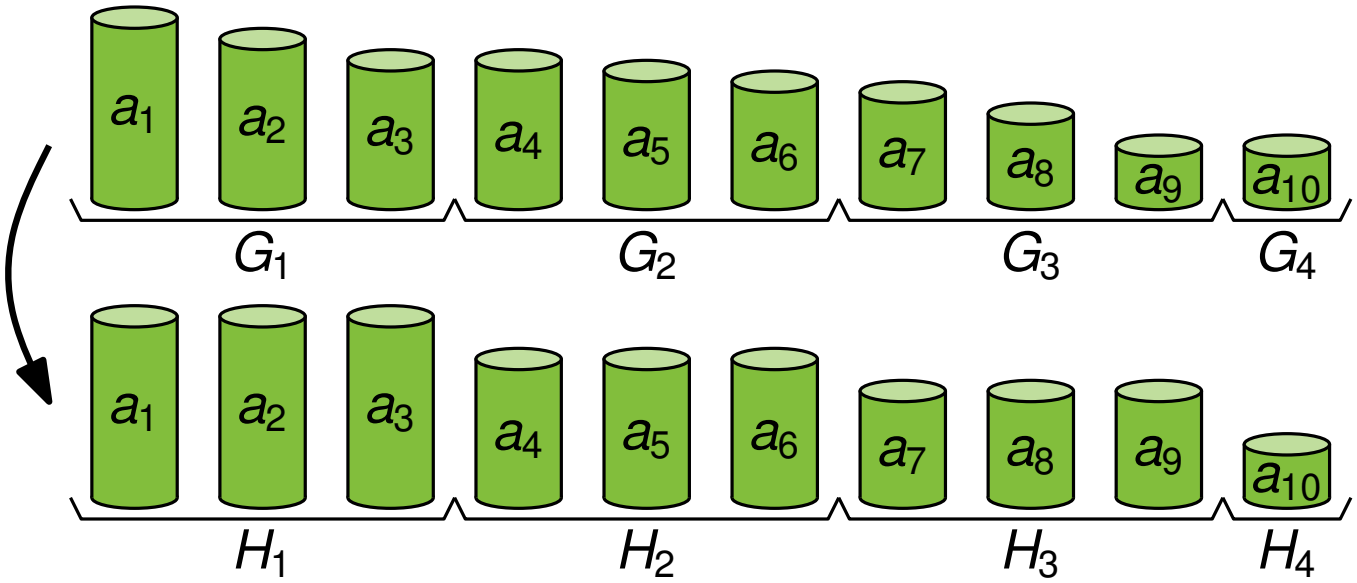
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

# Lineares Gruppieren



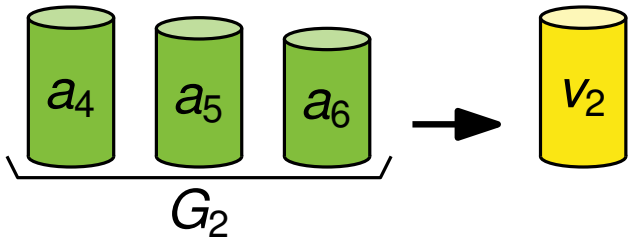

**Definition neuer Instanzen:**

- Sei $v_j$ das größte Element in $G_j$.
- Für $G_j$ definiere $H_j$ durch Erhöhung aller Elemente in $G_j$ auf $v_j$.



