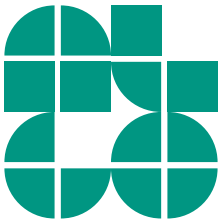
Übung Algorithmische Kartografie

Übungsblatt 4 & 5

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

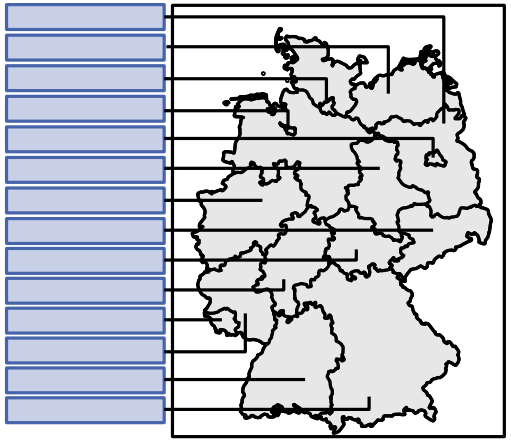
Benjamin Niedermann

06.06.2013

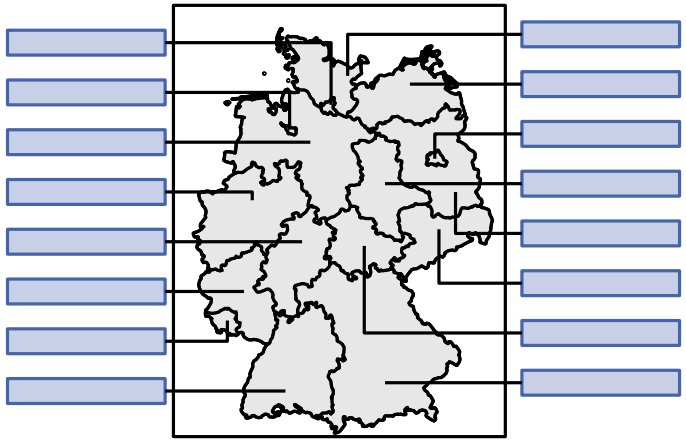


Übungsblatt 4

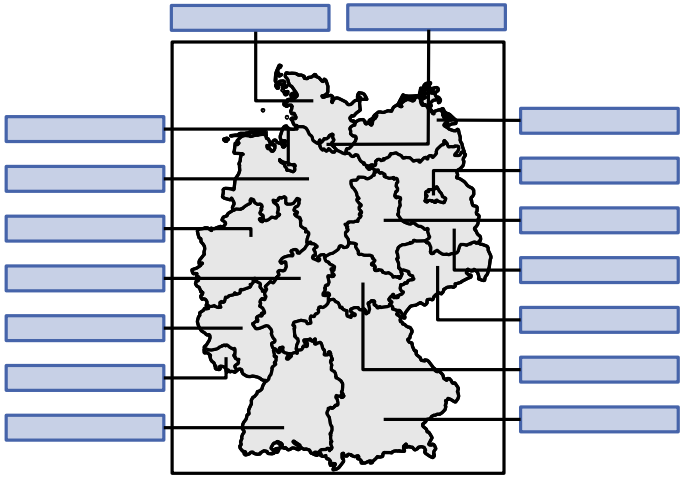
Arten von Boundary-Labeling



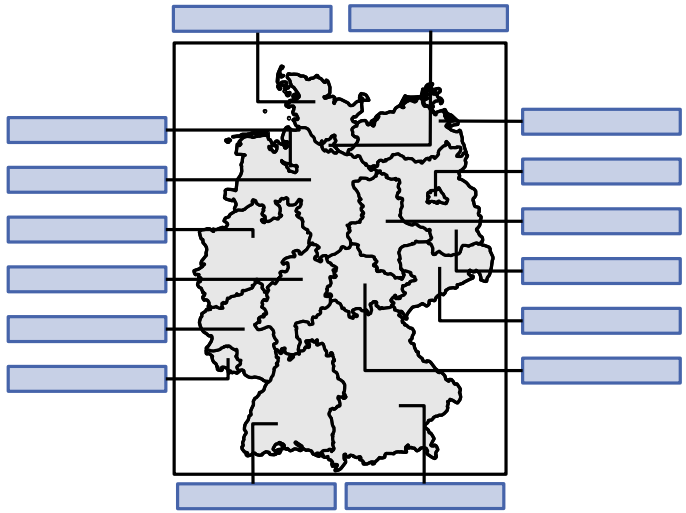
Einseitiger Fall



Zweiseitiger Fall

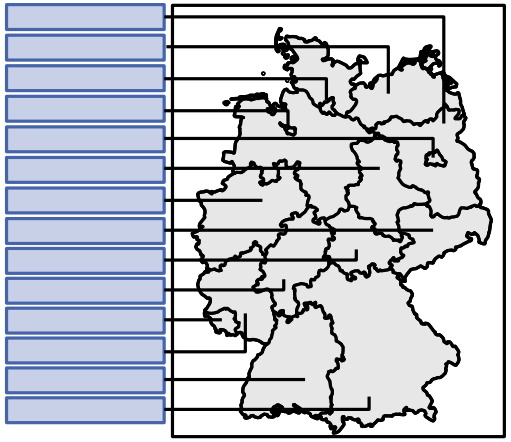


Dreiseitiger Fall

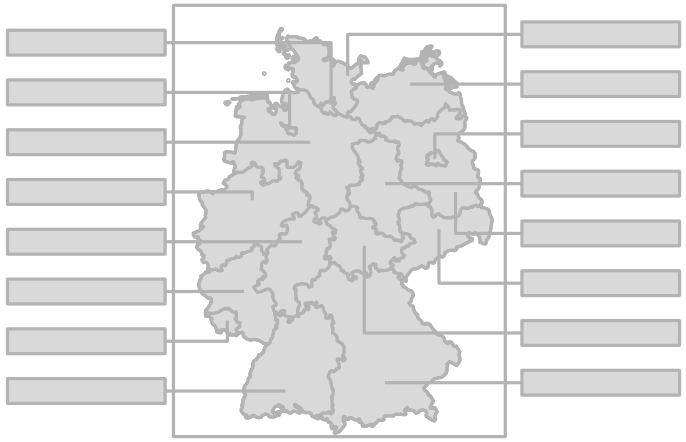


Vierseitiger Fall

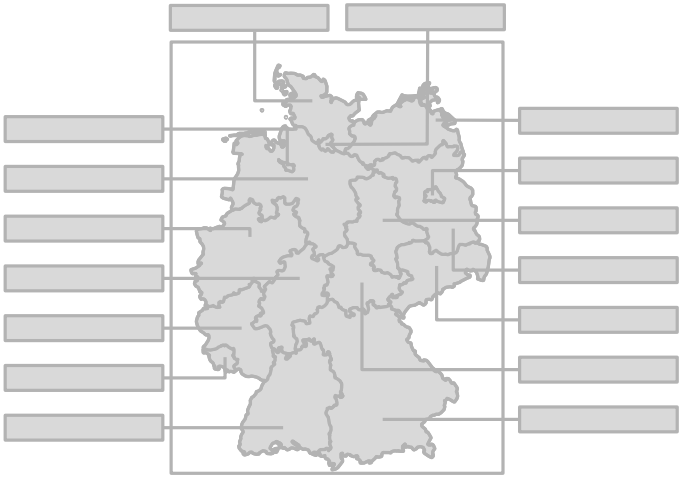
Arten von Boundary-Labeling



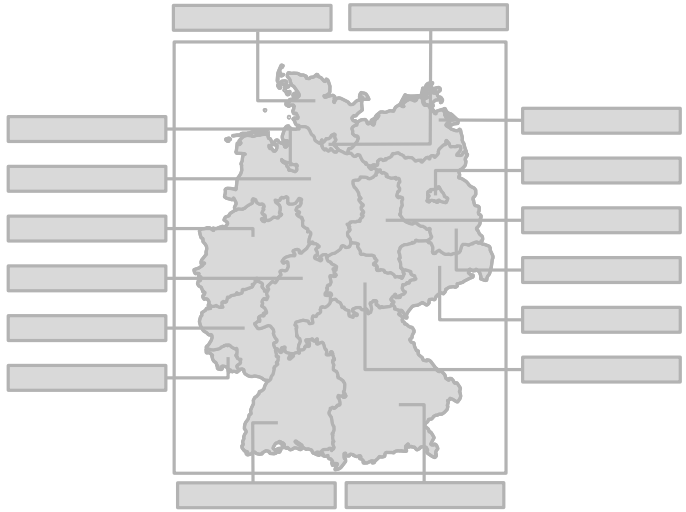
Einseitiger Fall



Zweiseitiger Fall



Dreiseitiger Fall

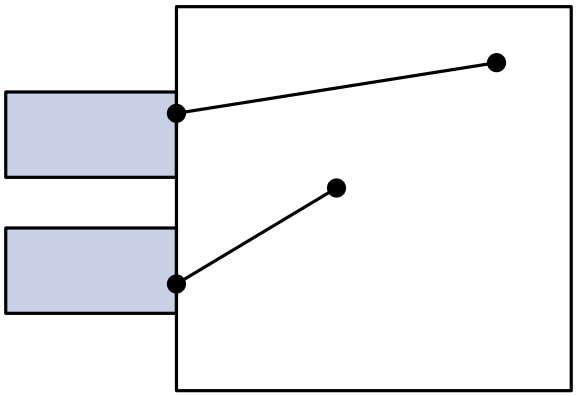


Vierseitiger Fall

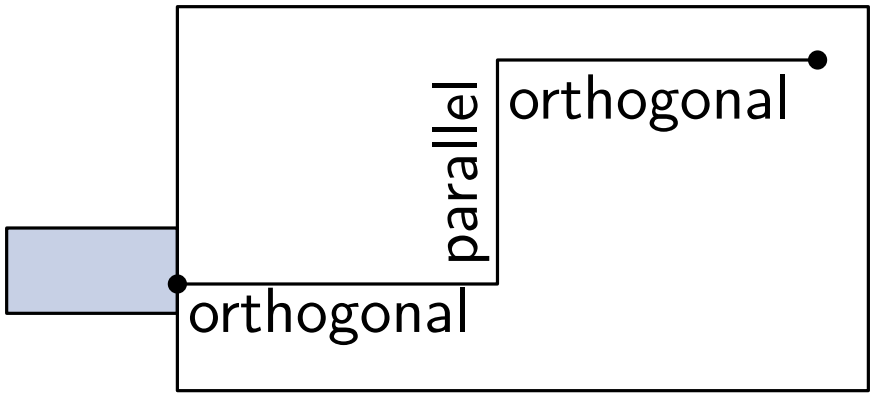
Typen von Leadern

Üblich: Leader werden aus Sequenz von Strecken zusammengesetzt.

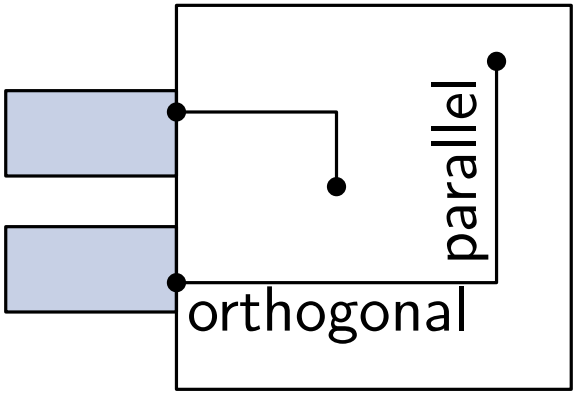
Typische Beispiele:



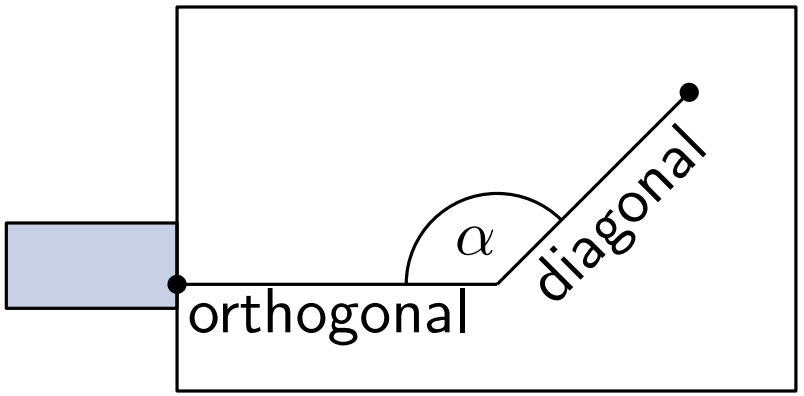
s-Leader



opo-Leader



po-Leader



do-Leader

Zielfunktion

Bewertungsfunktion:

$$\mathrm{bad} : L \times P \to \mathbb{R}$$

Bewertet Zuordnung von Labeln L und Punkten P.

Optimierungsproblem:
Finde für gegebene Instanz (R, P, L) eine kreuzungsfreie Zuordnung $\mathcal{L}$, sodass

$$\sum_{\lambda \in \mathcal{L}} \mathrm{bad}(\ell(\lambda), p(\lambda))$$

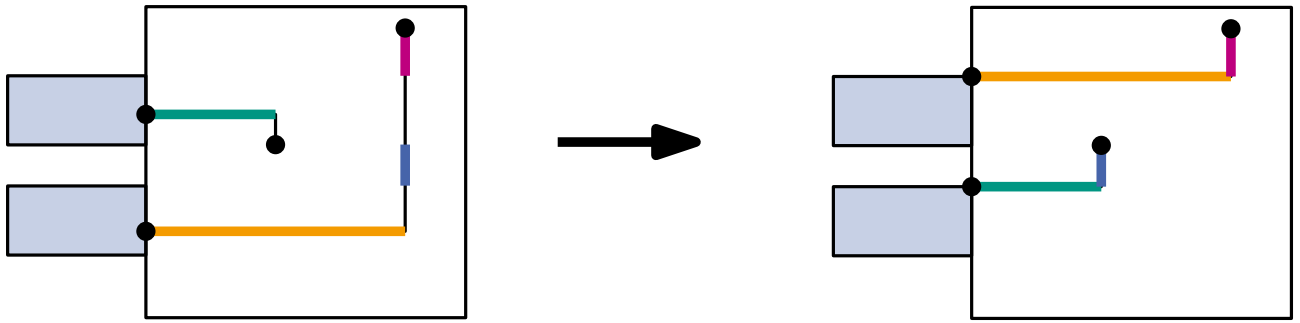
unter allen kreuzeungsfreien Zuordnungen minimiert wird.

Notation:

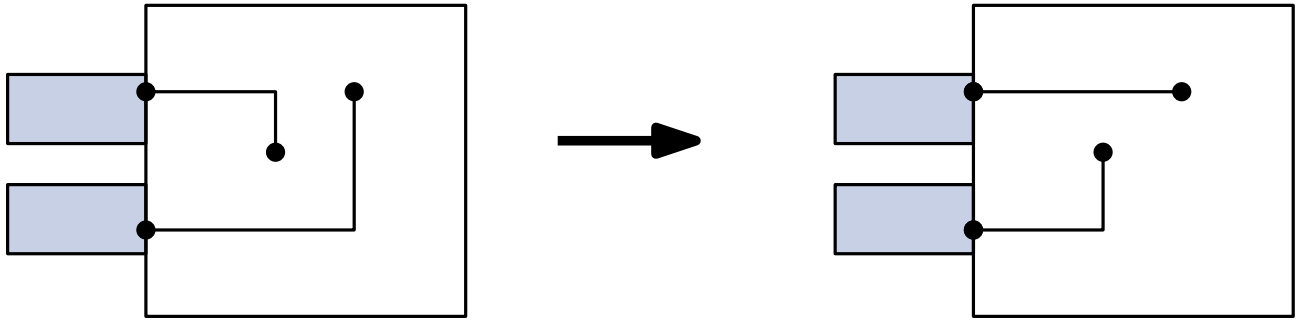
- $\ell(\lambda)$: Label aus L, das von Leader λ angebunden wird.
- $p(\lambda)$: Punkt aus p, der von Leader λ angebunden wird.

Beispiele für Bewertungsfunktionen

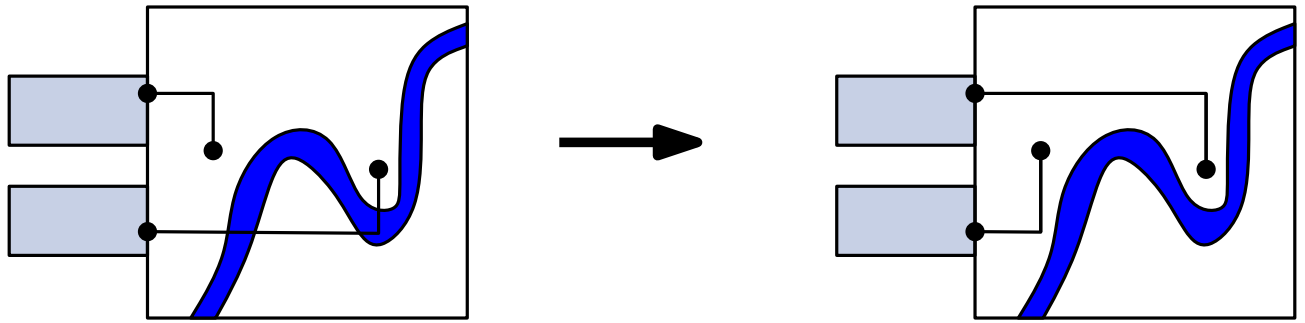
Längenminimierung:



Knickminimierung:



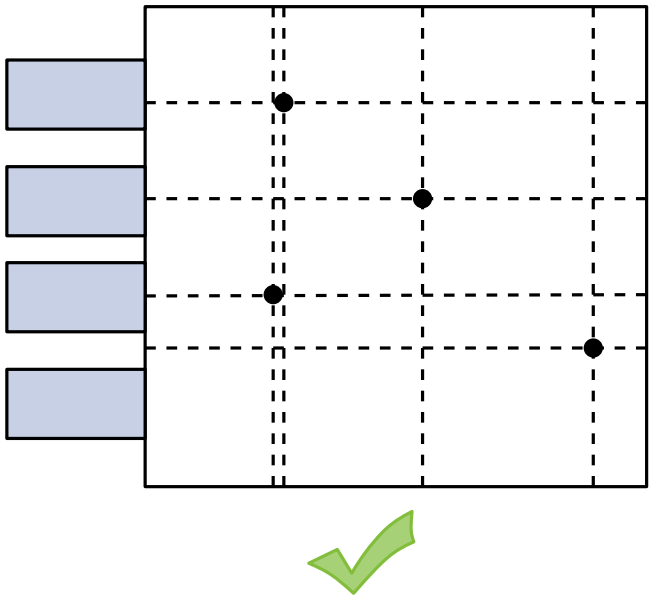
Interferenzminimierung mit Karte

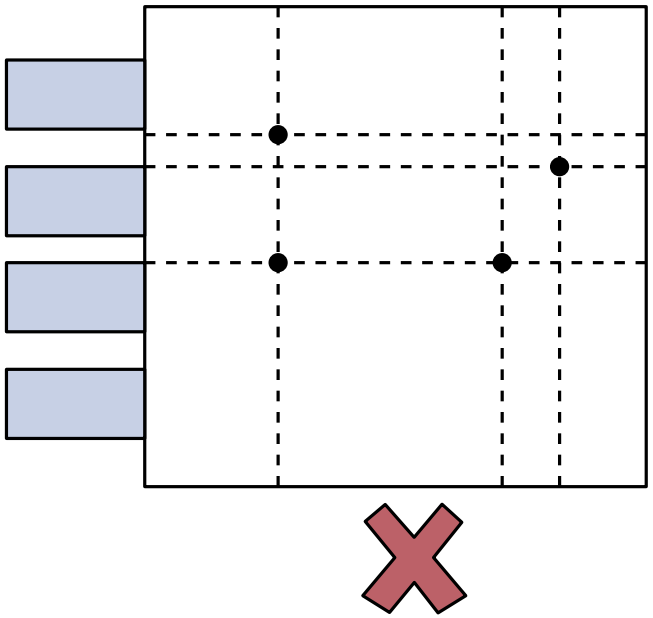


Einschränkung

Annahme: Allgemeine Lage der Punkte in P und der Label in L.

- Keine zwei Punkte in P liegen auf einer gemeinsamen vertikalen oder horizontalen Geraden.
- Kein Punkt in P liegt auf einer horizontalen Geraden, welche die obere oder untere Seite eines Labels in L verlängert.

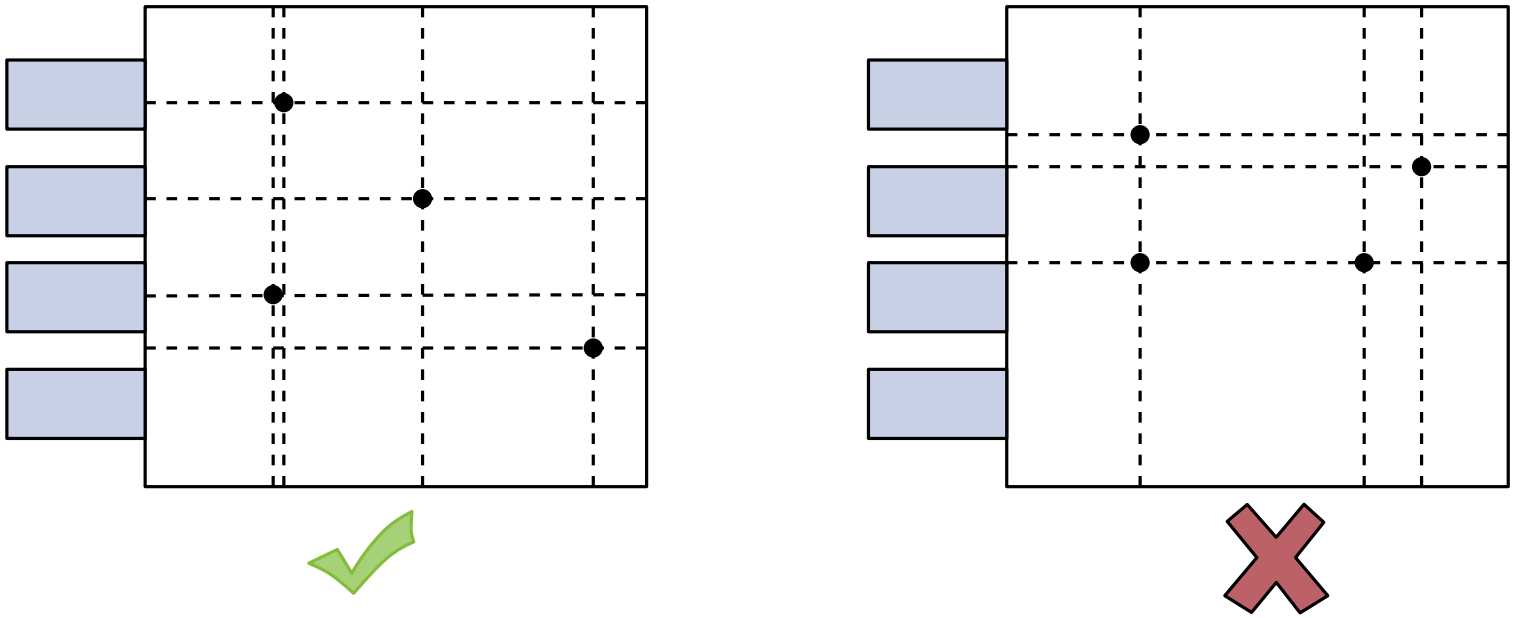




Einschränkung

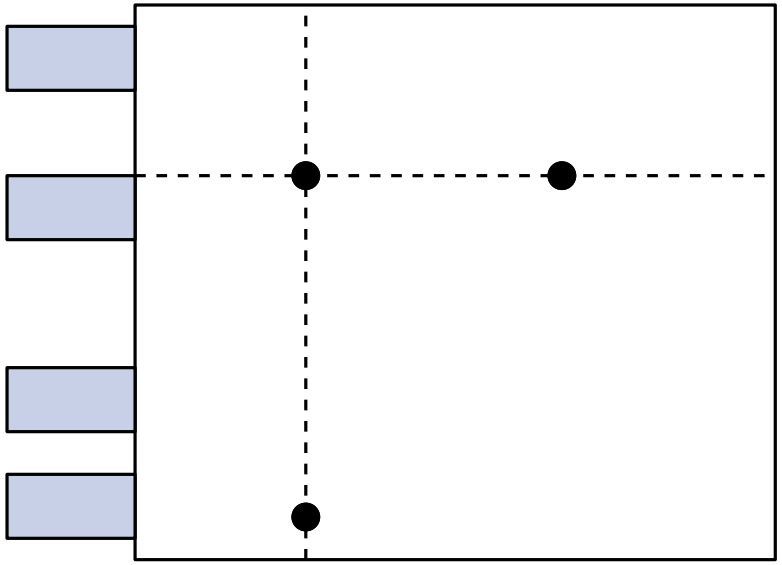
Annahme: Allgemeine Lage der Punkte in P und der Label in L.

- Keine zwei Punkte in P liegen auf einer gemeinsamen vertikalen oder horizontalen Geraden.
- Kein Punkt in P liegt auf einer horizontalen Geraden, welche die obere oder untere Seite eines Labels in L verlängert.

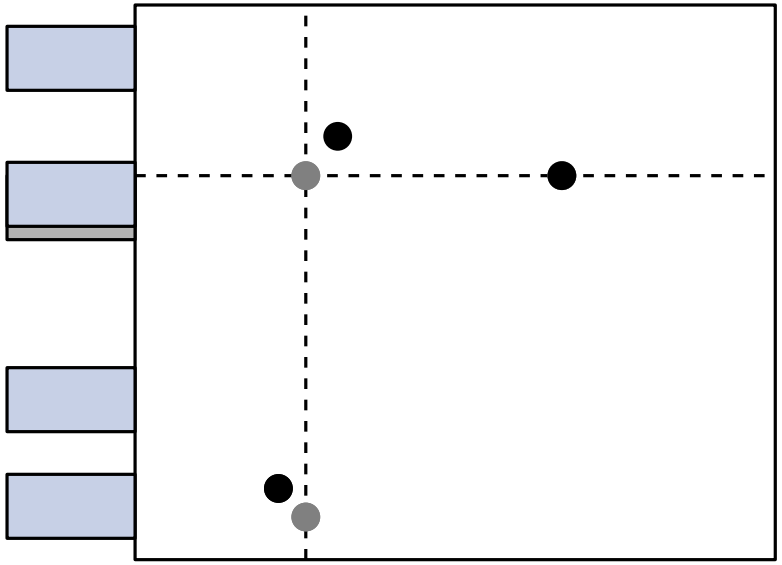


Wie kann Einschränkung aufgehoben werden?

Lösung

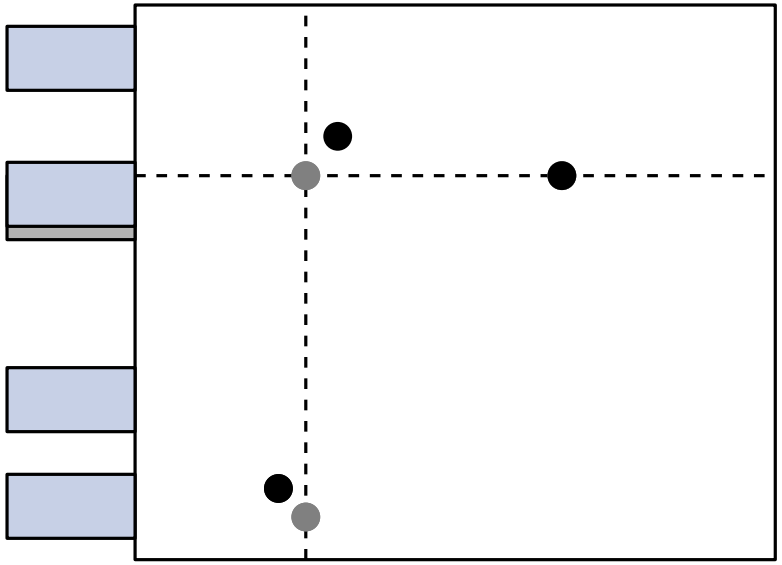


Lösung



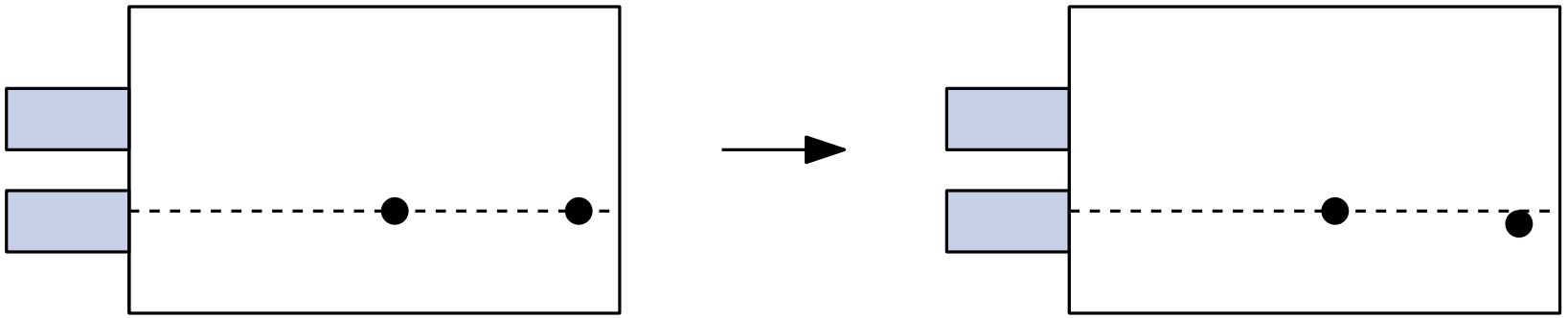
Füge *Störungen* ein: Bewege Punkte und Labels um kleine Distanz ϵ.

Lösung

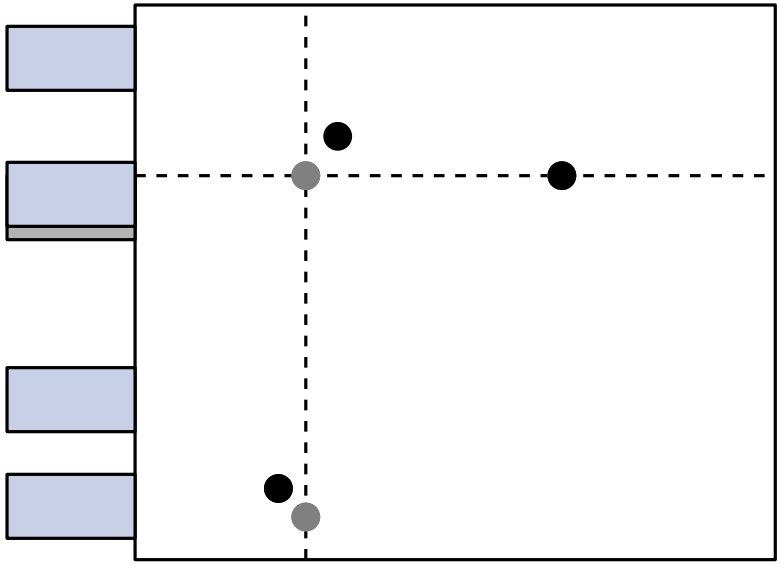


Füge *Störungen* ein: Bewege Punkte und Labels um kleine Distanz ϵ.

Problem: Es können unerwünschte Lösungen entstehen.

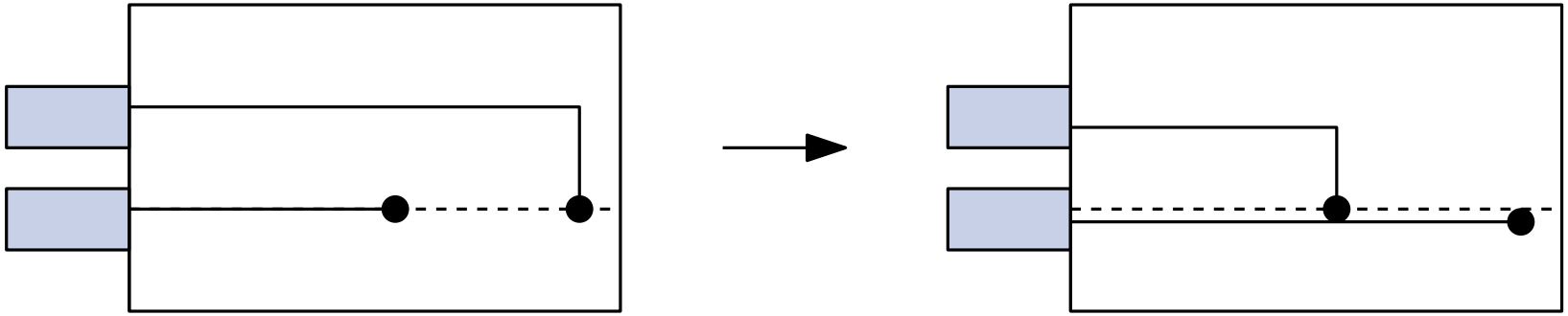


Lösung



Füge *Störungen* ein: Bewege Punkte und Labels um kleine Distanz ϵ.

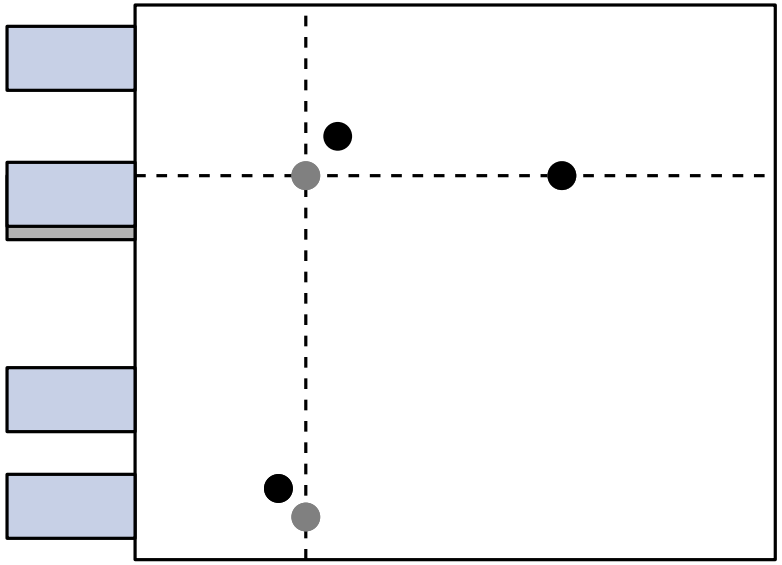
Problem: Es können unerwünschte Lösungen entstehen.



Nur eine Lösung möglich.

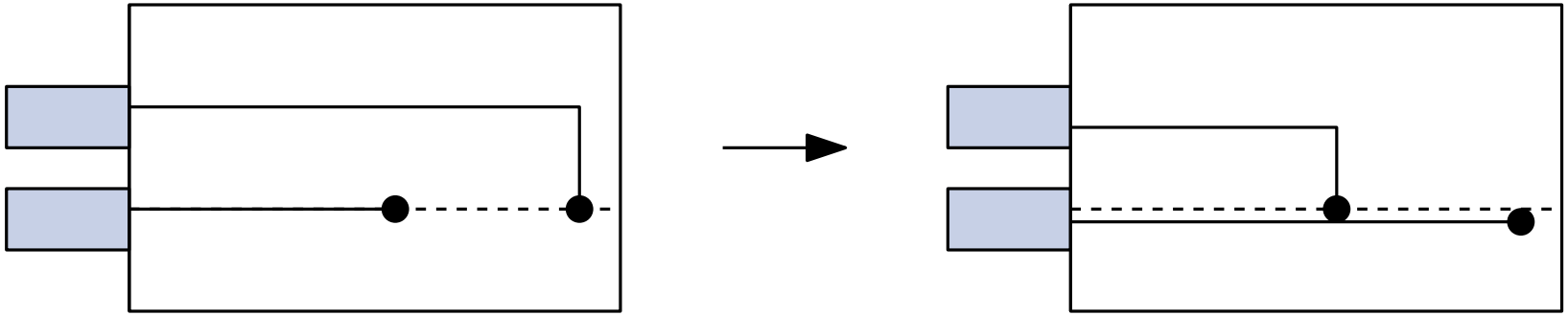
Leader führt sehr nahe an Punkt vorbei.

Lösung



Füge *Störungen* ein: Bewege Punkte und Labels um kleine Distanz ϵ.

Problem: Es können unerwünschte Lösungen entstehen.



Nur eine Lösung möglich.

Leader führt sehr nahe an Punkt vorbei.

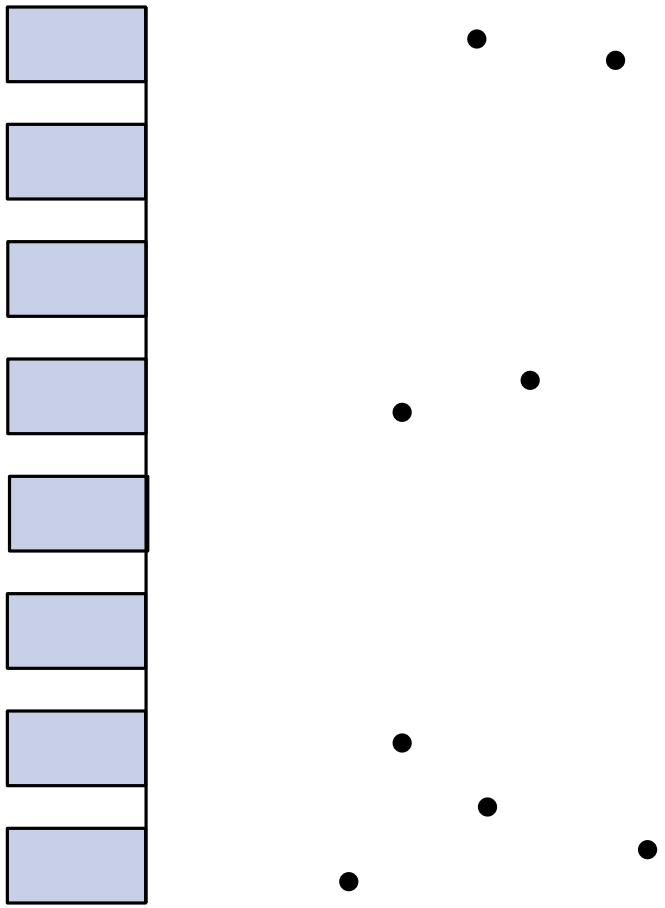
Lösung: Bestrafe Leader mit geringem Abstand zu Punkten.

Sweep-Line-Verfahren

Jetzt: Verfahren für Minimierung der Leader-Längen in $O(n \log n)$ Zeit.

