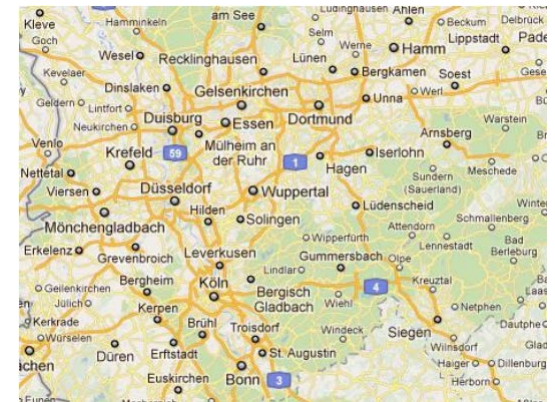
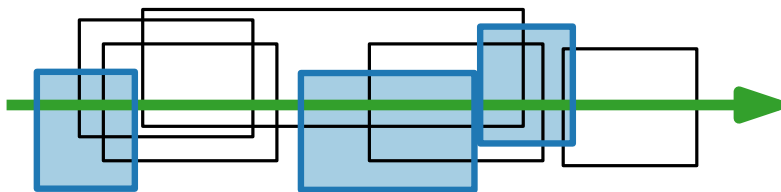


Algorithmen für geographische Informationssysteme

6. Vorlesung Kartenbeschriftung

Teil I: Das Beschriftungsproblem



Kartenbeschriftungen



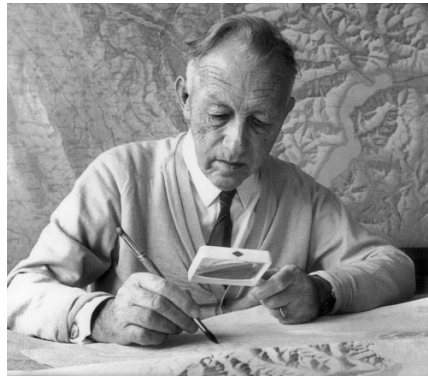
"Poor, sloppy, amateurisch type placement is irresponsible; it spoils even the best image and impedes reading."
[E. Imhof '75]

Beschriftungen für:

- Fläche
- Linien
- Punkte

abhängig vom Maßstab!
(Berlin: Fläche oder Punkt)

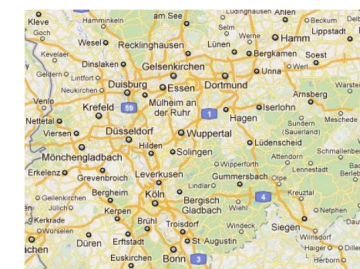
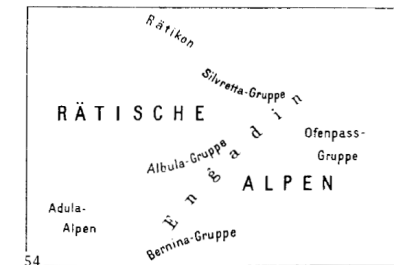
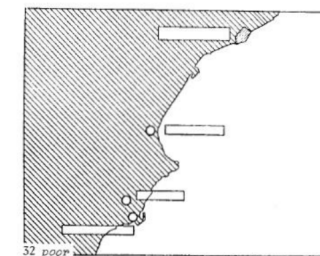
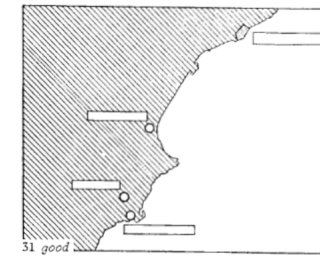
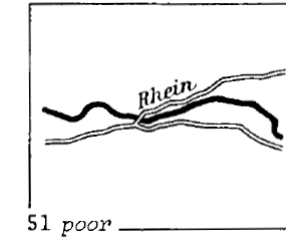
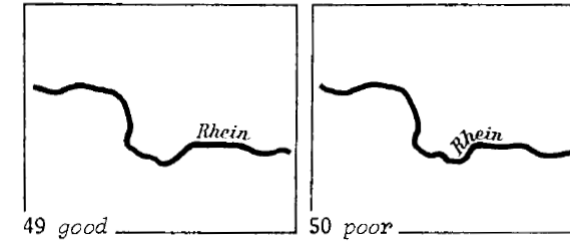
Eduard Imhof
*1895 Schiers, Schweiz
†1986 Erlenbach, Schweiz



Fotografie von Ernst Liniger, um 1980 (Alpines Museum der Schweiz, Bern)

Kriterien für Beschriftungen

- Lesbarkeit
- klare Zuordnung von Namen zu Objekten
- Namen sollen anderen Karteninhalt wenig stören (keine Verdeckung von Objekten)
- Namen sollen Verständnis von Kartenobjekten erleichtern
- Schrifttyp und -größe sollen Klassen und Hierarchien wiedergeben.
- Dichte der Namen soll angemessen variieren.



