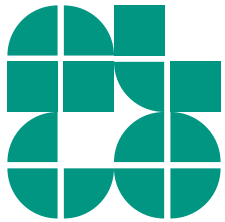
Übung Algorithmische Kartografie

Übungsblatt 2 & 3

LEHRSTUHL FÜR ALGORITHMIK I · INSTITUT FÜR THEORETISCHE INFORMATIK · FAKULTÄT FÜR INFORMATIK

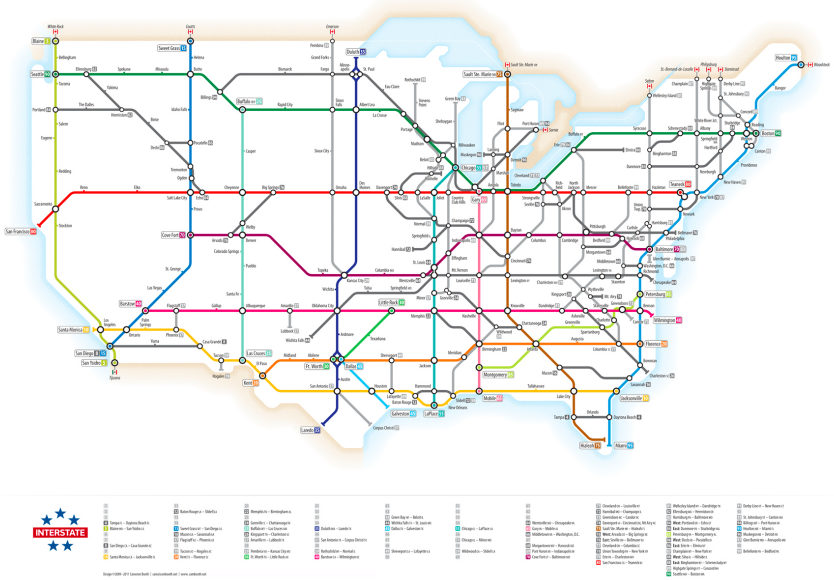
Benjamin Niedermann

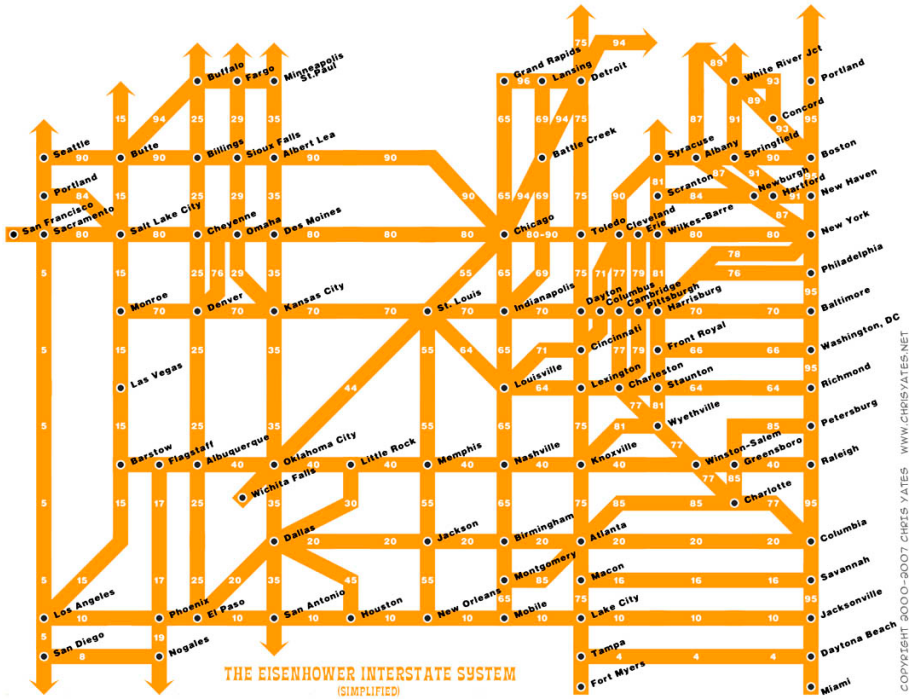
23.05.2013



Übungsblatt 2 – Schematisierung von Karten

Schematische Netzkarten





Anforderungen:

- sehr starke Vereinfachung/Abstraktion
- Erkennbarkeit erhalten
- Einschränkung der Kantenrichtungen

→ Zeichne eingebetteten Graphen mit festen Knotenpositionen

Schematisierung

Hier bedeutet Schematisierung das Zeichnen von Graphen mit diskreten und oft uniformen

Winkeleinschränkungen

für die Kantenrichtungen/-steigungen.

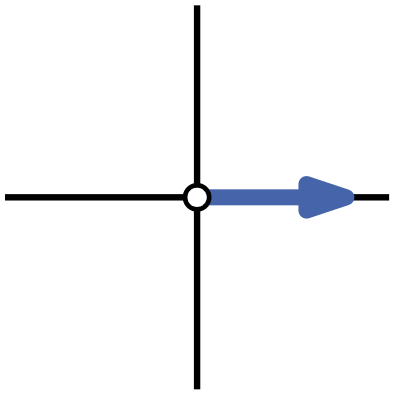
Schematisierung

Hier bedeutet Schematisierung das Zeichnen von Graphen mit diskreten und oft uniformen

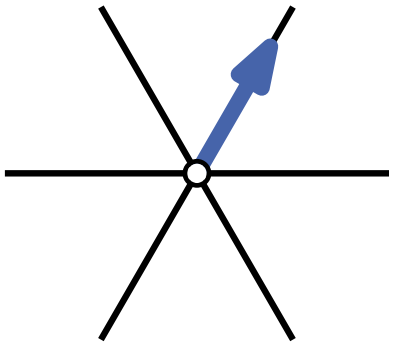
Winkeleinschränkungen

für die Kantenrichtungen/-steigungen.

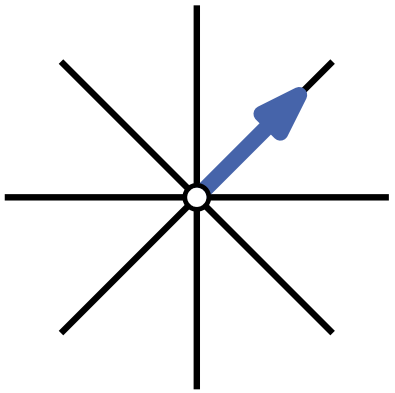
Beispiele:



rektilinear



hexilinear



oktilinear

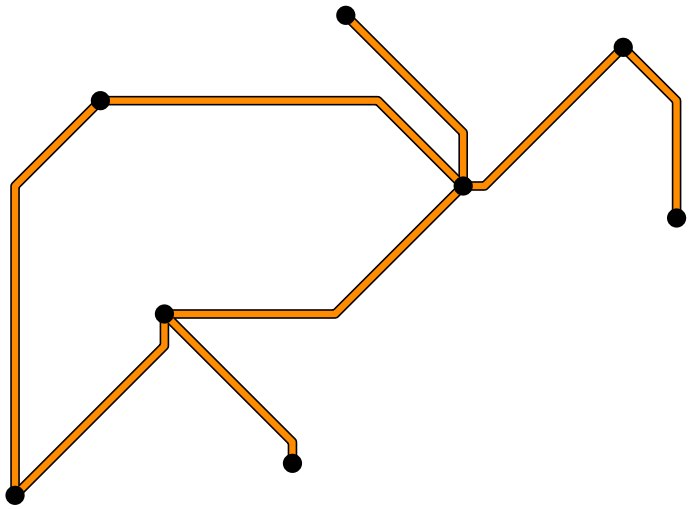
...

Schematische Straßenkarten

Geg: Straßenkarte $M = \{c_1, \ldots, c_m\}$, wobei jedes c_i einfacher Kantenzug ist und keine zwei c_i, c_j $(i \neq j)$ sich schneiden (außer an gemeinsamen Endknoten)

Ges: Topologisch äquivalente schematische Karte $M' = \{c'_1, \ldots, c'_m\}$ mit **gleichen Endpunkten** wie M, wobei jedes c'_i oktilinear ist und ≤ 2 Knicke hat
$\rightarrow$ verzerre Straßenform, aber nicht Kreuzungen und Orte

Eingabekarte M

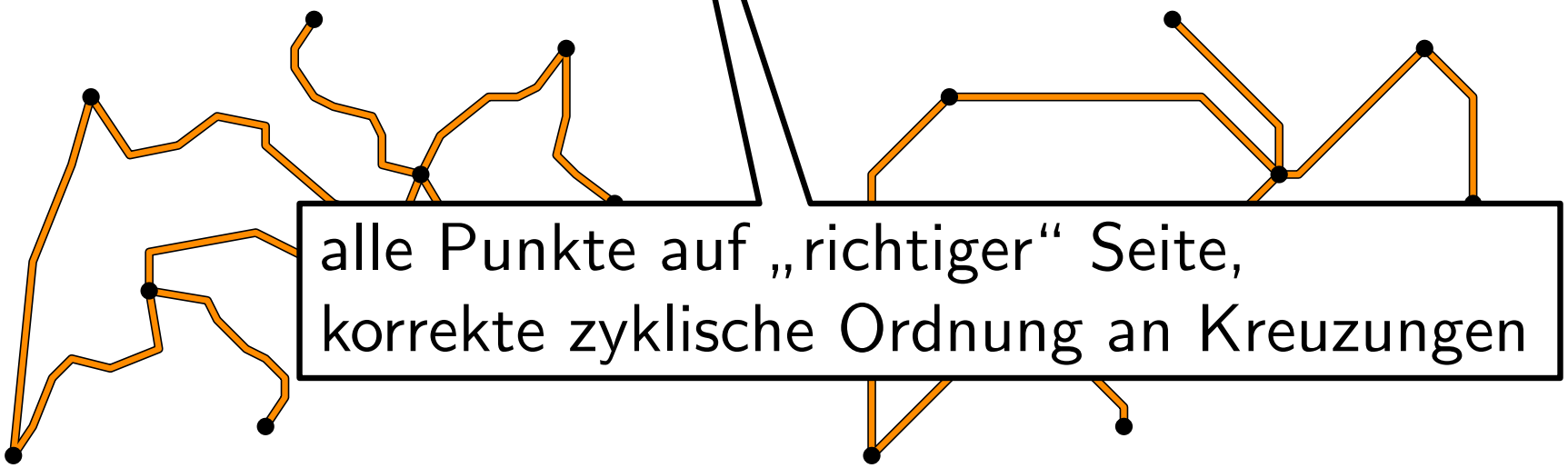


oktilineare Ausgabekarte M'

Schematische Straßenkarten

Geg: Straßenkarte $M = \{c_1, \ldots, c_m\}$, wobei jedes c_i einfacher Kantenzug ist und keine zwei c_i, c_j $(i \neq j)$ sich schneiden (außer an gemeinsamen Endknoten)

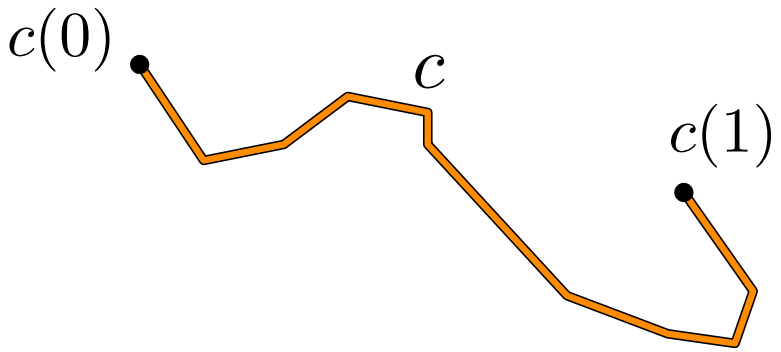
Ges: Topologisch äquivalente schematische Karte $M' = \{c'_1, \ldots, c'_m\}$ mit **gleichen Endpunkten** wie M, wobei jedes c'_i oktilinear ist und ≤ 2 Knicke hat
$\rightarrow$ verzerre Straßenform, aber nicht Kreuzungen und Orte



Eingabekarte M oktilineare Ausgabekarte M'

Notation

Betrachte Kantenzug/Pfad c als Abbildung $c : [0, 1] \to \mathbb{R}^2$, wobei die Endpunkte jeweils $c(0)$ und $c(1)$ sind.

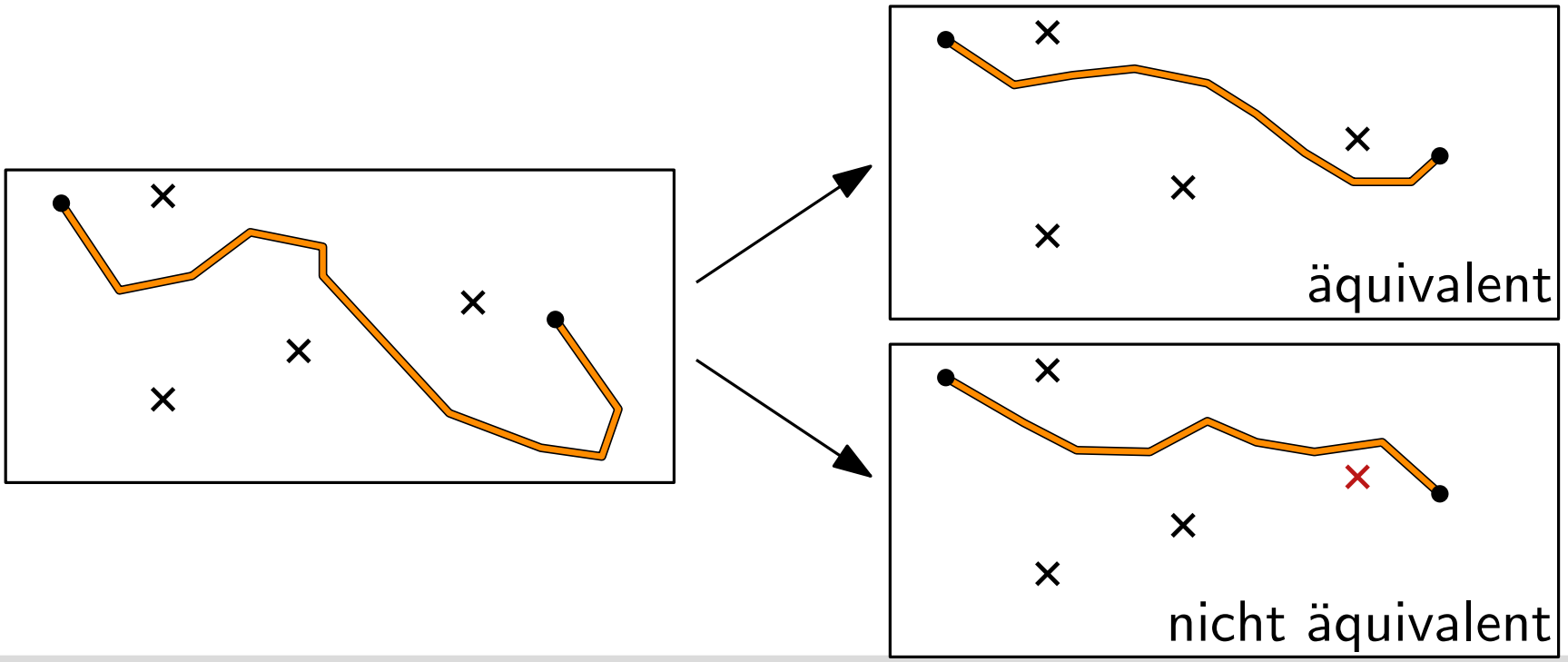


P_M: Menge der Endpunkte von M

Äquivalente Karten

Def: Gegeben Menge von Hindernissen P. Zwei Pfade c, c' sind **äquivalent**, falls es eine stetige Abbildung $F : [0,1]^2 \to \mathbb{R}^2 \setminus P$ gibt, die c in c' transformiert, d.h.

- $F(0,t) = c_1(t)$ und $F(1,t) = c_2(t)$ für alle $t \in [0,1]$.
- $F(s,0) = c_1(0) = c_2(0)$ und $F(s,1) = c_1(1) = c_2(1)$ für alle $s \in [0,1]$.



Äquivalente Karten

Def: Gegeben Menge von Hindernissen P. Zwei Pfade c, c' sind **äquivalent**, falls es eine stetige Abbildung $F : [0,1]^2 \to \mathbb{R}^2 \setminus P$ gibt, die c in c' transformiert, d.h.

- $F(0,t) = c_1(t)$ und $F(1,t) = c_2(t)$ für alle $t \in [0,1]$.
- $F(s,0) = c_1(0) = c_2(0)$ und $F(s,1) = c_1(1) = c_2(1)$ für alle $s \in [0,1]$.

Aufgabe 3.2: Zeigen Sie, dass die Relation *die Kurve c_i ist äquivalent zur Kurve c_k* tatsächlich eine Äquivalenzrelation ist.

Äquivalente Karten

Def: Gegeben Menge von Hindernissen P. Zwei Pfade c, c' sind **äquivalent**, falls es eine stetige Abbildung $F : [0,1]^2 \to \mathbb{R}^2 \setminus P$ gibt, die c in c' transformiert, d.h.

- $F(0,t) = c_1(t)$ und $F(1,t) = c_2(t)$ für alle $t \in [0,1]$.
- $F(s,0) = c_1(0) = c_2(0)$ und $F(s,1) = c_1(1) = c_2(1)$ für alle $s \in [0,1]$.

Reflexiv: Für alle $s,t \in [0,1]$ setze $F(s,t) = c_1(t)$.

Äquivalente Karten

Def: Gegeben Menge von Hindernissen P. Zwei Pfade c, c' sind **äquivalent**, falls es eine stetige Abbildung $F : [0,1]^2 \to \mathbb{R}^2 \setminus P$ gibt, die c in c' transformiert, d.h.

- $F(0,t) = c_1(t)$ und $F(1,t) = c_2(t)$ für alle $t \in [0,1]$.
- $F(s,0) = c_1(0) = c_2(0)$ und $F(s,1) = c_1(1) = c_2(1)$ für alle $s \in [0,1]$.

Symmetrisch:

- Gegeben kontinuierliche Funktion G die c_1 in c_2 transformiert.
- Für alle $s,t \in [0,1]$ setze $F(s,t) = G(1-s,t)$.

Offensichtlich: F ist gesuchte Funktion.

Äquivalente Karten

Def: Gegeben Menge von Hindernissen P. Zwei Pfade c, c' sind **äquivalent**, falls es eine stetige Abbildung $F : [0,1]^2 \to \mathbb{R}^2 \setminus P$ gibt, die c in c' transformiert, d.h.

- $F(0,t) = c_1(t)$ und $F(1,t) = c_2(t)$ für alle $t \in [0,1]$.
- $F(s,0) = c_1(0) = c_2(0)$ und $F(s,1) = c_1(1) = c_2(1)$ für alle $s \in [0,1]$.

Transitiv:

- Gegeben kontinuierliche Funktion F_1, die c_1 in c_2 transformiert und Funktion F_2, die c_2 in c_3 transformiert.
- $F_2 \circ F_1$ transformiert die Kurve c_1 in die Kurve c_3

Da F_1 und F_2 bezüglich $\mathbb{R}^2 \setminus P$ kontinuierlich sind, muss auch $F_2 \circ F_1$ bezüglich $R^2 \setminus P$ kontinuierlich sein.

Äquivalente Karten

Def: Gegeben Menge von Hindernissen P. Zwei Pfade c, c' sind **äquivalent**, falls es eine stetige Abbildung $F : [0,1]^2 \to \mathbb{R}^2 \setminus P$ gibt, die c in c' transformiert, d.h.

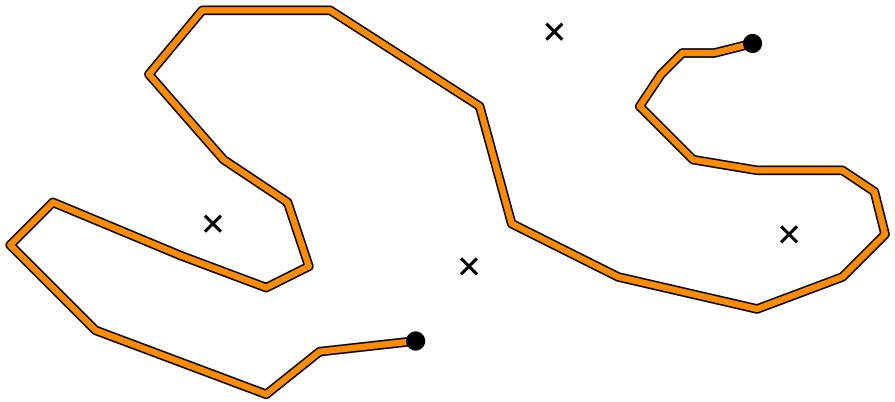
- $F(0,t) = c_1(t)$ und $F(1,t) = c_2(t)$ für alle $t \in [0,1]$.
- $F(s,0) = c_1(0) = c_2(0)$ und $F(s,1) = c_1(1) = c_2(1)$ für alle $s \in [0,1]$.

Def: Zwei Karten M und M' sind **äquivalent**, falls alle $c_i \in M$ und $c_i' \in M'$ äquivalent in $(\mathbb{R}^2 \setminus P_m) \cup \{c_i(0), c_i(1)\}$ sind

Vertikale Pfadordnung

Def: Für Pfad a und Punkt p liegt p **oberhalb/unterhalb** von a, falls für **jeden** zu a äquivalenten Pfad a' in $\mathbb{R}^2 \setminus \{p\}$ der entsprechende vertikale Strahl von p Pfad a' schneidet.

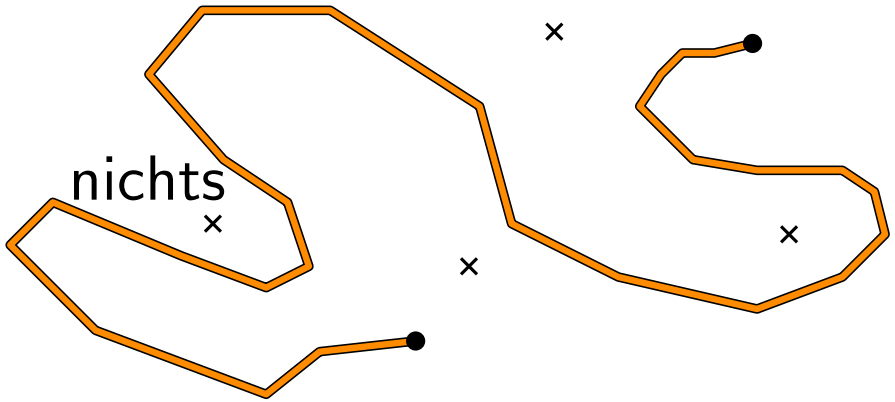
Die Ordnung zweier Pfade a, b ist definiert durch die Lage der Endpunkte des einen Pfades zum anderen Pfad.



Vertikale Pfadordnung

Def: Für Pfad a und Punkt p liegt p **oberhalb/unterhalb** von a, falls für **jeden** zu a äquivalenten Pfad a' in $\mathbb{R}^2 \setminus \{p\}$ der entsprechende vertikale Strahl von p Pfad a' schneidet.