Vorgehen in zwei Phasen:

1. Teile Instanz in unabhängige Teilinstanzen auf.
2. Löse Teilinstanzen mithilfe eines Sweep-Line-Verfahrens.

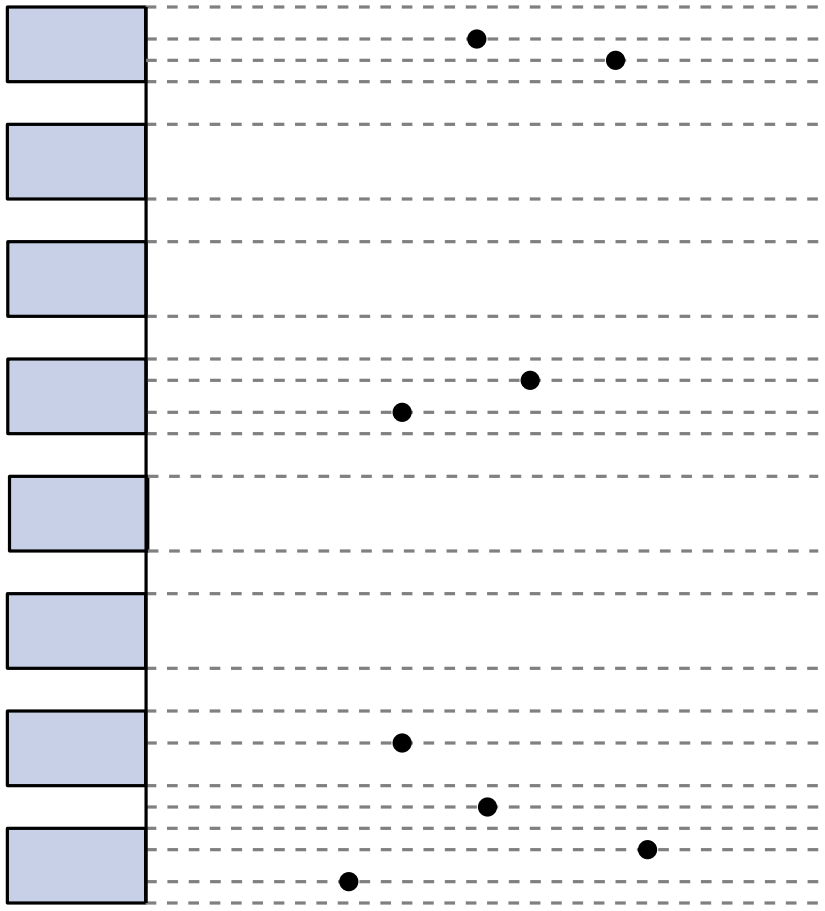


Sweep-Line-Verfahren

Jetzt: Verfahren für Minimierung der Leader-Längen in $O(n \log n)$ Zeit.

Vorgehen in zwei Phasen:

1. Teile Instanz in unabhängige Teilinstanzen auf.
2. Löse Teilinstanzen mithilfe eines Sweep-Line-Verfahrens.

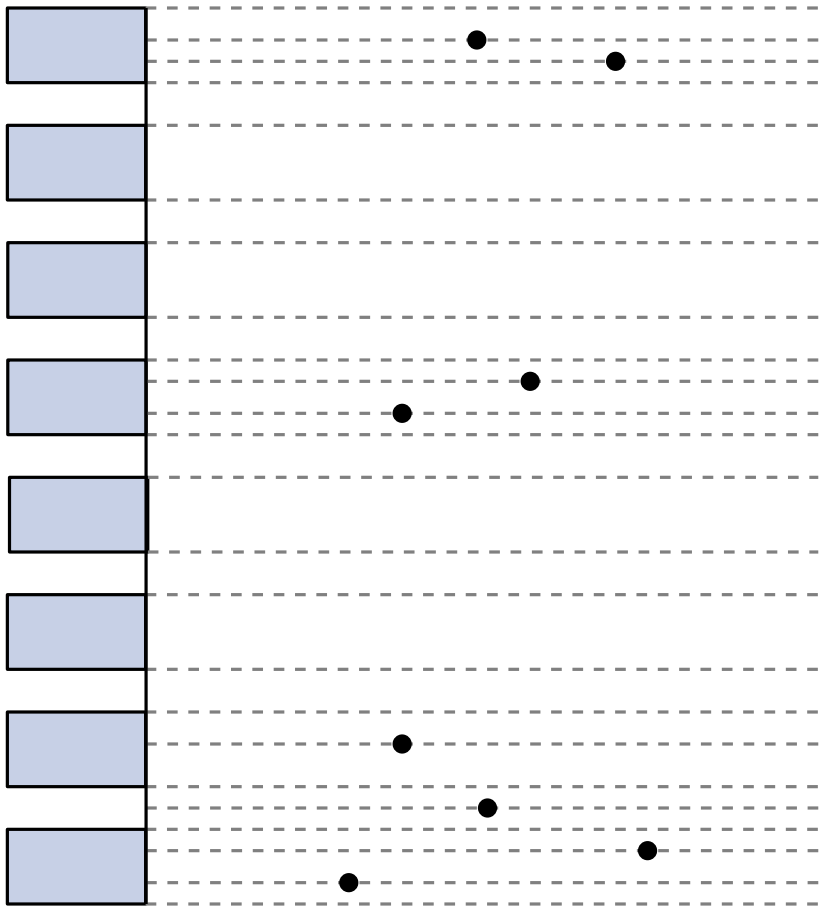


Unterteile hierzu Instanz in Streifen induziert durch Punkte in P und horizontale Seiten der Label.

Sweep-Line-Verfahren

1. Phase: Teile Instanz in unabhängige Teilinstanzen auf.

Traversiere Streifen von unten nach oben und bestimme folgende Anzahlen für jeden Streifen σ:



pa_σ =#Punkte oberhalb von σ inkl. Punkte auf oberer Kante von σ.

ℓa_σ =#Label oberhalb von σ inkl. Label die σ schneiden.

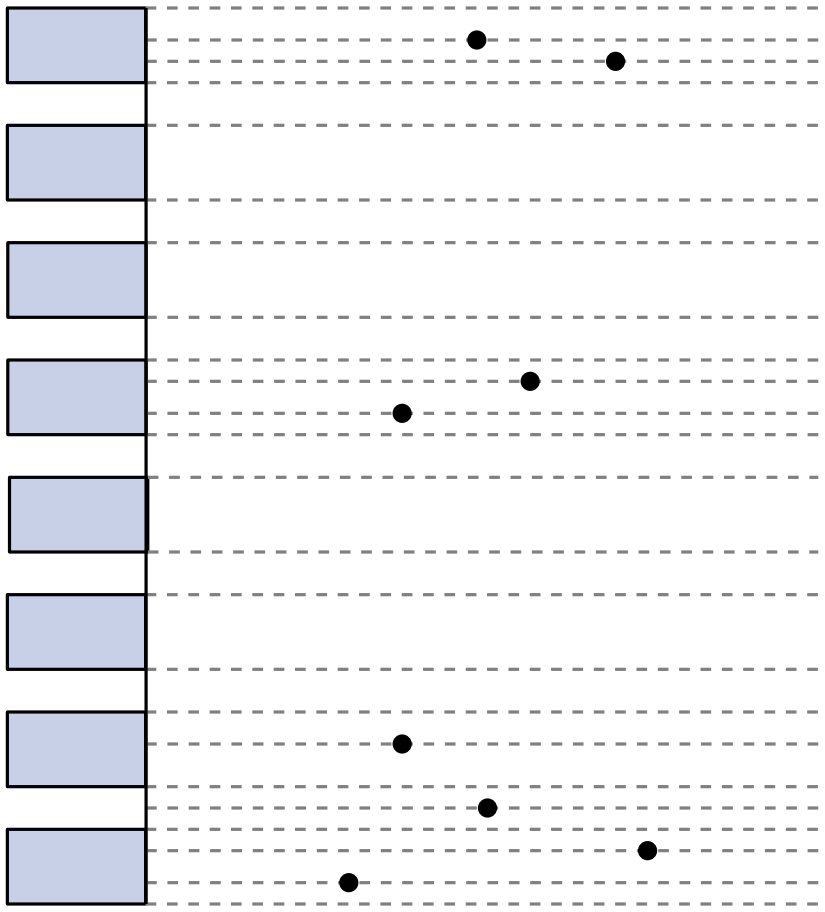
pb_σ =#Punkte unterhalb von σ inkl. Punkte auf unterer Kante von σ.

ℓb_σ =#Label unterhalb von σ inkl. Label die σ schneiden.

Sweep-Line-Verfahren

1. Phase: Teile Instanz in unabhängige Teilinstanzen auf.

Traversiere Streifen von unten nach oben und bestimme folgende Anzahlen für jeden Streifen σ:



pa_σ =#Punkte oberhalb von σ inkl. Punkte auf oberer Kante von σ.

ℓa_σ =#Label oberhalb von σ inkl. Label die σ schneiden.

pb_σ =#Punkte unterhalb von σ inkl. Punkte auf unterer Kante von σ.

ℓb_σ =#Label unterhalb von σ inkl. Label die σ schneiden.

Ein Streifen σ heißt:

absteigend, falls $pa_\sigma > \ell a_\sigma$

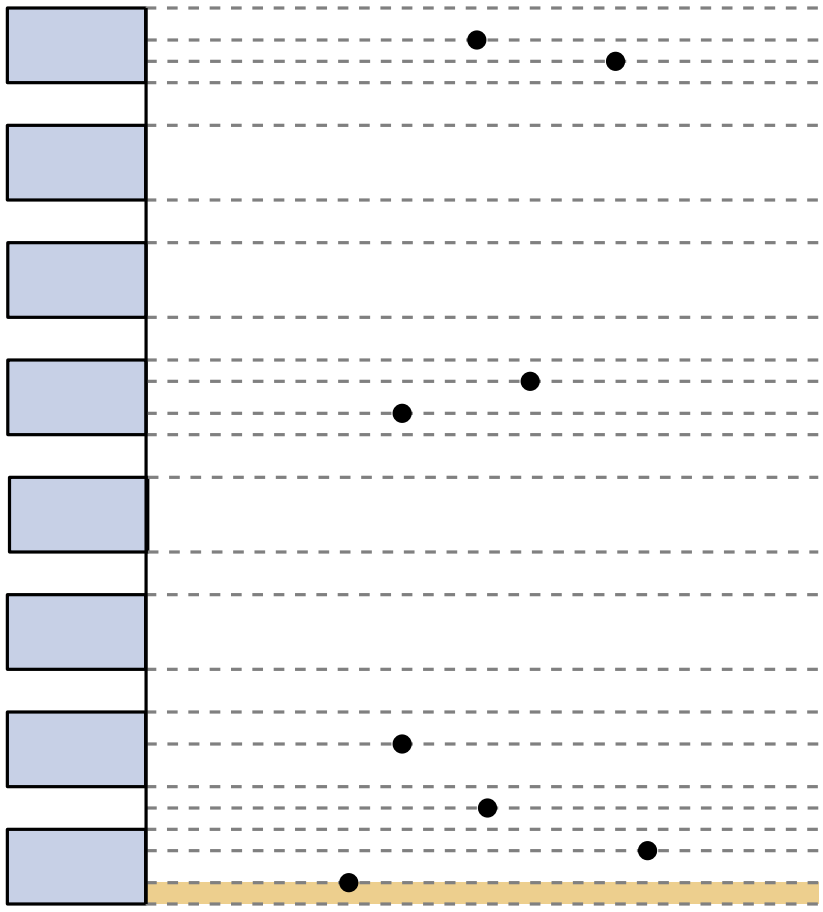
aufsteigend, falls $pb_\sigma > \ell b_\sigma$

neutral, in allen anderen Fällen.

Sweep-Line-Verfahren

1. Phase: Teile Instanz in unabhängige Teilinstanzen auf.

Traversiere Streifen von unten nach oben und bestimme folgende Anzahlen für jeden Streifen σ:



pa_σ =#Punkte oberhalb von σ inkl. Punkte auf oberer Kante von σ.

ℓa_σ =#Label oberhalb von σ inkl. Label die σ schneiden.

pb_σ =#Punkte unterhalb von σ inkl. Punkte auf unterer Kante von σ.

ℓb_σ =#Label unterhalb von σ inkl. Label die σ schneiden.

Ein Streifen σ heißt:

absteigend, falls $pa_\sigma > \ell a_\sigma$

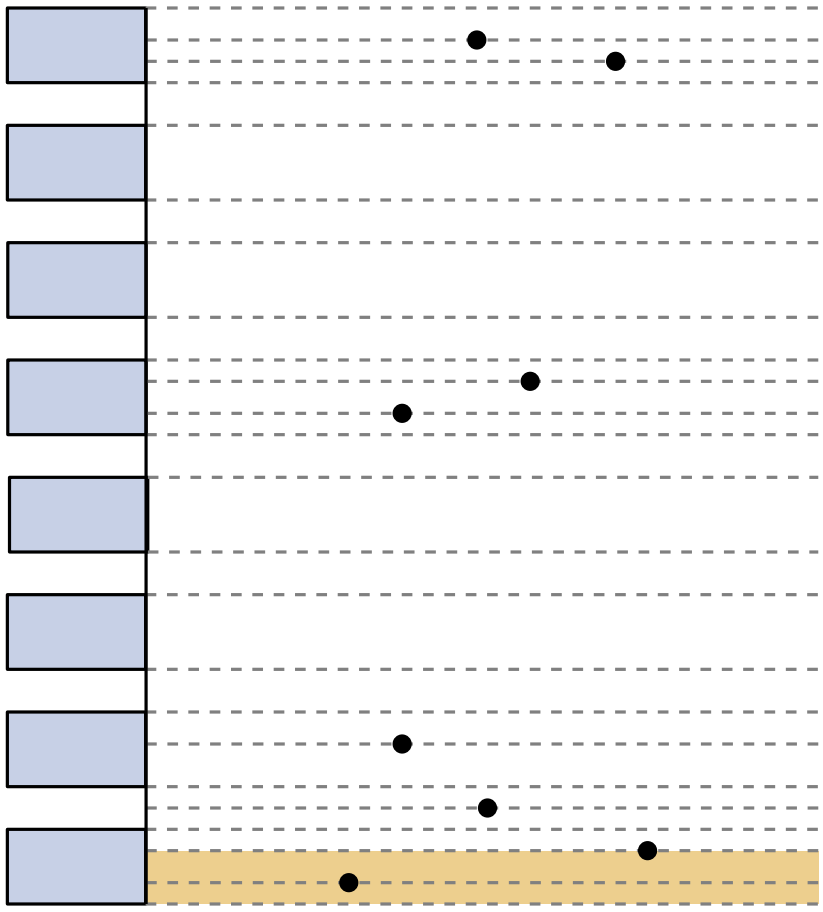
aufsteigend, falls $pb_\sigma > \ell b_\sigma$

neutral, in allen anderen Fällen.

Sweep-Line-Verfahren

1. Phase: Teile Instanz in unabhängige Teilinstanzen auf.

Traversiere Streifen von unten nach oben und bestimme folgende Anzahlen für jeden Streifen σ:



pa_σ =#Punkte oberhalb von σ inkl. Punkte auf oberer Kante von σ.

ℓa_σ =#Label oberhalb von σ inkl. Label die σ schneiden.

pb_σ =#Punkte unterhalb von σ inkl. Punkte auf unterer Kante von σ.

ℓb_σ =#Label unterhalb von σ inkl. Label die σ schneiden.

Ein Streifen σ heißt:

absteigend, falls $pa_\sigma > \ell a_\sigma$

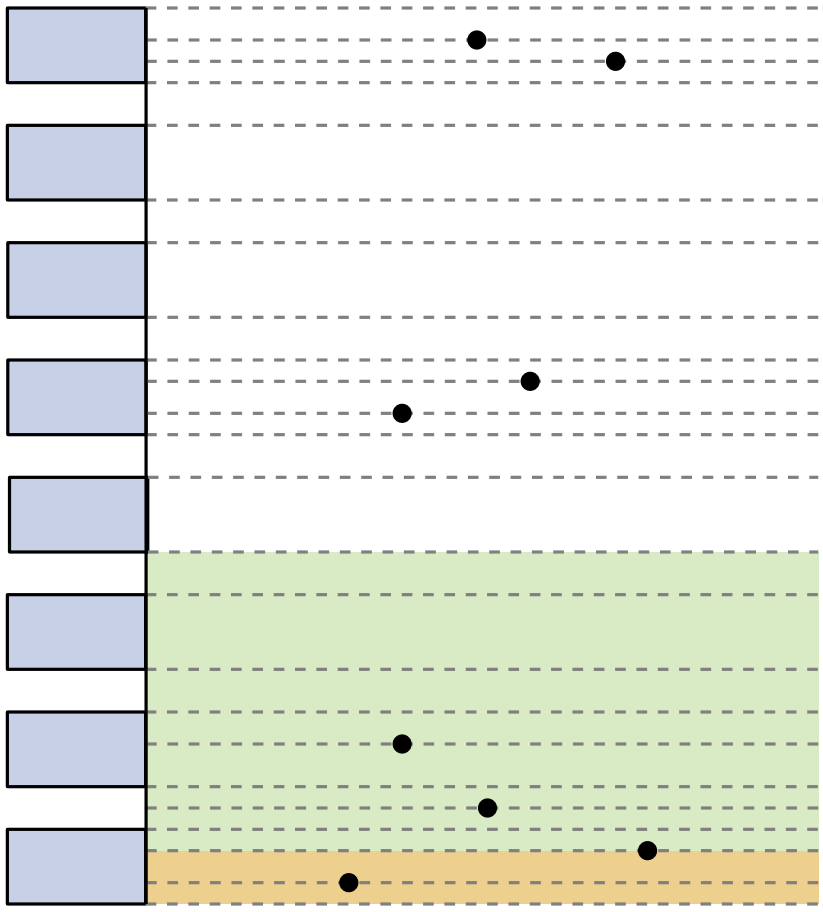
aufsteigend, falls $pb_\sigma > \ell b_\sigma$

neutral, in allen anderen Fällen.

Sweep-Line-Verfahren

1. Phase: Teile Instanz in unabhängige Teilinstanzen auf.

Traversiere Streifen von unten nach oben und bestimme folgende Anzahlen für jeden Streifen σ:



pa_σ =#Punkte oberhalb von σ inkl. Punkte auf oberer Kante von σ.

ℓa_σ =#Label oberhalb von σ inkl. Label die σ schneiden.

pb_σ =#Punkte unterhalb von σ inkl. Punkte auf unterer Kante von σ.

ℓb_σ =#Label unterhalb von σ inkl. Label die σ schneiden.

Ein Streifen σ heißt:

absteigend, falls $pa_\sigma > \ell a_\sigma$

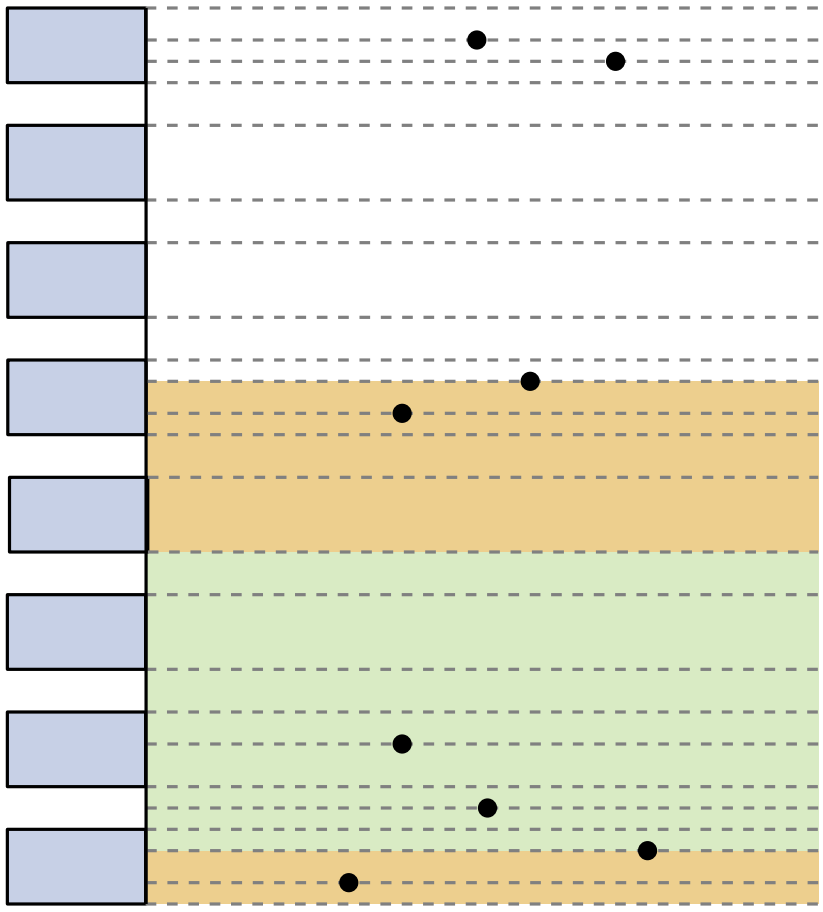
aufsteigend, falls $pb_\sigma > \ell b_\sigma$

neutral, in allen anderen Fällen.

Sweep-Line-Verfahren

1. Phase: Teile Instanz in unabhängige Teilinstanzen auf.

Traversiere Streifen von unten nach oben und bestimme folgende Anzahlen für jeden Streifen σ:



pa_σ =#Punkte oberhalb von σ inkl. Punkte auf oberer Kante von σ.

ℓa_σ =#Label oberhalb von σ inkl. Label die σ schneiden.

pb_σ =#Punkte unterhalb von σ inkl. Punkte auf unterer Kante von σ.

ℓb_σ =#Label unterhalb von σ inkl. Label die σ schneiden.

Ein Streifen σ heißt:

absteigend, falls $pa_\sigma > \ell a_\sigma$

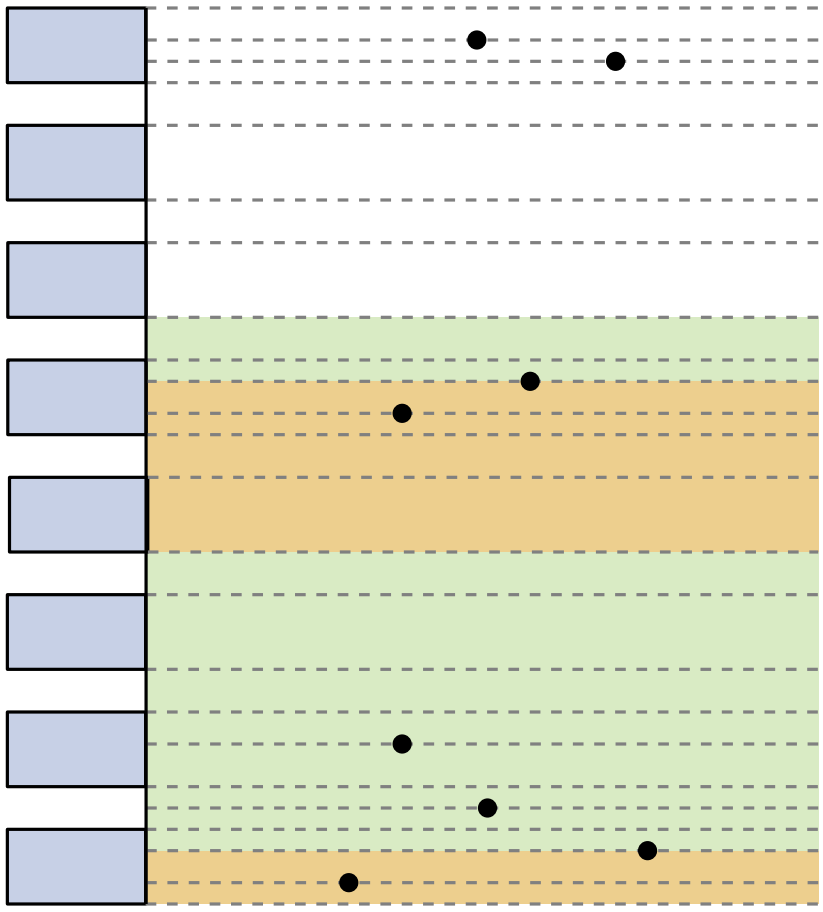
aufsteigend, falls $pb_\sigma > \ell b_\sigma$

neutral, in allen anderen Fällen.

Sweep-Line-Verfahren

1. Phase: Teile Instanz in unabhängige Teilinstanzen auf.

Traversiere Streifen von unten nach oben und bestimme folgende Anzahlen für jeden Streifen σ:



pa_σ =#Punkte oberhalb von σ inkl. Punkte auf oberer Kante von σ.

ℓa_σ =#Label oberhalb von σ inkl. Label die σ schneiden.

pb_σ =#Punkte unterhalb von σ inkl. Punkte auf unterer Kante von σ.

ℓb_σ =#Label unterhalb von σ inkl. Label die σ schneiden.

Ein Streifen σ heißt:

absteigend, falls $pa_\sigma > \ell a_\sigma$

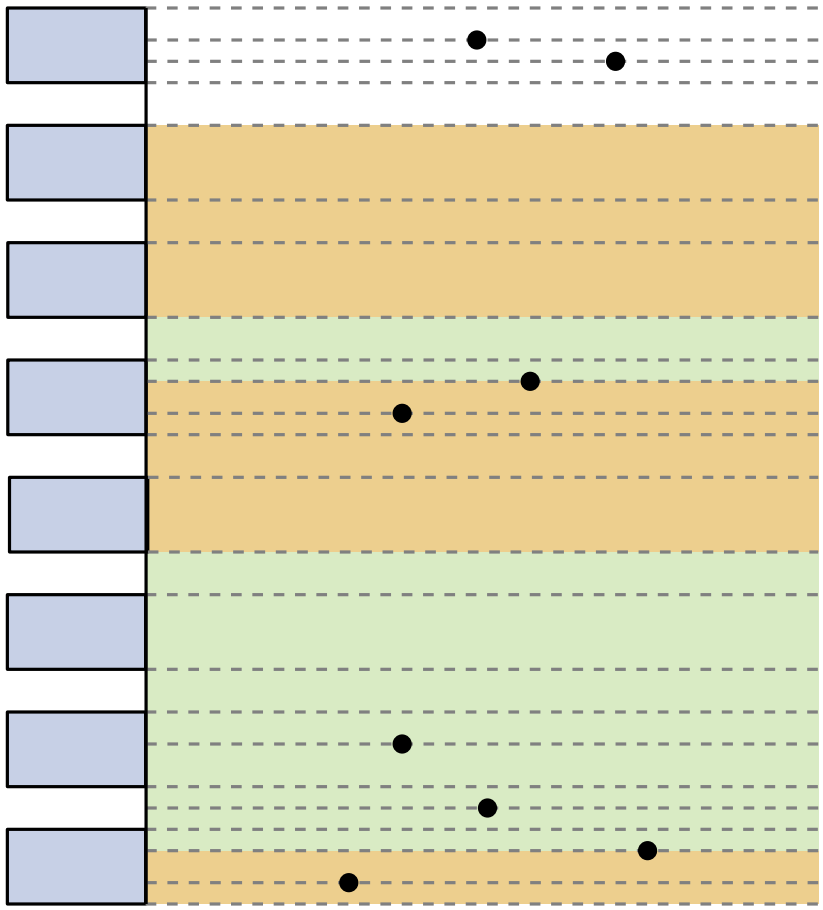
aufsteigend, falls $pb_\sigma > \ell b_\sigma$

neutral, in allen anderen Fällen.

Sweep-Line-Verfahren

1. Phase: Teile Instanz in unabhängige Teilinstanzen auf.

Traversiere Streifen von unten nach oben und bestimme folgende Anzahlen für jeden Streifen σ:



pa_σ =#Punkte oberhalb von σ inkl. Punkte auf oberer Kante von σ.

ℓa_σ =#Label oberhalb von σ inkl. Label die σ schneiden.

pb_σ =#Punkte unterhalb von σ inkl. Punkte auf unterer Kante von σ.

ℓb_σ =#Label unterhalb von σ inkl. Label die σ schneiden.

Ein Streifen σ heißt:

absteigend, falls $pa_\sigma > \ell a_\sigma$

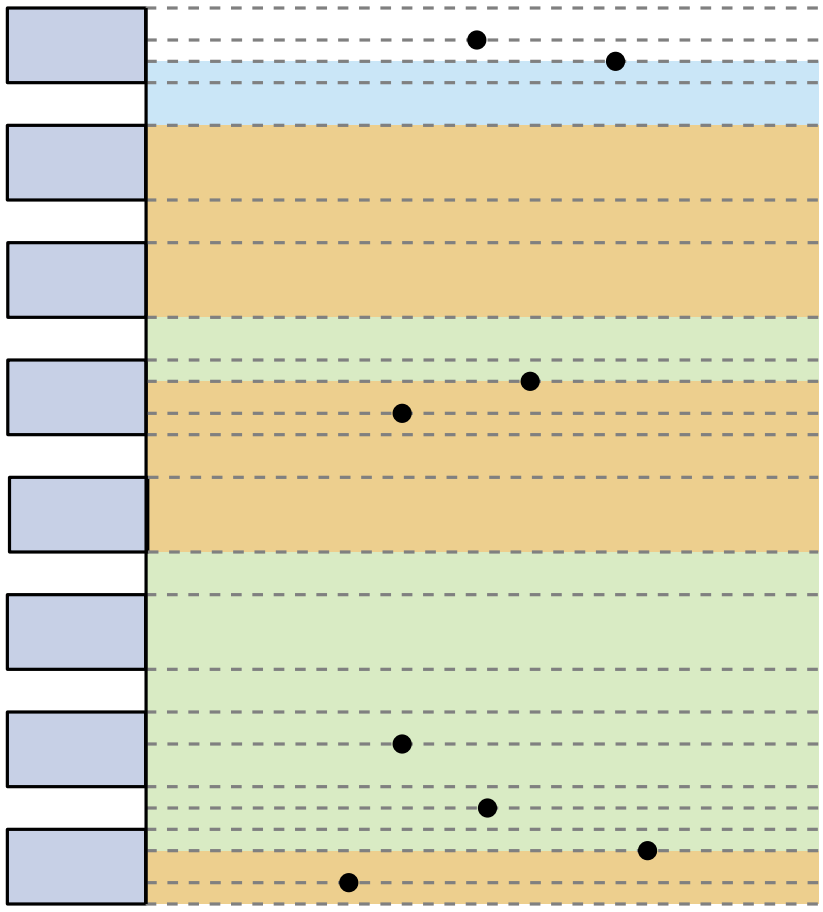
aufsteigend, falls $pb_\sigma > \ell b_\sigma$

neutral, in allen anderen Fällen.

Sweep-Line-Verfahren

1. Phase: Teile Instanz in unabhängige Teilinstanzen auf.

Traversiere Streifen von unten nach oben und bestimme folgende Anzahlen für jeden Streifen σ:



pa_σ =#Punkte oberhalb von σ inkl. Punkte auf oberer Kante von σ.

ℓa_σ =#Label oberhalb von σ inkl. Label die σ schneiden.

pb_σ =#Punkte unterhalb von σ inkl. Punkte auf unterer Kante von σ.

ℓb_σ =#Label unterhalb von σ inkl. Label die σ schneiden.

Ein Streifen σ heißt:

absteigend, falls $pa_\sigma > \ell a_\sigma$

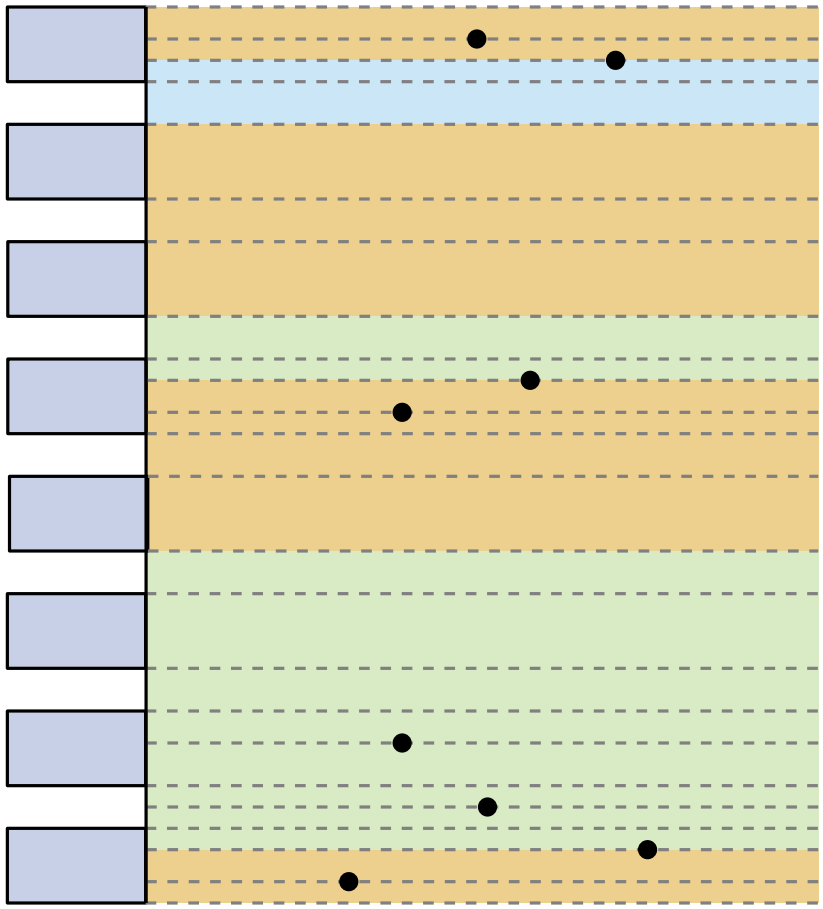
aufsteigend, falls $pb_\sigma > \ell b_\sigma$

neutral, in allen anderen Fällen.

Sweep-Line-Verfahren

1. Phase: Teile Instanz in unabhängige Teilinstanzen auf.

Traversiere Streifen von unten nach oben und bestimme folgende Anzahlen für jeden Streifen σ:



pa_σ =#Punkte oberhalb von σ inkl. Punkte auf oberer Kante von σ.

ℓa_σ =#Label oberhalb von σ inkl. Label die σ schneiden.

pb_σ =#Punkte unterhalb von σ inkl. Punkte auf unterer Kante von σ.

ℓb_σ =#Label unterhalb von σ inkl. Label die σ schneiden.

Ein Streifen σ heißt:

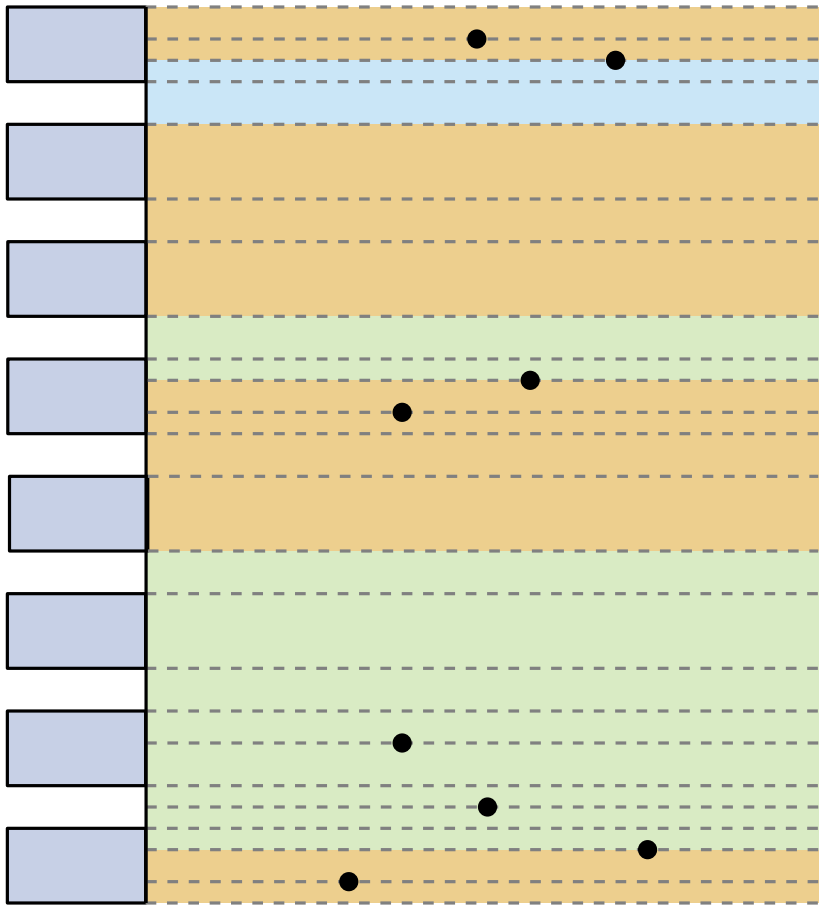
absteigend, falls $pa_\sigma > \ell a_\sigma$

aufsteigend, falls $pb_\sigma > \ell b_\sigma$

neutral, in allen anderen Fällen.

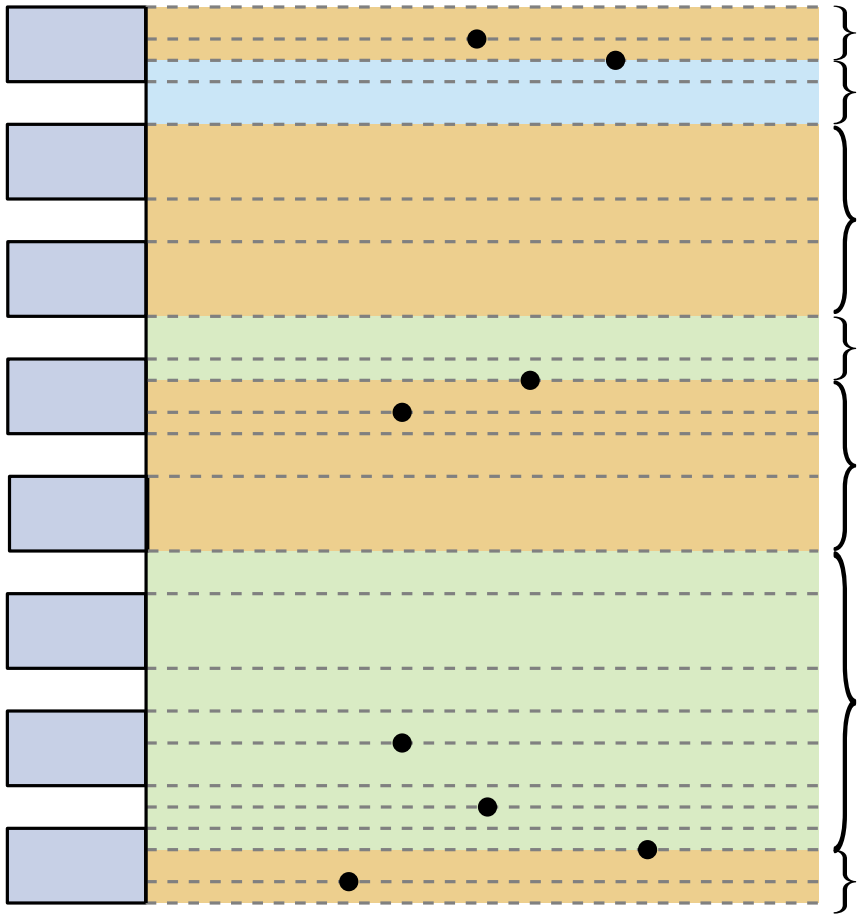
Sweep-Line-Verfahren

2. Phase: Löse Teilinstanzen mithilfe eines Sweep-Line-Verfahrens.



Sweep-Line-Verfahren

2. Phase: Löse Teilinstanzen mithilfe eines Sweep-Line-Verfahrens.



absteigende Instanz: inklusive der Grenzen

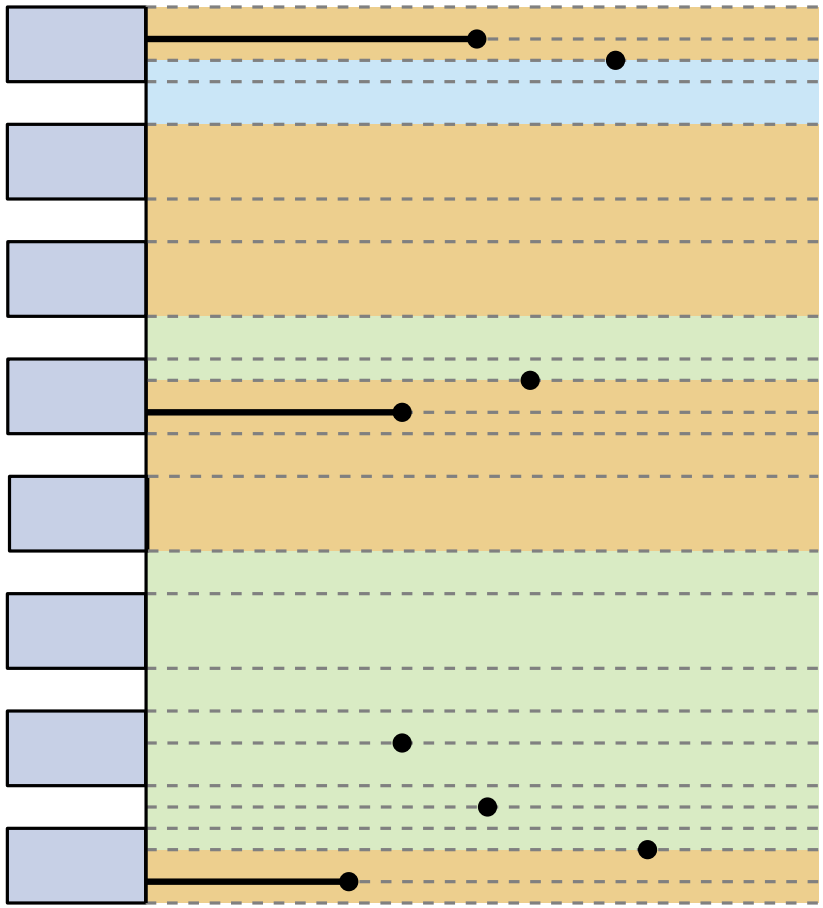
neutrale Instanz: exklusive der Grenzen

aufsteigende Instanz: inklusive der Grenzen

Sweep-Line-Verfahren

2. Phase: Löse Teilinstanzen mithilfe eines Sweep-Line-Verfahrens.

Neutrale Instanz: Verbinde Punkte und Label mit horizontaler Strecke.