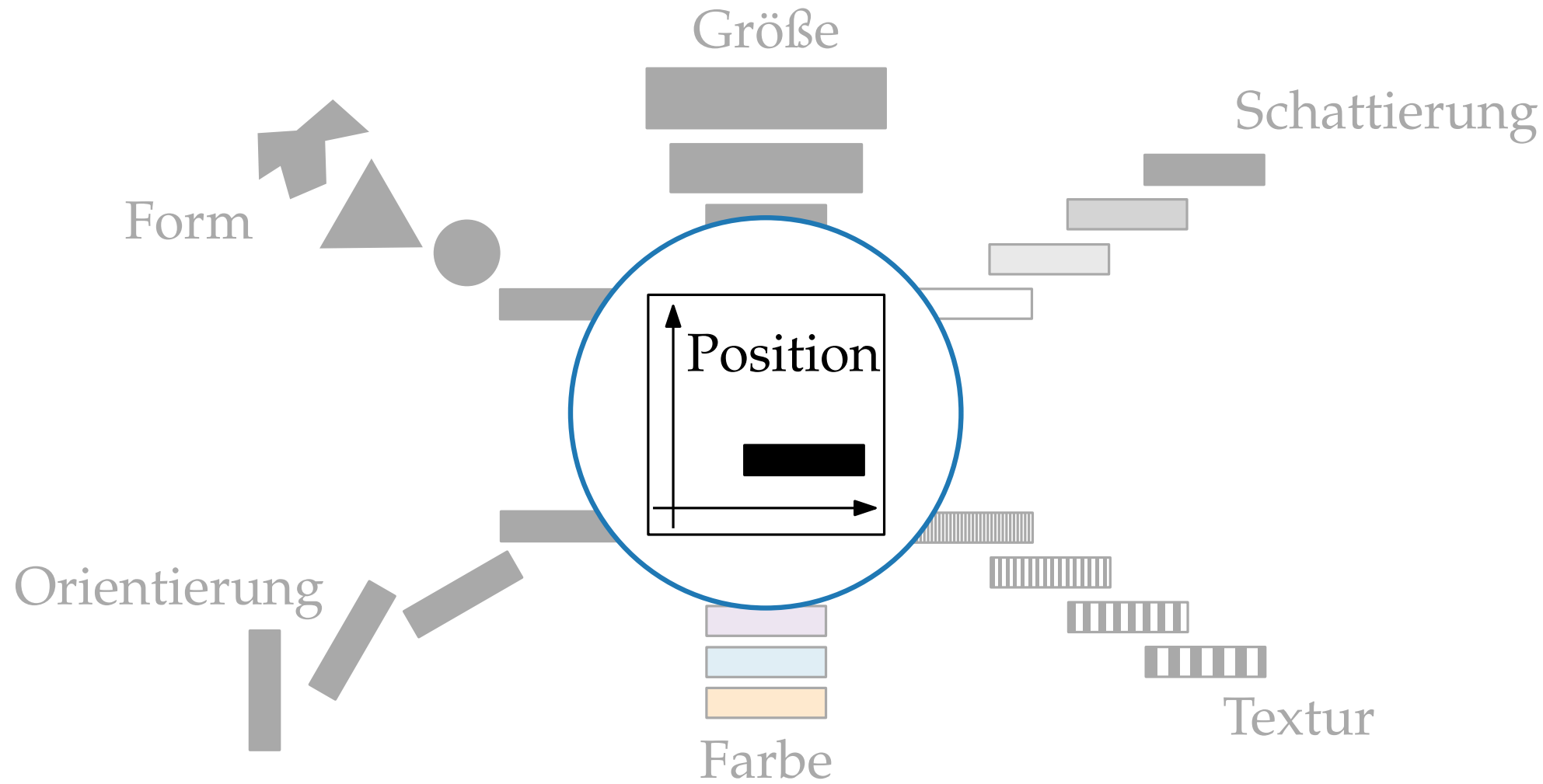
Eduard Imhof
*1895 Schiers, Schweiz
†1986 Erlenbach, Schweiz



Fotografie von Ernst Liniger,
um 1980 (Alpines Museum
der Schweiz, Bern)

Was können wir anpassen?

Jacques Bertin definierte visuelle Variablen [Bertin'67]

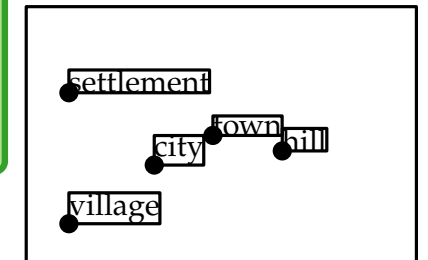
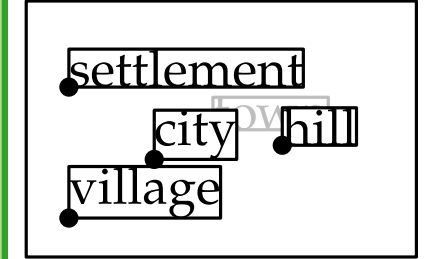


Jacques Bertin
 *1918 Maisons-Laffitte, Frankreich
 †2010 Paris, Frankreich

Geometrische Beschriftungsmodelle

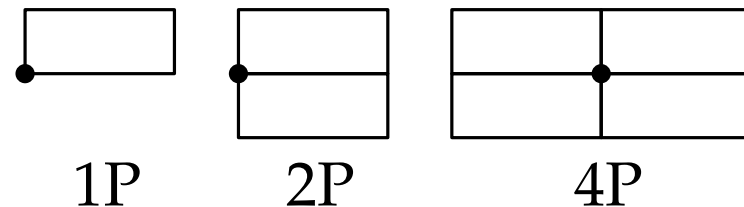


- Geg.:** n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)
- Ges.:** zulässige Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNUMBER)
- oder zulässige Beschriftung **aller** Label, so dass sich keine zwei Label schneiden und die **Schriftgröße** maximal ist (MaxSIZE)

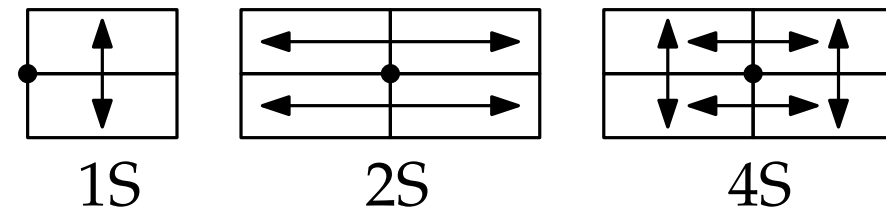


Was ist eine zulässige Beschriftung?

diskrete Modelle



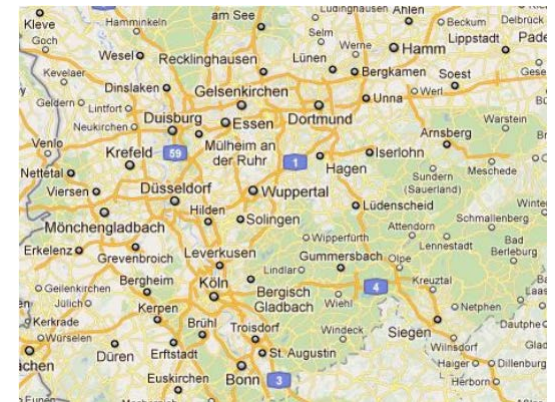
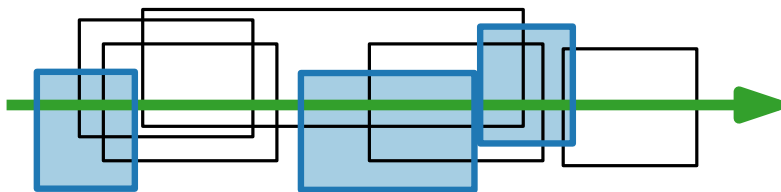
Slider-Modelle



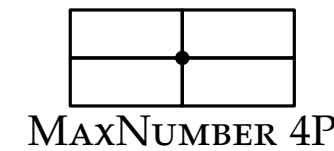
Algorithmen für geographische Informationssysteme