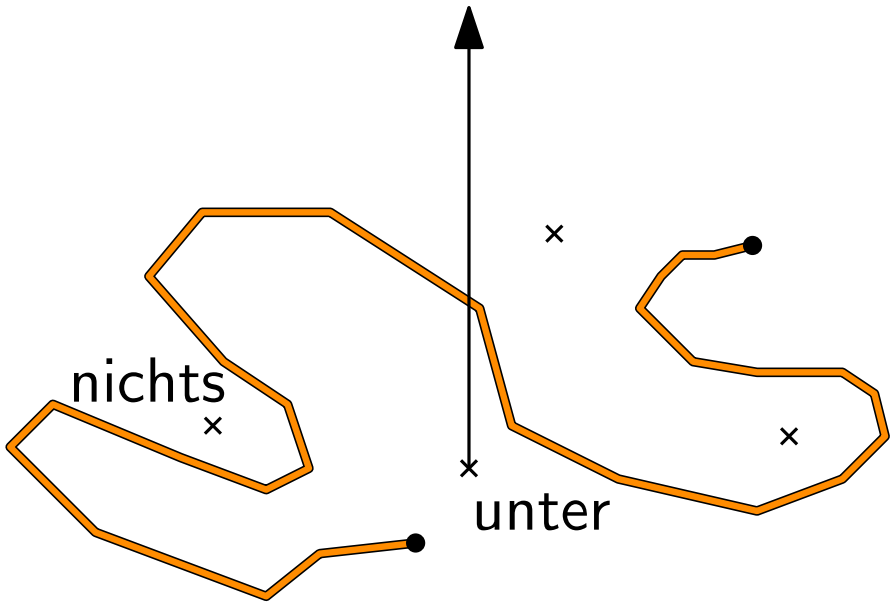
Die Ordnung zweier Pfade a, b ist definiert durch die Lage der Endpunkte des einen Pfades zum anderen Pfad.



Vertikale Pfadordnung

Def: Für Pfad a und Punkt p liegt p **oberhalb/unterhalb** von a, falls für **jeden** zu a äquivalenten Pfad a' in $\mathbb{R}^2 \setminus \{p\}$ der entsprechende vertikale Strahl von p Pfad a' schneidet.

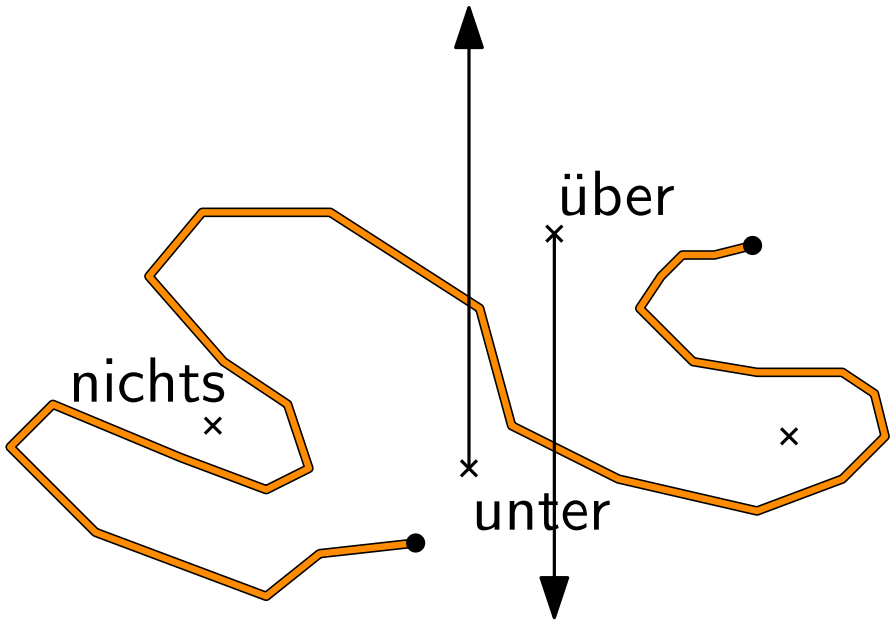
Die Ordnung zweier Pfade a, b ist definiert durch die Lage der Endpunkte des einen Pfades zum anderen Pfad.



Vertikale Pfadordnung

Def: Für Pfad a und Punkt p liegt p **oberhalb/unterhalb** von a, falls für **jeden** zu a äquivalenten Pfad a' in $\mathbb{R}^2 \setminus \{p\}$ der entsprechende vertikale Strahl von p Pfad a' schneidet.

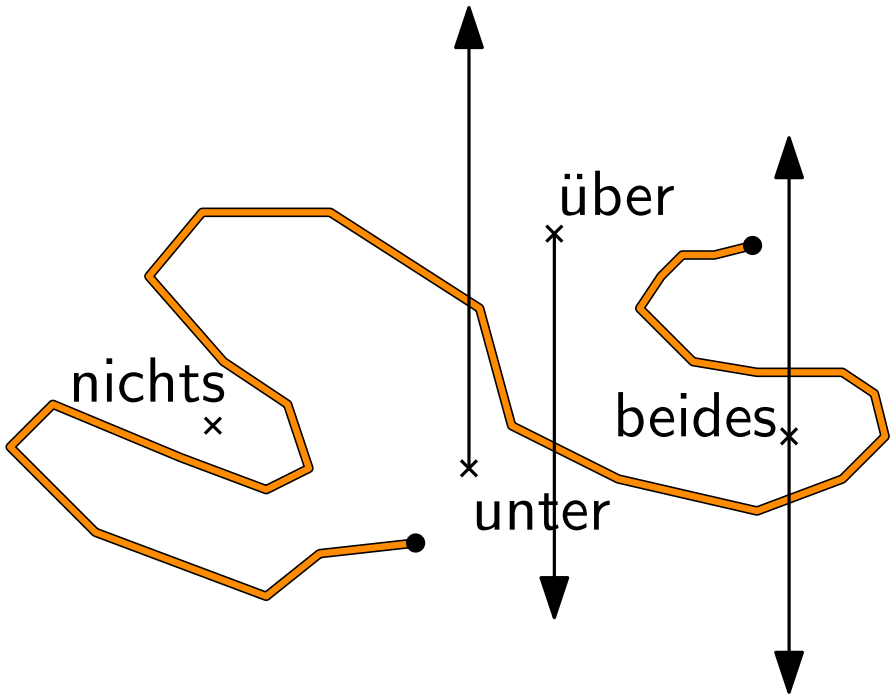
Die Ordnung zweier Pfade a, b ist definiert durch die Lage der Endpunkte des einen Pfades zum anderen Pfad.



Vertikale Pfadordnung

Def: Für Pfad a und Punkt p liegt p **oberhalb/unterhalb** von a, falls für **jeden** zu a äquivalenten Pfad a' in $\mathbb{R}^2 \setminus \{p\}$ der entsprechende vertikale Strahl von p Pfad a' schneidet.

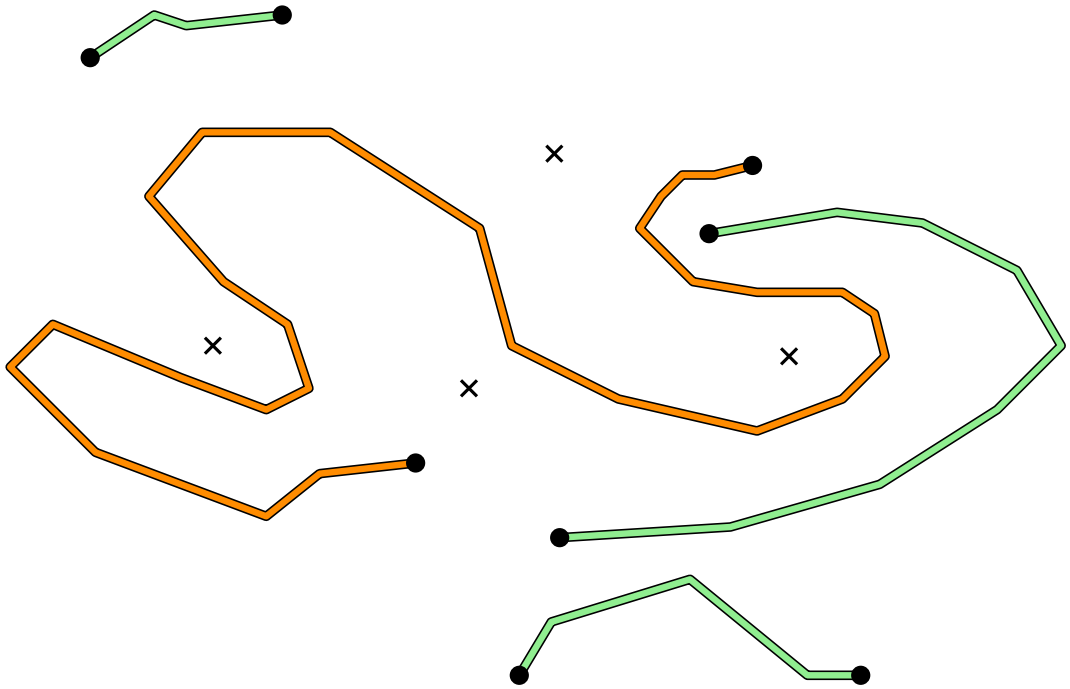
Die Ordnung zweier Pfade a, b ist definiert durch die Lage der Endpunkte des einen Pfades zum anderen Pfad.



Vertikale Pfadordnung

Def: Für Pfad a und Punkt p liegt p **oberhalb/unterhalb** von a, falls für **jeden** zu a äquivalenten Pfad a' in $\mathbb{R}^2 \setminus \{p\}$ der entsprechende vertikale Strahl von p Pfad a' schneidet.

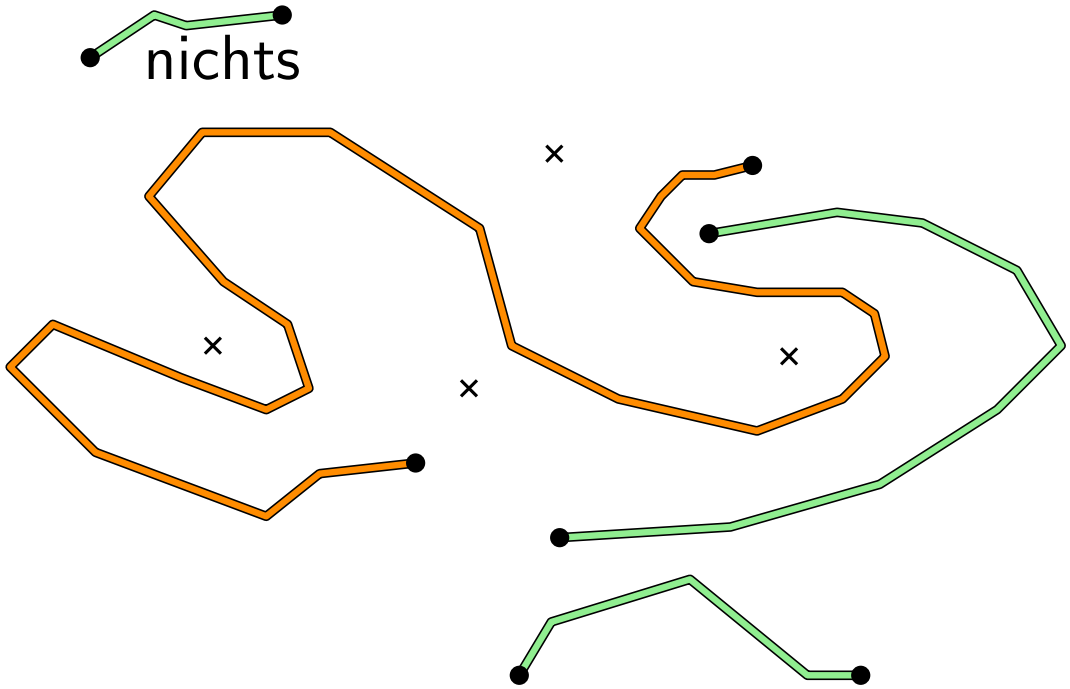
Die Ordnung zweier Pfade a, b ist definiert durch die Lage der Endpunkte des einen Pfades zum anderen Pfad.



Vertikale Pfadordnung

Def: Für Pfad a und Punkt p liegt p **oberhalb/unterhalb** von a, falls für **jeden** zu a äquivalenten Pfad a' in $\mathbb{R}^2 \setminus \{p\}$ der entsprechende vertikale Strahl von p Pfad a' schneidet.

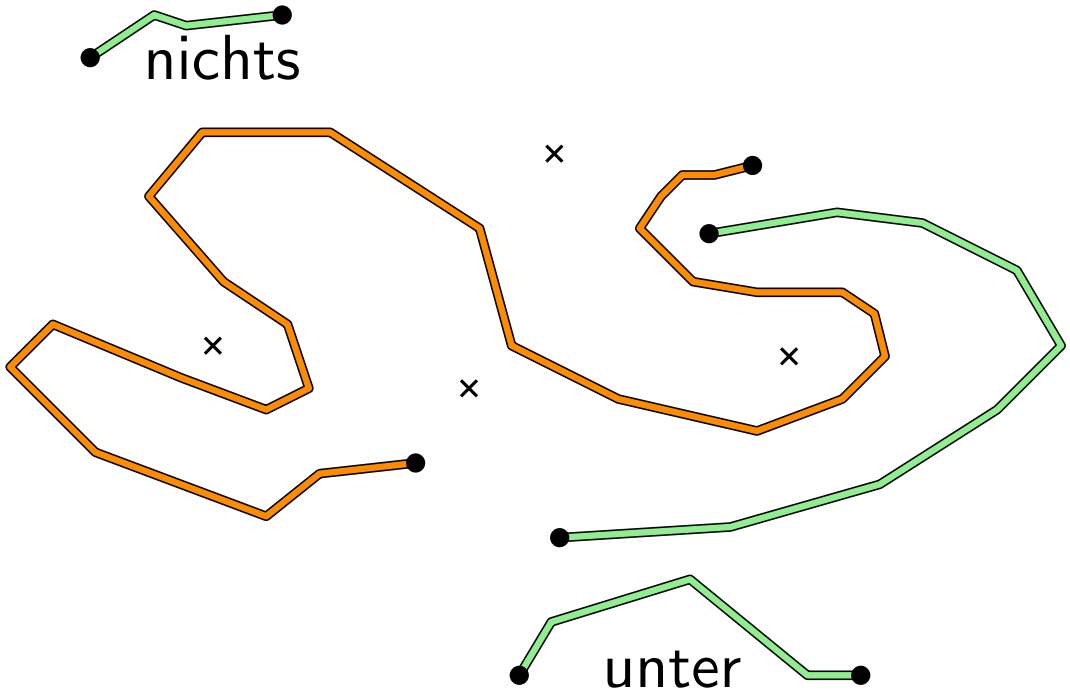
Die Ordnung zweier Pfade a, b ist definiert durch die Lage der Endpunkte des einen Pfades zum anderen Pfad.



Vertikale Pfadordnung

Def: Für Pfad a und Punkt p liegt p **oberhalb/unterhalb** von a, falls für **jeden** zu a äquivalenten Pfad a' in $\mathbb{R}^2 \setminus \{p\}$ der entsprechende vertikale Strahl von p Pfad a' schneidet.

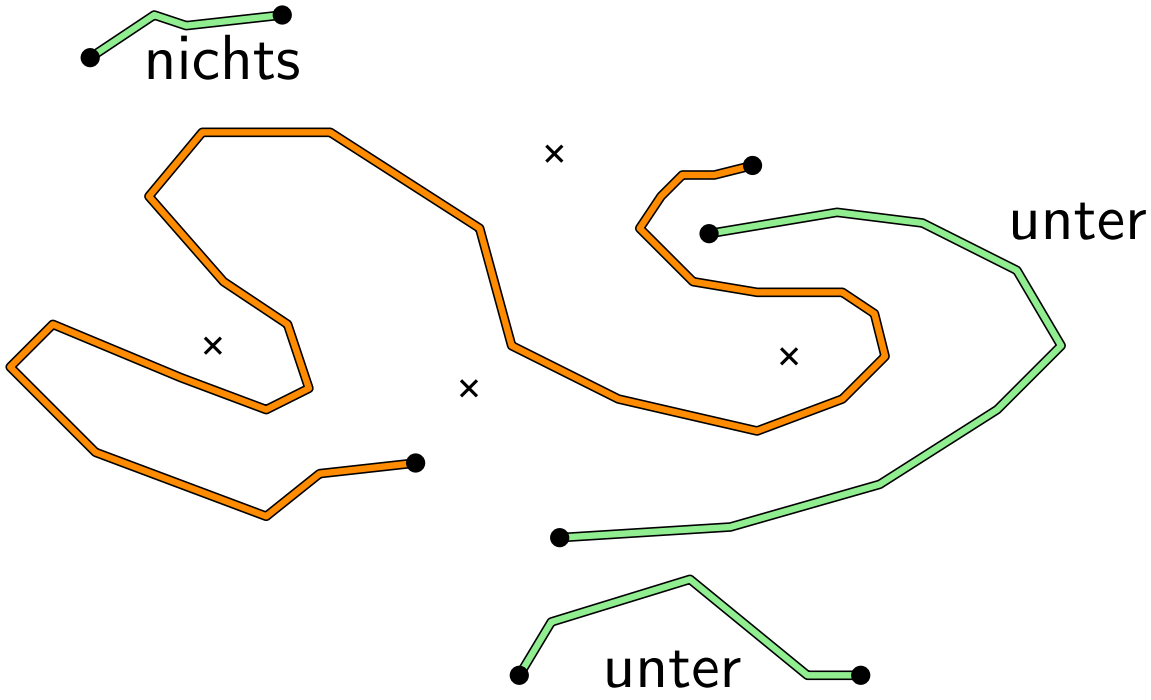
Die Ordnung zweier Pfade a, b ist definiert durch die Lage der Endpunkte des einen Pfades zum anderen Pfad.



Vertikale Pfadordnung

Def: Für Pfad a und Punkt p liegt p **oberhalb/unterhalb** von a, falls für **jeden** zu a äquivalenten Pfad a' in $\mathbb{R}^2 \setminus \{p\}$ der entsprechende vertikale Strahl von p Pfad a' schneidet.

Die Ordnung zweier Pfade a, b ist definiert durch die Lage der Endpunkte des einen Pfades zum anderen Pfad.

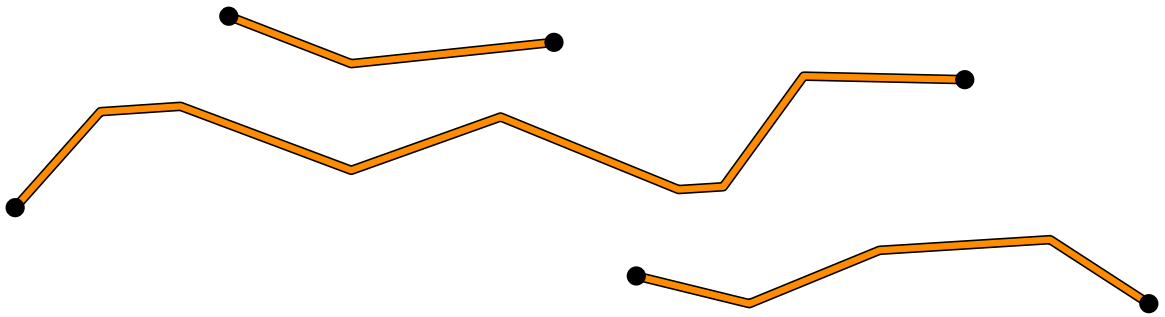


Vertikale Pfadordnung

Def: Für Pfad a und Punkt p liegt p **oberhalb/unterhalb** von a, falls für **jeden** zu a äquivalenten Pfad a' in $\mathbb{R}^2 \setminus \{p\}$ der entsprechende vertikale Strahl von p Pfad a' schneidet.

Die Ordnung zweier Pfade a, b ist definiert durch die Lage der Endpunkte des einen Pfades zum anderen Pfad.

Bem: Für x-monotone Pfade ist die Definition äquivalent zur Definition a **oberhalb** b $(a \succ b)$ gdw. es Punkte $(x, y_a) \in a$ und $(x, y_b) \in b$ gibt mit $y_a > y_b$.



Ordnung in monotonen Karten

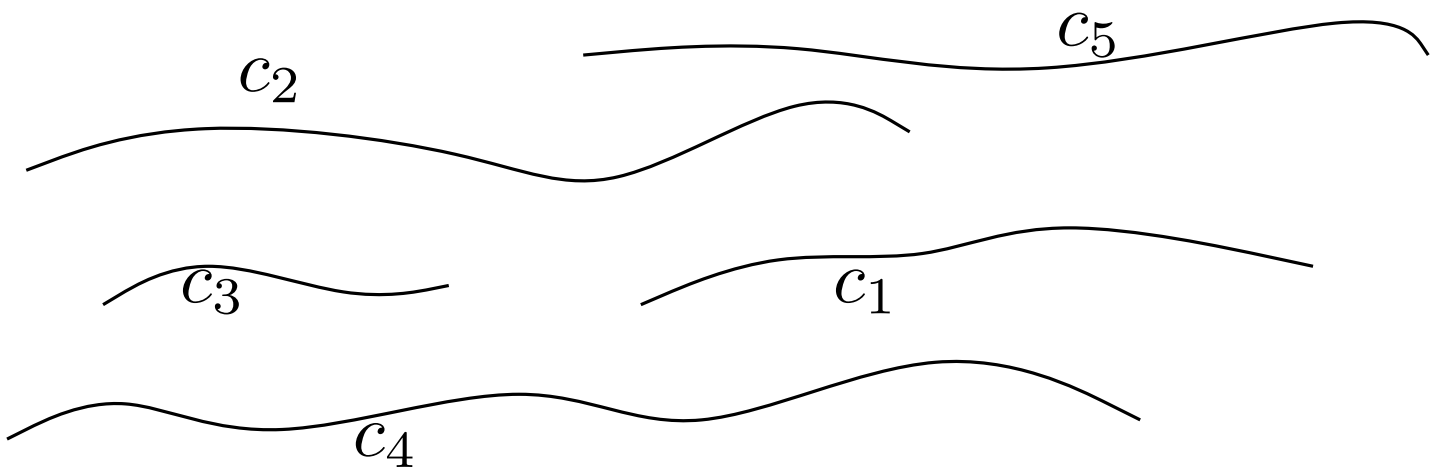
Karte M ist **monoton**, falls jeder Pfad in M x-monoton ist.

Lemma 1: Für monotone Karte M ist die Vertikalordnung $\succ$ azyklisch; eine Totalordnung, die $\succ$ erweitert, kann in $O(n \log n)$ Zeit bestimmt werden, wobei n die Knotenzahl in M ist.