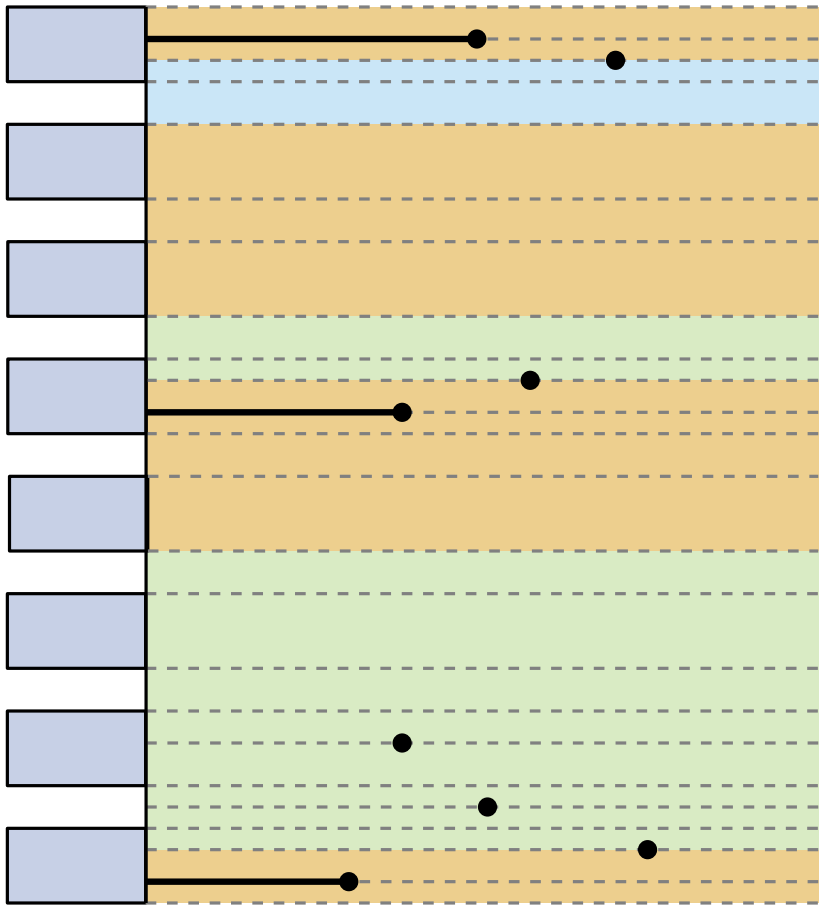


Sweep-Line-Verfahren

2. Phase: Löse Teilinstanzen mithilfe eines Sweep-Line-Verfahrens.

Neutrale Instanz: Verbinde Punkte und Label mit horizontaler Strecke.

Aufsteigende Instanz: Sweep-Line-Verfahren von unten nach oben.



Sweep-Line stoppt bei unterer Kante der Label und bei Punkten:

Punkt-Event p:
Füge p zu einer nach x-Koordinanten sortierten Warteliste W hinzu.

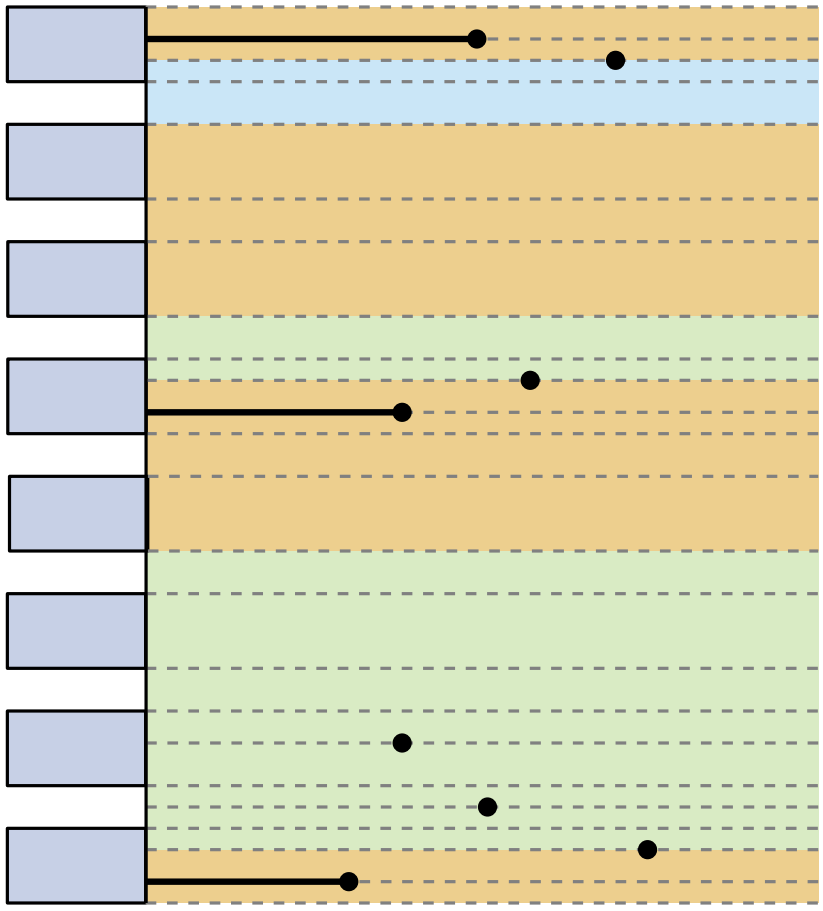
Label-Event ℓ:
Entferne Punkt p aus W, der am weitesten links liegt. Verbinde p mit ℓ mittels kürzesten Leader.

Sweep-Line-Verfahren

2. Phase: Löse Teilinstanzen mithilfe eines Sweep-Line-Verfahrens.

Neutrale Instanz: Verbinde Punkte und Label mit horizontaler Strecke.

Aufsteigende Instanz: Sweep-Line-Verfahren von unten nach oben.



Sweep-Line stoppt bei unterer Kante der Label und bei Punkten:

Punkt-Event p:
Füge p zu einer nach x-Koordinanten sortierten Warteliste W hinzu.

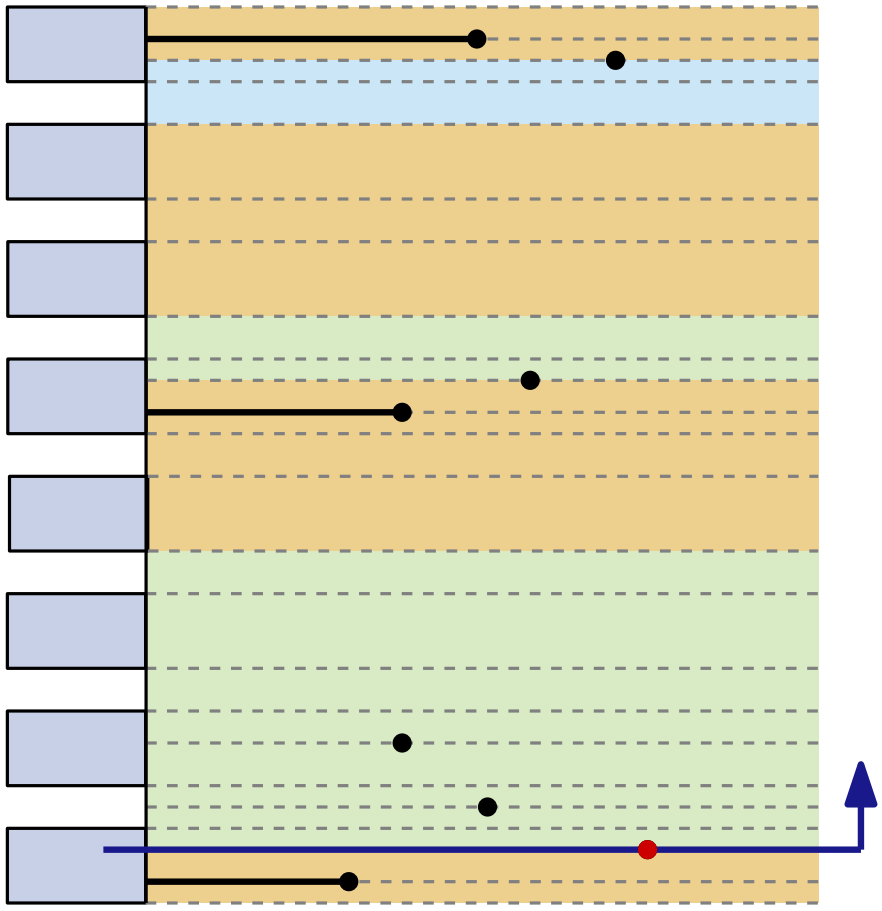
Label-Event ℓ:
Entferne Punkt p aus W, der am weitesten links liegt. Verbinde p mit ℓ mittels kürzesten Leader.

Sweep-Line-Verfahren

2. Phase: Löse Teilinstanzen mithilfe eines Sweep-Line-Verfahrens.

Neutrale Instanz: Verbinde Punkte und Label mit horizontaler Strecke.

Aufsteigende Instanz: Sweep-Line-Verfahren von unten nach oben.



Sweep-Line stoppt bei unterer Kante der Label und bei Punkten:

Punkt-Event p:
Füge p zu einer nach x-Koordinanten sortierten Warteliste W hinzu.

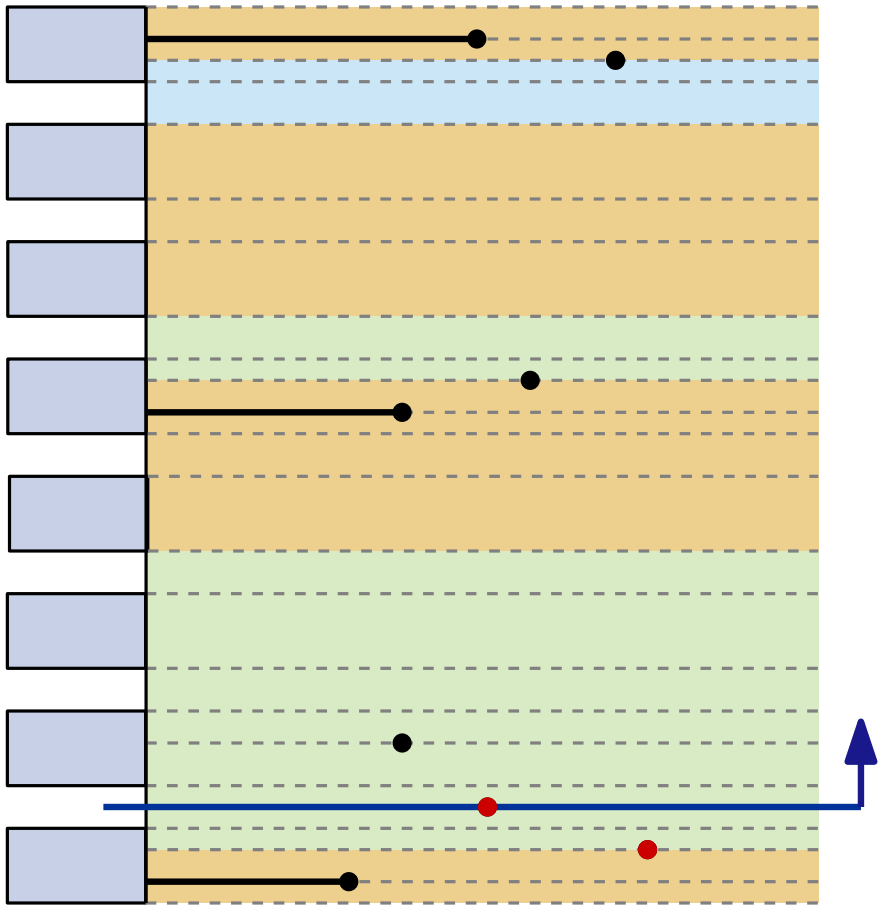
Label-Event ℓ:
Entferne Punkt p aus W, der am weitesten links liegt. Verbinde p mit ℓ mittels kürzesten Leader.

Sweep-Line-Verfahren

2. Phase: Löse Teilinstanzen mithilfe eines Sweep-Line-Verfahrens.

Neutrale Instanz: Verbinde Punkte und Label mit horizontaler Strecke.

Aufsteigende Instanz: Sweep-Line-Verfahren von unten nach oben.



Sweep-Line stoppt bei unterer Kante der Label und bei Punkten:

Punkt-Event p:
Füge p zu einer nach x-Koordinanten sortierten Warteliste W hinzu.

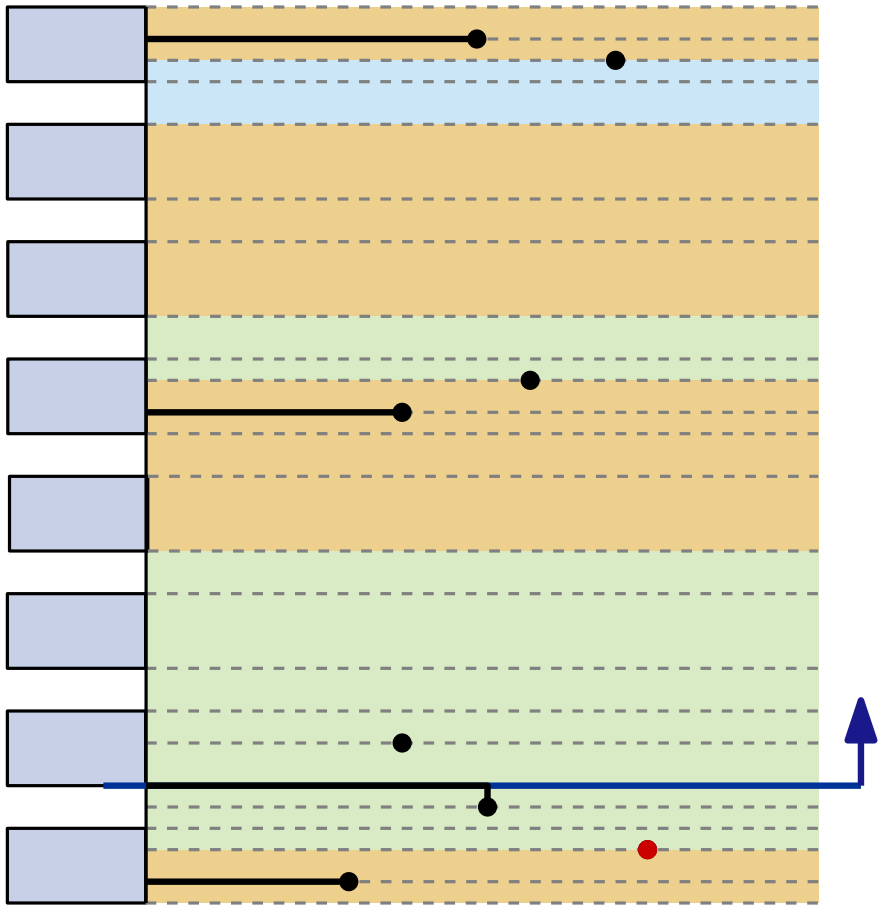
Label-Event ℓ:
Entferne Punkt p aus W, der am weitesten links liegt. Verbinde p mit ℓ mittels kürzesten Leader.

Sweep-Line-Verfahren

2. Phase: Löse Teilinstanzen mithilfe eines Sweep-Line-Verfahrens.

Neutrale Instanz: Verbinde Punkte und Label mit horizontaler Strecke.

Aufsteigende Instanz: Sweep-Line-Verfahren von unten nach oben.



Sweep-Line stoppt bei unterer Kante der Label und bei Punkten:

Punkt-Event p:
Füge p zu einer nach x-Koordinanten sortierten Warteliste W hinzu.

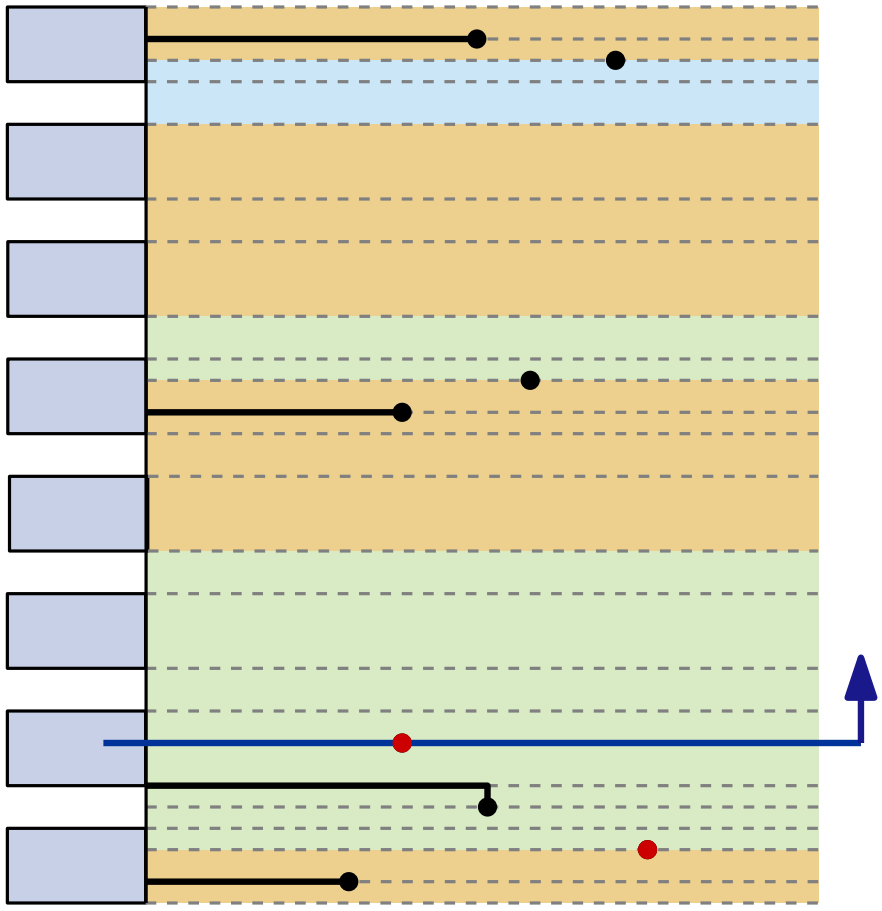
Label-Event ℓ:
Entferne Punkt p aus W, der am weitesten links liegt. Verbinde p mit ℓ mittels kürzesten Leader.

Sweep-Line-Verfahren

2. Phase: Löse Teilinstanzen mithilfe eines Sweep-Line-Verfahrens.

Neutrale Instanz: Verbinde Punkte und Label mit horizontaler Strecke.

Aufsteigende Instanz: Sweep-Line-Verfahren von unten nach oben.



Sweep-Line stoppt bei unterer Kante der Label und bei Punkten:

Punkt-Event p:
Füge p zu einer nach x-Koordinanten sortierten Warteliste W hinzu.

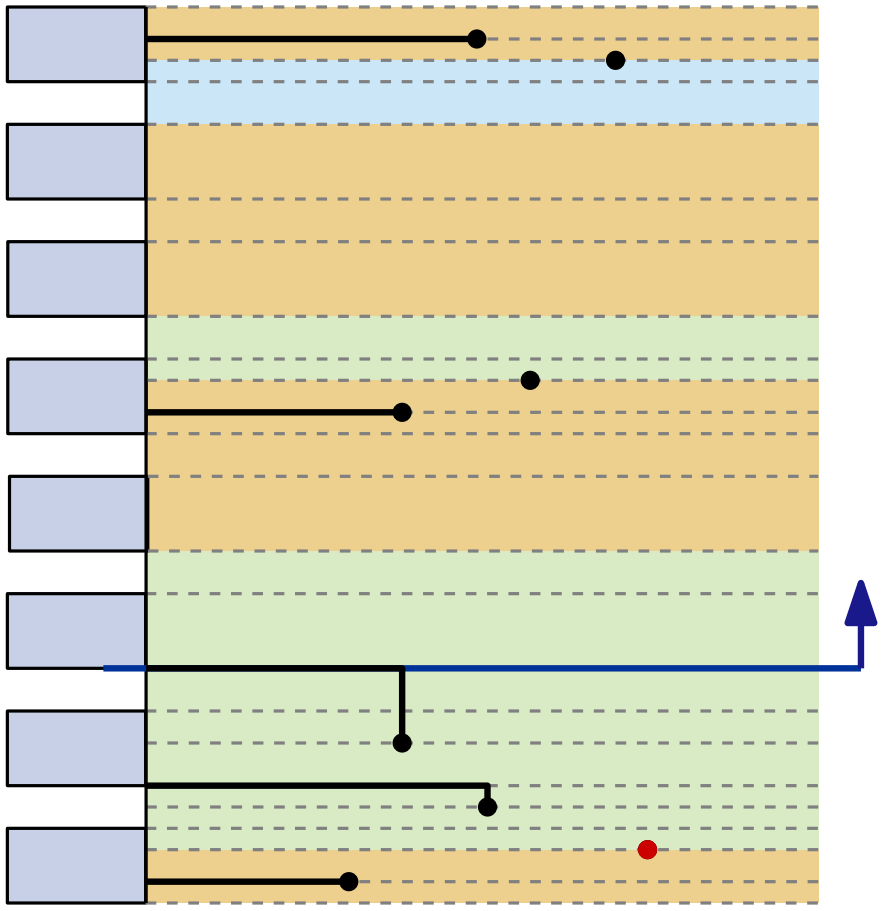
Label-Event ℓ:
Entferne Punkt p aus W, der am weitesten links liegt. Verbinde p mit ℓ mittels kürzesten Leader.

Sweep-Line-Verfahren

2. Phase: Löse Teilinstanzen mithilfe eines Sweep-Line-Verfahrens.

Neutrale Instanz: Verbinde Punkte und Label mit horizontaler Strecke.

Aufsteigende Instanz: Sweep-Line-Verfahren von unten nach oben.



Sweep-Line stoppt bei unterer Kante der Label und bei Punkten:

Punkt-Event p:
Füge p zu einer nach x-Koordinanten sortierten Warteliste W hinzu.

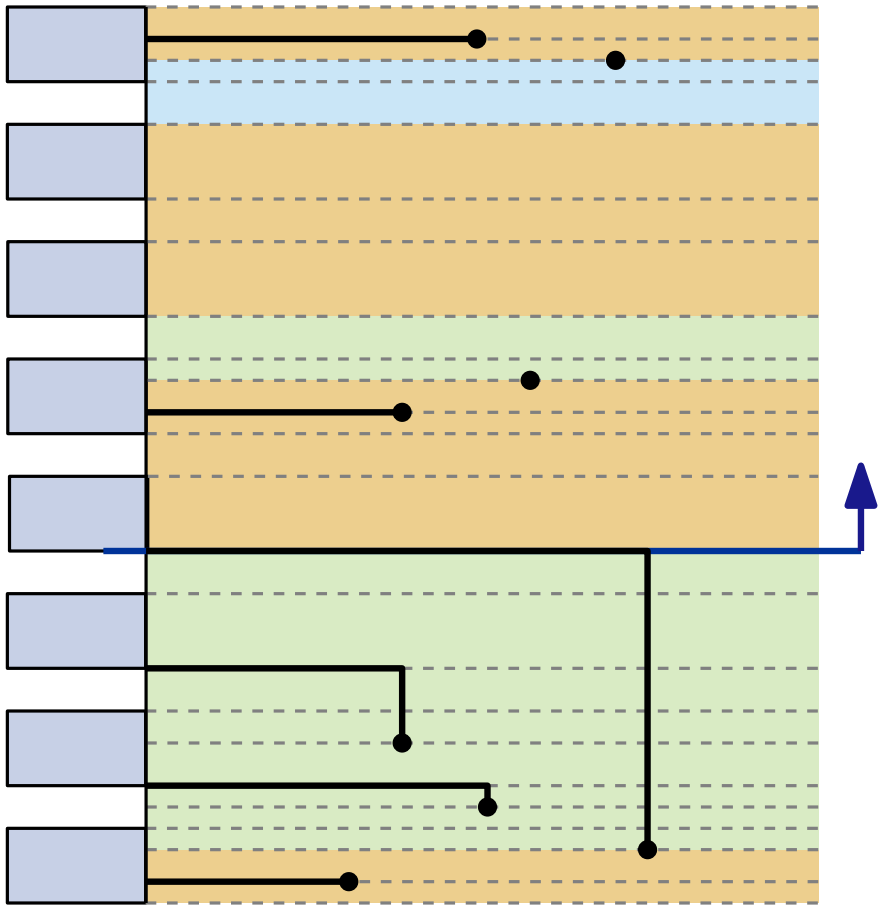
Label-Event ℓ:
Entferne Punkt p aus W, der am weitesten links liegt. Verbinde p mit ℓ mittels kürzesten Leader.

Sweep-Line-Verfahren

2. Phase: Löse Teilinstanzen mithilfe eines Sweep-Line-Verfahrens.

Neutrale Instanz: Verbinde Punkte und Label mit horizontaler Strecke.

Aufsteigende Instanz: Sweep-Line-Verfahren von unten nach oben.



Sweep-Line stoppt bei unterer Kante der Label und bei Punkten:

Punkt-Event p:
Füge p zu einer nach x-Koordinanten sortierten Warteliste W hinzu.

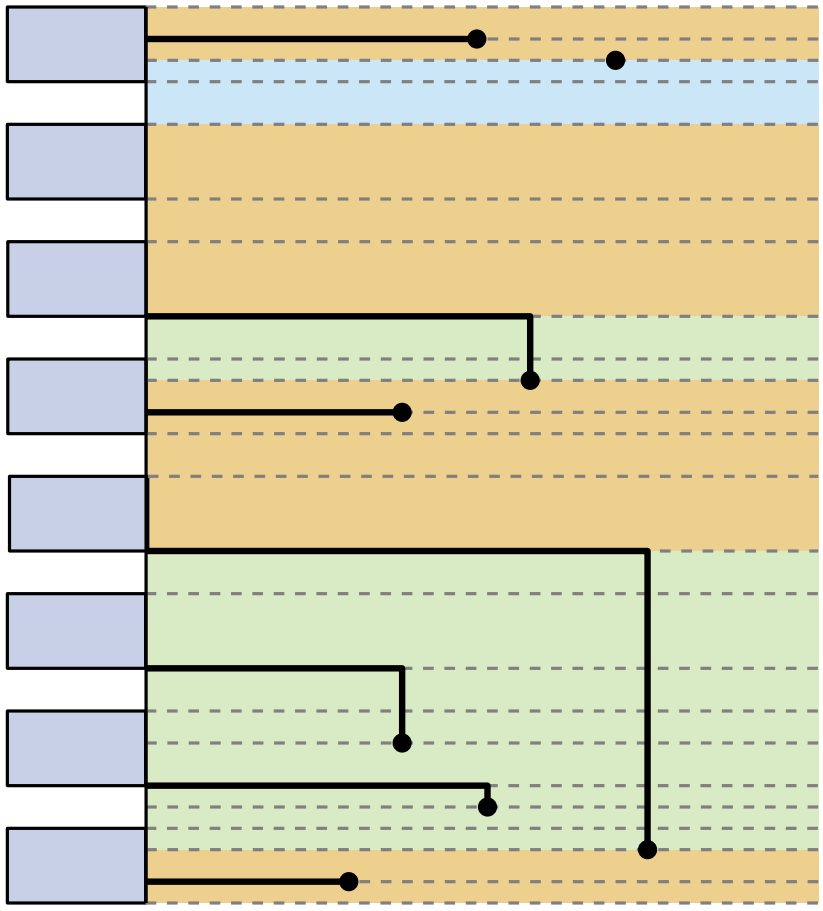
Label-Event ℓ:
Entferne Punkt p aus W, der am weitesten links liegt. Verbinde p mit ℓ mittels kürzesten Leader.

Sweep-Line-Verfahren

2. Phase: Löse Teilinstanzen mithilfe eines Sweep-Line-Verfahrens.

Neutrale Instanz: Verbinde Punkte und Label mit horizontaler Strecke.

Aufsteigende Instanz: Sweep-Line-Verfahren von unten nach oben.



Sweep-Line stoppt bei unterer Kante der Label und bei Punkten:

Punkt-Event p:
Füge p zu einer nach x-Koordinanten sortierten Warteliste W hinzu.

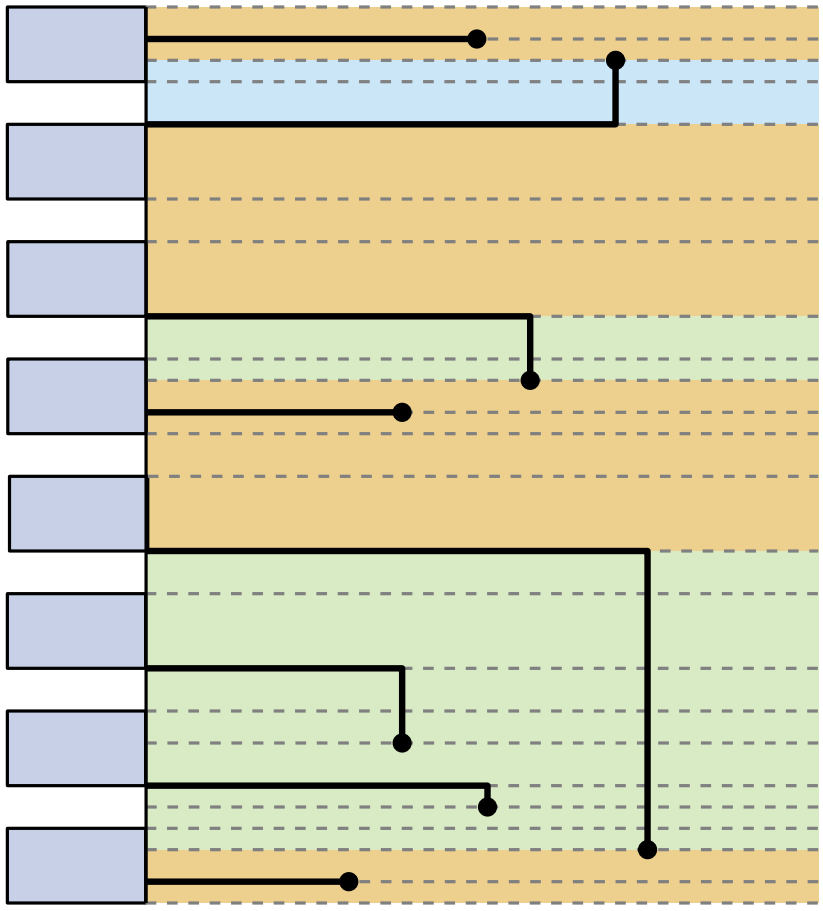
Label-Event ℓ:
Entferne Punkt p aus W, der am weitesten links liegt. Verbinde p mit ℓ mittels kürzesten Leader.

Sweep-Line-Verfahren

2. Phase: Löse Teilinstanzen mithilfe eines Sweep-Line-Verfahrens.

Neutrale Instanz: Verbinde Punkte und Label mit horizontaler Strecke.

Aufsteigende Instanz: Sweep-Line-Verfahren von unten nach oben.



Sweep-Line stoppt bei unterer Kante der Label und bei Punkten:

Punkt-Event p:
Füge p zu einer nach x-Koordinanten sortierten Warteliste W hinzu.

Label-Event ℓ:
Entferne Punkt p aus W, der am weitesten links liegt. Verbinde p mit ℓ mittels kürzesten Leader.

Analoges Vorgehen für absteigende Instanz.

Sweep-Line-Verfahren

Lemma 1: Für jede kreuzungsfreie Zuordnung mit minimaler Länge gilt, dass kein Leader einen neutralen Streifen kreuzt (Übung).

Sweep-Line-Verfahren

Lemma 1: Für jede kreuzungsfreie Zuordnung mit minimaler Länge gilt, dass kein Leader einen neutralen Streifen kreuzt (Übung).

Satz 1: Für jede Zuordnung $\mathcal{L}^*$ mit po-Leadern und minimaler Länge gibt es eine kreuzungsfreie Zuordnung $\mathcal{L}$ mit po-Leadern selber Länge. $\mathcal{L}$ kann mithilfe von $\mathcal{L}^*$ in $O(n^2)$ Zeit konstruiert werden. (Bekos et al. '07 und Benkert et al. '09)

Sweep-Line-Verfahren

Lemma 1: Für jede kreuzungsfreie Zuordnung mit minimaler Länge gilt, dass kein Leader einen neutralen Streifen kreuzt (Übung).

Lemma 2: Für eine Instanz (P, L, R) kann eine kreuzungsfreie Zuordnung minimaler Länge mithilfe von po-Leadern in $O(n \log n)$ Zeit und $O(n)$ Speicher berechnet werden.

Sweep-Line-Verfahren

Lemma 1: Für jede kreuzungsfreie Zuordnung mit minimaler Länge gilt, dass kein Leader einen neutralen Streifen kreuzt (Übung).

Lemma 2: Für eine Instanz (P, L, R) kann eine kreuzungsfreie Zuordnung minimaler Länge mithilfe von po-Leadern in $O(n \log n)$ Zeit und $O(n)$ Speicher berechnet werden.

Lemma 3: Eine kreuzungsfreie Zuordnung minimaler Länge kann nicht schneller als in $\Omega(n \log n)$ Zeit berechnet werden (s. Übung).

Sweep-Line-Verfahren

Lemma 1: Für jede kreuzungsfreie Zuordnung mit minimaler Länge gilt, dass kein Leader einen neutralen Streifen kreuzt (Übung).

Lemma 2: Für eine Instanz (P, L, R) kann eine kreuzungsfreie Zuordnung minimaler Länge mithilfe von po-Leadern in $O(n \log n)$ Zeit und $O(n)$ Speicher berechnet werden.

Lemma 3: Eine kreuzungsfreie Zuordnung minimaler Länge kann nicht schneller als in $\Omega(n \log n)$ Zeit berechnet werden (s. Übung).

Satz 2: Die Berechnung einer kreuzungsfreien Zuordnung minimaler Länge einer Instanz (P, L, R) mit po-Leadern benötigt Laufzeit $\Theta(n \log n)$ Zeit und $\Theta(n)$ Speicher.

Beweis Lemma 3

Lemma 3: Eine kreuzungsfreie Zuordnung minimaler Länge kann nicht schneller als in $\Omega(n \log n)$ Zeit berechnet werden (s. Übung).

Beweis Lemma 3

Lemma 3: Eine kreuzungsfreie Zuordnung minimaler Länge kann nicht schneller als in $\Omega(n \log n)$ Zeit berechnet werden (s. Übung).

Idee: Reduziere Sortieren von n Zahlen auf Problem.

Beweis Lemma 3

Lemma 3: Eine kreuzungsfreie Zuordnung minimaler Länge kann nicht schneller als in $\Omega(n \log n)$ Zeit berechnet werden (s. Übung).

Idee: Reduziere Sortieren von n Zahlen auf Problem.

↳ $\Omega(n \log n)$ Zeit in vergleichsbasierten Rechnermodell.

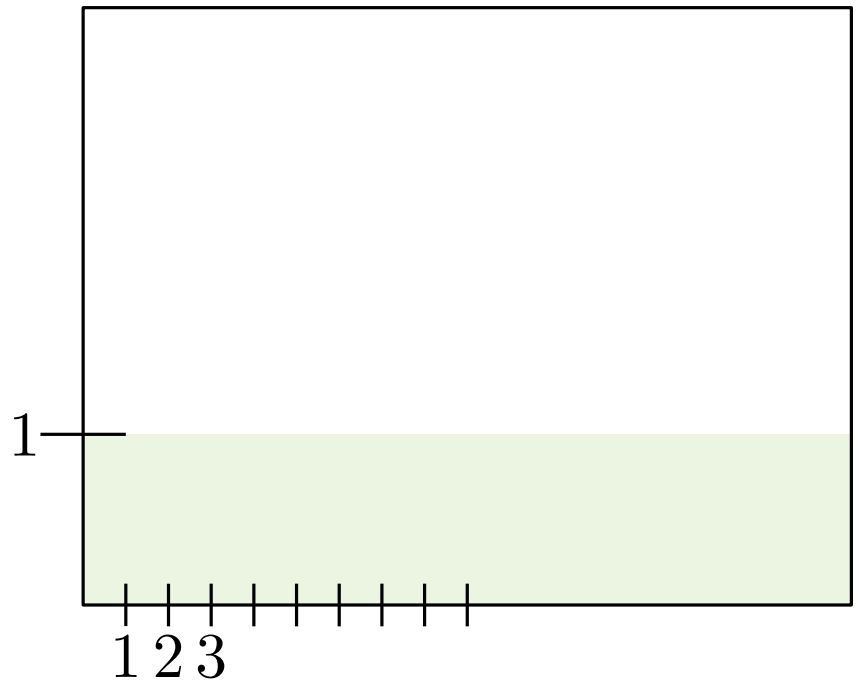
Beweis Lemma 3

Lemma 3: Eine kreuzungsfreie Zuordnung minimaler Länge kann nicht schneller als in $\Omega(n \log n)$ Zeit berechnet werden (s. Übung).

Idee: Reduziere Sortieren von n Zahlen auf Problem.

↳ $\Omega(n \log n)$ Zeit in vergleichsbasierten Rechnermodell.

Gegeben: Zahlen $x_1, \ldots, x_n$



Punkt (x_i, y_i) mit $0 \leq y_i \leq 1$ für jede Zahl x_i.

