2D-Guillotine-Zuschnitt und Wang-Algorithmus

Francesco Kriegel & Matthias Lange

05. Februar 2009
WS 2008/2009

TECHNISCHE
UNIVERSITÄT
DRESDEN

Inhaltsverzeichnis

Inhaltsverzeichnis

1.1. Packungs- und Zuschnittsprobleme

Anwendungsbereiche:

Anwendungsbereiche:

- Gegenstände in einen begrenzten Stauraum „möglichst gut“ einordnen

Anwendungsbereiche:

- Gegenstände in einen begrenzten Stauraum „möglichst gut“ einordnen
- Aus einem Stück Material „möglichst gut“ ein paar Nutzgegenstände herausschneiden

- Wir beschränken uns hier auf die Betrachtung zweidimensionaler Probleme

- Wir beschränken uns hier auf die Betrachtung zweidimensionaler Probleme
- Unsere „Grundfläche“ ist ein Rechteck und unsere Einzelteile sind horizontal oder vertikal an diesem Rechteck ausgerichtete kleinere Rechtecke

- Wir beschränken uns hier auf die Betrachtung zweidimensionaler Probleme
- Unsere „Grundfläche“ ist ein Rechteck und unsere Einzelteile sind horizontal oder vertikal an diesem Rechteck ausgerichtete kleinere Rechtecke
- Jedes Rechteck kann nur begrenzt oft aus dem großen Rechteck herausgeschnitten werden, und jedes herausgeschnittene Stück hat einen bestimmten „Wert“.

- Wir beschränken uns hier auf die Betrachtung zweidimensionaler Probleme
- Unsere „Grundfläche“ ist ein Rechteck und unsere Einzelteile sind horizontal oder vertikal an diesem Rechteck ausgerichtete kleinere Rechtecke
- Jedes Rechteck kann nur begrenzt oft aus dem großen Rechteck herausgeschnitten werden, und jedes herausgeschnittene Stück hat einen bestimmten „Wert“.
- Wir versuchen, Einzelteile mit einer möglichst großer „Wert-Gesamtsumme“ herauszuschneiden.

Packungsproblem:

Packungsproblem:

- Auf einem gegebenen Rechteck der Länge L und der Breite B sind kleinere Rechtecke von bestimmtem Typ anzuordnen.

Packungsproblem:

- Auf einem gegebenen Rechteck der Länge L und der Breite B sind kleinere Rechtecke von bestimmtem Typ anzuordnen.
- Ein Rechtecktyp R_i ($i \in 1 \ldots m$) ist ein Tupel $R_i = (l_i, b_i, w_i, m_i)$, wobei l_i als dessen Länge, b_i als dessen Breite und w_i als dessen Wert interpretiert wird und m_i angibt, wie oft der Rechtecktyp in unserer Anordnung vorkommen darf

Packungsproblem:

- Auf einem gegebenen Rechteck der Länge L und der Breite B sind kleinere Rechtecke von bestimmtem Typ anzuordnen.
- Ein Rechtecktyp R_i ($i \in 1 \ldots m$) ist ein Tupel $R_i = (l_i, b_i, w_i, m_i)$, wobei l_i als dessen Länge, b_i als dessen Breite und w_i als dessen Wert interpretiert wird und m_i angibt, wie oft der Rechtecktyp in unserer Anordnung vorkommen darf
- Gesucht ist eine sich nicht überschneidende Anordnung der Rechtecke mit maximalem Wert

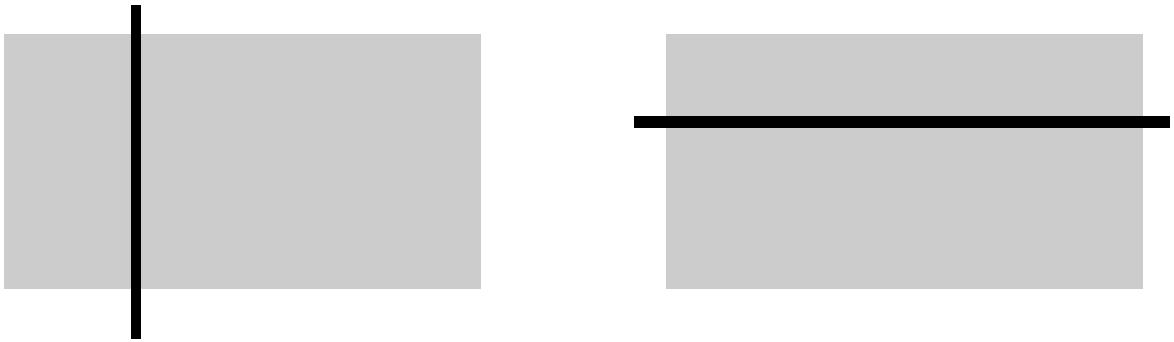
1.2. Guillotine-Anordnungen

- Es gibt einen Typ von Anordnungen, der in der Praxis oft leichter realisiert werden kann: Guillotine-Anordnungen

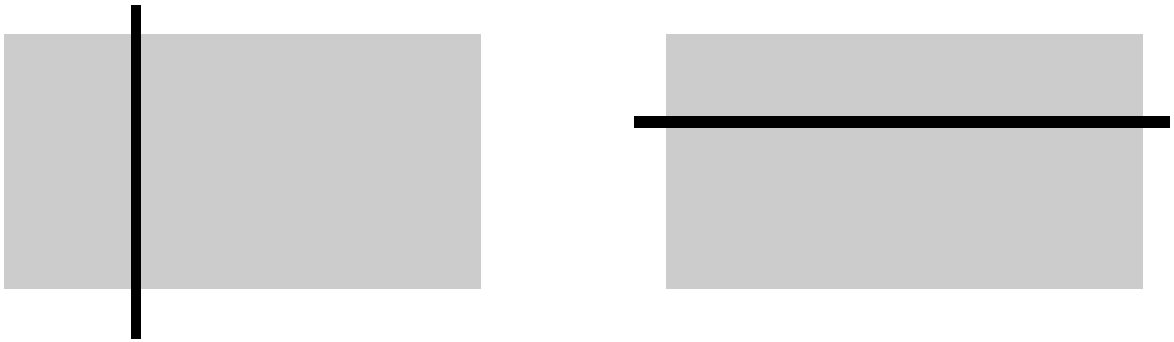
- Es gibt einen Typ von Anordnungen, der in der Praxis oft leichter realisiert werden kann: Guillotine-Anordnungen
- Ein Guillotine-Schnitt zerlegt ein gegebenes Rechteck in zwei Einzelrechtecke:

- Es gibt einen Typ von Anordnungen, der in der Praxis oft leichter realisiert werden kann: Guillotine-Anordnungen
- Ein Guillotine-Schnitt zerlegt ein gegebenes Rechteck in zwei Einzelrechtecke:

- Es gibt einen Typ von Anordnungen, der in der Praxis oft leichter realisiert werden kann: Guillotine-Anordnungen
- Ein Guillotine-Schnitt zerlegt ein gegebenes Rechteck in zwei Einzelrechtecke:



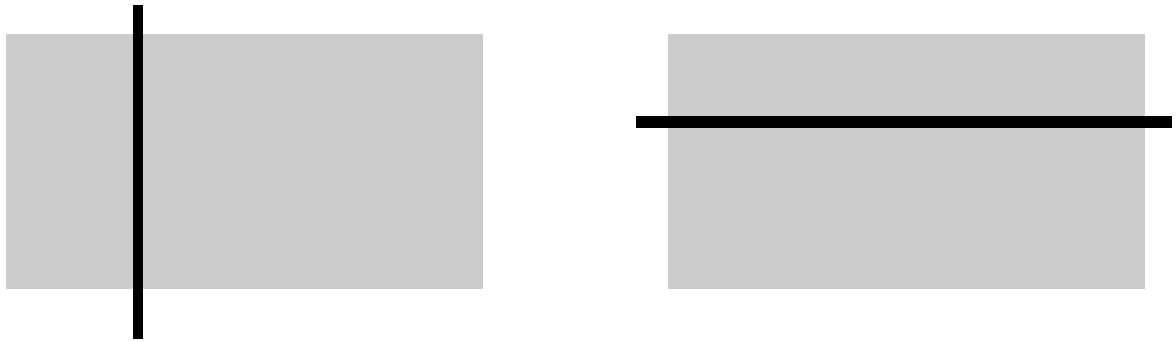
- Es gibt einen Typ von Anordnungen, der in der Praxis oft leichter realisiert werden kann: Guillotine-Anordnungen
- Ein Guillotine-Schnitt zerlegt ein gegebenes Rechteck in zwei Einzelrechtecke:



- Jedes der beiden Teilrechtecke kann nun wieder

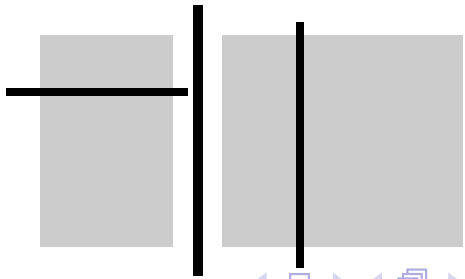
guillotine-geschnitten werden:

- Es gibt einen Typ von Anordnungen, der in der Praxis oft leichter realisiert werden kann: Guillotine-Anordnungen
- Ein Guillotine-Schnitt zerlegt ein gegebenes Rechteck in zwei Einzelrechtecke:



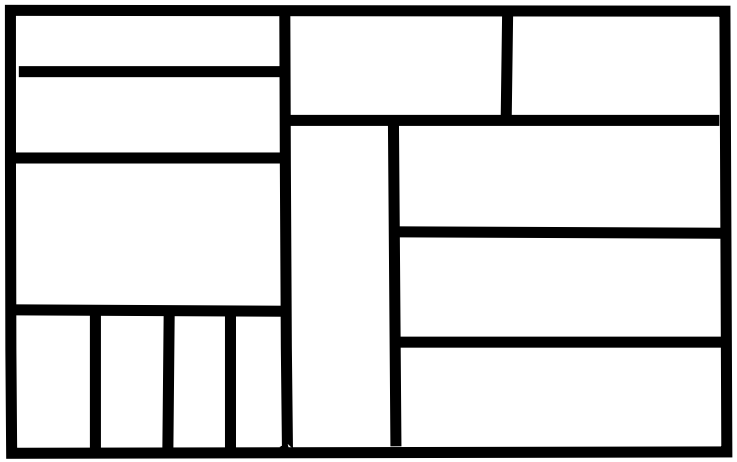
- Jedes der beiden Teilrechtecke kann nun wieder

guillotine-geschnitten werden:

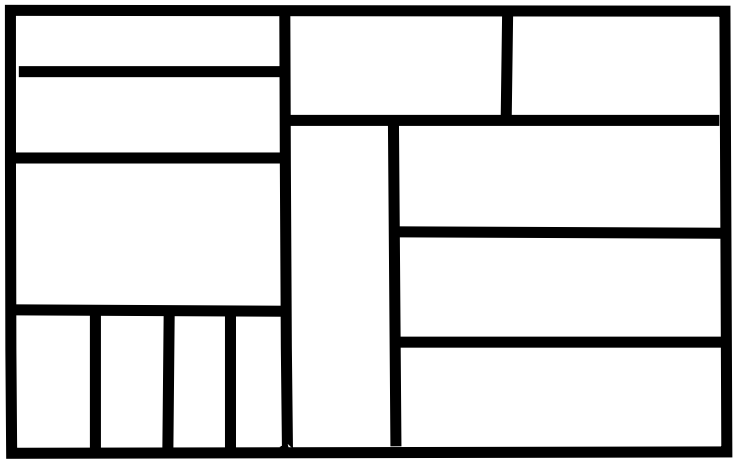


Eine Guillotine-Anordnung ist eine Anordnung von Rechtecken, die durch mehrere Guillotine-Schnitte aus dem Rechteck entstanden ist. Beispiele:

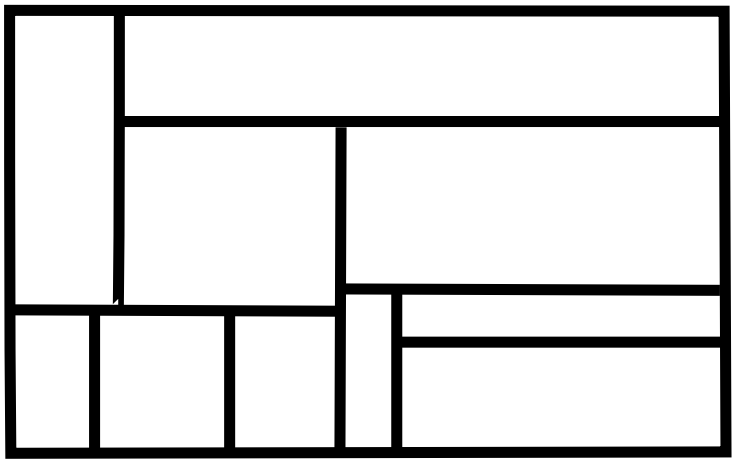
Eine Guillotine-Anordnung ist eine Anordnung von Rechtecken, die durch mehrere Guillotine-Schnitte aus dem Rechteck entstanden ist. Beispiele:



Eine Guillotine-Anordnung ist eine Anordnung von Rechtecken, die durch mehrere Guillotine-Schnitte aus dem Rechteck entstanden ist. Beispiele:

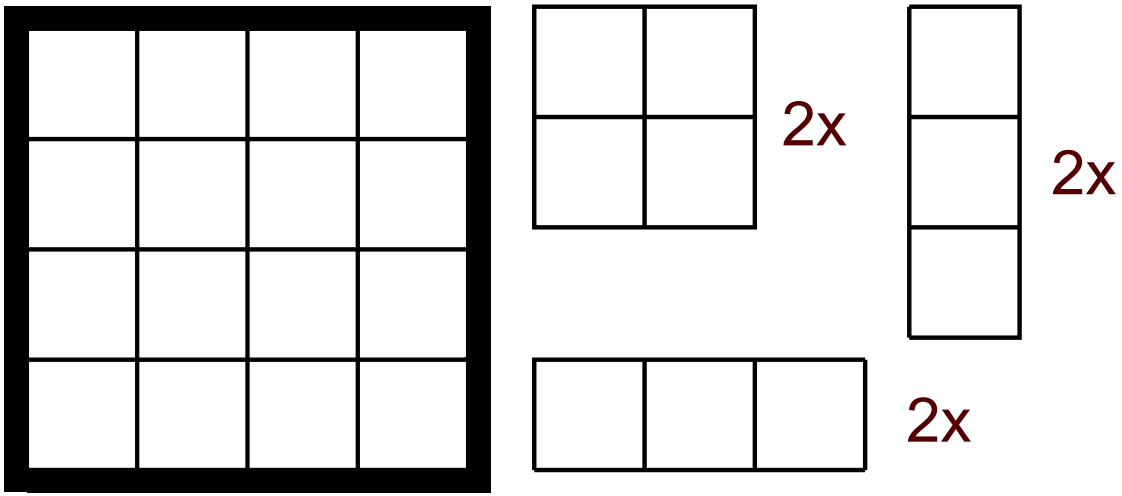


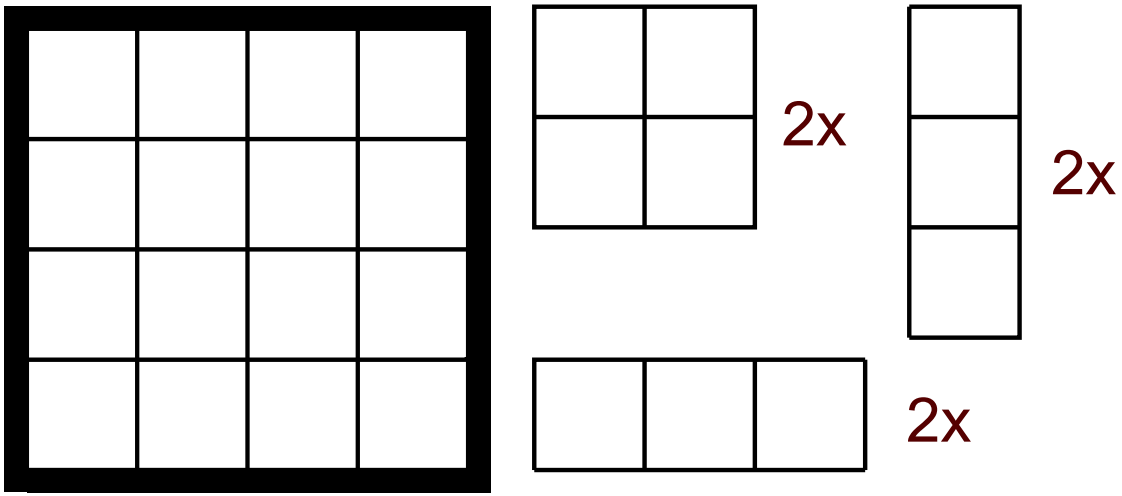
Guillotine-Anordnung



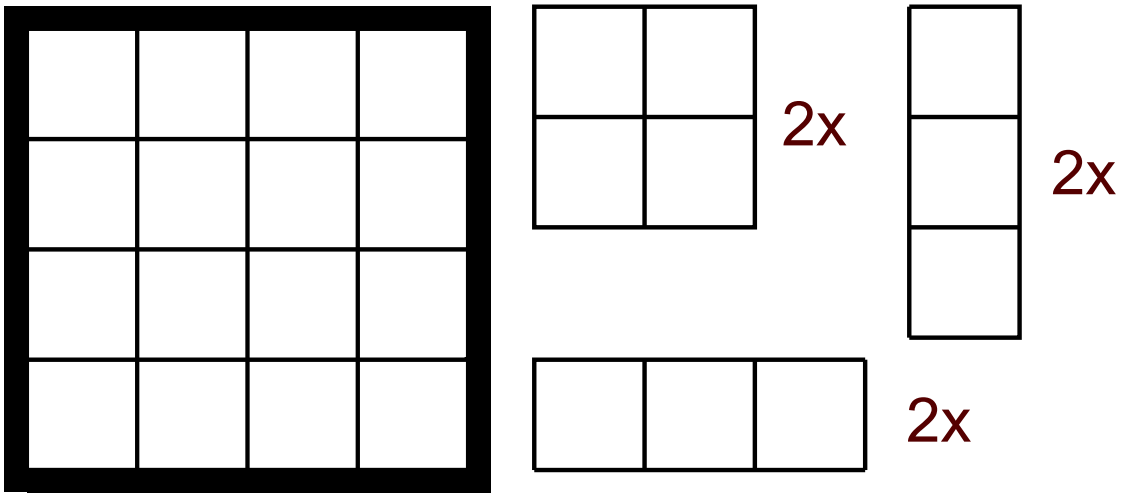
Nichtguillotine-Anordnung

2.1. Beispiel für den Wang-Algorithmus


2x
2x
2x

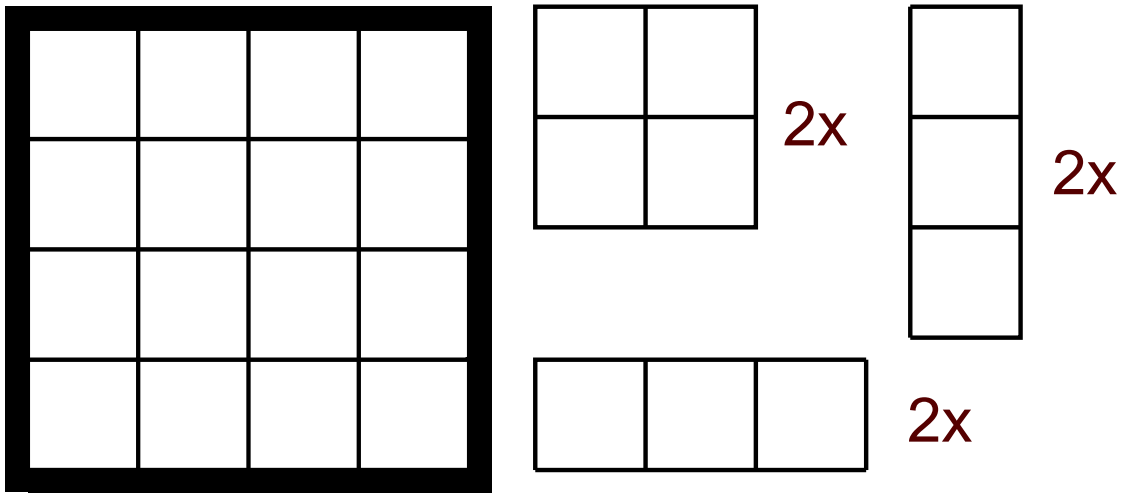


In ein Rechteck mit Maßen $L = B = 4$ werden



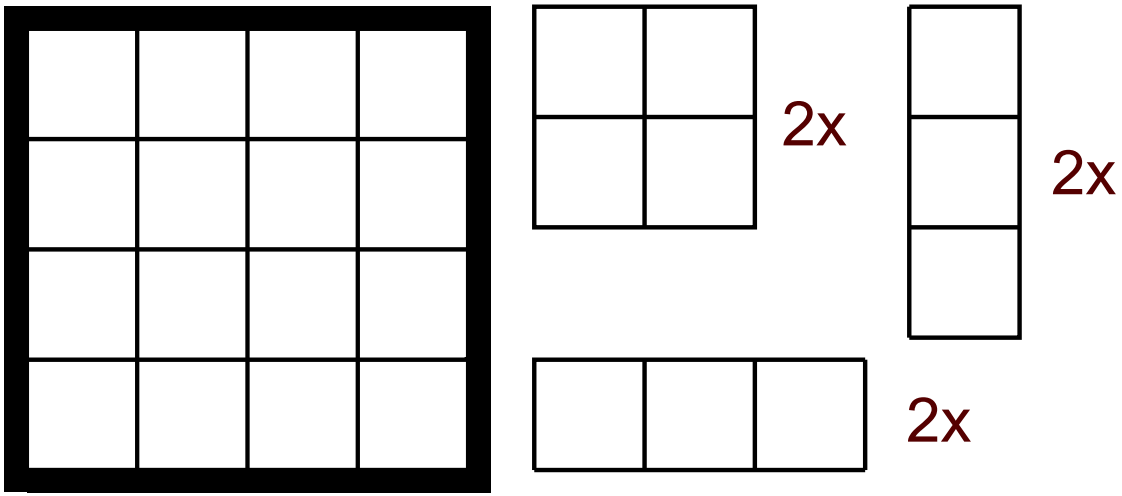
In ein Rechteck mit Maßen $L = B = 4$ werden

- $R_1 = (2, 2, 4, 2)$ (2×2, Wert: 4, Anzahl: 2)



In ein Rechteck mit Maßen $L = B = 4$ werden

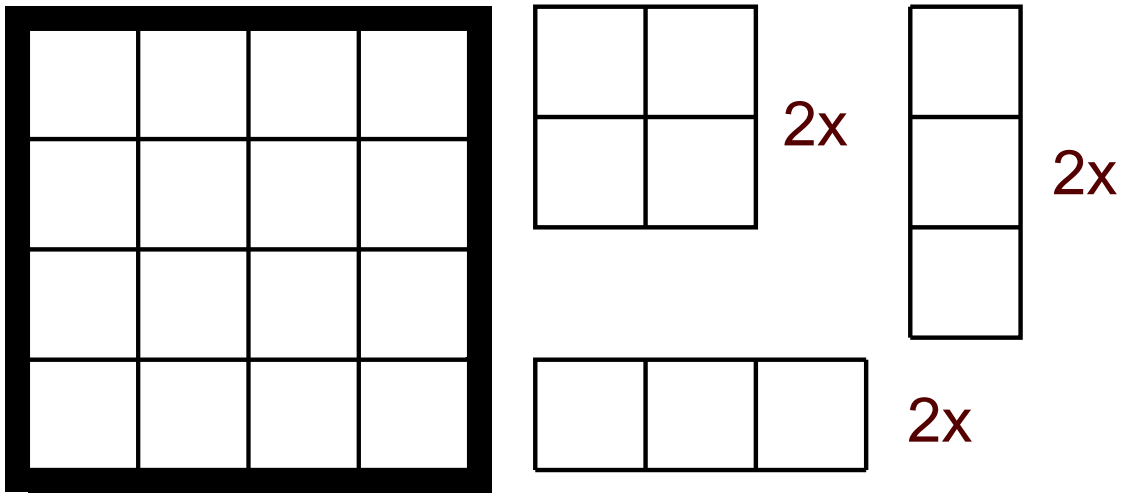
- $R_1 = (2, 2, 4, 2)$ (2×2, Wert: 4, Anzahl: 2)
- $R_2 = (1, 3, 3, 2)$ (1×3, Wert: 3, Anzahl: 2)



In ein Rechteck mit Maßen $L = B = 4$ werden

- $R_1 = (2, 2, 4, 2)$ (2×2, Wert: 4, Anzahl: 2)
- $R_2 = (1, 3, 3, 2)$ (1×3, Wert: 3, Anzahl: 2)
- $R_3 = (3, 1, 3, 2)$ (3×1, Wert: 3, Anzahl: 2)

eingefügt



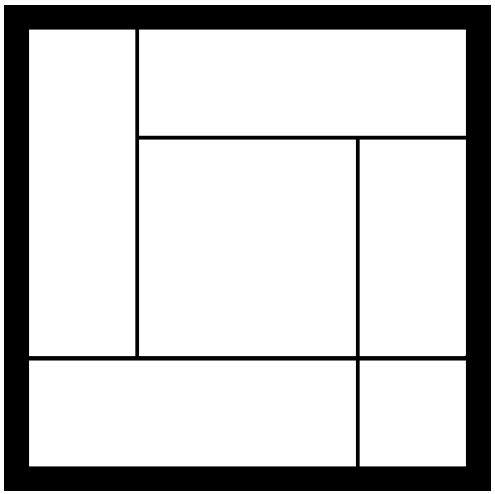
In ein Rechteck mit Maßen $L = B = 4$ werden

- $R_1 = (2, 2, 4, 2)$ (2×2, Wert: 4, Anzahl: 2)
- $R_2 = (1, 3, 3, 2)$ (1×3, Wert: 3, Anzahl: 2)
- $R_3 = (3, 1, 3, 2)$ (3×1, Wert: 3, Anzahl: 2)

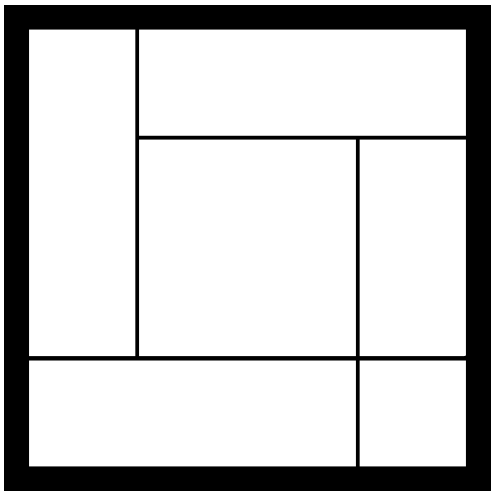
eingefügt
$w = l \cdot b \implies$ Ziel: Maximale Reduktion des Abfalls

Als klassisches Packungsproblem an, erreichen wir das Optimum:

Als klassisches Packungsproblem an, erreichen wir das Optimum:



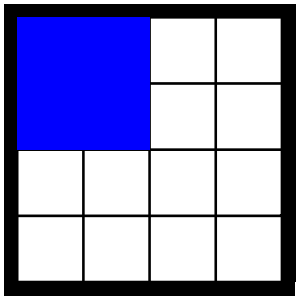
Als klassisches Packungsproblem an, erreichen wir das Optimum:



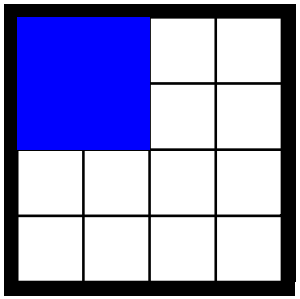
Hierbei handelt es sich aber nicht um eine Guillotine-Anordnung!