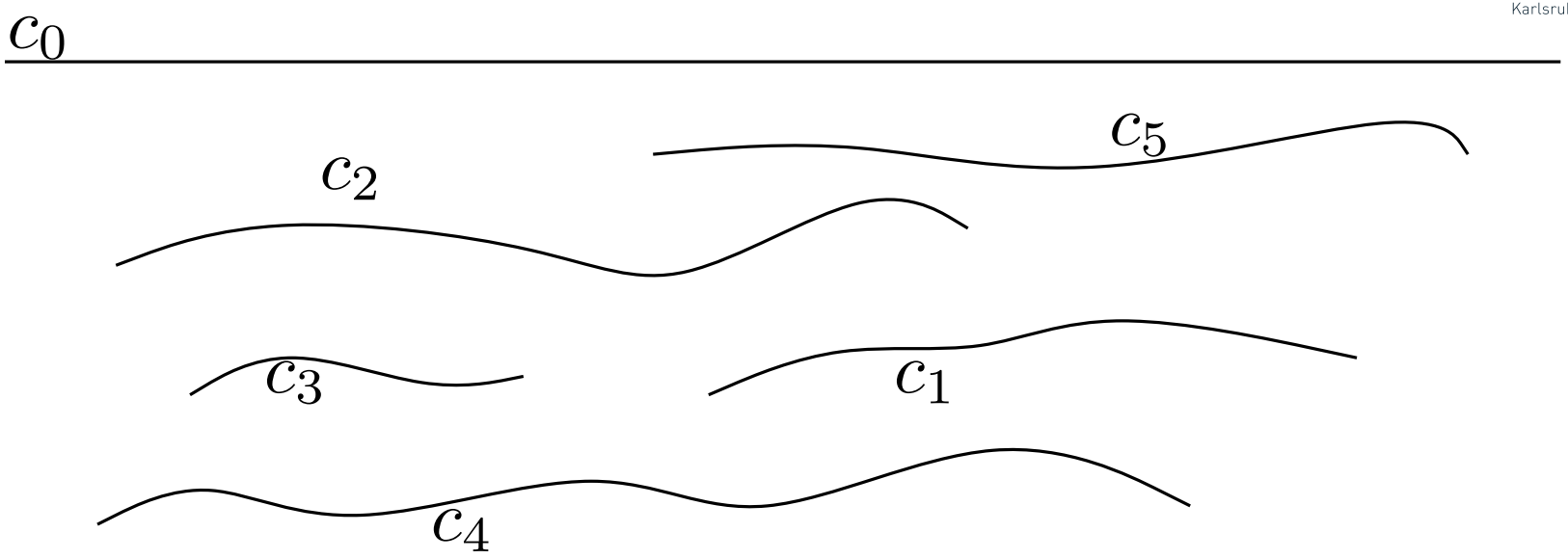
Beweis:

1. $\succ$ ist azyklisch: siehe Tafel.
2. $\succ$ kann in $O(n \log n)$ zu Totalordnung erweitert werden.

Erweiterung zur totalen Ordnung

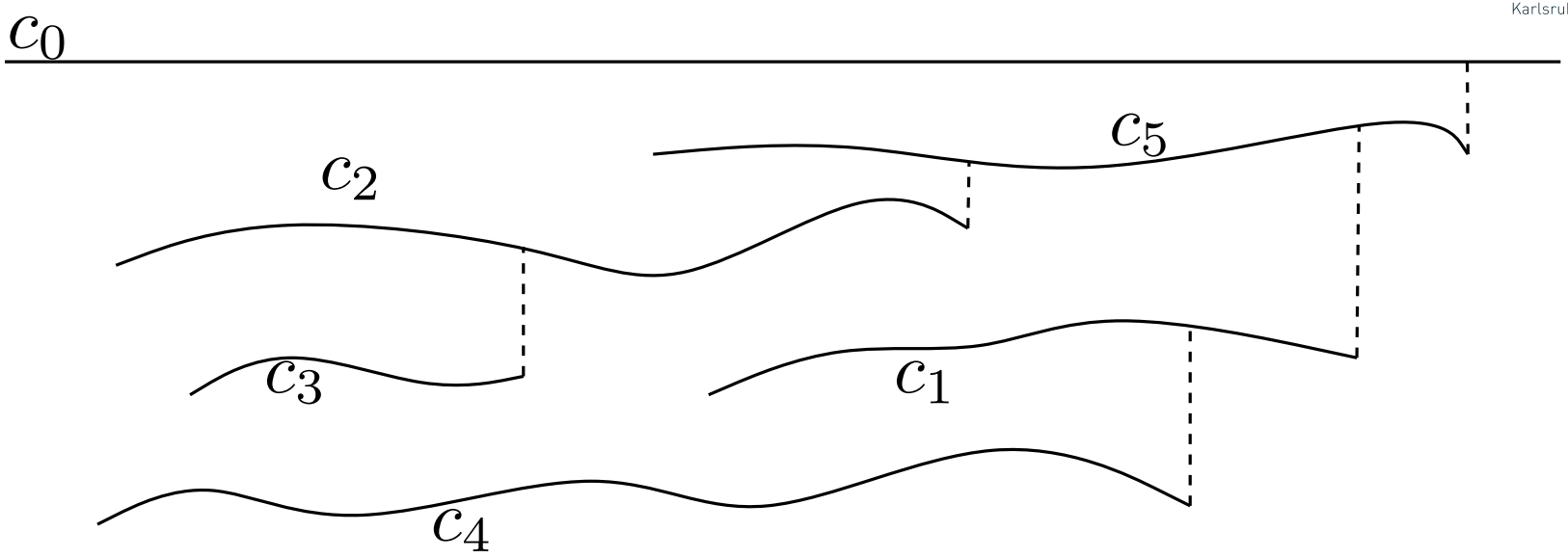


Erweiterung zur totalen Ordnung



1. Füge Stecke mit Endpunkten $(-\infty, \infty)$ und (∞, ∞) ein.

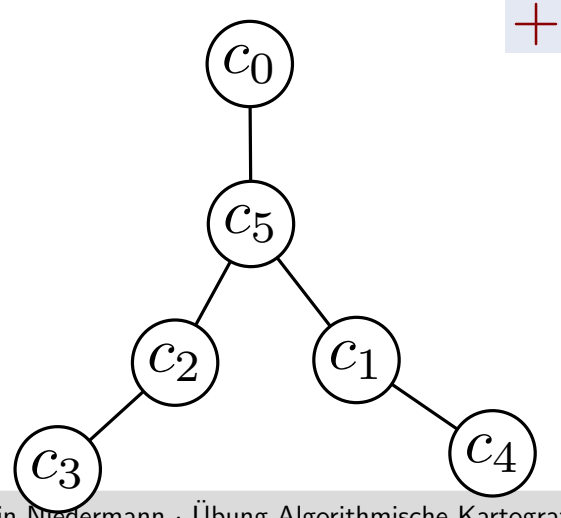
Erweiterung zur totalen Ordnung



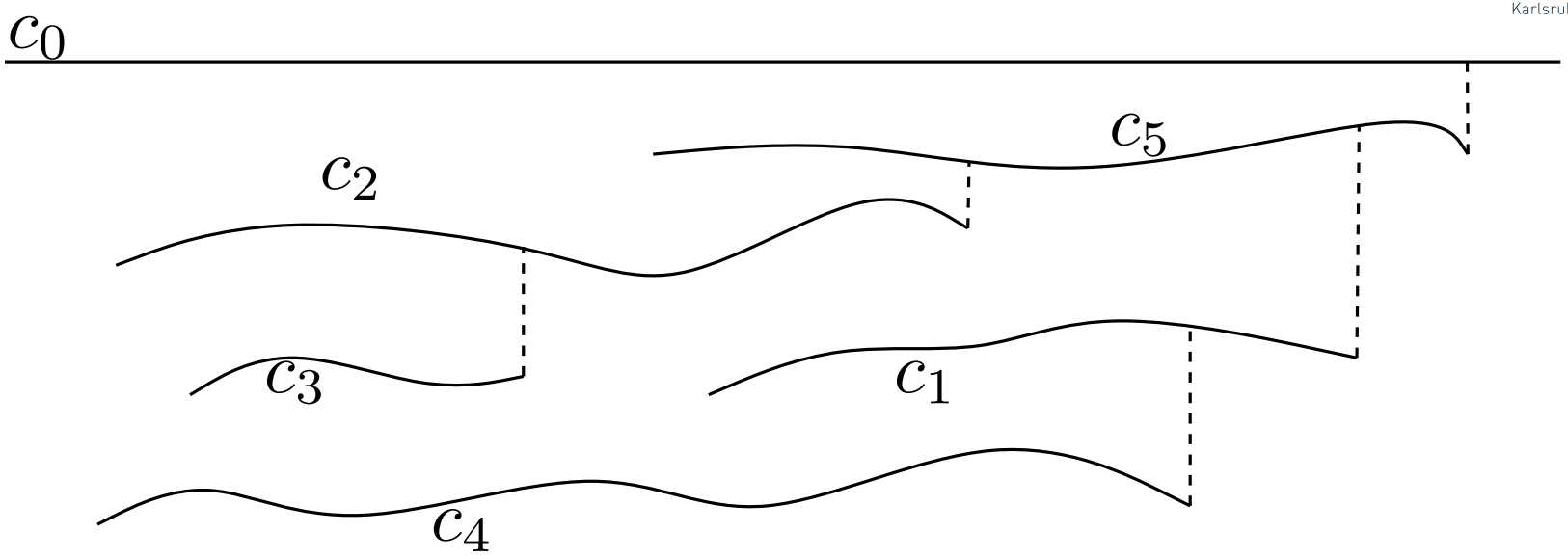
1. Füge Stecke mit Endpunkten $(-\infty, \infty)$ und (∞, ∞) ein.
2. Erstelle Baum T mit Knoten $c_0, \ldots, c_n$.

   c_i ist Kind von c_j gdw. c_j liegt direkt über rechtem Endpunkt von c_i.

   + halte horizontale Ordnung der Eckpunkte ein.



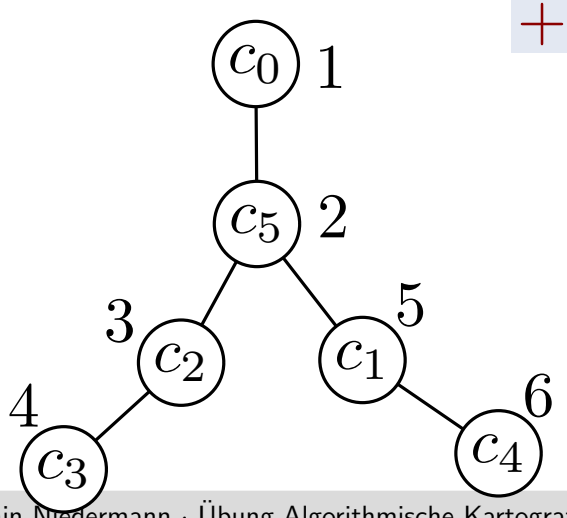
Erweiterung zur totalen Ordnung



1. Füge Stecke mit Endpunkten $(-\infty, \infty)$ und (∞, ∞) ein.
2. Erstelle Baum T mit Knoten $c_0, \ldots, c_n$.

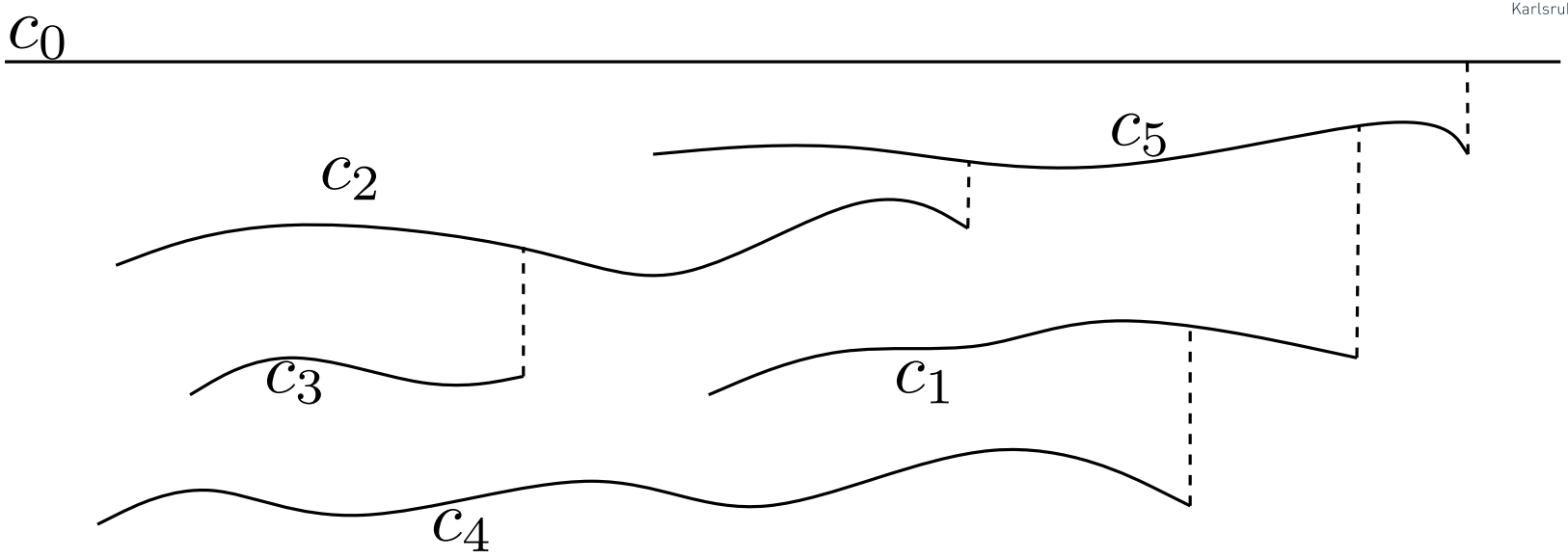
   c_i ist Kind von c_j gdw. c_j liegt direkt über rechtem Endpunkt von c_i.

   + halte horizontale Ordnung der Eckpunkte ein.



3. Traversiere Baum *inorder* und sortiere Knoten nach erstem Auftreten.

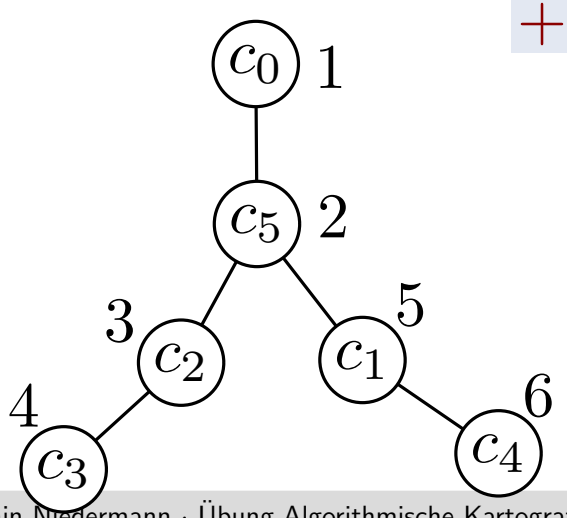
Erweiterung zur totalen Ordnung



1. Füge Stecke mit Endpunkten $(-\infty, \infty)$ und (∞, ∞) ein.
2. Erstelle Baum T mit Knoten $c_0, \ldots, c_n$.

   c_i ist Kind von c_j gdw. c_j liegt direkt über rechtem Endpunkt von c_i.

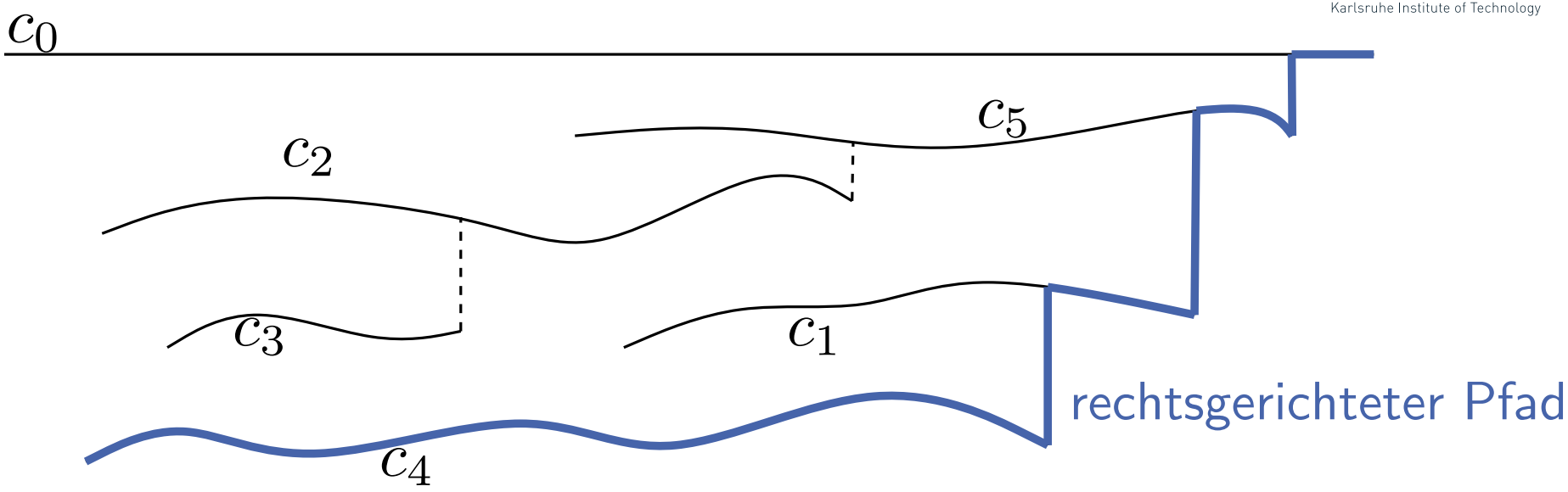
   + halte horizontale Ordnung der Eckpunkte ein.



3. Traversiere Baum *inorder* und sortiere Knoten nach erstem Auftreten.

Warum bleibt $\succ$ erhalten?

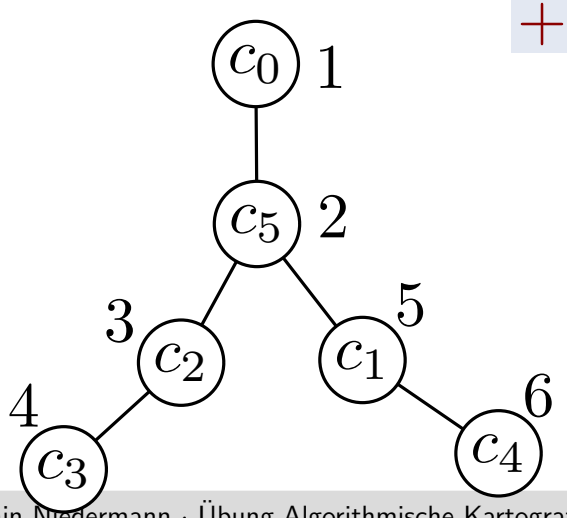
Erweiterung zur totalen Ordnung



1. Füge Stecke mit Endpunkten $(-\infty, \infty)$ und (∞, ∞) ein.
2. Erstelle Baum T mit Knoten $c_0, \ldots, c_n$.

   c_i ist Kind von c_j gdw. c_j liegt direkt über rechtem Endpunkt von c_i.

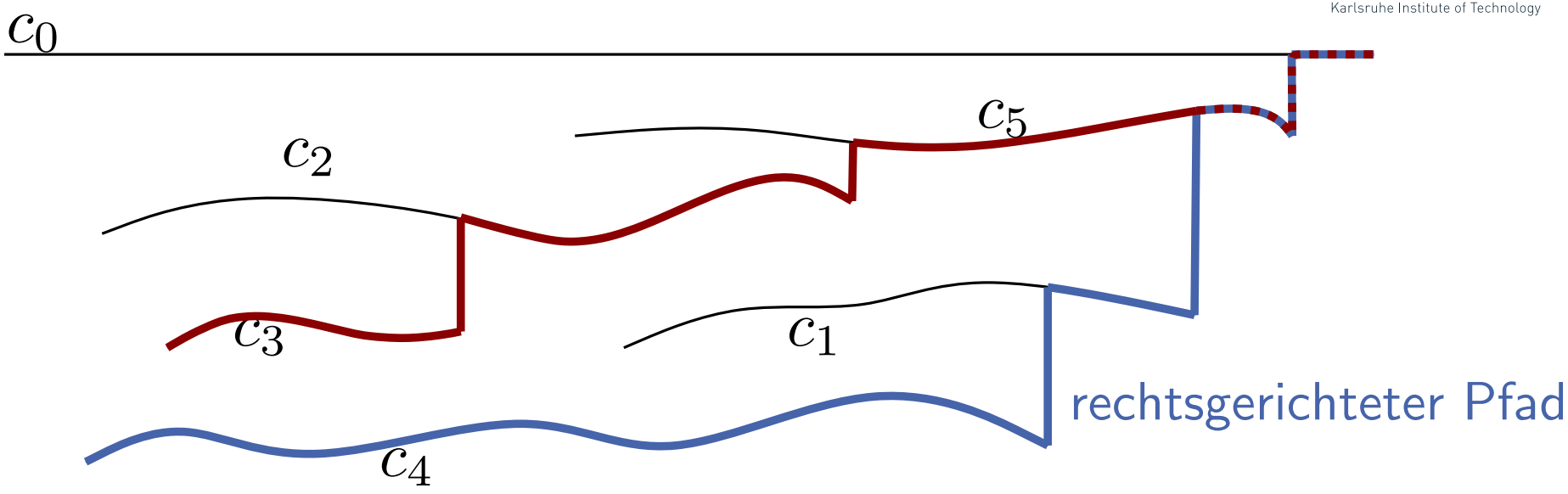
   + halte horizontale Ordnung der Eckpunkte ein.



3. Traversiere Baum *inorder* und sortiere Knoten nach erstem Auftreten.

Warum bleibt $\succ$ erhalten?

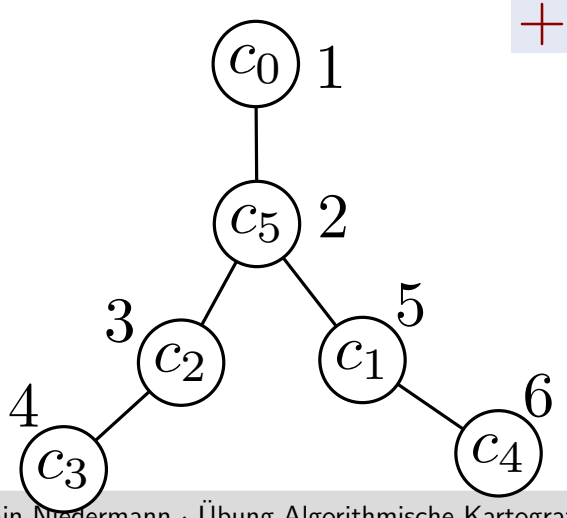
Erweiterung zur totalen Ordnung



1. Füge Stecke mit Endpunkten $(-\infty, \infty)$ und (∞, ∞) ein.
2. Erstelle Baum T mit Knoten $c_0, \ldots, c_n$.

   c_i ist Kind von c_j gdw. c_j liegt direkt über rechtem Endpunkt von c_i.

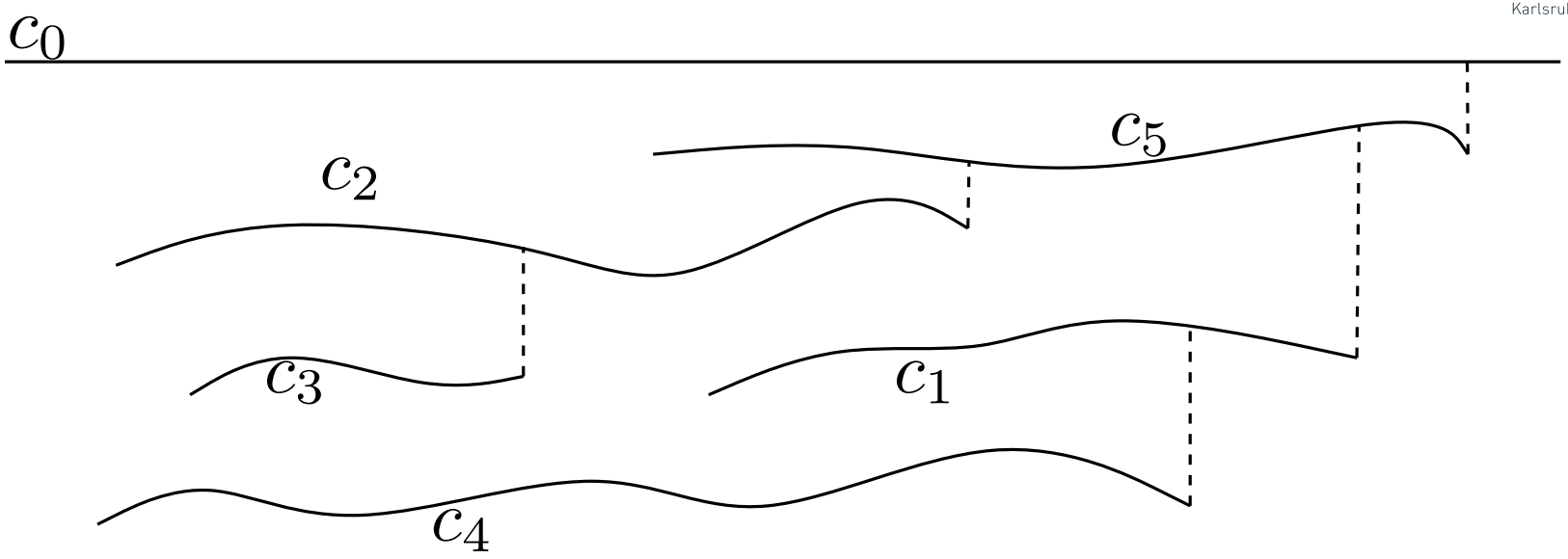
   + halte horizontale Ordnung der Eckpunkte ein.