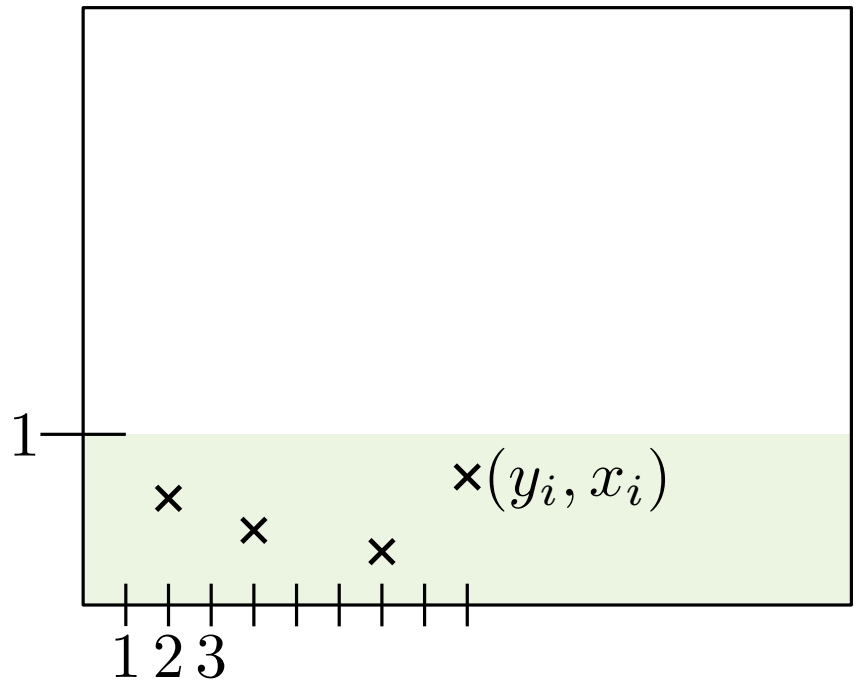
Beweis Lemma 3

Lemma 3: Eine kreuzungsfreie Zuordnung minimaler Länge kann nicht schneller als in $\Omega(n \log n)$ Zeit berechnet werden (s. Übung).

Idee: Reduziere Sortieren von n Zahlen auf Problem.

↳ $\Omega(n \log n)$ Zeit in vergleichsbasierten Rechnermodell.

Gegeben: Zahlen $x_1, \ldots, x_n$



Punkt (x_i, y_i) mit $0 \leq y_i \leq 1$ für jede Zahl x_i.

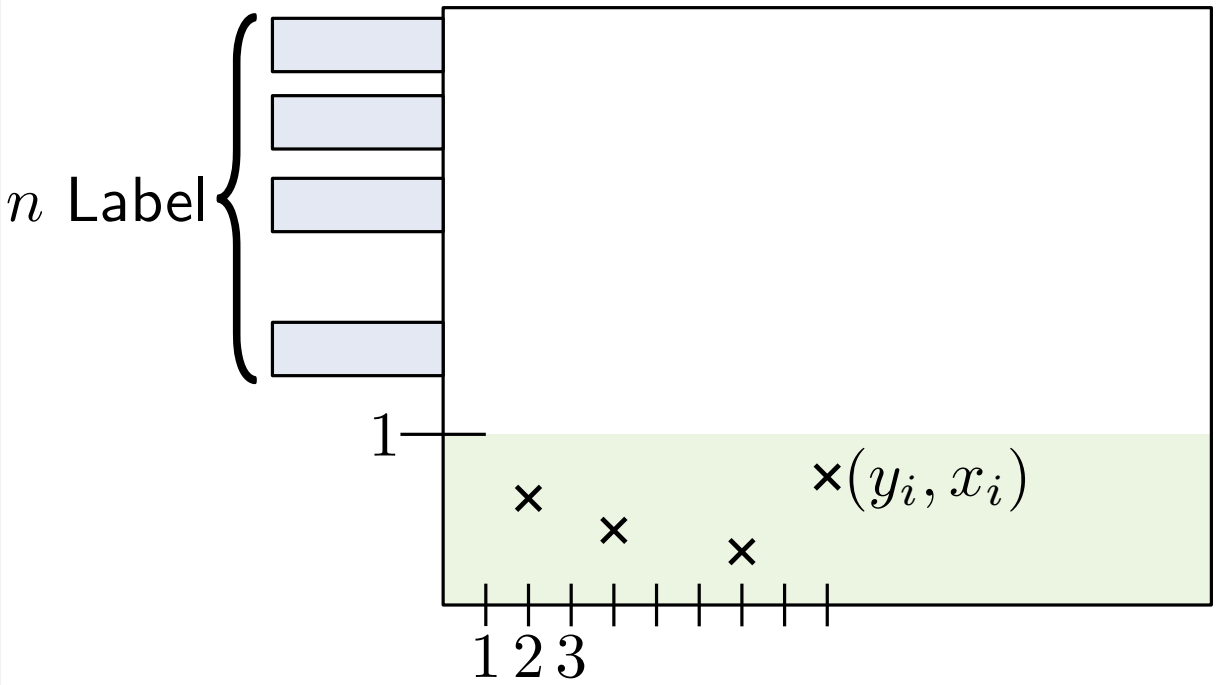
Beweis Lemma 3

Lemma 3: Eine kreuzungsfreie Zuordnung minimaler Länge kann nicht schneller als in $\Omega(n \log n)$ Zeit berechnet werden (s. Übung).

Idee: Reduziere Sortieren von n Zahlen auf Problem.

↳ $\Omega(n \log n)$ Zeit in vergleichsbasierten Rechnermodell.

Gegeben: Zahlen $x_1, \ldots, x_n$



Punkt (x_i, y_i) mit $0 \leq y_i \leq 1$ für jede Zahl x_i.

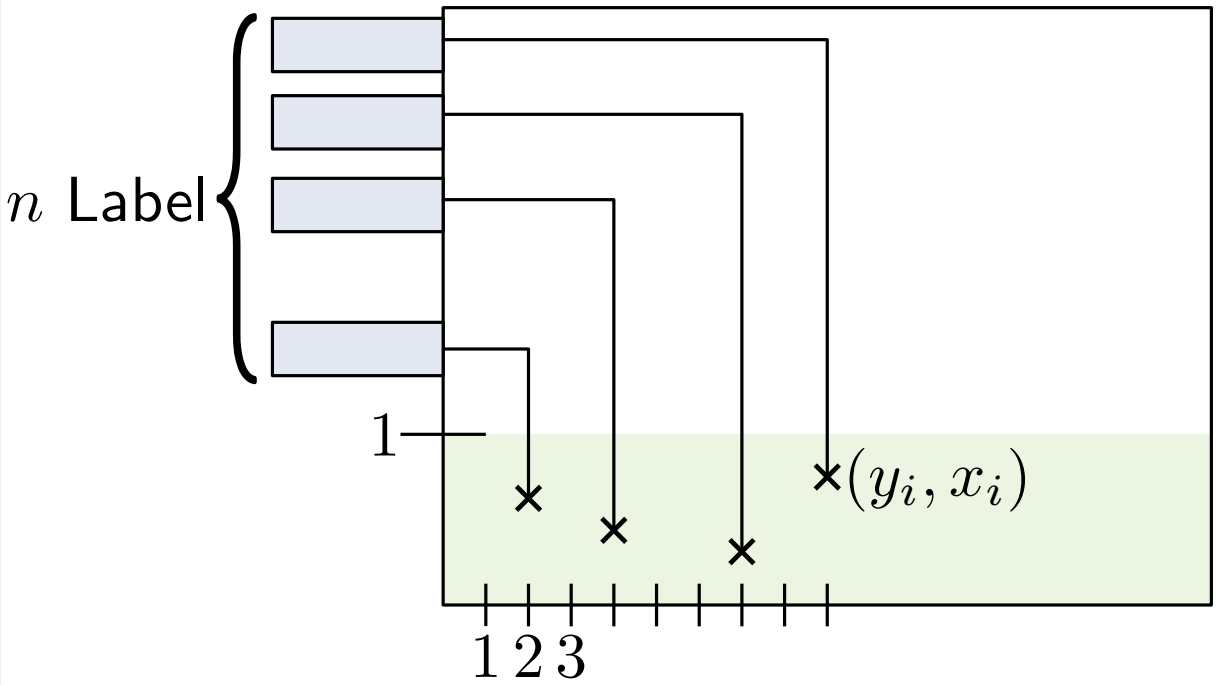
Beweis Lemma 3

Lemma 3: Eine kreuzungsfreie Zuordnung minimaler Länge kann nicht schneller als in $\Omega(n \log n)$ Zeit berechnet werden (s. Übung).

Idee: Reduziere Sortieren von n Zahlen auf Problem.

↳ $\Omega(n \log n)$ Zeit in vergleichsbasierten Rechnermodell.

Gegeben: Zahlen $x_1, \ldots, x_n$



Punkt (x_i, y_i) mit $0 \leq y_i \leq 1$ für jede Zahl x_i.

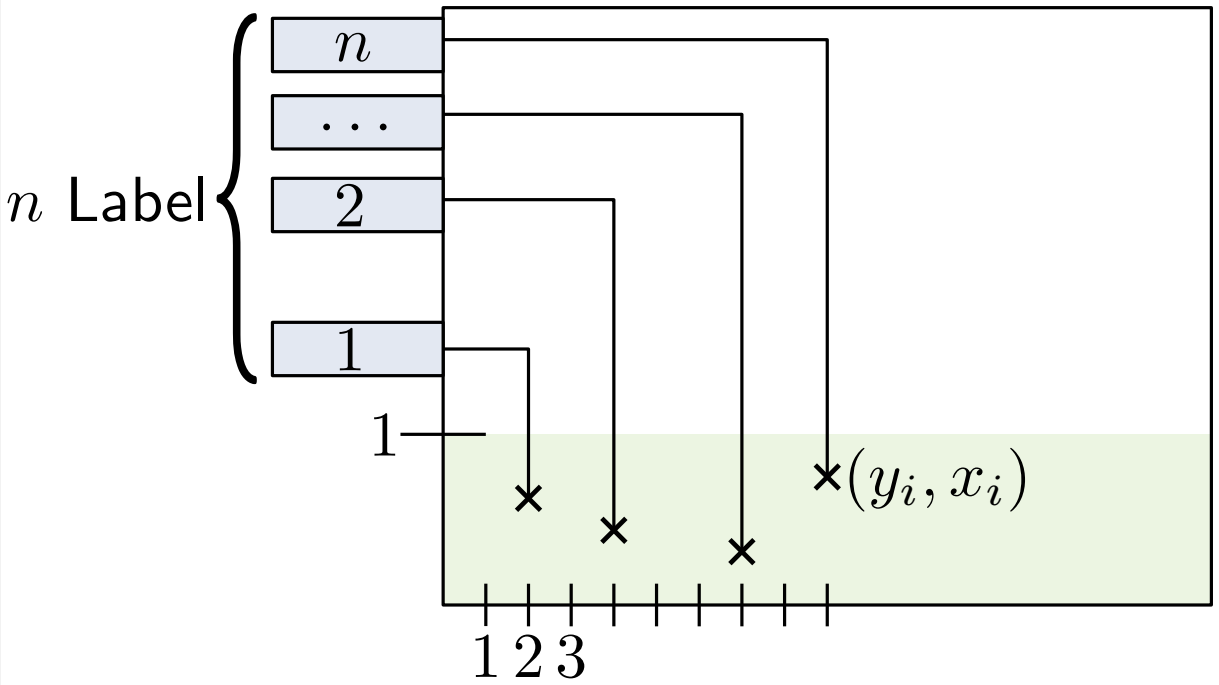
Beweis Lemma 3

Lemma 3: Eine kreuzungsfreie Zuordnung minimaler Länge kann nicht schneller als in $\Omega(n \log n)$ Zeit berechnet werden (s. Übung).

Idee: Reduziere Sortieren von n Zahlen auf Problem.

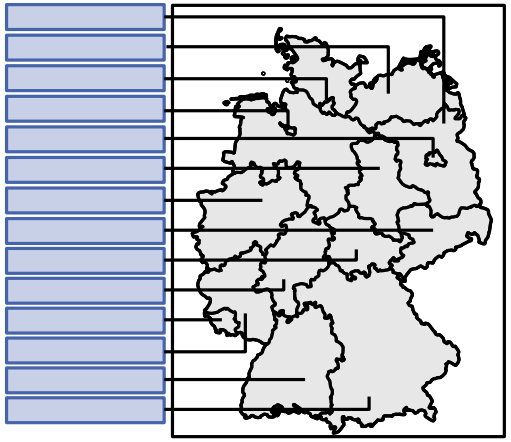
↳ $\Omega(n \log n)$ Zeit in vergleichsbasierten Rechnermodell.

Gegeben: Zahlen $x_1, \ldots, x_n$

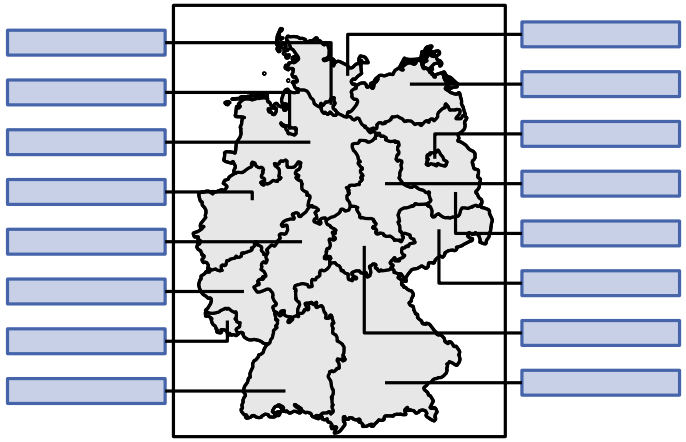


Punkt (x_i, y_i) mit $0 \leq y_i \leq 1$ für jede Zahl x_i.

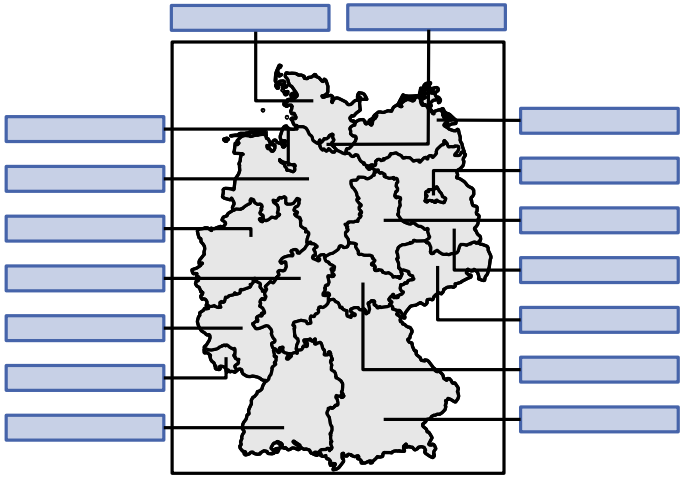
Arten von Boundary-Labeling



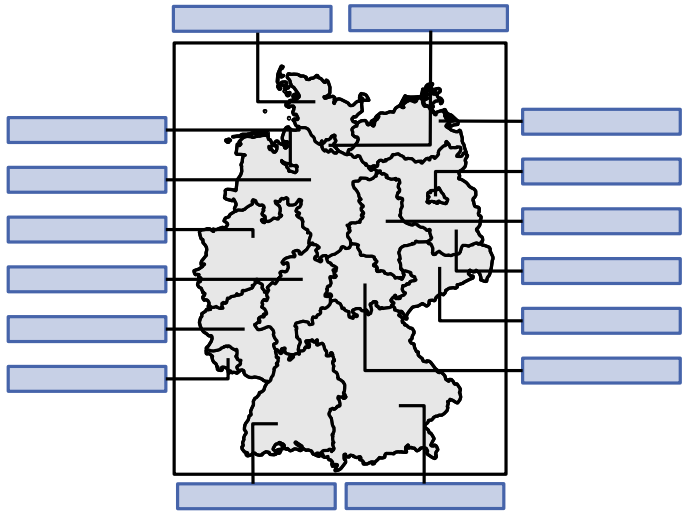
Einseitiger Fall



Zweiseitiger Fall

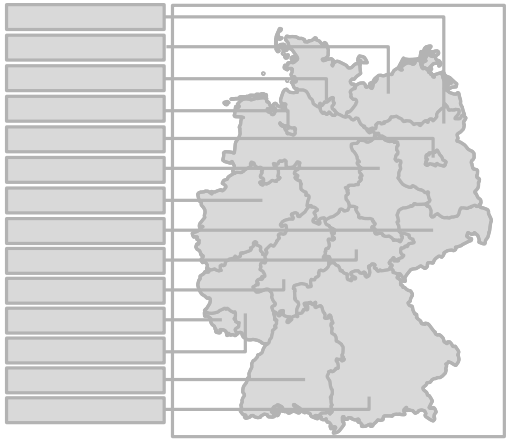


Dreiseitiger Fall

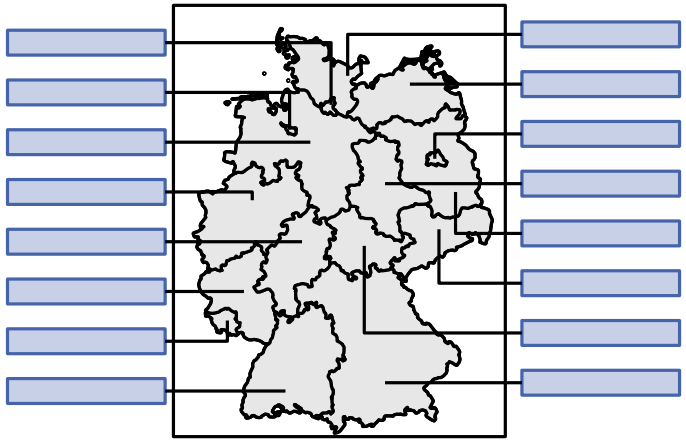


Vierseitiger Fall

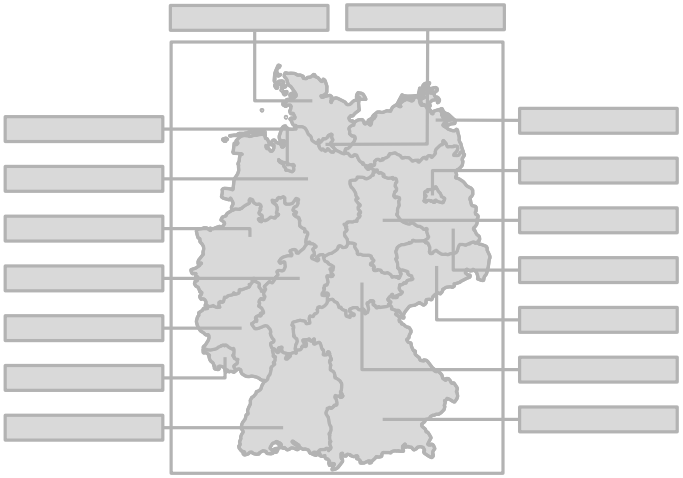
Arten von Boundary-Labeling



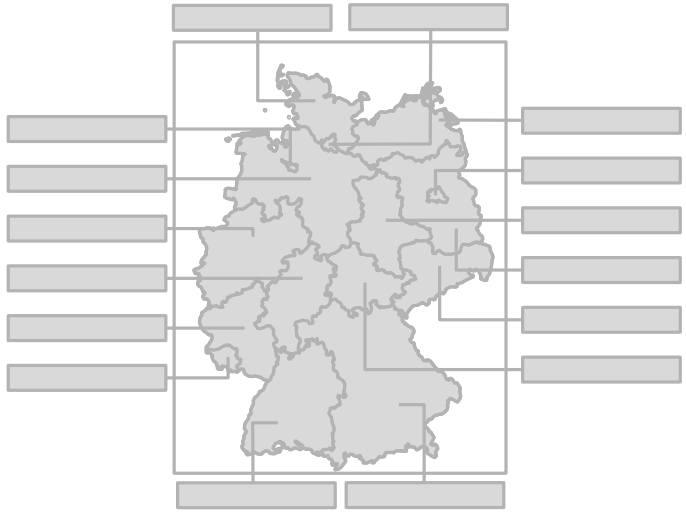
Einseitiger Fall



Zweiseitiger Fall



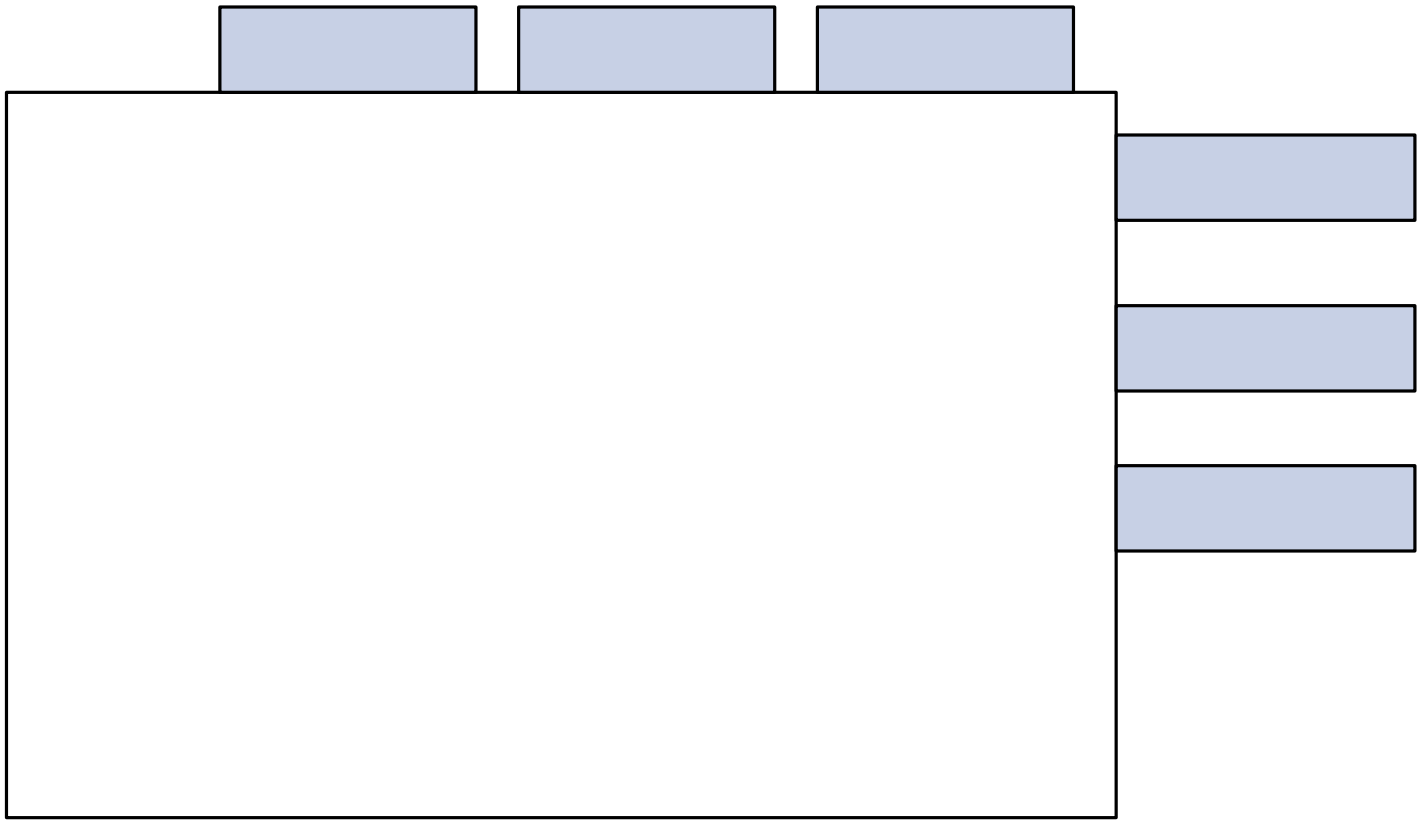
Dreiseitiger Fall



Vierseitiger Fall

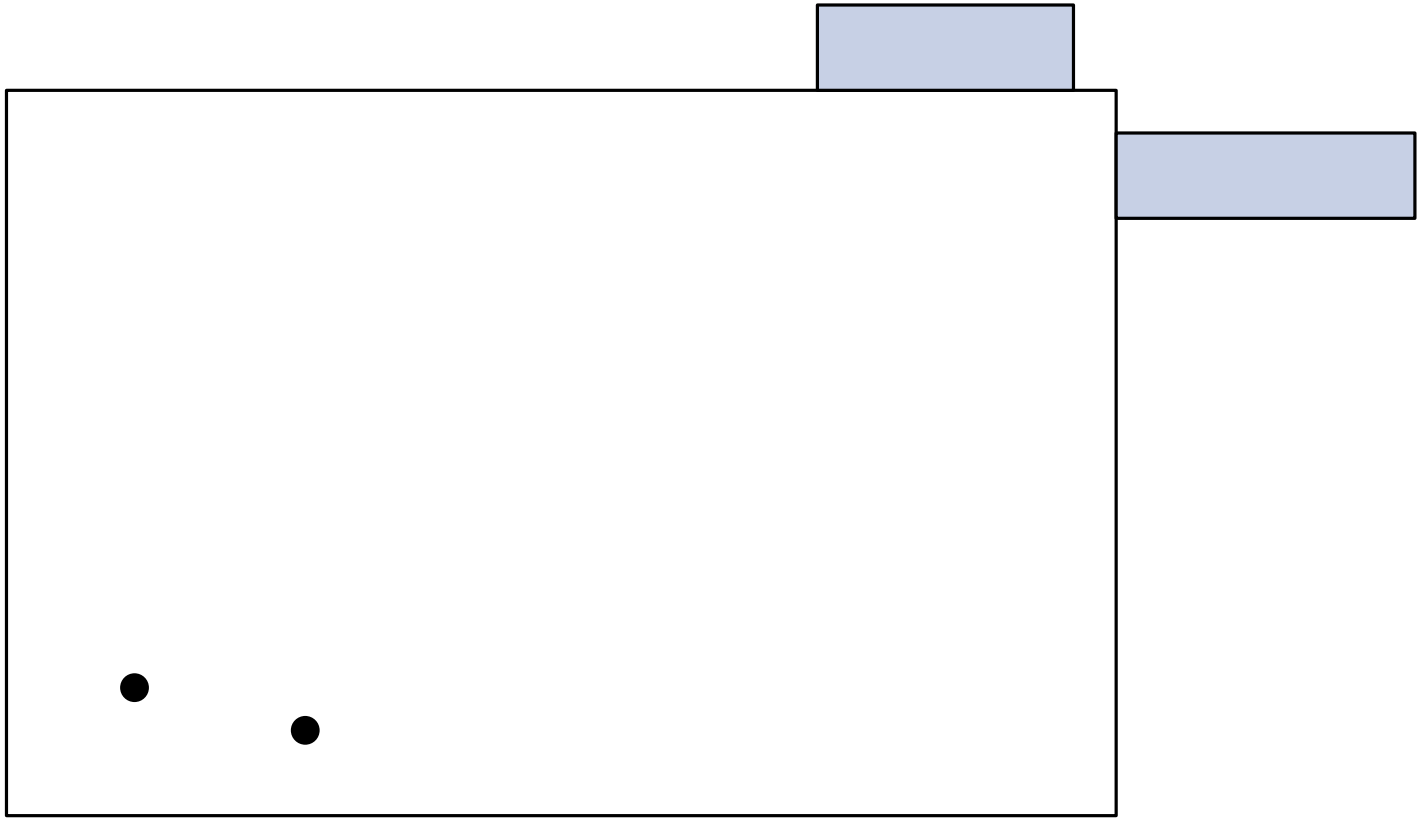
Aufgabe 3.1

Nehmen Sie an, dass die Label an zwei benachbarten Seiten von R angrenzen. Kann für n Label und n Punkte immer eine kreuzungsfreie Zuordnung gefunden werden?



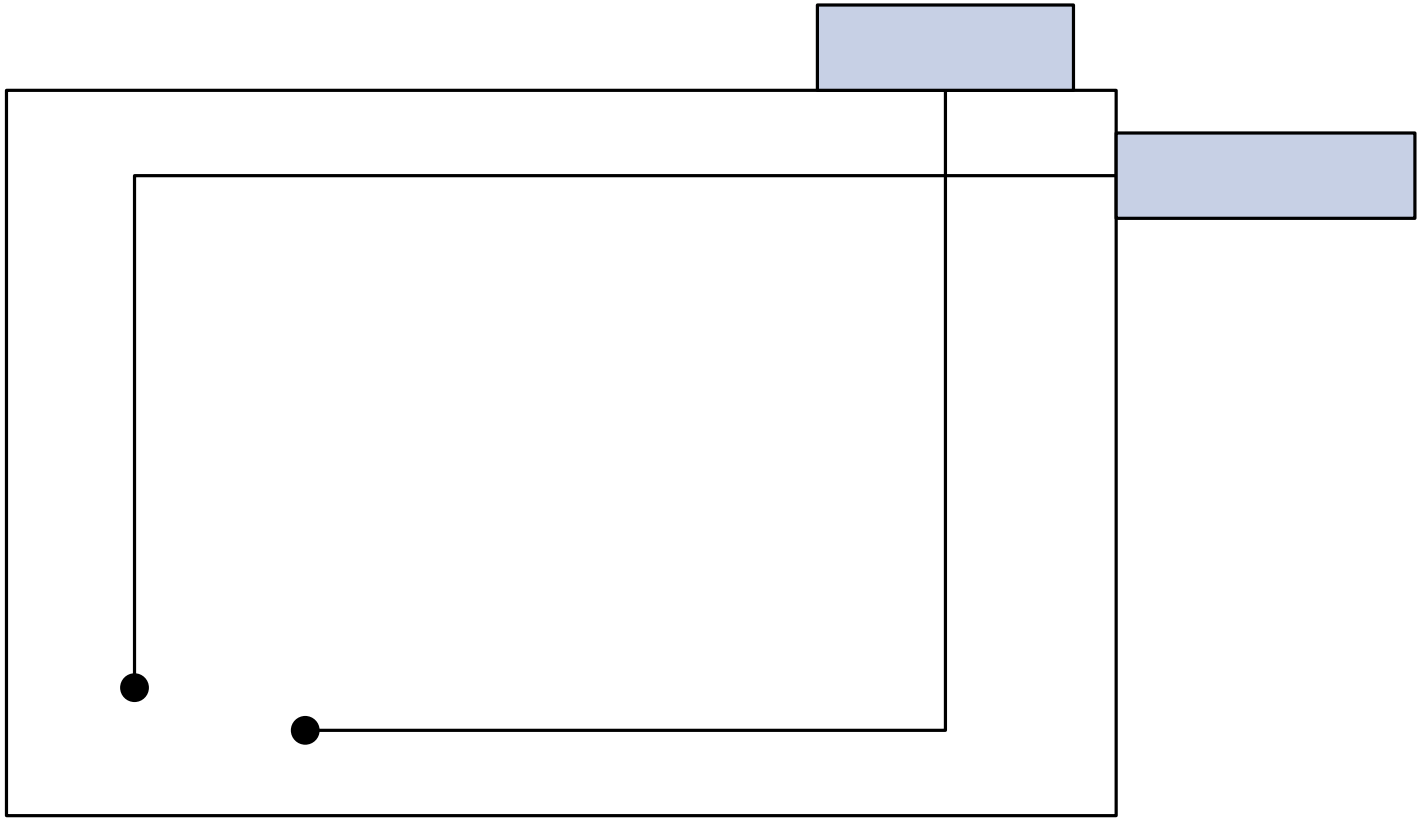
Aufgabe 3.1

Nehmen Sie an, dass die Label an zwei benachbarten Seiten von R angrenzen. Kann für n Label und n Punkte immer eine kreuzungsfreie Zuordnung gefunden werden?



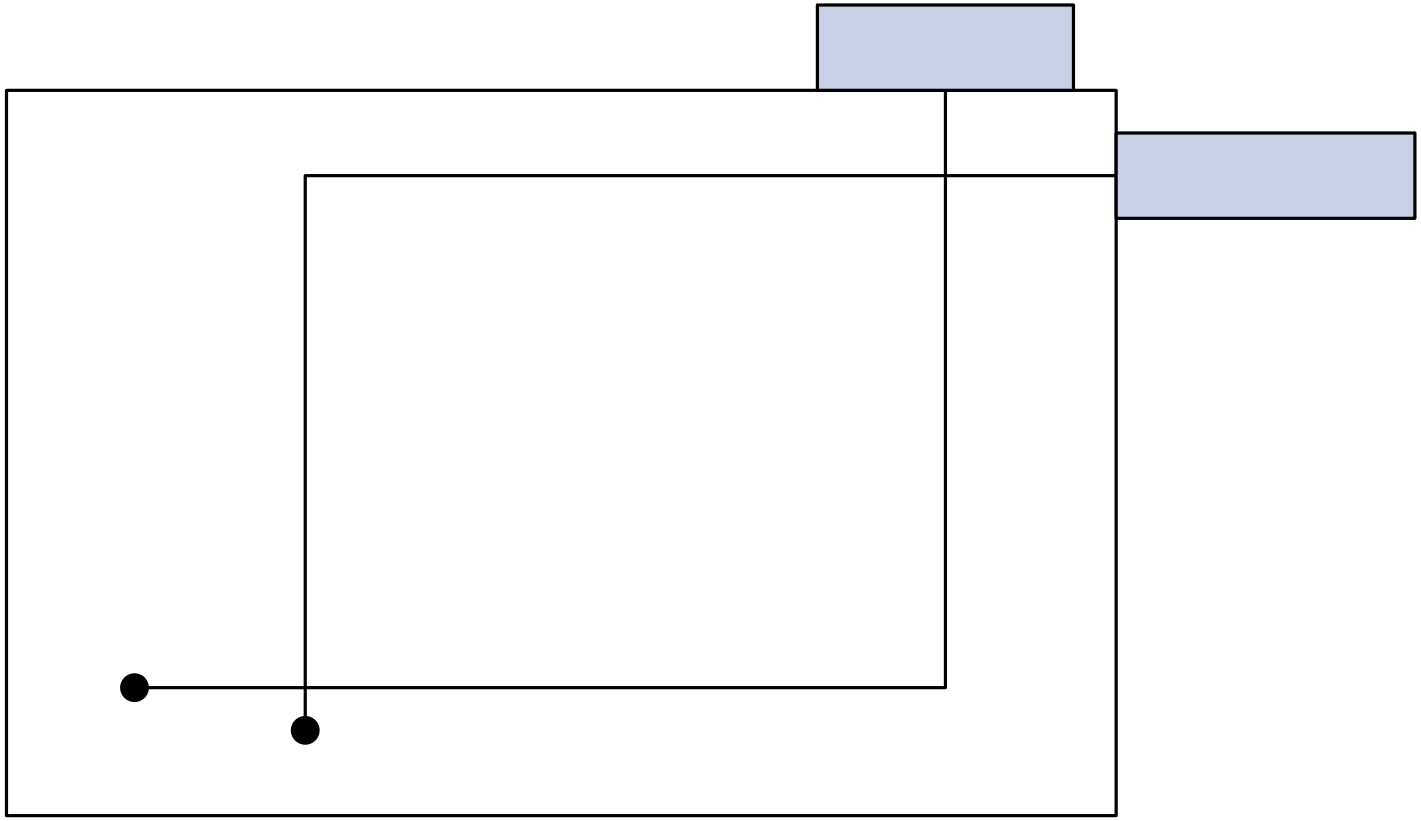
Aufgabe 3.1

Nehmen Sie an, dass die Label an zwei benachbarten Seiten von R angrenzen. Kann für n Label und n Punkte immer eine kreuzungsfreie Zuordnung gefunden werden?



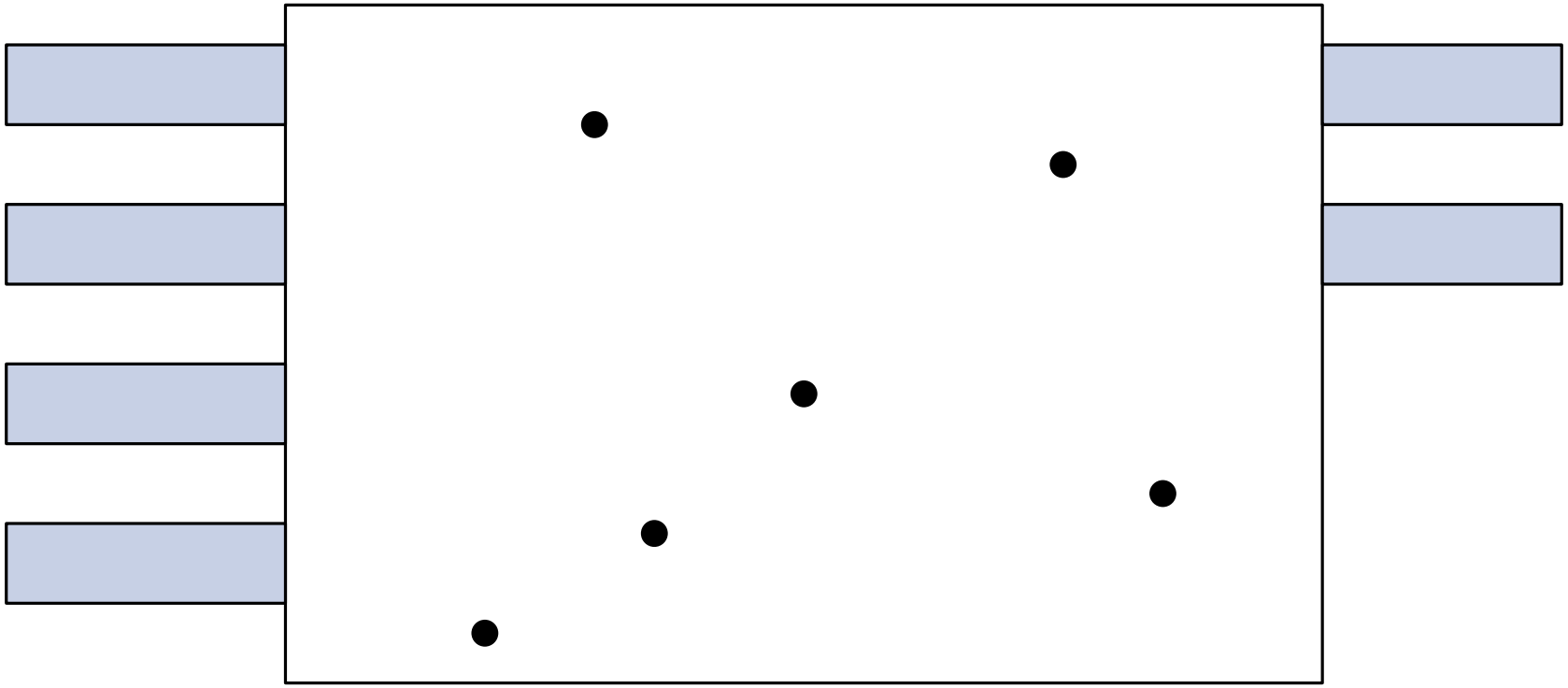
Aufgabe 3.1

Nehmen Sie an, dass die Label an zwei benachbarten Seiten von R angrenzen. Kann für n Label und n Punkte immer eine kreuzungsfreie Zuordnung gefunden werden?