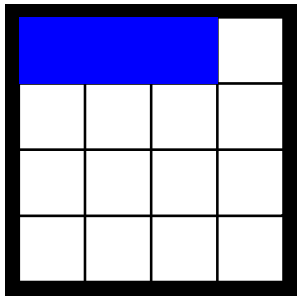
Suche nach einer guten Guillotine-Anordnung. Ansatz:

Suche nach einer guten Guillotine-Anordnung. Ansatz:

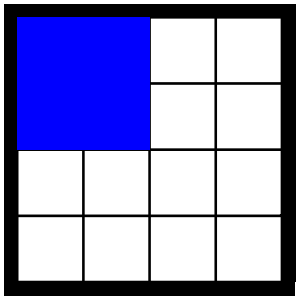
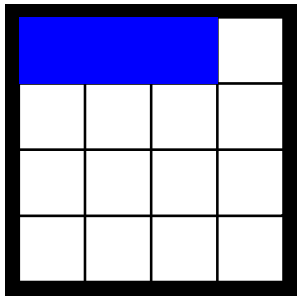
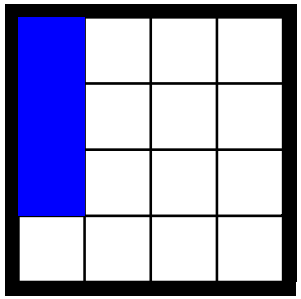


Suche nach einer guten Guillotine-Anordnung. Ansatz:



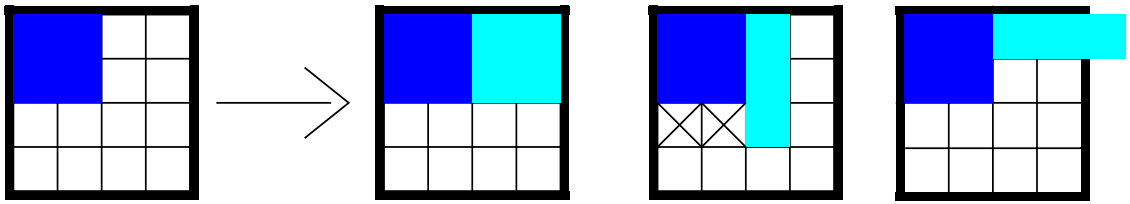


Suche nach einer guten Guillotine-Anordnung. Ansatz:

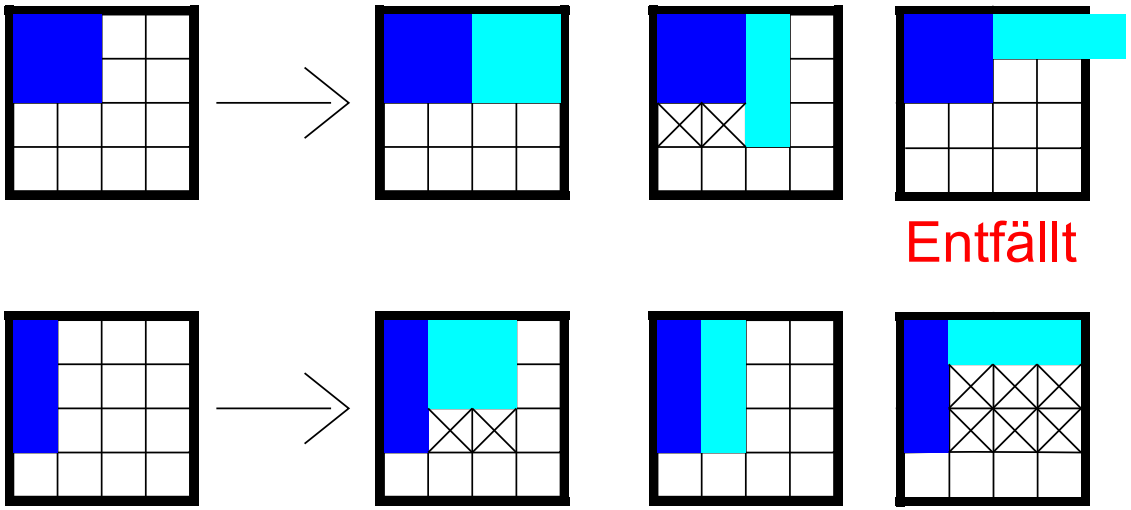




Horizontales Anfügen:

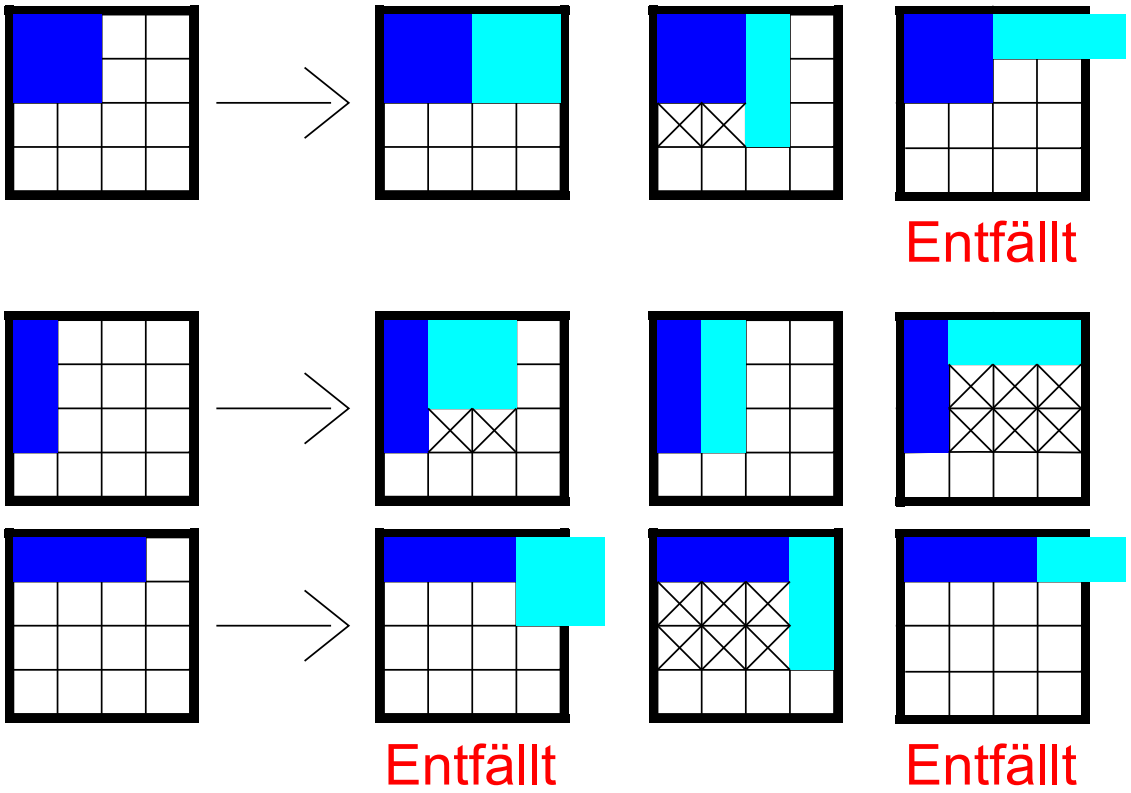
Horizontales Anfügen:



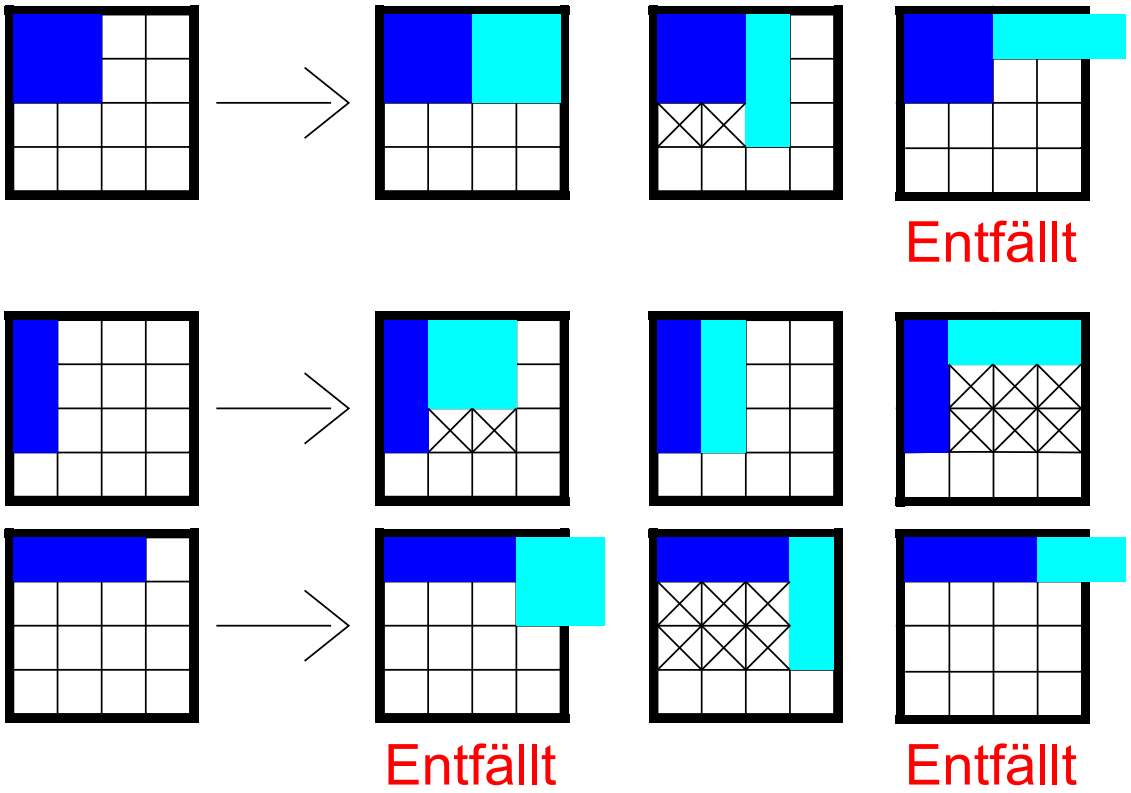
Horizontales Anfügen:



Horizontales Anfügen:

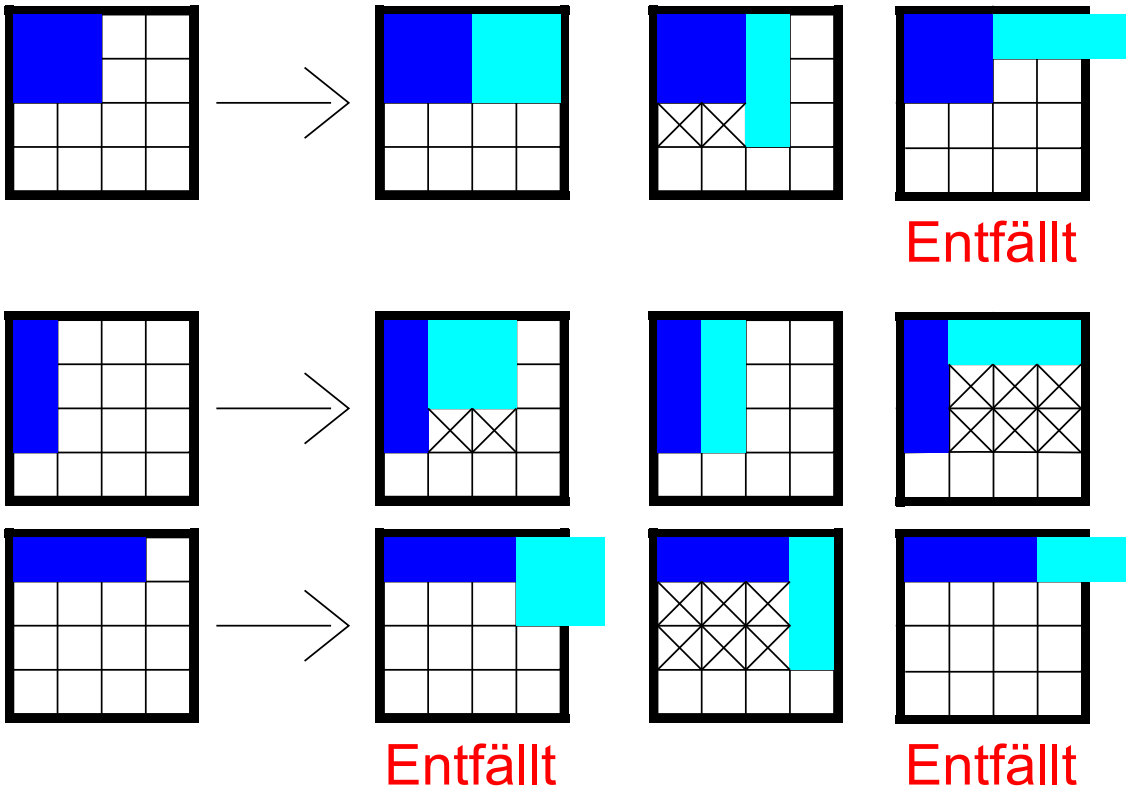


Horizontales Anfügen:



- Vorgabe Guillotine-Anordnung $\Longrightarrow$ Anordnung zu Rechteck aufgefüllt

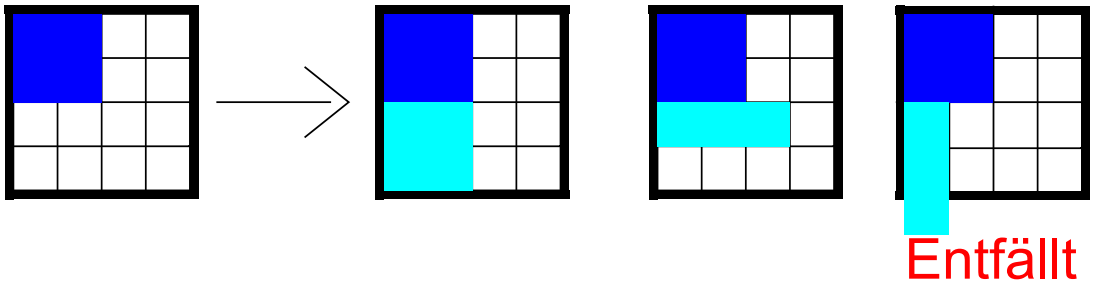
Horizontales Anfügen:



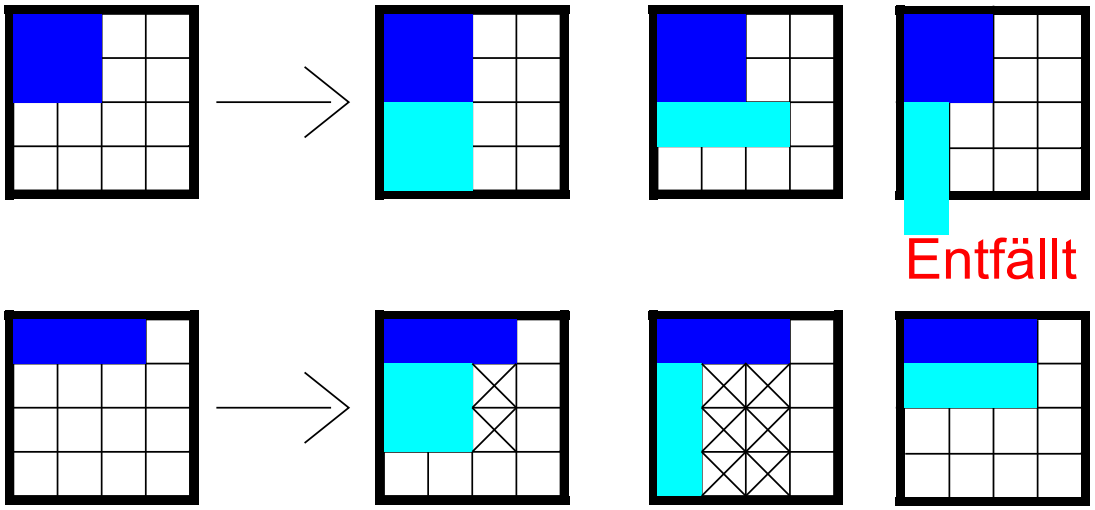
- Vorgabe Guillotine-Anordnung $\Longrightarrow$ Anordnung zu Rechteck aufgefüllt
- Herausragen über den Rand $\Longrightarrow$ Lösung ausschließen

Analog vertikale Anfügen:

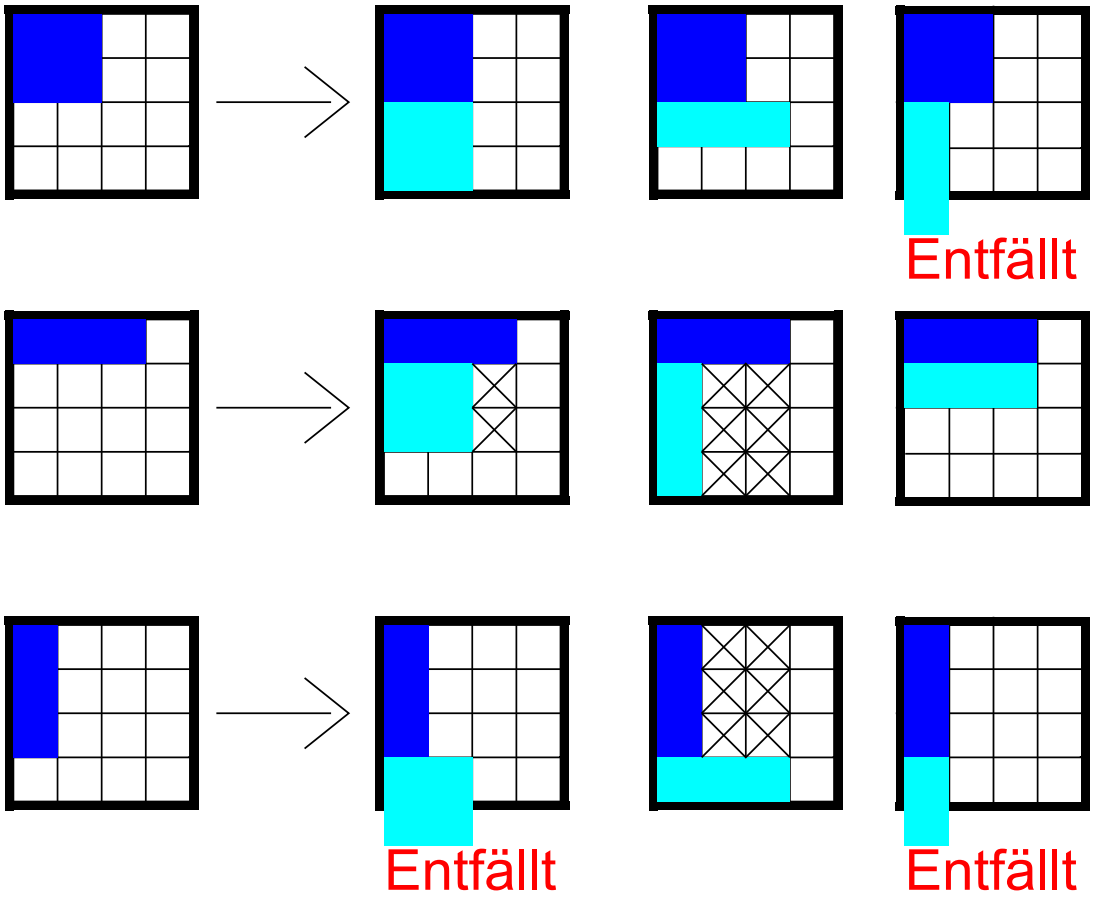
Analog vertikale Anfügen:



Analog vertikale Anfügen:

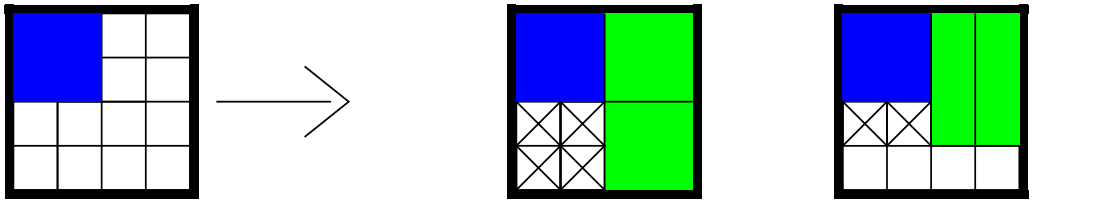


Analog vertikale Anfügen:

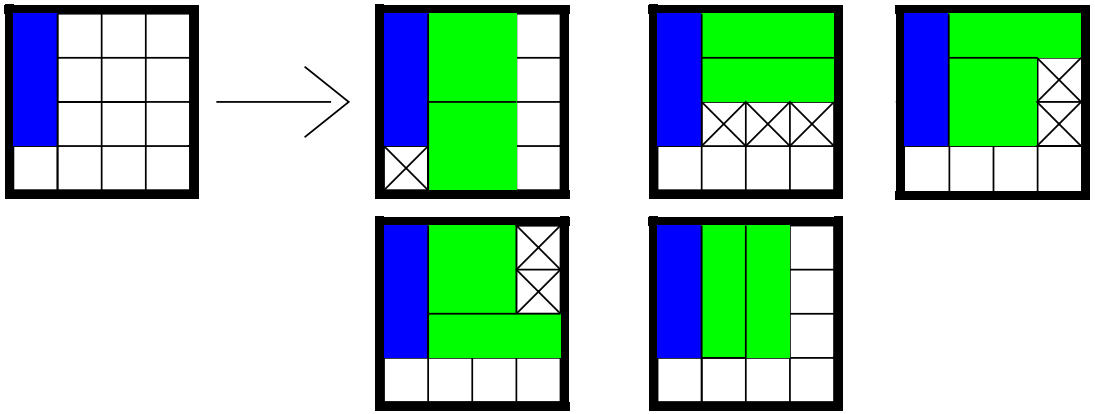


Zweiter Iterationsschritt: Anordnungen mit bis zu 2 Teilen aneinanderfügen

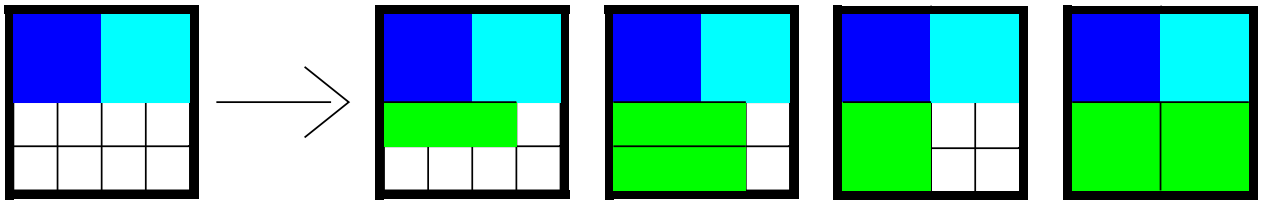
Zweiter Iterationsschritt: Anordnungen mit bis zu 2 Teilen aneinanderfügen



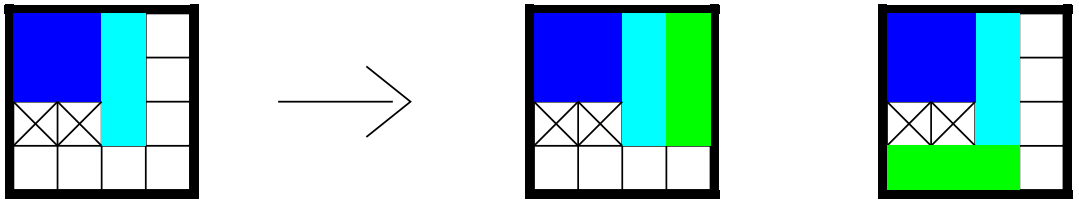
Entfällt (zuviele Teile)

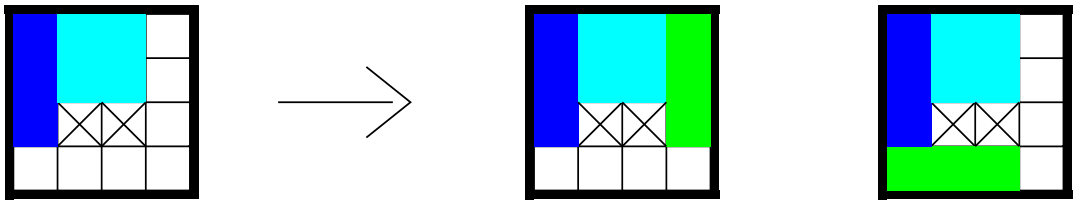


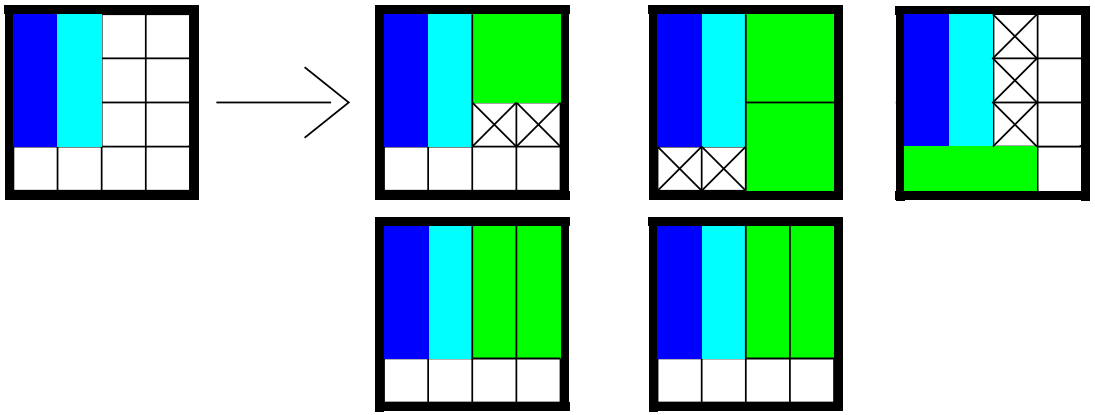
Entfällt (zuviele Teile)



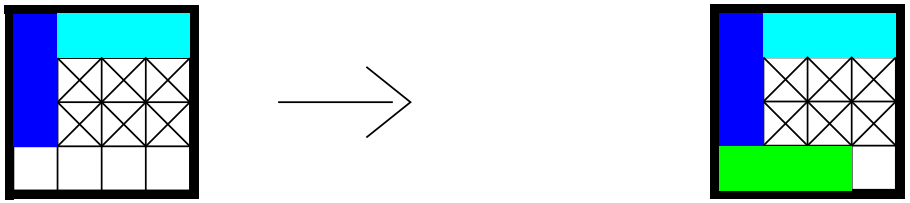
Entfallen (zuviele Teile)

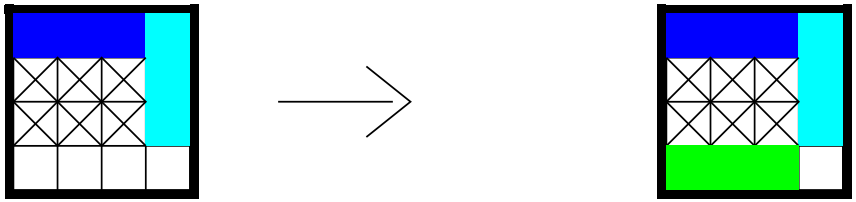






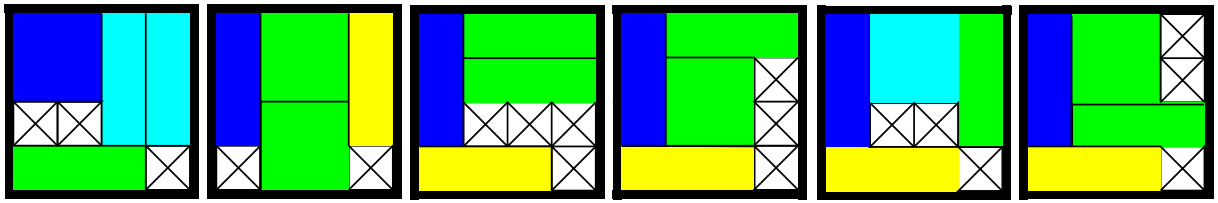
Entfallen (Zuviele Teile)