3. Traversiere Baum *inorder* und sortiere Knoten nach erstem Auftreten.

Warum bleibt $\succ$ erhalten?

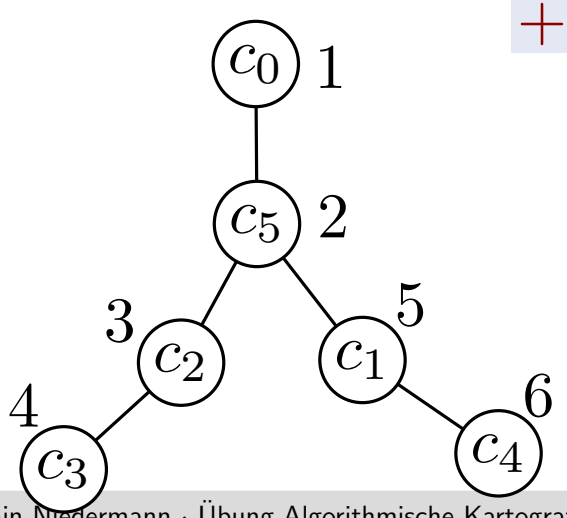
Erweiterung zur totalen Ordnung



1. Füge Stecke mit Endpunkten $(-\infty, \infty)$ und (∞, ∞) ein.
2. Erstelle Baum T mit Knoten $c_0, \ldots, c_n$.

   c_i ist Kind von c_j gdw. c_j liegt direkt über rechtem Endpunkt von c_i.

   + halte horizontale Ordnung der Eckpunkte ein.

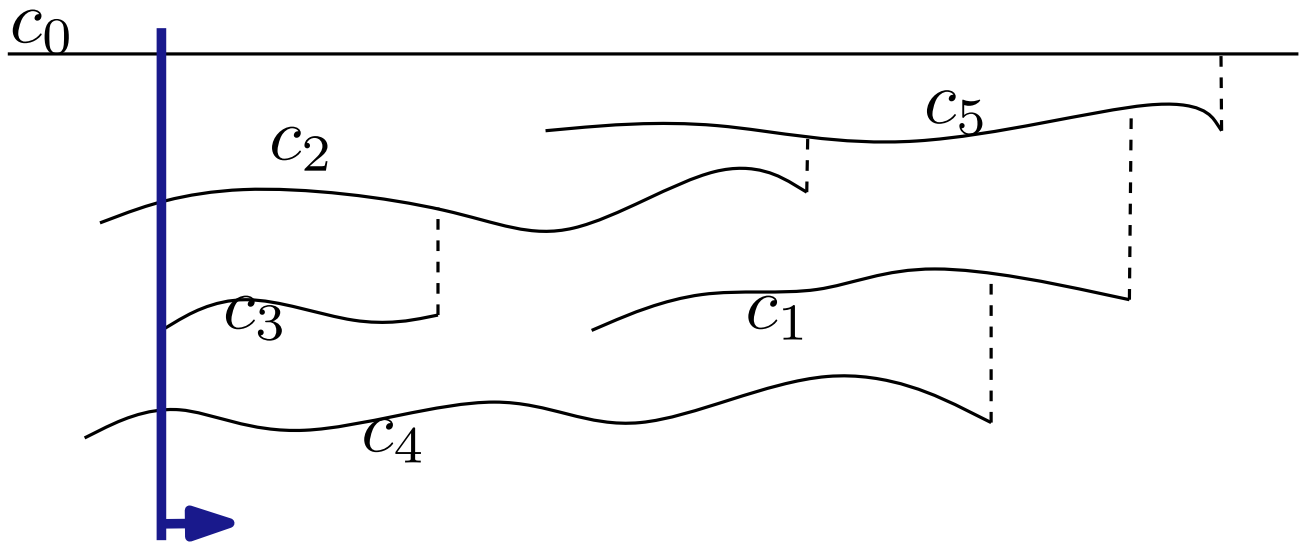


3. Traversiere Baum *inorder* und sortiere Knoten nach erstem Auftreten.

Warum bleibt $\succ$ erhalten?

Wie schnell T berechnen?

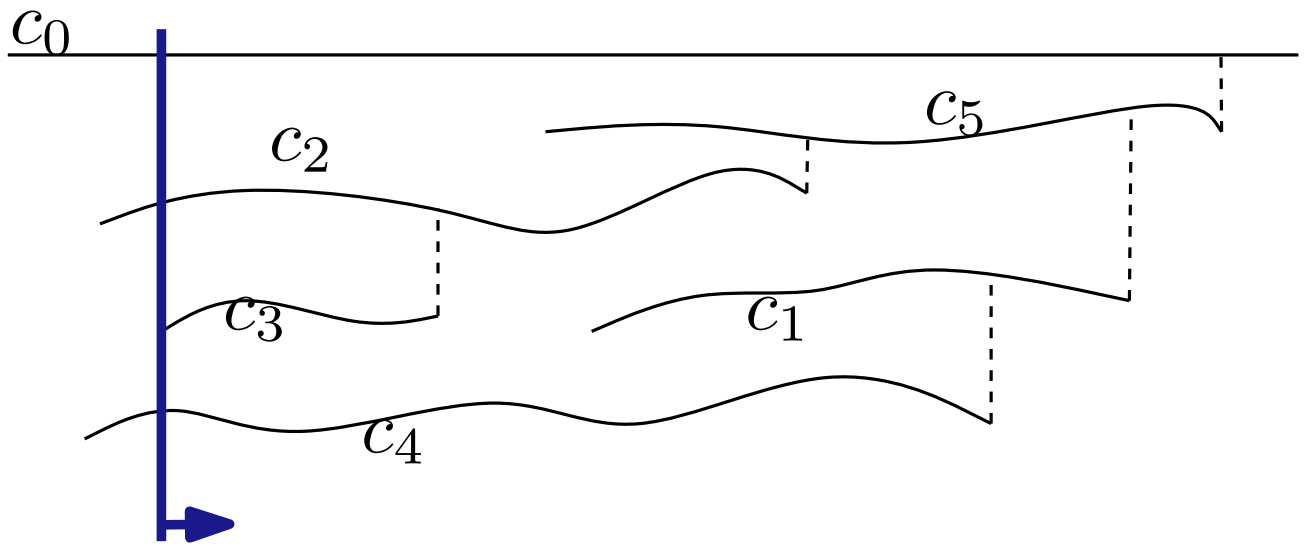
Berechnung von T



Vertikale Sweep-Line von links nach rechts:

Events: Endpunkte der Kurven.

Berechnung von T

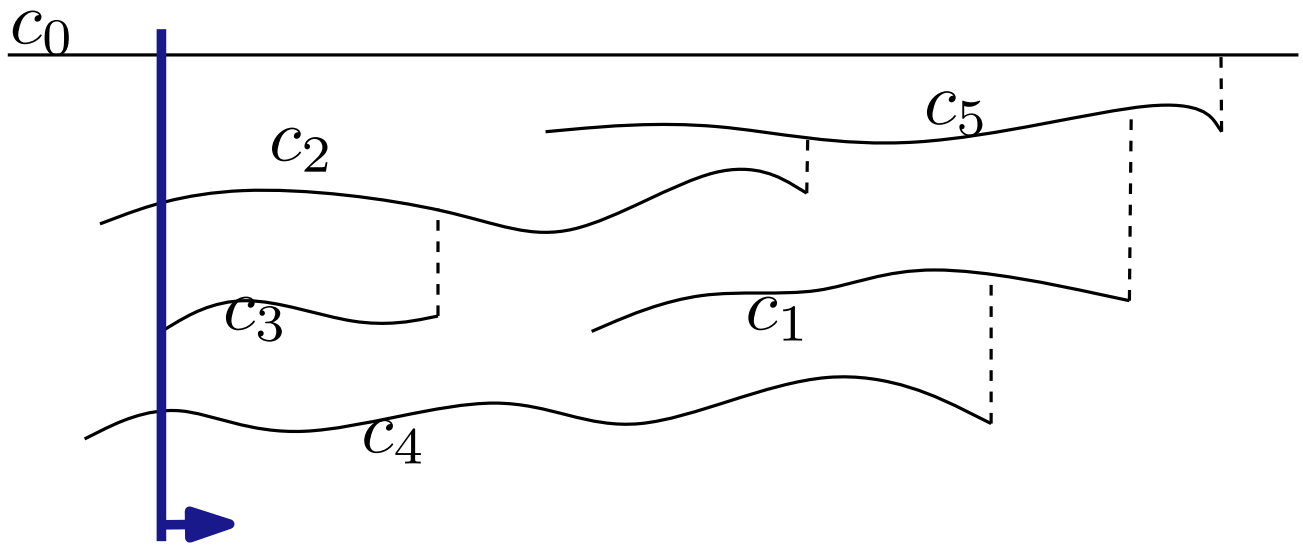


Vertikale Sweep-Line von links nach rechts:

Events: Endpunkte der Kurven.

Sweep-Line-Zustand: Kurven die von Sweep-Line geschnitten werden. Repräsentation als Binärbaum.

Berechnung von T



Vertikale Sweep-Line von links nach rechts:

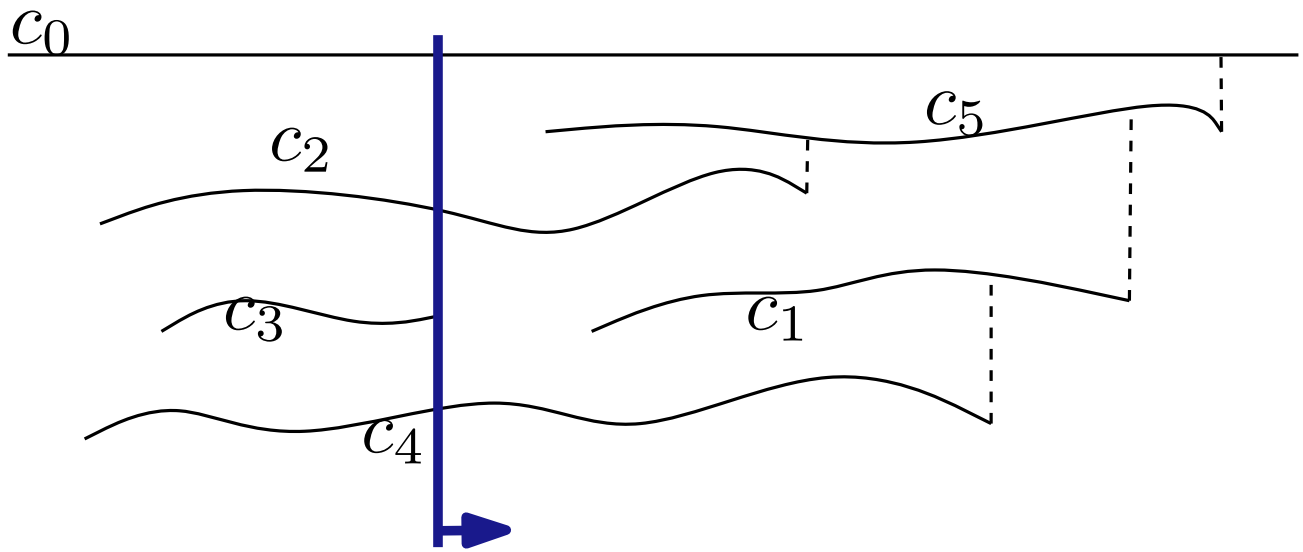
Events: Endpunkte der Kurven.

Sweep-Line-Zustand: Kurven die von Sweep-Line geschnitten werden. Repräsentation als Binärbaum.

Abarbeitung der Events p:

p ist linker Endpunkt einer Kurve c_i: c_i wird in S einsortiert.

Berechnung von T



Vertikale Sweep-Line von links nach rechts:

Events: Endpunkte der Kurven.

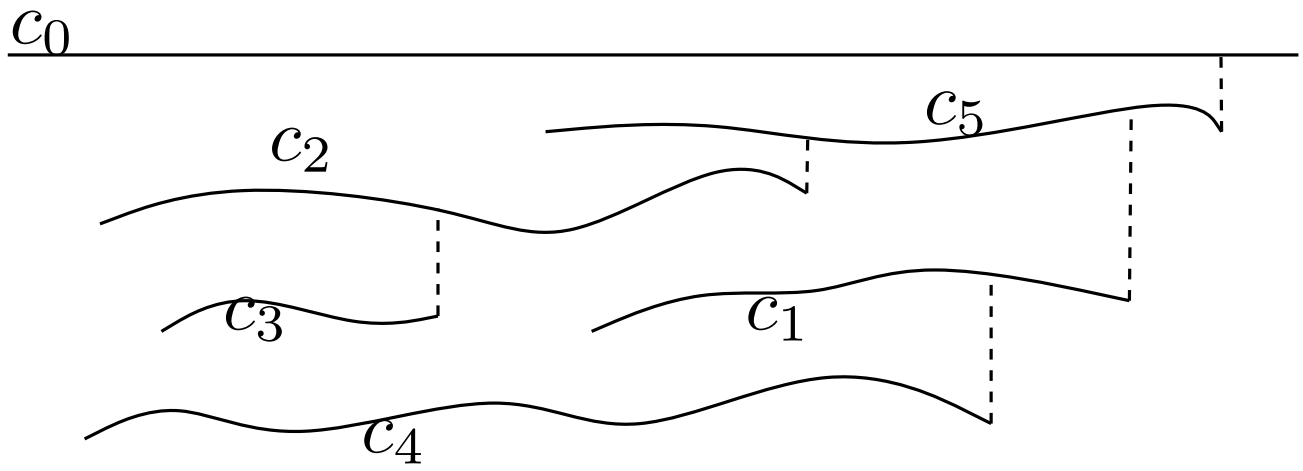
Sweep-Line-Zustand: Kurven die von Sweep-Line geschnitten werden. Repräsentation als Binärbaum.

Abarbeitung der Events p:

p ist linker Endpunkt einer Kurve c_i: c_i wird in S einsortiert.

p ist rechter Endpunkt einer Kurve c_i: Entferne c_i aus S und setze Kurve, die Sweep-Line direkt über c_i schneidet als Vater von c_i.

Berechnung von T



Laufzeit: $O(n \log n)$

Vertikale Sweep-Line von links nach rechts:

Events: Endpunkte der Kurven.

Sweep-Line-Zustand: Kurven die von Sweep-Line geschnitten werden. Repräsentation als Binärbaum.

Abarbeitung der Events p:

p ist linker Endpunkt einer Kurve c_i: c_i wird in S einsortiert.

p ist rechter Endpunkt einer Kurve c_i: Entferne c_i aus S und setze Kurve, die Sweep-Line direkt über c_i schneidet als Vater von c_i.

Ordnung in monotonen Karten

Karte M ist **monoton**, falls jeder Pfad in M x-monoton ist.

Lemma 1: Für monotone Karte M ist die Vertikalordnung $\succ$ azyklisch; eine Totalordnung, die $\succ$ erweitert, kann in $O(n \log n)$ Zeit bestimmt werden, wobei n die Knotenzahl in M ist.

Ordnung in monotonen Karten

Karte M ist **monoton**, falls jeder Pfad in M x-monoton ist.

Lemma 1: Für monotone Karte M ist die Vertikalordnung $\succ$ azyklisch; eine Totalordnung, die $\succ$ erweitert, kann in $O(n \log n)$ Zeit bestimmt werden, wobei n die Knotenzahl in M ist.

Lemma 2: Zwei monotone Karten M und M' sind äquivalent gdw. sie die gleiche Vertikalordnung $\succ$ definieren.

Ordnung in monotonen Karten

Karte M ist **monoton**, falls jeder Pfad in M x-monoton ist.

Lemma 1: Für monotone Karte M ist die Vertikalordnung $\succ$ azyklisch; eine Totalordnung, die $\succ$ erweitert, kann in $O(n \log n)$ Zeit bestimmt werden, wobei n die Knotenzahl in M ist.

Lemma 2: Zwei monotone Karten M und M' sind äquivalent gdw. sie die gleiche Vertikalordnung $\succ$ definieren.

Gilt Lemma 2 auch für nicht monotone Karten?

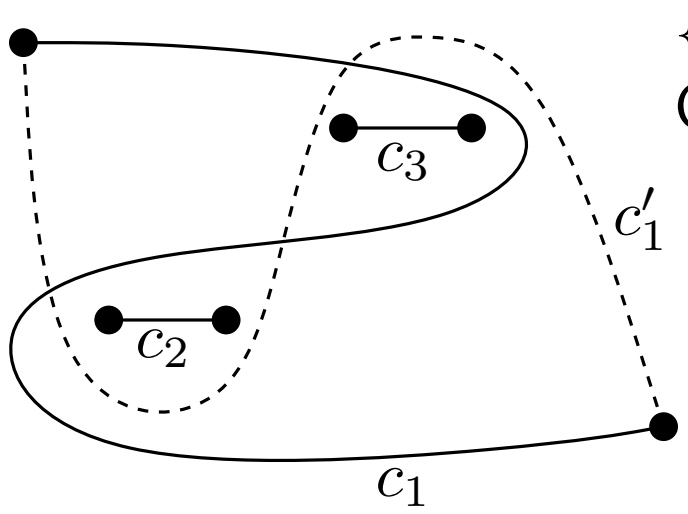
Ordnung in monotonen Karten

Karte M ist **monoton**, falls jeder Pfad in M x-monoton ist.

Lemma 1: Für monotone Karte M ist die Vertikalordnung $\succ$ azyklisch; eine Totalordnung, die $\succ$ erweitert, kann in $O(n \log n)$ Zeit bestimmt werden, wobei n die Knotenzahl in M ist.

Lemma 2: Zwei monotone Karten M und M' sind äquivalent gdw. sie die gleiche Vertikalordnung $\succ$ definieren.

Gilt Lemma 2 auch für nicht monotone Karten?



$\{c_1, c_2, c_3\}$ und $\{c_1', c_2, c_3\}$ bilden gleiche Oberhalb-Ordnung, sind aber nicht äquivalent.

Skizze des Schematisierungs-Algorithmus

[Cabello, de Berg, van Kreveld '05]

Input: Karte M

Output: schematisierte äquivalente Karte M' (falls existiert)

1. Zerlege nicht-monotone Pfade in monotone Teile
2. Erstelle rektifizierte Karte R mit achsenparallelen Pfaden
3. Wandle R in kanonische Form R_c um
4. Berechne Totalordnung $>$ der Pfade aus R_c
5. Konstruiere inkrementell schematische Karte M' durch höchstmögliches Hinzufügen von Pfaden entsprechend $>$

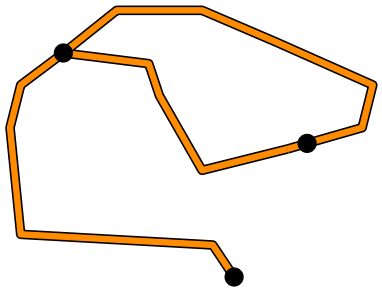
Skizze des Schematisierungs-Algorithmus

[Cabello, de Berg, van Kreveld '05]

Input: Karte M

Output: schematisierte äquivalente Karte M' (falls existiert)

1. Zerlege nicht-monotone Pfade in monotone Teile
2. Erstelle rektifizierte Karte R mit achsenparallelen Pfaden
3. Wandle R in kanonische Form R_c um
4. Berechne Totalordnung $>$ der Pfade aus R_c
5. Konstruiere inkrementell schematische Karte M' durch höchstmögliches Hinzufügen von Pfaden entsprechend $>$



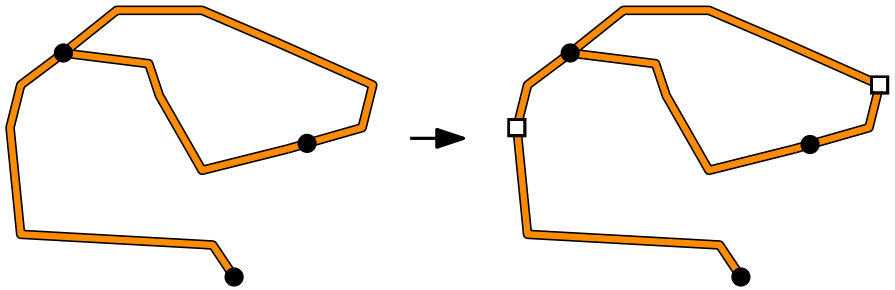
Skizze des Schematisierungs-Algorithmus

[Cabello, de Berg, van Kreveld '05]

Input: Karte M

Output: schematisierte äquivalente Karte M' (falls existiert)

1. Zerlege nicht-monotone Pfade in monotone Teile
2. Erstelle rektifizierte Karte R mit achsenparallelen Pfaden
3. Wandle R in kanonische Form R_c um
4. Berechne Totalordnung $>$ der Pfade aus R_c
5. Konstruiere inkrementell schematische Karte M' durch höchstmögliches Hinzufügen von Pfaden entsprechend $>$



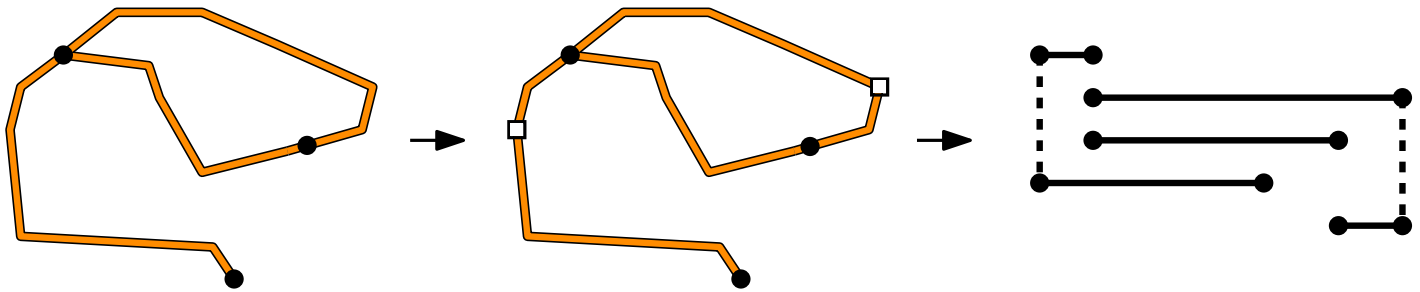
Skizze des Schematisierungs-Algorithmus

[Cabello, de Berg, van Kreveld '05]

