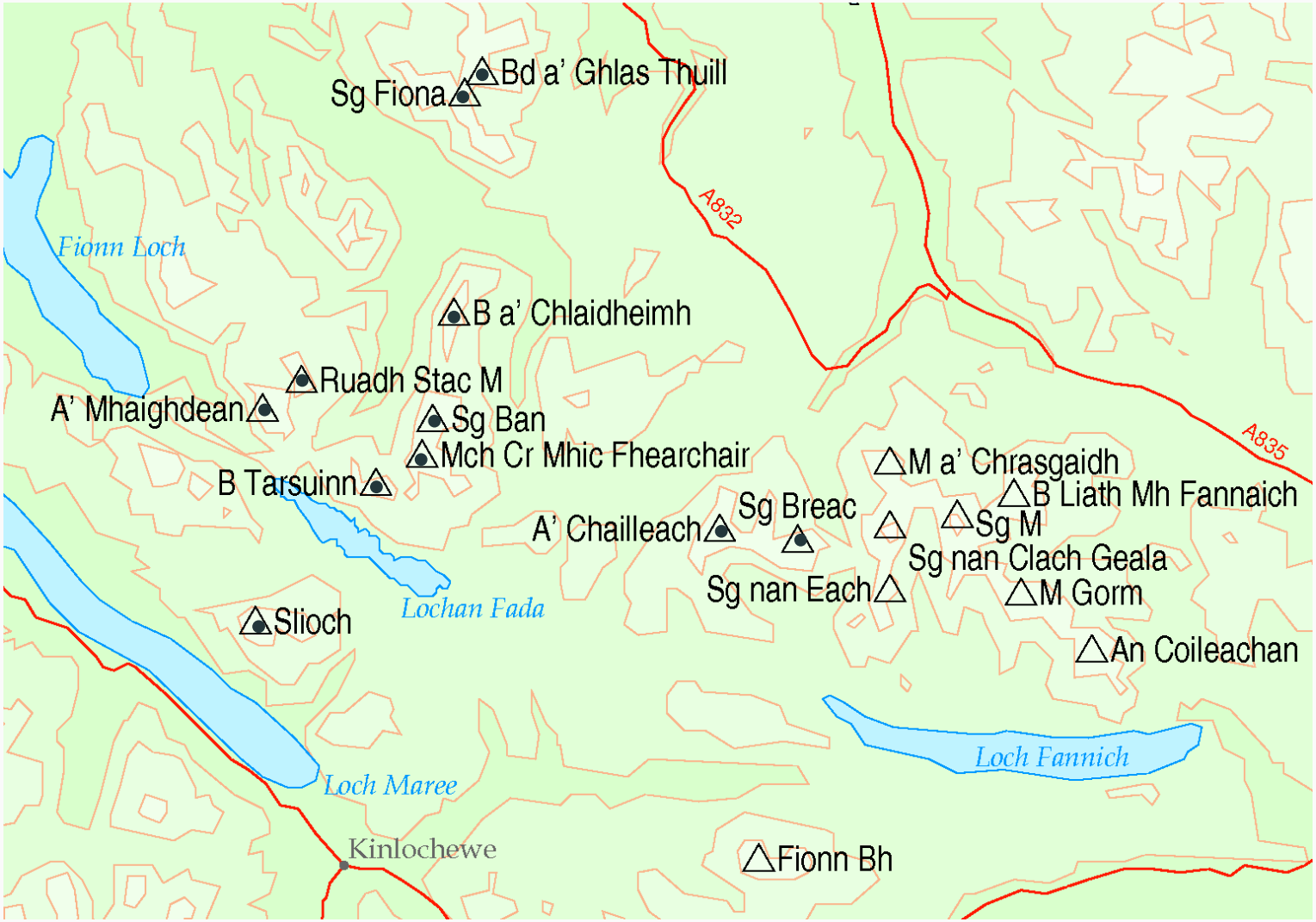
Label Placement by Maximum Independent Set in Rectangles

Moritz Halm

25. Juni 2018

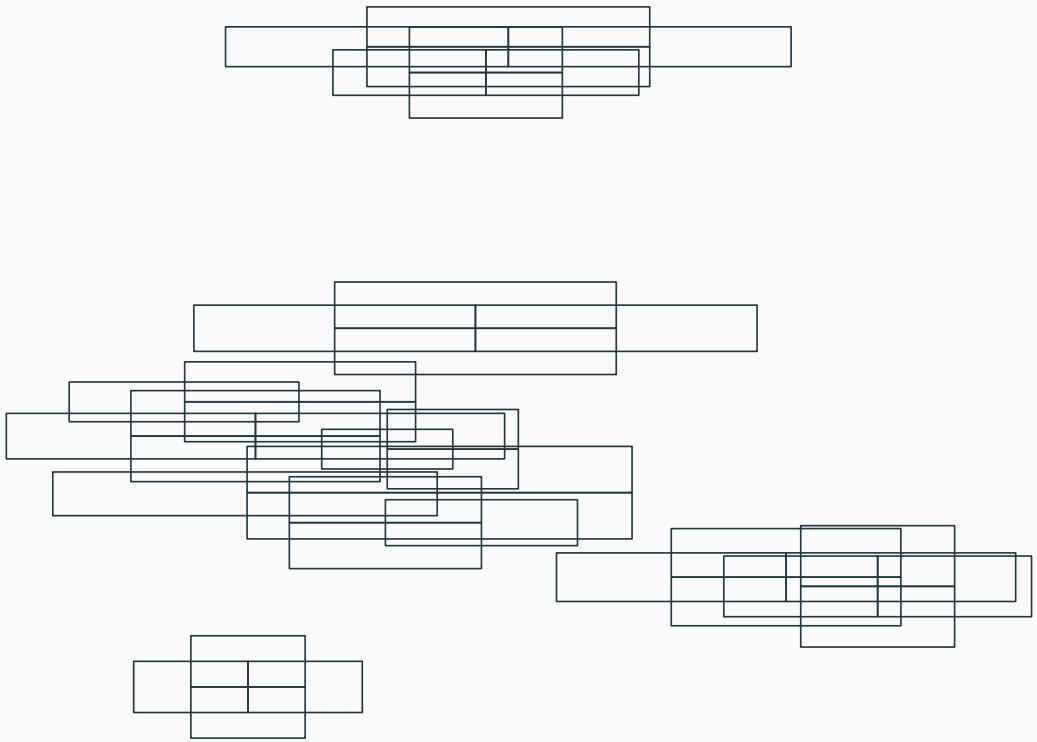
ITI Wagner, Proseminar Algorithmen für NP-schwere Probleme

Beschrifte Karten



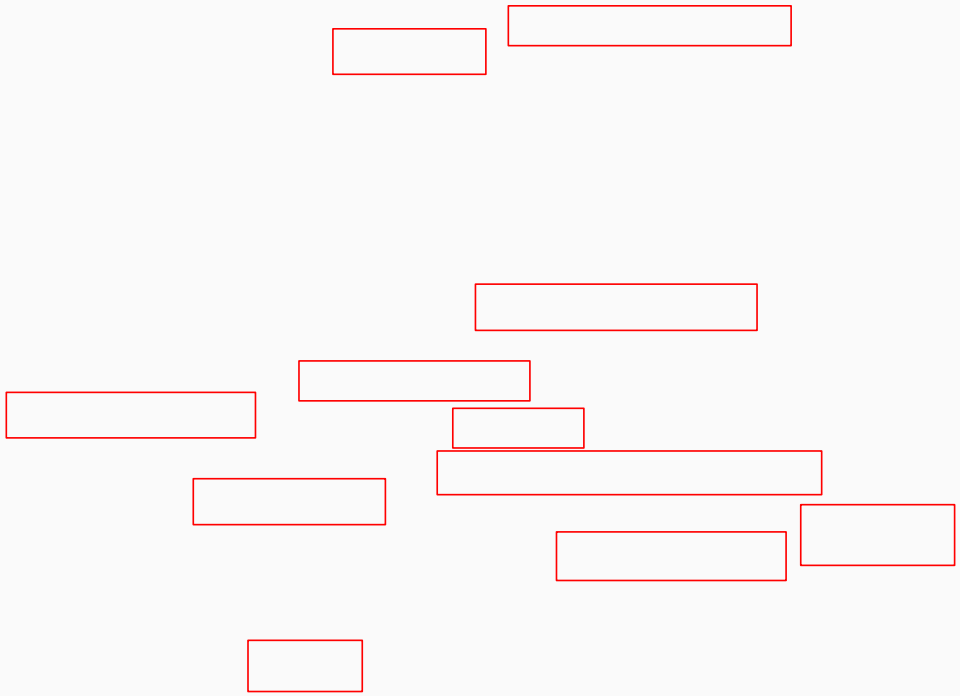
Motivation

Beschrifte Karten



Motivation

Beschrifte Karten



Andere Ansätze

Für **quadratische** Labels:

- Exakter superpolynomieller Algorithmus [Kuc+93]
- 2-Approximation für variable Labelgröße in $\mathcal{O}(n \log n)$ (optimal) [FW91]

Problemstellung

Maximum Independet Set in Rectangles

Geg. Menge von n achsenparallelen Rechtecken R,
finde $I \subseteq R : \forall r, s \in I, r \neq s : r \cap s = \emptyset$, sodass $|I|$ maximal.

Problemstellung

Maximum Independet Set in Rectangles

Geg. Menge von n achsenparallelen Rechtecken R,
finde $I \subseteq R : \forall r, s \in I, r \neq s : r \cap s = \emptyset$, sodass $|I|$ maximal.

- NP-schwer

Problemstellung

Maximum Independet Set in Rectangles

Geg. Menge von n achsenparallelen Rechtecken R,
finde $I \subseteq R : \forall r, s \in I, r \neq s : r \cap s = \emptyset$, sodass $|I|$ maximal.

- NP-schwer
- Aber approximierbar (Hier vorgestellt [AKS98])

Gliederung

Eindimensionaler Fall

Beliebige achsenparallele Rechtecke: $1/\log_2(n)$-Approximation

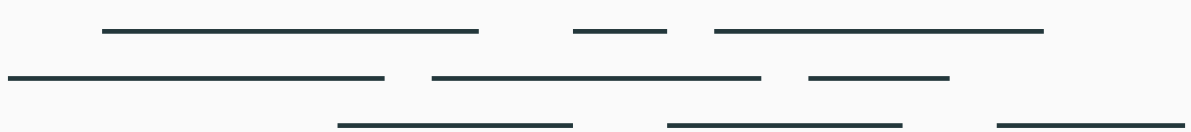
Rechtecke gleicher Höhe: $1/(1 + 1/k)$-Approximation

Eindimensionaler Fall

Aufwärmübung

Activity Selection Problem

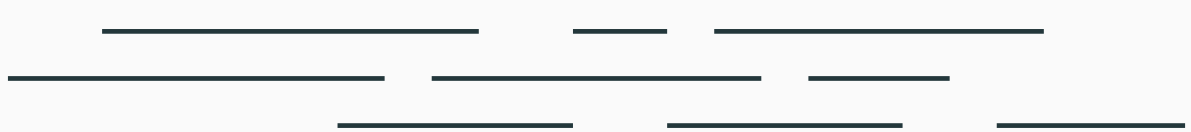
Finde für eine Menge M von Intervallen („Aktivitäten“) $(begin_i, end_i)$ Teilmenge $I \subseteq M$, mit $\forall r, s \in I, r \neq s : r \cap s = \emptyset$ und $|I|$ maximal.



Aufwärmübung

Activity Selection Problem

Finde für eine Menge M von Intervallen („Aktivitäten“) $(begin_i, end_i)$ Teilmenge $I \subseteq M$, mit $\forall r, s \in I, r \neq s : r \cap s = \emptyset$ und $|I|$ maximal.



Optimal in $\mathcal{O}(n \log n)$Zeit lösbar.

Algorithmus

Algorithmus: ActivitySelector

```
Sortiere M nach Endpunkten
I = {M[0]}
k = 0
for i from 1 to n:
    if begin[i] ≥ end[k]:
        k = i
        I = I ∪ {M[i]}
return I
```

Beispiel

Beispiel

Beispiel

Beispiel

Beispiel

Beispiel

Beliebige achsenparallele Rechtecke: $1/\log_2(n)$-Approximation

Divide-and-Conquer-Algorithmus

Vorberechnung: Sortiere alle vertikalen Kanten nach x-Koordinate.

Divide-and-Conquer-Algorithmus

Vorberechnung: Sortiere alle vertikalen Kanten nach x-Koordinate.

1. Falls $n = |R| \leq 2$, berechne I direkt (Basisfall)

Divide-and-Conquer-Algorithmus

Vorberechnung: Sortiere alle vertikalen Kanten nach x-Koordinate.

1. Falls $n = |R| \leq 2$, berechne I direkt (Basisfall)
2. Sei $\tilde{x}$ Median aller x-Koordinaten und ℓ: $x = \tilde{x}$ Linie durch $\tilde{x}$.

Divide-and-Conquer-Algorithmus

Vorberechnung: Sortiere alle vertikalen Kanten nach x-Koordinate.

1. Falls $n = |R| \leq 2$, berechne I direkt (Basisfall)
2. Sei $\tilde{x}$ Median aller x-Koordinaten und $\ell\colon x = \tilde{x}$ Linie durch $\tilde{x}$.
3. Partitioniere R:
   $R_1 := \{r \in R \mid r \text{ links von } \ell\}$
   $R_2 := \{r \in R \mid r \text{ rechts von } \ell\}$
   $R_{12} := \{r \in R \mid r \text{ schneidet } \ell\}$

Divide-and-Conquer-Algorithmus

Vorberechnung: Sortiere alle vertikalen Kanten nach x-Koordinate.