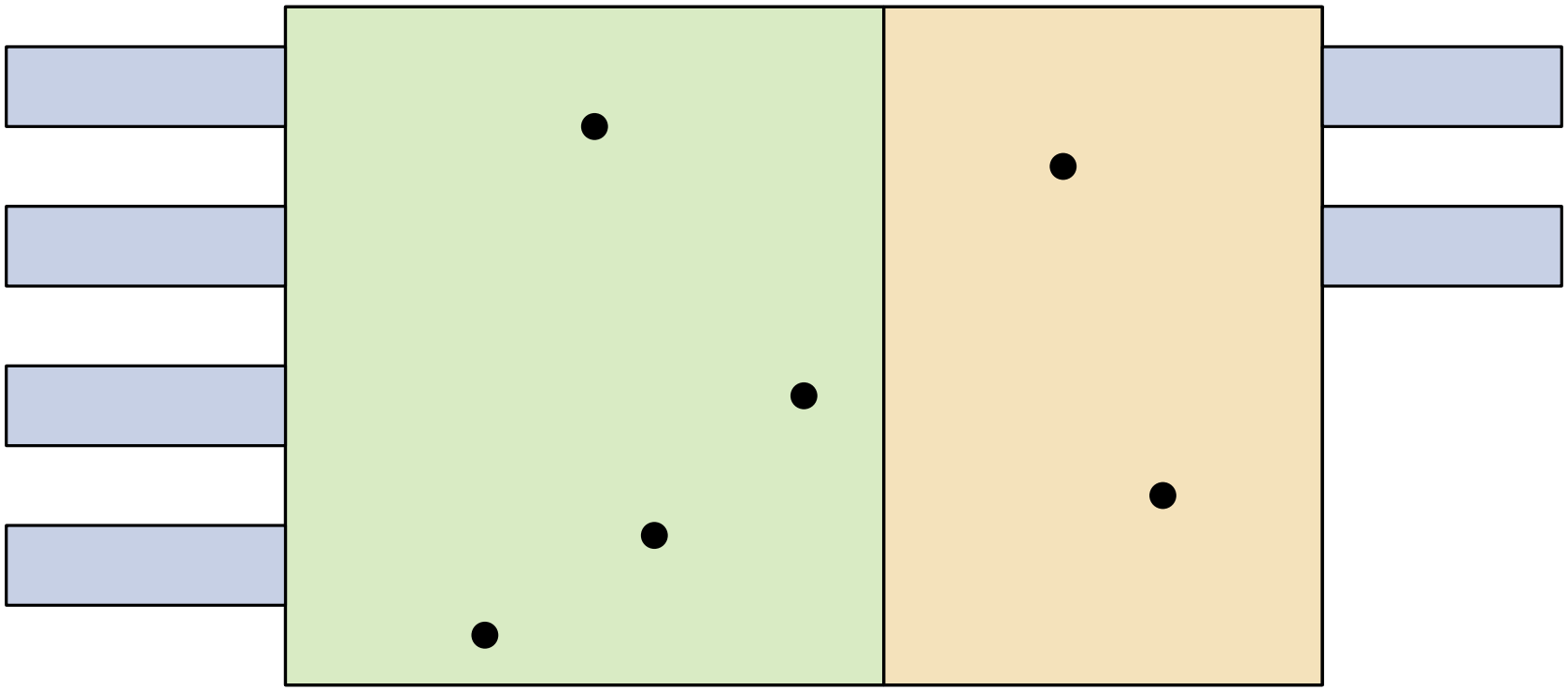
Aufgabe 3.2

Nehmen Sie an, dass die Label an zwei gegenüberliegenden Seiten von R angrenzen. Kann für n Label und n Punkte immer eine kreuzungsfreie Zuordnung gefunden werden?



Aufgabe 3.2

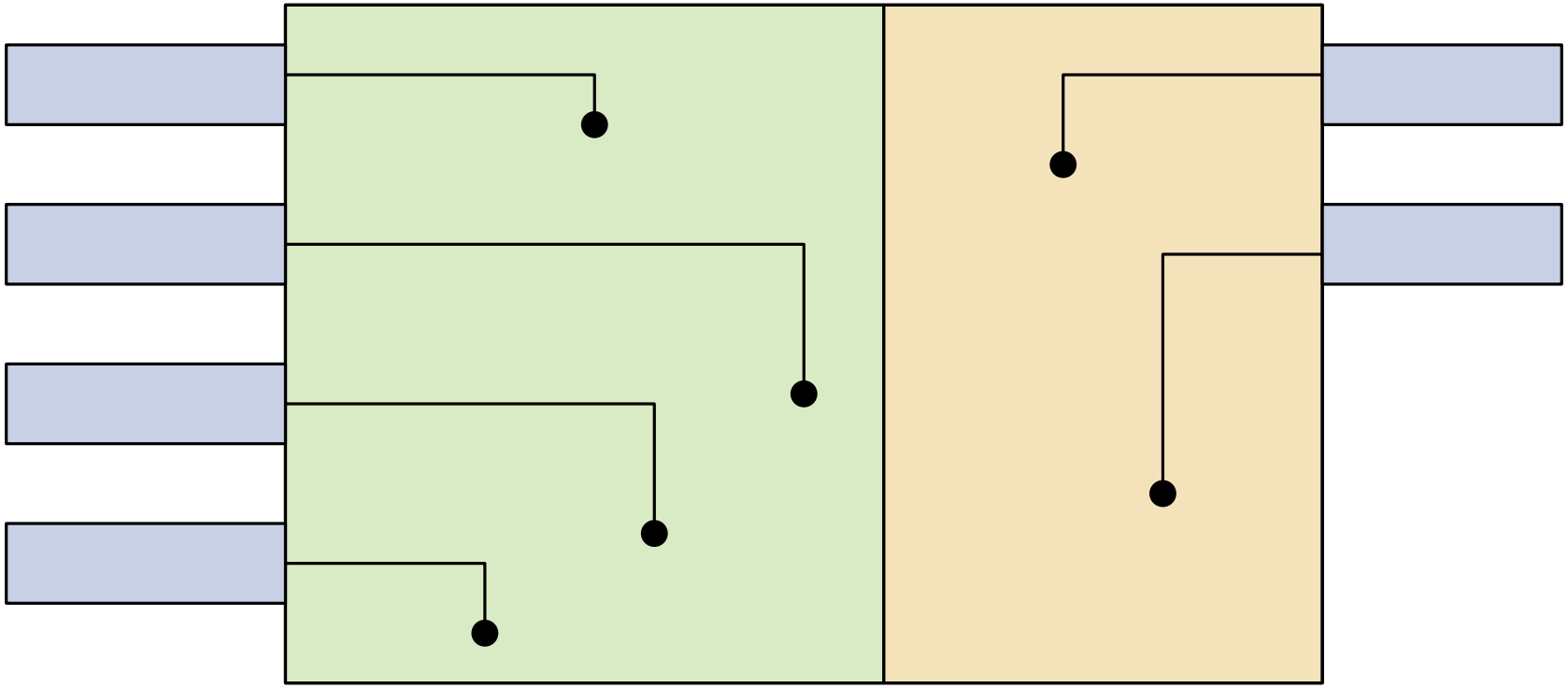
Nehmen Sie an, dass die Label an zwei gegenüberliegenden Seiten von R angrenzen. Kann für n Label und n Punkte immer eine kreuzungsfreie Zuordnung gefunden werden?



1. Teile zweiseitiges Problem in zwei unabhängige einseitige Probleme auf, so dass die Label- und Punktanzahlen balanciert sind.

Aufgabe 3.2

Nehmen Sie an, dass die Label an zwei gegenüberliegenden Seiten von R angrenzen. Kann für n Label und n Punkte immer eine kreuzungsfreie Zuordnung gefunden werden?



1. Teile zweiseitiges Problem in zwei unabhängige einseitige Probleme auf, so dass die Label- und Punktanzahlen balanciert sind.

2. Löse jedes Problem für sich (immer möglich).

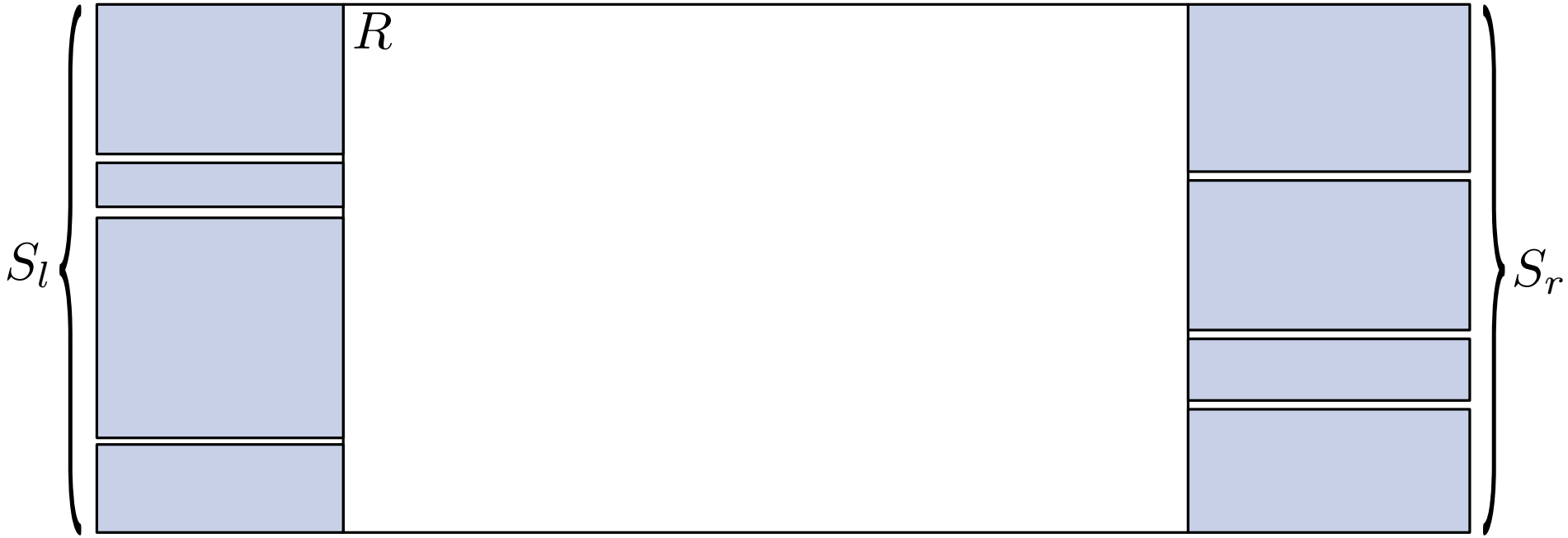
Übungsblatt 5

Warum immer uniforme Labels?

LabelZuweisung:

Gegeben: Rechteck R und Menge L an Labels unterschiedlicher Höhe. Gesamthöhe der Labels entspricht zweimal der Höhe von R.

Gesucht: Platzierung der Label entlang der linken und rechten Seite von R, sodass kein Label die obere noch die untere Seite über- bzw. unterschreitet.



Warum immer uniforme Labels?

LabelZuweisung:

Gegeben: Rechteck R und Menge L an Labels unterschiedlicher Höhe. Gesamthöhe der Labels entspricht zweimal der Höhe von R.

Gesucht: Platzierung der Label entlang der linken und rechten Seite von R, sodass kein Label die obere noch die untere Seite über- bzw. unterschreitet.

Problem ist $\mathcal{NP}$-vollständig.

1. LabelZuweisung $\in \mathcal{NP}$:
   a.) Rate Mengen $S_l, S_r \subseteq L$.
   b.) Überprüfe ob Summe der Höhen der Label in S_l bzw. S_r Höhe von R einhält.

Warum immer uniforme Labels?

1. LabelZuweisung ist $\mathcal{NP}$-schwer:

Warum immer uniforme Labels?

1. LabelZuweisung ist $\mathcal{NP}$-schwer:

Partition:

Gegeben: Menge S an Zahlen.

Gesucht: Mengen $S_1 \subseteq S$ und $S_2 \subseteq S$, sodass

1. $S_1 \cap S_2 = \emptyset$ und $S_1 \cup S_2 = S$
2. $\sum_{x \in S_1} x = \sum_{x \in S_2} x$

Partition ist $\mathcal{NP}$-vollständig.

Warum immer uniforme Labels?

1. LabelZuweisung ist $\mathcal{NP}$-schwer:

Partition:

Gegeben: Menge S an Zahlen.

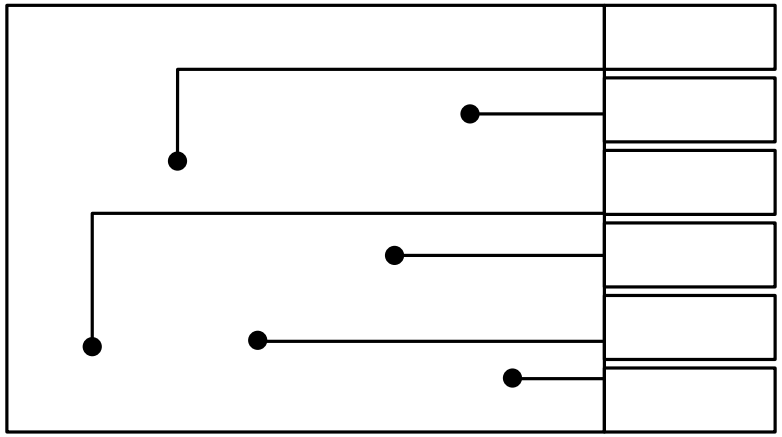
Gesucht: Mengen $S_1 \subseteq S$ und $S_2 \subseteq S$, sodass

1. $S_1 \cap S_2 = \emptyset$ und $S_1 \cup S_2 = S$
2. $\sum_{x \in S_1} x = \sum_{x \in S_2} x$

Partition ist $\mathcal{N}P$-vollständig.

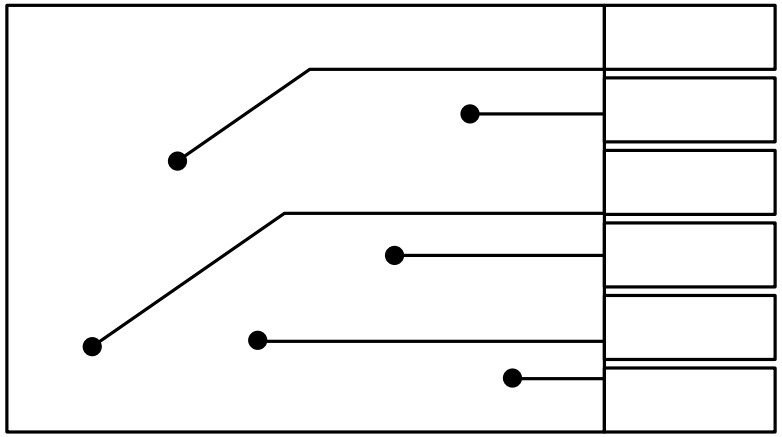
Reduktion: Übersetze $x \in S$ in Label l_x mit Höhe x.

Allgemeine Bewertungsfunktion



- einseitig
- uniforme Label
- feste Labelpositionen
- kreuzungsfrei
- po-Leader
- sliding Ports
- **allgemeine Bewertungsfunktion**

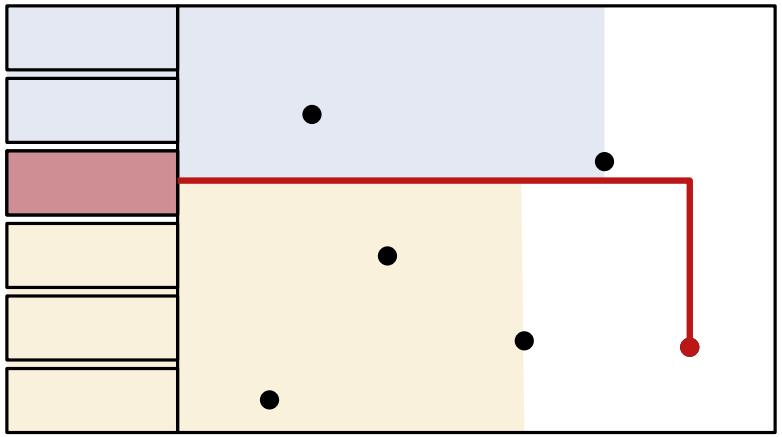
Allgemeine Bewertungsfunktion



- einseitig
- uniforme Label
- feste Labelpositionen
- kreuzungsfrei
- ~~po-Leader~~ do-Leader
- sliding Ports
- **allgemeine Bewertungsfunktion**

Dynamisches Programm

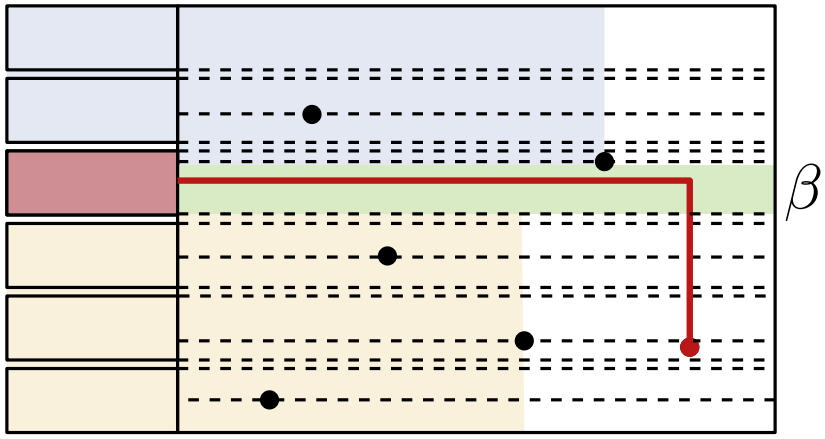
Erinnerung:



Leader des rechtesten Punktes zerlegt Instanz in zwei unabhängige Teile.

Dynamisches Programm

Erinnerung:

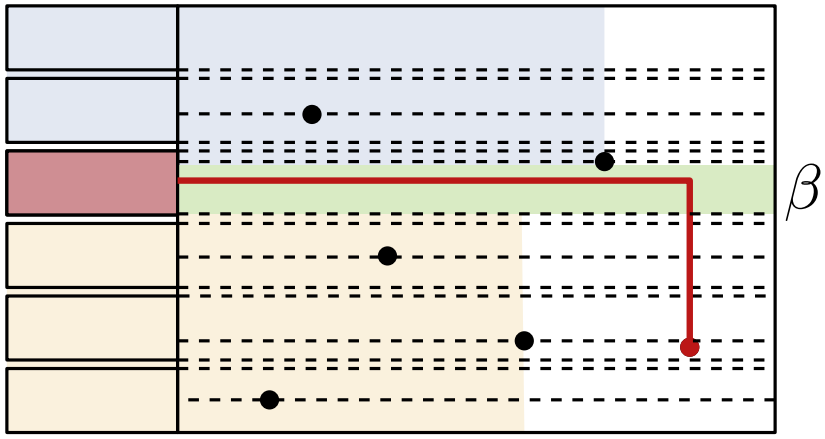


Leader des rechtesten Punktes zerlegt Instanz in zwei unabhängige Teile.

Streifen β, in dem der Arm des Leaders liegt, definiert Trennung.

Dynamisches Programm

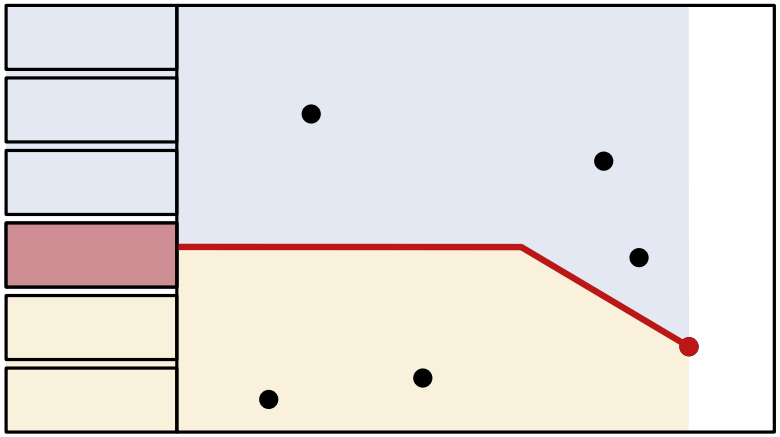
Erinnerung:



Leader des rechtesten Punktes zerlegt Instanz in zwei unabhängige Teile.

Streifen β, in dem der Arm des Leaders liegt, definiert Trennung.

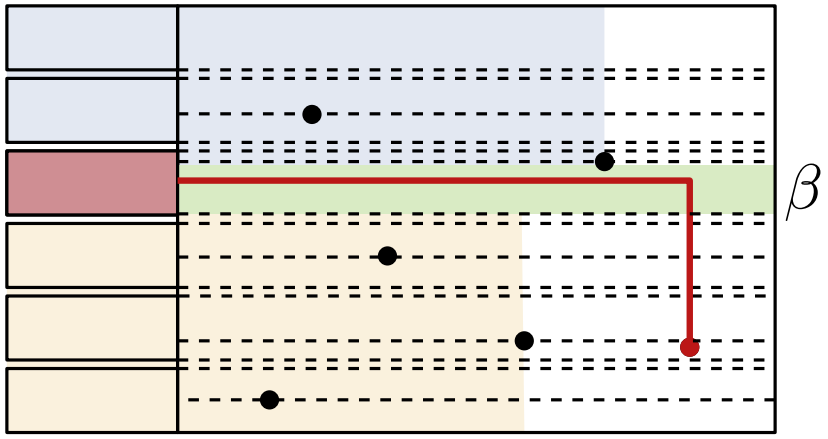
do-Leader:



Leader des rechtesten Punktes zerlegt Instanz in zwei unabhängige Teile.

Dynamisches Programm

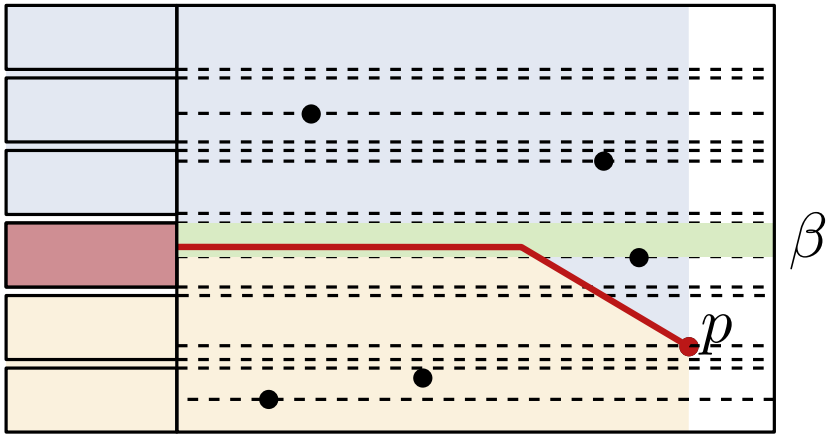
Erinnerung:



Leader des rechtesten Punktes zerlegt Instanz in zwei unabhängige Teile.

Streifen β, in dem der Arm des Leaders liegt, definiert Trennung.

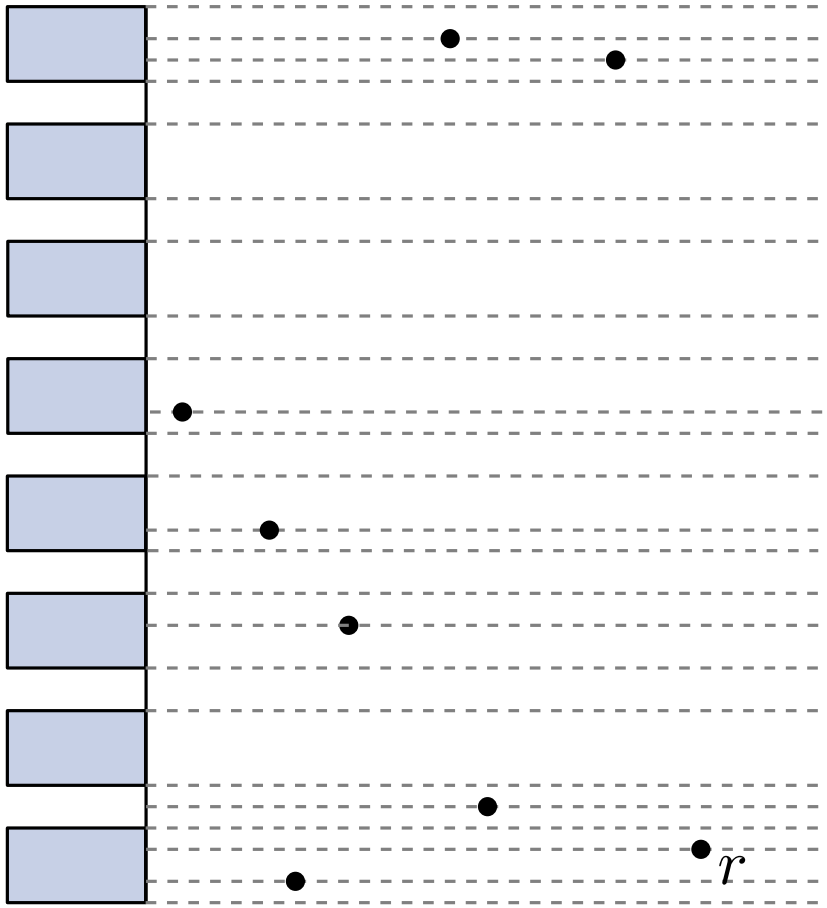
do-Leader:



Leader des rechtesten Punktes zerlegt Instanz in zwei unabhängige Teile.

Streifen β, in dem der Arm des Leaders liegt, und angebundener Punkt p definieren Trennung.

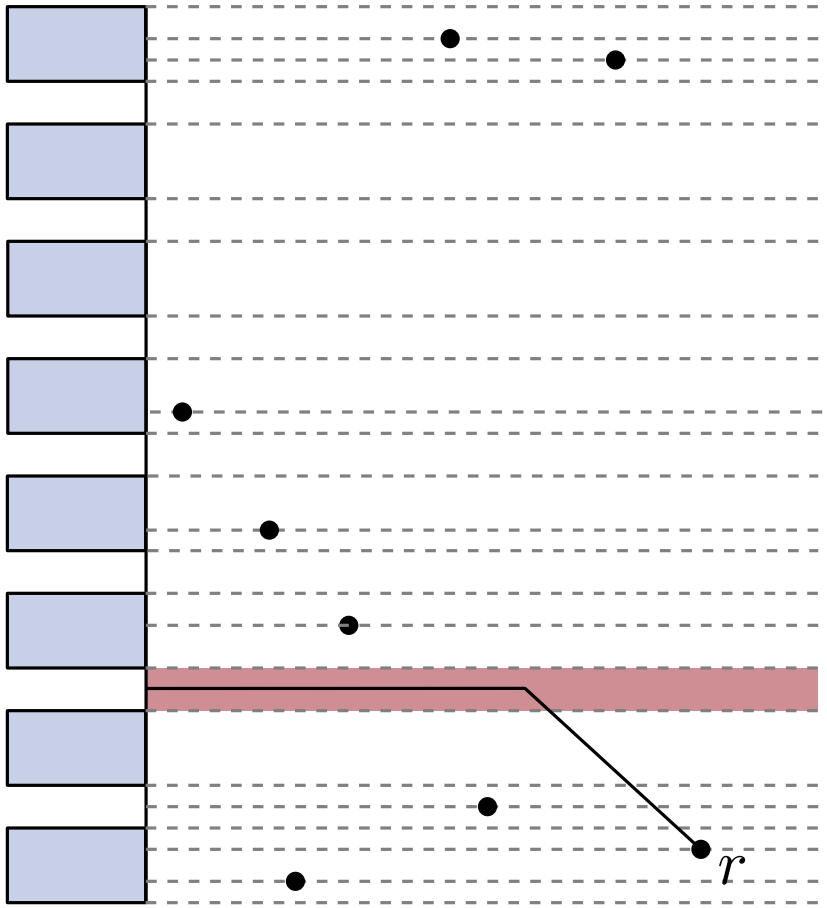
Dynamisches Programm



Streifen ist bzgl. Punkt r *nicht zulässig*, wenn:

1. der Streifen kein Label schneidet, oder
2. #Punkte $\neq$ #Label für Teilinstanzen, oder
3. die Hand des Leaders zu weit links endet.

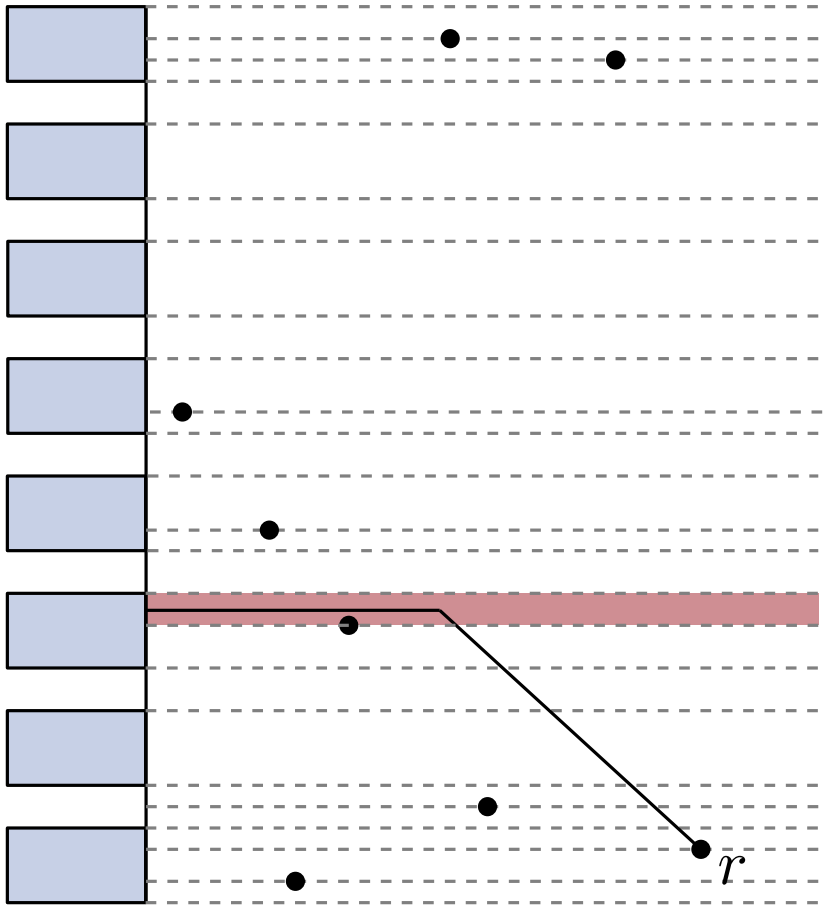
Dynamisches Programm



Streifen ist bzgl. Punkt r *nicht zulässig*, wenn:

1. der Streifen kein Label schneidet, oder
2. #Punkte $\neq$ #Label für Teilinstanzen, oder
3. die Hand des Leaders zu weit links endet.

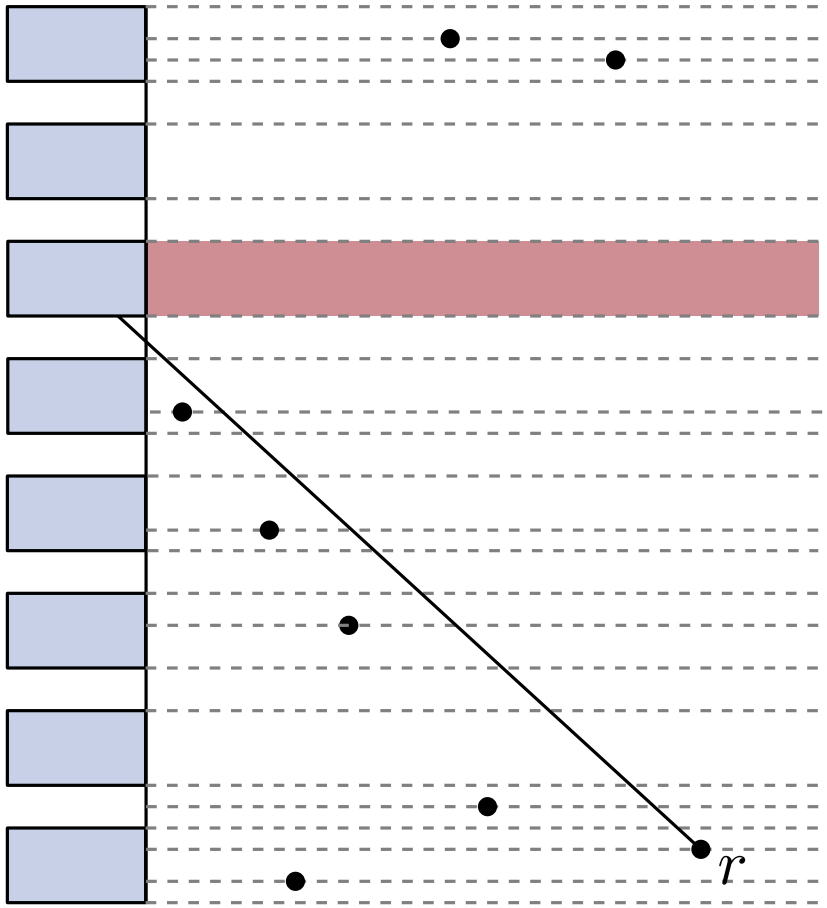
Dynamisches Programm



Streifen ist bzgl. Punkt r *nicht zulässig*, wenn:

1. der Streifen kein Label schneidet, oder
2. #Punkte $\neq$ #Label für Teilinstanzen, oder
3. die Hand des Leaders zu weit links endet.

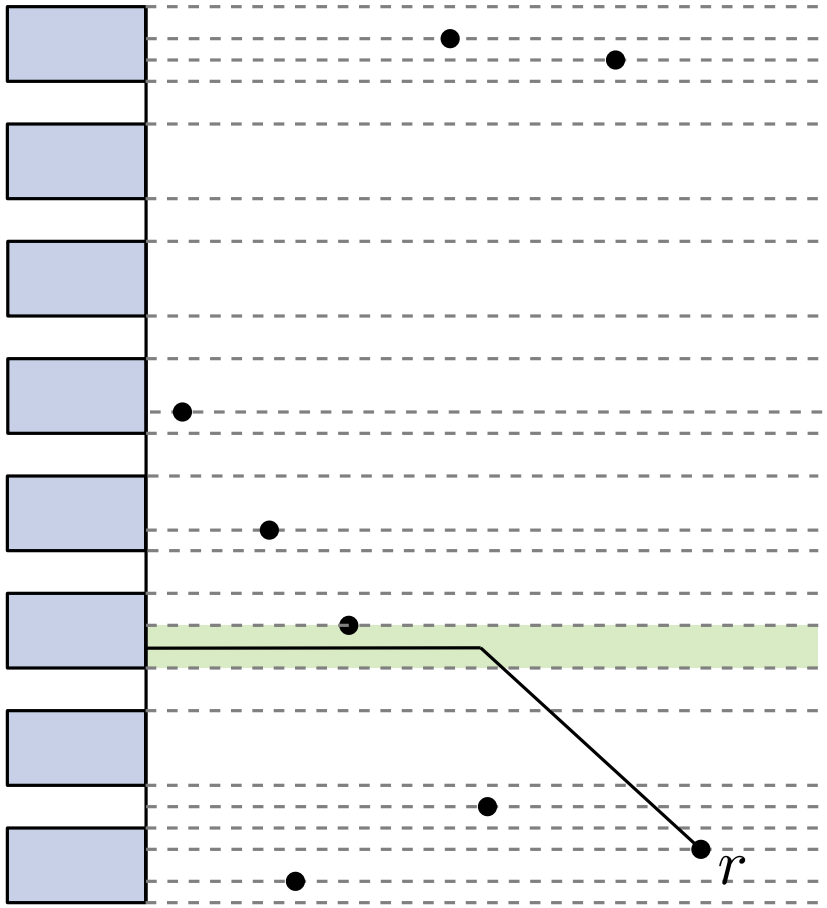
Dynamisches Programm



Streifen ist bzgl. Punkt r *nicht zulässig*, wenn:

1. der Streifen kein Label schneidet, oder
2. #Punkte $\neq$ #Label für Teilinstanzen, oder
3. die Hand des Leaders zu weit links endet.

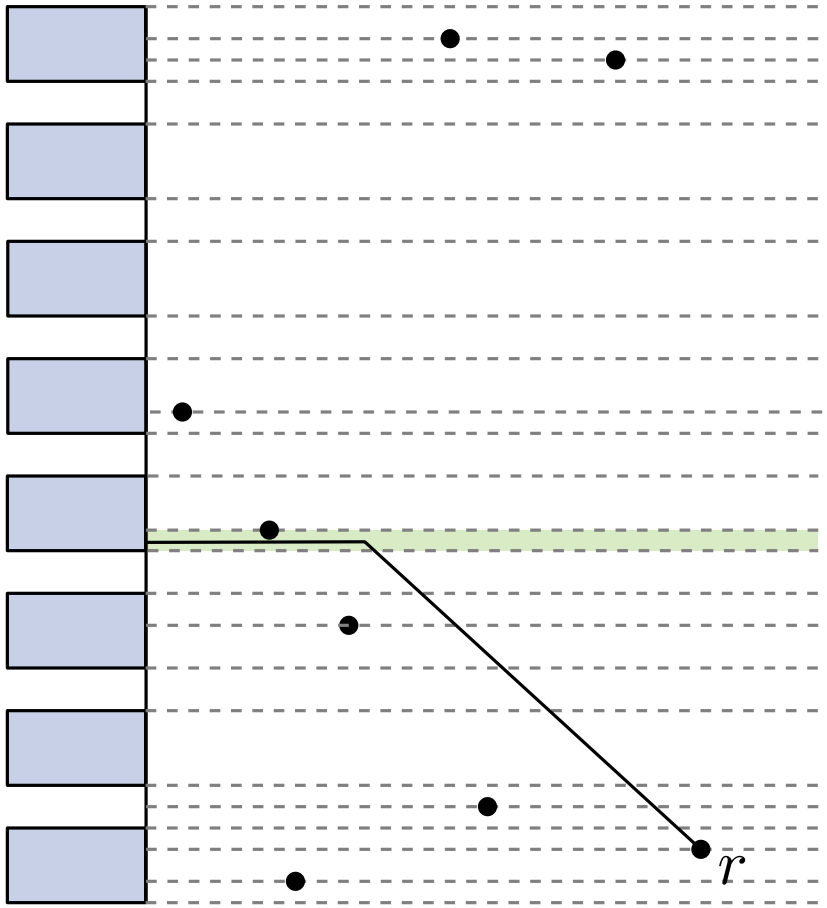
Dynamisches Programm



Streifen ist bzgl. Punkt r *nicht zulässig*, wenn:

1. der Streifen kein Label schneidet, oder
2. #Punkte $\neq$ #Label für Teilinstanzen, oder
3. die Hand des Leaders zu weit links endet.