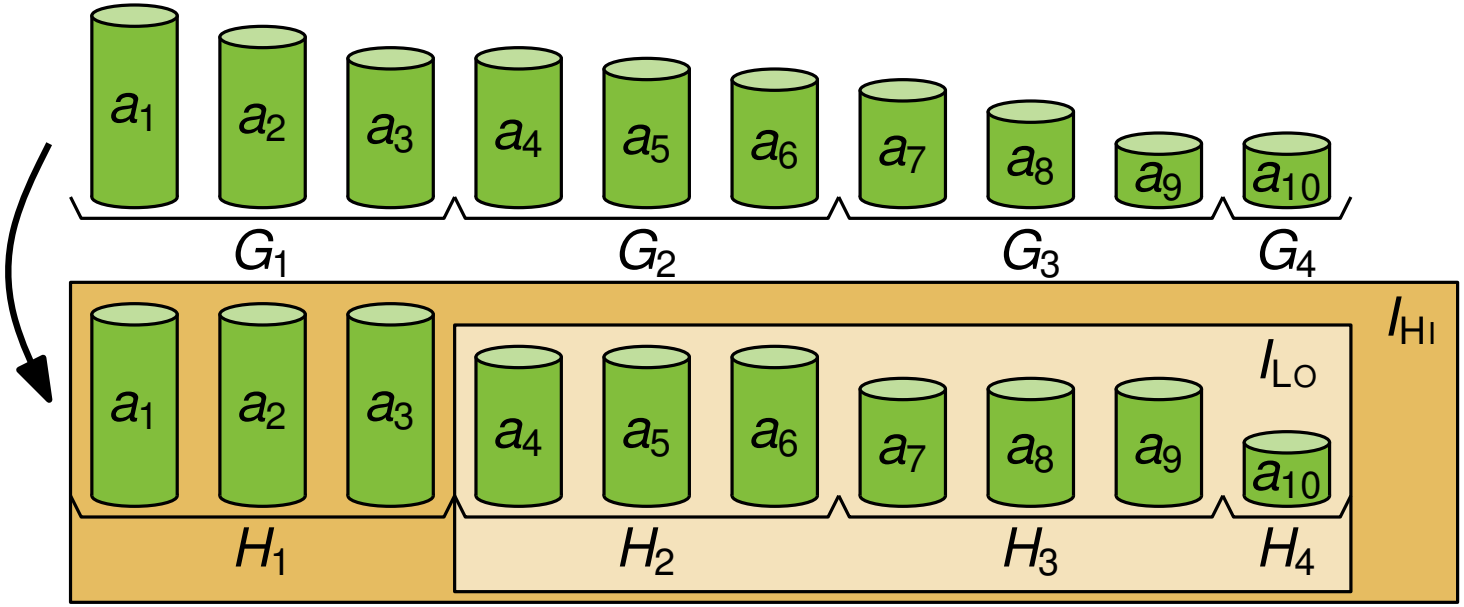

Es gilt: $H_1 \geq G_1 \geq H_2 \geq G_2 \cdots \geq H_{m+1} \geq G_{m+1}$

# Lineares Gruppieren




**Definition neuer Instanzen:**

- Sei $v_j$ das größte Element in $G_j$.
- Für $G_j$ definiere $H_j$ durch Erhöhung aller Elemente in $G_j$ auf $v_j$.




**Instanz $I_{Lo}$:**
$H_2 \cup \cdots \cup H_{m+1}$

**Instanz $I_{Hi}$:**
$H_1 \cup \cdots \cup H_{m+1}$

Es gilt: $H_1 \geq G_1 \geq H_2 \geq G_2 \cdots \geq H_{m+1} \geq G_{m+1}$

**Lemma: Einordnung der Originalinstanz $I$** **(Lemma 7.25)**

$$\text{OPT}(I_{Lo}) \leq \text{OPT}(I) \leq \text{OPT}(I_{Hi}) \leq \text{OPT}(I_{Lo}) + k$$

$$\text{SIZE}(I_{Lo}) \leq \text{SIZE}(I) \leq \text{SIZE}(I_{Hi}) \leq \text{SIZE}(I_{Lo}) + k$$



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

# Lineares Gruppieren

**Lemma: Einordnung der Originalinstanz $I$** **(Lemma 7.25)**

$$\textsc{Opt}(I_{\text{Lo}}) \leq \textsc{Opt}(I) \leq \textsc{Opt}(I_{\text{Hi}}) \leq \textsc{Opt}(I_{\text{Lo}}) + k$$

$$\textsc{Size}(I_{\text{Lo}}) \leq \textsc{Size}(I) \leq \textsc{Size}(I_{\text{Hi}}) \leq \textsc{Size}(I_{\text{Lo}}) + k$$

$\Rightarrow$ Man kann Lösung für $I$ mit maximal $\textsc{Opt}(I) + k$ Bins finden, indem man $I_{\text{Lo}}$ löst.
**Beachte:** $I_{\text{Lo}}$ ist Instanz von RBP$[\delta, m]$.