6. Vorlesung Kartenbeschriftung

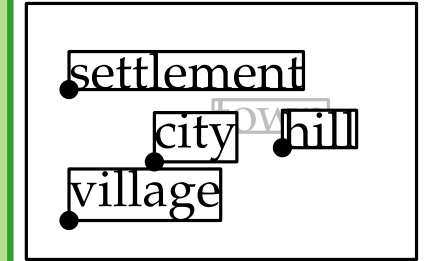
Teil II: Diskretes Modell



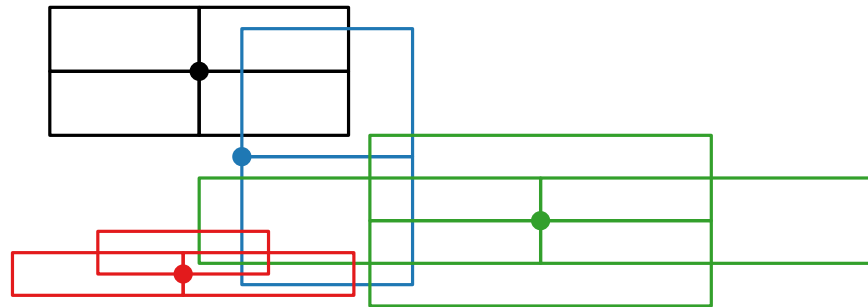
Umformulierung



- Geg.:** n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)
- Ges.:** zulässige Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MAXNUMBER)



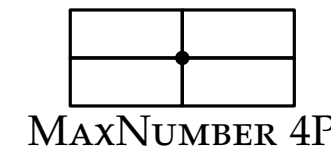
Für jeden Punkt p_i , $1 \leq i \leq m$ gibt es eine diskrete Menge von Labelpositionen, d.h., eine Menge R_i von achsenparallelen Rechtecken, wobei $p_i \in r$ für jedes $r \in R_i$.



- Geg.:** Eine Menge $R = R_1 \cup \dots \cup R_n$ von Rechtecken
- Ges.:** Eine **maximale Teilmenge** aus R ohne Überlappung

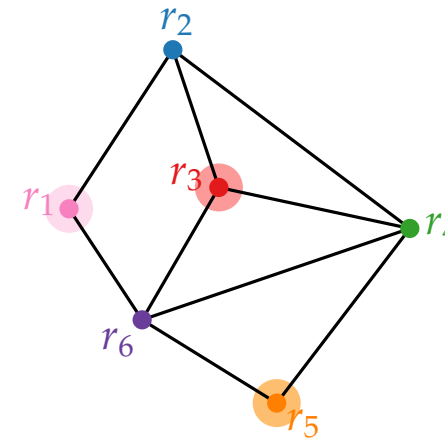
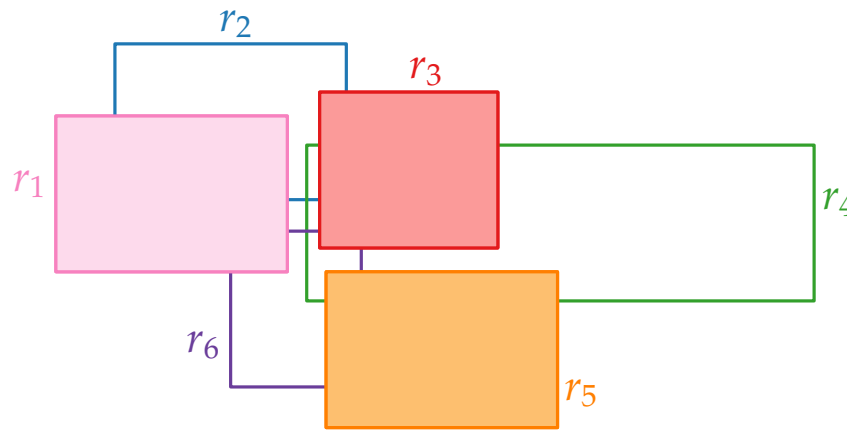
Es werden nie zwei Rechtecke r_1, r_2 für denselben Punkt p gewählt, da $p \in r_1$ und $p \in r_2$; r_1 und r_2 schneiden sich also.

Schnittgraph



Geg.: Eine Menge $R = R_1 \cup \dots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung



Der **Schnittgraph** $G = (R, E)$ enthält einen Knoten für jedes Rechteck und eine Kante (u, v) genau dann wenn sich die Rechtecke u und v schneiden.

Lösung: Finde **größte unabhängige Menge** in G

- Für allgemeine Graphen NP-schwer
- Für Schnittgarphen von uniformen Quadraten NP-schwer [Imai & Asano 1983]

Ziel: Approximationsalgorithmus

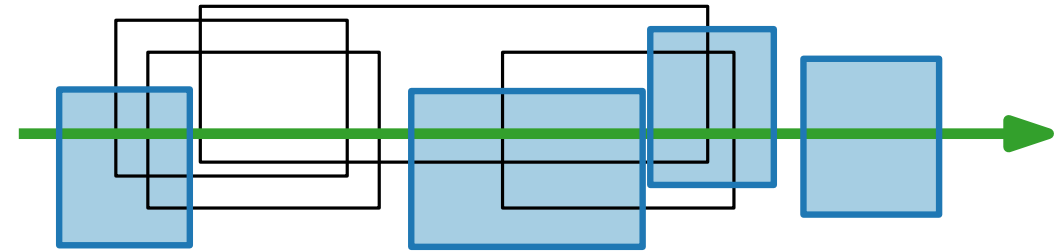
Spezialfall

Geg.: Eine Menge $R = R_1 \cup \dots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

Angenommen, es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.

Strategie: Greedy!



L = Rechtecke sortiert nach x Koordinate des rechten Randes

$S = \emptyset$; $r = L.\text{head}$

while $r \neq \text{nil}$ **do**

$S = S \cup \{r\}$; $r' = r.\text{next}$

while $r' \neq \text{nil}$ **and** $r'.x\text{Min} \leq r.x\text{Max}$ **do**

$r' = r'.\text{next}$

$r = r'$

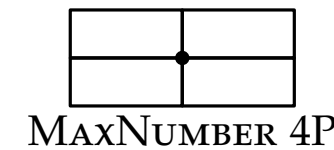
return S

Laufzeit?