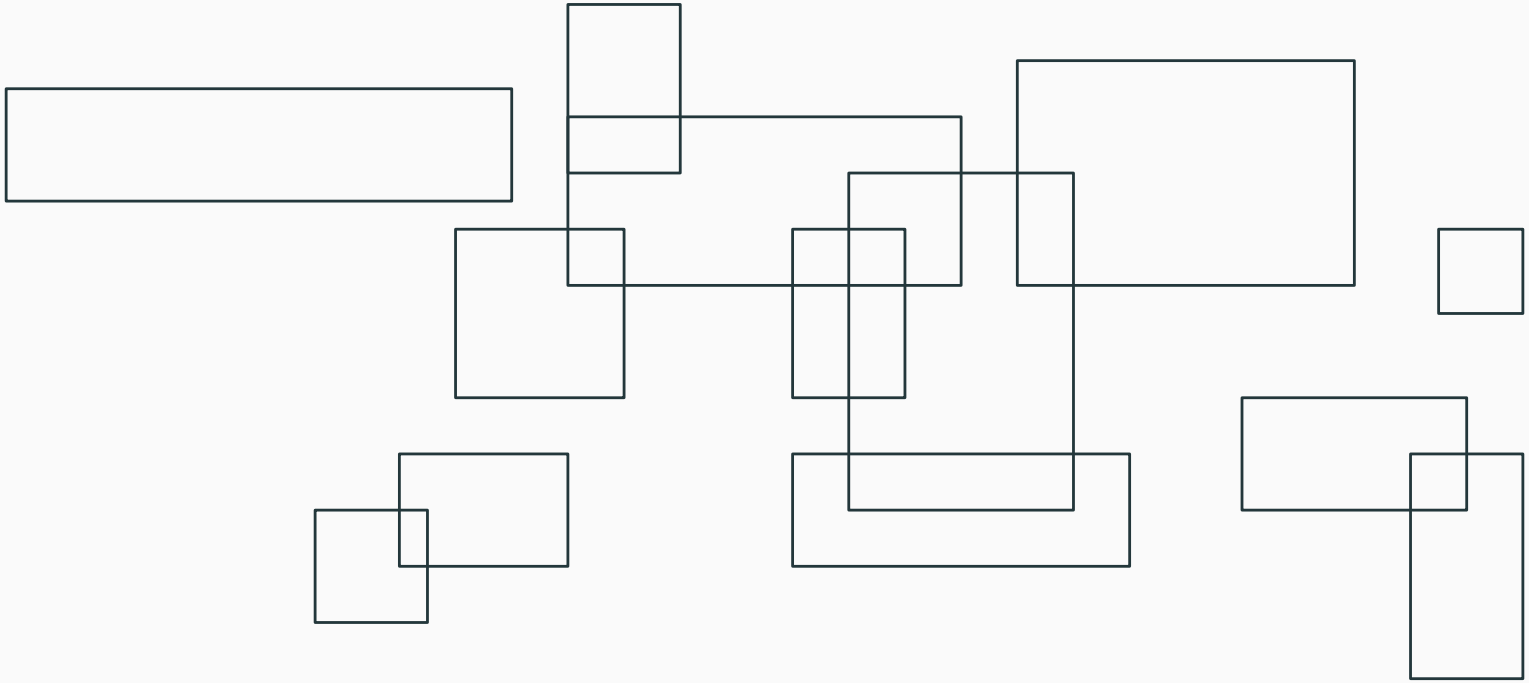
1. Falls $n = |R| \leq 2$, berechne I direkt (Basisfall)
2. Sei $\tilde{x}$ Median aller x-Koordinaten und $\ell\colon x = \tilde{x}$ Linie durch $\tilde{x}$.
3. Partitioniere R:
   $R_1 := \{r \in R \mid r \text{ links von } \ell\}$
   $R_2 := \{r \in R \mid r \text{ rechts von } \ell\}$
   $R_{12} := \{r \in R \mid r \text{ schneidet } \ell\}$
4. Berechne unabhängige Teilmengen $I_1 := I(R_1)$ und $I_2 := I(R_2)$ rekursiv, $I_{12} := I(R_{12})$ direkt mit ActivitySelector.

Divide-and-Conquer-Algorithmus

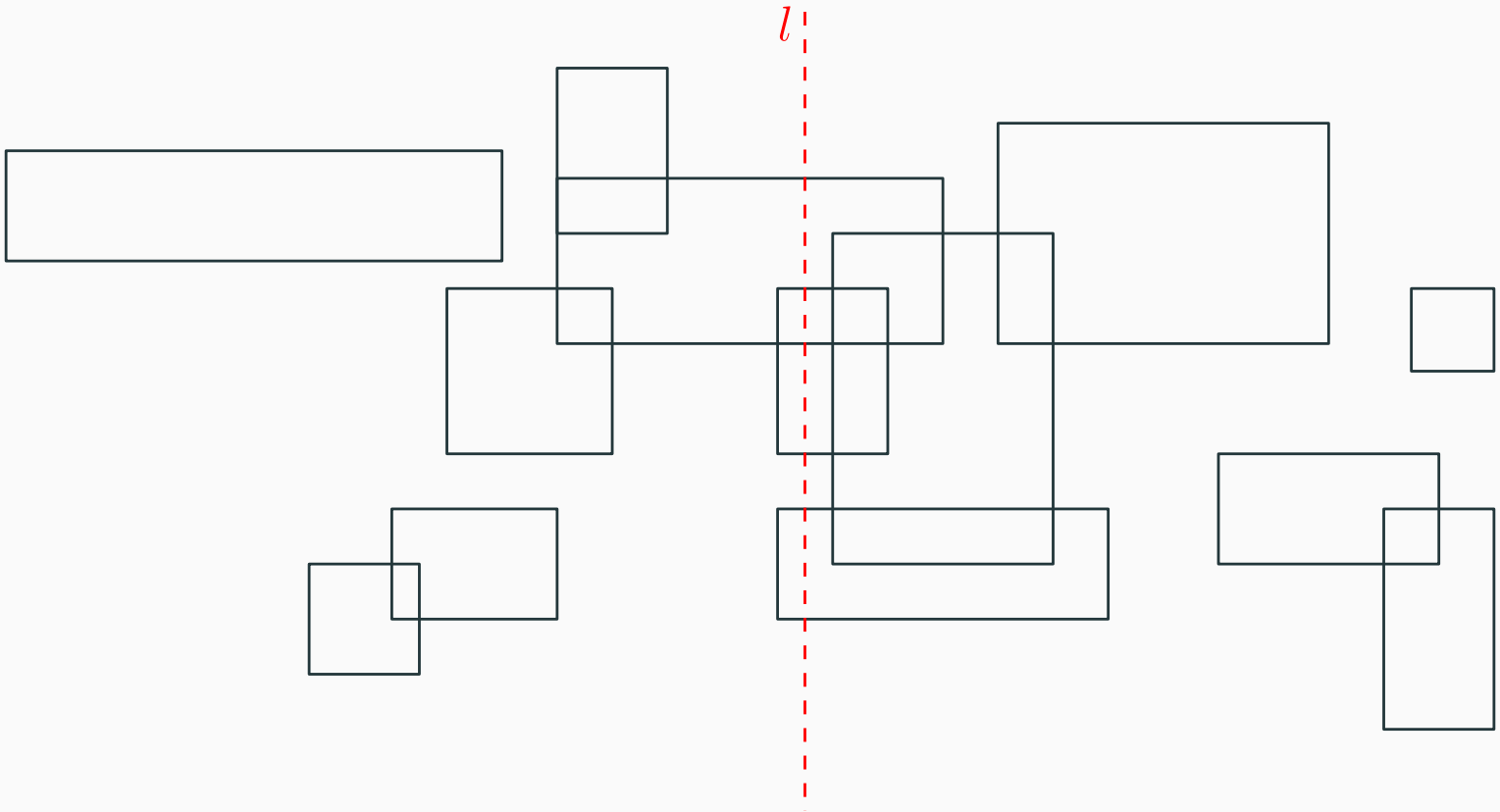
Vorberechnung: Sortiere alle vertikalen Kanten nach x-Koordinate.

1. Falls $n = |R| \leq 2$, berechne I direkt (Basisfall)
2. Sei $\tilde{x}$ Median aller x-Koordinaten und $\ell\colon x = \tilde{x}$ Linie durch $\tilde{x}$.
3. Partitioniere R:
   $R_1 := \{r \in R \mid r \text{ links von } \ell\}$
   $R_2 := \{r \in R \mid r \text{ rechts von } \ell\}$
   $R_{12} := \{r \in R \mid r \text{ schneidet } \ell\}$
4. Berechne unabhängige Teilmengen $I_1 := I(R_1)$ und $I_2 := I(R_2)$ rekursiv, $I_{12} := I(R_{12})$ direkt mit ActivitySelector.
5. Wenn $|I_{12}| > |I_1| + |I_2|$ gib I_{12} zurück, sonst $I_1 \cup I_2$.