Input: Karte M

Output: schematisierte äquivalente Karte M' (falls existiert)

1. Zerlege nicht-monotone Pfade in monotone Teile
2. Erstelle rektifizierte Karte R mit achsenparallelen Pfaden
3. Wandle R in kanonische Form R_c um
4. Berechne Totalordnung $>$ der Pfade aus R_c
5. Konstruiere inkrementell schematische Karte M' durch höchstmögliches Hinzufügen von Pfaden entsprechend $>$



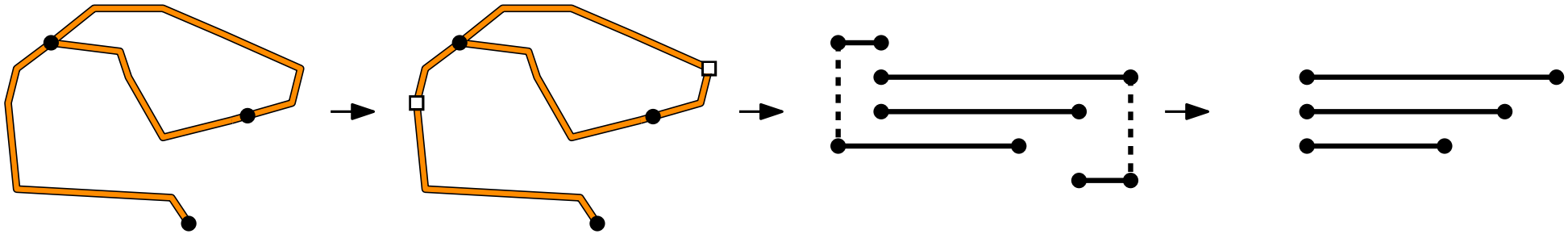
Skizze des Schematisierungs-Algorithmus

[Cabello, de Berg, van Kreveld '05]

Input: Karte M

Output: schematisierte äquivalente Karte M' (falls existiert)

1. Zerlege nicht-monotone Pfade in monotone Teile
2. Erstelle rektifizierte Karte R mit achsenparallelen Pfaden
3. Wandle R in kanonische Form R_c um
4. Berechne Totalordnung $>$ der Pfade aus R_c
5. Konstruiere inkrementell schematische Karte M' durch höchstmögliches Hinzufügen von Pfaden entsprechend $>$



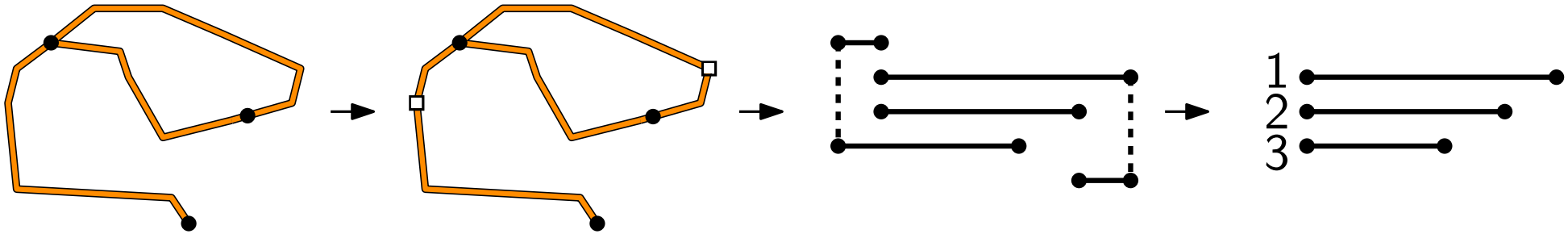
Skizze des Schematisierungs-Algorithmus

[Cabello, de Berg, van Kreveld '05]

Input: Karte M

Output: schematisierte äquivalente Karte M' (falls existiert)

1. Zerlege nicht-monotone Pfade in monotone Teile
2. Erstelle rektifizierte Karte R mit achsenparallelen Pfaden
3. Wandle R in kanonische Form R_c um
4. Berechne Totalordnung $>$ der Pfade aus R_c
5. Konstruiere inkrementell schematische Karte M' durch höchstmögliches Hinzufügen von Pfaden entsprechend $>$



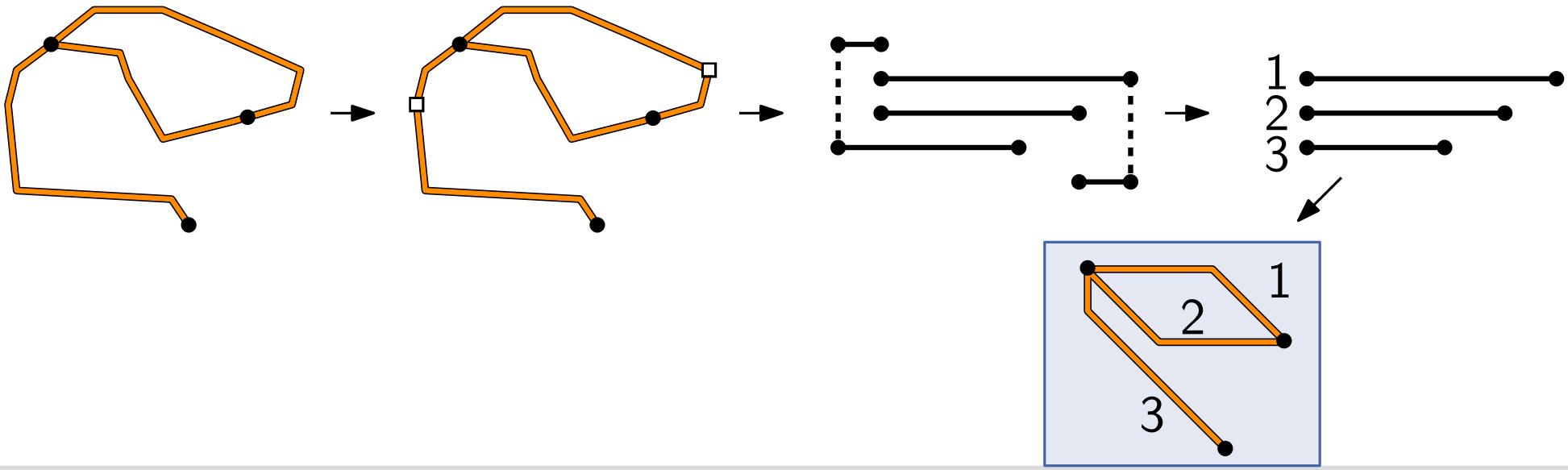
Skizze des Schematisierungs-Algorithmus

[Cabello, de Berg, van Kreveld '05]

Input: Karte M

Output: schematisierte äquivalente Karte M' (falls existiert)

1. Zerlege nicht-monotone Pfade in monotone Teile
2. Erstelle rektifizierte Karte R mit achsenparallelen Pfaden
3. Wandle R in kanonische Form R_c um
4. Berechne Totalordnung $>$ der Pfade aus R_c
5. Konstruiere inkrementell schematische Karte M' durch höchstmögliches Hinzufügen von Pfaden entsprechend $>$



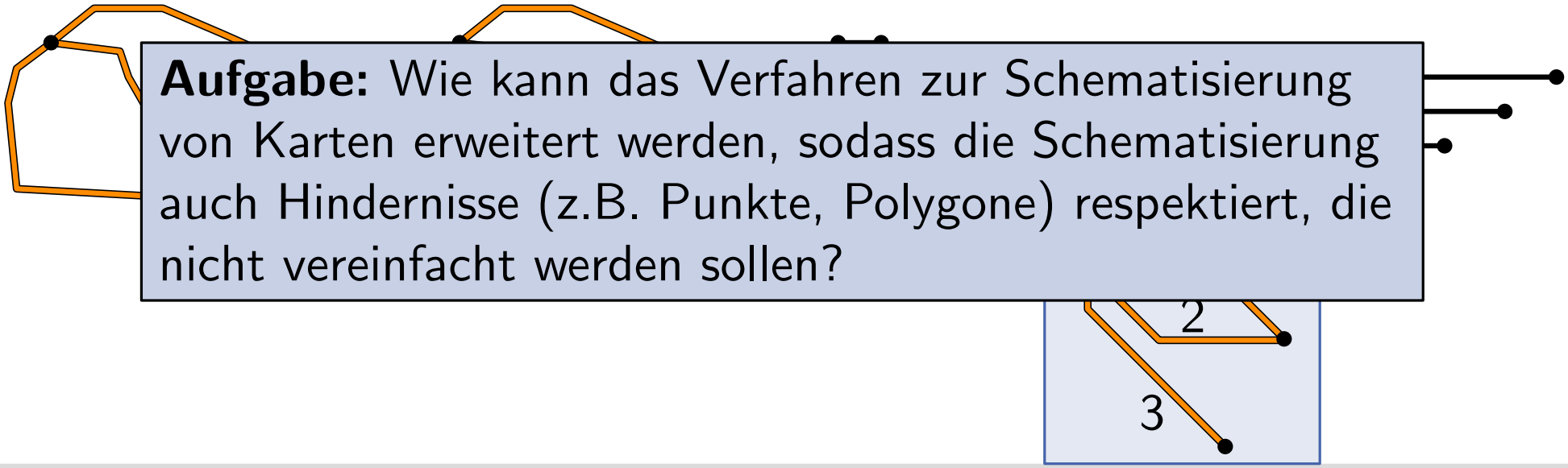
Skizze des Schematisierungs-Algorithmus

[Cabello, de Berg, van Kreveld '05]

Input: Karte M

Output: schematisierte äquivalente Karte M' (falls existiert)

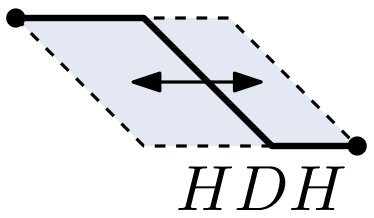
1. Zerlege nicht-monotone Pfade in monotone Teile
2. Erstelle rektifizierte Karte R mit achsenparallelen Pfaden
3. Wandle R in kanonische Form R_c um
4. Berechne Totalordnung $>$ der Pfade aus R_c
5. Konstruiere inkrementell schematische Karte M' durch höchstmögliches Hinzufügen von Pfaden entsprechend $>$

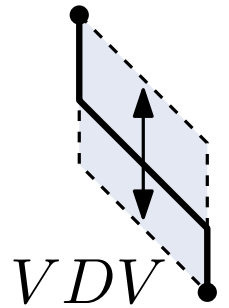


Aufgabe: Wie kann das Verfahren zur Schematisierung von Karten erweitert werden, sodass die Schematisierung auch Hindernisse (z.B. Punkte, Polygone) respektiert, die nicht vereinfacht werden sollen?

Platzieren von Pfaden

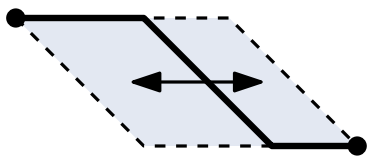
Definiere Typen zulässiger Pfade, z.B. $\{HDH, VDV\}$



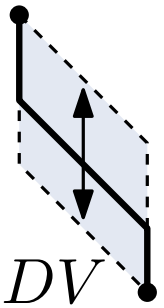


Platzieren von Pfaden

Definiere Typen zulässiger Pfade, z.B. $\{HDH, VDV\}$



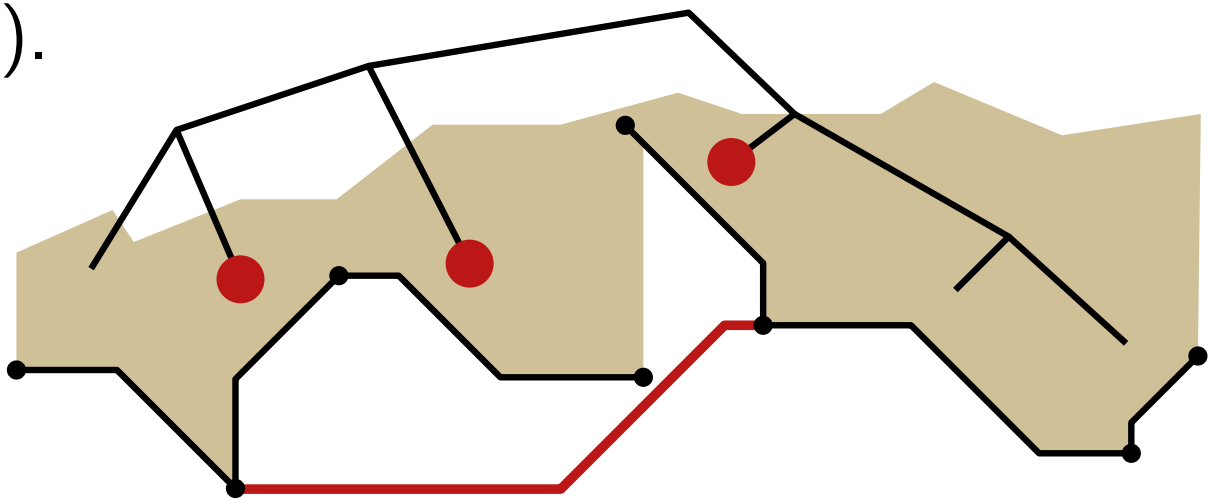
HDH



VDV

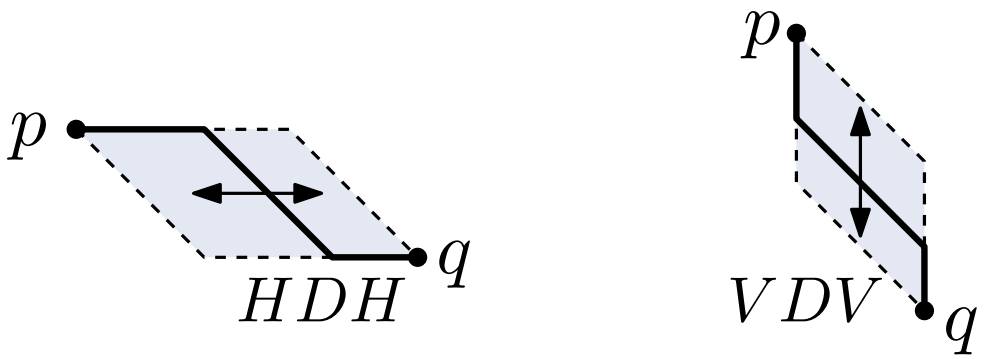
Füge Pfade von M als HDH- oder VDV-Pfade nach der Totalordnung der kanonischen Rektifizierung ein.

Verwalte lower envelope der Pfade als binären Suchbaum und platziere jeden neuen Pfad höchstmöglich (maximaler Platz für den Rest).

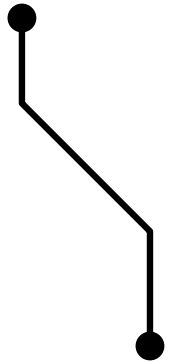


Eindeutige Wahl

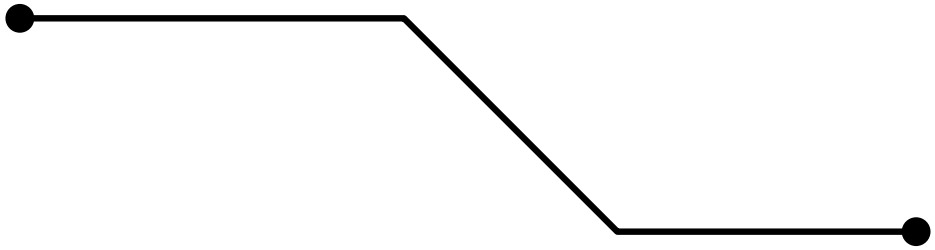
Pfadtyp hängt von Distanz zwischen p und q ab.



Vertikale Distanz $>$ horizontale Distanz: VDV-Pfad

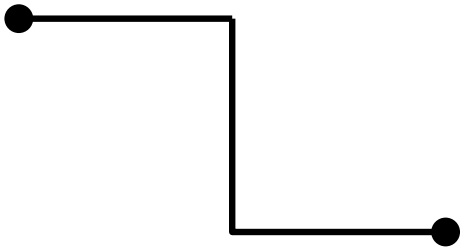


Horizontale Distanz $>$ vertikale Distanz: HDH-Pfad

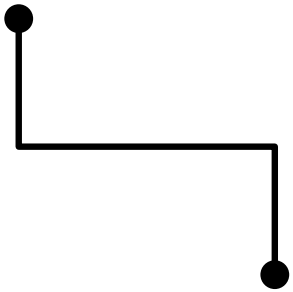


Platzieren von Pfaden

Ist das Verfahren auch für $\{HVH, VHV\}$-Pfade zulässig?



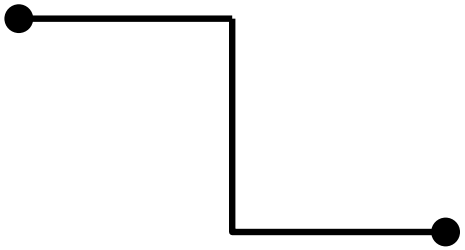
HVH-Pfad



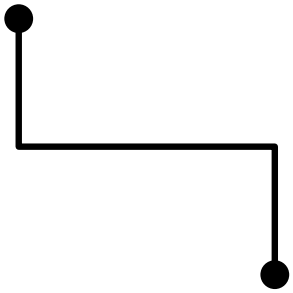
VHV-Pfad

Platzieren von Pfaden

Ist das Verfahren auch für $\{HVH, VHV\}$-Pfade zulässig?

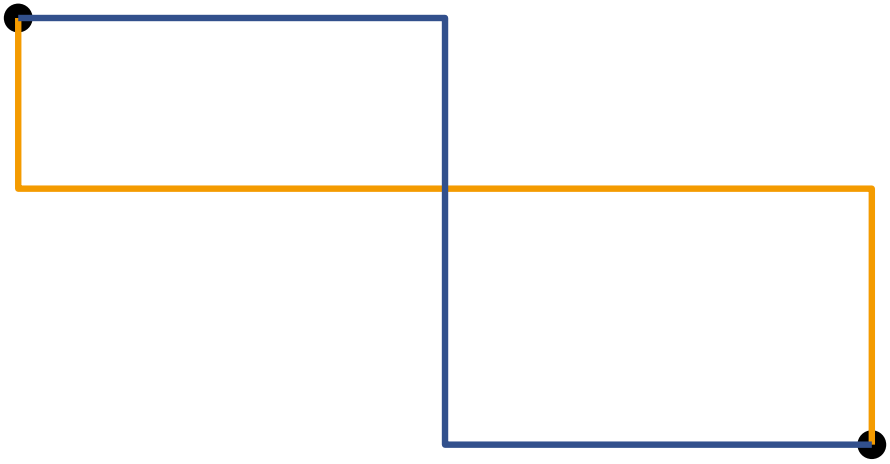


HVH-Pfad



VHV-Pfad

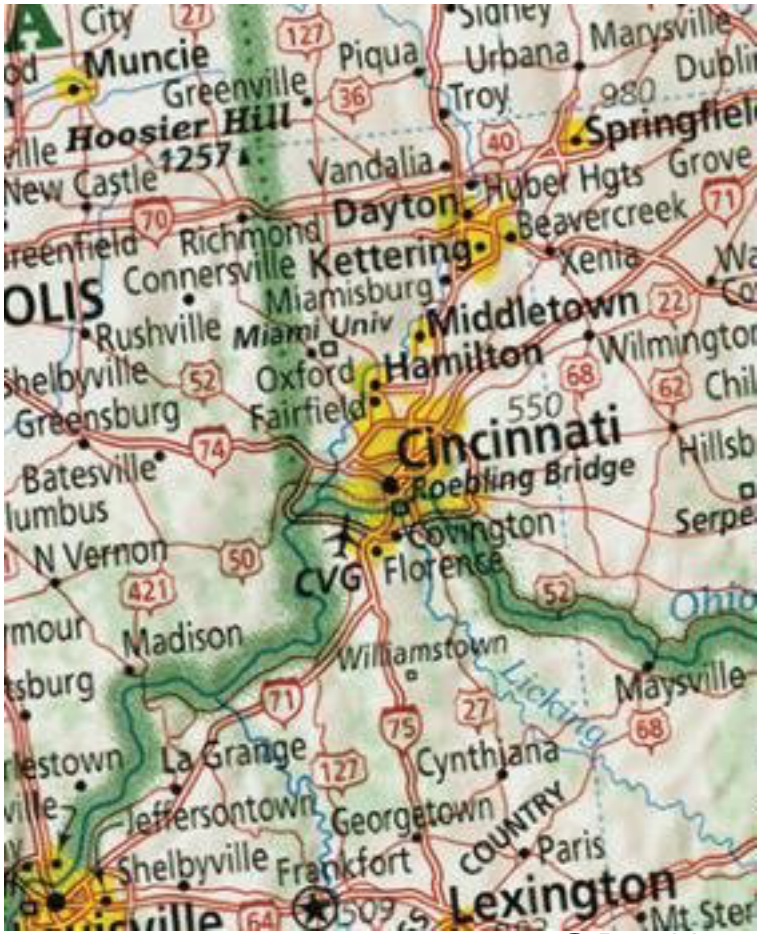
Nein, solche Pfade sind nicht zulässig, da Auswahl nicht eindeutig.



Karlsruhe Institute of Technology

Übungsblatt 3 – Punktbeschriftung

Klassische Kartenbeschriftung



©David Imus

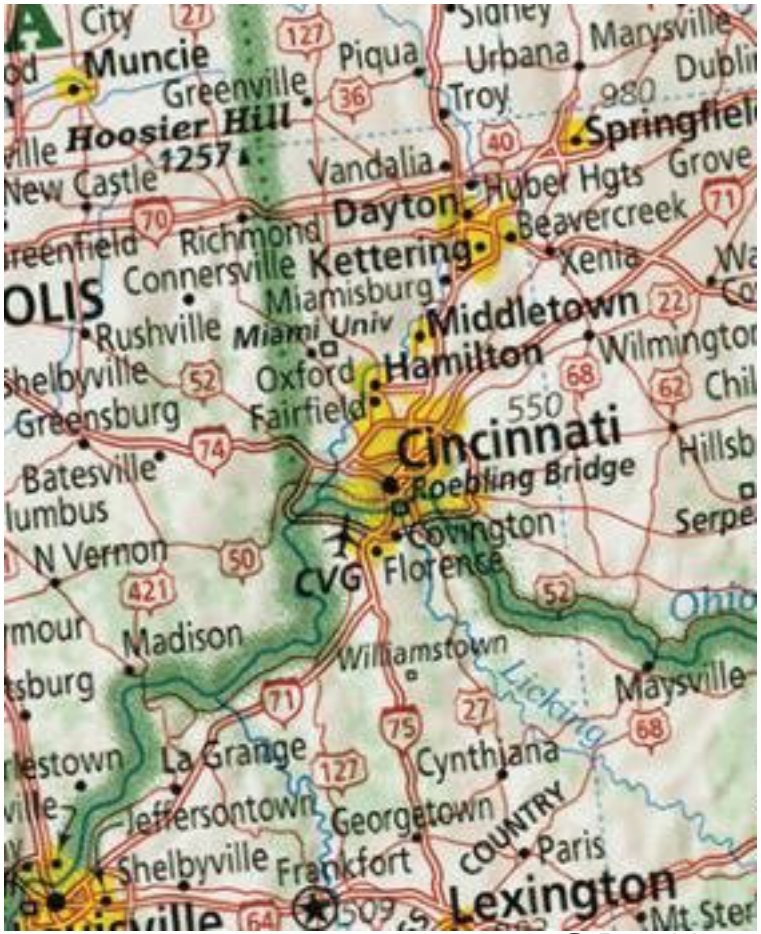
"Poor, sloppy, amateurisch type placement is irresponsible; it spoils even the best image and impedes reading." (E. Imhof '75)

Kartografie hat lange Erfahrung mit manueller Beschriftung in Karten.

einige Richtlinien:

- neben, über oder unter dem Objekt
- am besten oben rechts
- Überlappungen und Überdeckungen vermeiden
- eindeutige grafische Zuordnung
- ...