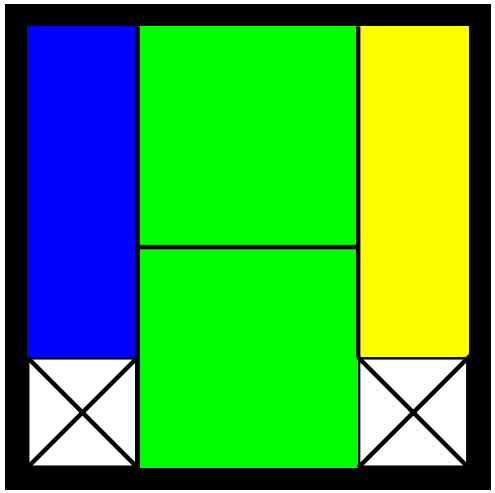
Dritter Iterationsschritt: Anordnungen mit bis zu 3 Teilen aneinanderfügen

Dritter Iterationsschritt: Anordnungen mit bis zu 3 Teilen aneinanderfügen

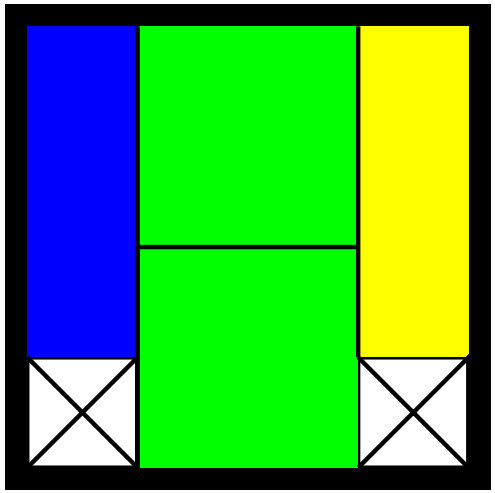


Keine weitere Anordnung möglich. Wähle eine Anordnung mit höchstem Wert:

Keine weitere Anordnung möglich. Wähle eine Anordnung mit höchstem Wert:



Keine weitere Anordnung möglich. Wähle eine Anordnung mit höchstem Wert:



Wert: 14, Abfall: 2

2.2. Formale Problembeschreibung

Definition

Ein „Rechtecktyp“ ist ein Tupel $R = (l, b, w, m) \in \mathbb{N}^3 \times (\mathbb{N} \cup \infty)$. Wir interpretieren dabei

Definition

Ein „Rechtecktyp“ ist ein Tupel $R = (l, b, w, m) \in \mathbb{N}^3 \times (\mathbb{N} \cup \infty)$. Wir interpretieren dabei

- Den Wert l als „Länge“ von R
- Den Wert b als „Breite“ von R
- Den Wert w als „Wert“ von R
- Den Wert m als maximale Anzahl verfügbarer Rechtecke des Typs R

Definition

Ein „Packungsproblem“ ist ein Tripel $P = (L, B, \mathcal{R})$, wobei $L \in \mathbb{N}$ („Länge“), $B \in \mathbb{N}$ („Breite“) und $\mathcal{R}$ eine endliche Menge von Rechtecktypen ist.

Definition

1. Eine „Anordnung“ von Rechtecken des Typs $R_i = (l_i, b_i, w_i, m_i)$ $(i = 1 \ldots n)$ ist ein Tupel $A = (l, b, w, a)$, wobei
   - $l, b, w \in \mathbb{N}$
   - $a = (a_1 \ldots a_n \in \mathbb{N}^n$ („Anzahlvektor“, Anzahl a_i der Rechtecke R_i in$A$

Definition

1. Eine „Anordnung“ von Rechtecken des Typs $R_i = (l_i, b_i, w_i, m_i)$ $(i = 1 \ldots n)$ ist ein Tupel $A = (l, b, w, a)$, wobei
   - $l, b, w \in \mathbb{N}$
   - $a = (a_1 \ldots a_n \in \mathbb{N}^n$ („Anzahlvektor“, Anzahl a_i der Rechtecke R_i in$A$

   Die Menge aller Anordnungen wird mit $\mathbb{A}$ bezeichnet

Definition

1. Eine „Anordnung“ von Rechtecken des Typs $R_i = (l_i, b_i, w_i, m_i)$ $(i = 1 \ldots n)$ ist ein Tupel $A = (l, b, w, a)$, wobei
   - $l, b, w \in \mathbb{N}$
   - $a = (a_1 \ldots a_n \in \mathbb{N}^n$ („Anzahlvektor“, Anzahl a_i der Rechtecke R_i in$A$

   Die Menge aller Anordnungen wird mit $\mathbb{A}$ bezeichnet
2. Die Anordnung $A = (l, b, w, a)$ von Rechtecken der Typen $\mathcal{R} = \{R_i | i = 1 \ldots n\}$ heißt „zulässig“ im Packungsproblem $P = (L, B, \mathcal{R})$, wenn

Definition

1. Eine „Anordnung“ von Rechtecken des Typs $R_i = (l_i, b_i, w_i, m_i)$ $(i = 1 \ldots n)$ ist ein Tupel $A = (l, b, w, a)$, wobei
   - $l, b, w \in \mathbb{N}$
   - $a = (a_1 \ldots a_n \in \mathbb{N}^n$ („Anzahlvektor“, Anzahl a_i der Rechtecke R_i in$A$

   Die Menge aller Anordnungen wird mit $\mathbb{A}$ bezeichnet
2. Die Anordnung $A = (l, b, w, a)$ von Rechtecken der Typen $\mathcal{R} = \{R_i | i = 1 \ldots n\}$ heißt „zulässig“ im Packungsproblem $P = (L, B, \mathcal{R})$, wenn
   - $l \leq L$
   - $b \leq B$
   - $a \leq m$, d.h. $\forall i = 1 \ldots n : a_i \leq m_i$

Definition

1. Eine „Anordnung“ von Rechtecken des Typs $R_i = (l_i, b_i, w_i, m_i)$ $(i = 1 \ldots n)$ ist ein Tupel $A = (l, b, w, a)$, wobei
   - $l, b, w \in \mathbb{N}$
   - $a = (a_1 \ldots a_n \in \mathbb{N}^n$ („Anzahlvektor“, Anzahl a_i der Rechtecke R_i in$A$

   Die Menge aller Anordnungen wird mit $\mathbb{A}$ bezeichnet
2. Die Anordnung $A = (l, b, w, a)$ von Rechtecken der Typen $\mathcal{R} = \{R_i | i = 1 \ldots n\}$ heißt „zulässig“ im Packungsproblem $P = (L, B, \mathcal{R})$, wenn
   - $l \leq L$
   - $b \leq B$
   - $a \leq m$, d.h. $\forall i = 1 \ldots n : a_i \leq m_i$

   Die Menge der zulässigen Anordnungen in P wird mit $\mathbb{A}_P$ bezeichnet.

Definition

Eine Anordnung $A = (l, b, w, a) \in \mathbb{A}_P$ heißt „Lösung“ von P, wenn A maximalen Wert w bezüglich der Menge aller zulässigen Anordnungen aus A_P hat.

Definition

Sei $P = (L, B, \mathcal{R})$ ein Packungsproblem mit $\mathcal{R} = \{R_i \mid i \in \{1, \ldots, n\}\}$.
Für jedes $R_i = (l_i, b_i, w_i, m_i) \in \mathcal{R}$ heißt

$$A_i = (l_i, b_i, w_i, e_i)$$

primitive Anordnung.
$\mathbb{A}_{prim}$ ist die Menge aller primitiver Anordnungen.

Definition

Wir erklären den *horizontalen Kombinationsoperator*

$$\begin{array}{rcl} +_h\colon \quad \mathbb{A} \times \mathbb{A} & \to & \mathbb{A} \\ ((l,b,w,a),(l',b',w',a')) & \mapsto & (\max l,l',b+b',w+w',a+a') \end{array}$$

und den *vertikalen Kombinationsoperator*

$$\begin{array}{rcl} +_v\colon \quad \mathbb{A} \times \mathbb{A} & \to & \mathbb{A} \\ ((l,b,w,a),(l',b',w',a')) & \mapsto & (l+l',\max(b,b'),w+w',a+a'). \end{array}$$

Für $\mathbb{B},\mathbb{C} \subset \mathbb{A}$ schreiben wir $\mathbb{B} +_h \mathbb{C}$ bzw. $\mathbb{B} +_v \mathbb{C}$ für das elementweise Operieren.

Definition

Der Operator

$$\begin{aligned} \text{Prüf}: \quad \wp(\mathbb{A}) &\rightarrow \wp(\mathbb{A}_P) \\ X &\mapsto X \cap \mathbb{A}_P \end{aligned}$$

heißt *Prüfoperator* und gibt aus einer Menge von Anordnungen die Menge der zulässigen Anordnungen zurück.

Definition

Der Operator

$$\begin{aligned} \text{Opt}: \quad \wp(\mathbb{A}_P) &\rightarrow \wp(\mathbb{A}_P) \\ X &\mapsto \{A = (l, b, w, a) \in X | w \text{ ist maximal}\} \end{aligned}$$

gibt aus einer Menge von Anordnungen die Menge der optimalen Anordnungen zurück.