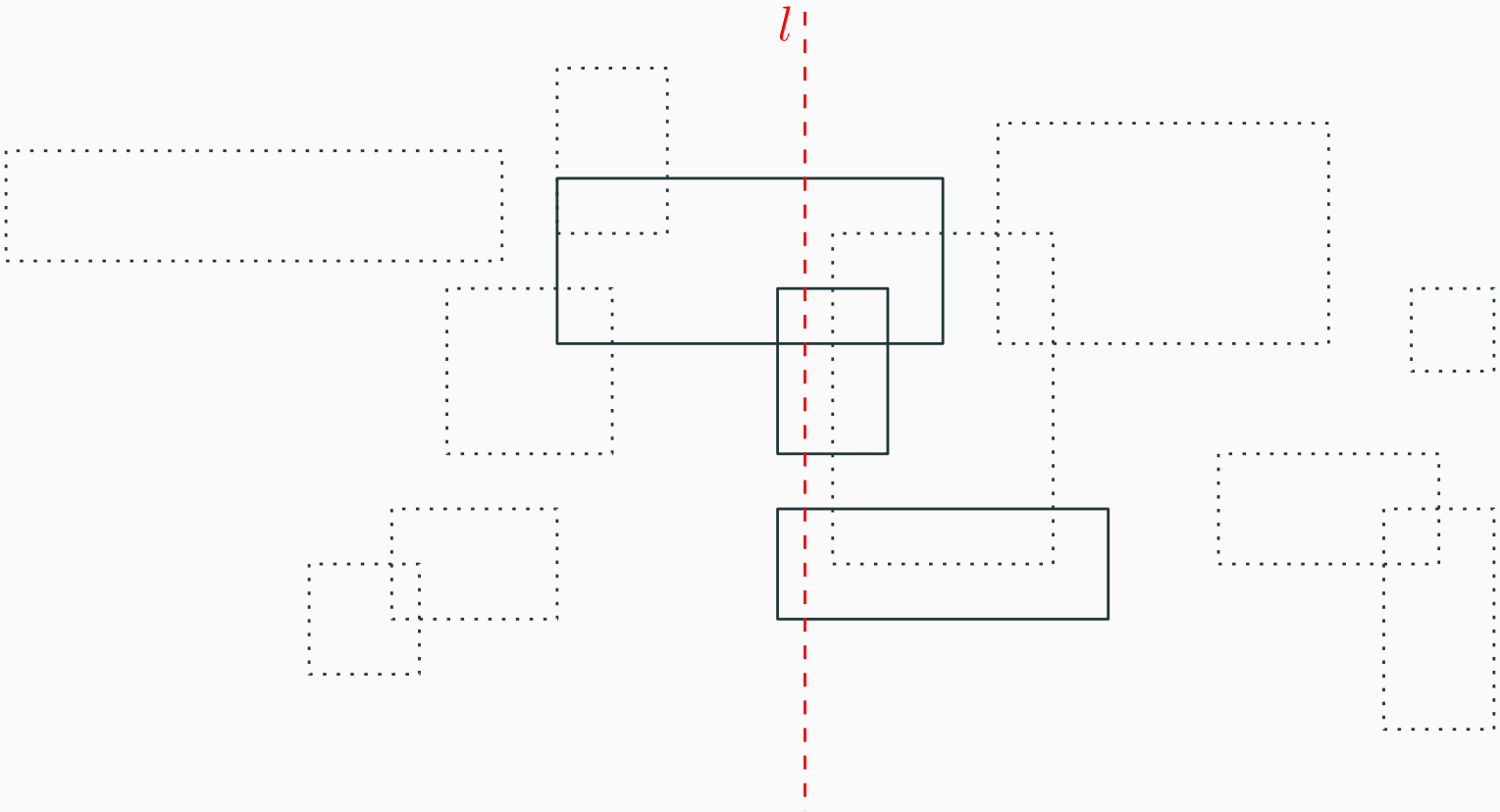
Beispiel



Beispiel

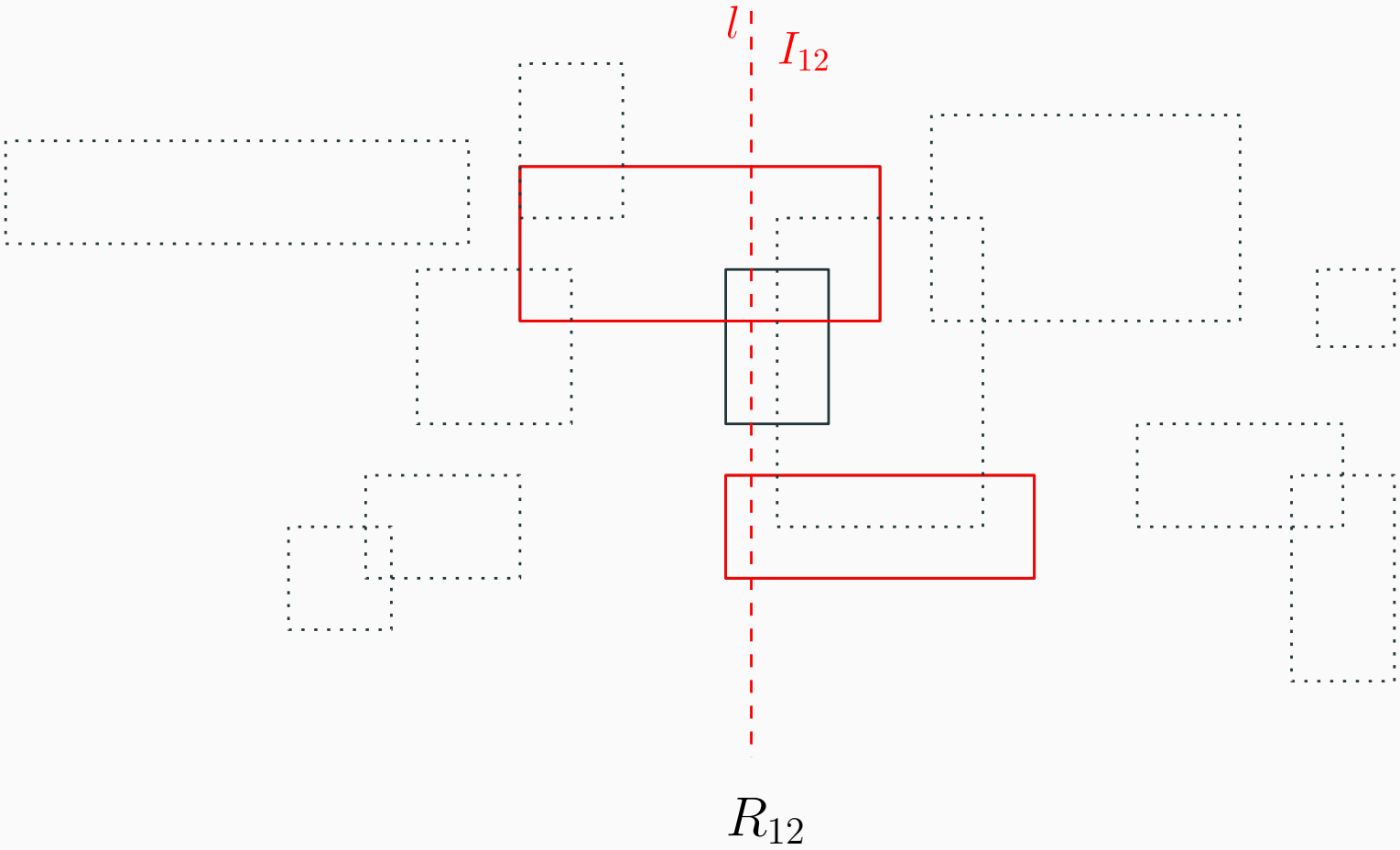


Beispiel

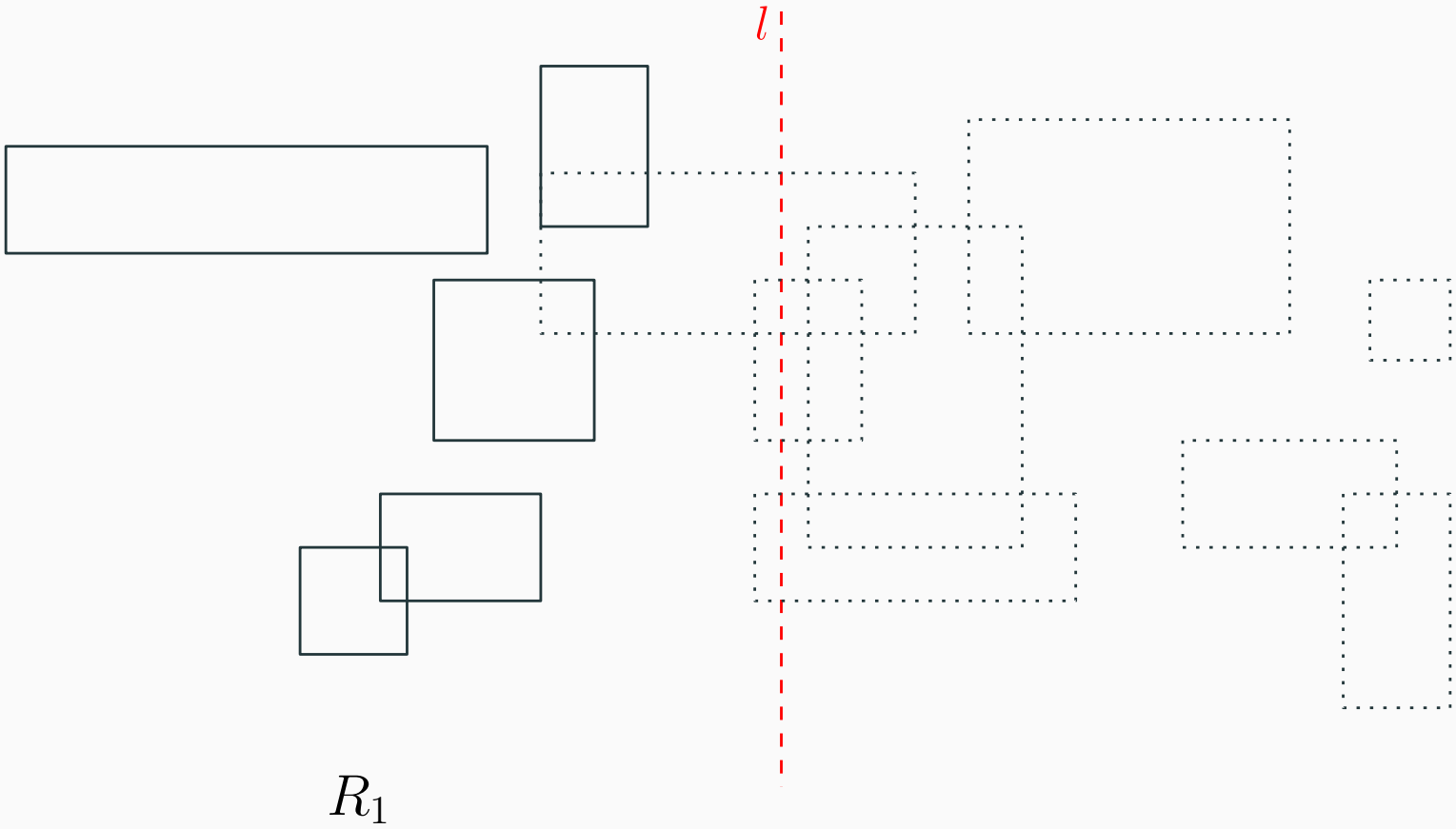


R_{12}

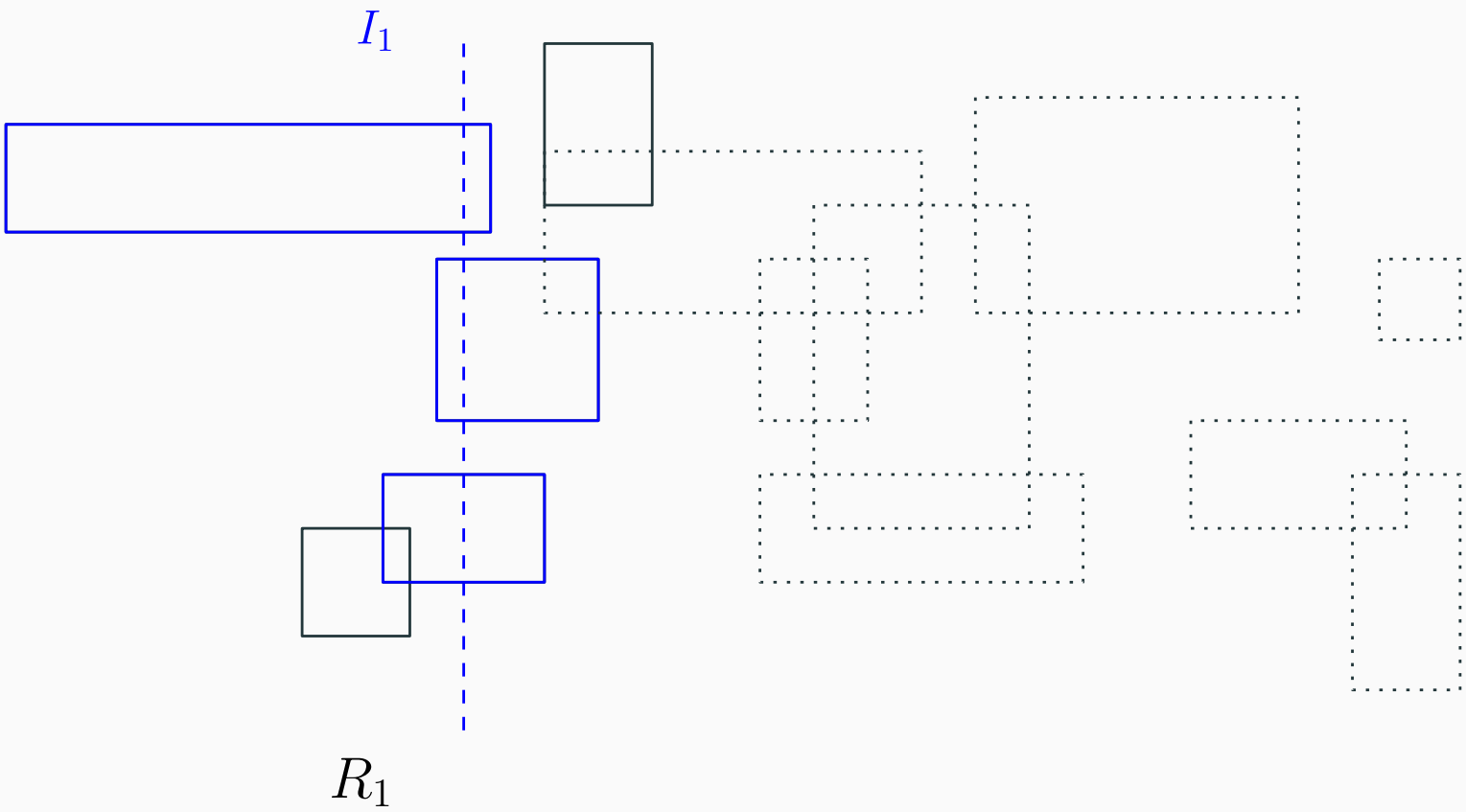
Beispiel



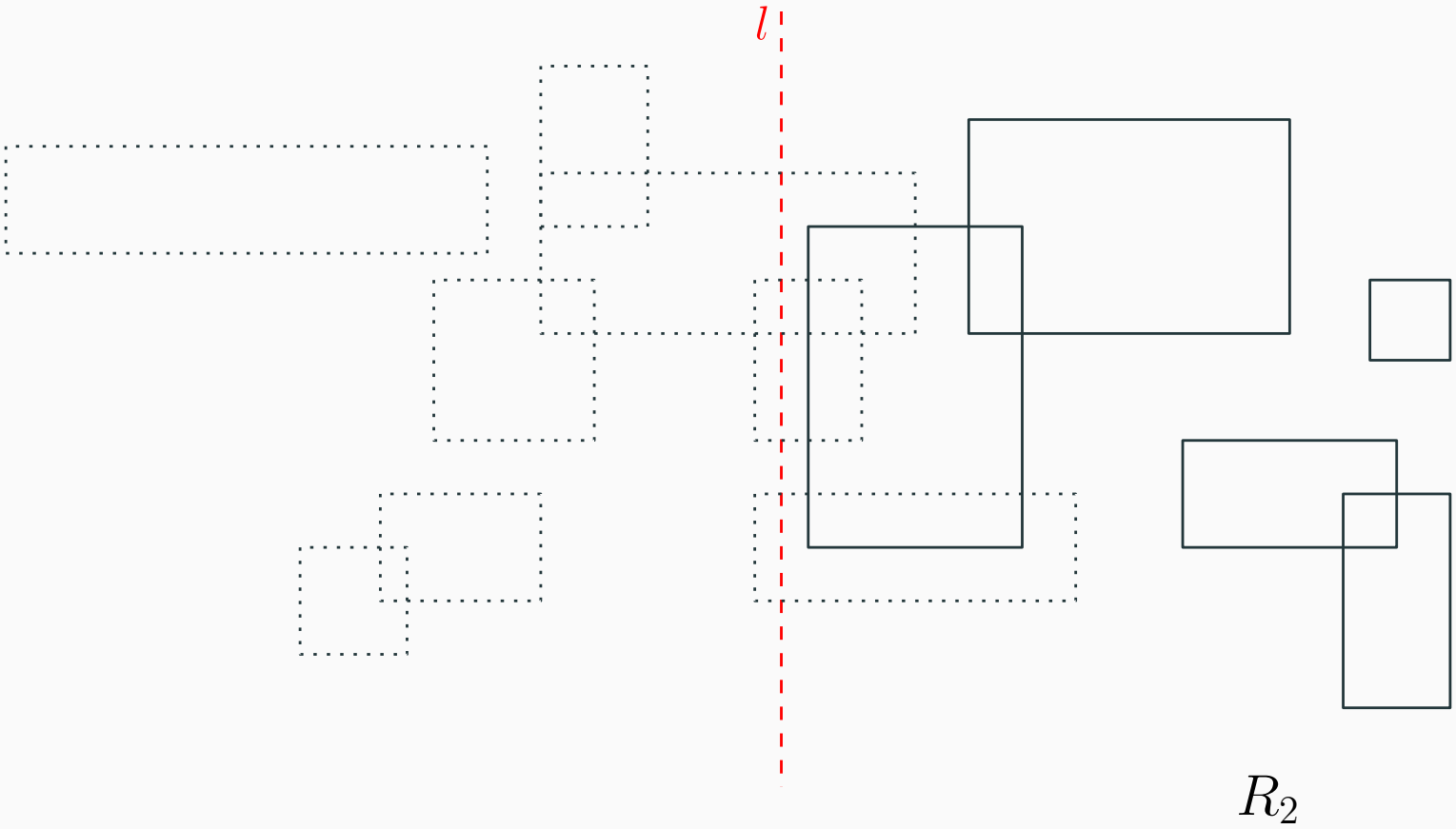
Beispiel



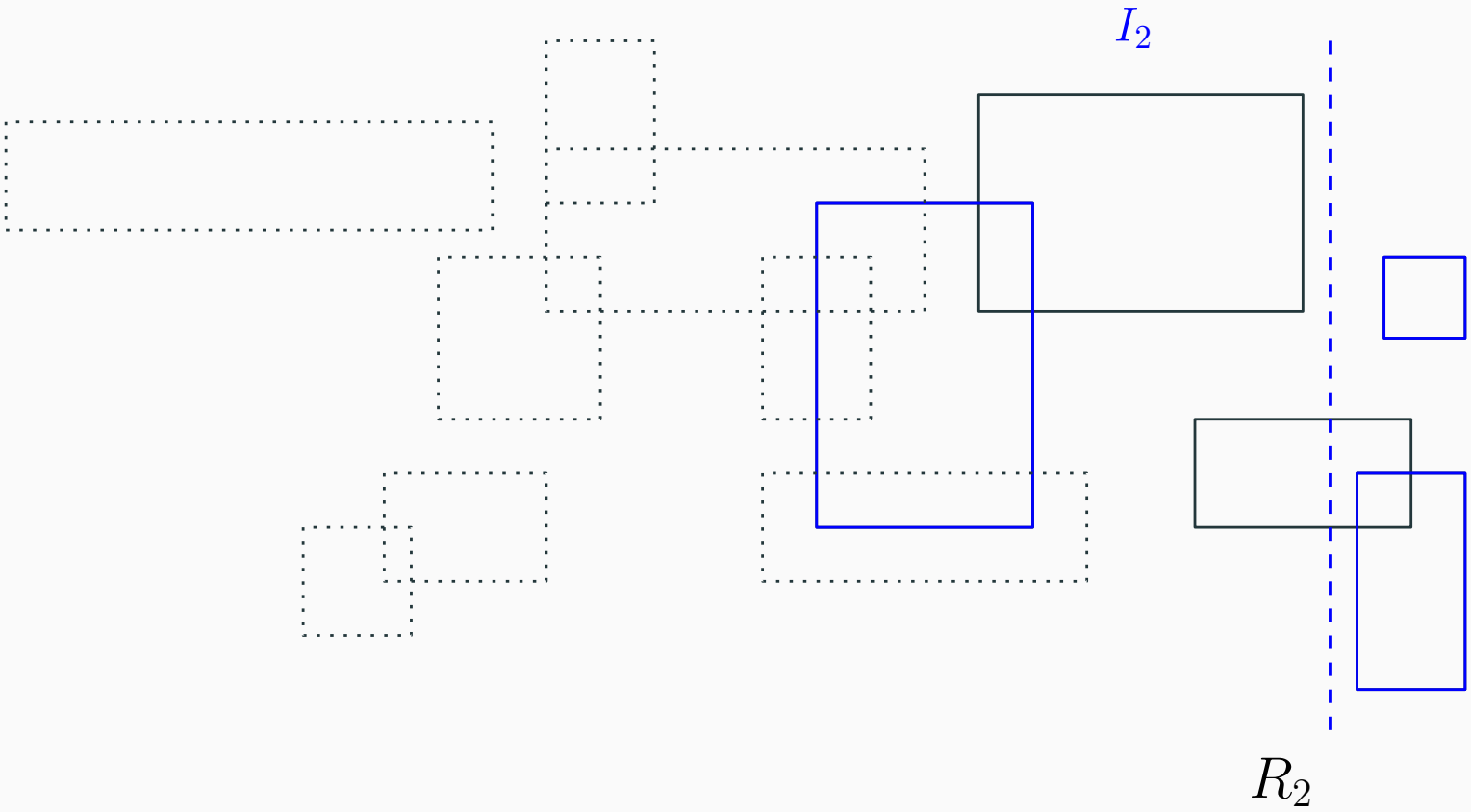
Beispiel



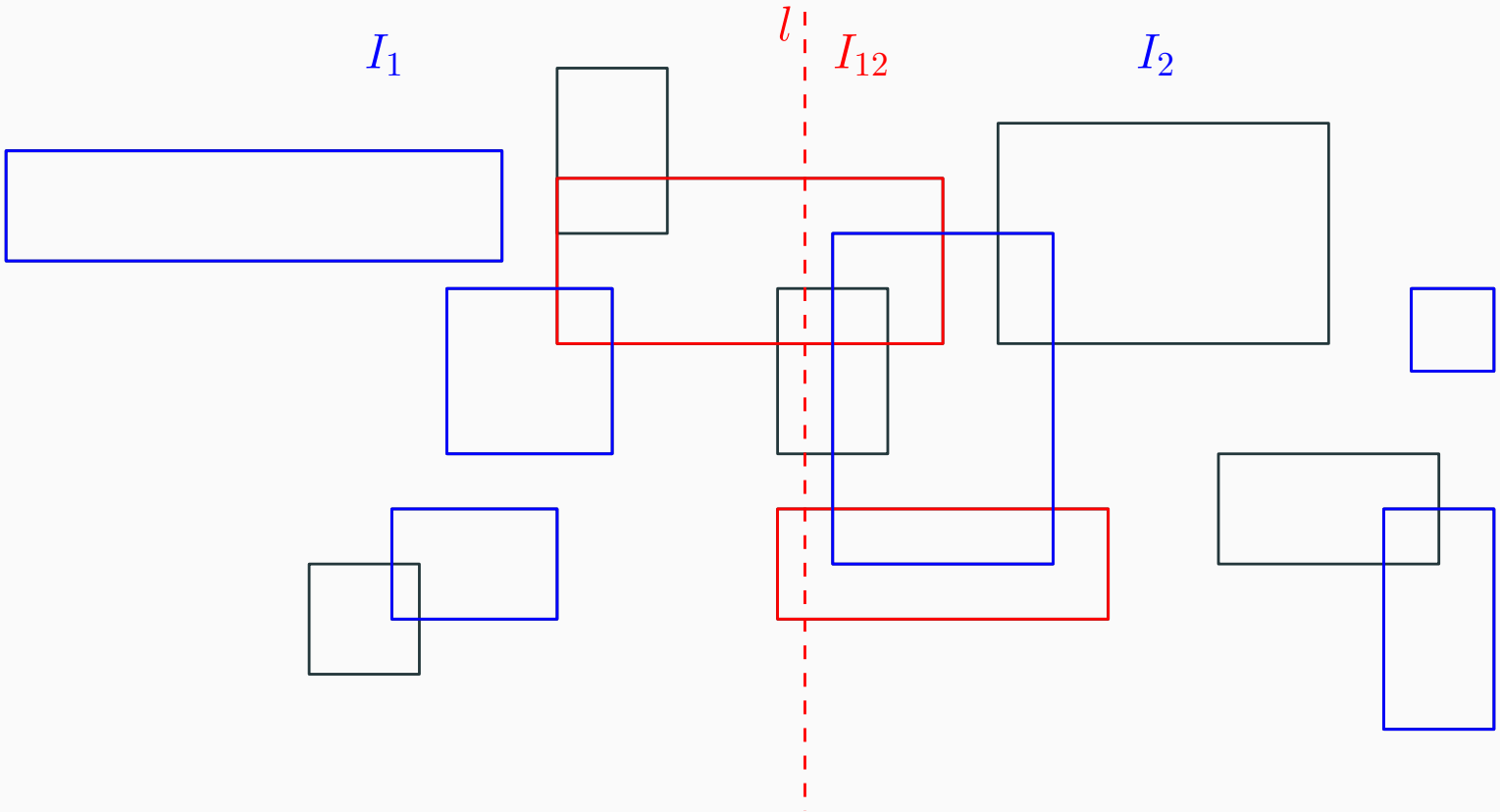
Beispiel



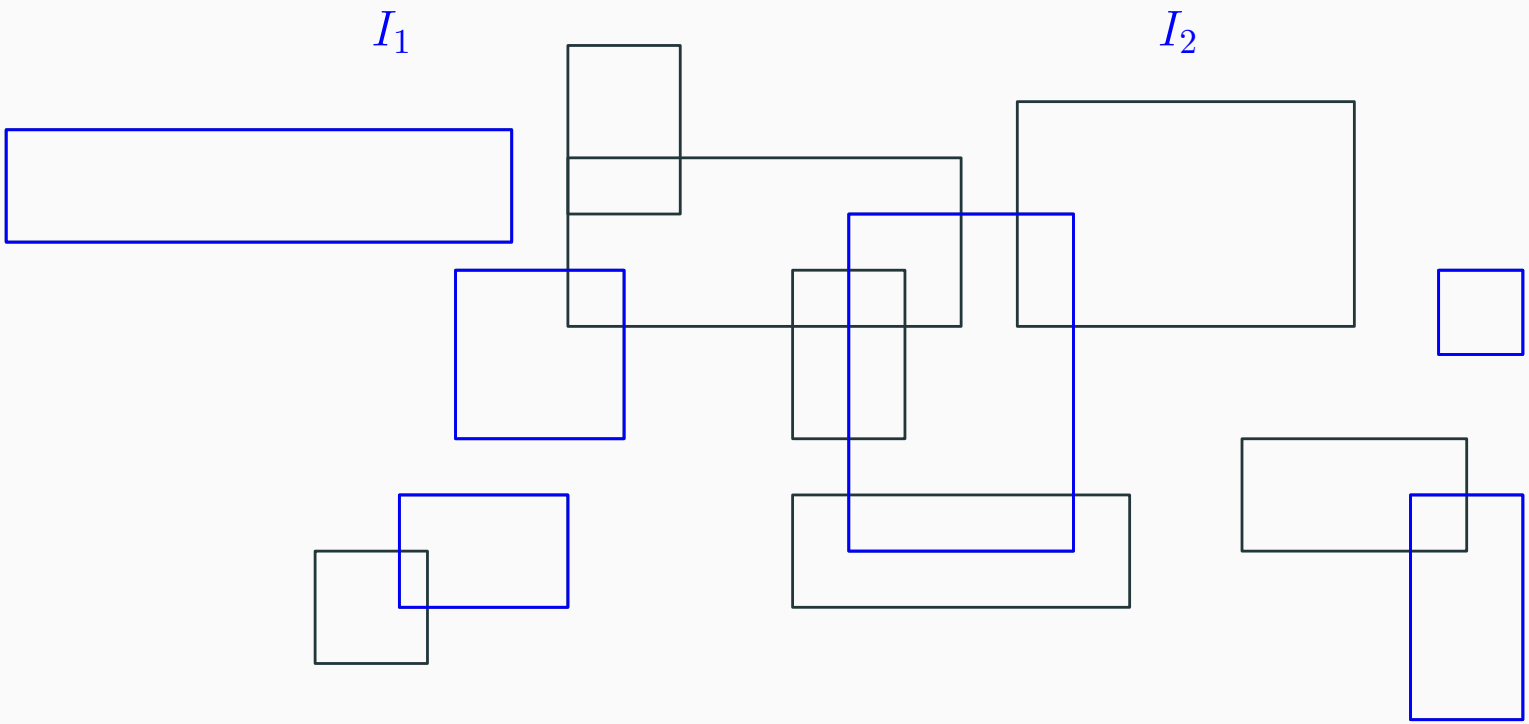
Beispiel



Beispiel



Beispiel



Laufzeitanalyse

1. Vorberechnung: Sortieren in $\mathcal{O}(n \log n)$.

Laufzeitanalyse

1. Vorberechnung: Sortieren in $\mathcal{O}(n \log n)$.
2. Partitionierung in $\mathcal{O}(n)$ auf jeder Rekursionsstufe.

Laufzeitanalyse

1. Vorberechnung: Sortieren in $\mathcal{O}(n \log n)$.
2. Partitionierung in $\mathcal{O}(n)$ auf jeder Rekursionsstufe.
   Rekursionstiefe in $\mathcal{O}(\log n)$ (da $|R_1|, |R_2| \leq \frac{n}{2}$).

Laufzeitanalyse

1. Vorberechnung: Sortieren in $\mathcal{O}(n \log n)$.
2. Partitionierung in $\mathcal{O}(n)$ auf jeder Rekursionsstufe. Rekursionstiefe in $\mathcal{O}(\log n)$ (da $|R_1|, |R_2| \leq \frac{n}{2}$). Insgesamt $\mathcal{O}(n \log n)$.

Laufzeitanalyse