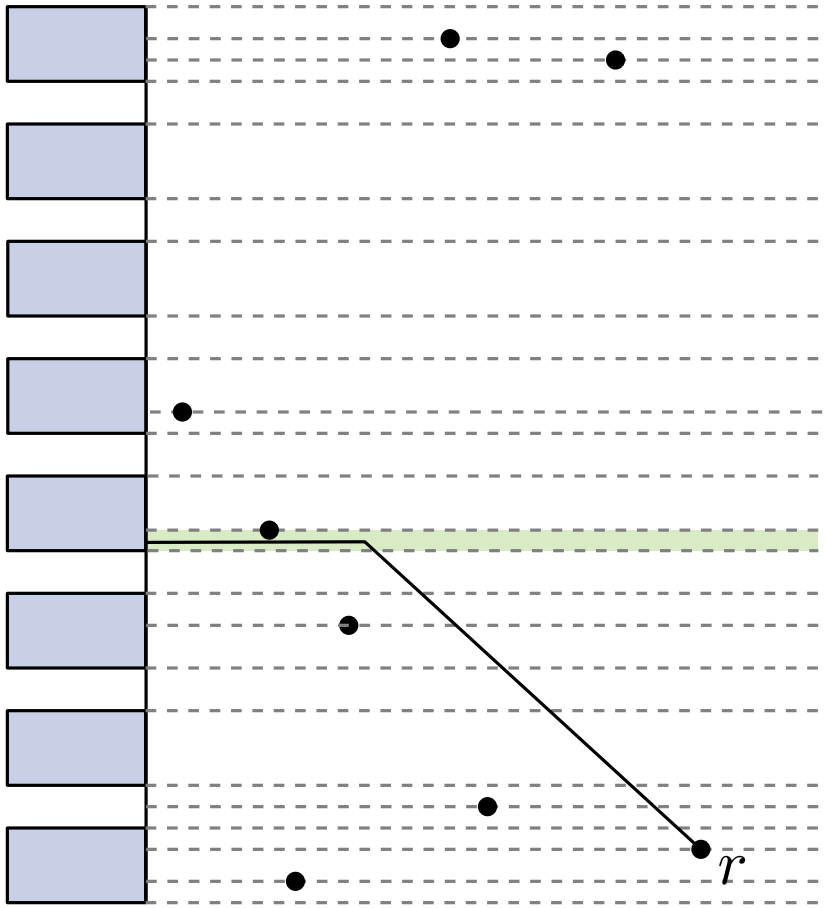
Dynamisches Programm



Streifen ist bzgl. Punkt r *nicht zulässig*, wenn:

1. der Streifen kein Label schneidet, oder
2. #Punkte $\neq$ #Label für Teilinstanzen, oder
3. die Hand des Leaders zu weit links endet.

Dynamisches Programm



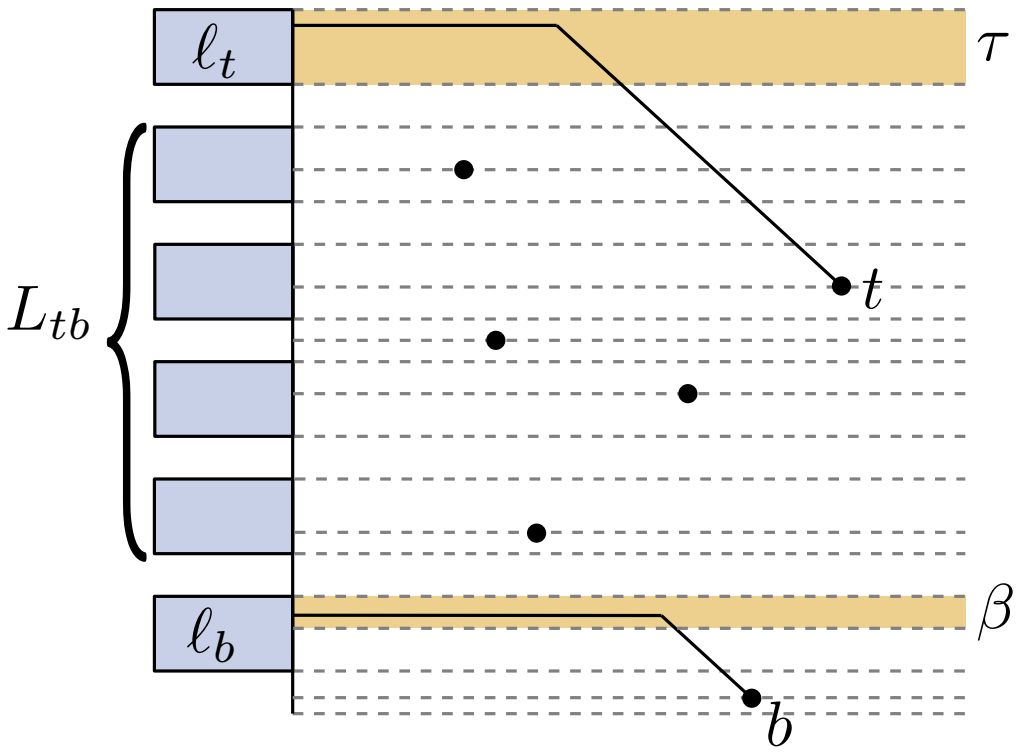
Idee:
Optimiere über alle zulässigen Streifen

Streifen ist bzgl. Punkt r *nicht zulässig*, wenn:

1. der Streifen kein Label schneidet, oder
2. #Punkte $\neq$ #Label für Teilinstanzen, oder
3. die Hand des Leaders zu weit links endet.

Dynamisches Programm

Betrachte folgende Instanz:

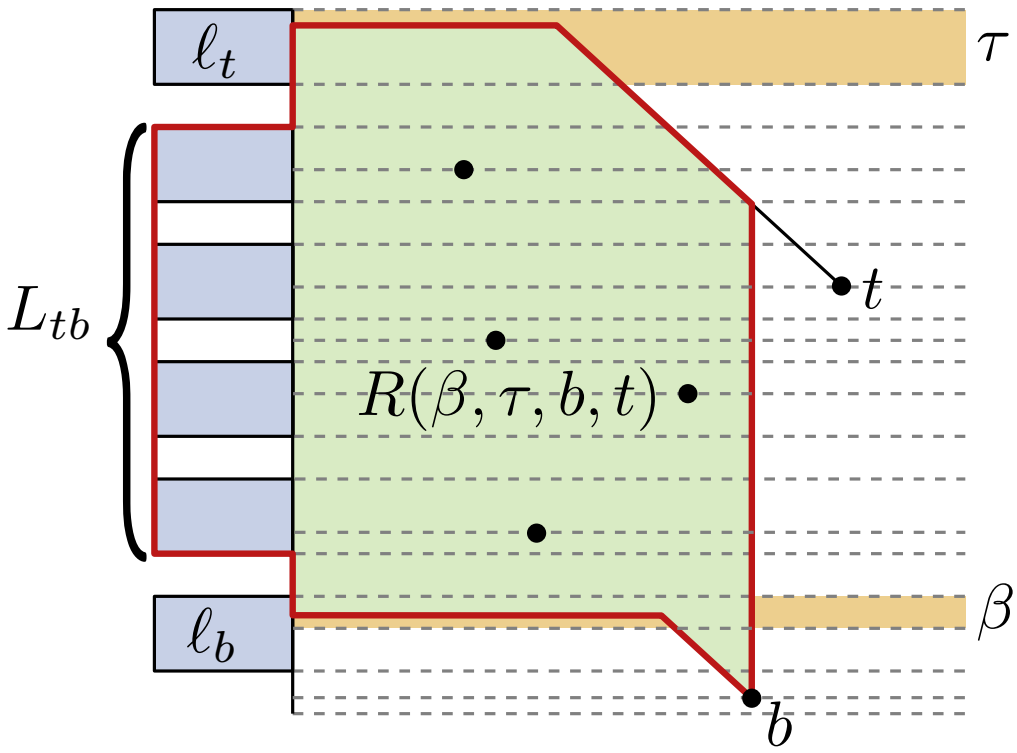


Annahme:

- Punkte rechts von b und t sind bereits angebunden.
- Labels in L_{tb} sind noch nicht angebunden.

Dynamisches Programm

Betrachte folgende Instanz:



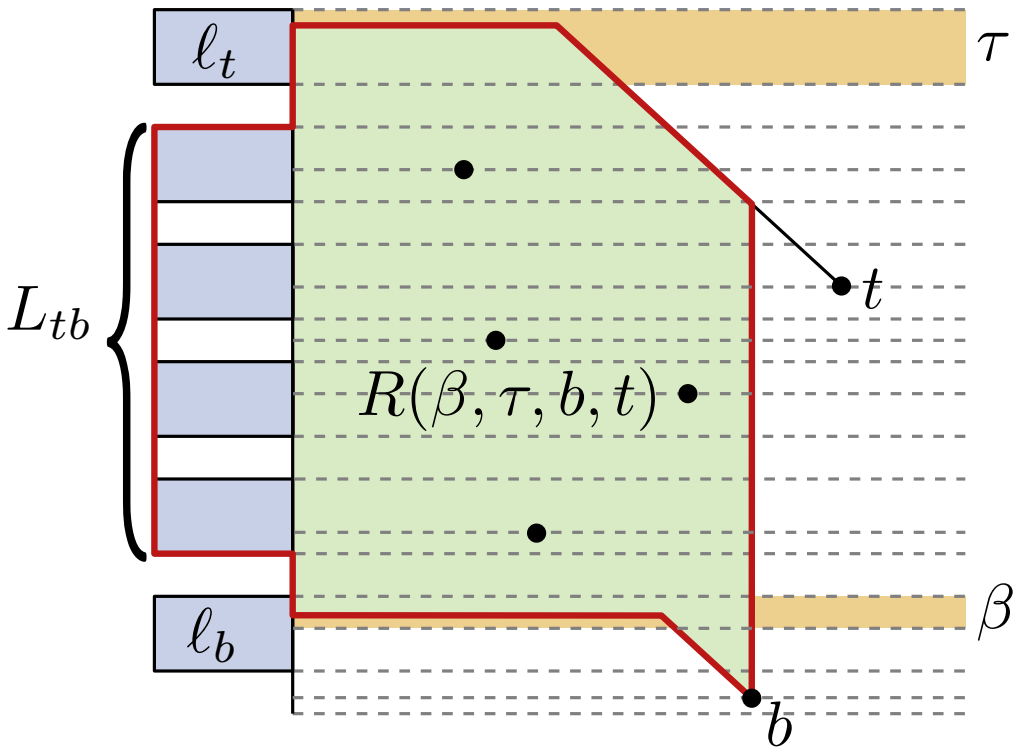
Annahme:

- Punkte rechts von b und t sind bereits angebunden.
- Labels in L_{tb} sind noch nicht angebunden.

→ Label in L_{tb} können nur mit Punkten in $R(\beta, \tau, b, t)$ verbunden werden.

Dynamisches Programm

Betrachte folgende Instanz:



Annahme:

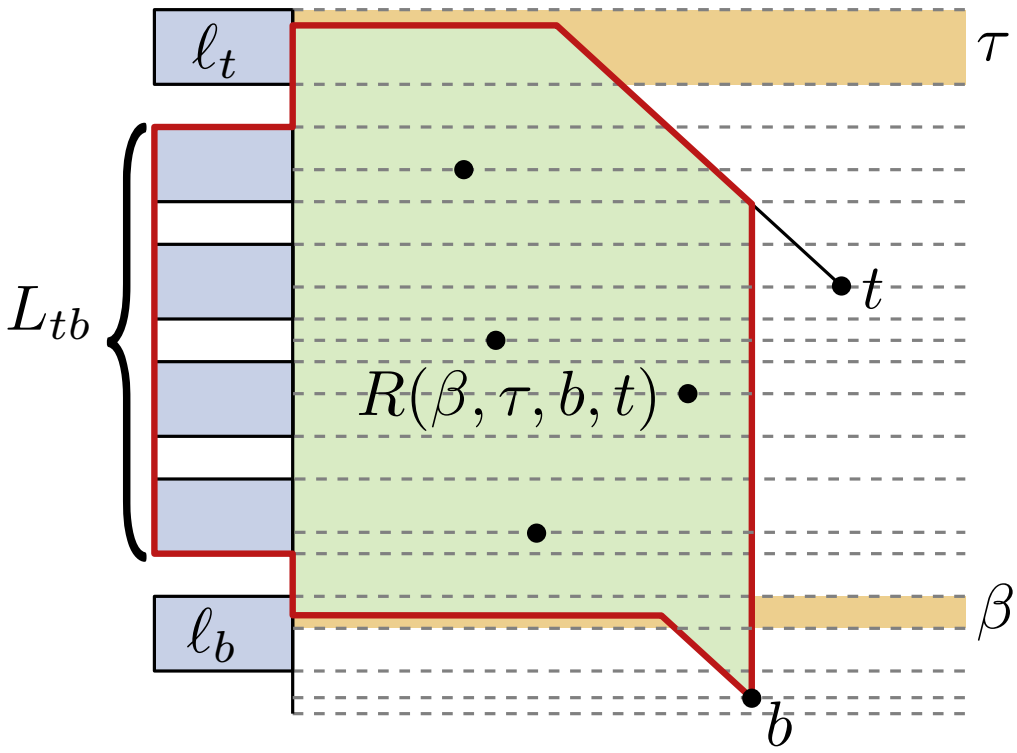
- Punkte rechts von b und t sind bereits angebunden.
- Labels in L_{tb} sind noch nicht angebunden.

→ Label in L_{tb} können nur mit Punkten in $R(\beta, \tau, b, t)$ verbunden werden.

→ $R(\beta, \tau, b, t)$ induziert unabhängige Instanz I.

Dynamisches Programm

Betrachte folgende Instanz:



Schreibe $I = (\beta, \tau, b, t)$ für Instanz I induziert durch $R(\beta, \tau, b, t)$

Annahme:

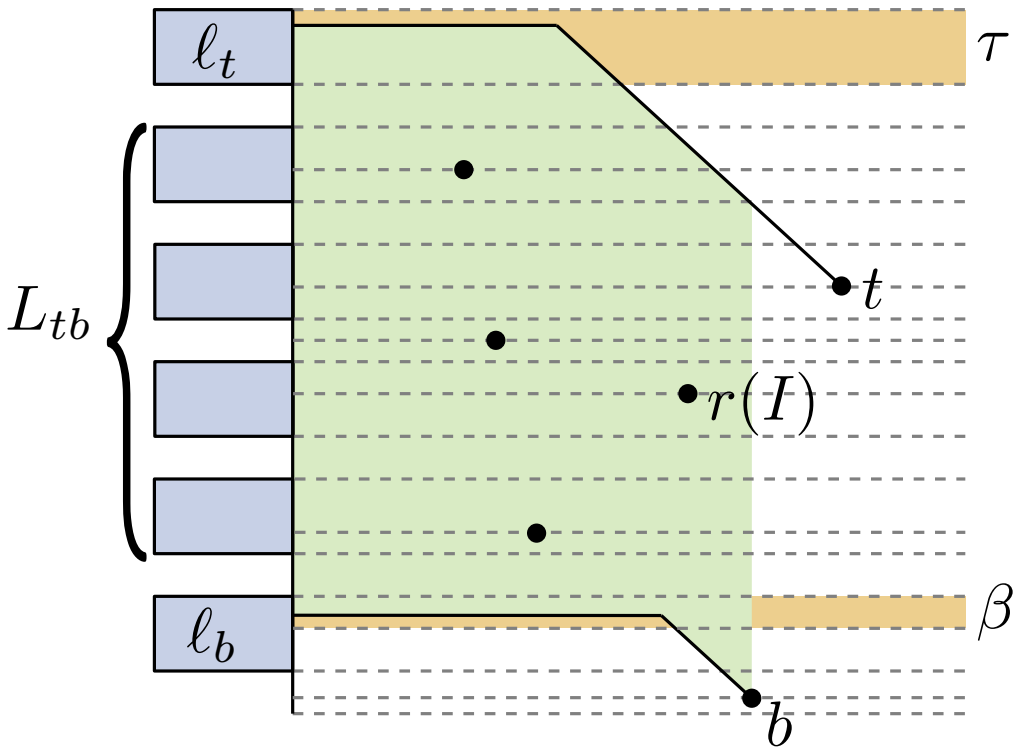
- Punkte rechts von b und t sind bereits angebunden.
- Labels in L_{tb} sind noch nicht angebunden.

→ Label in L_{tb} können nur mit Punkten in $R(\beta, \tau, b, t)$ verbunden werden.

→ $R(\beta, \tau, b, t)$ induziert unabhängige Instanz I.

Dynamisches Programm

Betrachte folgende Instanz:



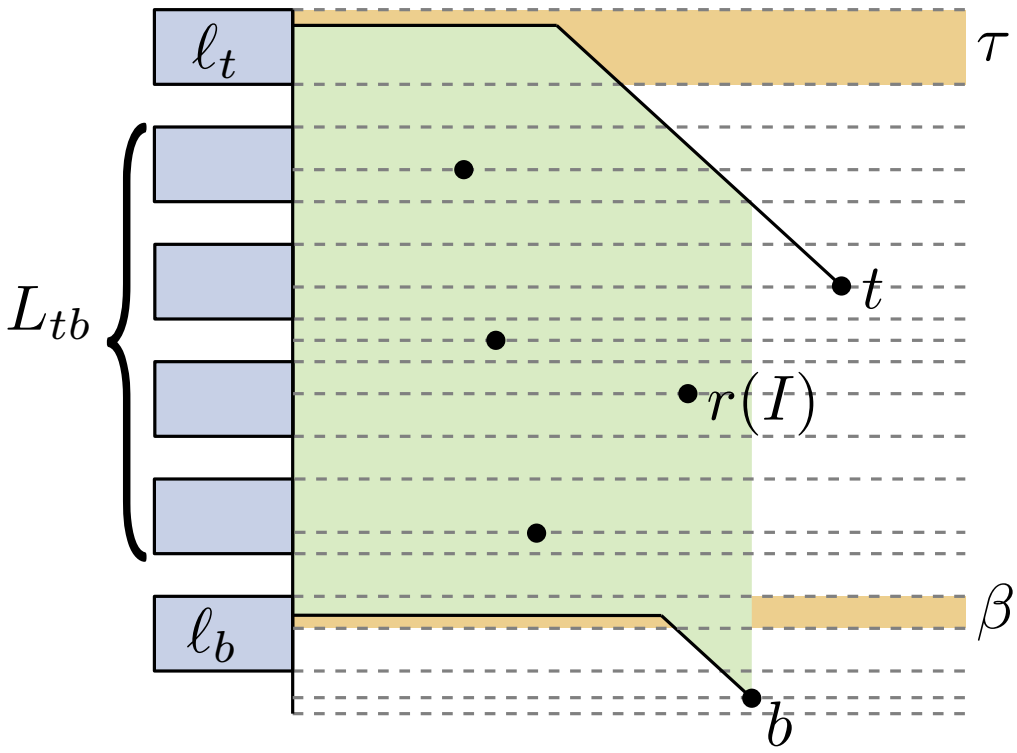
Schreibe $I = (\beta, \tau, b, t)$ für Instanz I induziert durch $R(\beta, \tau, b, t)$

$r(I)$: rechtester Punkt in I.

Betrachte alle *zulässigen* Streifen σ von $r(I)$.

Dynamisches Programm

Betrachte folgende Instanz:



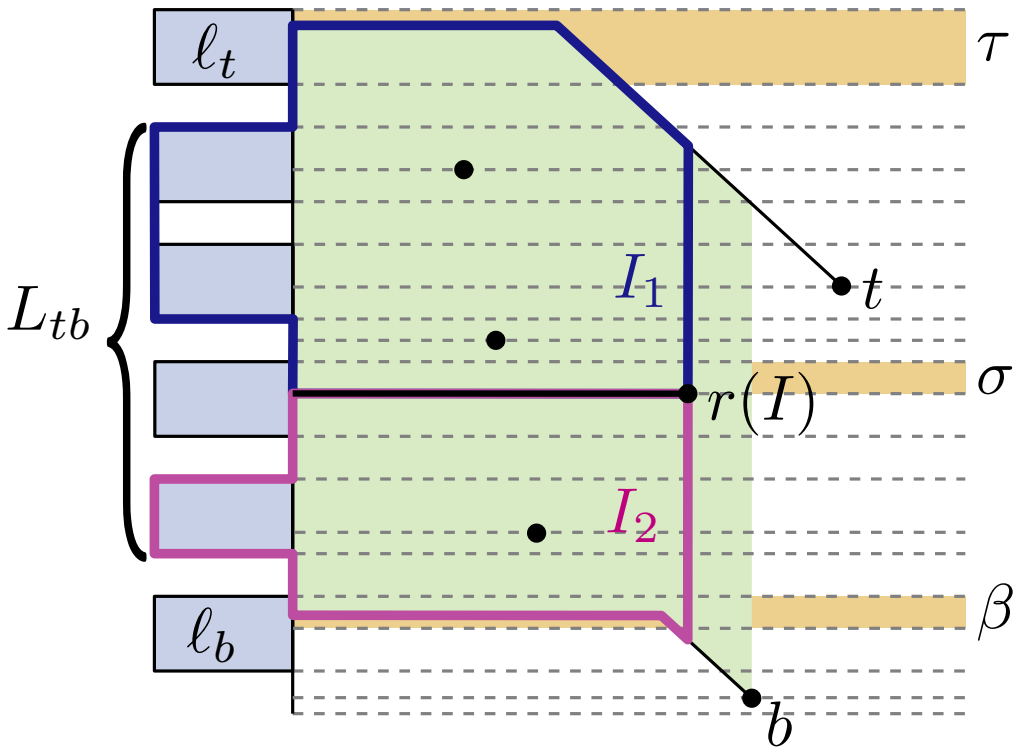
Schreibe $I = (\beta, \tau, b, t)$ für Instanz I induziert durch $R(\beta, \tau, b, t)$

$r(I)$: rechtester Punkt in I.

Betrachte alle *zulässigen* Streifen σ von $r(I)$.

Dynamisches Programm

Betrachte folgende Instanz:



Schreibe $I = (\beta, \tau, b, t)$ für Instanz I induziert durch $R(\beta, \tau, b, t)$

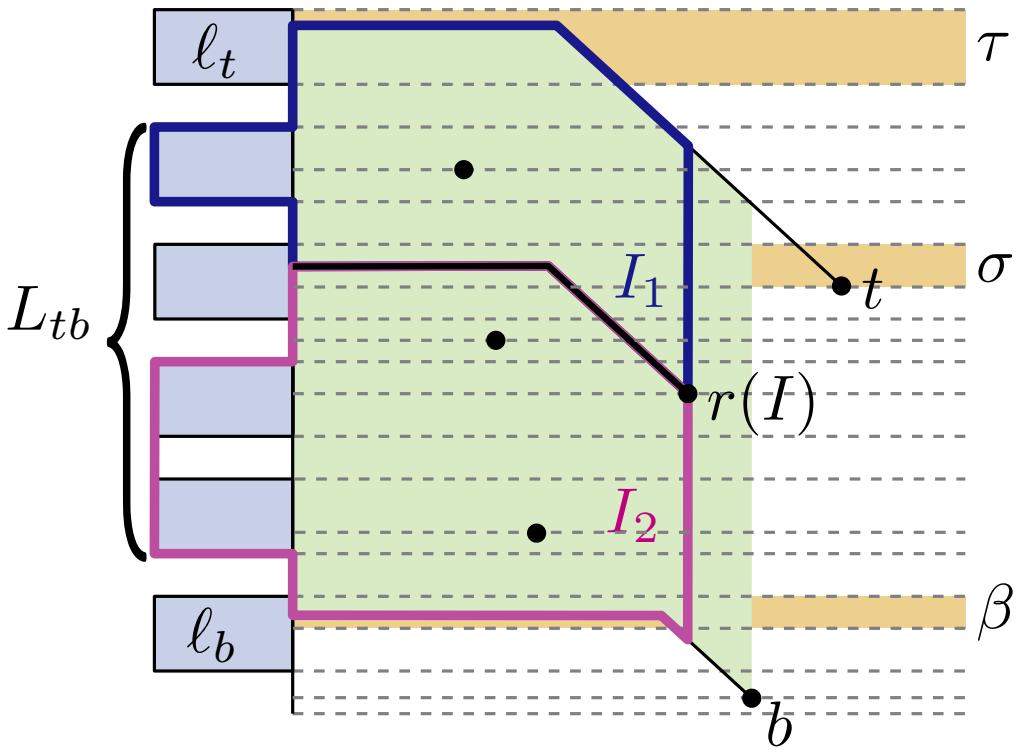
$r(I)$: rechtester Punkt in I.

Betrachte alle *zulässigen* Streifen σ von $r(I)$.

→ I zerfällt in Teilinstanzen $I_1 = (\sigma, \tau, r(I), t)$ und $I_2 = (\beta, \sigma, b, r(I))$

Dynamisches Programm

Betrachte folgende Instanz:



Schreibe $I = (\beta, \tau, b, t)$ für Instanz I induziert durch $R(\beta, \tau, b, t)$

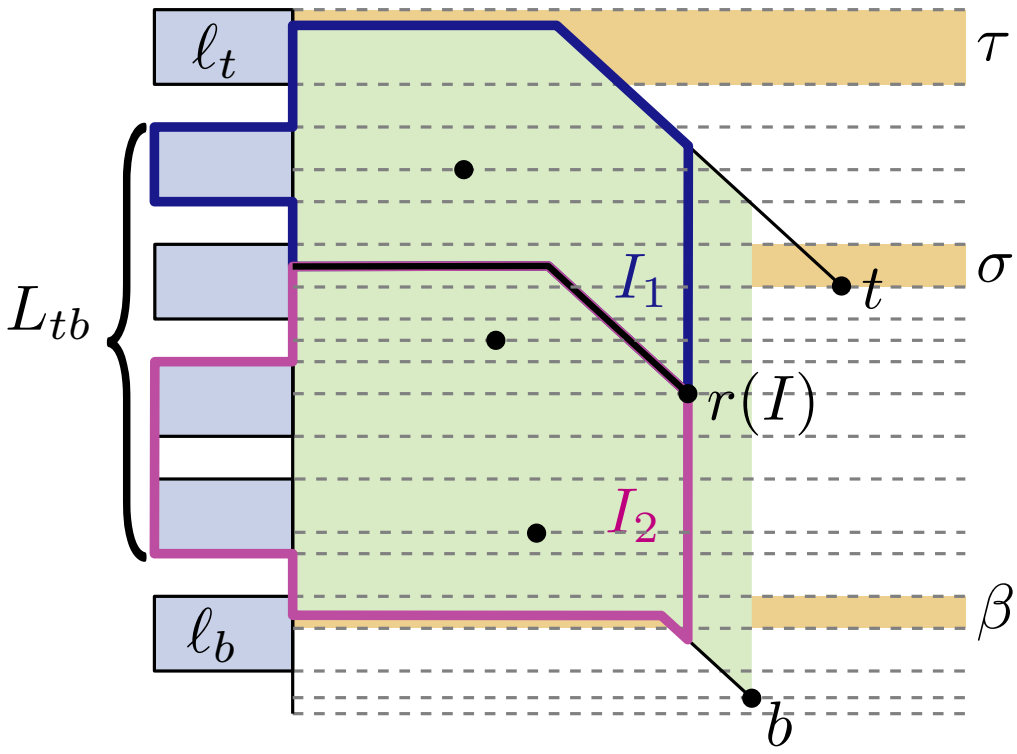
$r(I)$: rechtester Punkt in I.

Betrachte alle *zulässigen* Streifen σ von $r(I)$.

→ I zerfällt in Teilinstanzen $I_1 = (\sigma, \tau, r(I), t)$ und $I_2 = (\beta, \sigma, b, r(I))$

Dynamisches Programm

Betrachte folgende Instanz:



Schreibe $I = (\beta, \tau, b, t)$ für Instanz I induziert durch $R(\beta, \tau, b, t)$

$r(I)$: rechtester Punkt in I.

$S(\beta, \tau)$: Streifen zwischen β und τ.

Betrachte alle *zulässigen* Streifen σ von $r(I)$.

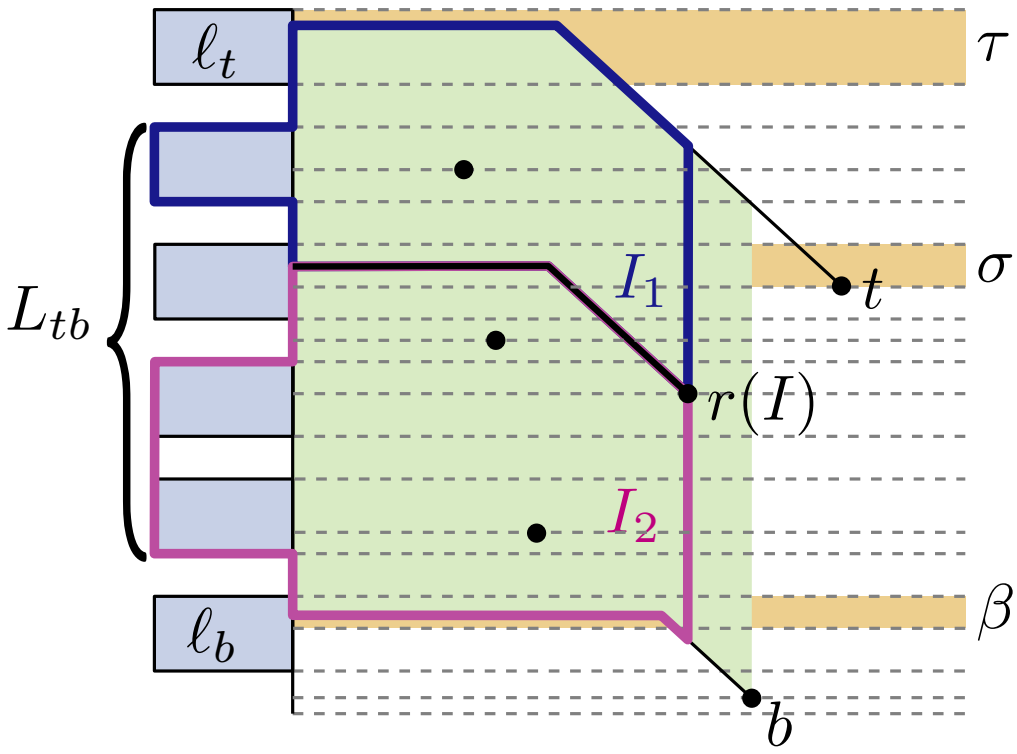
→ I zerfällt in Teilinstanzen $I_1 = (\sigma, \tau, r(I), t)$ und $I_2 = (\beta, \sigma, b, r(I))$

$$T[I] = \min_{\text{zul. } \sigma \in S(\beta,\tau)} \text{bad}(\lambda^\star(r(I), \sigma)) + T[I_1] + T[I_2]$$

$\lambda^\star(r, \sigma)$: optimaler Leader für Punkt r in Streifen σ

Dynamisches Programm

Betrachte folgende Instanz:



Schreibe $I = (\beta, \tau, b, t)$ für Instanz I induziert durch $R(\beta, \tau, b, t)$

$r(I)$: rechtester Punkt in I.

$S(\beta, \tau)$: Streifen zwischen β und τ.

$O(n^4)$ Speicher und
$O(n^5)$ Laufzeit.

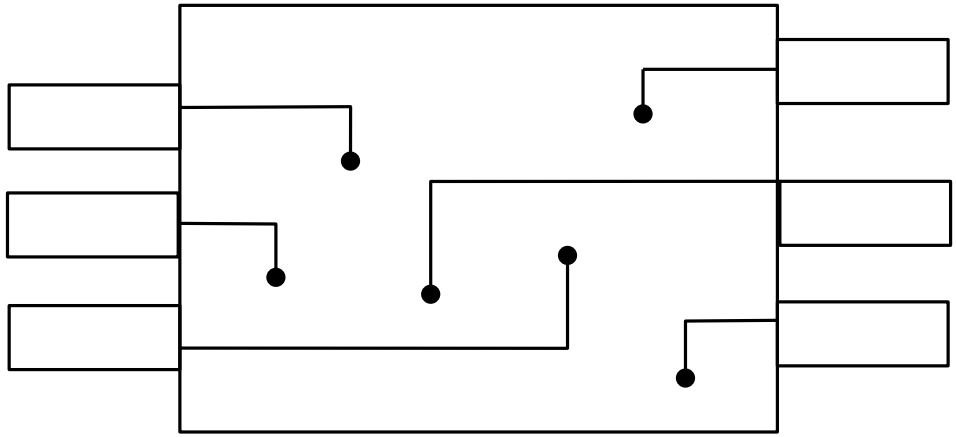
Betrachte alle *zulässigen* Streifen σ von $r(I)$.

→ I zerfällt in Teilinstanzen $I_1 = (\sigma, \tau, r(I), t)$ und $I_2 = (\beta, \sigma, b, r(I))$

$$T[I] = \min_{\text{zul. } \sigma \in S(\beta,\tau)} \text{bad}(\lambda^\star(r(I), \sigma)) + T[I_1] + T[I_2]$$

$\lambda^\star(r, \sigma)$: optimaler Leader für Punkt r in Streifen σ

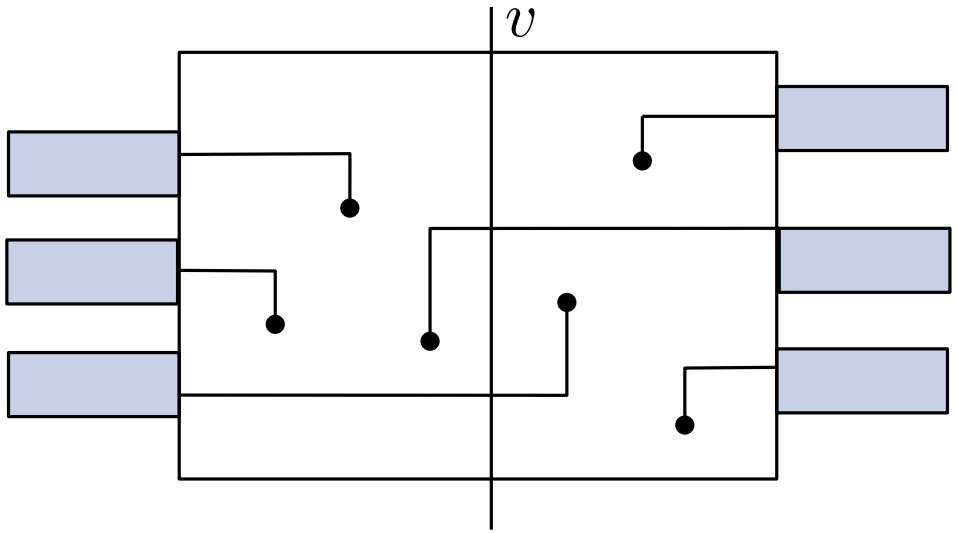
Allgemeine Bewertungsfunktion



- **zweiseitig (gegenüberliegende Seiten)**
- uniforme Label
- feste Labelpositionen
- kreuzungsfrei
- po-Leader
- sliding Ports
- **allgemeine Bewertungsfunktion**

Dynamisches Programm (Skizze)

Betrachte optimale Lösung $\mathcal{L}$ einer Instanz:

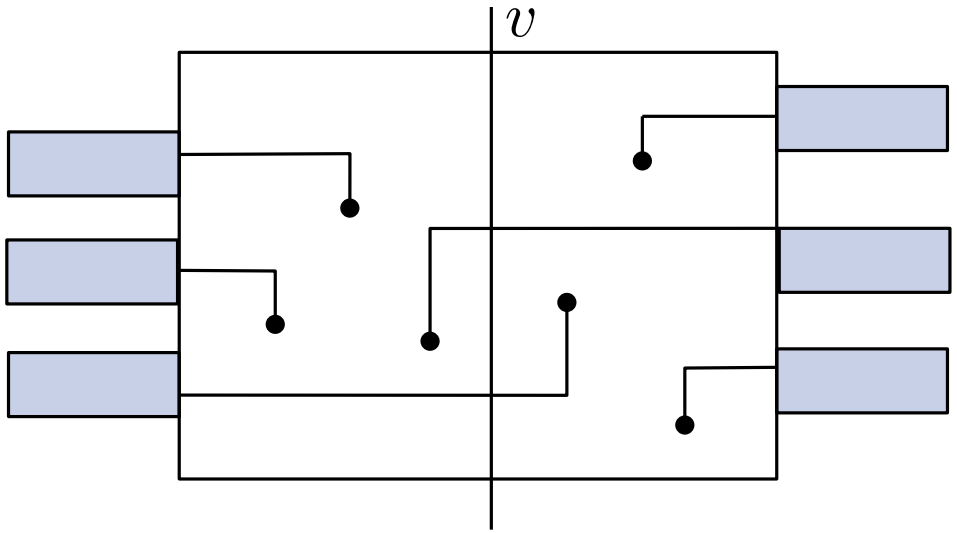


Sei v vertikale Gerade, sodass

- die Anzahl der Punkte und Labels links von v übereinstimmt,
- die Anzahl der Punkte und Labels rechts von v übereinstimmt.

Dynamisches Programm (Skizze)

Betrachte optimale Lösung $\mathcal{L}$ einer Instanz:



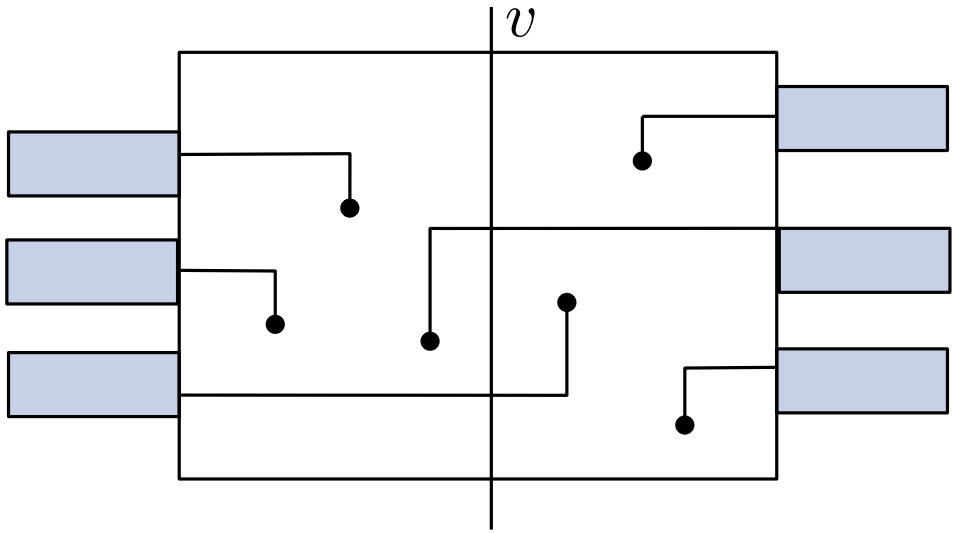
Sei v vertikale Gerade, sodass

- die Anzahl der Punkte und Labels links von v übereinstimmt,
- die Anzahl der Punkte und Labels rechts von v übereinstimmt.

Annahme: Es gibt Leader die v schneiden.