Außerdem kann $I_{\text{Lo}}$ in $O(n \log n)$ Zeit aus $I$ berechnet werden (Sortieren).

# APAS für Bin Packing

APAS $\mathcal{A}_\varepsilon(I = (s_1 \geq \cdots \geq s_n))$

$\delta \leftarrow \frac{\varepsilon}{2}$

$J \leftarrow$ Instanz von RBP$[\delta, n']$ bestehend aus allen Elementen in $I$ der Größe $\geq \delta$

$k \leftarrow \left\lceil \frac{\varepsilon^2}{2} \cdot n' \right\rceil$

$(H_1, \ldots, H_{m+1}) \leftarrow$ Gruppen der vergrößerten Elemente, wobei $m = \left\lfloor \frac{n'}{k} \right\rfloor$

$J_{\text{LO}} \leftarrow$ Instanz von RBP$[\delta, m]$ bestehend aus $H_2, \ldots, H_{m+1}$

$J_{\text{HI}} \leftarrow$ Instanz von Bin Packing bestehend aus $H_1, \ldots, H_{m+1}$

Berechne optimale Lösung von $J_{\text{LO}}$ (mittels ILP)

Füge die $k$ Elemente aus $H_1$ in maximal $k$ zusätzliche Bins ein $\rightarrow$ Lösung von $J_{\text{HI}}$

Berechne daraus Lösung für $J$ ohne zusätzliche Bins

Erweitere Lösung von $J$ mittels First Fit zu Lösung von $I$.

**Satz: APAS** **(Satz 7.26)**

Für den Algorithmus $A_\varepsilon$ gilt: $\mathcal{A}_\varepsilon(I) \leq (1 + \varepsilon) \cdot \text{OPT}(I) + 1$

$A_\varepsilon$ hat Laufzeit $O(n \cdot \log n + c_\varepsilon)$, wobei $c_\varepsilon$ eine „Konstante“ ist, die von $\varepsilon$ abhängt.

# APAS für Bin Packing

APAS $\mathcal{A}_\varepsilon(I = (s_1 \geq \cdots \geq s_n))$

$\delta \leftarrow \frac{\varepsilon}{2}$

$J \leftarrow$ Instanz von RBP$[\delta, n']$ bestehend aus allen Elementen in $I$ der Größe $\geq \delta$

$k \leftarrow \left\lceil \frac{\varepsilon^2}{2} \cdot n' \right\rceil$

$(H_1, \ldots, H_{m+1}) \leftarrow$ Gruppen der vergrößerten Elemente, wobei $m = \left\lfloor \frac{n'}{k} \right\rfloor$

$J_{\text{LO}} \leftarrow$ Instanz von RBP$[\delta, m]$ bestehend aus $H_2, \ldots, H_{m+1}$

$J_{\text{HI}} \leftarrow$ Instanz von Bin Packing bestehend aus $H_1, \ldots, H_{m+1}$

Berechne optimale Lösung von $J_{\text{LO}}$ (mittels ILP)

Füge die $k$ Elemente aus $H_1$ in maximal $k$ zusätzliche Bins ein $\rightarrow$ Lösung von $J_{\text{HI}}$

Berechne daraus Lösung für $J$ ohne zusätzliche Bins

Erweitere Lösung von $J$ mittels First Fit zu Lösung von $I$.

**Satz: APAS** **(Satz 7.26)**

Für den Algorithmus $A_\varepsilon$ gilt: $\mathcal{A}_\varepsilon(I) \leq (1 + \varepsilon) \cdot \text{OPT}(I) + 1$

$A_\varepsilon$ hat Laufzeit $O(n \cdot \log n + c_\varepsilon)$, wobei $c_\varepsilon$ eine „Konstante“ ist, die von $\varepsilon$ abhängt.

**Beachte:** $c_\varepsilon$ ist exponentiell in $\frac{1}{\varepsilon}$ (Lösen des ILP's)



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

# AFPAS für Bin Packing



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

# Relaxierung des ILP

**Problem:** Die Laufzeit des vorgestellten Algorithmus $\mathcal{A}_\varepsilon$ ist exponentiell in $\frac{1}{\varepsilon}$.