1. Vorberechnung: Sortieren in $\mathcal{O}(n \log n)$.
2. Partitionierung in $\mathcal{O}(n)$ auf jeder Rekursionsstufe. Rekursionstiefe in $\mathcal{O}(\log n)$ (da $|R_1|, |R_2| \leq \frac{n}{2}$). Insgesamt $\mathcal{O}(n \log n)$.
3. ActivitySelector für jedes Rechteck höchstens einmal. Insgesamt $\mathcal{O}(n \log n)$.

Gesamtlaufzeit: $\mathcal{O}(n \log n)$

Gütegarantie (1)

Notation:

$I^*, I_1^*, I_2^*, I_{12}^*$: *maximum* independent sets von R, R_1, R_2, R_{12}.

I, I_1, I_2, I_{12}: vom Algorithmus berechnete independent sets von R, R_1, R_2, R_{12}.

Gütegarantie (1)

Notation:
$I^*, I_1^*, I_2^*, I_{12}^*$: *maximum* independent sets von R, R_1, R_2, R_{12}.
I, I_1, I_2, I_{12}: vom Algorithmus berechnete independent sets von R, R_1, R_2, R_{12}.

Zu zeigen: Der Algorithmus liefert eine $\frac{1}{\log n}$-Approximation, d.h. $|I| \geq \frac{|I^*|}{\log n}$.

Gütegarantie (1)

Notation:
$I^*, I_1^*, I_2^*, I_{12}^*$: *maximum* independent sets von R, R_1, R_2, R_{12}.
I, I_1, I_2, I_{12}: vom Algorithmus berechnete independent sets von R, R_1, R_2, R_{12}.

Zu zeigen: Der Algorithmus liefert eine $\frac{1}{\log n}$-Approximation, d.h. $|I| \geq \frac{|I^*|}{\log n}$.