■ $\mathcal{O}(n \log n)$

■ $\mathcal{O}(n)$ falls vorsortiert

Allgemeinfall



Geg.: Eine Menge $R = R_1 \cup \dots \cup R_n$ von Rechtecken

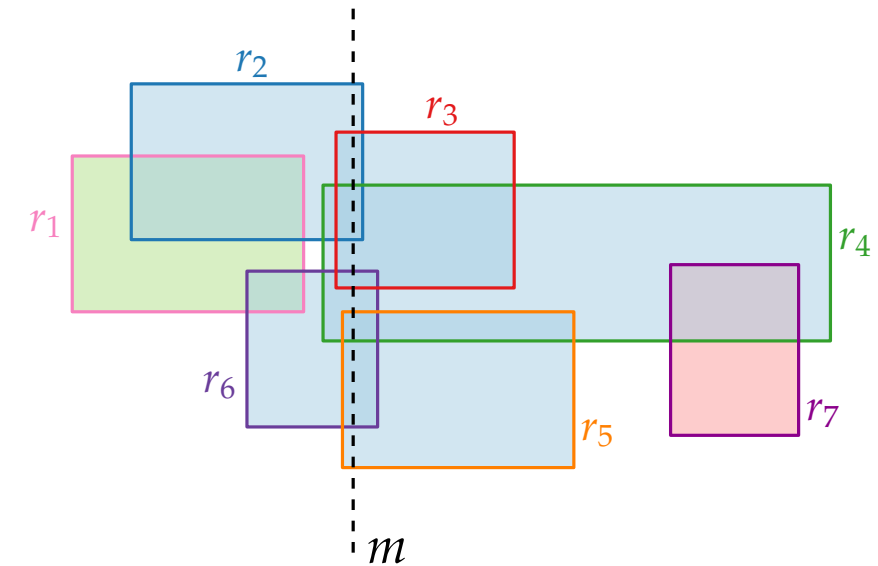
Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

- Bestimme Median m von $\bigcup_{r \in R} \{r.xMin, r.xMax\}$
- $R_m \subseteq R$: Rechtecke auf der Vertikalen durch m
- $R_\ell \subseteq R$: Rechtecke links der Vertikalen durch m
- $R_r \subseteq R$: Rechtecke rechts der Vertikalen durch m

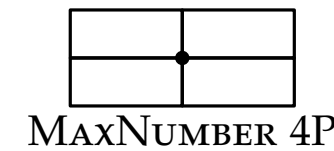
Algorithmus für $n \leq 2$:

```

if  $n = 0$  then return  $\emptyset$ 
if  $n = 1$  then return  $\{r_1\}$ 
if  $r_1 \cap r_2$  then return  $\{r_1\}$ 
return  $\{r_1, r_2\}$ 
  
```



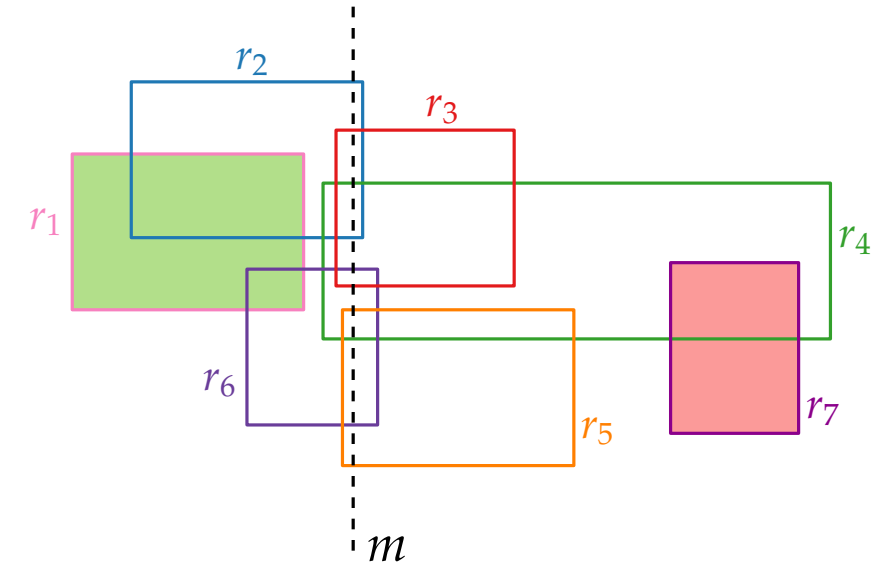
Allgemeinfall



Geg.: Eine Menge $R = R_1 \cup \dots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

- Bestimme Median m von $\bigcup_{r \in R} \{r.xMin, r.xMax\}$
- $R_m \subseteq R$: Rechtecke auf der Vertikalen durch m
- $R_\ell \subseteq R$: Rechtecke links der Vertikalen durch m
- $R_r \subseteq R$: Rechtecke rechts der Vertikalen durch m



MAXNUMBERRECT(R)

Berechne m, R_ℓ, R_m, R_r optimal

$L_m = \text{GREEDYMAXNUMBER}(R_m)$

$L_\ell = \text{MAXNUMBERRECT}(R_\ell)$

$L_r = \text{MAXNUMBERRECT}(R_r)$

if $|L_m| \geq |L_\ell| + |L_r|$ **then**

return L_m

return $L_\ell \cup L_r$

Laufzeit?

$\mathcal{O}(n)$

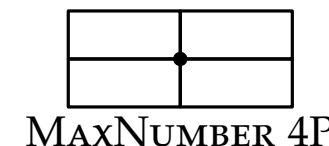
$\mathcal{O}(n)$

$\leq T(n/2)$

$\leq T(n/2)$

$$T(n) \leq 2T(n/2) + \mathcal{O}(n) \stackrel{\text{wie MERGESORT}}{=} \mathcal{O}(n \log n)$$

Approximationsfaktor



Sei L^* eine optimale Lösung für R .

Satz. MAXNUMBERRECT(R) findet eine gültige Lösung L mit $|L| \geq |L^*| / \log n$.

Beweis. Durch Induktion über n .

(IA) MAXNUMBERRECT ist optimal für $n \leq 2$.

(IV) Satz gilt für alle $|R| < n$.

(IS) Sei $|R| = n$.

Seien L_m^* , L_ℓ^* , L_r^* optimale Lösungen für R_m , R_ℓ , R_r .

$$\blacksquare |L_m| = |L_m^*| \geq |L^* \cap R_m|$$

$$\blacksquare |R_\ell| \leq n/2 \quad (n \text{ vertikale Rände links von } m)$$

$$\rightarrow |L_\ell| \stackrel{(IV)}{\geq} |L_\ell^*| / \log(n/2) \geq |L^* \cap R_\ell| / (\log n - 1)$$

$$\rightarrow |L_r| \geq |L^* \cap R_r| / (\log n - 1)$$

$$\blacksquare |L| = \max\{|L_m|, |L_\ell| + |L_r|\} \geq \max\left\{|L^* \cap R_m|, \frac{|L^* \cap R_\ell| + |L^* \cap R_r|}{\log n - 1}\right\}$$

MAXNUMBERRECT(R)

if $n = 0$ **then return** $\emptyset$

if $n = 1$ **then return** $\{r_1\}$

if $n = 2$ **then**

if $r_1 \cap r_2$ **then return** $\{r_1\}$

else return $\{r_1, r_2\}$

Berechne m , R_ℓ , R_m , R_r optimal

$L_m = \text{GREEDYMAXNUMBER}(R_m)$

$L_\ell = \text{MAXNUMBERRECT}(R_\ell)$

$L_r = \text{MAXNUMBERRECT}(R_r)$

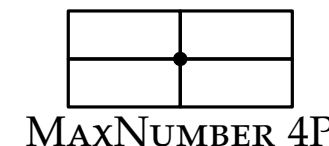
if $|L_m| \geq |L_\ell| + |L_r|$ **then**

return L_m

return $L_\ell \cup L_r$

$$= |L^*| - |L^* \cap R_m|$$

Approximationsfaktor



Sei L^* eine optimale Lösung für R .