Algorithmus

(1) $\Omega_0 := \mathbb{A}_{prim}$

(2) $\Omega_{i+1} := \text{Prüf}((\Omega_i +_v \bigcup_{j=1}^{i} \Omega_j) \cup (\Omega_i +_h \bigcup_{j=1}^{i} \Omega_j))$

(3) $\Omega_{i+1} = \emptyset \rightarrow (4)$
$\Omega_{i+1} \neq \emptyset \rightarrow (2)$

(4) *Ausgabe:* $\text{Opt}(\Omega_i)$, w

Wir beschränken uns auf den Fall $w = l \cdot b$.

- relativer Verschnitt
- absoluter Verschnitt

Definition

Der Operator

$$\begin{aligned} \text{PrufRel}_{\beta_r}: \quad \wp(\mathbb{A}) &\rightarrow \wp(\mathbb{A}_P) \\ X &\mapsto \left\{ A = (l, b, w, a) \in X | A \in \mathbb{A}_P, \frac{l \cdot b - w}{l \cdot b} \leq \beta_r \right\} \end{aligned}$$

heißt *relativer Prüfoperator* und gibt aus einer Menge von Anordnungen die Menge der zulässigen Lösungen mit hinreichend kleinem relativem Abfall zurückgibt.

Definition

Der Operator

$$\begin{aligned} \text{PrufAbs}_{\beta_a}: \quad \wp(\mathbb{A}) &\rightarrow \wp(\mathbb{A}_P) \\ X &\mapsto \left\{ A = (l, b, w, a) \in X | A \in \mathbb{A}_P, \frac{l \cdot b - w}{L \cdot B} \leq \beta_a \right\} \end{aligned}$$

heißt *absoluter Prüfoperator* und gibt aus einer Menge von Anordnungen die Menge der zulässigen Lösungen mit hinreichend kleinem absolutem Abfall zurückgibt.

Algorithmus

(1) $\Omega_0 := \mathbb{A}_{prim}$

(2) $\Omega_{i+1} := \text{PrüfRel}_{\beta_r}((\Omega_i +_v \bigcup_{j=1}^{i} \Omega_j) \cup (\Omega_i +_h \bigcup_{j=1}^{i} \Omega_j))$

(3) $\Omega_{i+1} = \emptyset \rightarrow (4)$
$\Omega_{i+1} \neq \emptyset \rightarrow (2)$

(4) *Ausgabe:* $\text{Opt}(\Omega_i), w$

Algorithmus

(1) $\Omega_0 := \mathbb{A}_{prim}$

(2) $\Omega_{i+1} := \text{PrüfAbs}_{\beta_a}((\Omega_i +_v \bigcup_{j=1}^{i} \Omega_j) \cup (\Omega_i +_h \bigcup_{j=1}^{i} \Omega_j))$

(3) $\Omega_{i+1} = \emptyset \rightarrow (4)$
$\Omega_{i+1} \neq \emptyset \rightarrow (2)$

(4) *Ausgabe:* $\text{Opt}(\Omega_i), w$

Beachte: Je kleiner β_r bzw. β_a, desto größer die Wahrscheinlichkeit, dass zuviele Anordnungen aussortiert werden und die optimale Anordnung nicht gefunden werden kann.

Definition

Das kleinstmögliche β_r bzw. β_a, für das das Verfahren die optimale Anordnung zurückgibt, wird mit β_r^* bzw. β_a^* bezeichnet.

Satz

Bei der Methode des absoluten Verschnitts gilt für die optimale Fehlertoleranz

$$\beta_a^* \leq \frac{V}{L \cdot B}$$

und bei der Methode des relativen Verschnitts gilt

$$\beta_r^* \leq \frac{V}{l_0 \cdot b_0},$$

dabei ist $V = L \cdot B - (l \cdot b - w)$ der Verschnitt einer beliebigen Anordnung (l, b, w, a) und (l_0, b_0, w_0, a_0) ist eine Anordnung, die als horizontale oder vertikale Kombination zweier Grundrechtecke bzw. primitver Anordnungen entsteht, mit minimaler Fläche $l_0 \cdot b_0$ bzw. minimalem Wert w_0.

Definition

Zwei Anordnungen $A_1 = (l_1, b_1, w_1, m_1)$ und $A_2 = (l_2, b_2, w_2, m_2)$ heißen *äquivalent*, wenn $l_1 = l_2$, $b_1 = b_2$, $w_1 = w_2$ und $m_1 = m_2$ gelten. Wir schreiben dafür $A_1 \approx A_2$ und entsprechend ist $[A] := A\approx$ die Äquivalenzklasse aller Anordnungen, die zu A äquivalent sind.

Weiter setzen wir die Operatoren horizontale Kombination $+_h$ und vertikale Kombination $+_v$ auf die Faktormenge $\mathbb{A}/\approx$ aller Äquivalenzklassen fort, vermöge

$$[A_1] +_h [A_2] := [A_1 +_h A_2]$$

und

$$[A_1] +_v [A_2] := [A_1 +_v A_2].$$

Analog setzen wir noch

$$\text{Prüf}\{[A_i] \mid i \in I\} := \{[\text{Prüf}\{A_i\}] \mid i \in I\}$$

sowie

$$\text{Opt}\{[A_i] \mid i \in I\} := \{[\text{Opt}\{A_i\}] \mid i \in I\}$$

und entsprechend auch für PrüfRel_{β_r} und PrüfAbs_{β_a}. Dann können wir den Suchraum weiter einschränken, indem mit den Äquivalenzklassen gerechnet wird.

Algorithmus

(1) $\Omega_0 := \mathbb{A}_{prim}/\approx$

(2) $\Omega_{i+1} := \text{PrüfRel}_{\beta_r}((\Omega_i +_v \bigcup_{j=1}^{i} \Omega_j) \cup (\Omega_i +_h \bigcup_{j=1}^{i} \Omega_j))$

(3) $\Omega_{i+1} = \emptyset \rightarrow (4)$
$\Omega_{i+1} \neq \emptyset \rightarrow (2)$

(4) *Ausgabe:* $\text{Opt}(\Omega_i), w$

Algorithmus

(1) $\Omega_0 := \mathbb{A}_{prim}/\approx$

(2) $\Omega_{i+1} := \text{PrüfAbs}_{\beta_a}((\Omega_i +_v \bigcup_{j=1}^{i} \Omega_j) \cup (\Omega_i +_h \bigcup_{j=1}^{i} \Omega_j))$

(3) $\Omega_{i+1} = \emptyset \rightarrow (4)$
$\Omega_{i+1} \neq \emptyset \rightarrow (2)$

(4) *Ausgabe:* $\text{Opt}(\Omega_i), w$

Definition

Eine Anordnung $A_1 = (l_1, b_1, w_1, m_1)$ mit dem Verschnitt $v_1 = l_1 \cdot b_1 - w_1$ heißt *besser* als eine Anordnung $A_2 = (l_2, b_2, w_2, m_2)$ mit Verschnitt $v_2 = l_2 \cdot b_2 - w_2$, falls $v_1 \leq v_2$ und $m_1 = m_2$ gilt. Symbol: $A_1 \precsim A_2$. Für eine Anordnung A definieren wir ΥA als diejenige Anordnung, für die keine Anordnung A_0 mit $A_0 \precsim \Upsilon A$ existiert.

Wir setzen

$$\Upsilon\{A_i \mid i \in I\} := \{\Upsilon A_i \mid i \in I\}$$

und

$$\Upsilon[A] := [\Upsilon A]$$

sowie entsprechend

$$\Upsilon\{[A_i] \mid i \in I\} := \{[\Upsilon A_i] \mid i \in I\}$$

. Damit lässt sich der Suchraum des Algorithmus weiter einschränken, indem nur die relativ besten Anordnungen verwendet werden.

Algorithmus

(1) $\Omega_0 := \gamma\,\mathbb{A}_{prim}/\approx$

(2) $\Omega_{i+1} := \gamma\,\mathrm{PrüfRel}_{\beta_r}((\Omega_i +_v \bigcup_{j=1}^{i}\Omega_j) \cup (\Omega_i +_h \bigcup_{j=1}^{i}\Omega_j))$

(3) $\Omega_{i+1} = \emptyset \rightarrow (4)$
$\Omega_{i+1} \neq \emptyset \rightarrow (2)$

(4) *Ausgabe:* $\mathrm{Opt}(\Omega_i), w$

Algorithmus

(1) $\Omega_0 := \gamma\,\mathbb{A}_{prim}/\approx$

(2) $\Omega_{i+1} := \gamma\,\mathrm{PrüfAbs}_{\beta_a}((\Omega_i +_v \bigcup_{j=1}^{i}\Omega_j) \cup (\Omega_i +_h \bigcup_{j=1}^{i}\Omega_j))$

(3) $\Omega_{i+1} = \emptyset \rightarrow (4)$
$\Omega_{i+1} \neq \emptyset \rightarrow (2)$

(4) *Ausgabe:* $\mathrm{Opt}(\Omega_i), w$

P. Y. Wang.
Two algorithms for constrained two-dimensional cutting stock problems.
Operations Research, 31:573–586, 1983.

G. Scheithauer.
Zuschnitt- und Packungsoptimierung: Problemstellungen, Modellierungstechniken, Lösungsmethoden.
Vieweg+Teubner, 2008.