Beweis: (induktiv)
Induktionsanfang: $n \leq 2$
Es gilt: $|I| = |I^*| \geq \frac{|I^*|}{\log n}$. ✓

Gütegarantie (2)

IV: $|I| \geq \frac{|I^*|}{\log n'}$ für $n' < n$.

IS: Zeige $\max(|I_{12}|, |I_1| + |I_2|) \geq \frac{|I^*|}{\log n}$.

Gütegarantie (2)

IV: $|I| \geq \frac{|I^*|}{\log n'}$ für $n' < n$.

IS: Zeige $\max(|I_{12}|, |I_1| + |I_2|) \geq \frac{|I^*|}{\log n}$.

Beobachtungen:

(i) $|I_{12}| = |I^*_{12}| \geq |I^* \cap R_{12}|$

Gütegarantie (2)

IV: $|I| \geq \frac{|I^*|}{\log n'}$ für $n' < n$.

IS: Zeige $\max(|I_{12}|, |I_1| + |I_2|) \geq \frac{|I^*|}{\log n}$.

Beobachtungen:

(i) $|I_{12}| = |I^*_{12}| \geq |I^* \cap R_{12}|$

(ii) $|I_1| \overset{IV}{\geq} \frac{|I^*_1|}{\log |R_1|} \overset{|R_1| \leq n/2}{\geq} \frac{|I^*_1|}{\log n - 1} \geq \frac{|I^* \cap R_1|}{\log n - 1}$ (I_2 analog)

Gütegarantie (2)

IV: $|I| \geq \frac{|I^*|}{\log n'}$ für $n' < n$.

IS: Zeige $\max(|I_{12}|, |I_1| + |I_2|) \geq \frac{|I^*|}{\log n}$.

Beobachtungen:

(i) $|I_{12}| = |I^*_{12}| \geq |I^* \cap R_{12}|$

(ii) $|I_1| \overset{IV}{\geq} \frac{|I^*_1|}{\log |R_1|} \overset{|R_1| \leq n/2}{\geq} \frac{|I^*_1|}{\log n - 1} \geq \frac{|I^* \cap R_1|}{\log n - 1}$ (I_2 analog)

Fall 1: $|I^* \cap R_{12}| \geq \frac{|I^*|}{\log n}$.

Gütegarantie (2)

IV: $|I| \geq \frac{|I^*|}{\log n'}$ für $n' < n$.

IS: Zeige $\max(|I_{12}|, |I_1| + |I_2|) \geq \frac{|I^*|}{\log n}$.

Beobachtungen:

(i) $|I_{12}| = |I^*_{12}| \geq |I^* \cap R_{12}|$

(ii) $|I_1| \overset{IV}{\geq} \frac{|I^*_1|}{\log |R_1|} \overset{|R_1| \leq n/2}{\geq} \frac{|I^*_1|}{\log n - 1} \geq \frac{|I^* \cap R_1|}{\log n - 1}$ (I_2 analog)

Fall 1: $|I^* \cap R_{12}| \geq \frac{|I^*|}{\log n}$. Dann folgt mit (i): $|I_{12}| \geq \frac{|I^*|}{\log n}$. ✓

Gütegarantie (2)

IV: $|I| \geq \frac{|I^*|}{\log n'}$ für $n' < n$.

IS: Zeige $\max(|I_{12}|, |I_1| + |I_2|) \geq \frac{|I^*|}{\log n}$.

Beobachtungen:

(i) $|I_{12}| = |I^*_{12}| \geq |I^* \cap R_{12}|$

(ii) $|I_1| \overset{IV}{\geq} \frac{|I^*_1|}{\log |R_1|} \overset{|R_1| \leq n/2}{\geq} \frac{|I^*_1|}{\log n - 1} \geq \frac{|I^* \cap R_1|}{\log n - 1}$ (I_2 analog)

Fall 1: $|I^* \cap R_{12}| \geq \frac{|I^*|}{\log n}$. Dann folgt mit (i): $|I_{12}| \geq \frac{|I^*|}{\log n}$. ✓

Fall 2: $|I^* \cap R_{12}| < \frac{|I^*|}{\log n}$. Es gilt:

Gütegarantie (2)

IV: $|I| \geq \frac{|I^*|}{\log n'}$ für $n' < n$.

IS: Zeige $\max(|I_{12}|, |I_1| + |I_2|) \geq \frac{|I^*|}{\log n}$.

Beobachtungen:

(i) $|I_{12}| = |I_{12}^*| \geq |I^* \cap R_{12}|$

(ii) $|I_1| \overset{IV}{\geq} \frac{|I_1^*|}{\log |R_1|} \overset{|R_1| \leq n/2}{\geq} \frac{|I_1^*|}{\log n - 1} \geq \frac{|I^* \cap R_1|}{\log n - 1}$ (I_2 analog)

Fall 1: $|I^* \cap R_{12}| \geq \frac{|I^*|}{\log n}$. Dann folgt mit (i): $|I_{12}| \geq \frac{|I^*|}{\log n}$. ✓

Fall 2: $|I^* \cap R_{12}| < \frac{|I^*|}{\log n}$. Es gilt:

$$|I_1| + |I_2| \overset{(ii)}{\geq} \frac{|I^* \cap R_1| + |I^* \cap R_2|}{\log n - 1} = \frac{|I^*| - |I^* \cap R_{12}|}{\log n - 1} \overset{\text{Fall 2}}{\geq} \frac{|I^*| - \frac{|I^*|}{\log n}}{\log n - 1}$$

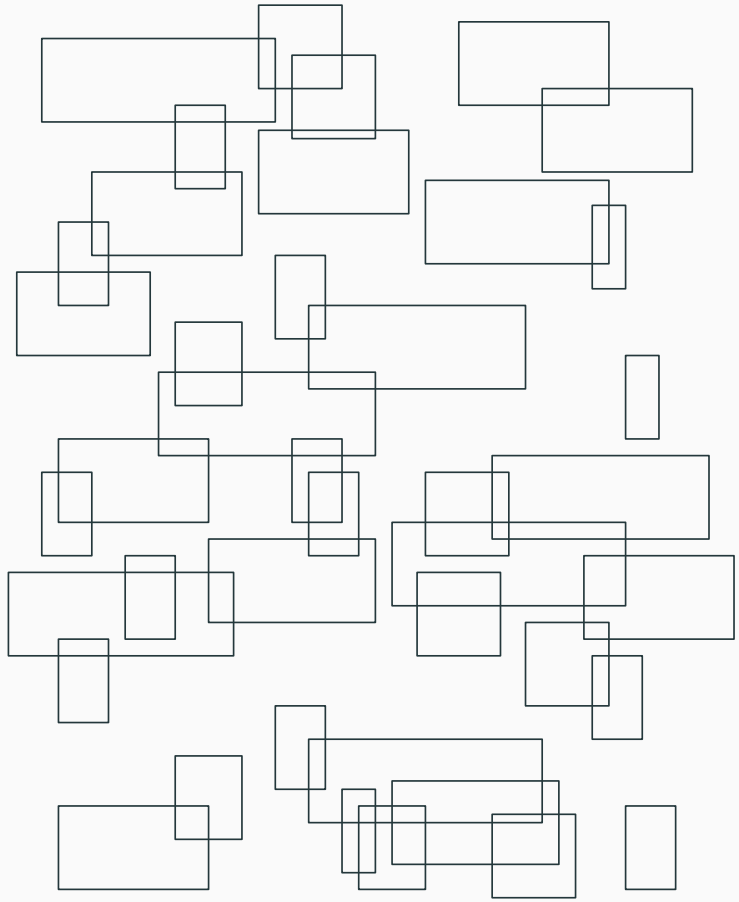
$$= |I^*| \cdot \frac{1 - \frac{1}{\log n}}{\log n - 1} = |I^*| \cdot \frac{\log n - 1}{\log n \cdot (\log n - 1)} = \frac{|I^*|}{\log n} \checkmark$$

□

Rechtecke gleicher Höhe: $1/(1+1/k)$-Approximation

Spezialfall $k = 1$

Jetzt: Alle Rechtecke haben Höhe 1.

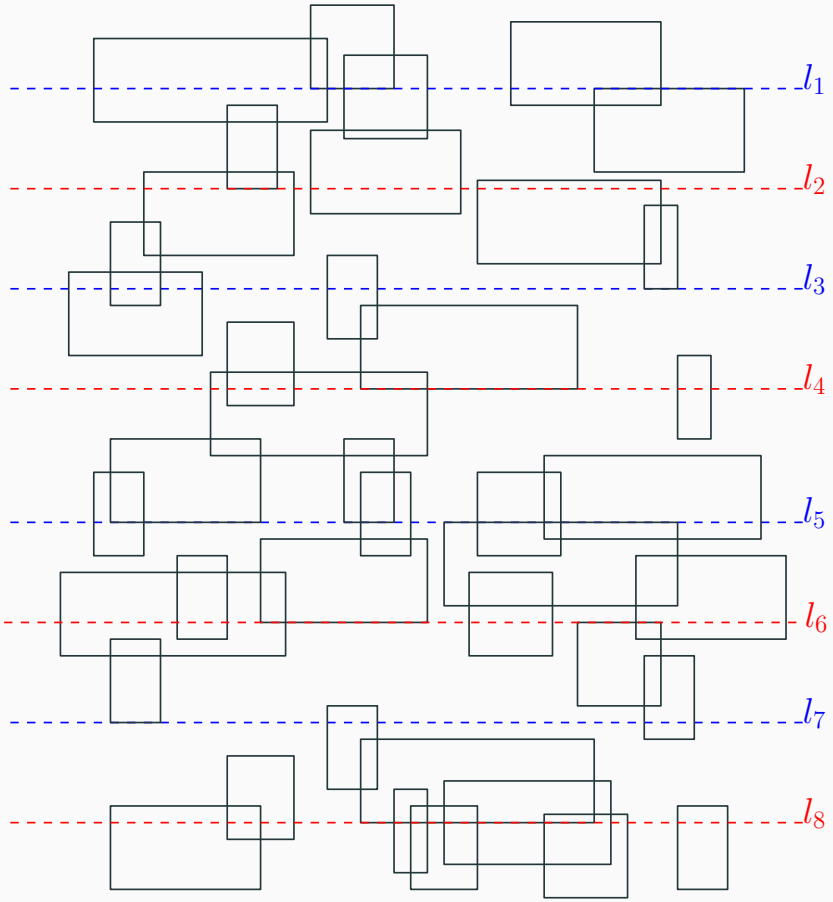


Spezialfall $k = 1$

Jetzt: Alle Rechtecke haben Höhe 1.

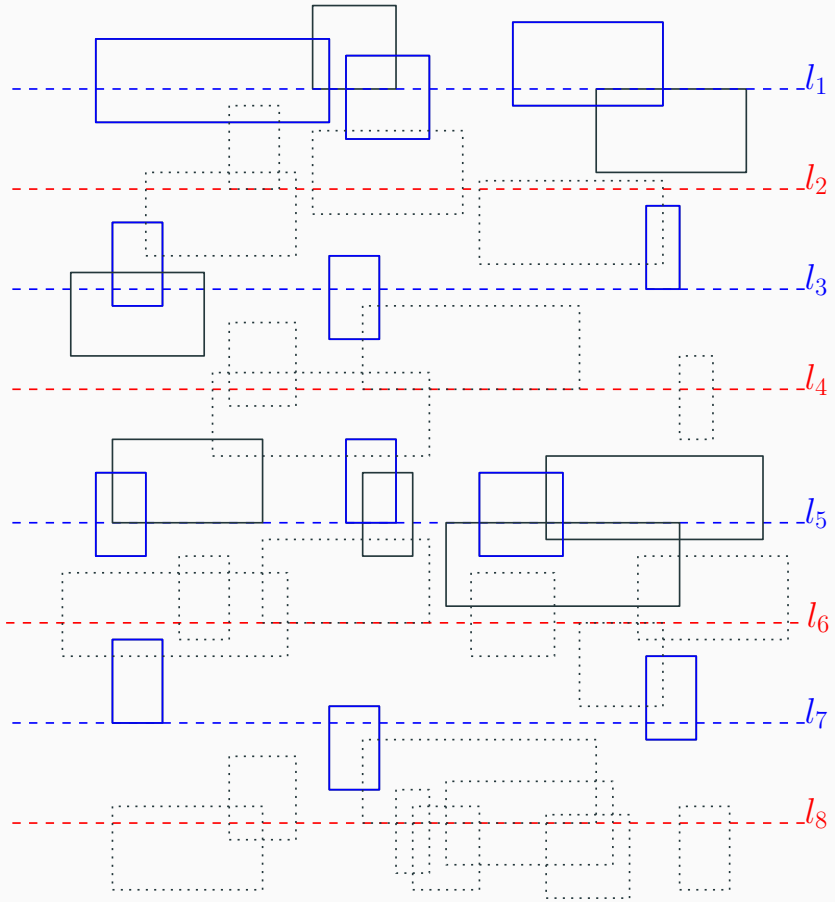
Idee: Zeichne Linien $\ell_1, \ldots \ell_m$, sodass jedes Rechteck von genau einer Linie geschnitten wird (greedy)

- Induziert Partitionierung: $R = R_1 \cup \cdots \cup R_m$
- Rechtecke in R_i schneiden nur Rechtecke in R_{i-1}, R_i und R_{i+1}.