- Lösen von RBP[$\delta$, $m$] mittels ILP ist teuer.
- Ziel: löse RBP[$\delta$, $m$] schneller mittels *Relaxierung*.

**Formulierung als ILP (in Matrixschreibweise):**

- $X = (x_1 \ldots x_q)$ ($x_t$ ist Anzahl Bins von Typ $T_t$)
- $q \times m$-Matrix $A$ (Eintrag $a_{t,j}$ beschreibt Anzahl Elemente der Größe $v_j$ in $T_t$)
- $N^T = (n_1 \ldots n_m)$ ($n_j$ beschreibt Anzahl Elemente der Größe $v_j$ in der Eingabe)

$$\text{Minimiere } (1 \ldots 1) \cdot X^T = \sum_{t=1}^{q} x_t$$

Unter den Bedingungen

$$X \cdot A = N \text{ und } x_t \geq 0 \text{ (für } 1 \leq t \leq q)$$

# Relaxierung des ILP

**Problem:** Die Laufzeit des vorgestellten Algorithmus $\mathcal{A}_\varepsilon$ ist exponentiell in $\frac{1}{\varepsilon}$.

- Lösen von RBP[$\delta$, $m$] mittels ILP ist teuer.
- Ziel: löse RBP[$\delta$, $m$] schneller mittels *Relaxierung*.

**Formulierung als ILP (in Matrixschreibweise):**

- $X = (x_1 \ldots x_q)$ ($x_t$ ist Anzahl Bins von Typ $T_t$)
- $q \times m$-Matrix $A$ (Eintrag $a_{t,j}$ beschreibt Anzahl Elemente der Größe $v_j$ in $T_t$)
- $N^T = (n_1 \ldots n_m)$ ($n_j$ beschreibt Anzahl Elemente der Größe $v_j$ in der Eingabe)

$$\text{Minimiere } (1 \ldots 1) \cdot X^T = \sum_{t=1}^{q} x_t$$

Unter den Bedingungen

$$X \cdot A = N \text{ und } x_t \geq 0 \text{ (für } 1 \leq t \leq q)$$

**Relaxierung:** Weglassen der Forderung nach Ganzzahligkeit liefert ein LP.

$\rightarrow$ Sei LIN($I$) der Wert einer optimalen Lösung des LP. (nicht immer ganzzahlig)

**Beispiel:** $x_t = 3\frac{3}{4}$ bedeutet, dass von Bin-Typ $T_t$ 3 ganze und $\frac{3}{4}$ eines weiteren Bins verwendet werden. $\rightarrow$ kann man daraus eine „richtige“ Lösung machen?

# Relaxierung des ILP

**Aus der Theorie der linearen Programmierung:**

- Ohne Einschränkungen gilt $\text{rang}(A) = m$ und die ersten $m$ Zeilen von $A$ bilden eine Basis.
- Eine Lösung $X^\star$ mit $x_t = 0$ für $t > m$ heißt *Basislösung*.
- Es gibt eine Basislösung, die optimal ist (also Wert $\text{LIN}(I)$ hat).

**Lemma: Lösung LP $\rightarrow$ Lösung ILP** **(Lemma 7.28)**

Für alle Instanzen $I$ von RBP$[\delta, m]$ gilt:

$$\text{SIZE}(I) \leq \text{LIN}(I) \leq \text{OPT}(I) \leq \text{LIN}(I) + \frac{m+1}{2}$$

# Relaxierung des ILP

**Aus der Theorie der linearen Programmierung:**

- Ohne Einschränkungen gilt $\text{rang}(A) = m$ und die ersten $m$ Zeilen von $A$ bilden eine Basis.
- Eine Lösung $X^\star$ mit $x_t = 0$ für $t > m$ heißt *Basislösung*.
- Es gibt eine Basislösung, die optimal ist (also Wert $\text{LIN}(I)$ hat).