Klassische Kartenbeschriftung



©David Imus

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

Kartografie hat lange Erfahrung mit manueller Beschriftung in Karten.

einige Richtlinien:

- neben, über oder unter dem Objekt
- am besten oben rechts
- Überlappungen und Überdeckungen vermeiden
- eindeutige grafische Zuordnung
- ...

→ lässt sich leicht in ein Problem der algorithmischen Geometrie zur automatischen Platzierung von Labeln übersetzen

Klassische Kartenbeschriftung

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

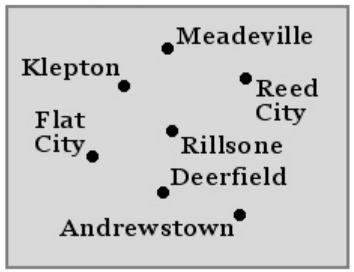
Viele andere Eigenschaften tragen zur Beschriftungsqualität bei:

- Font, Farbe
- **fett**/*kursiv* etc.
- Größe, Zeichenabstand

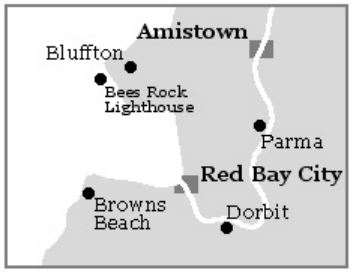
Hier: nur geometrische Platzierung

Poor type placement:

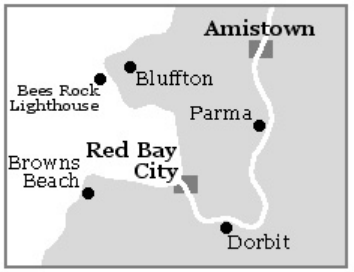
Good type placement:



Poor type placement:



Good type placement:



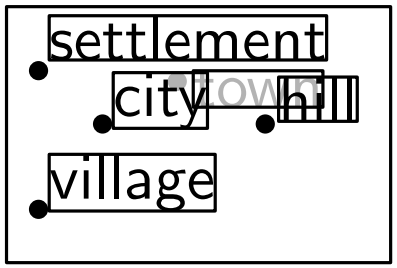
Beispiele von J B Krygier

→ lässt sich leicht in ein Problem der algorithmischen Geometrie zur automatischen Platzierung von Labeln übersetzen

Geometrische Beschriftungsmodelle

Geg: n Punkte in der Ebene und für jeden Punkt ein Label repräsentiert durch dessen bounding box

Ges: finde zulässige* Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)



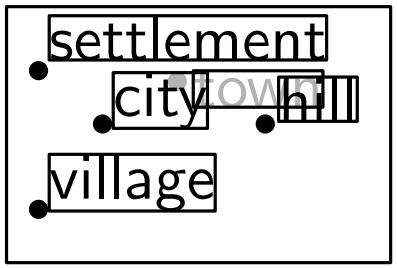
MaxNumber

Geometrische Beschriftungsmodelle

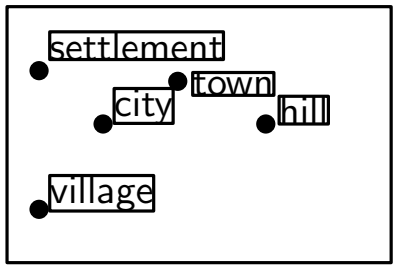
Geg: n Punkte in der Ebene und für jeden Punkt ein Label repräsentiert durch dessen bounding box

Ges: finde zulässige* Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)

oder finde zulässige* Beschriftung **aller** Label, so dass sich keine zwei Label schneiden und die **Schriftgröße** maximal ist (MaxSize)



MaxNumber



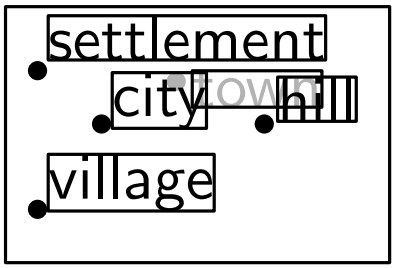
MaxSize

Geometrische Beschriftungsmodelle

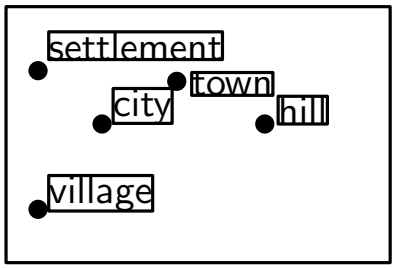
Geg: n Punkte in der Ebene und für jeden Punkt ein Label repräsentiert durch dessen bounding box

Ges: finde zulässige* Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)

oder finde zulässige* Beschriftung **aller** Label, so dass sich keine zwei Label schneiden und die **Schriftgröße** maximal ist (MaxSize)



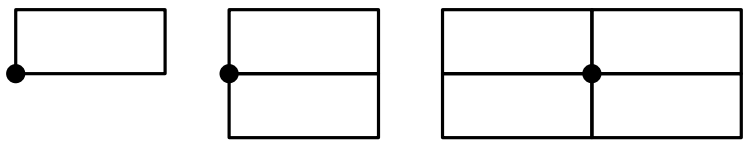
MaxNumber



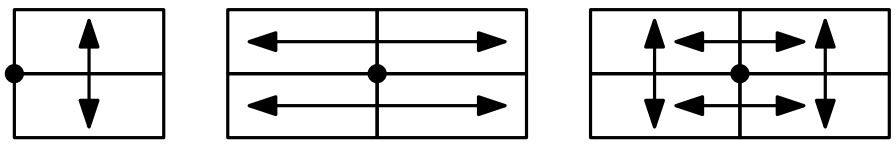
MaxSize

* Was ist eine **zulässige** Beschriftung?

diskrete Modelle



Slider-Modelle



Gütebegriff

Die $M_1 : M_2$-*Güte* zwischen zwei Labeling-Modellen M_1 und M_2 ist definiert als

$$\lim_{n \to \infty} \max \left\{ \frac{\text{Größe der opt. Lösung von } I \text{ bzgl. } M_1}{\text{Größe der opt. Lösung von } I \text{ bzgl. } M_2} \mid \text{Instanz } I \text{ der Größe } n. \right\}$$

Gütebegriff

Die $M_1 : M_2$-*Güte* zwischen zwei Labeling-Modellen M_1 und M_2 ist definiert als

$$\lim_{n\to\infty} \max \left\{ \frac{\text{Größe der opt. Lösung von } I \text{ bzgl. } M_1}{\text{Größe der opt. Lösung von } I \text{ bzgl. } M_2} \mid \text{Instanz } I \text{ der Größe } n. \right\}$$

Aufgabe: Zeigen Sie für Einheitsquadrate als Label, dass $\frac{3}{2}$ eine untere Schranke für die Güte zwischen dem Slider-Modell 4S und dem diskreten Modell ist, egal wie viele diskrete Positionen man im letzeren Modell zulässt.

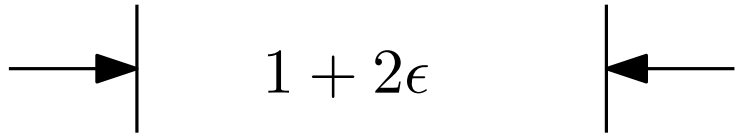
Gütebegriff

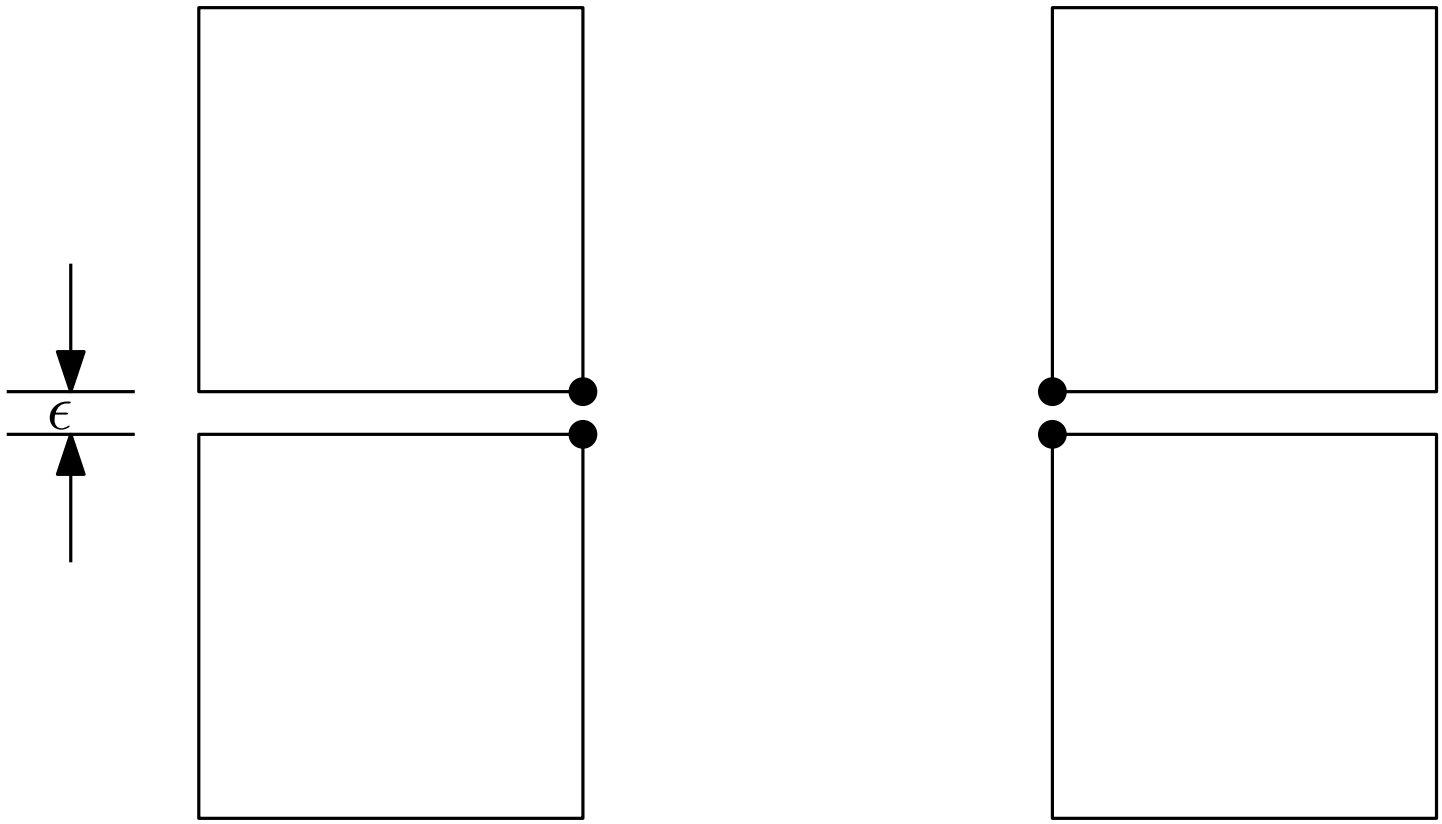
Die $M_1 : M_2$-*Güte* zwischen zwei Labeling-Modellen M_1 und M_2 ist definiert als

$$\lim_{n\to\infty} \max \left\{ \frac{\text{Größe der opt. Lösung von } I \text{ bzgl. } M_1}{\text{Größe der opt. Lösung von } I \text{ bzgl. } M_2} \mid \text{Instanz } I \text{ der Größe } n. \right\}$$

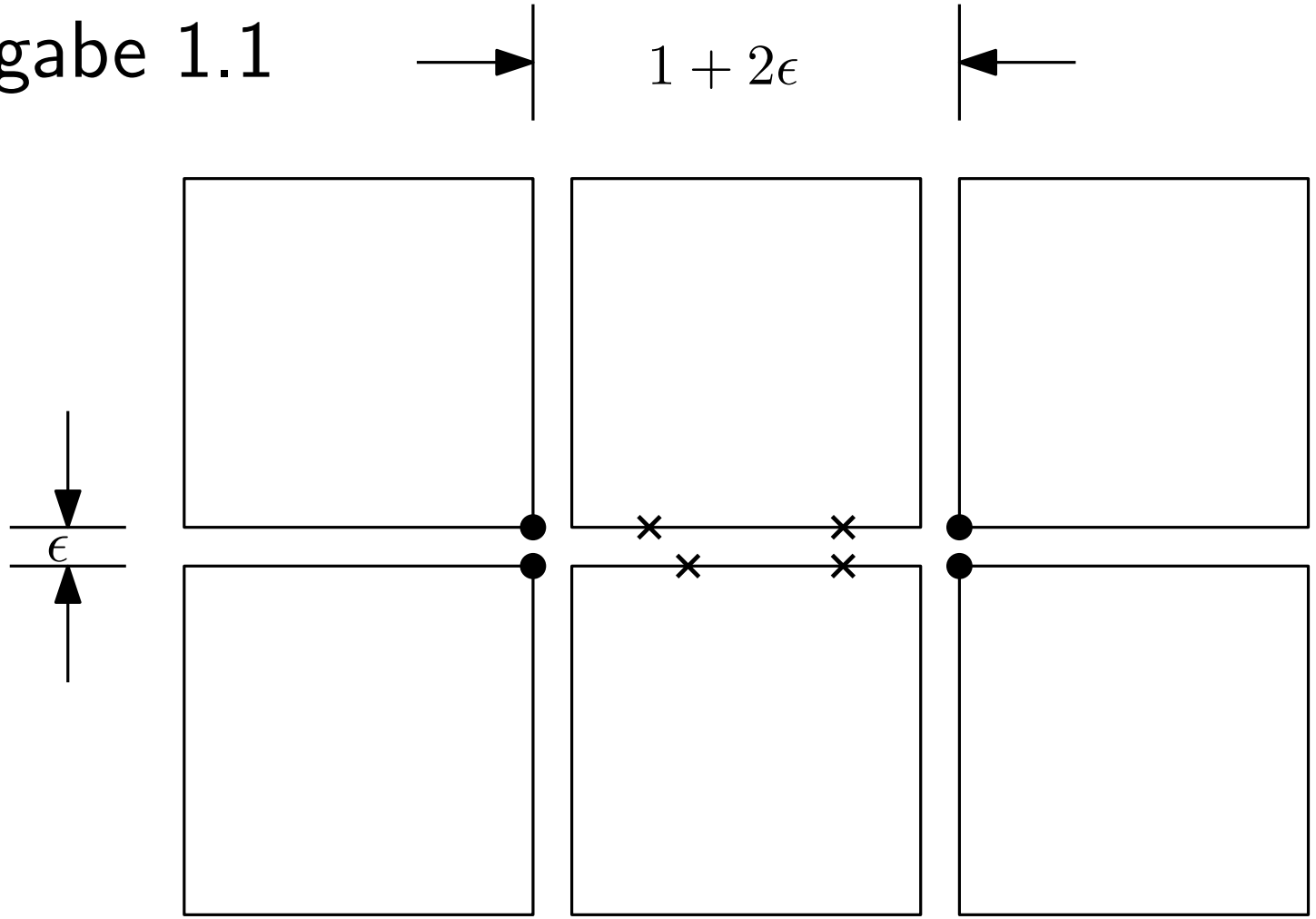
Aufgabe: Zeigen Sie für Einheitsquadrate als Label, dass $\frac{3}{2}$ eine untere Schranke für die Güte zwischen dem Slider-Modell 4S und dem diskreten Modell ist, egal wie viele diskrete Positionen man im letzeren Modell zulässt.

Aufgabe 1.1



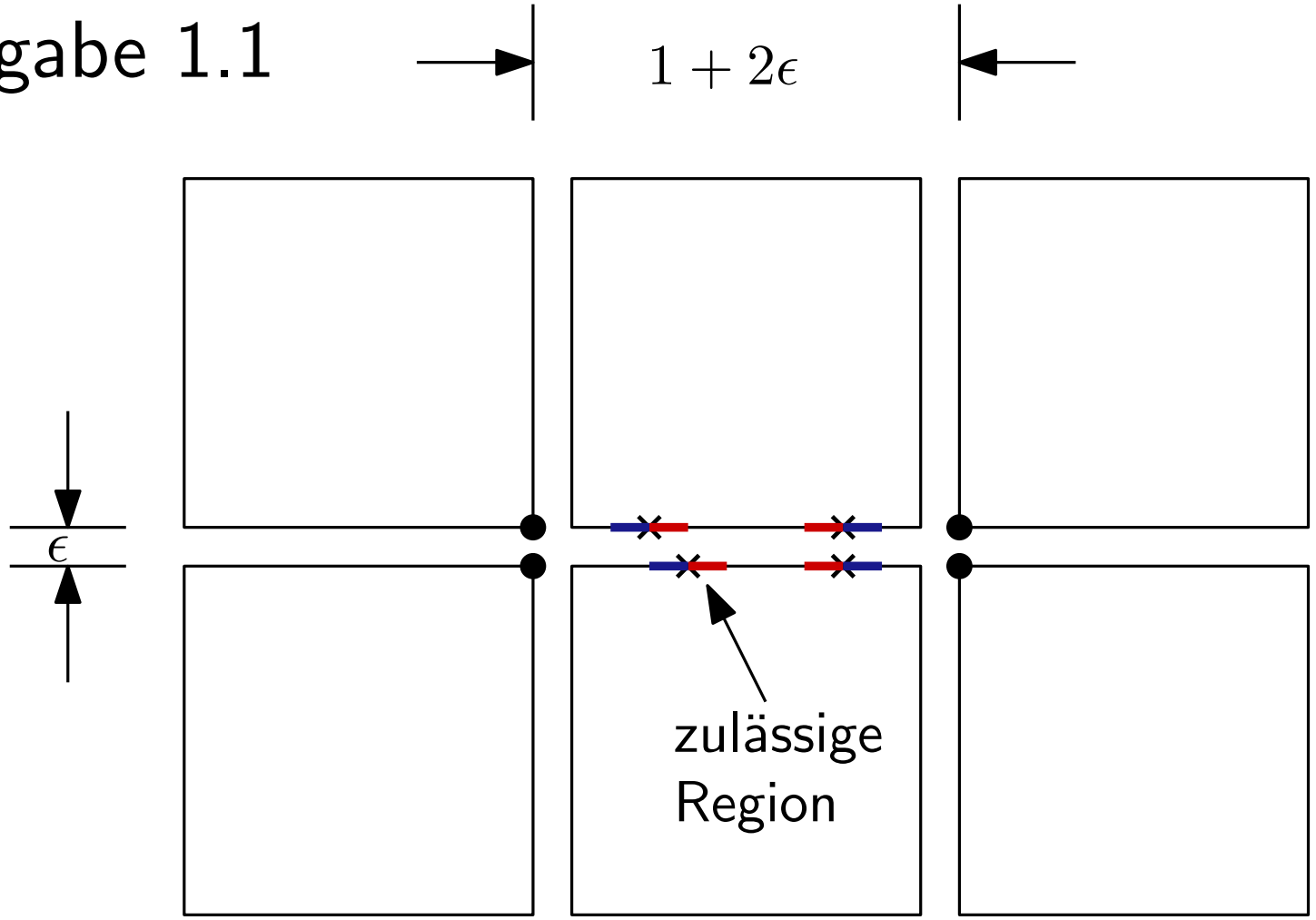


Aufgabe 1.1



× Vorgegebene Positionen im diskreten Modell.

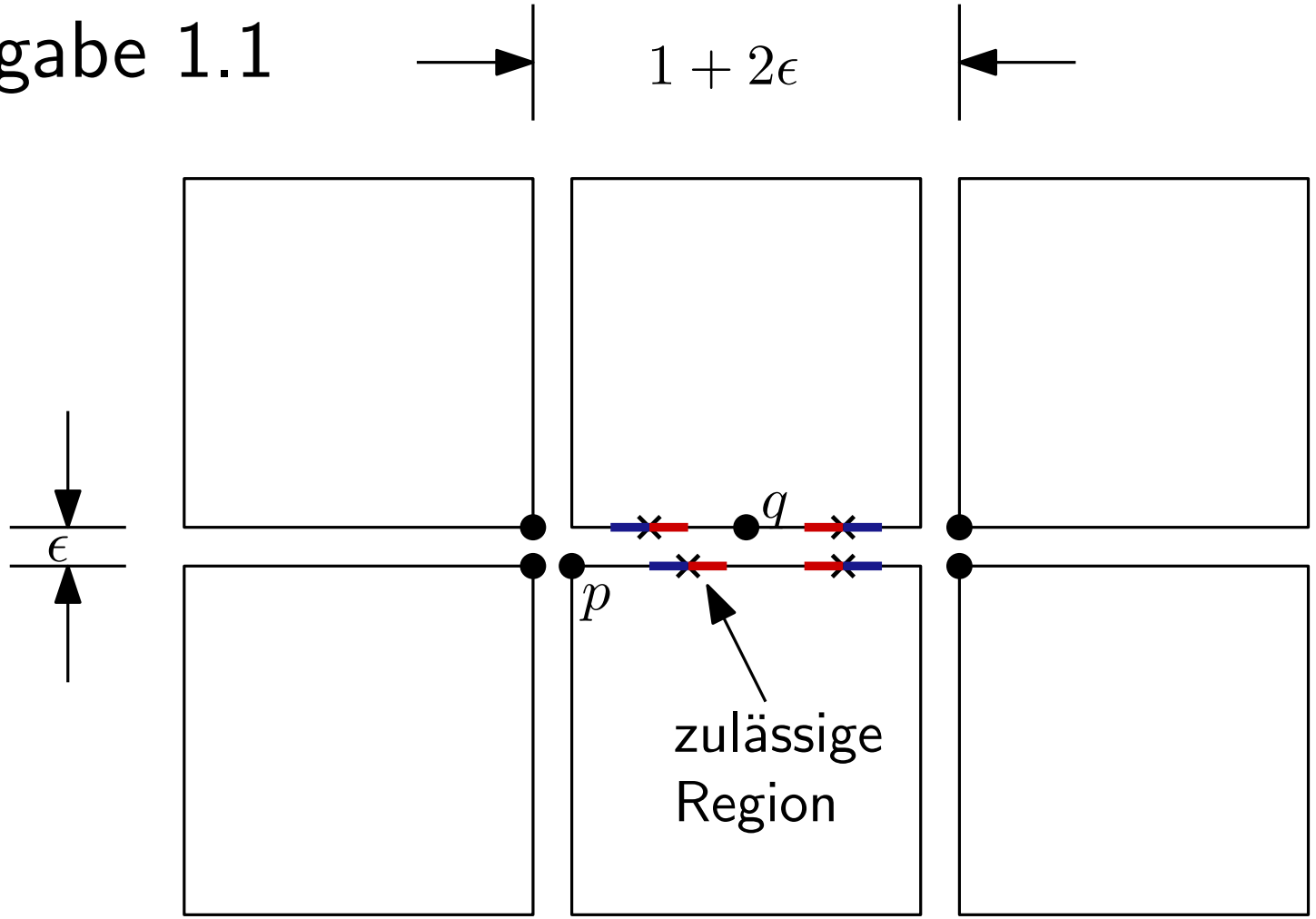
Aufgabe 1.1



× Vorgegebene Positionen im diskreten Modell.