Spezialfall $k = 1$

- Berechne $l_1, \ldots l_m$ mit ActivitySelector ($\mathcal{O}(n \log n)$)
- $l_{odd} := l_1 \cup l_3 \cup l_5 \ldots$ und $l_{even} := l_2 \cup l_4 \cup l_6 \ldots$ sind maximum independent sets.
- $|l_{odd}| \geq |l^* \cap (R_1 \cap R_3 \ldots)| \geq \frac{|l^*|}{2}$ oder $|l_{even}| \geq |l^* \cap (R_2 \cap R_4 \ldots)| \geq \frac{|l^*|}{2}$



Erweiterung auf größere k: Definitionen

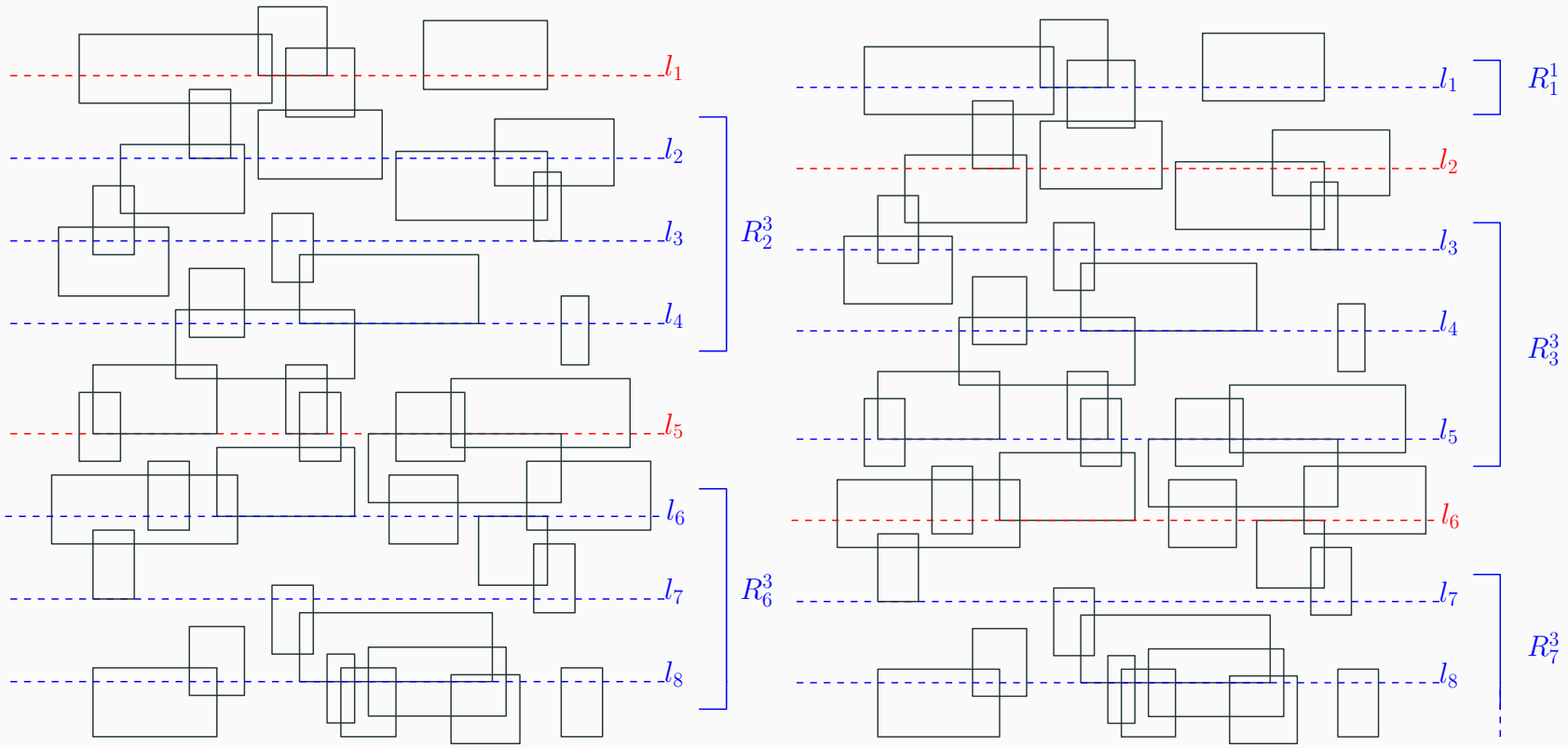


Abbildung 1: Die Gruppen G_1 und G_2 für $k = 3$.

- Blöcke aus k Linien: $R_i^k := R_i \cup \cdots \cup R_{i+k-1}$
- Gruppen: $G_j := R_1^{j-1} \cup \bigcup_{i \geq 0} R^k_{i(k+1)+j+1} = R \setminus \bigcup_{i \geq 0} R_{i(k+1)+j}$

  $(1 \leq j \leq k+1)$

Erweiterung auf größere k: Verschiebetechnik

Annahme: Wir können ein maximum independet set $I^*(R_x^k)$ auf einem Block berechnen.

Erweiterung auf größere k: Verschiebetechnik

Annahme: Wir können ein maximum independet set $I^*(R_x^k)$ auf einem Block berechnen.

- Dann ist $I^*(G_j) = I^*(R_1^{j-1}) \cup \bigcup_{i \geq 0} I^*(R^k_{i(k+1)+j+1})$, da sich die Blöcke nicht überschneiden.

Erweiterung auf größere k: Verschiebetechnik

Annahme: Wir können ein maximum independet set $I^*(R_x^k)$ auf einem Block berechnen.

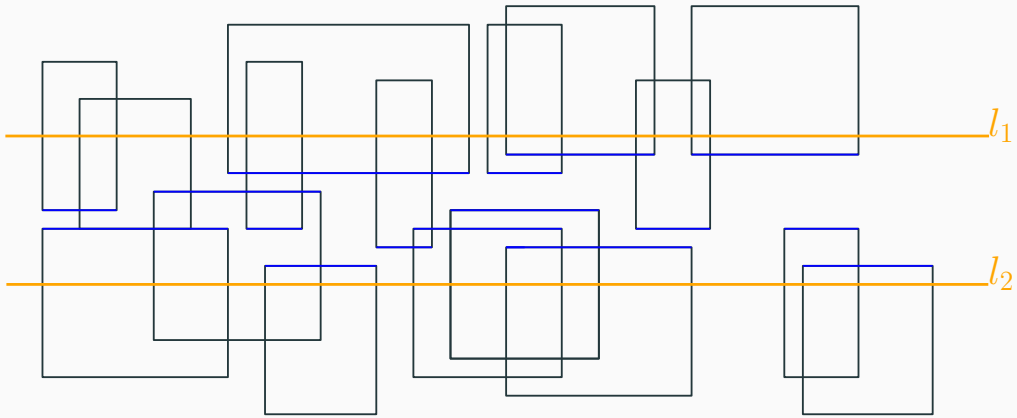
- Dann ist $I^*(G_j) = I^*(R_1^{j-1}) \cup \bigcup_{i \geq 0} I^*(R^k_{i(k+1)+j+1})$, da sich die Blöcke nicht überschneiden.
- Verschiebe Blöcke (alle Gruppen durchprobieren)
  $I := \max_{j \in [1,k+1]} I^*(G_j)$

MaxIndepSet auf Block der Höhe k

Betrachte zunächst nur Blöcke der Höhe $k = 2$

MaxIndepSet auf Block der Höhe k

Betrachte zunächst nur Blöcke der Höhe $k = 2$



Seien

- X die aufsteigend sortierten x-Koordinaten aller vertikalen Kanten
- Y die aufsteigend sortierten y-Koordinaten der Unterkanten der Rechtecke in R_1 und der Oberkanten der Rechtecke in R_2

Dann definiert $\tau = (p, q, t)$ ($x_p, x_q \in X, y_t \in Y$) eine Polylinie und R_τ die Rechtecke links davon:

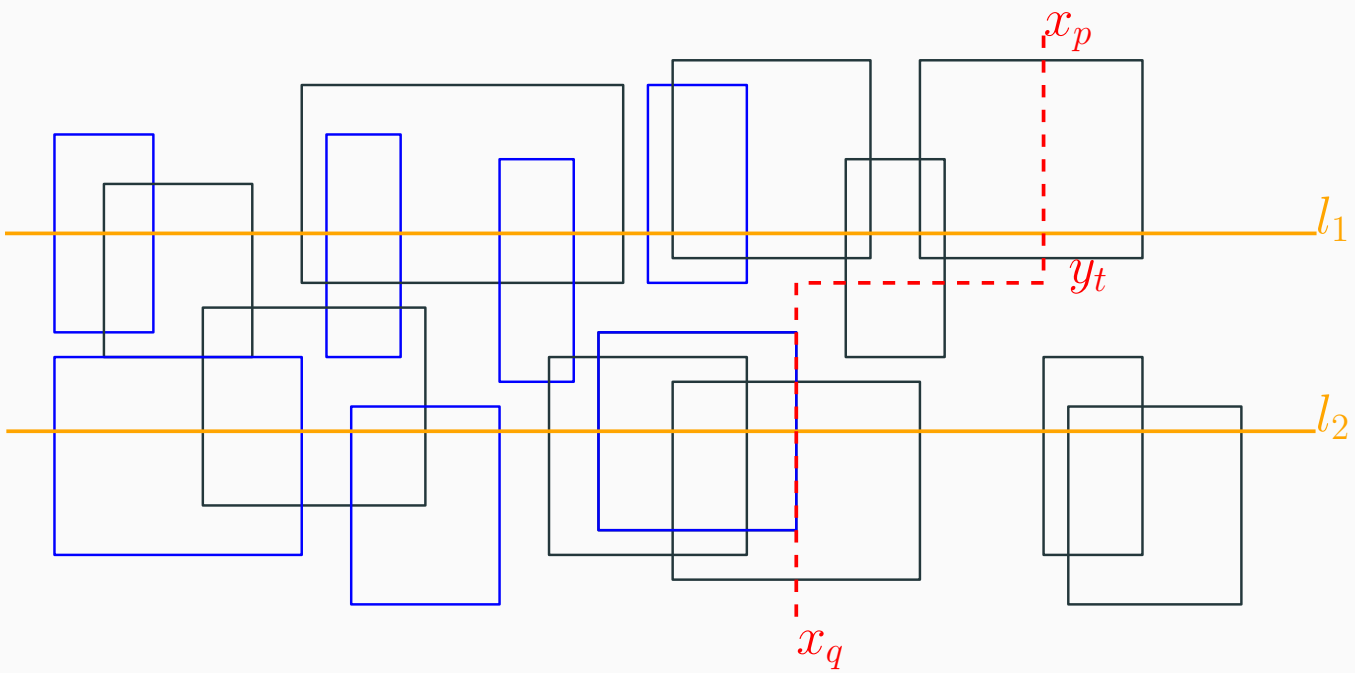


Abbildung 2: Polygonale Linie. $A[p, q, t] = 7$

Dynamische Programmierung (DP)

Dann definiert $\tau = (p, q, t)$ ($x_p, x_q \in X, y_t \in Y$) eine Polylinie und R_τ die Rechtecke links davon:

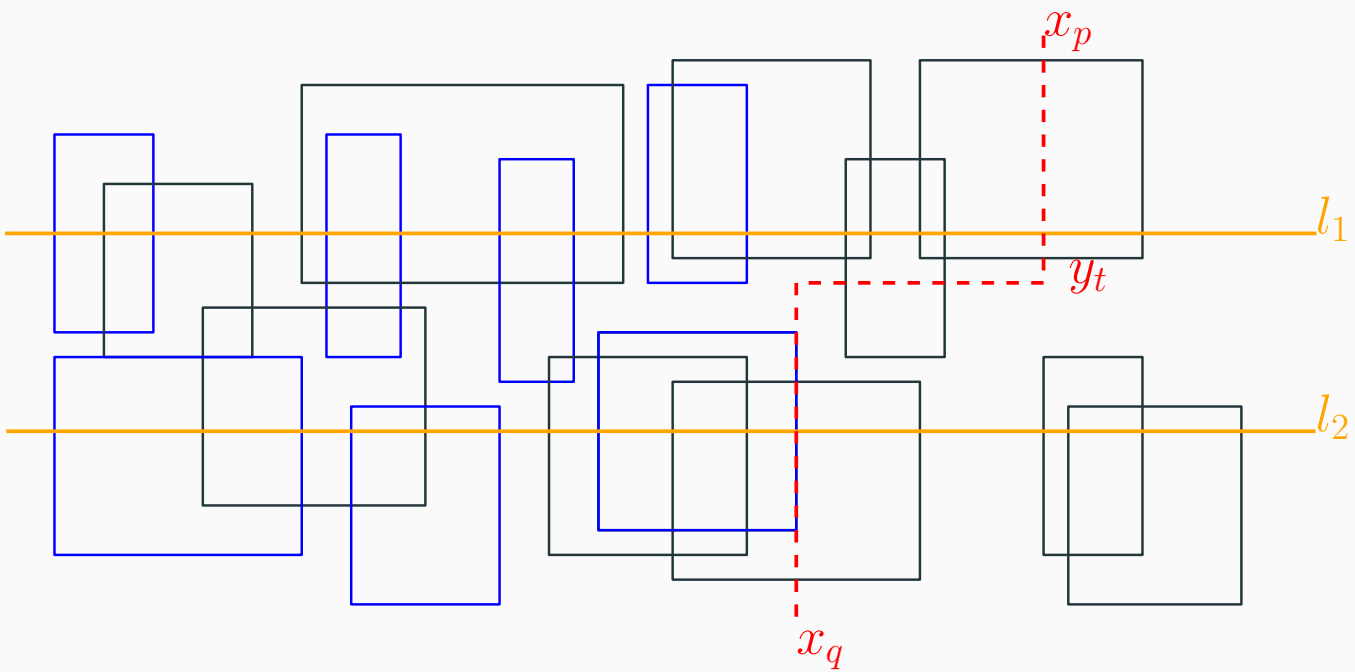


Abbildung 2: Polygonale Linie. $A[p, q, t] = 7$

Dreidimenisonale DP-Tabelle A, wobei $A_{p,q,t} = A_\tau = |I^*(R_\tau)|$.