Satz. MAXNUMBERRECT(R) findet eine gültige Lösung L mit $|L| \geq |L^*| / \log n$.

Beweis. Durch Induktion über n .

$$\blacksquare |L| \geq \max \left\{ |L^* \cap R_m|, \frac{|L^*| - |L^* \cap R_m|}{\log n - 1} \right\}$$

Fall 1: $|L^* \cap R_m| \geq |L^*| / \log n$
 $\rightarrow |L| \geq |L^* \cap R_m| \geq |L^*| / \log n$

Fall 2: $|L^* \cap R_m| < |L^*| / \log n$
 $\rightarrow |L| \geq \frac{|L^*| - |L^* \cap R_m|}{\log n - 1} > \frac{|L^*| - |L^*| / \log n}{\log n - 1}$

$$= \frac{|L^*|(1 - 1/\log n)}{\log n - 1} = \frac{|L^*|(1 - 1/\log n)}{\log n(1 - 1/\log n)}$$

$$= |L^*| / \log n$$

□

MAXNUMBERRECT(R)

```

if  $n = 0$  then return  $\emptyset$ 
if  $n = 1$  then return  $\{r_1\}$ 
if  $n = 2$  then
    if  $r_1 \cap r_2$  then return  $\{r_1\}$ 
    else return  $\{r_1, r_2\}$ 

```

Berechne m, R_ℓ, R_m, R_r optimal

$L_m = \text{GREEDYMAXNUMBER}(R_m)$

$L_\ell = \text{MAXNUMBERRECT}(R_\ell)$

$L_r = \text{MAXNUMBERRECT}(R_r)$

```

if  $|L_m| \geq |L_\ell| + |L_r|$  then
    return  $L_m$ 

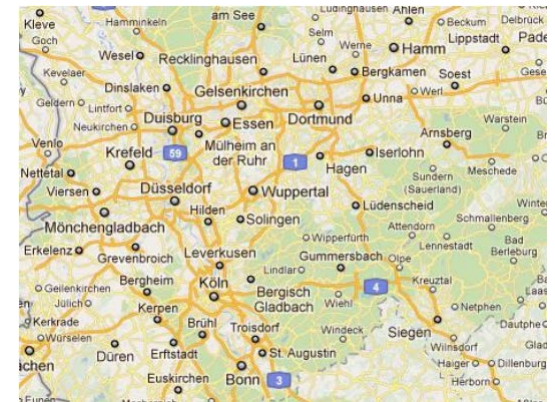
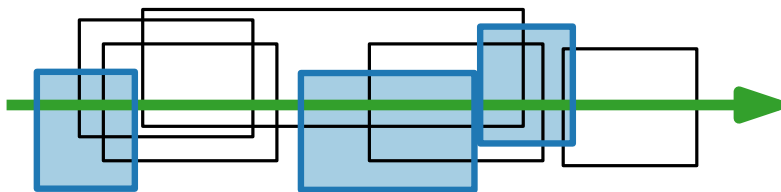
```

return $L_\ell \cup L_r$

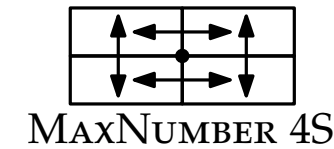
Algorithmen für geographische Informationssysteme

6. Vorlesung Kartenbeschriftung

Teil III: Slider-Modell

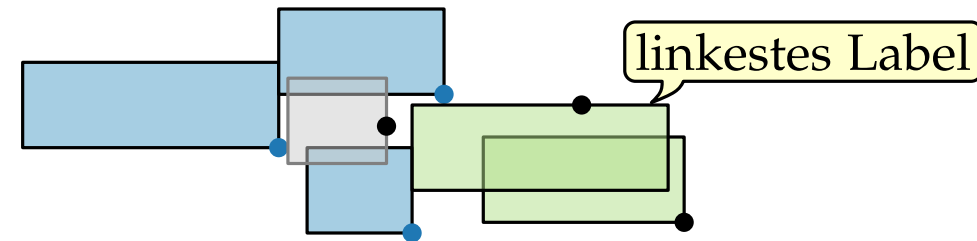


Noch ein Greedy Algorithmus



- Geg.:** n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)
- Ges.:** zulässige Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MAXNUMBER)

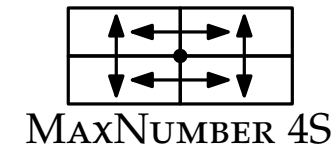
Für eine Menge P' von Punkten mit bereits platzierten Labeln heißt Label ℓ für Punkt $p \in P \setminus P'$ **linkestes** Label (in $P \setminus P'$), falls seine rechte Kante unter allen noch überlappungsfrei platzierbaren Labeln am weitesten links liegt.



GREEDY4S(P, L)

while linkestes Label ℓ existiert **do**
 $\perp$ platziere ℓ linkest möglich

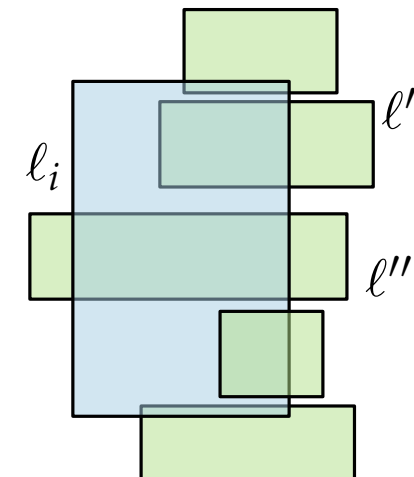
Approximationsfaktor



- Angenommen, alle Labels haben Höhe ~~1~~ $\in [1, m]$.
- Sei L Greedy Lösung, L^* optimale Lösung.
- Betrachte Labels $\ell_1, \dots, \ell_{|L|}$ in der Reihenfolge wie von GREEDY4S ausgewählt.
- Jedes Label ℓ_i schneidet mindestens ein Label in L^* (sonst L^* nicht optimal).
- Sei L_i die Menge der Label in L^* , die ℓ_i schneiden, aber nicht $\ell_1, \dots, \ell_{i-1}$.
- Wie groß ist $|L_i|$? $\rightarrow |L_i| \leq \cancel{2}! m + 1$
- $\ell' \in L_i$
 - $\rightarrow \ell_1, \dots, \ell_{i-1}, \ell'$ überlappungsfrei
 - $\rightarrow \ell'$ **rechter** als ℓ_i (wegen Greedy Wahl)

GREEDY4S(P, L)

while linkstes Label ℓ existiert **do**
 $\lfloor$ platziere ℓ linkest möglich



Laufzeit? $\mathcal{O}(n^3)$. Schneller möglich?