Wähle ϵ kleiner als die Hälfte des kleinsten Abstands zweier vorgegebenen Positionen.

Aufgabe 1.1

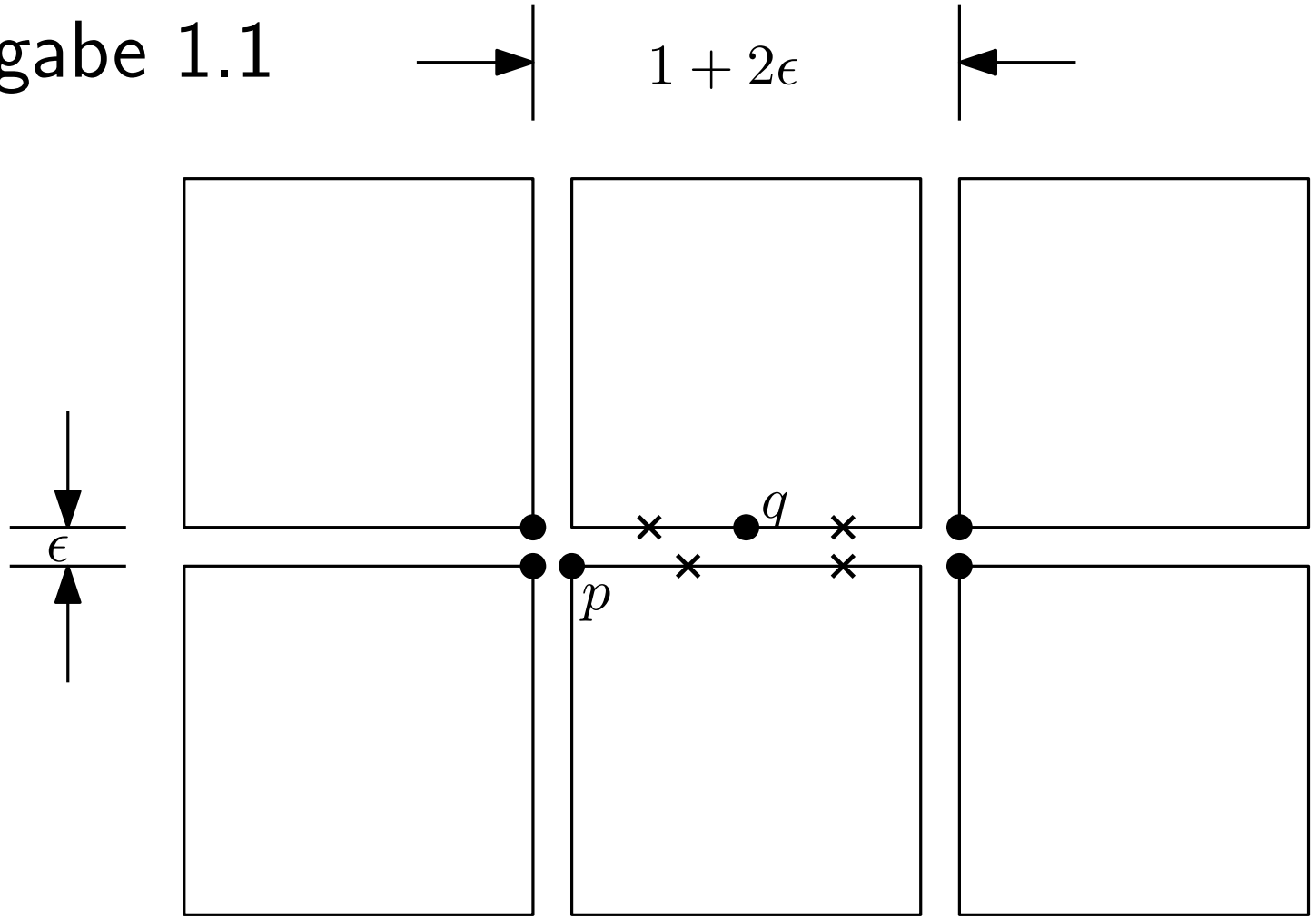


× Vorgegebene Positionen im diskreten Modell.

Wähle ϵ kleiner als die Hälfte des kleinsten Abstands zweier vorgegebenen Positionen.

Setze p und q außerhalb der zulässigen Region.

Aufgabe 1.1



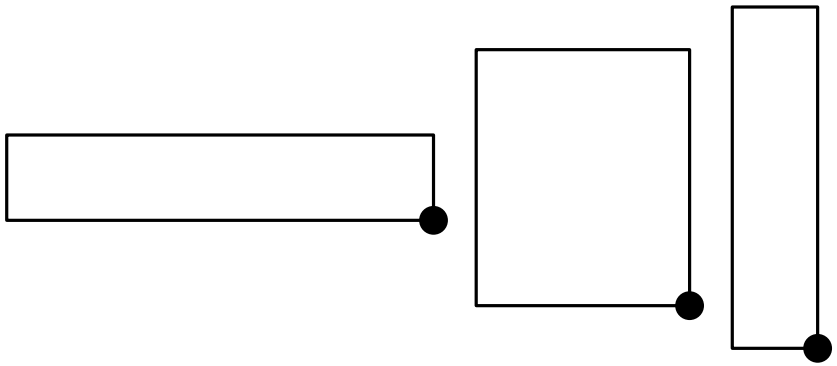
- Im Slider-Modell können alle sechs Punkte beschriftet werden.
- Im diskreten Modell können nur vier Punkte beschriftet werden.
- Wiederholung dieser Gruppe ergibt also eine Güte von $\frac{3}{2}$.

Aufgabe 1.2

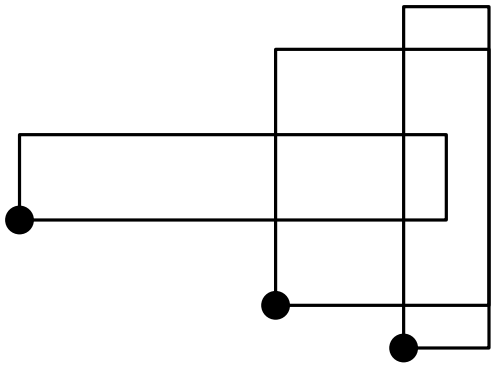
Aufgabe: Zeigen Sie, dass die $M_1 : M_2$-Güte zwischen zwei Modellen M_1 und M_2 beliebig schlecht werden kann, wenn man die Label nicht auf Einheitsquadrate beschränkt. Hinweis: Betrachten Sie hierzu die zwei diskreten Modelle 1P und 2P.

Aufgabe 1.2

Aufgabe: Zeigen Sie, dass die $M_1 : M_2$-Güte zwischen zwei Modellen M_1 und M_2 beliebig schlecht werden kann, wenn man die Label nicht auf Einheitsquadrate beschränkt. Hinweis: Betrachten Sie hierzu die zwei diskreten Modelle 1P und 2P.



Beschriftung aller Punkte in 2P möglich.



Beschriftung nur eines Punkts in 1P möglich.

Greedy-Algorithmus

Geg: Punktmenge $P = \{p_1, \ldots, p_n\}$, Menge rechteckiger Label $L = \{l_1, \ldots l_n\}$ mit Höhe 1 und variabler Breite

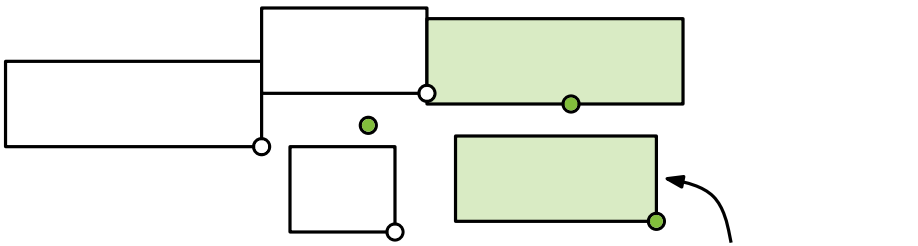
Modell: 4-Slider

Greedy-Algorithmus

Geg: Punktmenge $P = \{p_1, \ldots, p_n\}$, Menge rechteckiger Label $L = \{l_1, \ldots l_n\}$ mit Höhe 1 und variabler Breite

Modell: 4-Slider

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



Greedy-Algorithmus

Karlsruhe Institute of Technology

Geg: Punktmenge $P = \{p_1, \ldots, p_n\}$, Menge rechteckiger Label $L = \{l_1, \ldots l_n\}$ mit Höhe 1 und variabler Breite

Modell: 4-Slider

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

Algorithmus Greedy4S(P,L)

while linkestes Label l existiert **do**
  platziere l linkest möglich

Lemma 1: Greedy4S berechnet eine Faktor- Approximation.

Versuchen Sie den Approximationsfaktor zu bestimmen!

Greedy-Algorithmus

Geg: Punktmenge $P = \{p_1, \ldots, p_n\}$, Menge rechteckiger Label $L = \{l_1, \ldots l_n\}$ mit Höhe 1 und variabler Breite

Modell: 4-Slider

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

Algorithmus Greedy4S(P,L)

while linkestes Label l existiert **do**
  platziere l linkest möglich

Lemma 1: Greedy4S berechnet eine Faktor-1/2 Approximation.

Greedy-Algorithmus

Geg: Punktmenge $P = \{p_1, \ldots, p_n\}$, Menge rechteckiger Label $L = \{l_1, \ldots l_n\}$ mit Höhe 1 und variabler Breite

Modell: 4-Slider

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

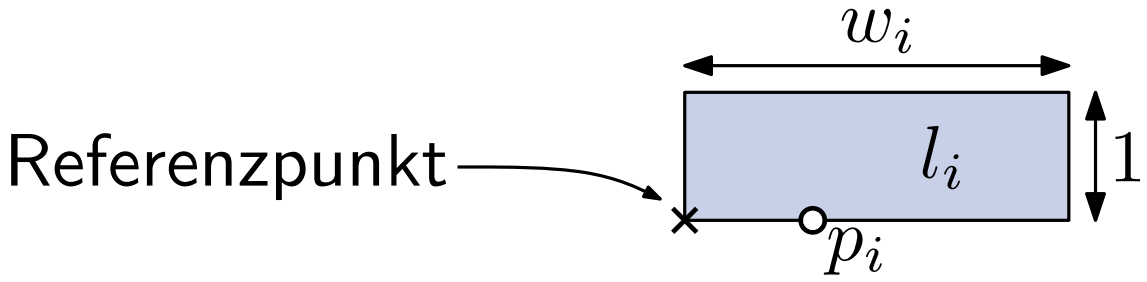
Algorithmus Greedy4S(P,L)

while linkestes Label l existiert **do**
  platziere l linkest möglich

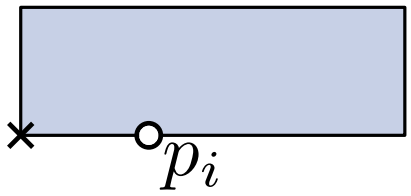
Lemma 1: Greedy4S berechnet eine Faktor-1/2 Approximation.

Aufgabe: Welcher Approximations-Faktor ergibt sich, wenn die Label unterschiedliche Höhen besitzen dürfen?

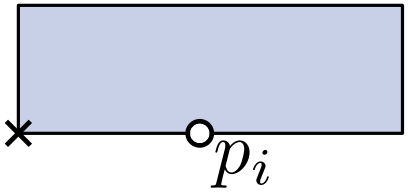
Notation



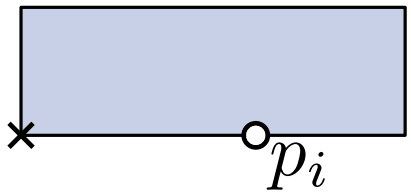
Notation



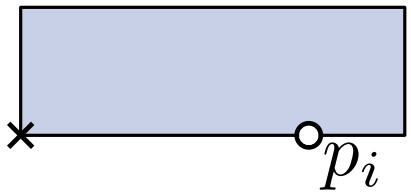
Notation



Notation



Notation

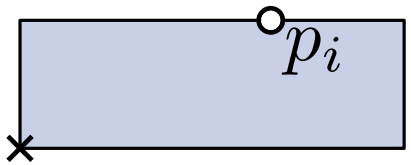


Notation

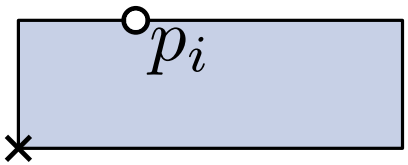
Notation

Notation

Notation

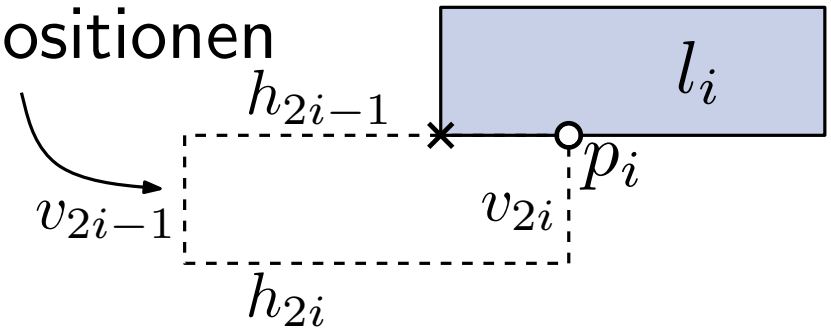


Notation



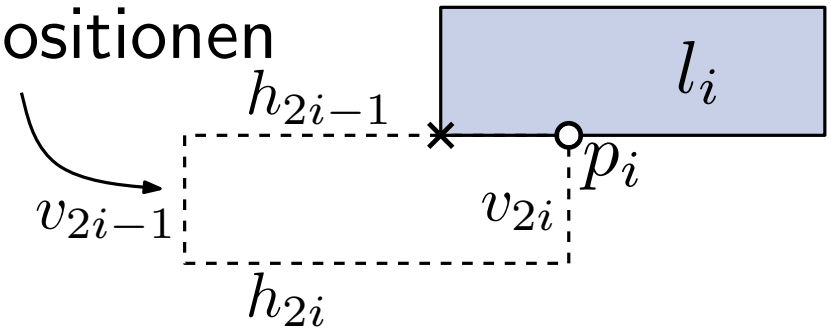
Notation

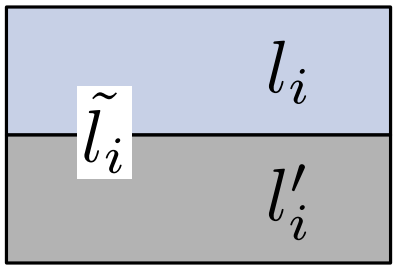
mögl. Positionen



Notation

mögl. Positionen

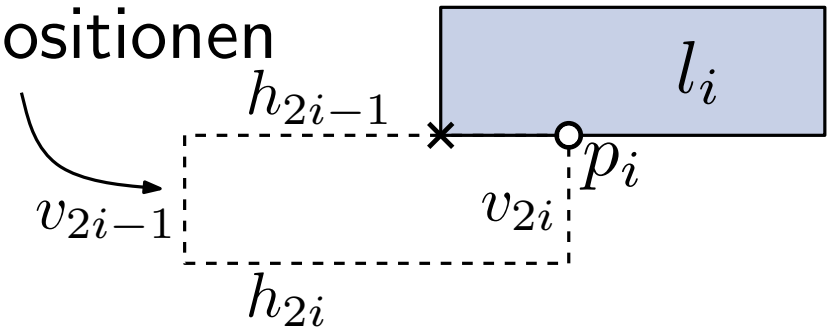


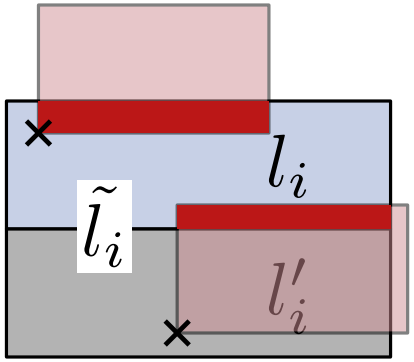


Label l_i und erweitertes Label $\tilde{l}_i$

Notation

mögl. Positionen



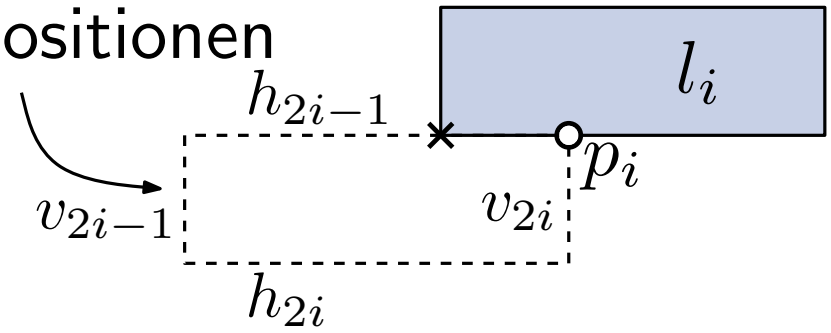


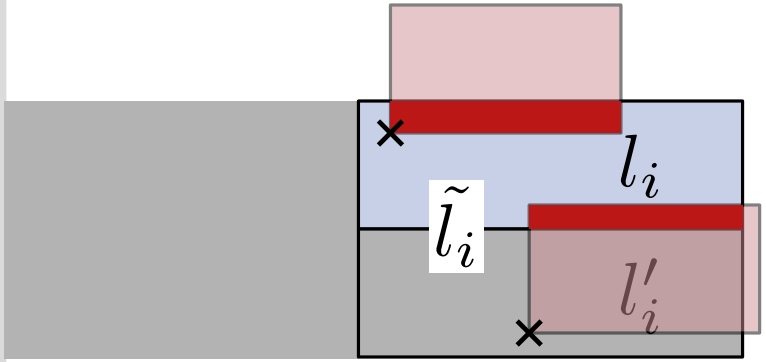
Label l_i und erweitertes Label $\tilde{l}_i$

kein Referenzpunkt darf in $\tilde{l}_i$ liegen

Notation

mögl. Positionen





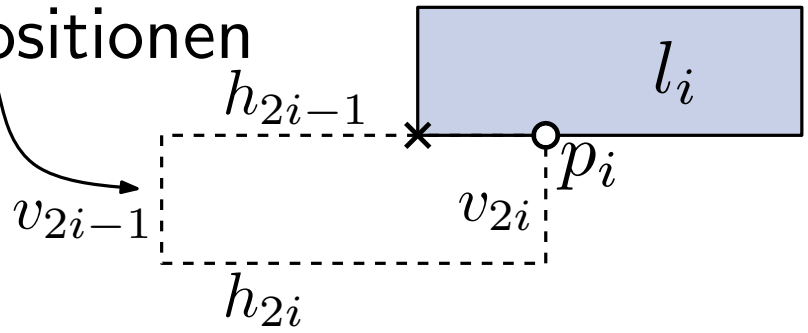
Label l_i und erweitertes Label $\tilde{l}_i$

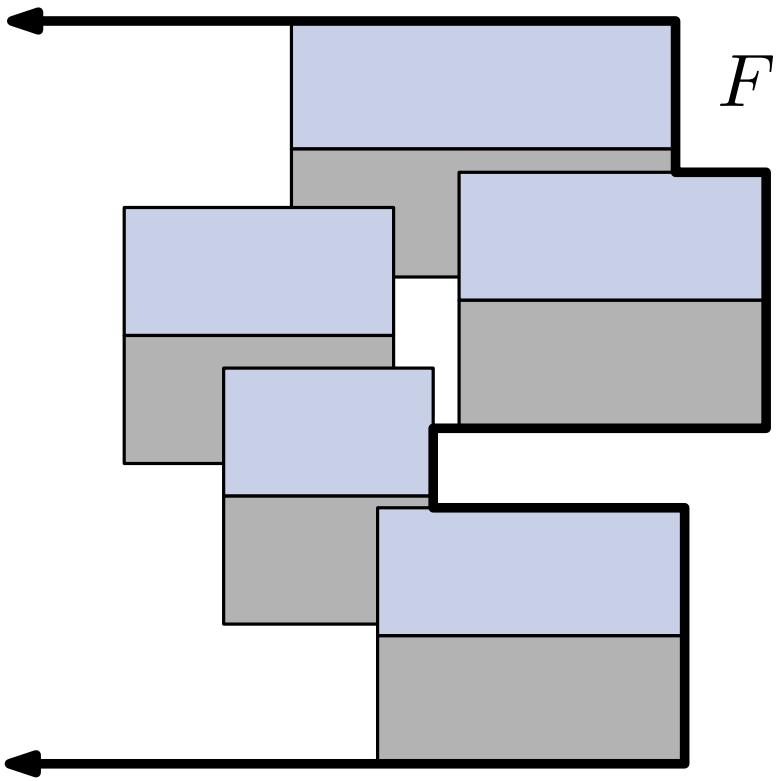
kein Referenzpunkt darf in $\tilde{l}_i$ liegen

da immer linkestes Label platziert wird, kein Referenzpunkt links von $\tilde{l}_i$

Notation

mögl. Positionen

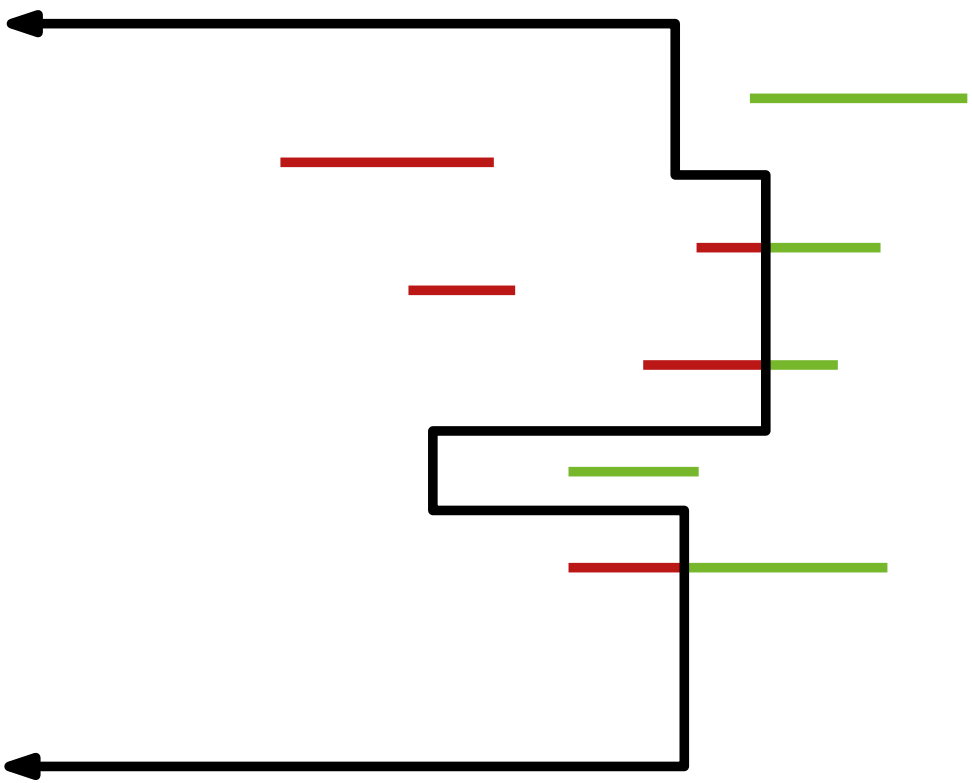




Grenze F als *right envelope* der platzierten Label

Grenze und linkestes Label

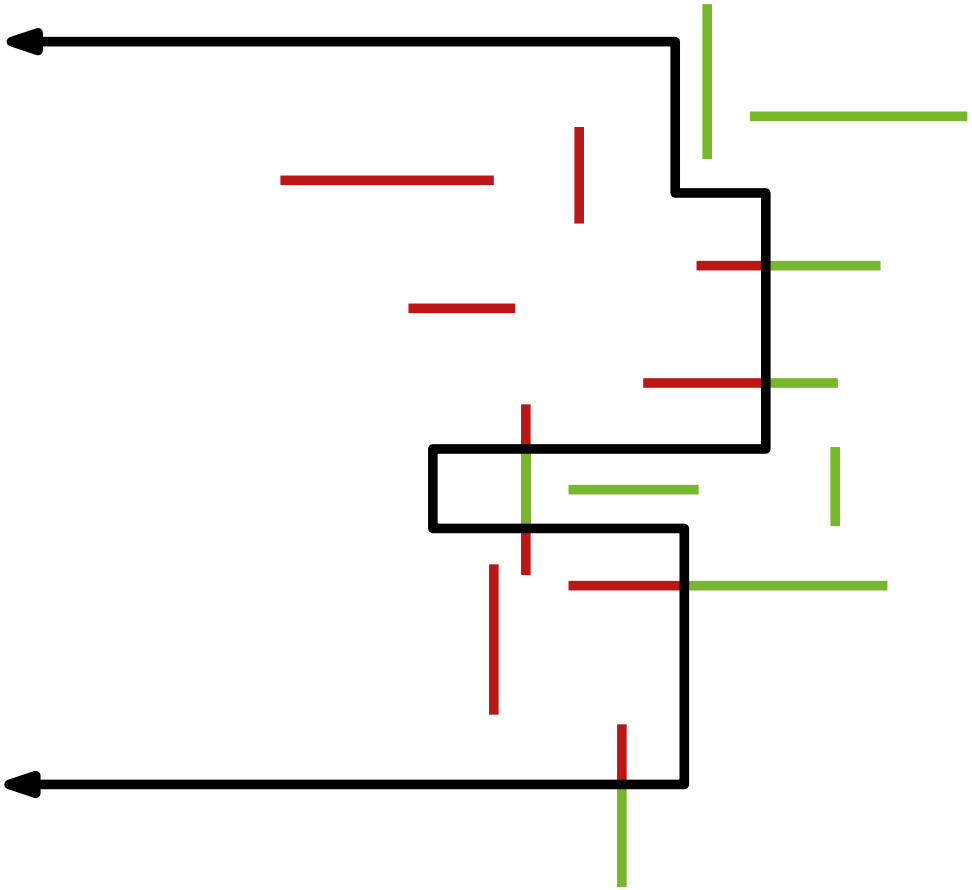
Zur Bestimmung des nächsten linkesten Labels genügt es F und die Strecken $h_{2i-1}, h_{2i}, v_{2i-1}, v_{2i}$ für alle Punkte p_i rechts von F zu betrachten.