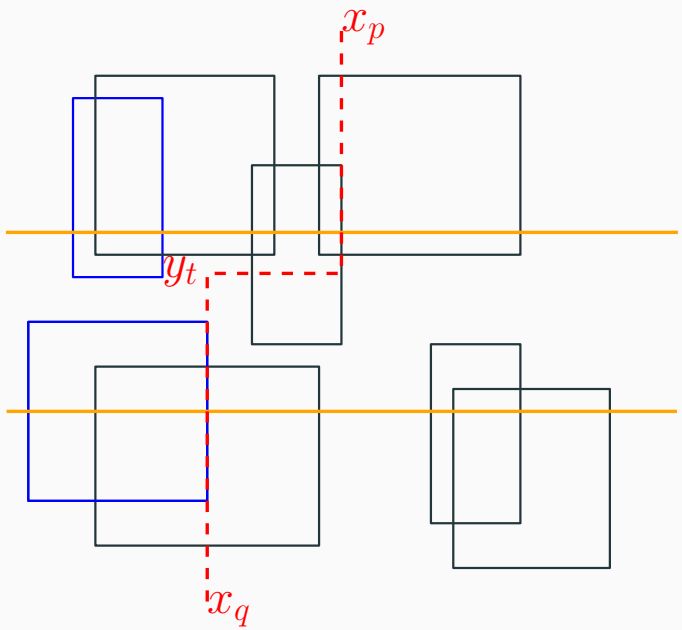
DP: Fall 1

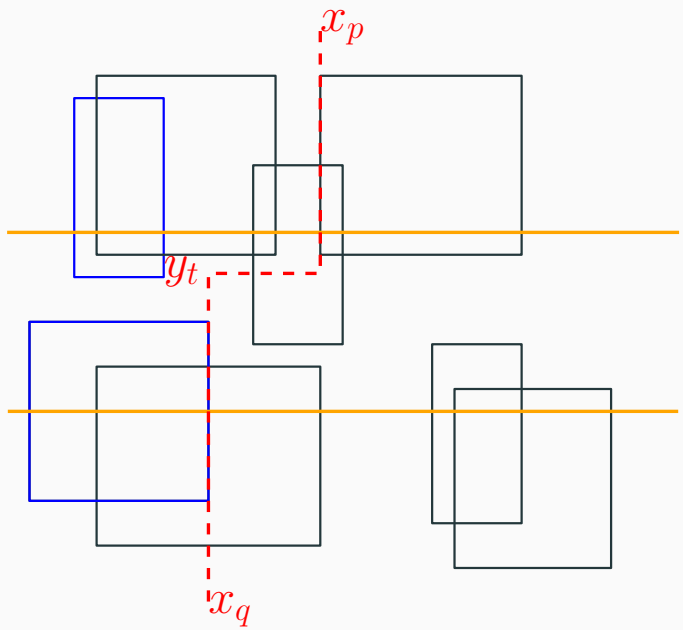
Sei O.b.d.A. $p > q$.

DP: Fall 1

Sei O.b.d.A. $p > q$.

Fall 1: x_p ist **keine rechte Kante** eines Rechteckes $r \in R_\tau \cap R_1$

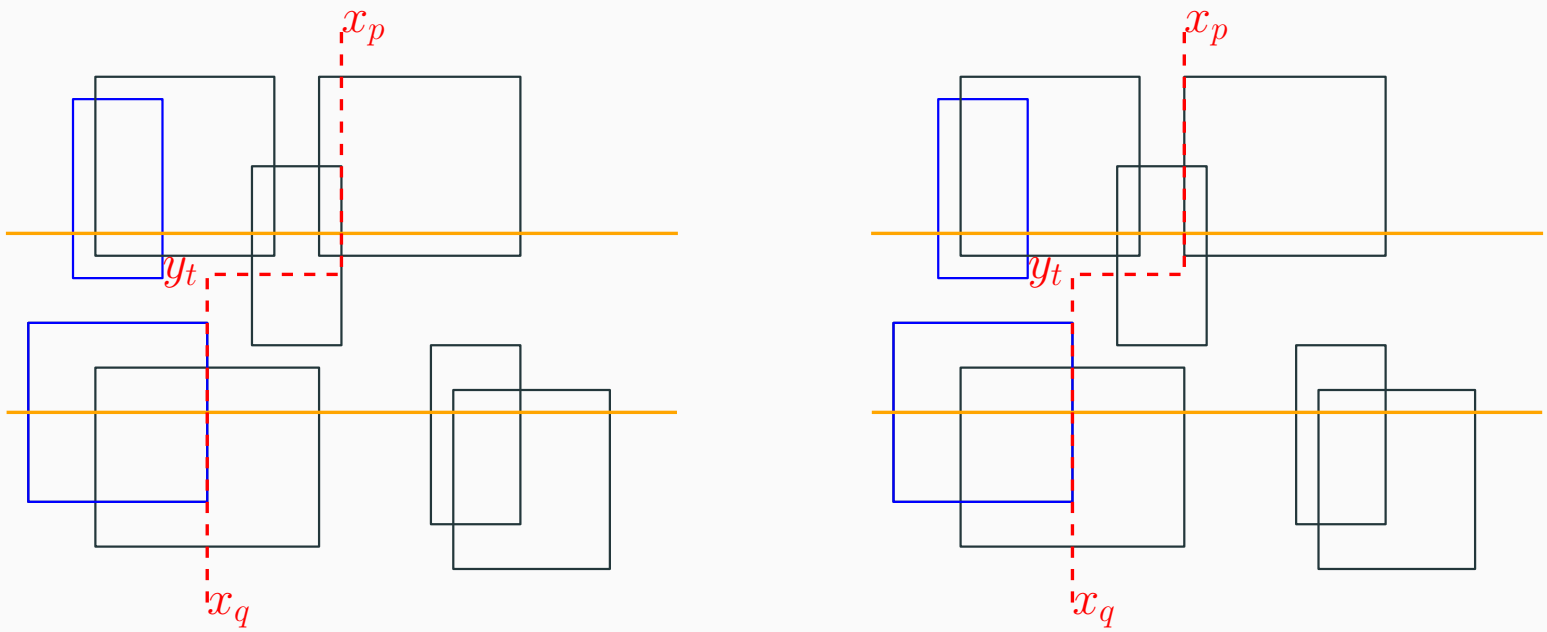




DP: Fall 1

Sei O.b.d.A. $p > q$.

Fall 1: x_p ist **keine rechte Kante** eines Rechteckes $r \in R_\tau \cap R_1$

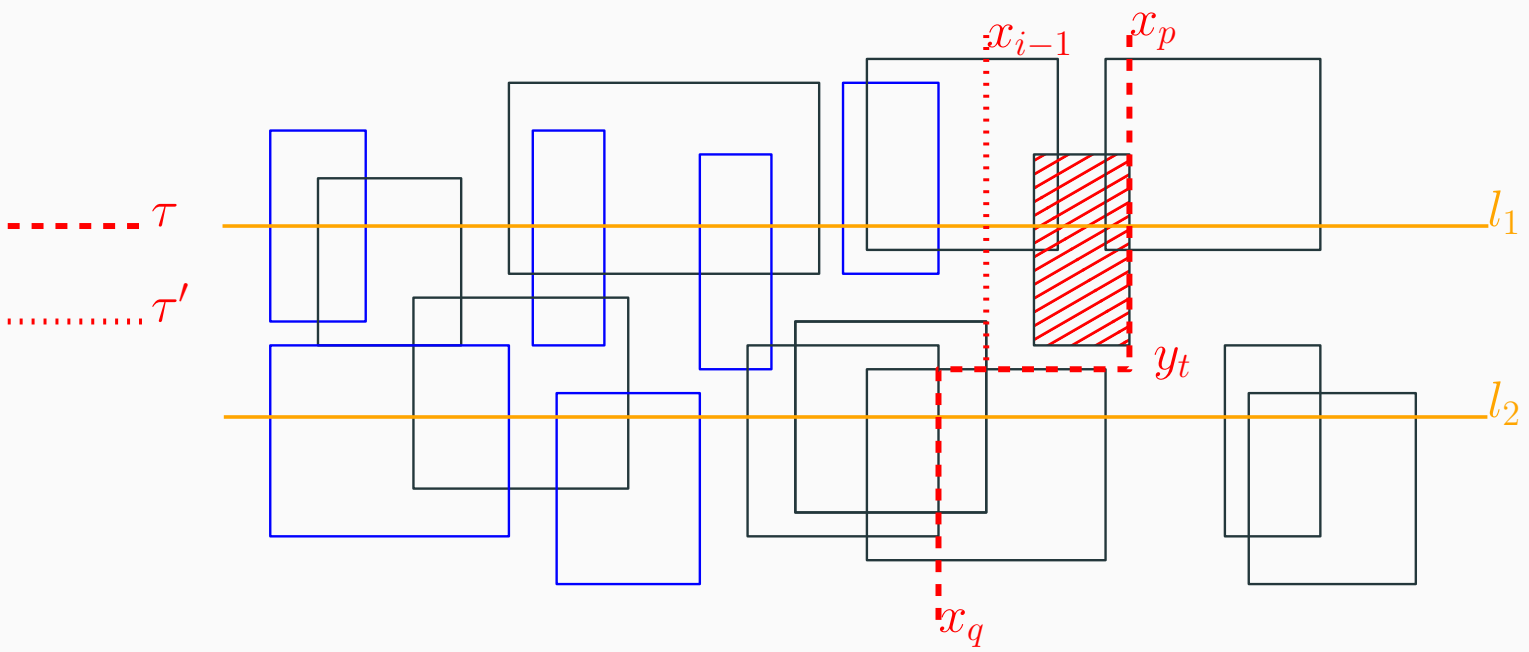


$\Rightarrow r \notin R_\tau$

$\Rightarrow A[p, q, t] = A[p - 1, q, t]$

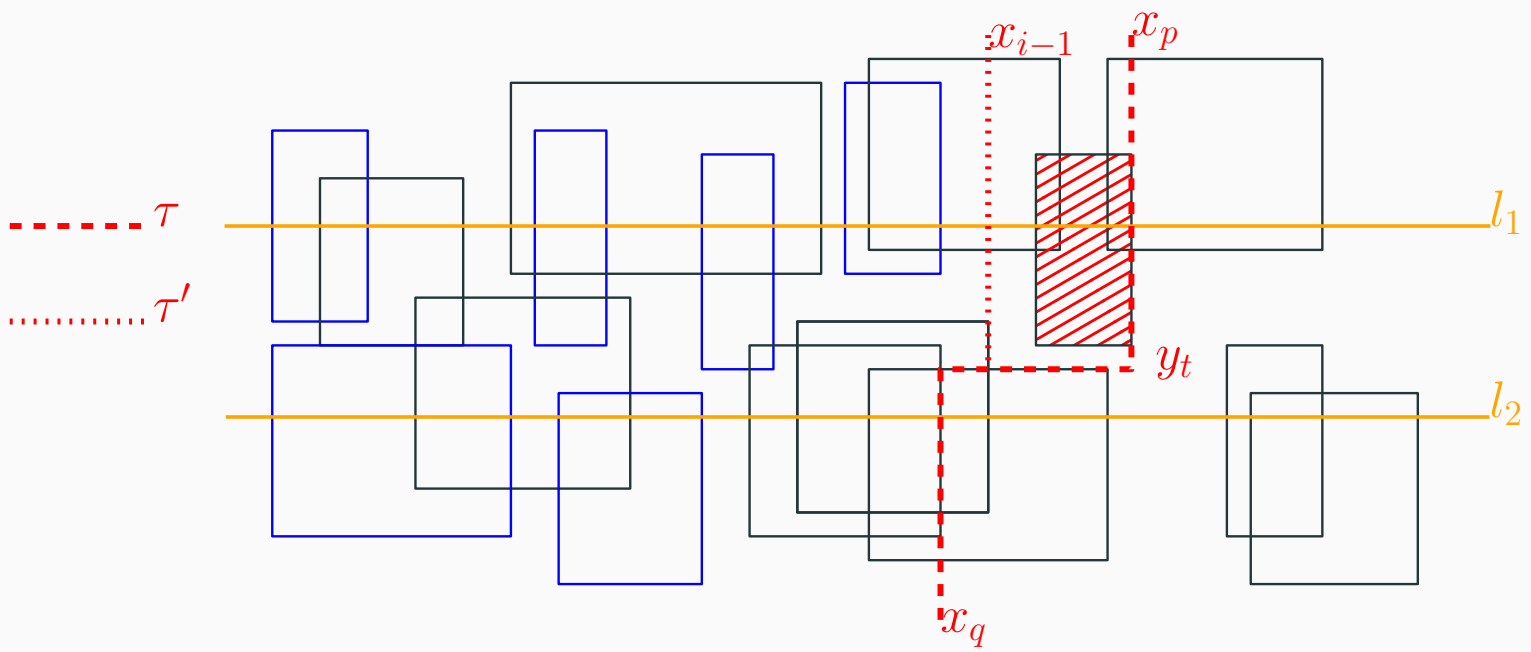
DP: Fall 2

Fall 2: x_p ist rechte Kante eines Rechteckes $r \in R_\tau \cap R_1$ mit linker Kante x_i und $i > q$:



DP: Fall 2

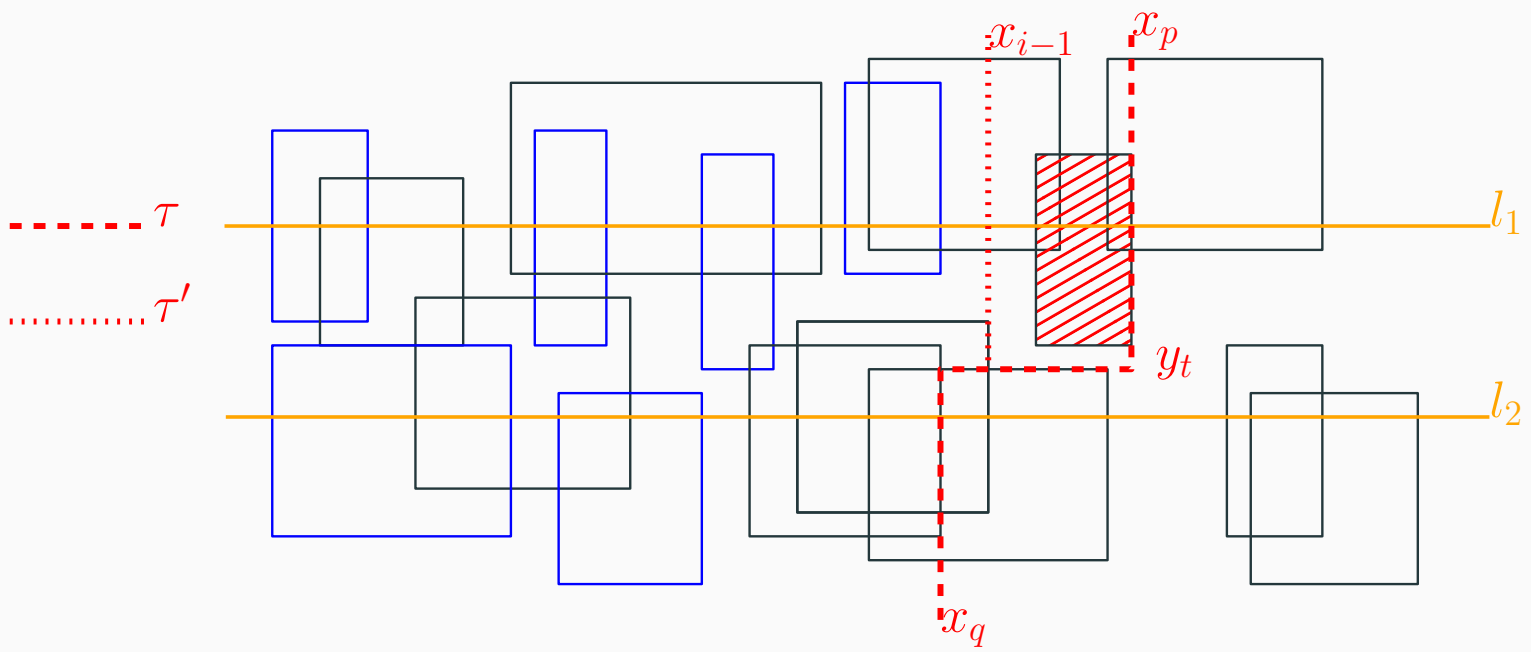
Fall 2: x_p ist rechte Kante eines Rechteckes $r \in R_\tau \cap R_1$ mit linker Kante x_i und $i > q$:



Rechtecke, die r nicht schneiden, liegen links von $\tau' = (i-1, q, j)$

DP: Fall 2

Fall 2: x_p ist rechte Kante eines Rechteckes $r \in R_\tau \cap R_1$ mit linker Kante x_i und $i > q$:

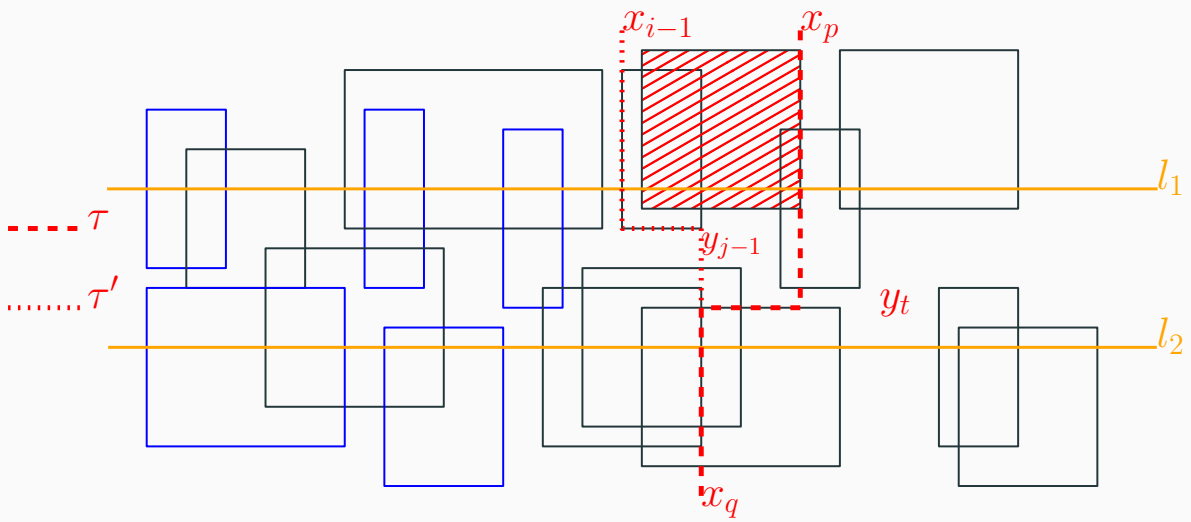


Rechtecke, die r nicht schneiden, liegen links von $\tau' = (i-1, q, j)$

$\Rightarrow A[p,q,t] = \max(A[p-1,q,t], A[i-1,q,j]+1)$

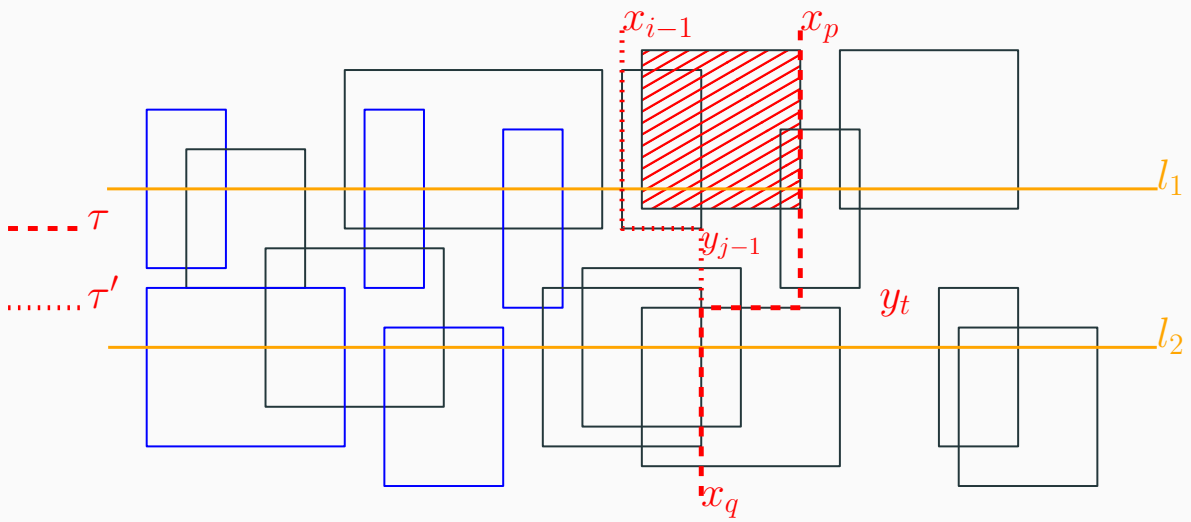
DP: Fall 3

Fall 3: x_p ist rechte Kante eines Rechteckes $r \in R_\tau \cap R_1$ mit linker Kante x_i, Unterkante y_j und $i \leq q$:



DP: Fall 3

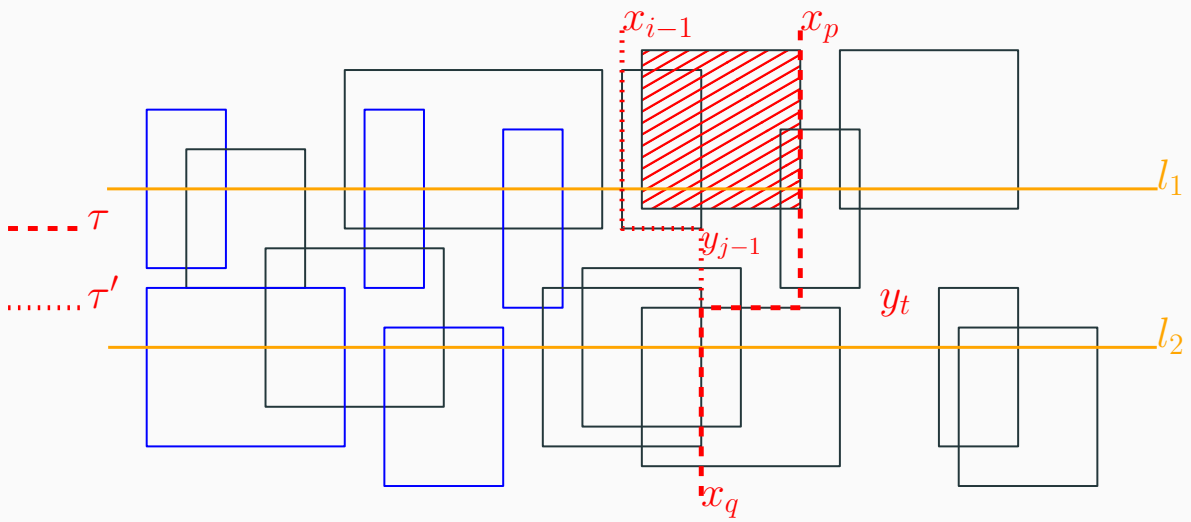
Fall 3: x_p ist rechte Kante eines Rechteckes $r \in R_\tau \cap R_1$ mit linker Kante x_i, Unterkante y_j und $i \leq q$:



Rechtecke, die r nicht schneiden, liegen links von $\tau' = (i-1, q, j-1)$

DP: Fall 3

Fall 3: x_p ist rechte Kante eines Rechteckes $r \in R_\tau \cap R_1$ mit linker Kante x_i, Unterkante y_j und $i \leq q$:



Rechtecke, die r nicht schneiden, liegen links von $\tau' = (i-1, q, j-1)$

$\Rightarrow A[p,q,t] = \max(A[p-1,q,t], A[i-1,q,j-1]+1)$

Laufzeitanalyse

- Sortieren und in Linien unterteilen $\mathcal{O}(n \log n)$

Laufzeitanalyse

- Sortieren und in Linien unterteilen $\mathcal{O}(n \log n)$
- DP mit $k = 2$: $\sum_{i=1}^{m-1} \mathcal{O}((|R_i| + |R_{i+1}|)^3) = \mathcal{O}(n^3)$

Laufzeitanalyse

- Sortieren und in Linien unterteilen $\mathcal{O}(n \log n)$
- DP mit $k = 2$: $\sum_{i=1}^{m-1} \mathcal{O}((|R_i| + |R_{i+1}|)^3) = \mathcal{O}(n^3)$
- Analog für $k > 2$:
  - Polylinie mit k Stufen (k vertikale und $k - 1$ horizontale Segmente)
  - $\Rightarrow$ Laufzeit: $\mathcal{O}(n^{2k-1})$

Laufzeitanalyse

- Sortieren und in Linien unterteilen $\mathcal{O}(n \log n)$
- DP mit $k = 2$: $\sum_{i=1}^{m-1} \mathcal{O}((|R_i| + |R_{i+1}|)^3) = \mathcal{O}(n^3)$
- Analog für $k > 2$:
  - Polylinie mit k Stufen (k vertikale und $k - 1$ horizontale Segmente)
  - $\Rightarrow$ Laufzeit: $\mathcal{O}(n^{2k-1})$

Insgesamt $\mathcal{O}(n \log n + n^{2k-1})$

Gütegarantie

Zu zeigen: $|I| \leq \frac{|I^*|}{1+1/k}$

Gütegarantie

Zu zeigen: $|I| \leq \frac{|I^*|}{1+1/k}$

- Differenz zur optimalen Lösung:
  $|I^*| - |I| \leq |I^* \cap (R \setminus G_j)| \leq \frac{|I^*|}{k+1}$

Gütegarantie

Zu zeigen: $|I| \leq \frac{|I^*|}{1+1/k}$

- Differenz zur optimalen Lösung:
  $|I^*| - |I| \leq |I^* \cap (R \setminus G_j)| \leq \frac{|I^*|}{k+1}$
  - *Schubfachprinzip:* Verteile $|I^*|$ viele Rechtecke auf $k+1$ „Fächer“ $R \setminus G_j$

Gütegarantie

Zu zeigen: $|I| \leq \frac{|I^*|}{1+1/k}$

- Differenz zur optimalen Lösung:
  $|I^*| - |I| \leq |I^* \cap (R \backslash G_j)| \leq \frac{|I^*|}{k+1}$
  - *Schubfachprinzip:* Verteile $|I^*|$ viele Rechtecke auf $k+1$ „Fächer“ $R \backslash G_j$
- Umstellen:
  $\Leftrightarrow (k+1) \cdot (|I^*| - |I|) \leq |I^*|$
  $\Leftrightarrow (k+1) \cdot |I^*| - |I^*| \leq (k+1) \cdot |I|$
  $\Leftrightarrow |I| \geq |I^*| - \frac{|I^*|}{k+1} = |I^*| \cdot (1 - \frac{1}{k+1}) = |I^*| \cdot \frac{k+1-1}{k+1} = |I^*| \cdot \frac{1}{1+1/k}$

Zusammenfassung

Independet Sets auf n achsenparalleln Rechtecken:

1. Divide-and-Conquer-Algorithmus
   - $1/\log_2(n)$-Approximation
   - Laufzeit $\mathcal{O}(n \log n)$

Zusammenfassung

Independet Sets auf n achsenparalleln Rechtecken:

1. Divide-and-Conquer-Algorithmus
   - $1/\log_2(n)$-Approximation
   - Laufzeit $\mathcal{O}(n \log n)$
2. Auf Rechtecken gleicher Größe (DP-Algorithmus)
   - $1/(1 + 1/k)$-Approximation
   - Laufzeit $\mathcal{O}(n \log n + n^{2k-1})$

Literatur

P.K. Agarwal, M. van Kreveld und S. Suri. “L abel placement by maximum independent set in rectangles”. In: *Computational Geometry 11* (1998), S. 209–218.

M. Formann und F. Wagner. “A packing problem with applications to lettering of maps”. In: *ACM Sympos. comput. Geom* (1991), S. 281–288.

L. Kucera u. a. “Exact algorithms for a geometric packing problem”. In: *Theoret. Aspects Comput. Sci., Lecture Notes in Computer Sience* 665 (1993), S. 317–322.