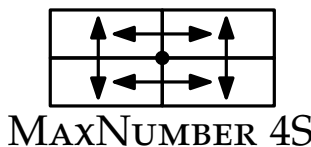
Satz. Für Labels der Höhe 1 berechnet GREEDY4S eine Faktor-1/2-Approximation.

Satz. Für Labels der Höhen in $[1, m]$ berechnet GREEDY4S eine Faktor-1/($m + 1$)-Approximation.

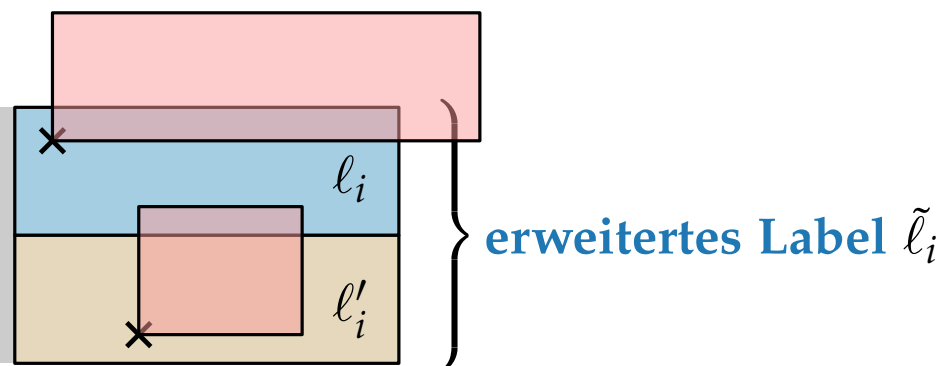
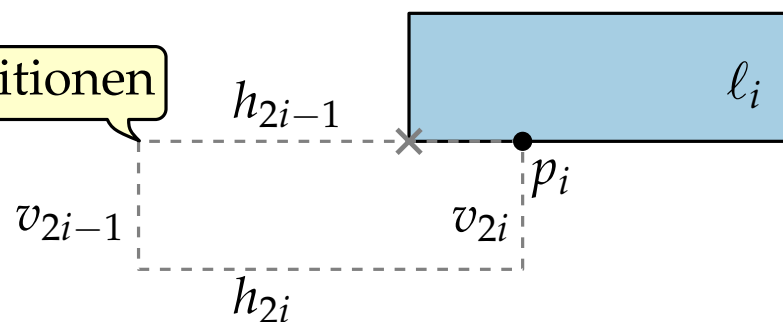
Notation

Wir betrachten nur

Fall mit Labelhöhe 1.



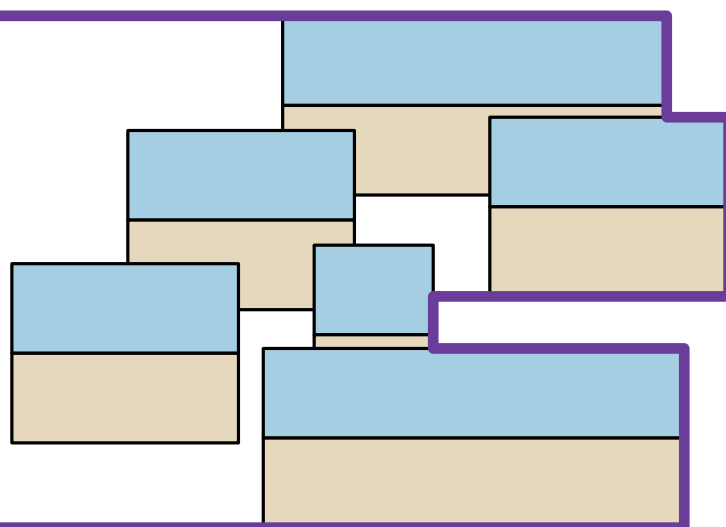
mögliche Positionen



GREEDY4S(P, L)

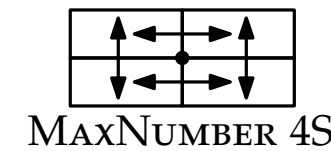
while linkstes Label ℓ existiert **do**
 $\lfloor$ platziere ℓ linkest möglich

- Kein Referenzpunkt darf in $\tilde{\ell}_i$ liegen.
- Da immer linkstes Label platziert wird, darf kein Referenzpunkt links von $\tilde{\ell}_i$ liegen.