Dynamisches Programm (Skizze)

Betrachte optimale Lösung $\mathcal{L}$ einer Instanz:



Sei v vertikale Gerade, sodass

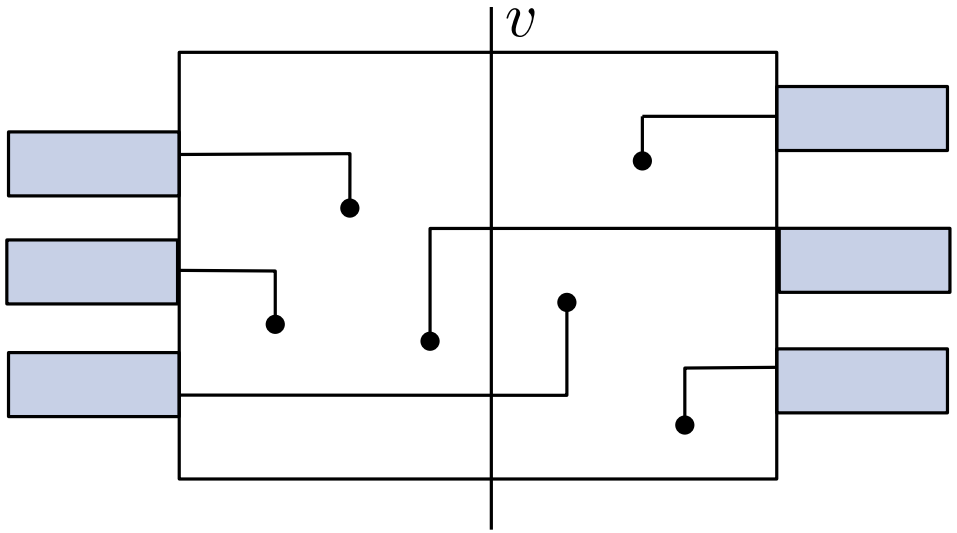
- die Anzahl der Punkte und Labels links von v übereinstimmt,
- die Anzahl der Punkte und Labels rechts von v übereinstimmt.

Annahme: Es gibt Leader die v schneiden.

→ Immer gleiche Anzahl *linker* und *rechter* Leader schneiden v.

Dynamisches Programm (Skizze)

Betrachte optimale Lösung $\mathcal{L}$ einer Instanz:



Sei v vertikale Gerade, sodass

- die Anzahl der Punkte und Labels links von v übereinstimmt,
- die Anzahl der Punkte und Labels rechts von v übereinstimmt.

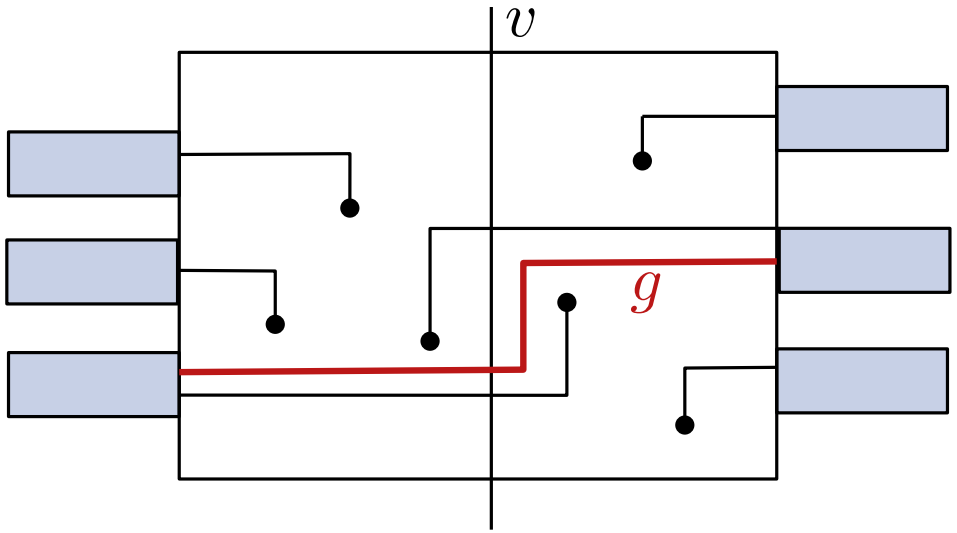
Annahme: Es gibt Leader die v schneiden.

→ Immer gleiche Anzahl *linker* und *rechter* Leader schneiden v.

→ Es gibt linken und rechten Leader, die v schneiden, sodass kein anderer Leader v dazwischen schneidet.

Dynamisches Programm (Skizze)

Betrachte optimale Lösung $\mathcal{L}$ einer Instanz:



Sei v vertikale Gerade, sodass

- die Anzahl der Punkte und Labels links von v übereinstimmt,
- die Anzahl der Punkte und Labels rechts von v übereinstimmt.

Annahme: Es gibt Leader die v schneiden.

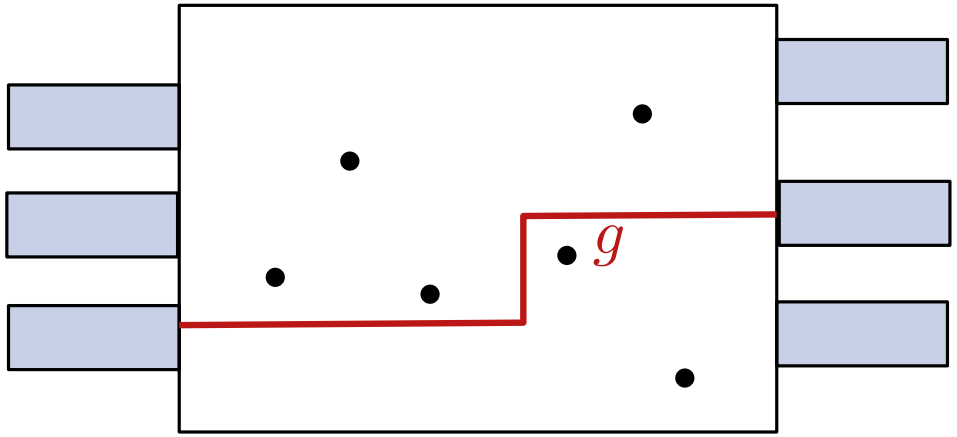
→ Immer gleiche Anzahl *linker* und *rechter* Leader schneiden v.

→ Es gibt linken und rechten Leader, die v schneiden, sodass kein anderer Leader v dazwischen schneidet.

→ Leader-Paar induziert unabhängige Teilinstanzen.

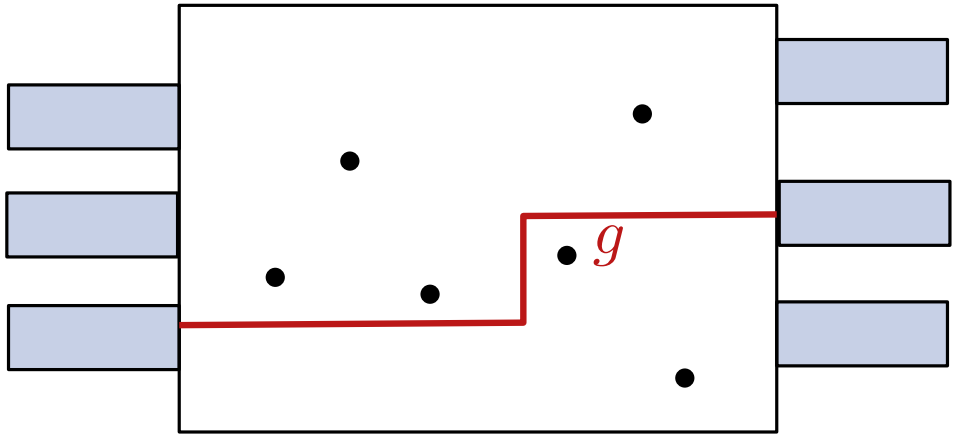
Dynamisches Programm (Skizze)

Wie viele verschiedene Trennkurven g kann es geben?



Dynamisches Programm (Skizze)

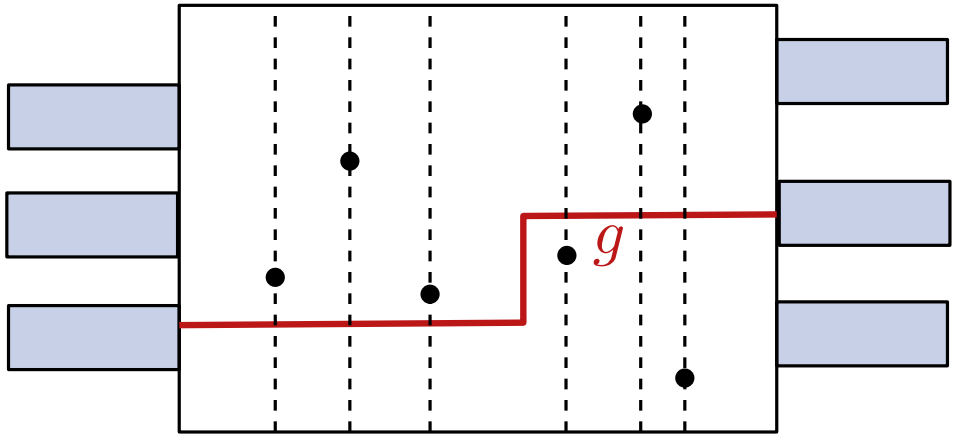
Wie viele verschiedene Trennkurven g kann es geben?



1. Es gibt $O(n^2)$ Paarungen der Labels.

Dynamisches Programm (Skizze)

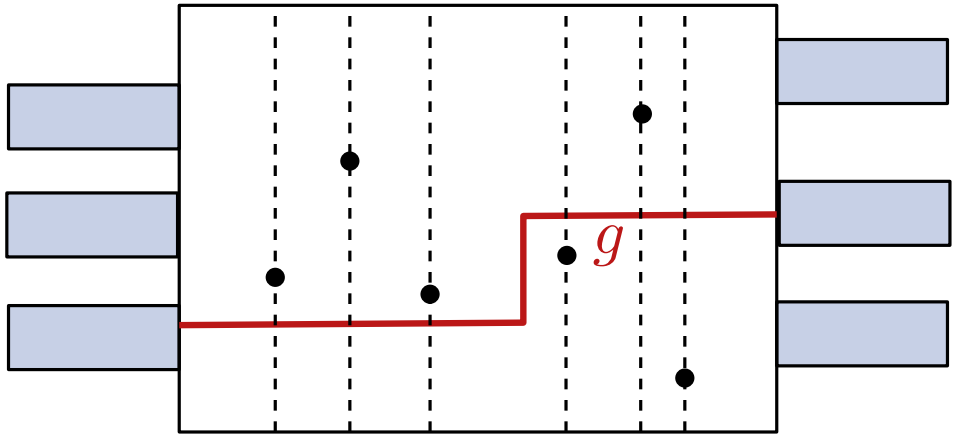
Wie viele verschiedene Trennkurven g kann es geben?



1. Es gibt $O(n^2)$ Paarungen der Labels.
2. Unterteile Rechteck in vertikale Streifen induziert durch Punkte in P.

Dynamisches Programm (Skizze)

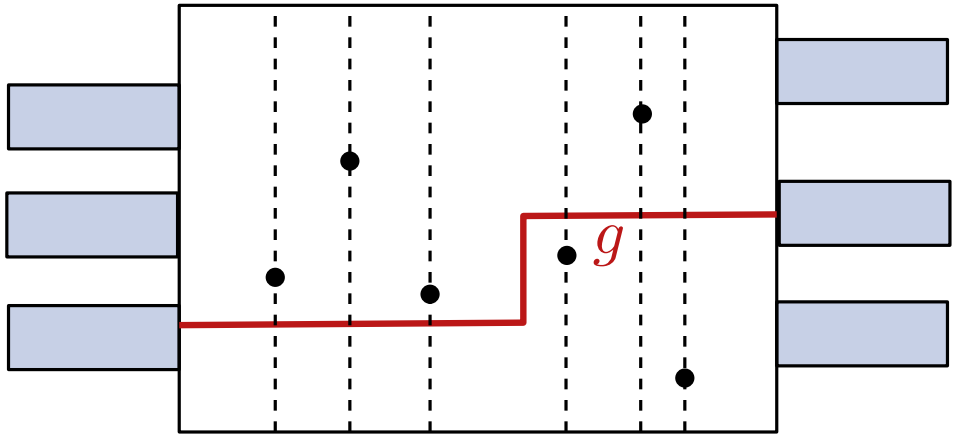
Wie viele verschiedene Trennkurven g kann es geben?



1. Es gibt $O(n^2)$ Paarungen der Labels.
2. Unterteile Rechteck in vertikale Streifen induziert durch Punkte in P.
   ⟶ vertikale Segment von g liegt in einem dieser Streifen.

Dynamisches Programm (Skizze)

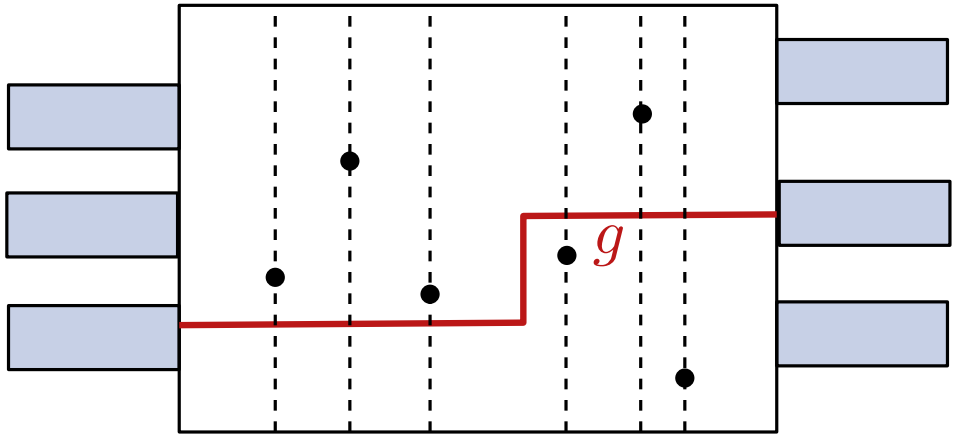
Wie viele verschiedene Trennkurven g kann es geben?



1. Es gibt $O(n^2)$ Paarungen der Labels.
2. Unterteile Rechteck in vertikale Streifen induziert durch Punkte in P.
   - ⟶ vertikale Segment von g liegt in einem dieser Streifen.
   - ⟶ Es gibt $O(n)$ viele mögliche Positionen für vertikales Segment.

Dynamisches Programm (Skizze)

Wie viele verschiedene Trennkurven g kann es geben?

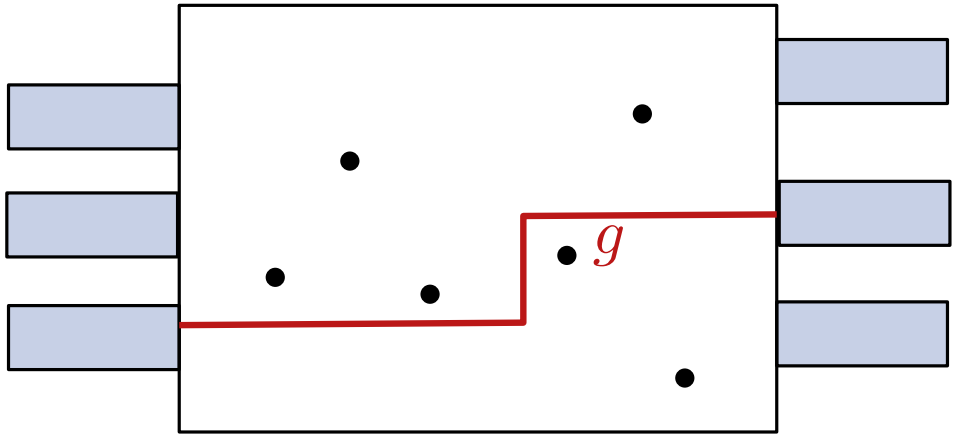


1. Es gibt $O(n^2)$ Paarungen der Labels.
2. Unterteile Rechteck in vertikale Streifen induziert durch Punkte in P.
   - → vertikale Segment von g liegt in einem dieser Streifen.
   - → Es gibt $O(n)$ viele mögliche Positionen für vertikales Segment.

→ Es gibt $O(n^3)$ viele Möglichkeiten für g.

Dynamisches Programm (Skizze)

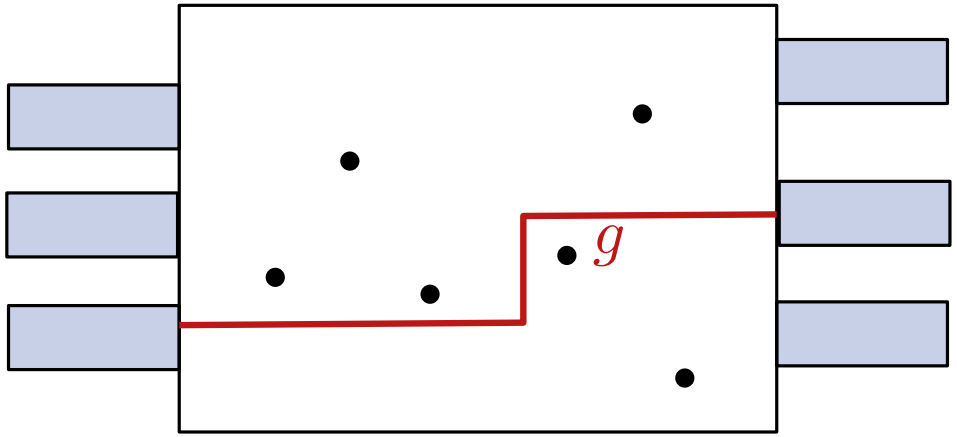
Wie viele verschiedene Trennkurven g kann es geben?



Unterscheide *zulässige* und *nicht zulässige* Trennkurven:

Dynamisches Programm (Skizze)

Wie viele verschiedene Trennkurven g kann es geben?



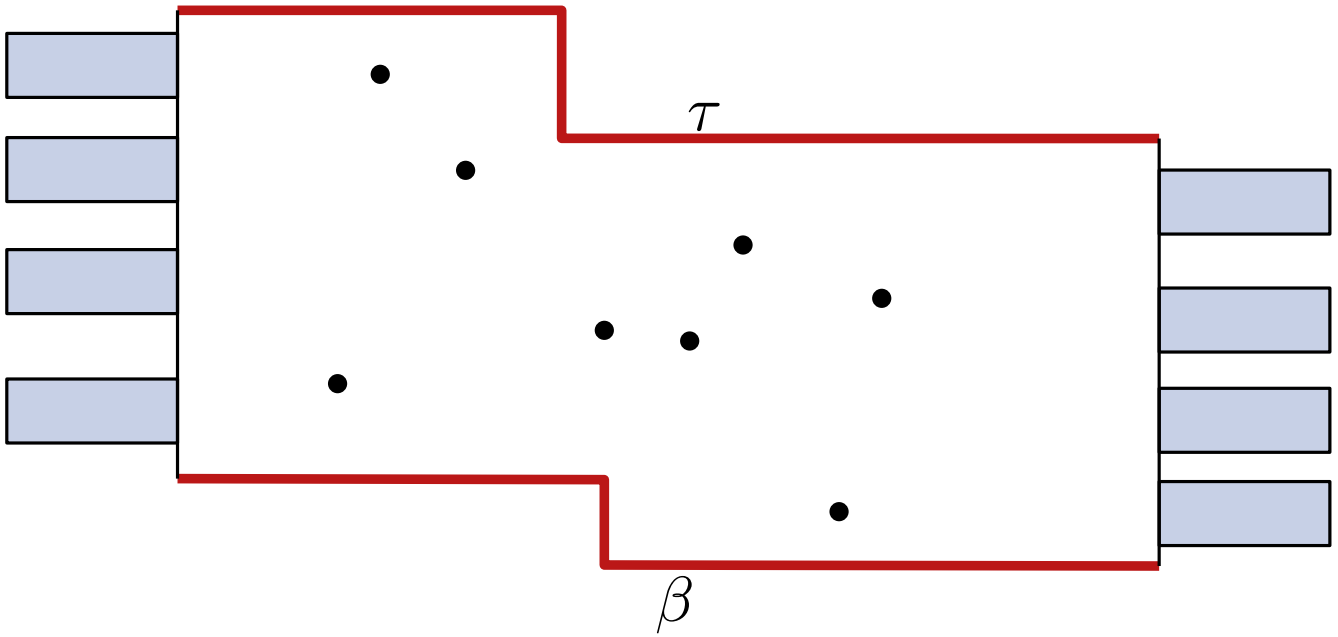
Unterscheide *zulässige* und *nicht zulässige* Trennkurven:

Für zulässige Instanzen muss gelten: #Punkte $\neq$ #Label für Teilinstanzen.

Dynamisches Programm (Skizze)

Teilinstanz ist definiert durch:

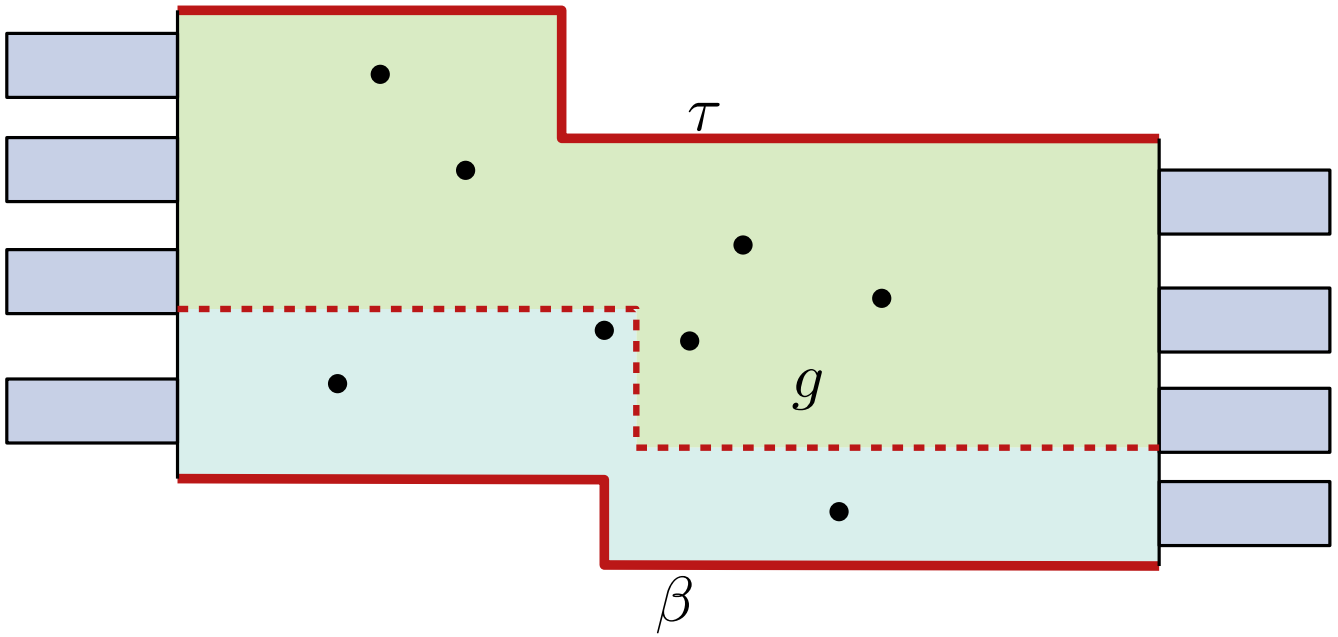
- untere Trennkurve β, und
- obere Trennkurve τ.



Dynamisches Programm (Skizze)

Teilinstanz ist definiert durch:

- untere Trennkurve β, und
- obere Trennkurve τ.

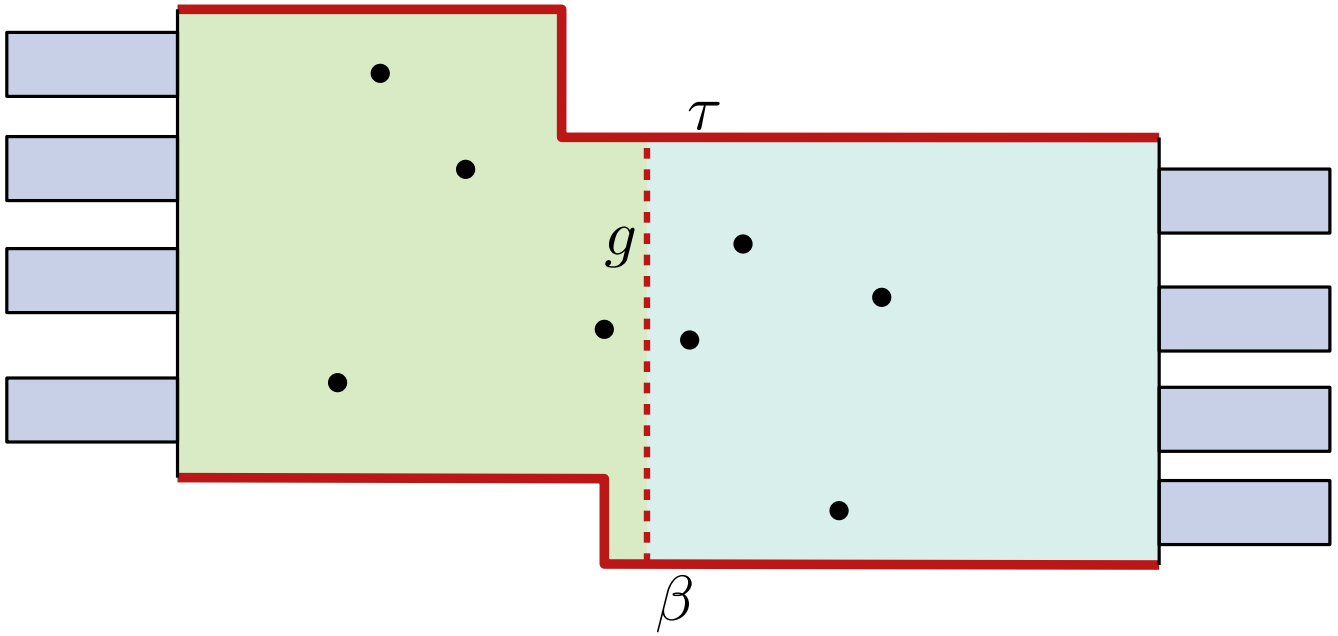


1. Probiere alle zulässigen Trennkurven durch.

Dynamisches Programm (Skizze)

Teilinstanz ist definiert durch:

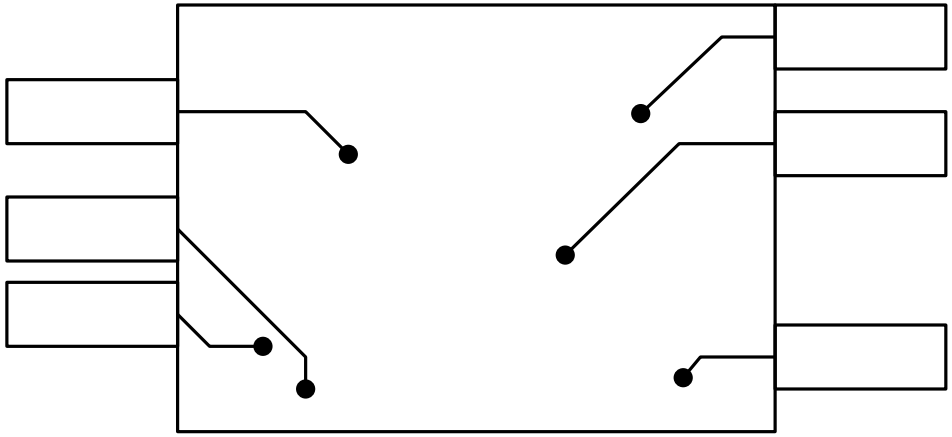
- untere Trennkurve β, und
- obere Trennkurve τ.



1. Probiere alle zulässigen Trennkurven durch.

2. Betrachte auch degenerierte Fälle:
Instanz zerfällt in zwei einseitige-Probleme.

Teil 2: Längenminimierung allgemein



- k-seitig
- uniforme Label
- feste Labelpositionen
- kreuzungsfrei
- verschiedene Leadertypen
- fixed Ports
- Längenminimierung

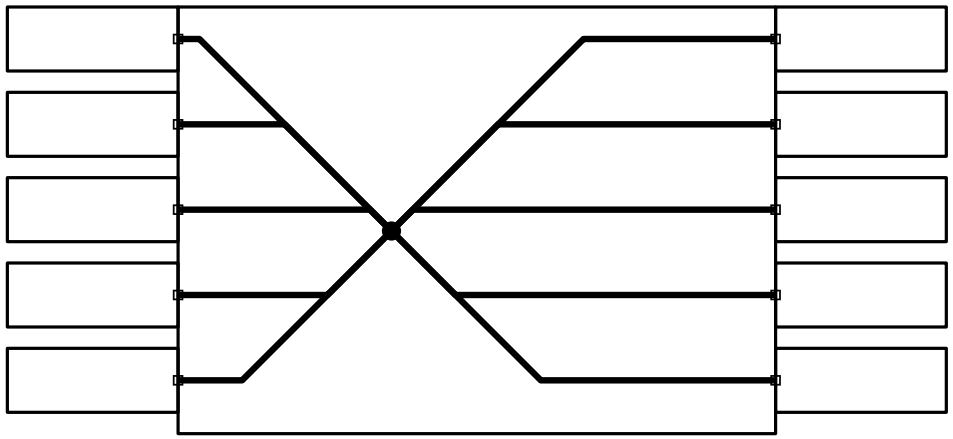
Generischer Algorithmus (Bekos et al. '10)

A) erstelle gewichteten vollständigen bipartiten Graphen $G = (P \cup L, E, w)$ für Punktmenge P, Labelmenge L, Punkt-Label-Kanten E und Kantenlängenfunktion w
B) bestimme bipartites Matching mit minimalem Gewicht in G
C) transformiere Matching in längenminimale Beschriftung
D) löse Kreuzungen längenneutral auf

Generischer Algorithmus (Bekos et al. '10)

A) erstelle gewichteten vollständigen bipartiten Graphen $G = (P \cup L, E, w)$ für Punktmenge P, Labelmenge L, Punkt-Label-Kanten E und Kantenlängenfunktion w
B) bestimme bipartites Matching mit minimalem Gewicht in G
C) transformiere Matching in längenminimale Beschriftung
D) löse Kreuzungen längenneutral auf

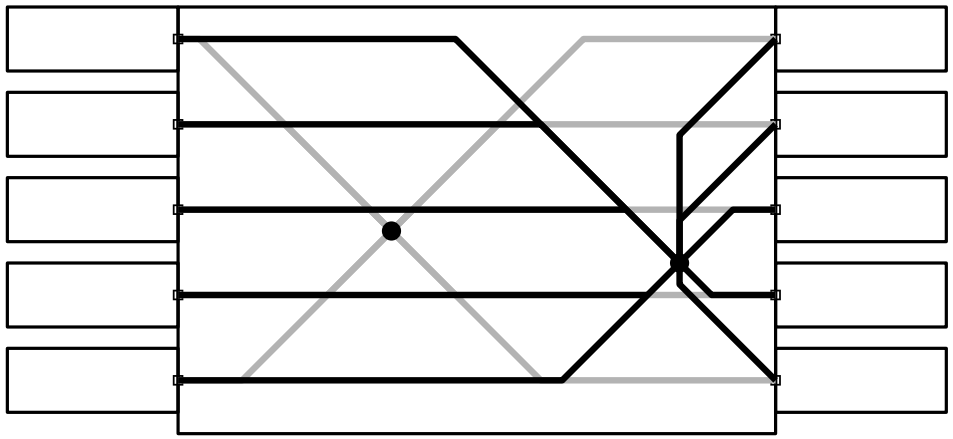
Beispiel: 2-seitig, do- und pd-Leader



Generischer Algorithmus (Bekos et al. '10)

A) erstelle gewichteten vollständigen bipartiten Graphen $G = (P \cup L, E, w)$ für Punktmenge P, Labelmenge L, Punkt-Label-Kanten E und Kantenlängenfunktion w
B) bestimme bipartites Matching mit minimalem Gewicht in G
C) transformiere Matching in längenminimale Beschriftung
D) löse Kreuzungen längenneutral auf

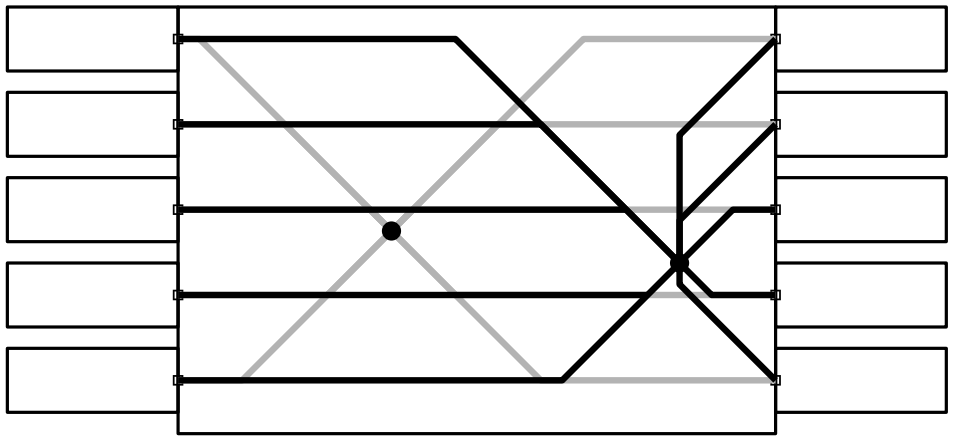
Beispiel: 2-seitig, do- und pd-Leader



Generischer Algorithmus (Bekos et al. '10)

A) erstelle gewichteten vollständigen bipartiten Graphen $G = (P \cup L, E, w)$ für Punktmenge P, Labelmenge L, Punkt-Label-Kanten E und Kantenlängenfunktion w
B) bestimme bipartites Matching mit minimalem Gewicht in G
C) transformiere Matching in längenminimale Beschriftung
D) löse Kreuzungen längenneutral auf

Beispiel: 2-seitig, do- und pd-Leader



Schritte A–C:

- Matching garantiert gültige Beschriftung
- Matchinggewicht entspricht Gesamtleaderlänge
- Schritt B in $O(n^3)$ Zeit [Kuhn '55]
- Kreuzungen werden ignoriert

Eigenschaften

Lemma 1: Sei M die Beschriftung aus Schritt C des Algorithmus. Dann gilt für sich kreuzende Leader

1. beide Leader gehen zur gleichen Seite von R
2. beide Leader sind vom gleichen Typ
3. beide Leader zeigen in die gleiche Richtung
4. durch Tauschen der Zuordnung löst sich die Kreuzung auf, die Gesamtlänge bleibt gleich, Typ und Orientierung bleiben gleich.

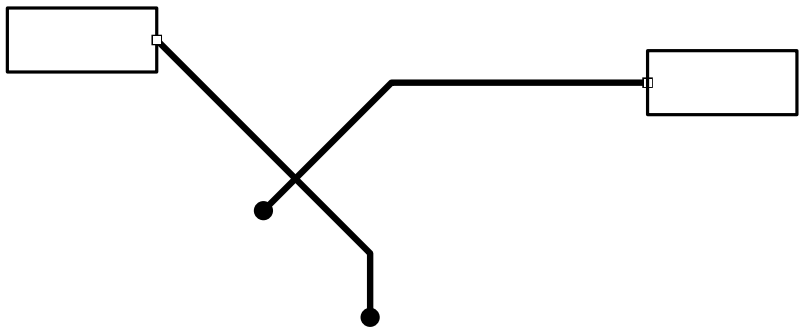
Eigenschaften

Lemma 1: Sei M die Beschriftung aus Schritt C des Algorithmus. Dann gilt für sich kreuzende Leader

1. beide Leader gehen zur gleichen Seite von R
2. beide Leader sind vom gleichen Typ
3. beide Leader zeigen in die gleiche Richtung
4. durch Tauschen der Zuordnung löst sich die Kreuzung auf, die Gesamtlänge bleibt gleich, Typ und Orientierung bleiben gleich.

Beweis:

1.) Fallunterscheidung, z.B.



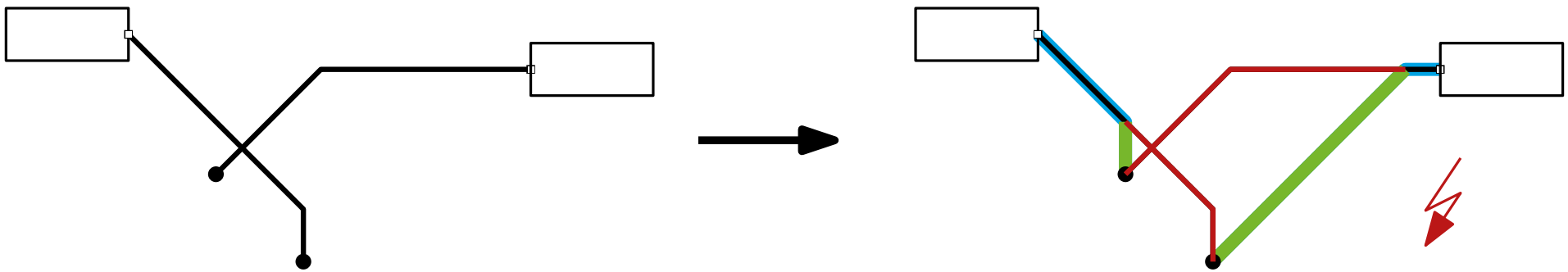
Eigenschaften

Lemma 1: Sei M die Beschriftung aus Schritt C des Algorithmus. Dann gilt für sich kreuzende Leader

1. beide Leader gehen zur gleichen Seite von R
2. beide Leader sind vom gleichen Typ
3. beide Leader zeigen in die gleiche Richtung
4. durch Tauschen der Zuordnung löst sich die Kreuzung auf, die Gesamtlänge bleibt gleich, Typ und Orientierung bleiben gleich.

Beweis:

1.) Fallunterscheidung, z.B.



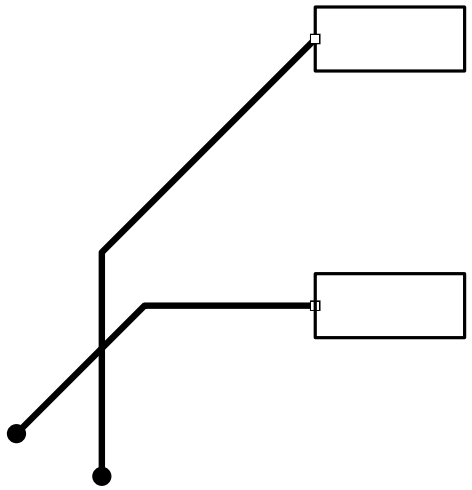
Eigenschaften

Lemma 1: Sei M die Beschriftung aus Schritt C des Algorithmus. Dann gilt für sich kreuzende Leader

1. beide Leader gehen zur gleichen Seite von R
2. beide Leader sind vom gleichen Typ
3. beide Leader zeigen in die gleiche Richtung
4. durch Tauschen der Zuordnung löst sich die Kreuzung auf, die Gesamtlänge bleibt gleich, Typ und Orientierung bleiben gleich.

Beweis:

2.) Fallunterscheidung, z.B.