- Menge H der horizontalen Strecken
- Menge V der vertikalen Strecken
- $H_{\text{right}} \subseteq H$: rechts von F
- $H_{\text{int}} \subseteq H$: schneiden F

Grenze und linkestes Label

Zur Bestimmung des nächsten linkesten 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
- $H_{\text{right}} \subseteq H$: 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

Horizontale Strecken

1) Menge H_{right}

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

Horizontale Strecken

1) Menge H_{right}

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

2) Menge H_{int}

- Red-Black Tree $\mathcal{T}_i$ 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
- min-Heap $\mathcal{H}_{\text{int}}$ für linkestes Label in jedem $\mathcal{T}_i$

Algorithmus

while $\mathcal{H}_{\text{right}}$, $\mathcal{H}_{\text{int}}$ oder $\mathcal{H}_V$ nicht leer **do**
- $v \leftarrow$ find-min($\mathcal{H}_V$)
- **while** v links von F **do**
  - delete-min($\mathcal{H}_V$); $v \leftarrow$ find-min($\mathcal{H}_V$)
- $l_i \leftarrow$ linkestes Label aus $\mathcal{H}_{\text{right}}$, $\mathcal{H}_{\text{int}}$ und $\mathcal{H}_V$
- füge l_i zur Beschriftung hinzu
- $f_{\text{new}} \leftarrow$ rechte Kante von $\tilde{l_i}$
- update F und $\mathcal{T}_V$ mit f_{new}
- suche mit der Region links von f_{new} in $\mathcal{P}_{\text{left}}$ und $\mathcal{P}_{\text{right}}$ und update $\mathcal{H}_{\text{right}}$, $\mathcal{P}_{\text{left}}$, $\mathcal{H}_{\text{int}}$, $\mathcal{T}_i$'s und $\mathcal{P}_{\text{right}}$ wie beschrieben
- entferne alle Verweise auf p_i aus den Datenstrukturen

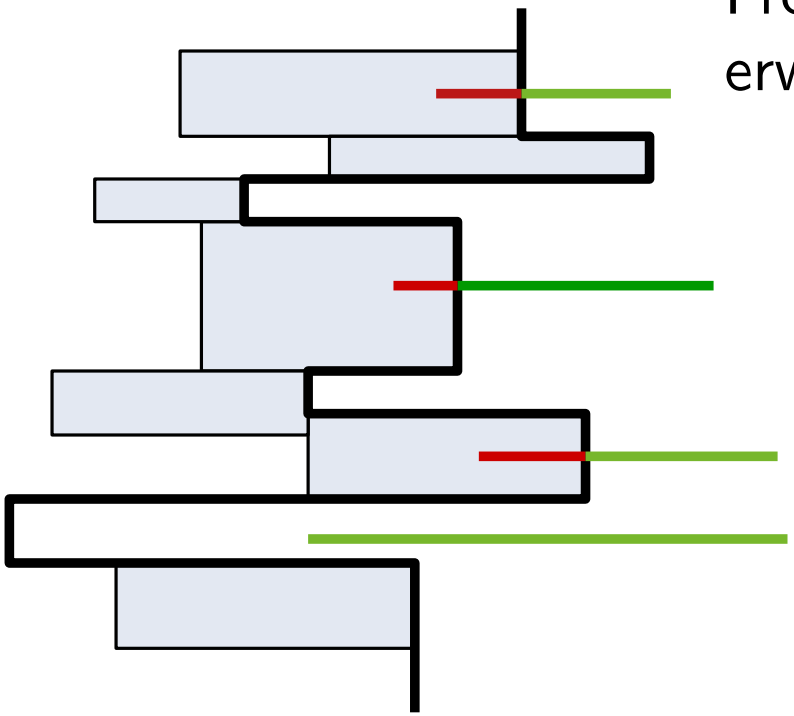
Algorithmus

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

- $v \leftarrow$ find-min($\mathcal{H}_V$)
- **while** v links von F **do**
  - delete-min($\mathcal{H}_V$); $v \leftarrow$ find-min($\mathcal{H}_V$)
- $l_i \leftarrow$ linkestes Label aus $\mathcal{H}_{\text{right}}$, $\mathcal{H}_{\text{int}}$ und $\mathcal{H}_V$
- füge l_i zur Beschriftung hinzu
- $f_{\text{new}} \leftarrow$ rechte Kante von $\tilde{l_i}$
- update F und $\mathcal{T}_V$ mit f_{new}
- suche mit der Region links von f_{new} in $\mathcal{P}_{\text{left}}$ und $\mathcal{P}_{\text{right}}$ und update $\mathcal{H}_{\text{right}}$, $\mathcal{P}_{\text{left}}$, $\mathcal{H}_{\text{int}}$, $\mathcal{T}_i$'s und $\mathcal{P}_{\text{right}}$ wie beschrieben
- entferne alle Verweise auf p_i aus den Datenstrukturen

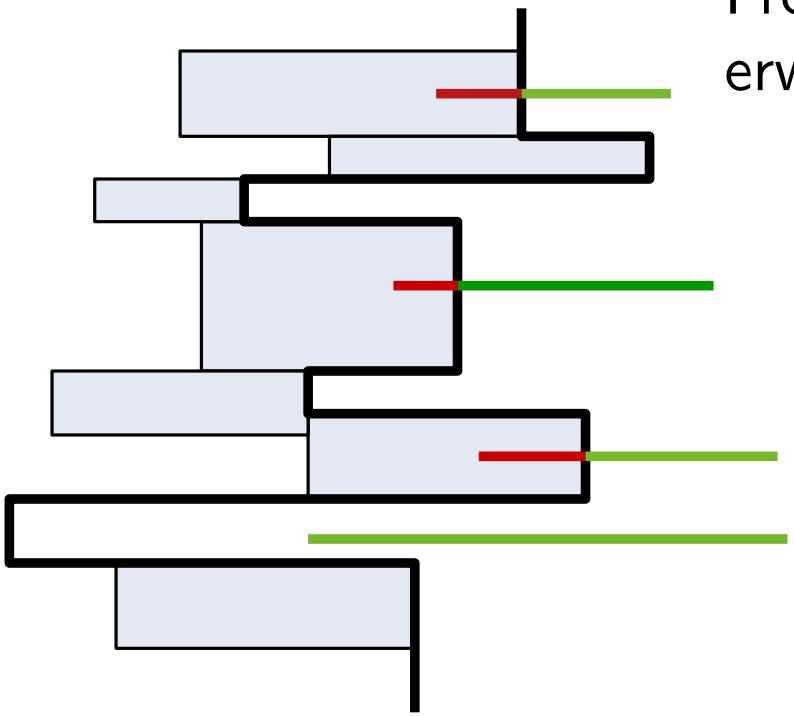
Aufgabe: Wie muss Algorithmus angepasst werden, damit Label verschiedener Höhen erlaubt sind?

Skizze



Front basiert nur auf Label und nicht auf erweiterte Label.

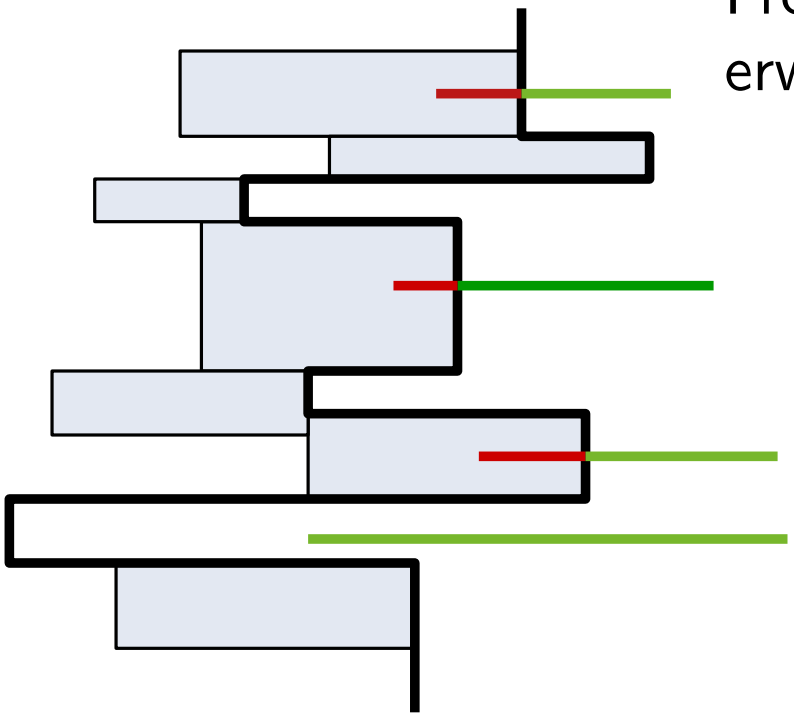
Skizze



Front basiert nur auf Label und nicht auf erweiterte Label.

Positionen auf einer Strecke links der Front sind nicht zulässig.

Skizze

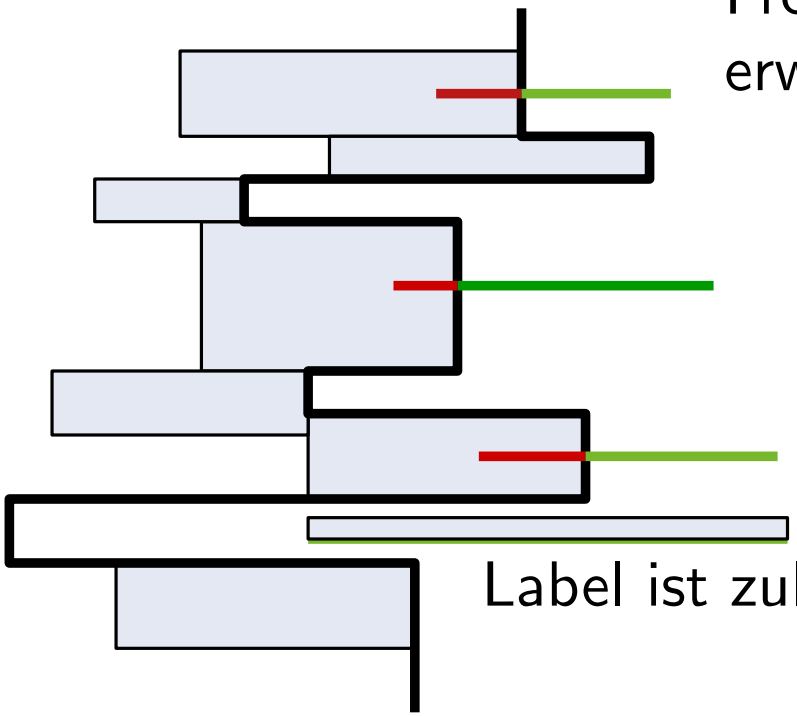


Front basiert nur auf Label und nicht auf erweiterte Label.

Positionen auf einer Strecke links der Front sind nicht zulässig.

Positionen auf einer Strecke rechts der Front **können** zulässig sein.

Skizze

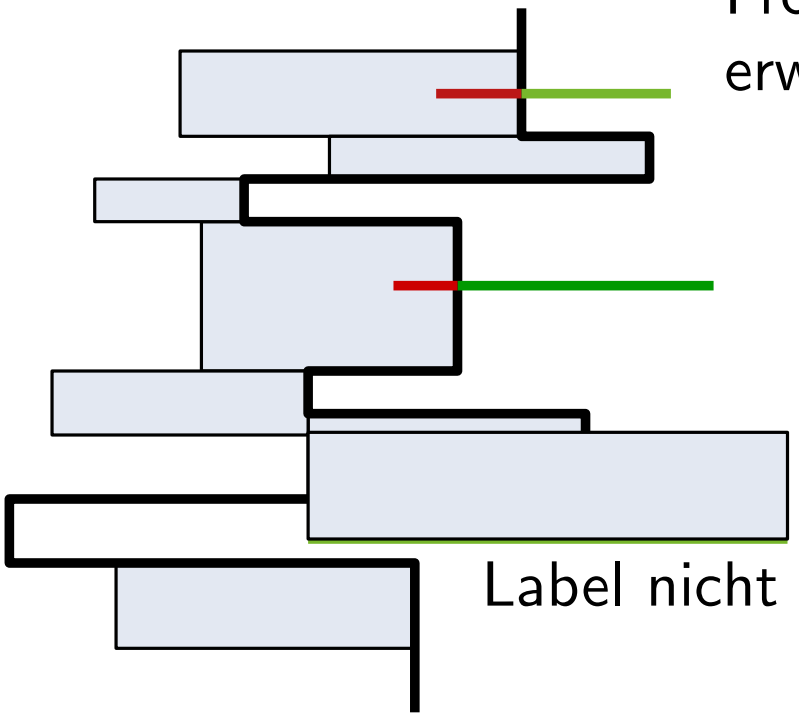


Front basiert nur auf Label und nicht auf erweiterte Label.

Positionen auf einer Strecke links der Front sind nicht zulässig.

Positionen auf einer Strecke rechts der Front **können** zulässig sein.

Skizze



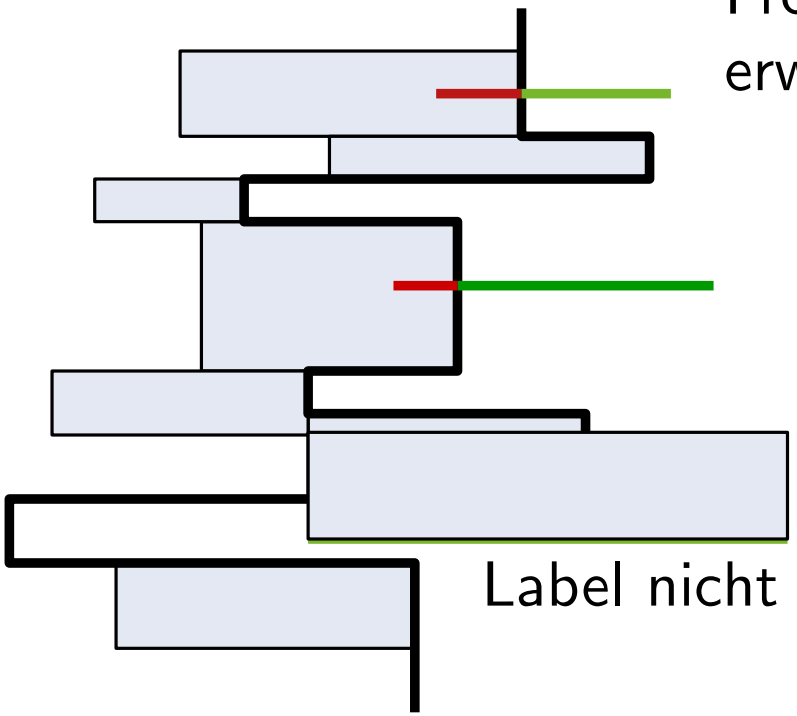
Front basiert nur auf Label und nicht auf erweiterte Label.

Label nicht zulässig

Positionen auf einer Strecke links der Front sind nicht zulässig.

Positionen auf einer Strecke rechts der Front **können** zulässig sein.

Skizze



Front basiert nur auf Label und nicht auf erweiterte Label.

Label nicht zulässig

Positionen auf einer Strecke links der Front sind nicht zulässig.

Positionen auf einer Strecke rechts der Front **können** zulässig sein.

Überprüfe bei Auswahl des linkesten Label l ob l zulässig ist.

Modifikation

while $\mathcal{H}_{\text{right}}$, $\mathcal{H}_{\text{int}}$ oder $\mathcal{H}_V$ nicht leer **do**
- $v \leftarrow$ find-min$(\mathcal{H}_V)$
- **while** v links von F **do**
  - delete-min$(\mathcal{H}_V)$; $v \leftarrow$ find-min$(\mathcal{H}_V)$
- $l_i \leftarrow$ linkestes Label aus $\mathcal{H}_{\text{right}}$, $\mathcal{H}_{\text{int}}$ und $\mathcal{H}_V$
- **while** l_i nicht zulässig **do**
  - $l_i \leftarrow$ linkestes Label aus $\mathcal{H}_{\text{right}}$, $\mathcal{H}_{\text{int}}$ und $\mathcal{H}_V$
- füge l_i zur Beschriftung hinzu
- $f_{\text{new}} \leftarrow$ rechte Kante von l_i
- update F und $\mathcal{T}_V$ mit f_{new}
- suche mit der Region links von f_{new} in $\mathcal{P}_{\text{left}}$ und $\mathcal{P}_{\text{right}}$ und update $\mathcal{H}_{\text{right}}$, $\mathcal{P}_{\text{left}}$, $\mathcal{H}_{\text{int}}$, $\mathcal{T}_i$'s und $\mathcal{P}_{\text{right}}$ wie beschrieben
- entferne alle Verweise auf p_i aus den Datenstrukturen

Algorithmus

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

- $v \leftarrow$ find-min$(\mathcal{H}_V)$
- **while** v links von F **do**
  - delete-min$(\mathcal{H}_V)$; $v \leftarrow$ find-min$(\mathcal{H}_V)$
- $l_i \leftarrow$ linkestes Label aus $\mathcal{H}_{\text{right}}$, $\mathcal{H}_{\text{int}}$ und $\mathcal{H}_V$
- füge l_i zur Beschriftung hinzu
- $f_{\text{new}} \leftarrow$ rechte Kante von $\tilde{l_i}$
- update F und $\mathcal{T}_V$ mit f_{new}
- suche mit der Region links von f_{new} in $\mathcal{P}_{\text{left}}$ und $\mathcal{P}_{\text{right}}$ und update $\mathcal{H}_{\text{right}}$, $\mathcal{P}_{\text{left}}$, $\mathcal{H}_{\text{int}}$, $\mathcal{T}_i$'s und $\mathcal{P}_{\text{right}}$ wie beschrieben
- entferne alle Verweise auf p_i aus den Datenstrukturen

Aufgabe: Anpassung auf diskrete Modelle?

Algorithmus

while $\mathcal{H}_{\text{right}}$, $\mathcal{H}_{\text{int}}$ oder $\mathcal{H}_V$ nicht leer **do**
- $v \leftarrow$ find-min$(\mathcal{H}_V)$
- **while** v links von F **do**
  - delete-min$(\mathcal{H}_V)$; $v \leftarrow$ find-min$(\mathcal{H}_V)$
- $l_i \leftarrow$ linkestes Label aus $\mathcal{H}_{\text{right}}$, $\mathcal{H}_{\text{int}}$ und $\mathcal{H}_V$
- füge l_i zur Beschriftung hinzu
- $f_{\text{new}} \leftarrow$ rechte Kante von $\tilde{l_i}$
- update F und $\mathcal{T}_V$ mit f_{new}
- suche mit der Region links von f_{new} in $\mathcal{P}_{\text{left}}$ und $\mathcal{P}_{\text{right}}$ und update $\mathcal{H}_{\text{right}}$, $\mathcal{P}_{\text{left}}$, $\mathcal{H}_{\text{int}}$, $\mathcal{T}_i$'s und $\mathcal{P}_{\text{right}}$ wie beschrieben
- entferne alle Verweise auf p_i aus den Datenstrukturen

Aufgabe: Anpassung auf diskrete Modelle?

- Nur vorgegeben Positionen eines Labels interessant.
- Positionen links der Front und auf der Front implizieren Konflikt.
- $\Rightarrow$ nur Positionen rechts der Front interessant.

Algorithmus

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

- ~~$v \leftarrow$ find-min$(\mathcal{H}_V)$~~
- ~~**while** v links von F **do**~~
  - ~~delete-min$(\mathcal{H}_V)$; $v \leftarrow$ find-min$(\mathcal{H}_V)$~~
- $l_i \leftarrow$ linkestes Label aus $\mathcal{H}_{\text{right}}$, ~~$\mathcal{H}_{\text{int}}$ und $\mathcal{H}_V$~~
- füge l_i zur Beschriftung hinzu
- $f_{\text{new}} \leftarrow$ rechte Kante von $\tilde{l_i}$
- update F ~~und $\mathcal{T}_V$~~ mit f_{new}
- suche mit der Region links von f_{new} in $\mathcal{P}_{\text{left}}$ ~~und $\mathcal{P}_{\text{right}}$~~ und update $\mathcal{H}_{\text{right}}$, $\mathcal{P}_{\text{left}}$, ~~$\mathcal{H}_{\text{int}}$~~, $\mathcal{T}_i$'s und ~~$\mathcal{P}_{\text{right}}$~~ wie beschrieben
- entferne alle Verweise auf p_i aus den Datenstrukturen

Aufgabe: Anpassung auf diskrete Modelle?

- Nur vorgegeben Positionen eines Labels interessant.
- Positionen links der Front und auf der Front implizieren Konflikt.
- $\Rightarrow$ nur Positionen rechts der Front interessant.