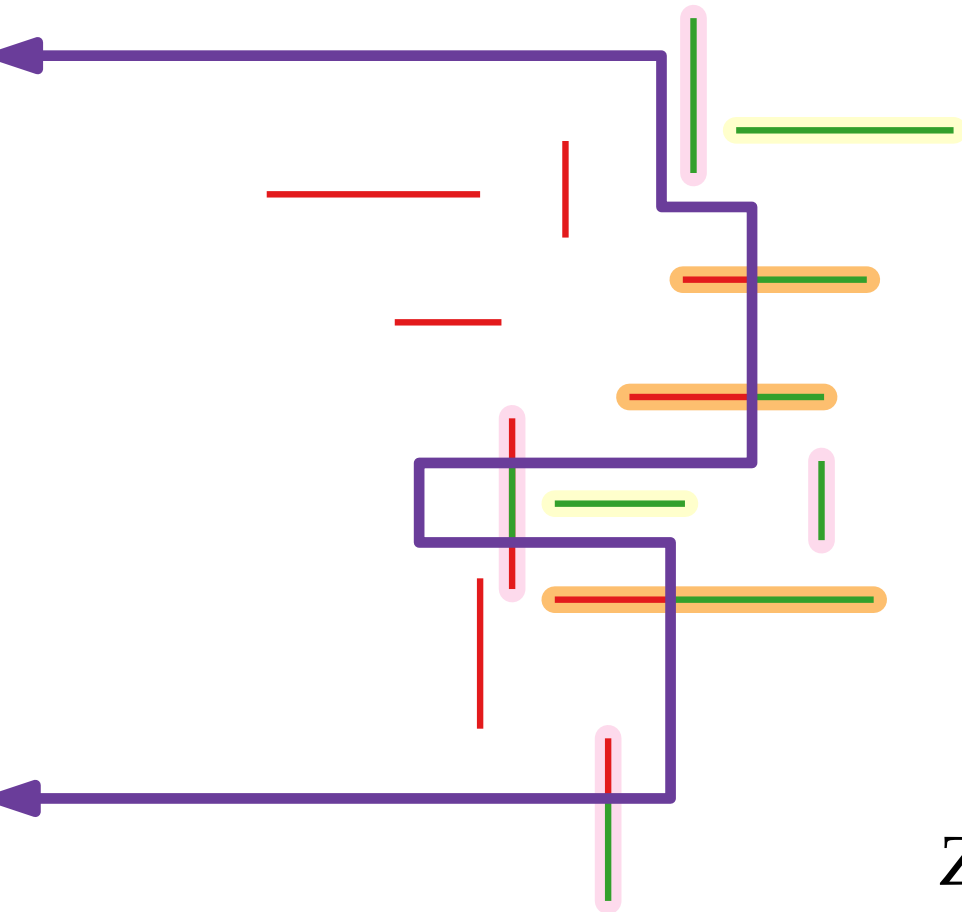


Grenze F als *right envelope* der platzierten erweiterten Label.

Grenze und linkstes Label



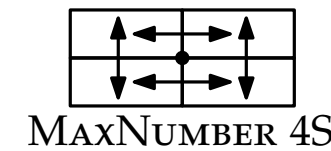
Zur Bestimmung des nächsten linken Labels genügt es, F und die Strecken $h_{2i-1}, h_{2i}, v_{2i-1}, v_{2i}$ für alle Punkte p_i rechts von F zu betrachten.



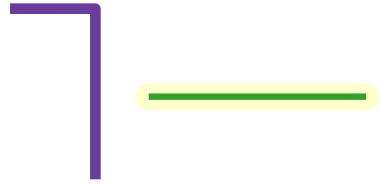
- Menge H der horizontalen Strecken
- Menge V der vertikalen Strecken
- $H_{\text{right}} \subseteq H$: komplett rechts von F
- $H_{\text{int}} \subseteq H$: schneiden F
- $V_{\text{int,right}} \subseteq V$: enthalten Punkt rechts von F
- Linkeste Punkte im grünen Bereich der Strecken sind zulässige Kandidaten

Zur effizienten Verwaltung der linkensten Kandidaten benötigen wir mehrere geometrische Datenstrukturen.

Datenstrukturen

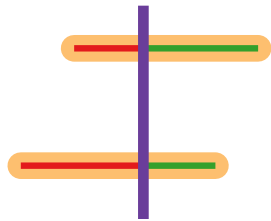


Menge H_{right} :



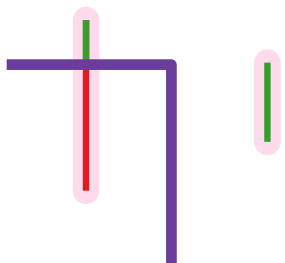
- speichere für jede Strecke in H_{right} deren rechten Endpunkt (= linkest mögliche Position der rechten Labelseite)
- Datenstruktur: MINHEAP $\mathcal{H}_{\text{right}}$

Menge H_{int} :



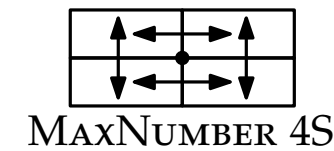
- Balanc. bin. Suchbaum $\mathcal{T}_i$ für jedes vertikale Segment f_i von F
- speichere Strecken aus H_{int} , die f_i schneiden sortiert nach y -Koordinaten in den Blättern von $\mathcal{T}_i$
- speichere zusätzlich Labelbreite an jedem Blatt
- an inneren Knoten minimale Labelbreite im Teilbaum
- MINHEAP $\mathcal{H}_{\text{int}}$ für linkestes Label in jedem $\mathcal{T}_i$

Menge $V_{\text{int,right}}$:

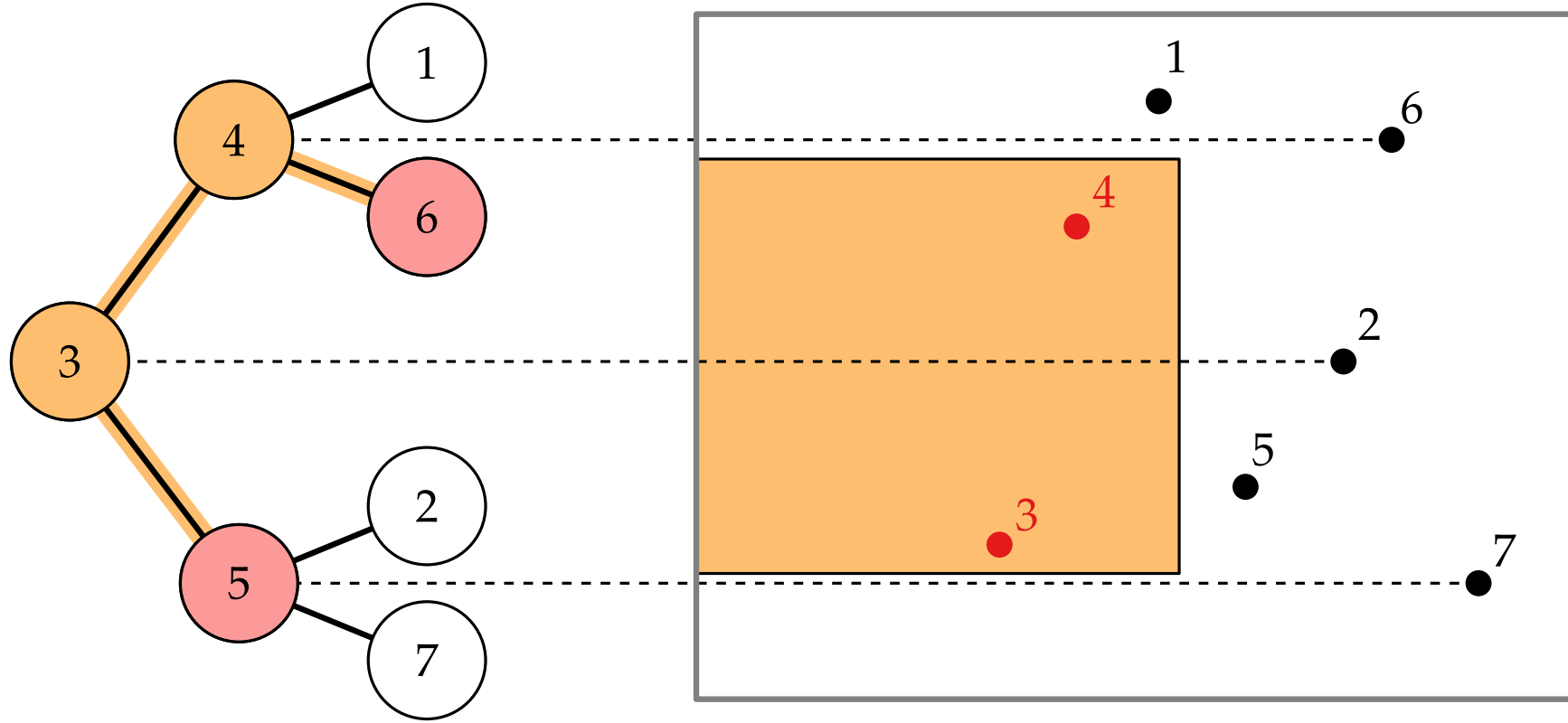


- MINHEAP $\mathcal{H}_V$ geordnet nach x -Koordinaten der rechten Labelgrenzen
- Lazy Strategie: Segment wird erst aus $\mathcal{H}_V$ geworfen, wenn es linkestes Label liefern würde, aber nicht mehr in $V_{\text{int,right}}$ liegt.

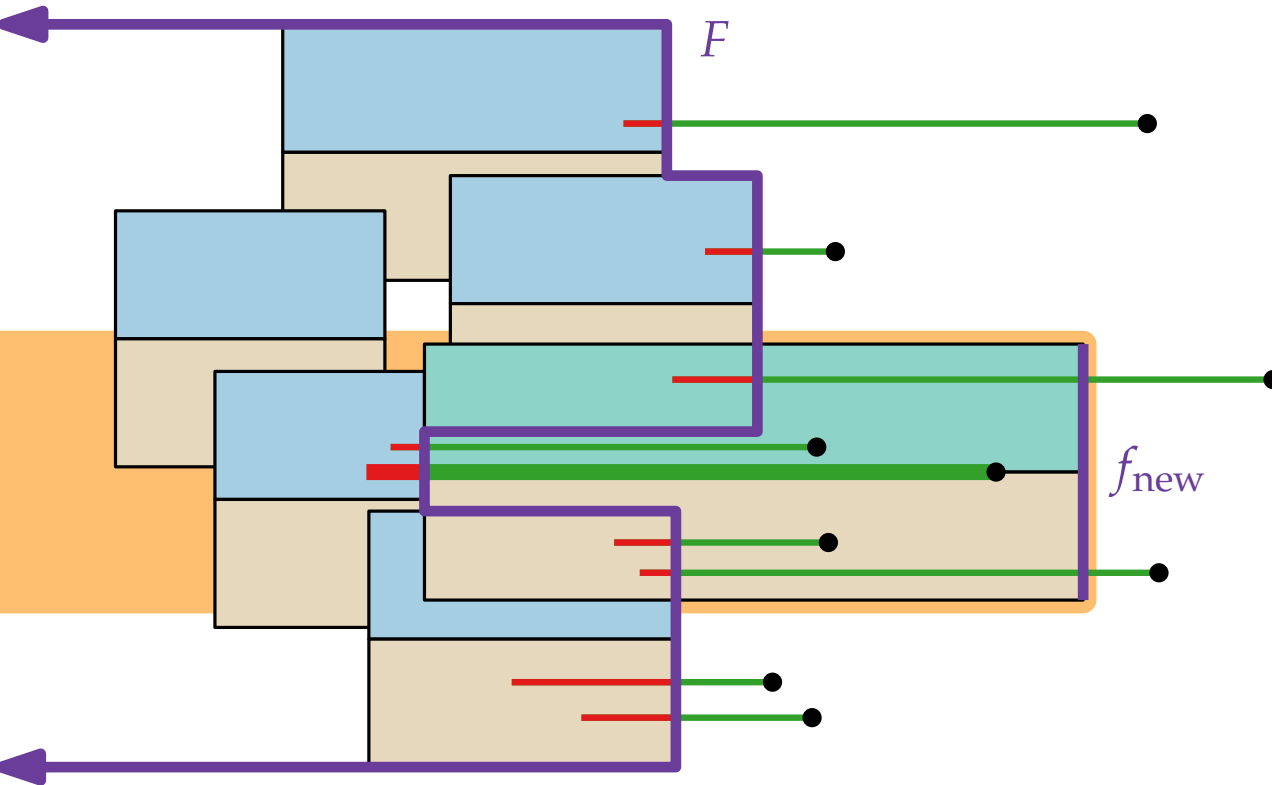
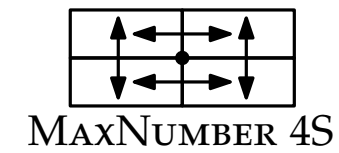
Priority Search Trees (PRT)



- Datenstruktur für Bereichsabfragen der Form $[-\infty, x] \times [y, y']$
- Heap-Eigenschaft für x
- Suchbaum für y
- Größe $\mathcal{O}(n)$
- Einfügen und Löschen in $\mathcal{O}(\log n)$ Zeit
- Suchen in $\mathcal{O}(k + \log n)$ Zeit ($k = \text{Ausgabegröße}$)



Updates



■ Updates von H_{right}

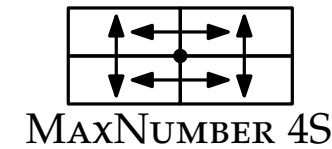
➡ nutze PST für linke Endpunkte der Strecken

■ Updates von H_{int}

➡ nutze PST für rechte Endpunkte der Strecken
Suchbäume und $\mathcal{H}_{\text{int}}$ anpassen

Der Algorithmus

von van Kreveld, Strijk und Wolff



GREEDY4S(P, L)

while $\mathcal{H}_{\text{right}}$, $\mathcal{H}_{\text{int}}$ oder $\mathcal{H}_V$ nicht leer **do**

$v = \text{FINDMIN}(\mathcal{H}_V)$

while v links von F **do**

$\text{DELETMIN}(\mathcal{H}_v)$

$v = \text{FINDMIN}(\mathcal{H}_V)$

$\ell_i =$ linkstes Label aus $\mathcal{H}_{\text{right}}$, $\mathcal{H}_{\text{int}}$ und $\mathcal{H}_V$

 füge ℓ_i zur Beschriftung hinzu

$f_{\text{new}} =$ rechte Kante von $\tilde{\ell}_i$

 update F und $\mathcal{T}_V$ mit f_{new}

 suche mit der Region links von f_{new} und update Datenstrukturen

 entferne alle Verweise auf ℓ_i aus den Datenstrukturen

➔ Laufzeit $\mathcal{O}(n \log n)$, Speicher $\mathcal{O}(n)$