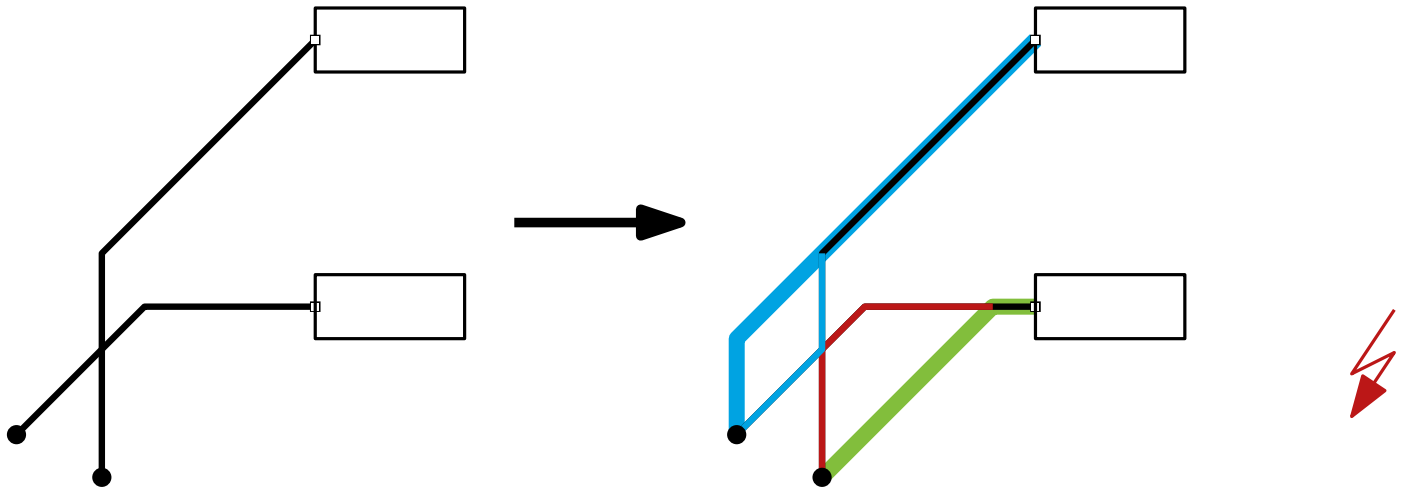
Eigenschaften

Lemma 1: Sei M die Beschriftung aus Schritt C des Algorithmus. Dann gilt für sich kreuzende Leader

1. beide Leader gehen zur gleichen Seite von R
2. beide Leader sind vom gleichen Typ
3. beide Leader zeigen in die gleiche Richtung
4. durch Tauschen der Zuordnung löst sich die Kreuzung auf, die Gesamtlänge bleibt gleich, Typ und Orientierung bleiben gleich.

Beweis:

2.) Fallunterscheidung, z.B.



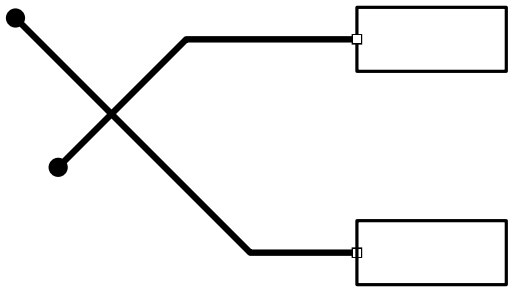
Eigenschaften

Lemma 1: Sei M die Beschriftung aus Schritt C des Algorithmus. Dann gilt für sich kreuzende Leader

1. beide Leader gehen zur gleichen Seite von R
2. beide Leader sind vom gleichen Typ
3. beide Leader zeigen in die gleiche Richtung
4. durch Tauschen der Zuordnung löst sich die Kreuzung auf, die Gesamtlänge bleibt gleich, Typ und Orientierung bleiben gleich.

Beweis:

3.) Fallunterscheidung, z.B.



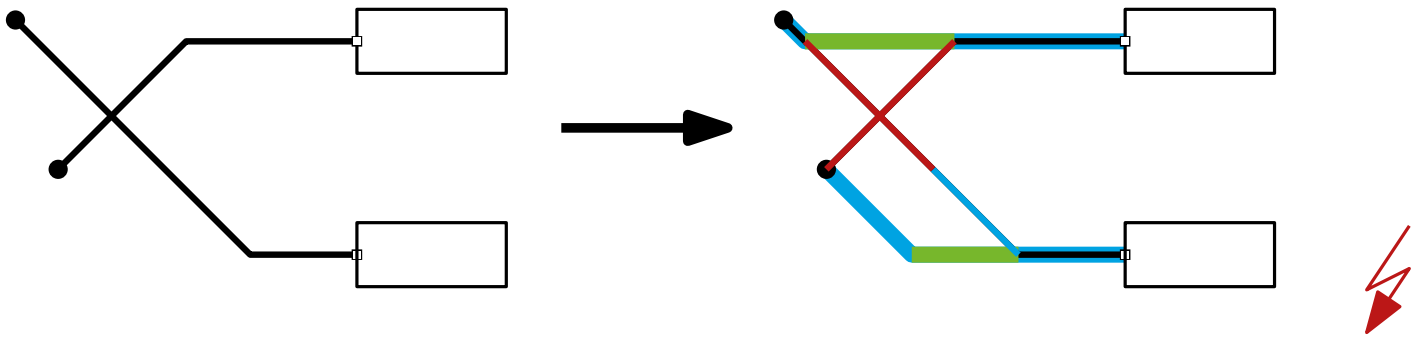
Eigenschaften

Lemma 1: Sei M die Beschriftung aus Schritt C des Algorithmus. Dann gilt für sich kreuzende Leader

1. beide Leader gehen zur gleichen Seite von R
2. beide Leader sind vom gleichen Typ
3. beide Leader zeigen in die gleiche Richtung
4. durch Tauschen der Zuordnung löst sich die Kreuzung auf, die Gesamtlänge bleibt gleich, Typ und Orientierung bleiben gleich.

Beweis:

3.) Fallunterscheidung, z.B.



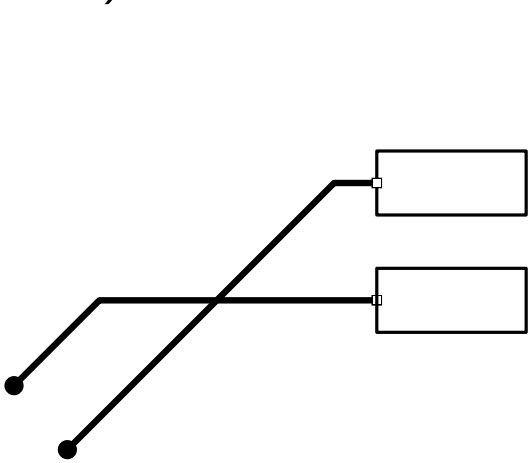
Eigenschaften

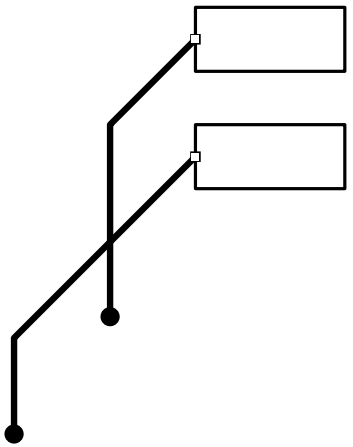
Lemma 1: Sei M die Beschriftung aus Schritt C des Algorithmus. Dann gilt für sich kreuzende Leader

1. beide Leader gehen zur gleichen Seite von R
2. beide Leader sind vom gleichen Typ
3. beide Leader zeigen in die gleiche Richtung
4. durch Tauschen der Zuordnung löst sich die Kreuzung auf, die Gesamtlänge bleibt gleich, Typ und Orientierung bleiben gleich.

Beweis:

4.) Fallunterscheidung





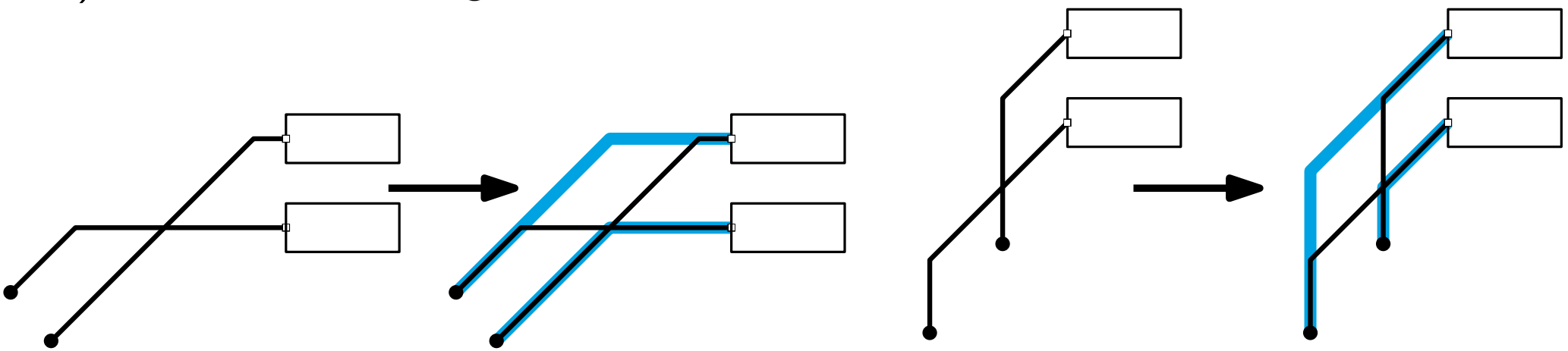
Eigenschaften

Lemma 1: Sei M die Beschriftung aus Schritt C des Algorithmus. Dann gilt für sich kreuzende Leader

1. beide Leader gehen zur gleichen Seite von R
2. beide Leader sind vom gleichen Typ
3. beide Leader zeigen in die gleiche Richtung
4. durch Tauschen der Zuordnung löst sich die Kreuzung auf, die Gesamtlänge bleibt gleich, Typ und Orientierung bleiben gleich.

Beweis:

4.) Fallunterscheidung



Sliding Ports

Wie kann der Algorithmus auf sliding Ports erweitert werden?

Sliding Ports

Wie kann der Algorithmus auf sliding Ports erweitert werden?

A) erstelle gewichteten vollständigen bipartiten Graphen $G = (P \cup L, E, w)$ für Punktmenge P, Labelmenge L, Punkt-Label-Kanten E und Kantenlängenfunktion w

Sliding Ports

Wie kann der Algorithmus auf sliding Ports erweitert werden?

A) erstelle gewichteten vollständigen bipartiten Graphen $G = (P \cup L, E, w)$ für Punktmenge P, Labelmenge L, Punkt-Label-Kanten E und Kantenlängenfunktion w

→ Für feste Ports sind Leader eindeutig: Somit Berechnung von w einfach.

Sliding Ports

Wie kann der Algorithmus auf sliding Ports erweitert werden?

A) erstelle gewichteten vollständigen bipartiten Graphen $G = (P \cup L, E, w)$ für Punktmenge P, Labelmenge L, Punkt-Label-Kanten E und Kantenlängenfunktion w

- Für feste Ports sind Leader eindeutig: Somit Berechnung von w einfach.
- Für sliding Ports sind Leader **nicht** eindeutig.

Sliding Ports

Wie kann der Algorithmus auf sliding Ports erweitert werden?

A) erstelle gewichteten vollständigen bipartiten Graphen $G = (P \cup L, E, w)$ für Punktmenge P, Labelmenge L, Punkt-Label-Kanten E und Kantenlängenfunktion w

- Für feste Ports sind Leader eindeutig: Somit Berechnung von w einfach.
- Für sliding Ports sind Leader **nicht** eindeutig.

Idee: Berechne für jeden Punkt p_i und jedes Label ℓ_j kürzesten Leader.

Sliding Ports

Wie kann der Algorithmus auf sliding Ports erweitert werden?

A) erstelle gewichteten vollständigen bipartiten Graphen $G = (P \cup L, E, w)$ für Punktmenge P, Labelmenge L, Punkt-Label-Kanten E und Kantenlängenfunktion w

- Für feste Ports sind Leader eindeutig: Somit Berechnung von w einfach.
- Für sliding Ports sind Leader **nicht** eindeutig.

Idee: Berechne für jeden Punkt p_i und jedes Label ℓ_j kürzesten Leader.

Problem:

d- und o-Leader können kürzeste Leader sein: bisher nicht betrachtet.

Sliding Ports

Wie kann der Algorithmus auf sliding Ports erweitert werden?

A) erstelle gewichteten vollständigen bipartiten Graphen $G = (P \cup L, E, w)$ für Punktmenge P, Labelmenge L, Punkt-Label-Kanten E und Kantenlängenfunktion w

- Für feste Ports sind Leader eindeutig: Somit Berechnung von w einfach.
- Für sliding Ports sind Leader **nicht** eindeutig.

Idee: Berechne für jeden Punkt p_i und jedes Label ℓ_j kürzesten Leader.

Problem:

d- und o-Leader können kürzeste Leader sein: bisher nicht betrachtet.

→ Lemmas nicht unbedingt korrekt für d- und o-Leader.

Sliding Ports

Wie kann der Algorithmus auf sliding Ports erweitert werden?

A) erstelle gewichteten vollständigen bipartiten Graphen $G = (P \cup L, E, w)$ für Punktmenge P, Labelmenge L, Punkt-Label-Kanten E und Kantenlängenfunktion w

- Für feste Ports sind Leader eindeutig: Somit Berechnung von w einfach.
- Für sliding Ports sind Leader **nicht** eindeutig.

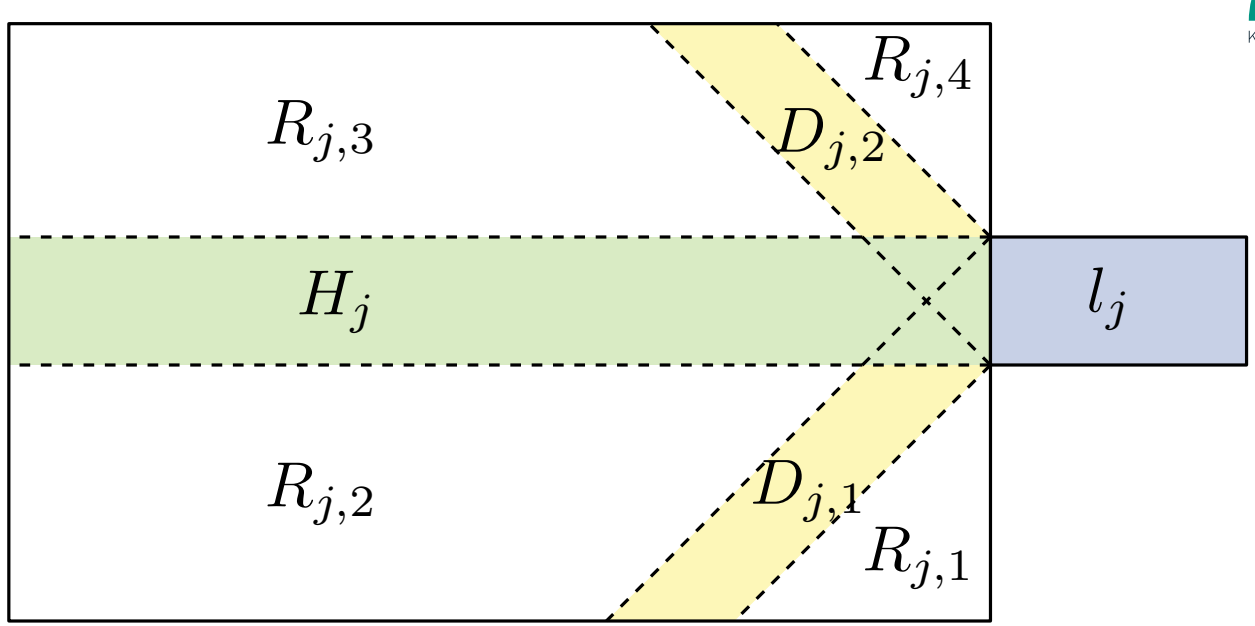
Idee: Berechne für jeden Punkt p_i und jedes Label ℓ_j kürzesten Leader.

Problem:

d- und o-Leader können kürzeste Leader sein: bisher nicht betrachtet.

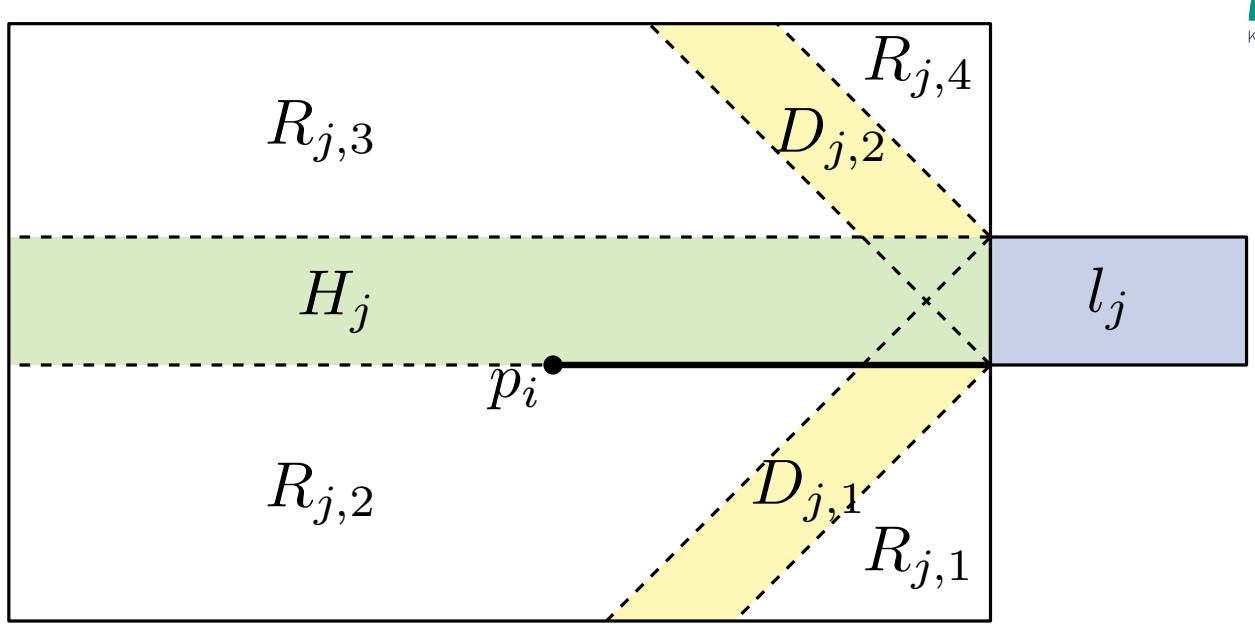
→ Lemmas nicht unbedingt korrekt für d- und o-Leader.

Sliding Ports



Kürzeste Leader:

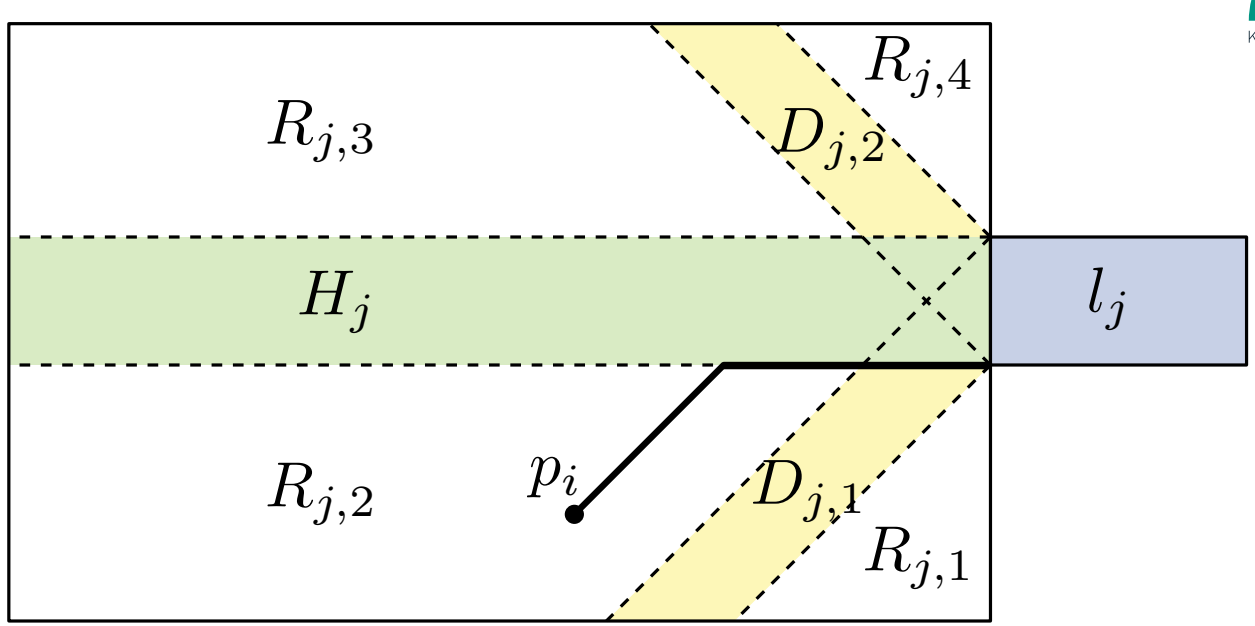
Sliding Ports



Kürzeste Leader:

Wenn p_i in H_j: o-Leader

Sliding Ports



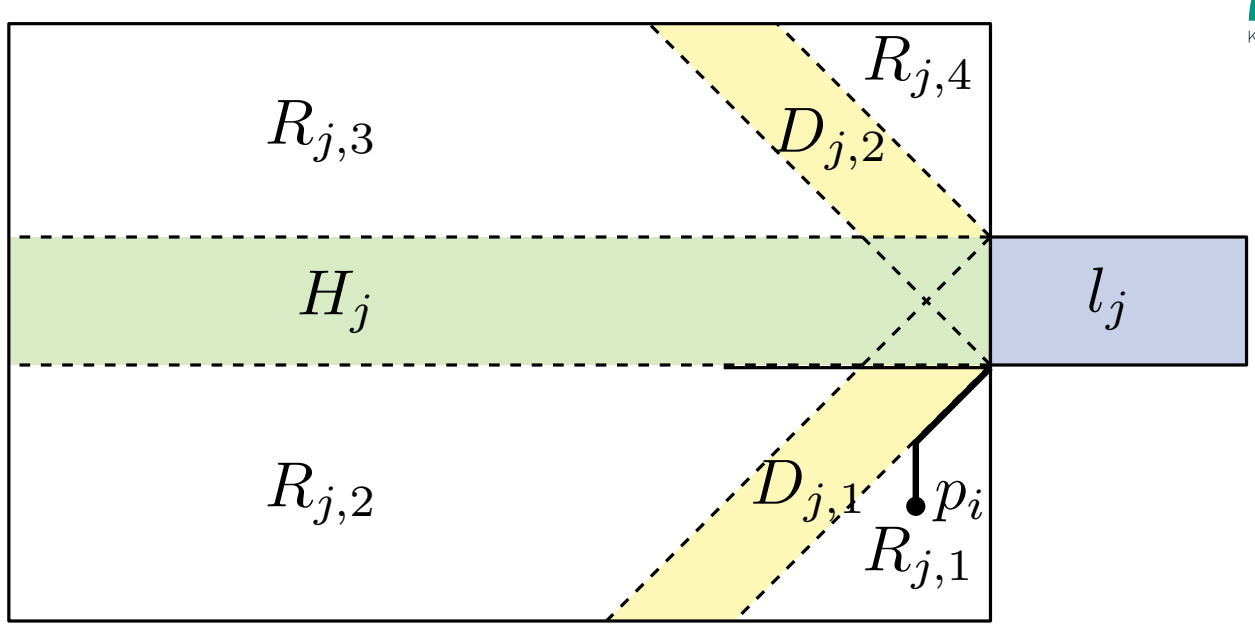
Kürzeste Leader:

Wenn p_i in H_j: o-Leader

Wenn p_i in $R_{j,2} \cup D_{j,1}$: od/do-Leader zur linken unteren Ecke von l_j

Wenn p_i in $R_{j,3} \cup D_{j,2}$: od/do-Leader zur linken oberen Ecke von l_j

Sliding Ports



Kürzeste Leader:

Wenn p_i in H_j: o-Leader

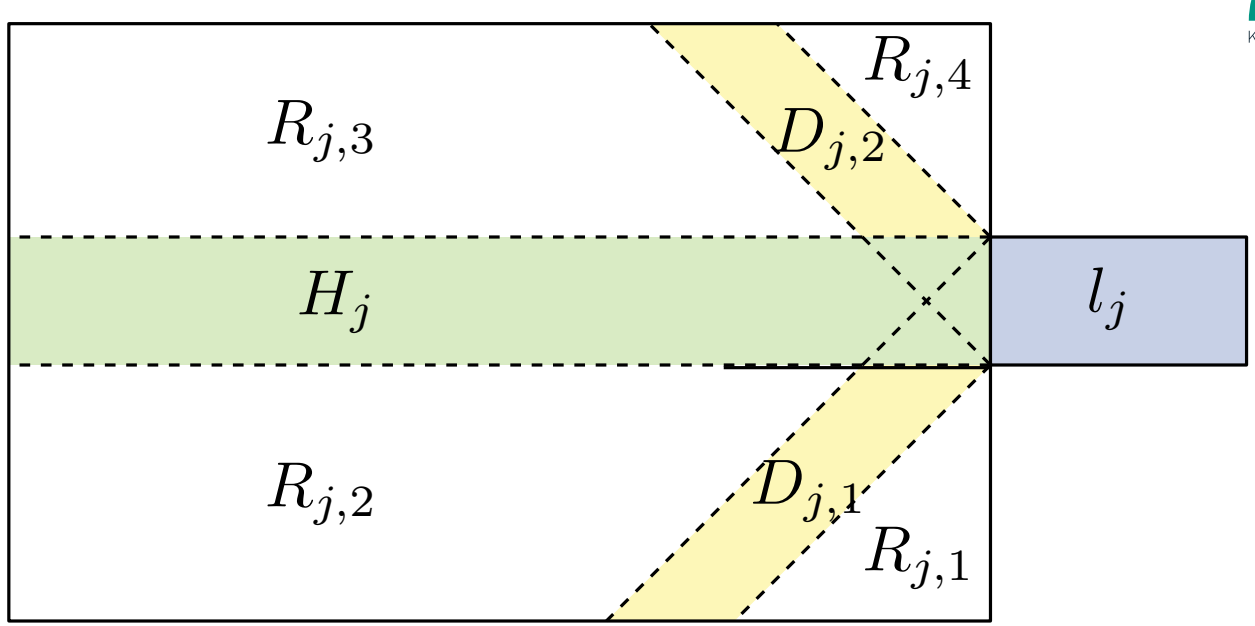
Wenn p_i in $R_{j,2} \cup D_{j,1}$: od/do-Leader zur linken unteren Ecke von l_j

Wenn p_i in $R_{j,3} \cup D_{j,2}$: od/do-Leader zur linken oberen Ecke von l_j

Wenn p_i in $R_{j,1}$: pd-Leader zur linken unteren Ecke von l_j

Wenn p_i in $R_{j,1}$: pd-Leader zur linken oberen Ecke von l_j

Sliding Ports



Kürzeste Leader:

Wenn p_i in H_j: o-Leader

Wenn p_i in $R_{j,2} \cup D_{j,1}$: od/do-Leader zur linken unteren Ecke von l_j

Wenn p_i in $R_{j,3} \cup D_{j,2}$: od/do-Leader zur linken oberen Ecke von l_j

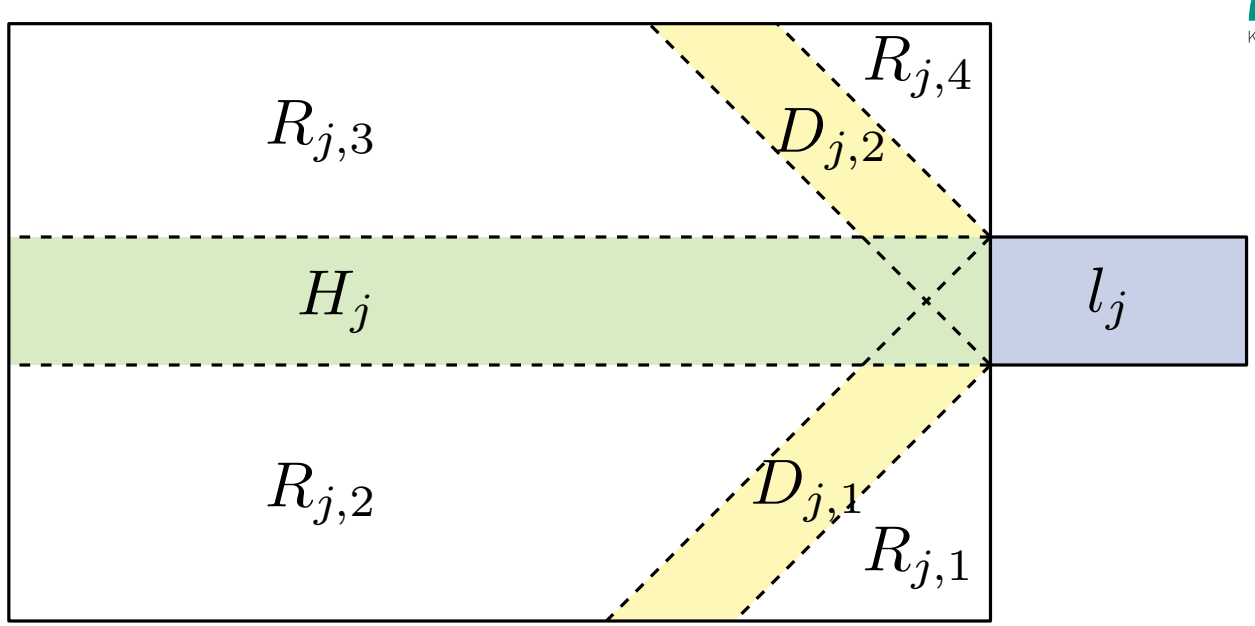
Wenn p_i in $R_{j,1}$: pd-Leader zur linken unteren Ecke von l_j

Wenn p_i in $R_{j,1}$: pd-Leader zur linken oberen Ecke von l_j

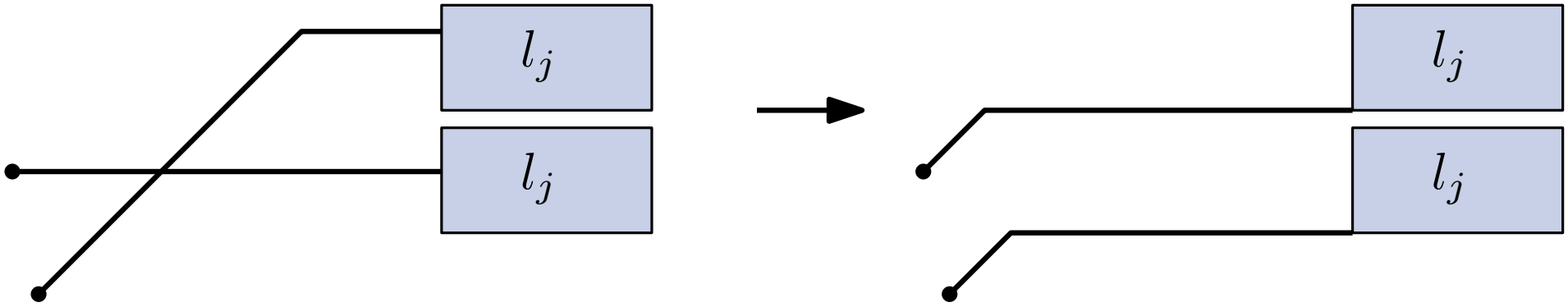
→ Alle bis auf o-Leader enden auf einer Ecke von l_j.

→ Wegen allgemeiner Lage: d-Leader nicht möglich.

Sliding Ports

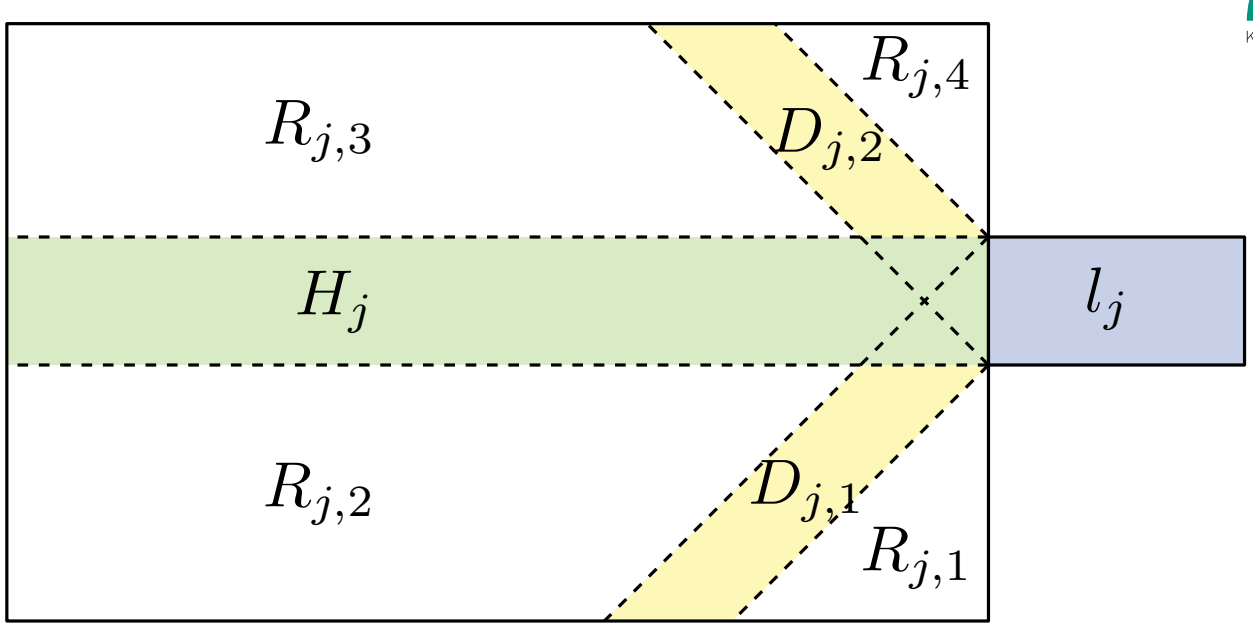


Keine Kreuzungen mit o-Leader möglich, da sonst nicht minimal:

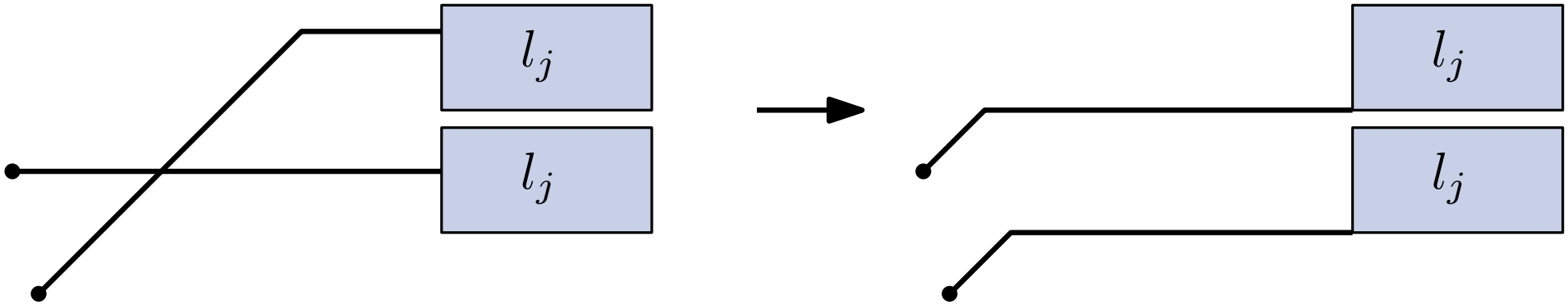


nicht minimal

Sliding Ports



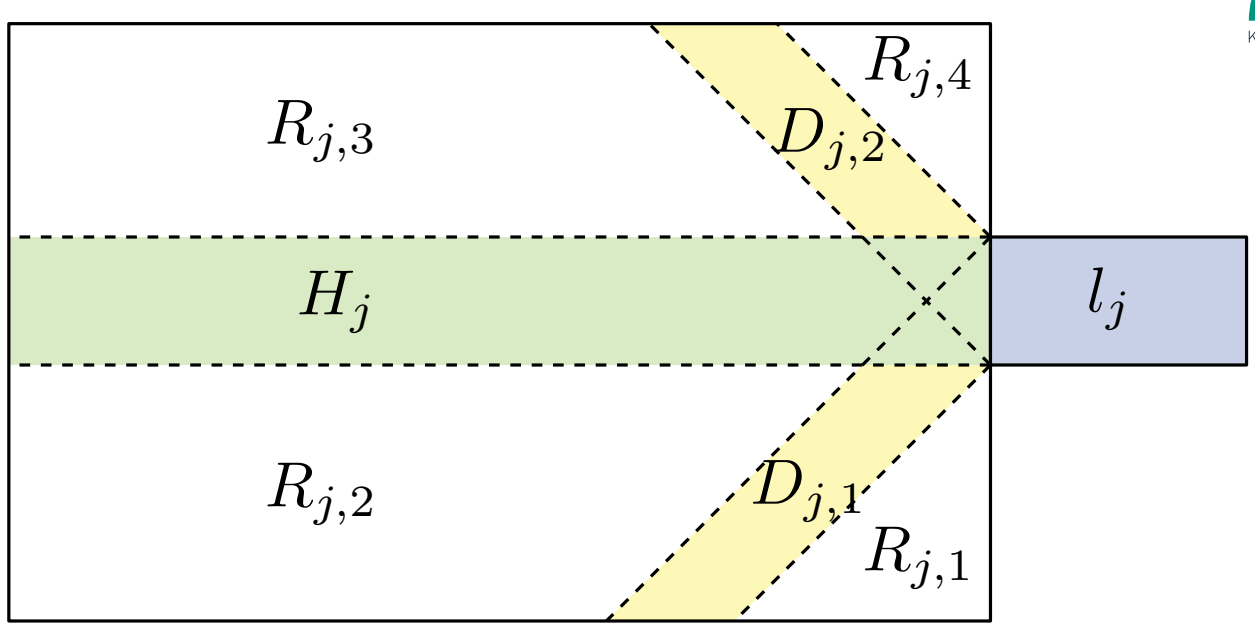
Keine Kreuzungen mit o-Leader möglich, da sonst nicht minimal:



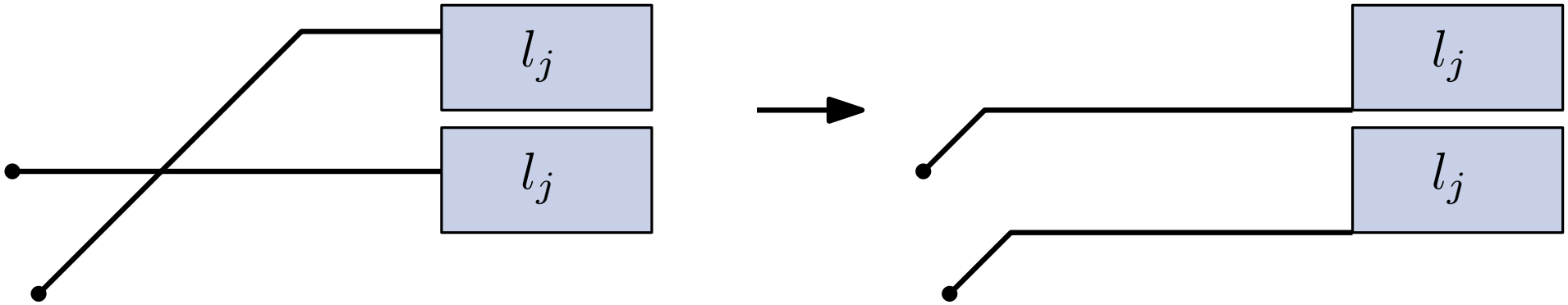
nicht minimal

→ d- und o-Leader doch kein Problem: Lemmas gelten weiterhin.

Sliding Ports



Keine Kreuzungen mit o-Leader möglich, da sonst nicht minimal:



nicht minimal

→ d- und o-Leader doch kein Problem: Lemmas gelten weiterhin.

→ Schritte B)-D) funktionieren wie bisher.