**Lemma: Lösung LP $\rightarrow$ Lösung ILP** **(Lemma 7.28)**

Für alle Instanzen $I$ von RBP$[\delta, m]$ gilt:

$$\text{SIZE}(I) \leq \text{LIN}(I) \leq \text{OPT}(I) \leq \text{LIN}(I) + \frac{m+1}{2}$$

**Satz (ohne Beweis): Vollpolynomieller Algorithmus** **(Satz 7.30)**

Es gibt einen vollpolynomiellen Algorithmus $\mathcal{A}$ zur Lösung einer Instanz von RBP$[\delta, m]$ mit

$$\mathcal{A}(I) \leq \text{LIN}(I) + \frac{m+1}{2} + 1$$

(Karmaker & Karp 1982)

# APAS für Bin Packing

APAS $\mathcal{A}_\varepsilon(I = (s_1 \geq \cdots \geq s_n))$

$\delta \leftarrow \frac{\varepsilon}{2}$

$J \leftarrow$ Instanz von RBP$[\delta, n']$ bestehend aus allen Elementen in $I$ der Größe $\geq \delta$

$k \leftarrow \left\lceil \frac{\varepsilon^2}{2} \cdot n' \right\rceil$

$(H_1, \ldots, H_{m+1}) \leftarrow$ Gruppen der vergrößerten Elemente, wobei $m = \left\lfloor \frac{n'}{k} \right\rfloor$

$J_{\text{LO}} \leftarrow$ Instanz von RBP$[\delta, m]$ bestehend aus $H_2, \ldots, H_{m+1}$

$J_{\text{HI}} \leftarrow$ Instanz von Bin Packing bestehend aus $H_1, \ldots, H_{m+1}$

Berechne optimale Lösung von $J_{\text{LO}}$ (mittels ILP)

Füge die $k$ Elemente aus $H_1$ in maximal $k$ zusätzliche Bins ein $\rightarrow$ Lösung von $J_{\text{HI}}$

Berechne daraus Lösung für $J$ ohne zusätzliche Bins

Erweitere Lösung von $J$ mittels First Fit zu Lösung von $I$.

**Satz: APAS** **(Satz 7.26)**

Für den Algorithmus $A_\varepsilon$ gilt:

$$\mathcal{A}_\varepsilon(I) \leq (1 + \varepsilon) \cdot \text{Opt}(I) + 1$$

# ~~APAS~~ AFPAS für Bin Packing

~~APAS~~ AFPAS $\mathcal{A}_\varepsilon(I = (s_1 \geq \cdots \geq s_n))$

$\delta \leftarrow \frac{\varepsilon}{2}$

$J \leftarrow$ Instanz von RBP$[\delta, n']$ bestehend aus allen Elementen in $I$ der Größe $\geq \delta$

$k \leftarrow \left\lceil \frac{\varepsilon^2}{2} \cdot n' \right\rceil$

$(H_1, \ldots, H_{m+1}) \leftarrow$ Gruppen der vergrößerten Elemente, wobei $m = \left\lfloor \frac{n'}{k} \right\rfloor$

$J_{\text{LO}} \leftarrow$ Instanz von RBP$[\delta, m]$ bestehend aus $H_2, \ldots, H_{m+1}$

$J_{\text{HI}} \leftarrow$ Instanz von Bin Packing bestehend aus $H_1, \ldots, H_{m+1}$

~~Berechne optimale Lösung von $J_{\text{LO}}$ (mittels ILP)~~

Berechne Lösung mit Wert $\leq \text{Lin}(I) + \frac{m+1}{2} + 1$ (mittels LP Relaxierung)

Füge die $k$ Elemente aus $H_1$ in maximal $k$ zusätzliche Bins ein $\rightarrow$ Lösung von $J_{\text{HI}}$

Berechne daraus Lösung für $J$ ohne zusätzliche Bins

Erweitere Lösung von $J$ mittels First Fit zu Lösung von $I$.

**Satz: ~~APAS~~ AFPAS** **(Satz 7.~~26~~ 32)**

Für den Algorithmus $A_\varepsilon$ gilt:

$$\mathcal{A}_\varepsilon(I) \leq (1+\varepsilon) \cdot \text{Opt}(I) \; \cancel{+1} \; + \frac{1}{\varepsilon^2} + 3$$



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner