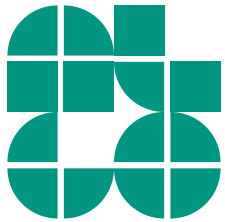
Übung Algorithmische Kartografie

Übungsblatt 8

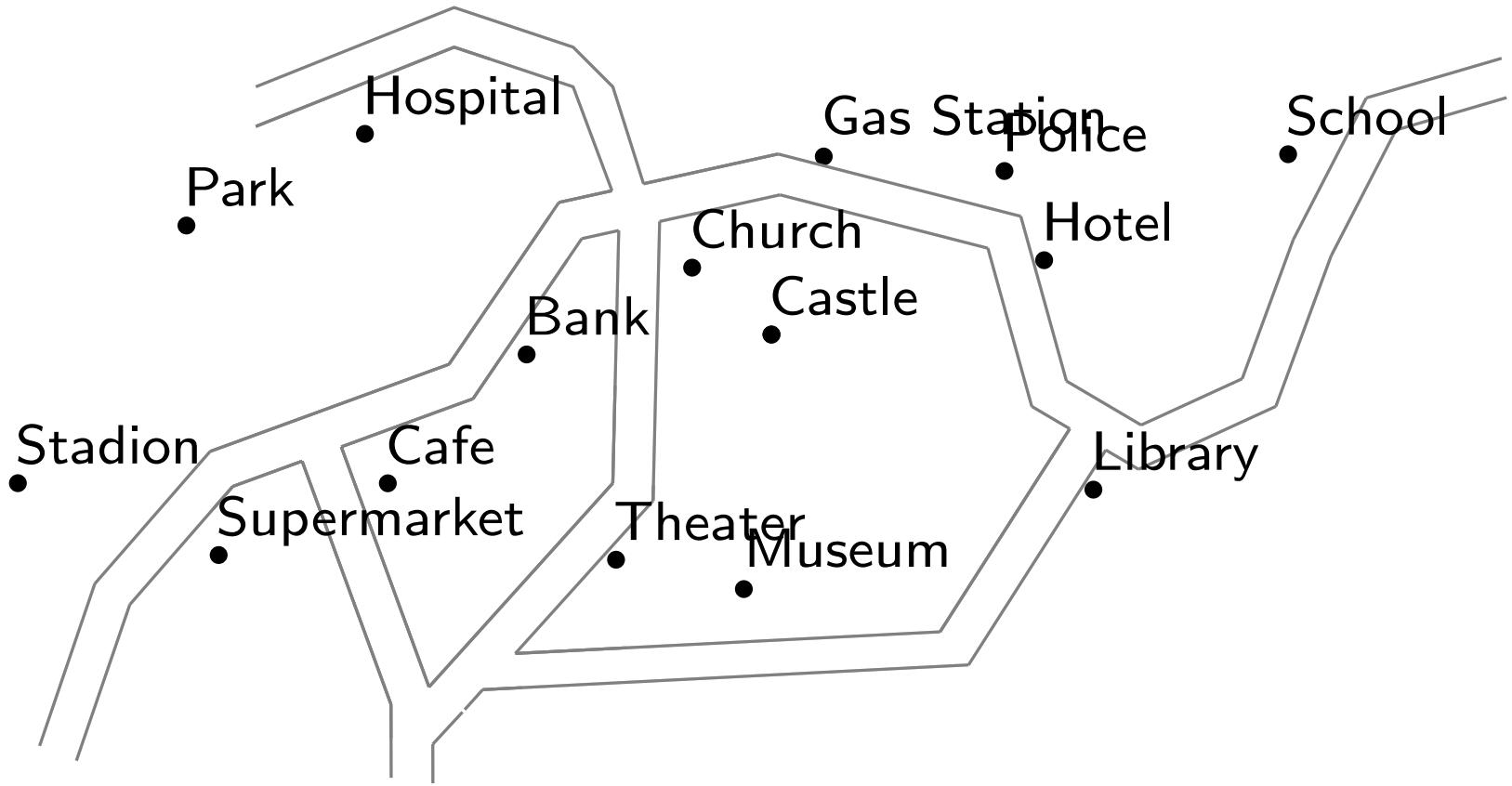
LEHRSTUHL FÜR ALGORITHMIK I · INSTITUT FÜR THEORETISCHE INFORMATIK · FAKULTÄT FÜR INFORMATIK

Benjamin Niedermann

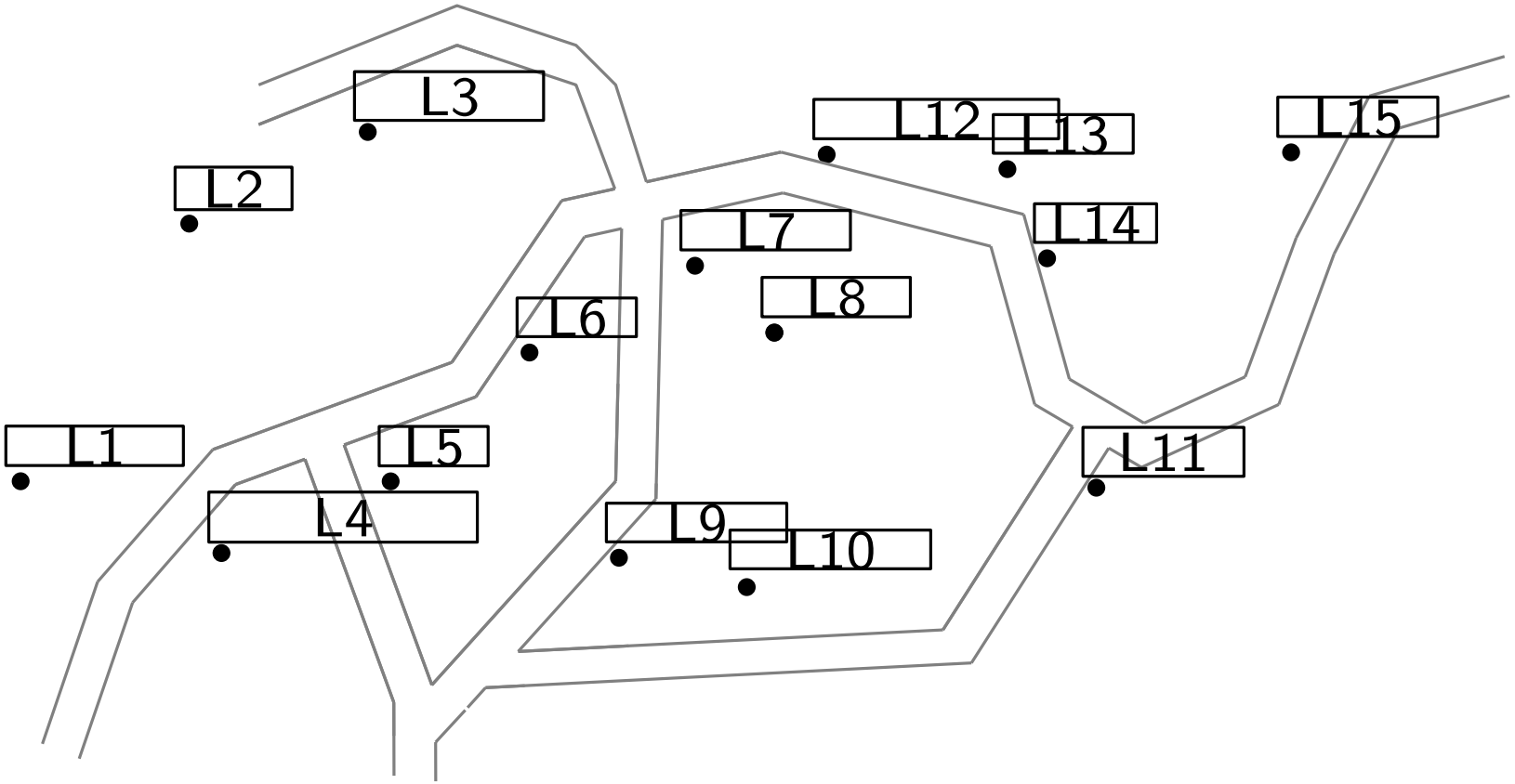
27.06.2013



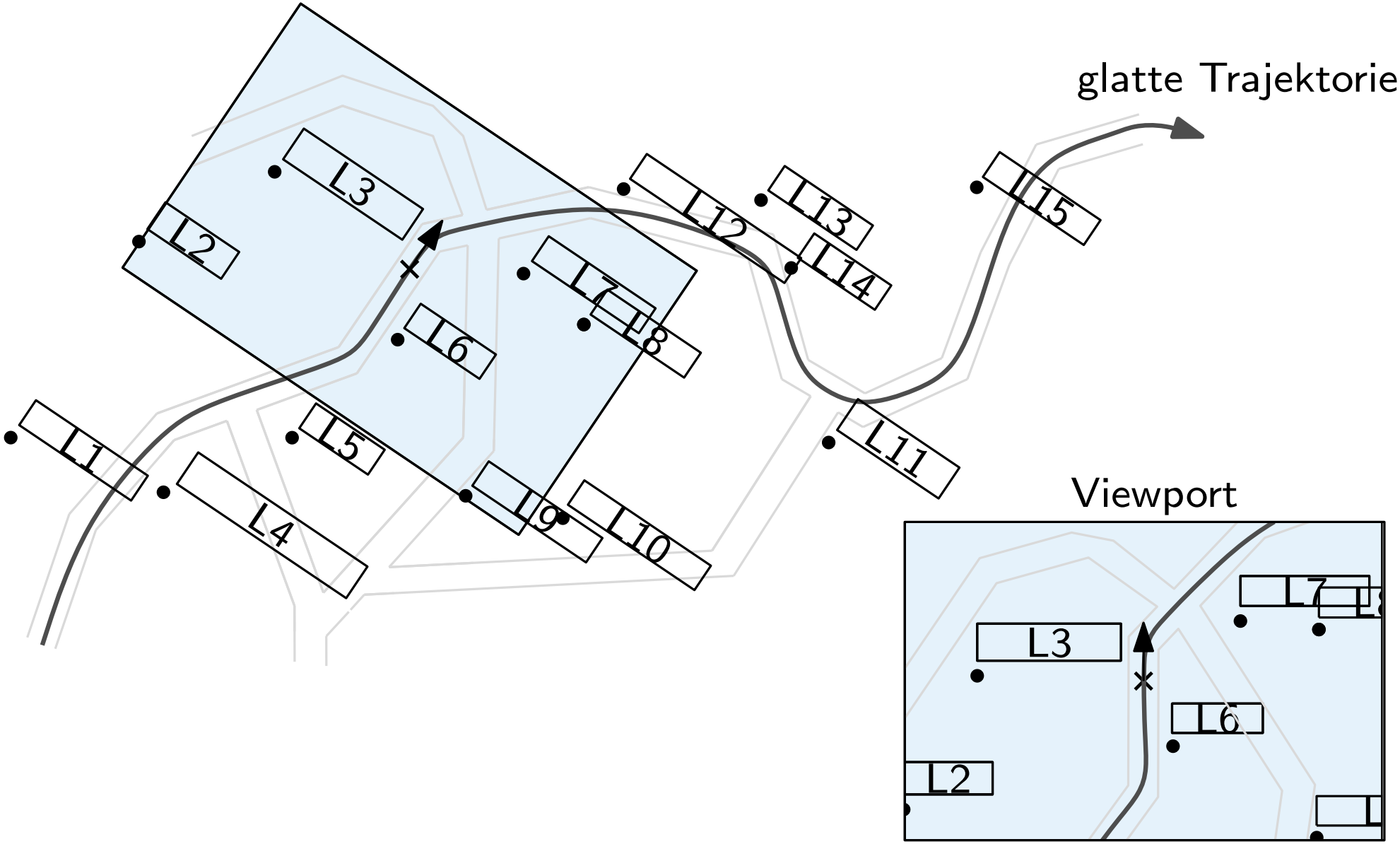
Trajectory-Based Dynamic Map Labeling



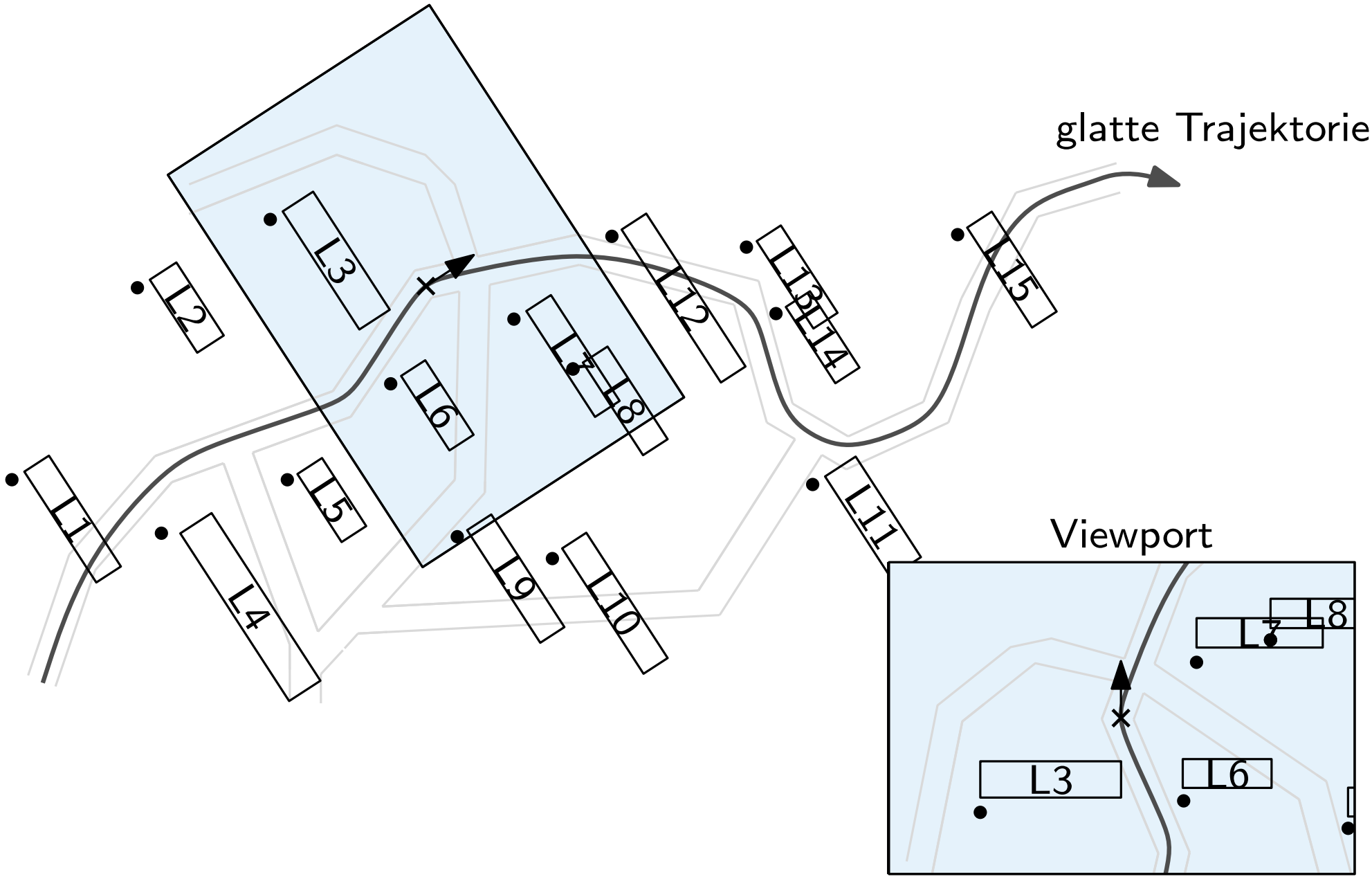
Trajectory-Based Dynamic Map Labeling



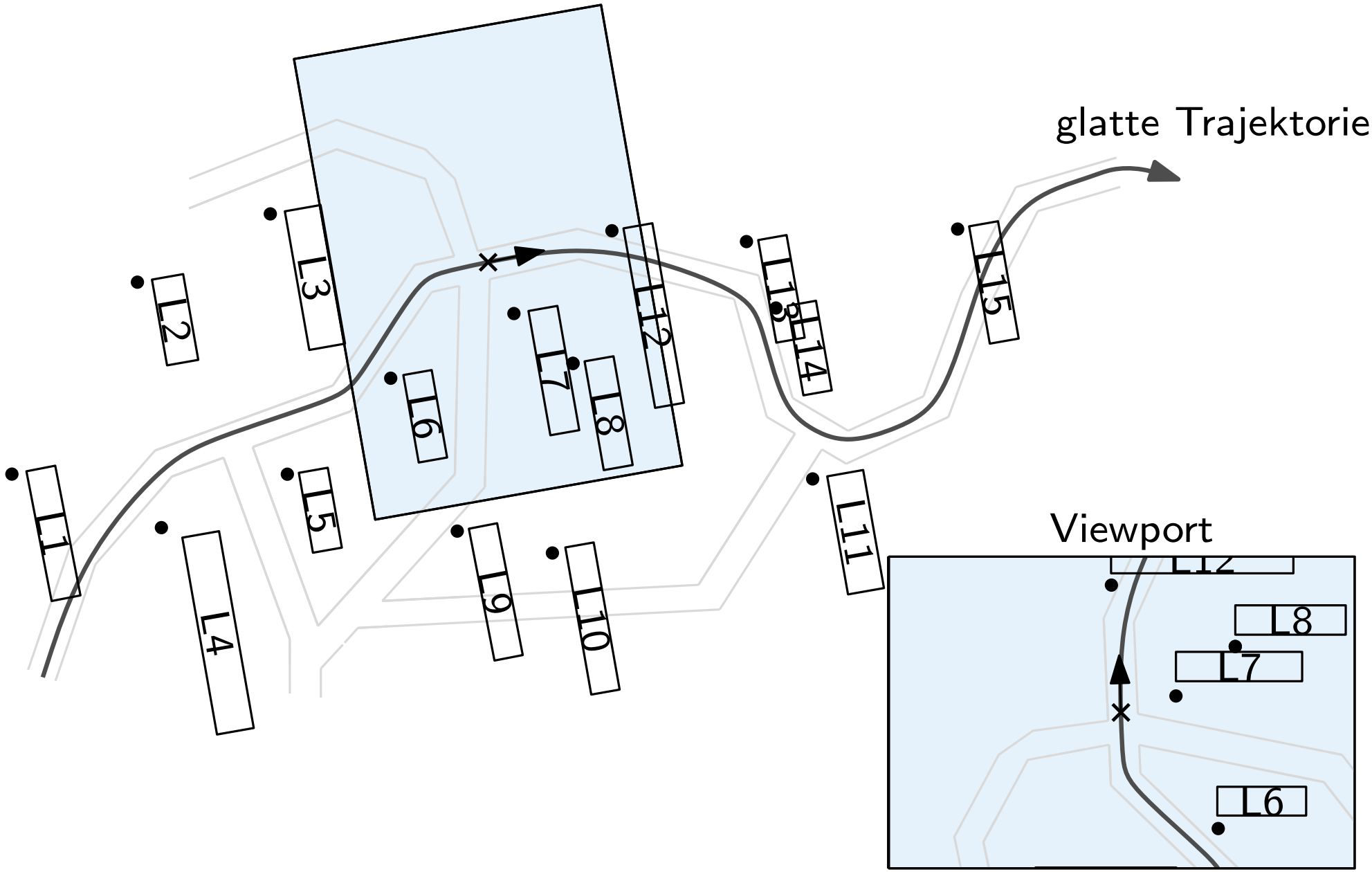
Trajectory-Based Dynamic Map Labeling



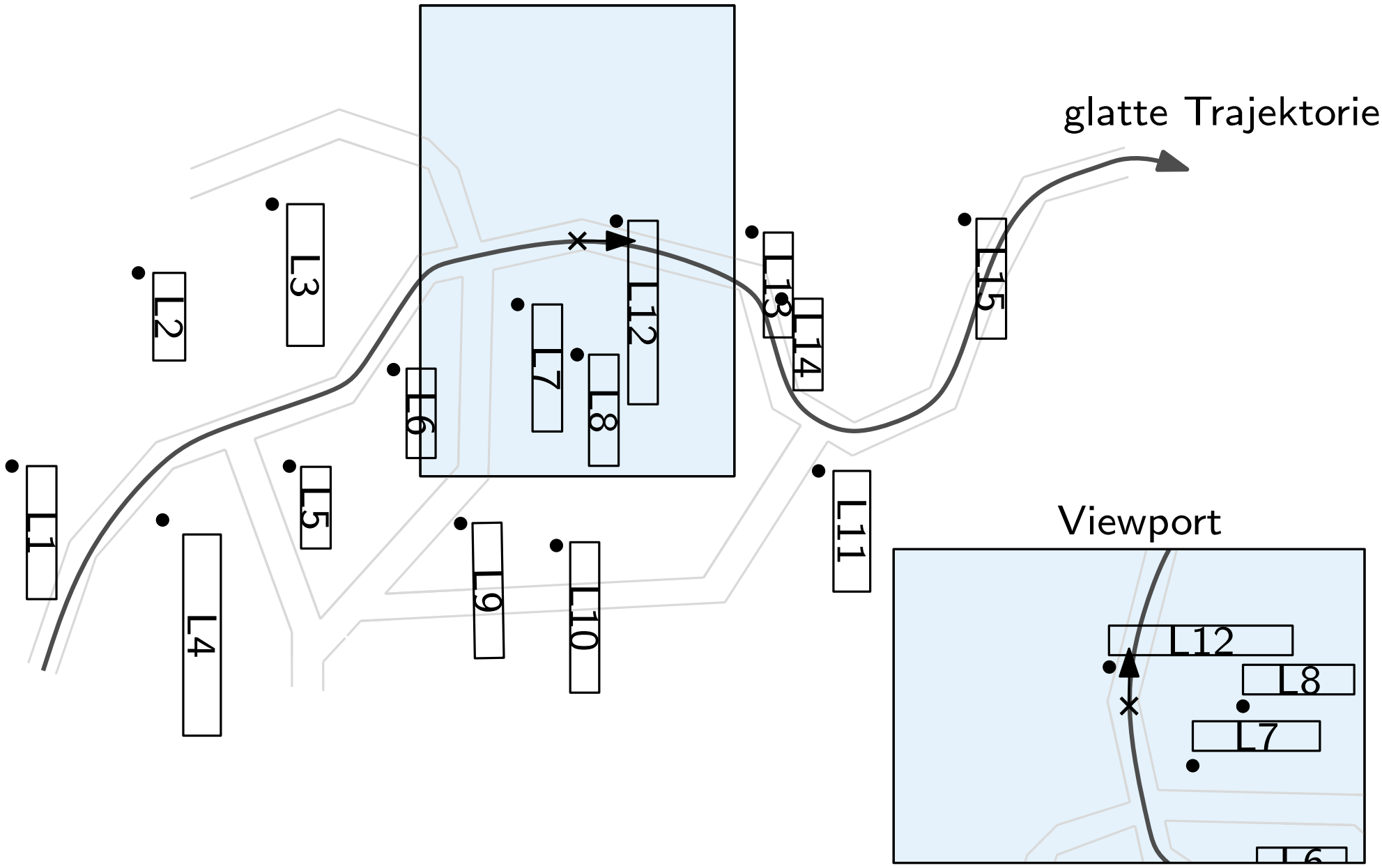
Trajectory-Based Dynamic Map Labeling



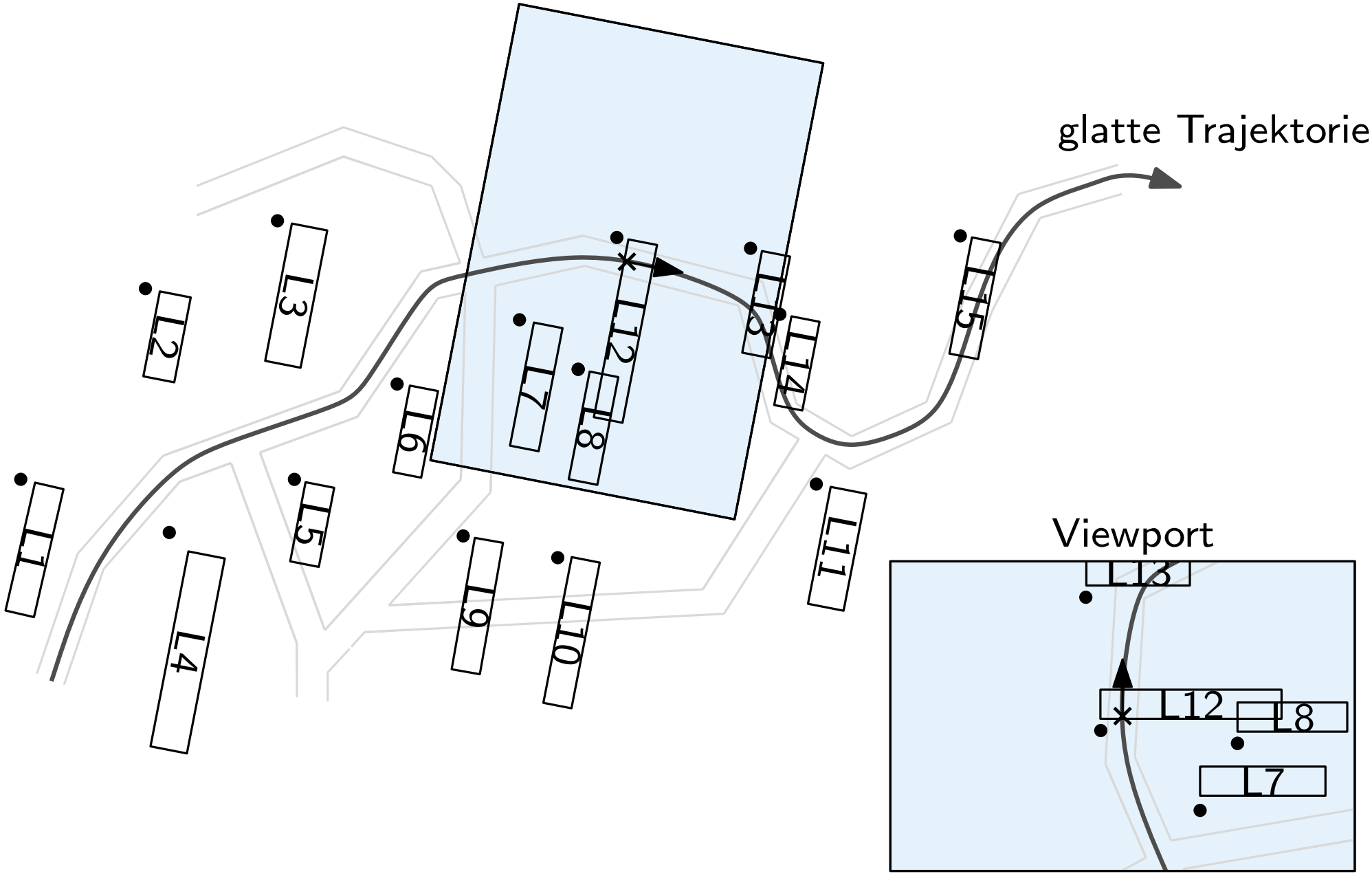
Trajectory-Based Dynamic Map Labeling



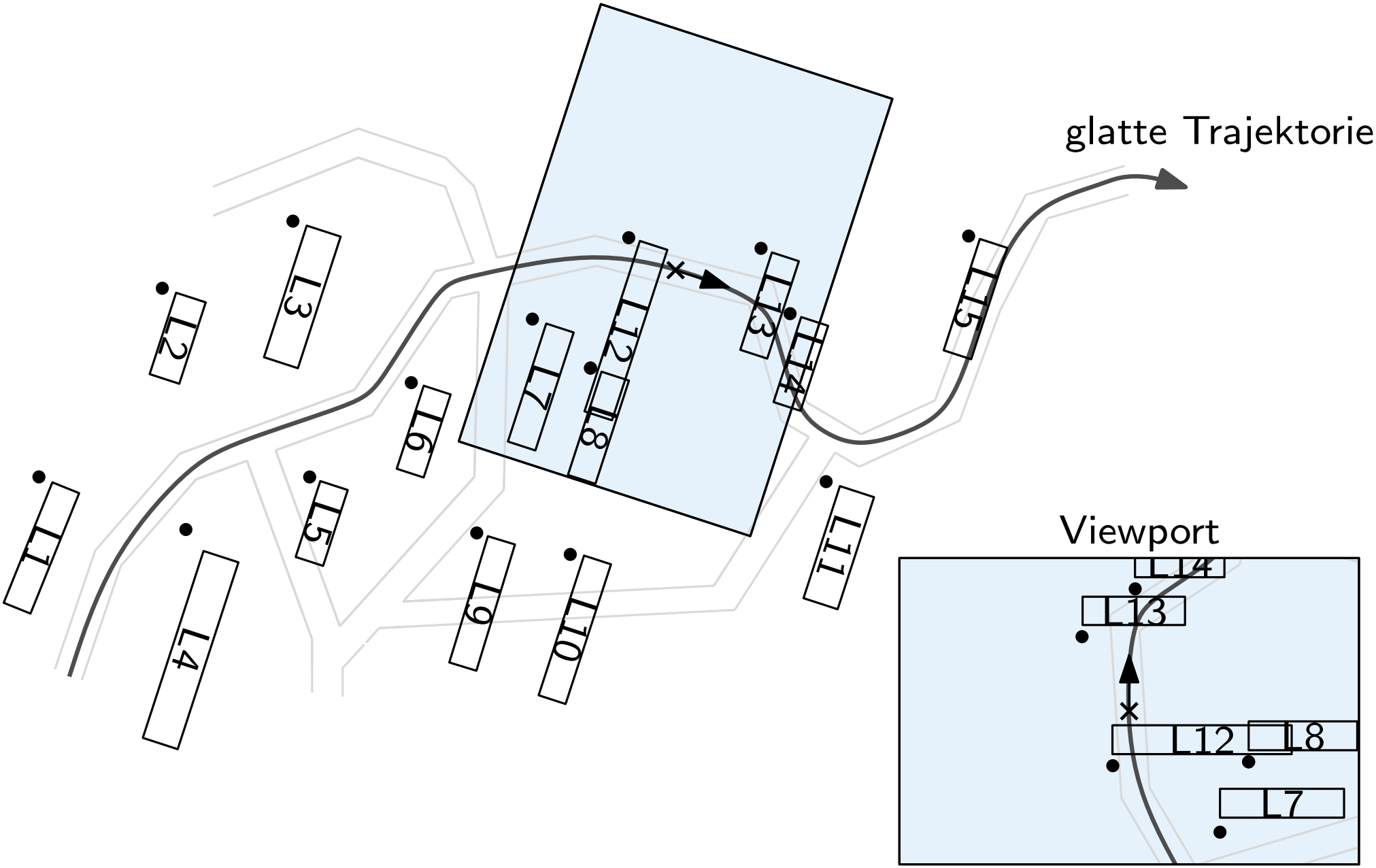
Trajectory-Based Dynamic Map Labeling



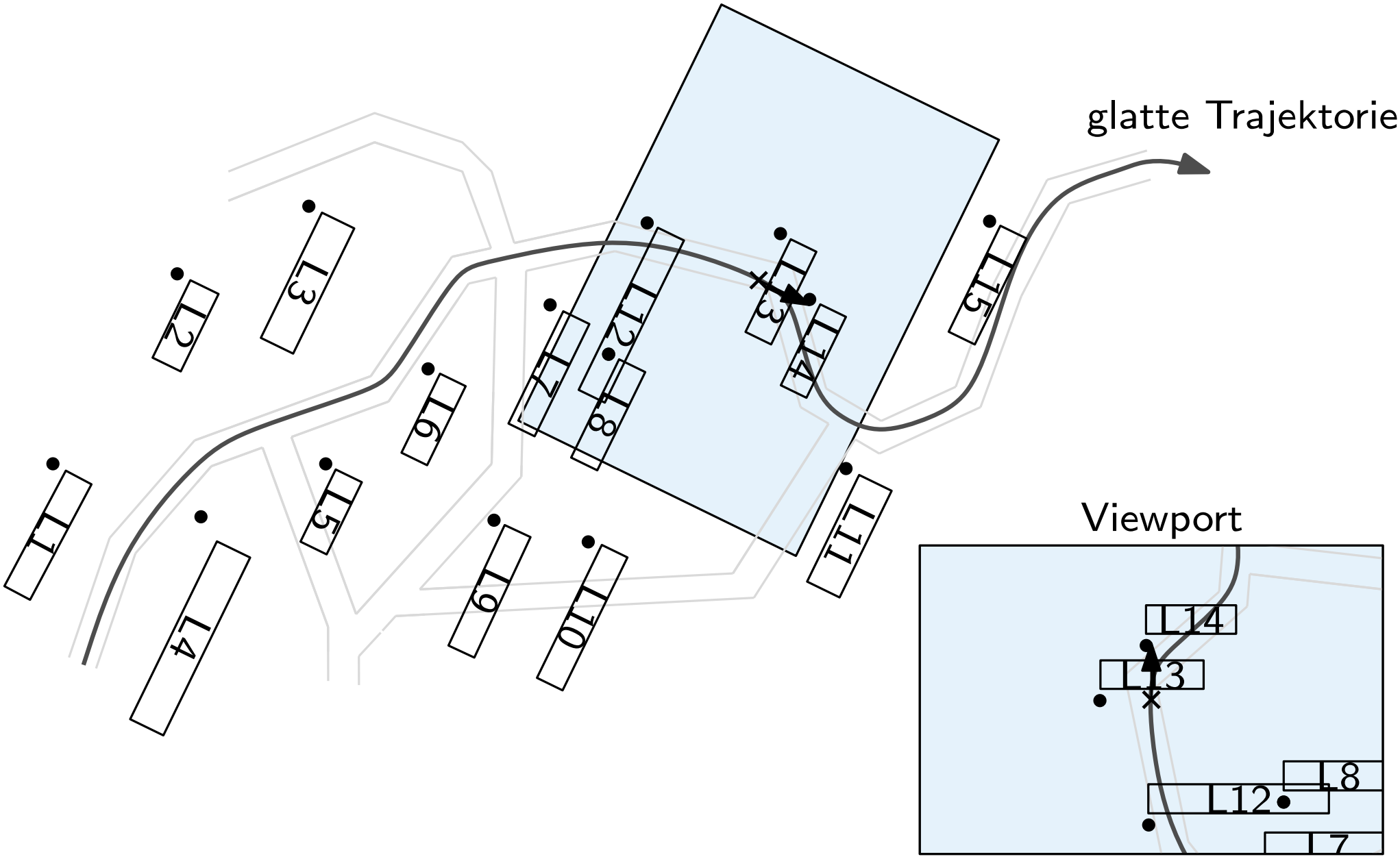
Trajectory-Based Dynamic Map Labeling



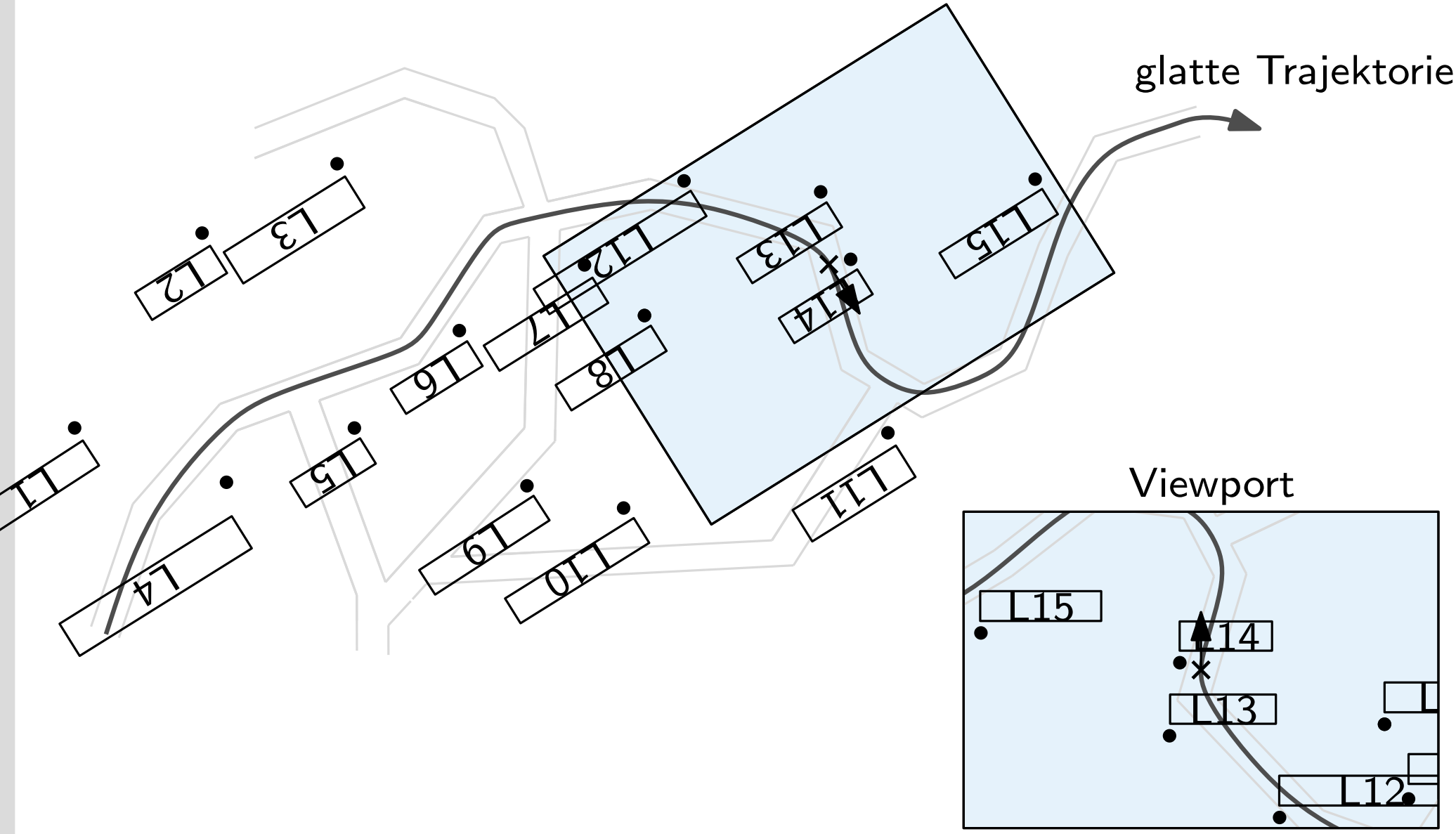
Trajectory-Based Dynamic Map Labeling



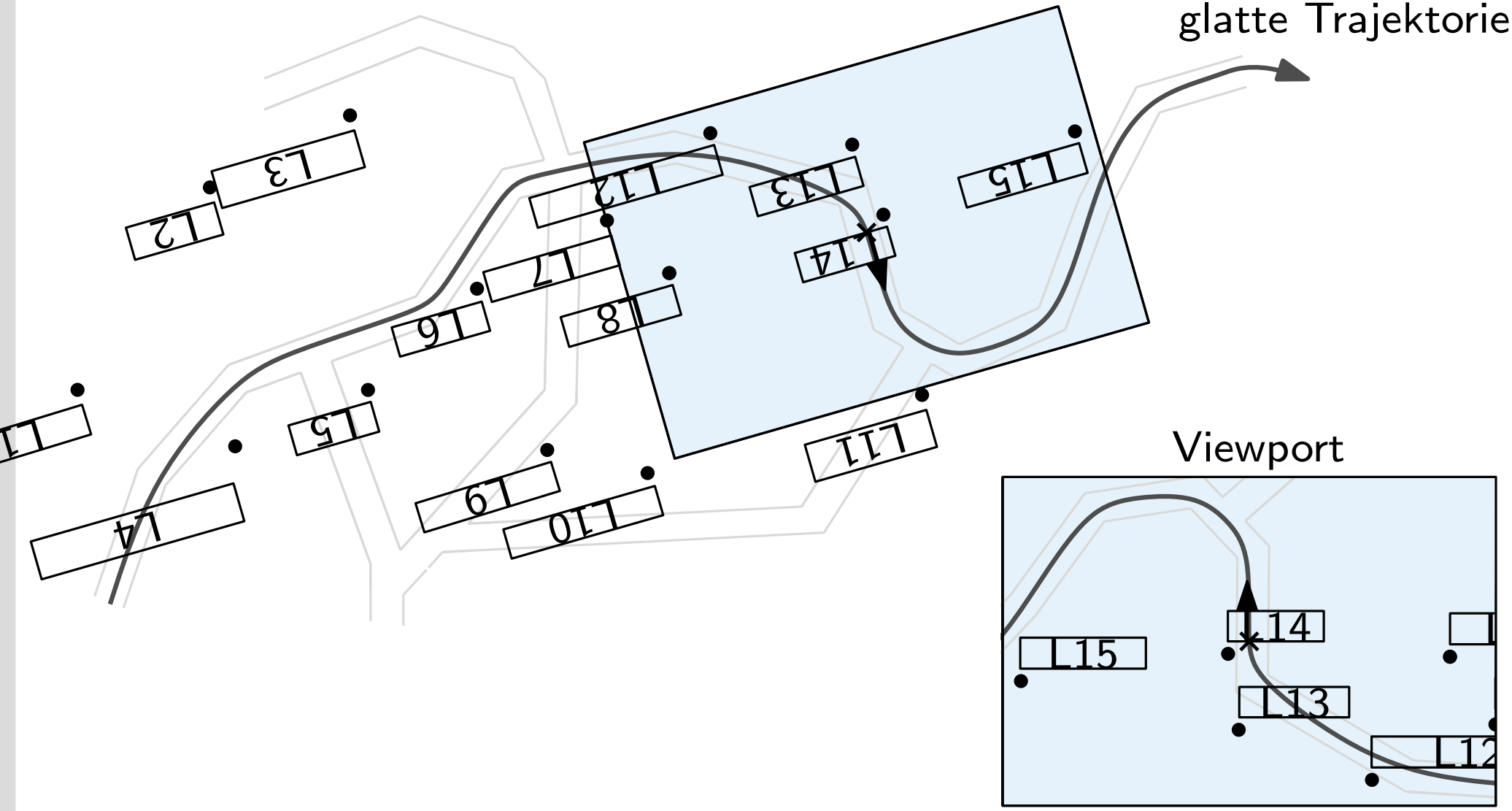
Trajectory-Based Dynamic Map Labeling



Trajectory-Based Dynamic Map Labeling



Trajectory-Based Dynamic Map Labeling



Trajectory-Based Dynamic Map Labeling

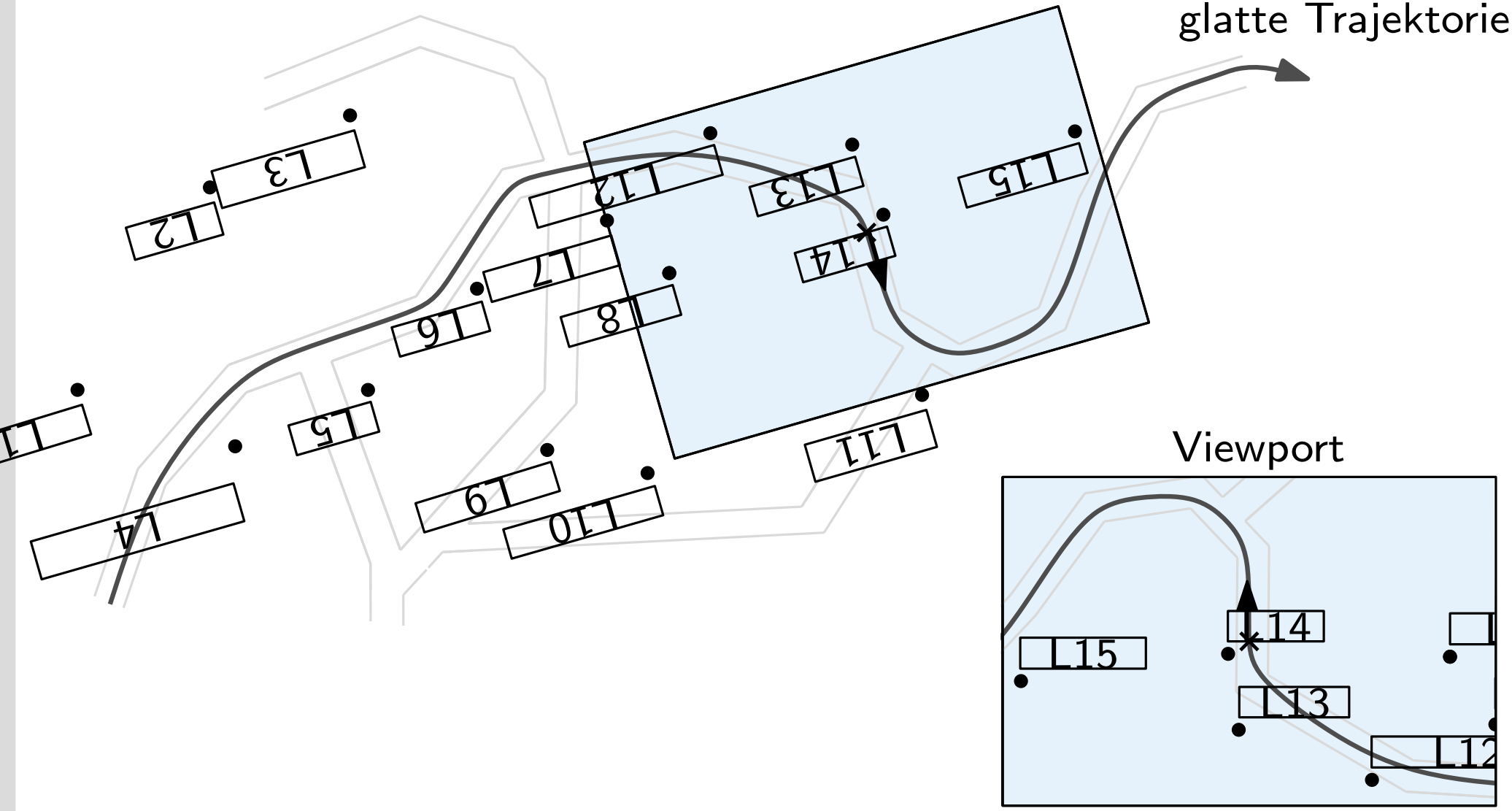
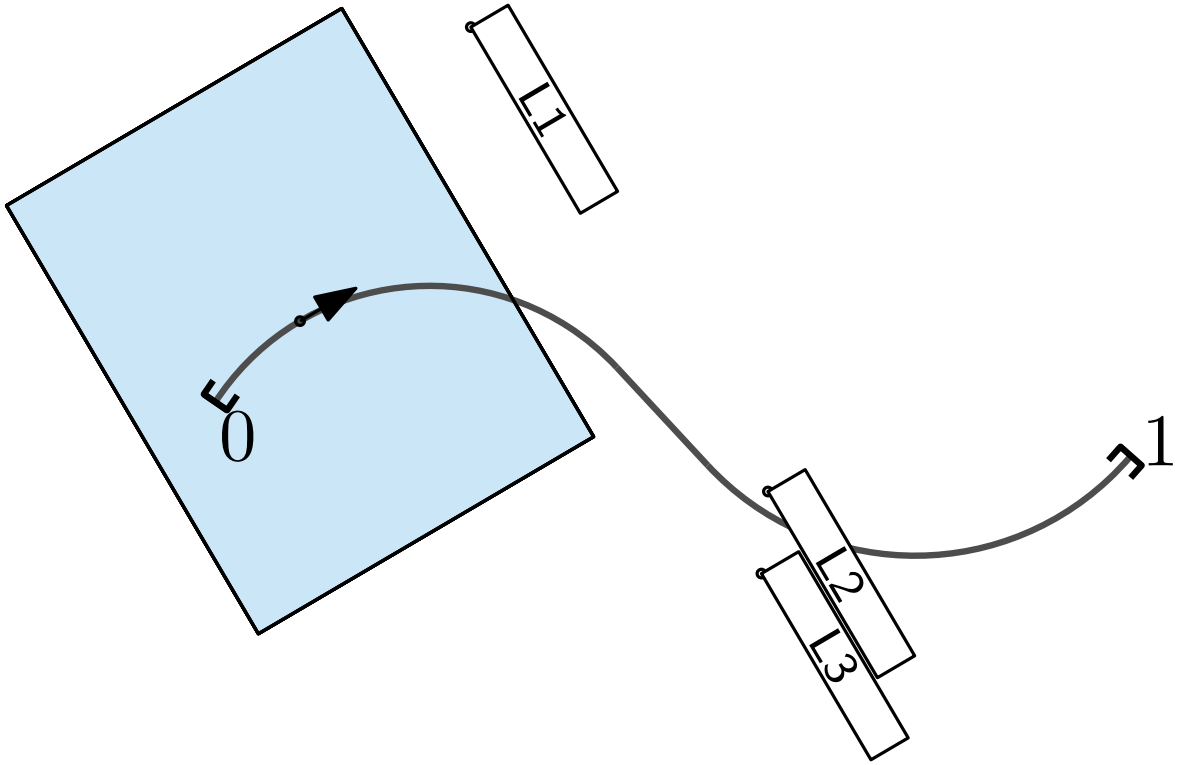


Abbildung auf Intervalle

Label ist **präsent** zum Zeitpunkt t falls es den Viewport zum Zeitpunkt t schneidet.



Zeit:

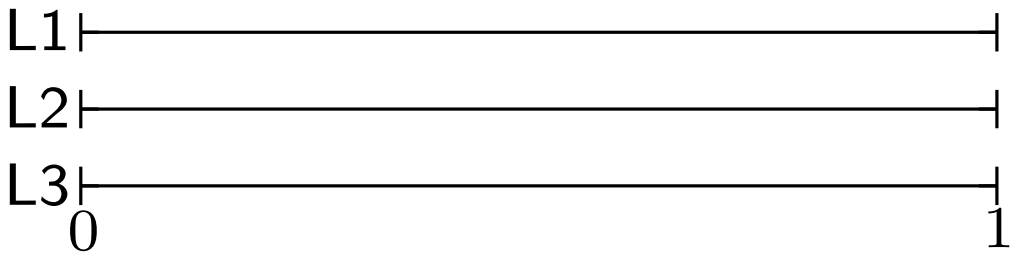
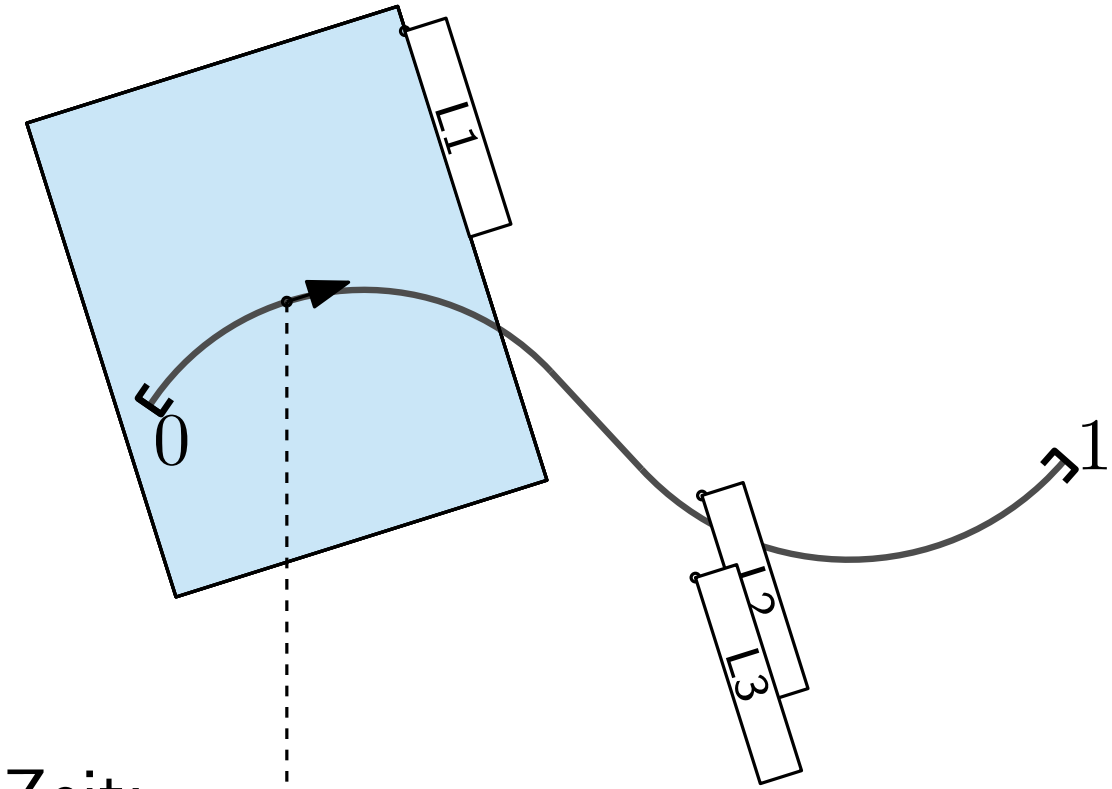


Abbildung auf Intervalle

Label ist **präsent** zum Zeitpunkt t falls es den Viewport zum Zeitpunkt t schneidet.



Zeit:

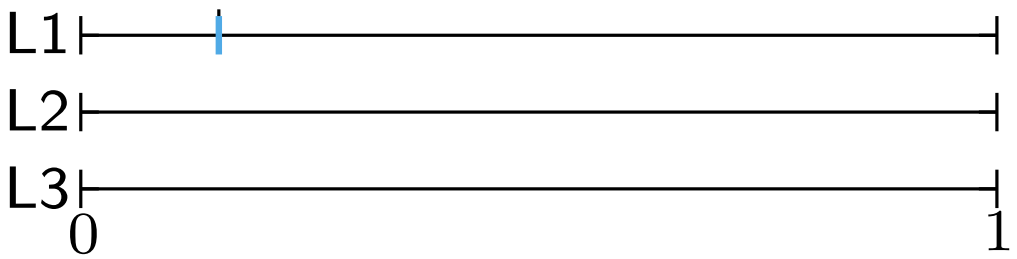
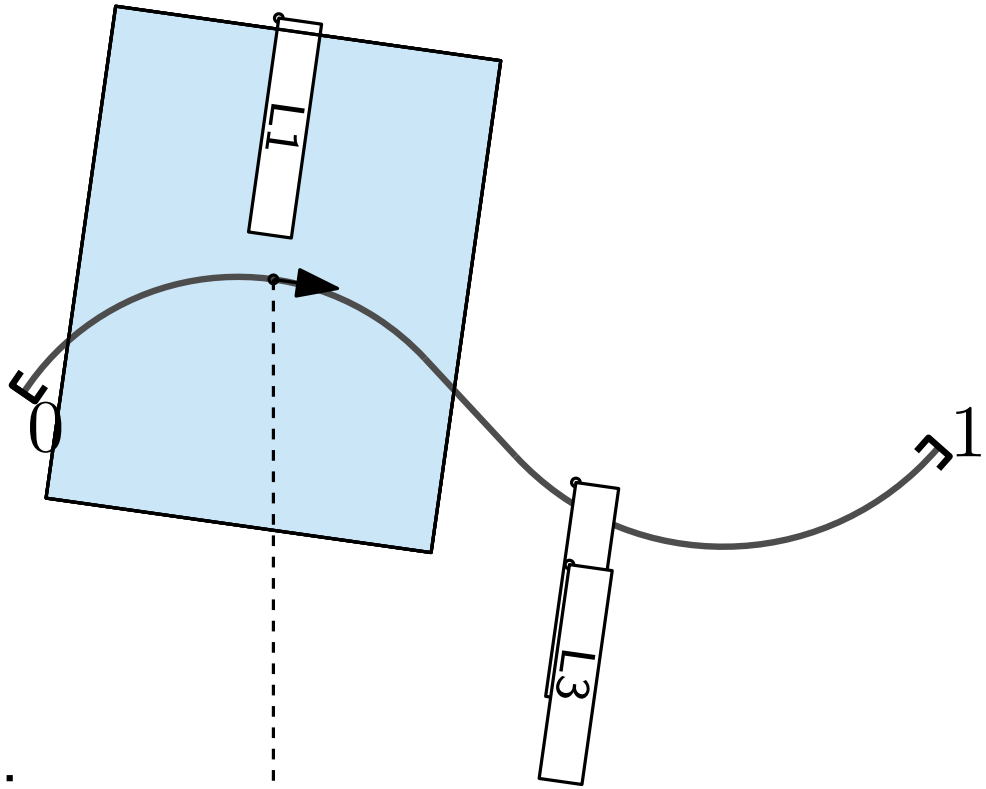


Abbildung auf Intervalle

Label ist **präsent** zum Zeitpunkt t falls es den Viewport zum Zeitpunkt t schneidet.



Zeit:

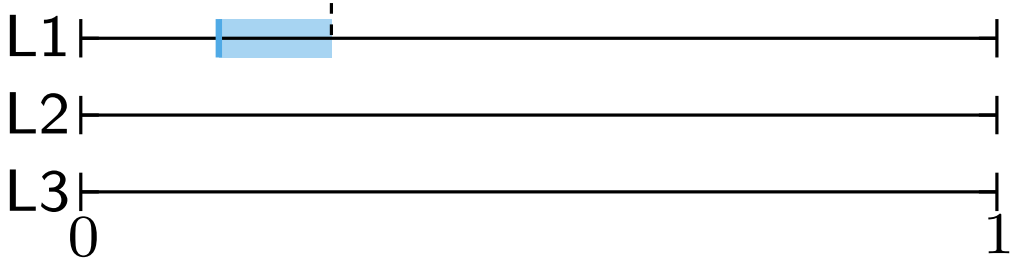
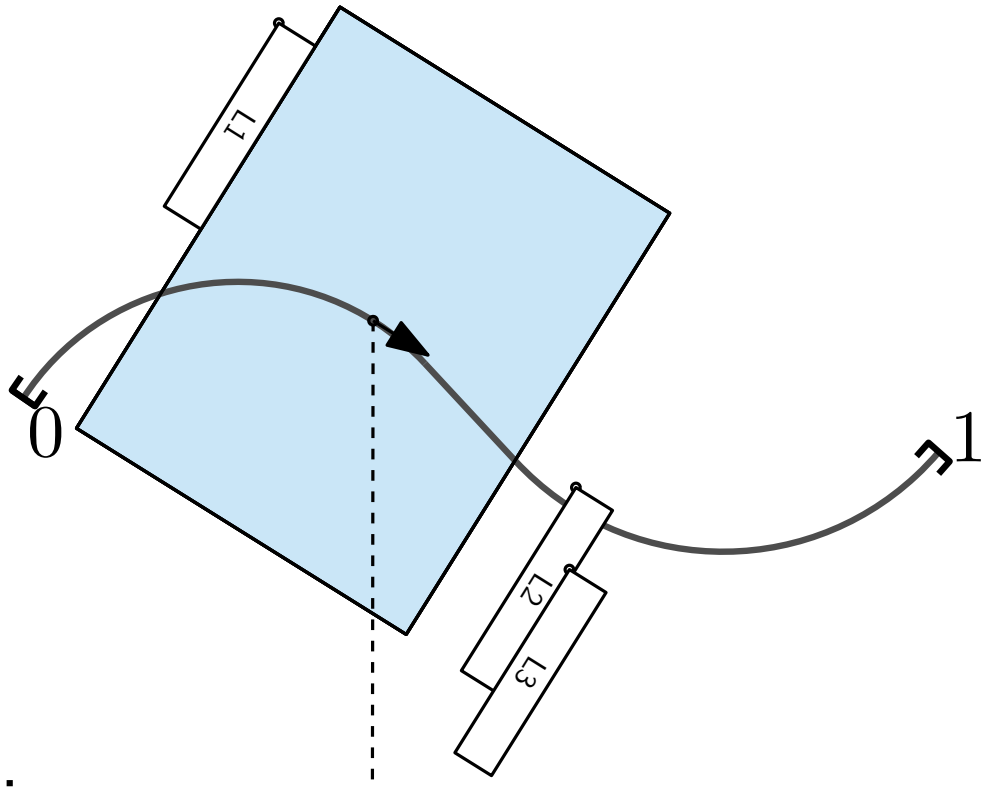


Abbildung auf Intervalle

Label ist **präsent** zum Zeitpunkt t falls es den Viewport zum Zeitpunkt t schneidet.



Zeit:

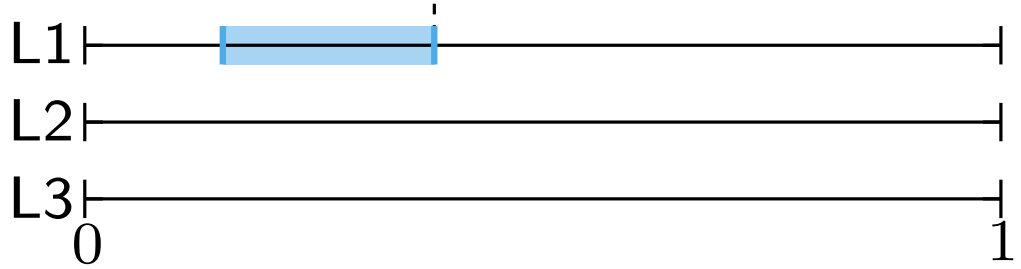
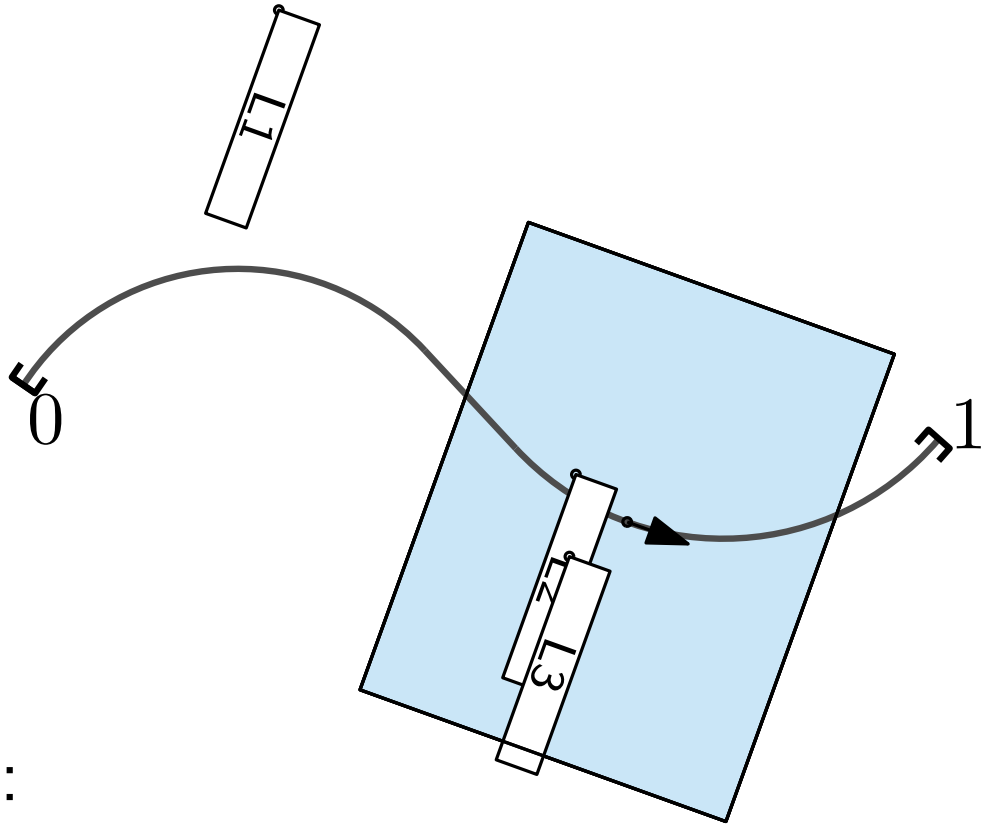
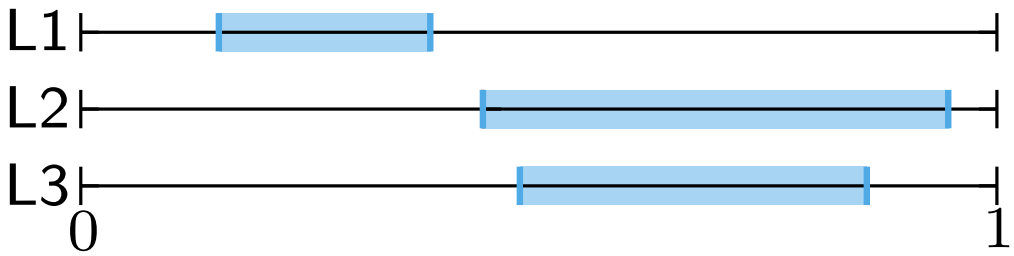


Abbildung auf Intervalle

Label ist **präsent** zum Zeitpunkt t falls es den Viewport zum Zeitpunkt t schneidet.



Zeit:



Menge von *Präsenzintervallen*

Abbildung auf Intervalle

Zwei Labels stehen in einem **Konflikt** zum Zeitpunkt t, falls sie sich zum Zeitpunkt t schneiden.

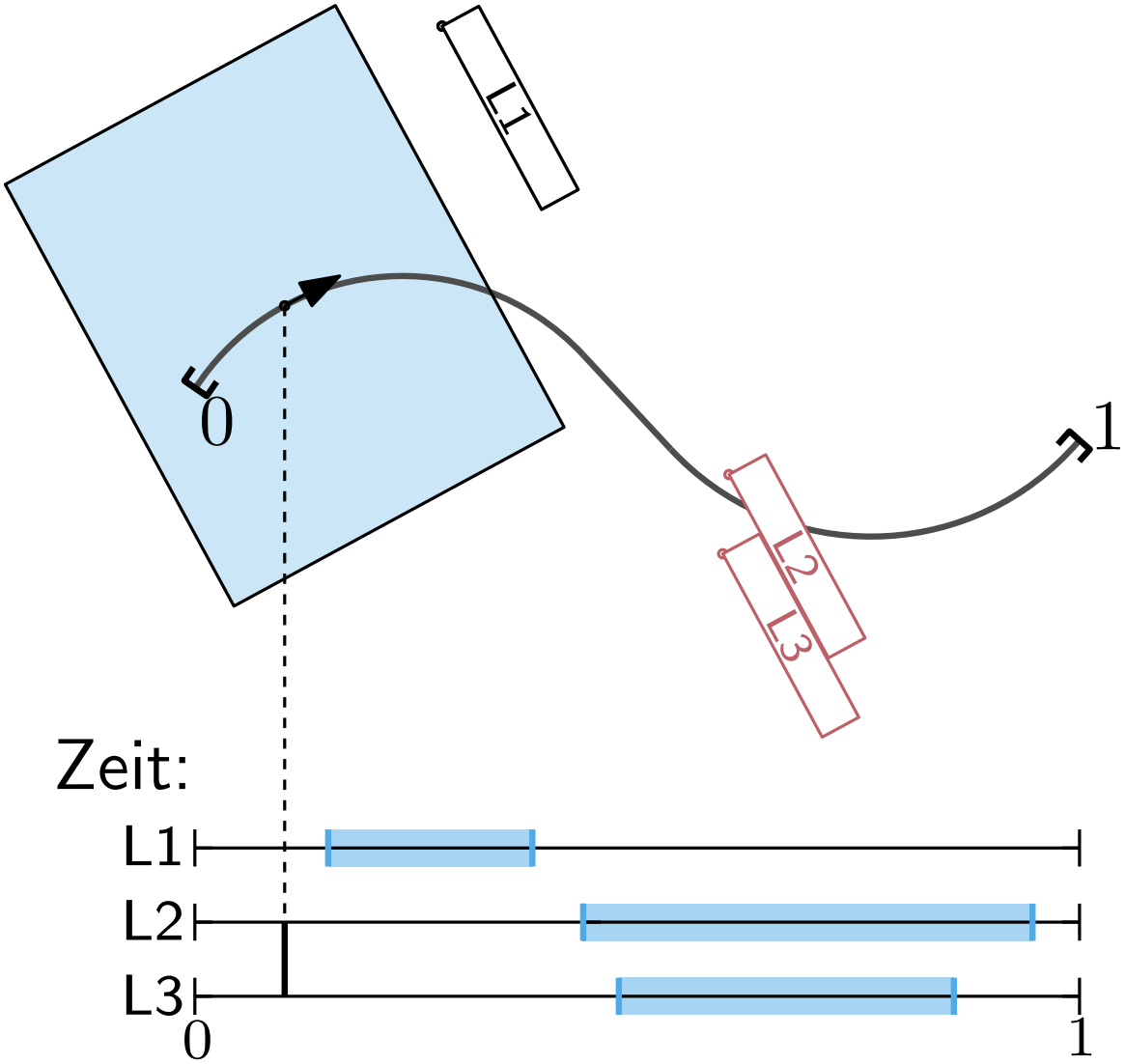
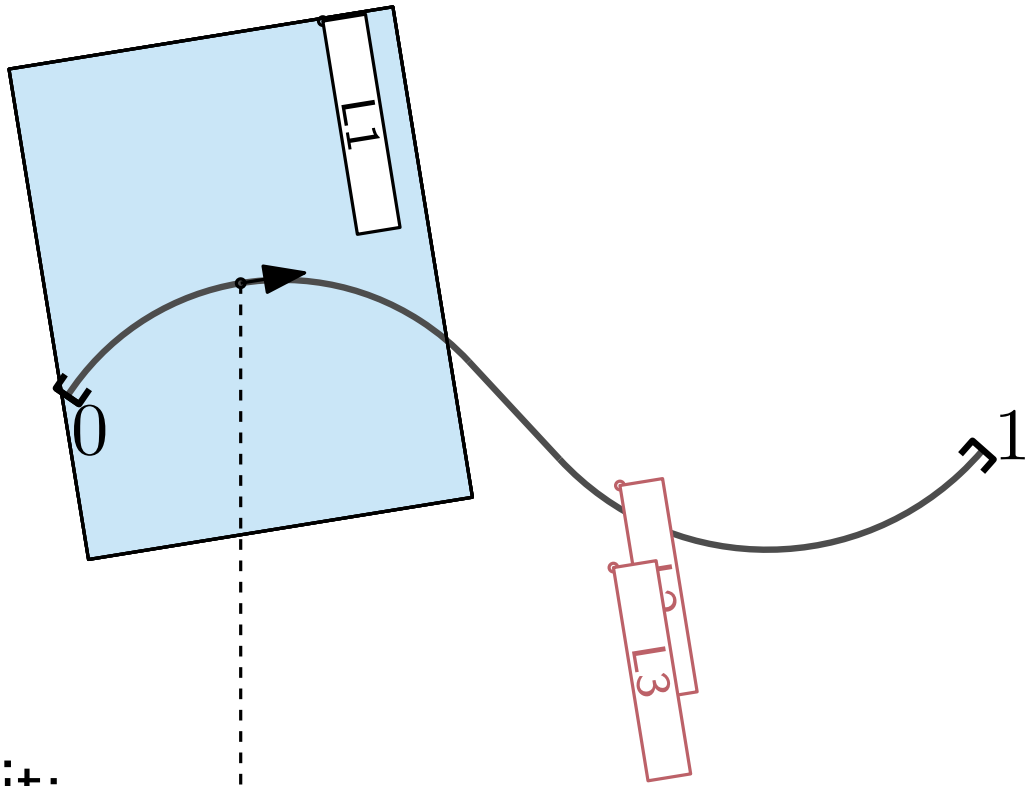


Abbildung auf Intervalle

Zwei Labels stehen in einem **Konflikt** zum Zeitpunkt t, falls sie sich zum Zeitpunkt t schneiden.



Zeit:

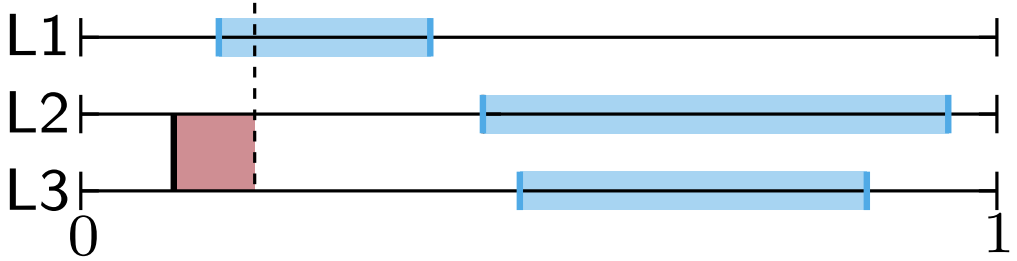
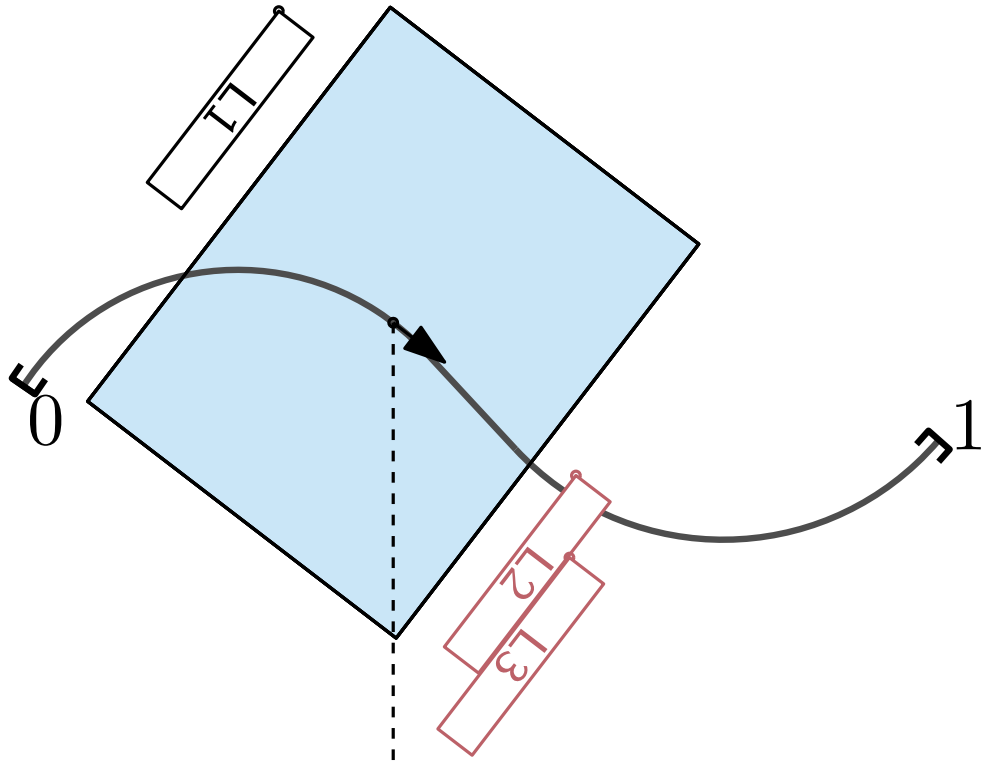


Abbildung auf Intervalle

Zwei Labels stehen in einem **Konflikt** zum Zeitpunkt t, falls sie sich zum Zeitpunkt t schneiden.



Zeit:

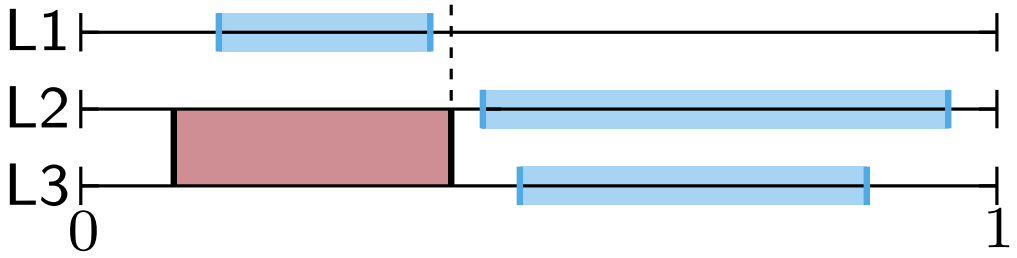
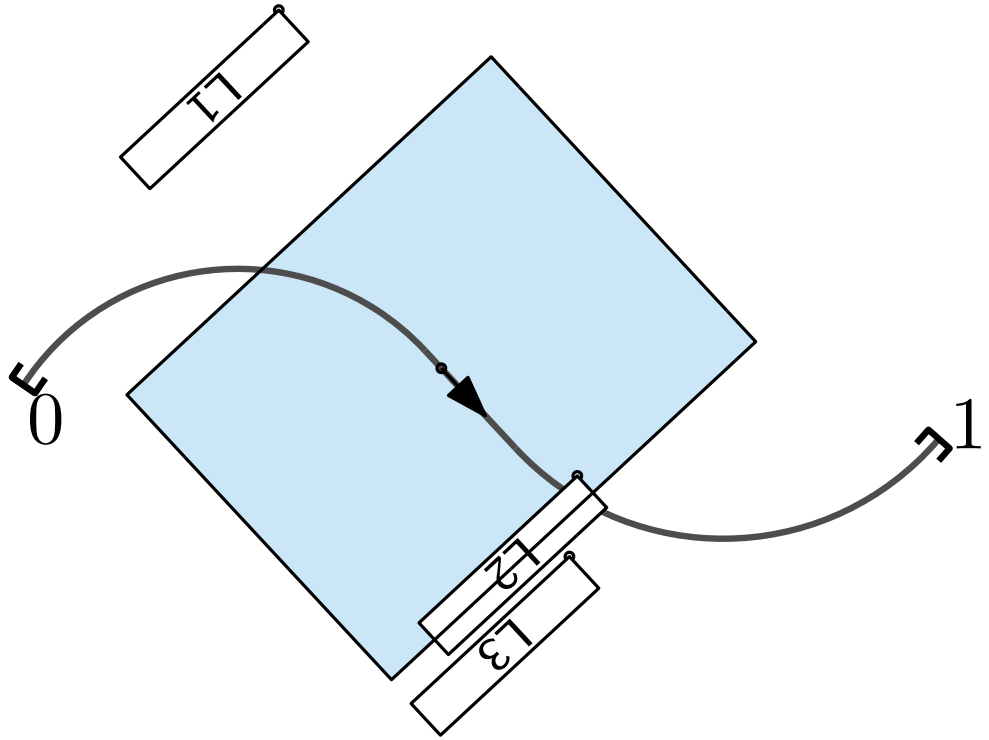


Abbildung auf Intervalle

Zwei Labels stehen in einem **Konflikt** zum Zeitpunkt t, falls sie sich zum Zeitpunkt t schneiden.



Zeit:

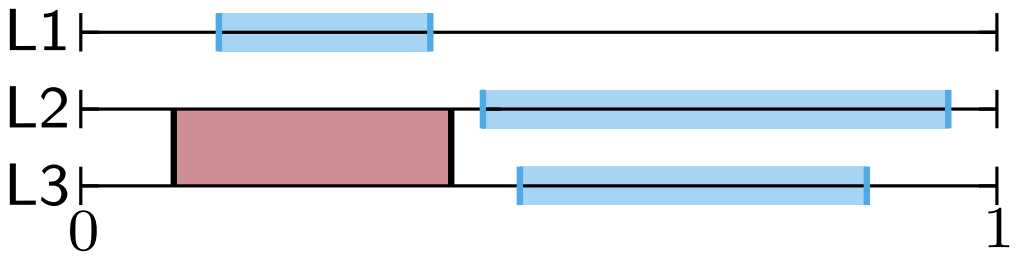
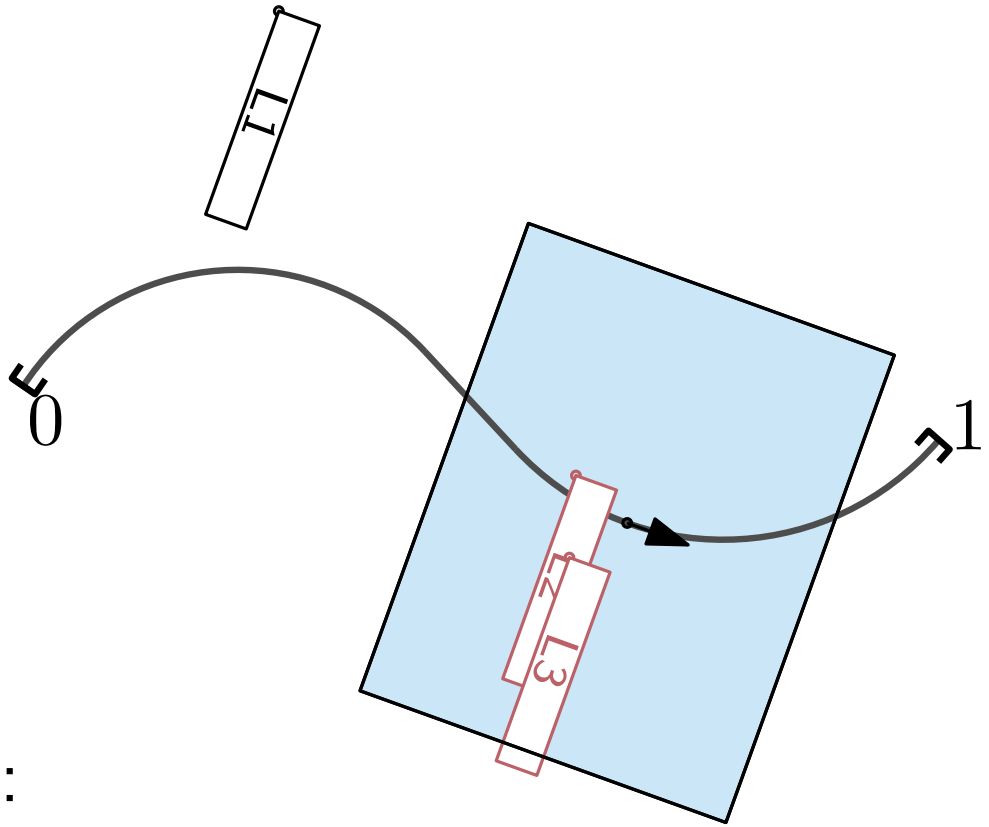
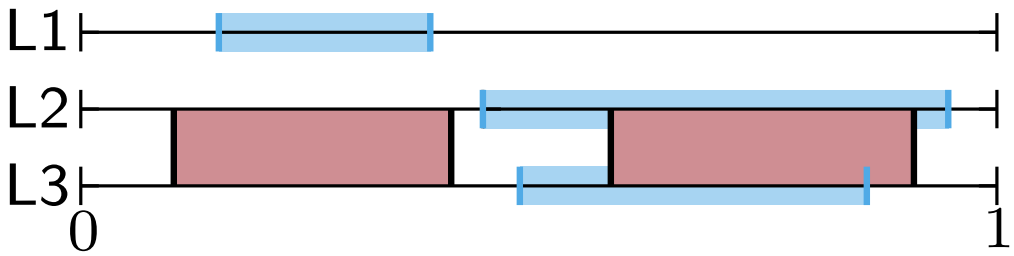


Abbildung auf Intervalle

Zwei Labels stehen in einem **Konflikt** zum Zeitpunkt t, falls sie sich zum Zeitpunkt t schneiden.



Zeit:



Menge von **Konfliktintervallen**

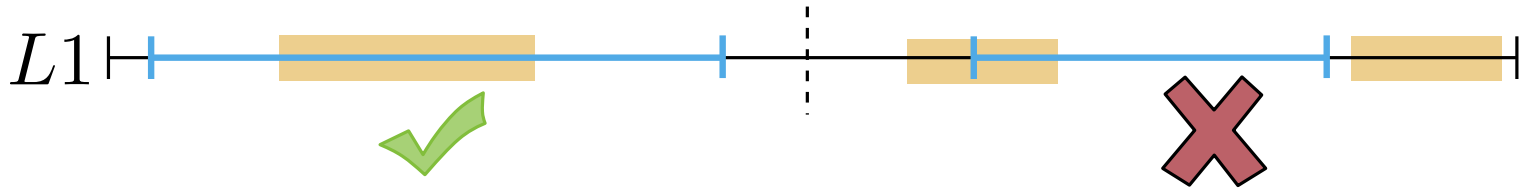
Aktivität von Labels

Label ist **aktiv** zum Zeitpunkt t falls es zum Zeitpunkt t angezeigt wird.

Label ist präsent. Label ist aktiv. Labels sind in Konflikt.

Aktivität von Labels

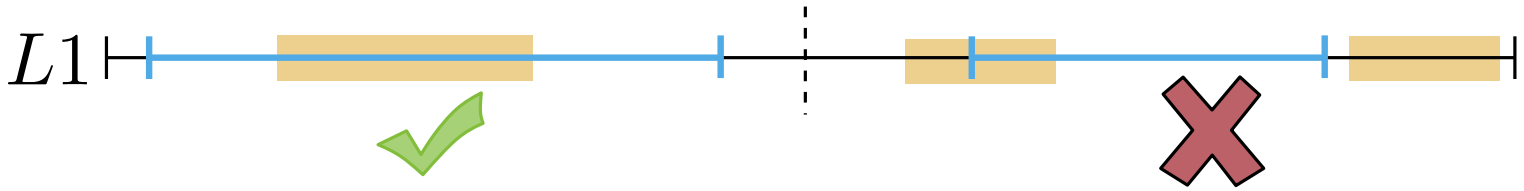
Label ist **aktiv** zum Zeitpunkt t falls es zum Zeitpunkt t angezeigt wird.
Nur aktiv wenn auch präsent.



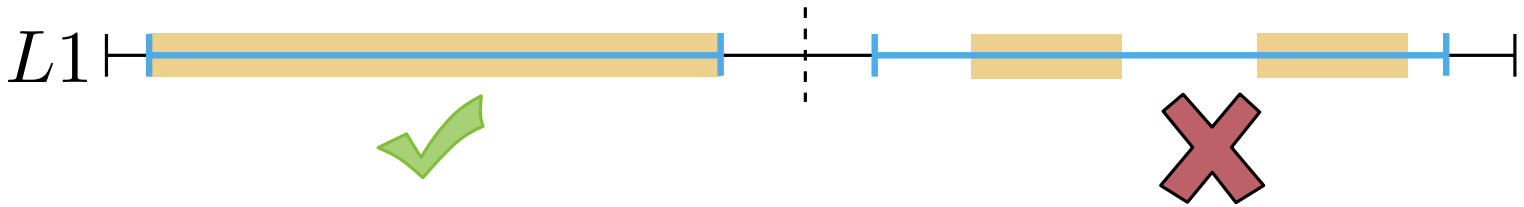
Label ist präsent. Label ist aktiv. Labels sind in Konflikt.

Aktivität von Labels

Label ist **aktiv** zum Zeitpunkt t falls es zum Zeitpunkt t angezeigt wird.
Nur aktiv wenn auch präsent.



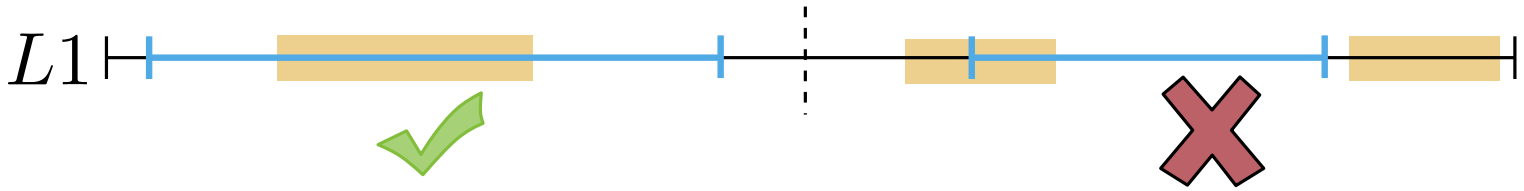
Entweder aktiv für **gesamtes** Präsenzintervall oder **gar nicht**.



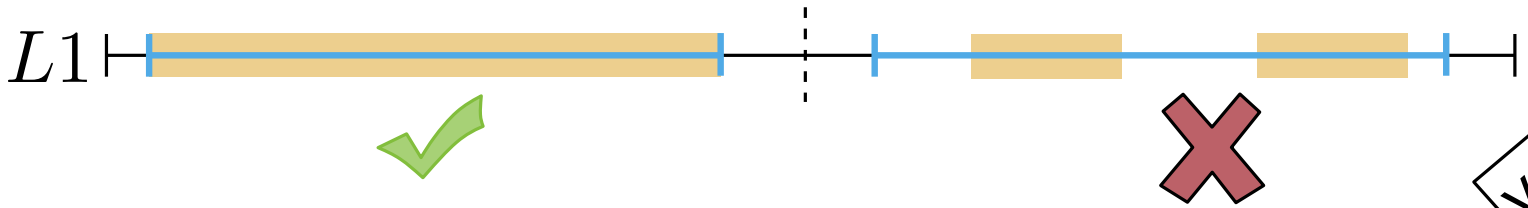
Label ist präsent. Label ist aktiv. Labels sind in Konflikt.

Aktivität von Labels

Label ist **aktiv** zum Zeitpunkt t falls es zum Zeitpunkt t angezeigt wird.
Nur aktiv wenn auch präsent.



Entweder aktiv für **gesamtes** Präsenzintervall oder **gar nicht**.

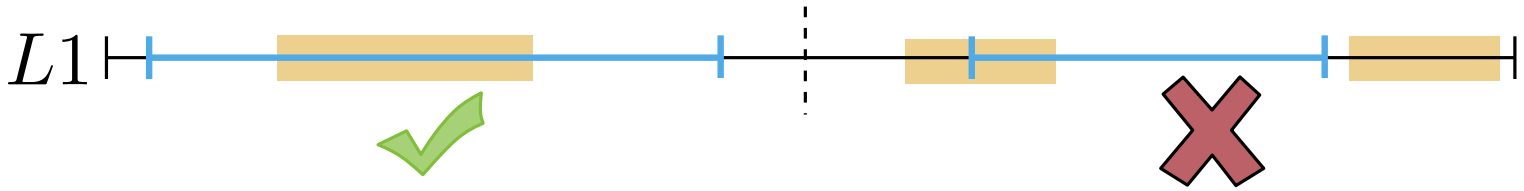


vermeidet Flackern

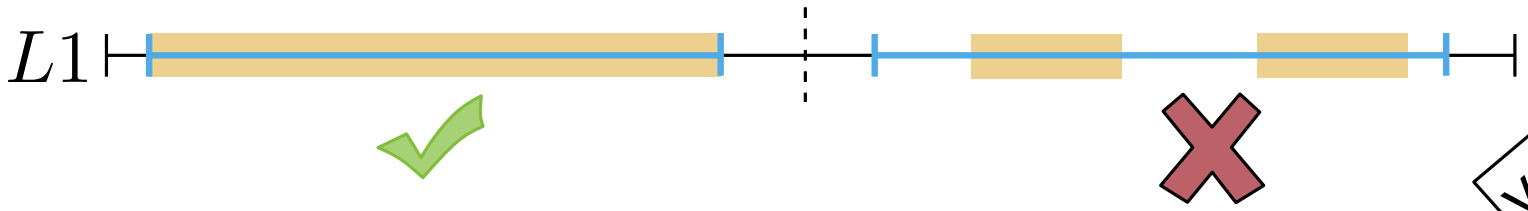
Label ist präsent. Label ist aktiv. Labels sind in Konflikt.

Aktivität von Labels

Label ist **aktiv** zum Zeitpunkt t falls es zum Zeitpunkt t angezeigt wird.
Nur aktiv wenn auch präsent.

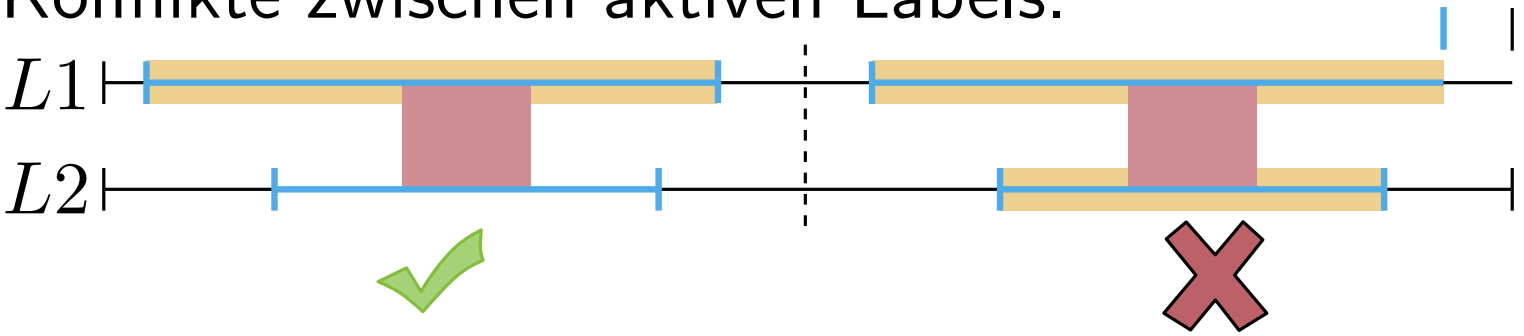


Entweder aktiv für **gesamtes** Präsenzintervall oder **gar nicht**.



vermeidet Flackern

Keine Konflikte zwischen aktiven Labels.

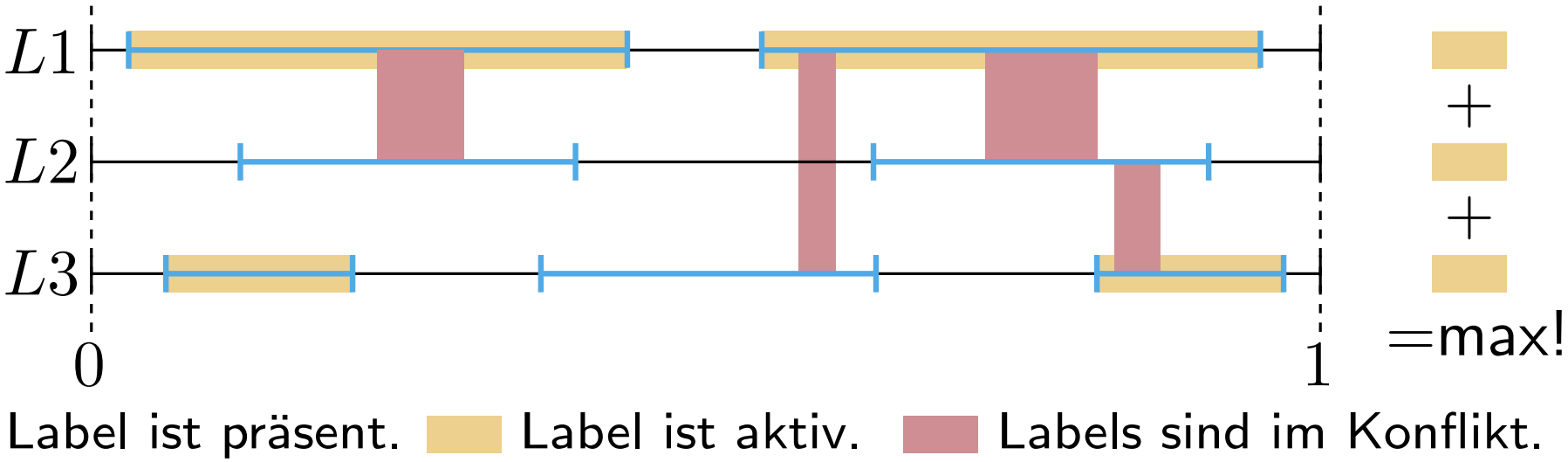


Label ist präsent. Label ist aktiv. Labels sind in Konflikt.

Problemdefinition

Problem GeneralMaxTotal:
Gegeben: Menge I an Präsenzintervallen, Konflikte
Gesucht: $J \subseteq I$ sodass $\sum_{j \in J} \text{length}(j)$ maximal und J konfliktfrei ist.

Menge J an Intervallen ist **konfliktfrei** falls keine zwei Intervalle in J im Konflikt stehen.



Problemdefinition

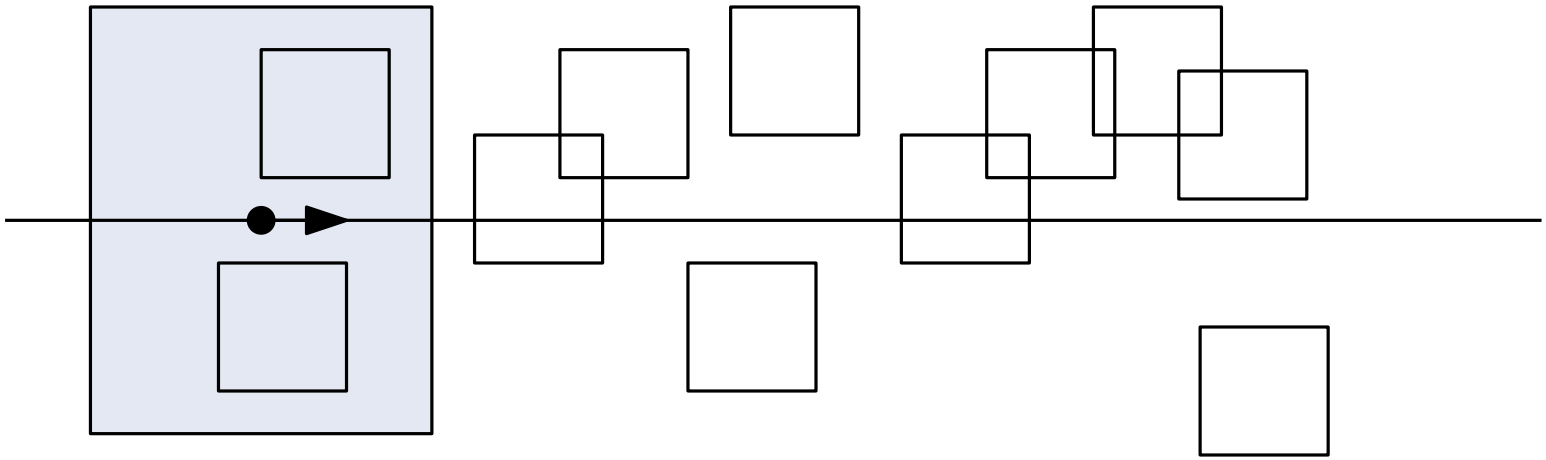
Problem GeneralMaxTotal:
Gegeben: Menge I an Präsenzintervallen, Konflikte
Gesucht: $J \subseteq I$ sodass $\sum_{j \in J} \text{length}(j)$ maximal und J konfliktfrei ist.

Menge J an Intervallen ist **konfliktfrei** falls keine zwei Intervalle in J im Konflikt stehen.

Theorem 5

GeneralMaxTotal ist NP-vollständig

Geradlinige Kamerafahrt

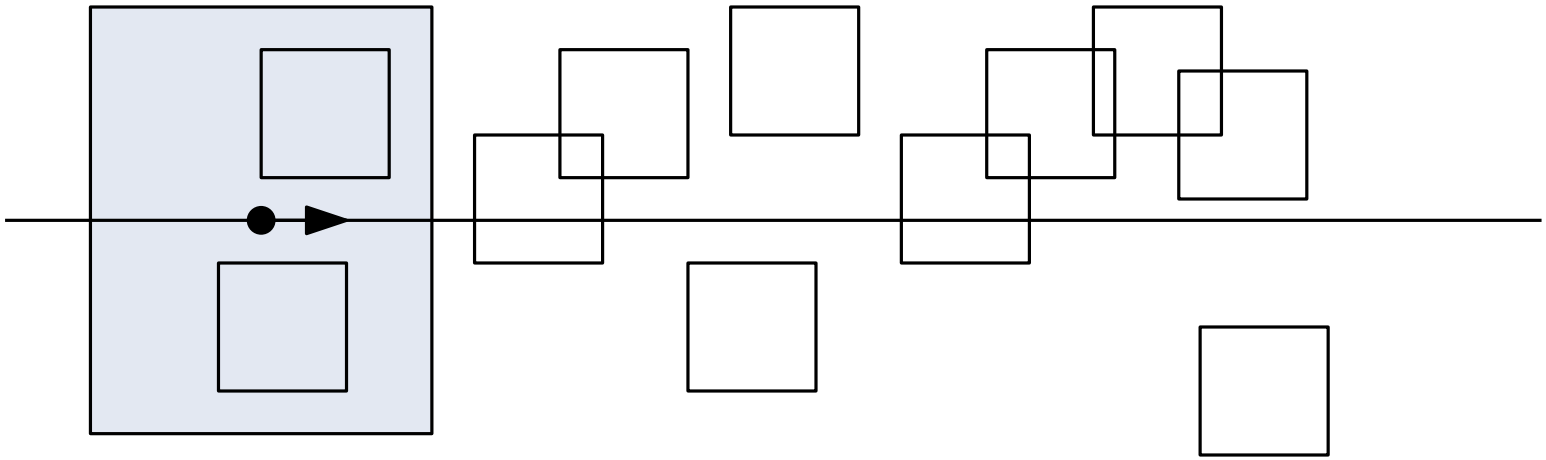


Annahme:

- Labels sind Quadrate (und in allgemeiner Lage).
- Trajektorie besteht aus Quadraten.

Gesucht: $\frac{1}{2}$-Approximation

Geradlinige Kamerafahrt



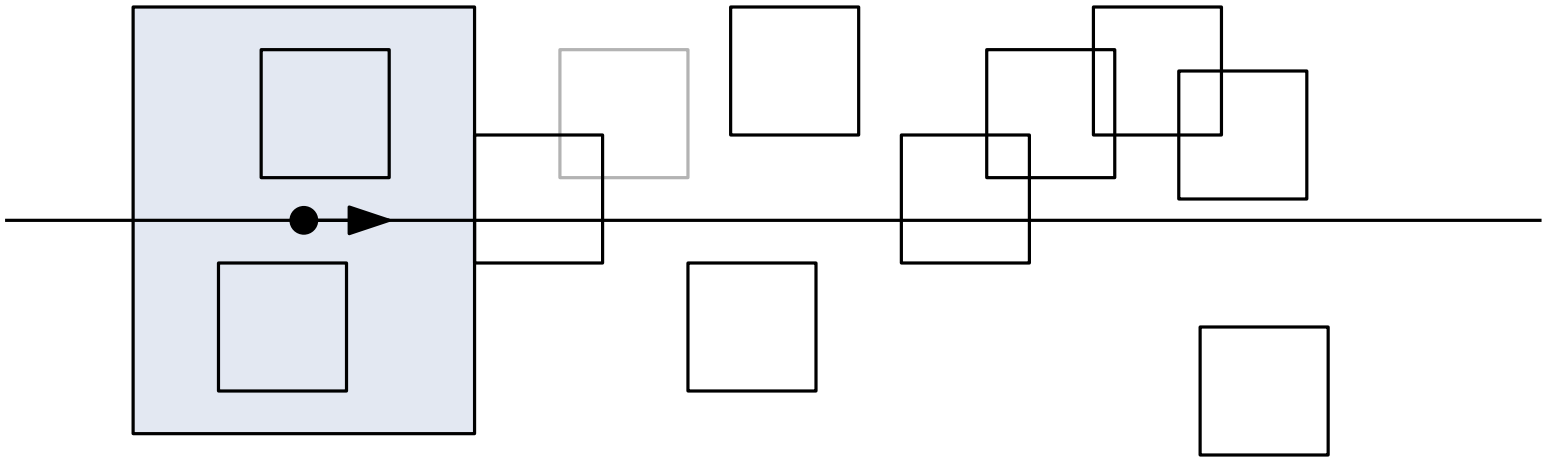
Annahme:

- Labels sind Quadrate (und in allgemeiner Lage).
- Trajektorie besteht aus Quadraten.

Greedy-Verfahren:

> Wenn zwei Labels im Konflikt stehen, dann schalte dasjenige von beiden aktiv, das zuerst in den Viewport eintritt und das andere inaktiv.

Geradlinige Kamerafahrt



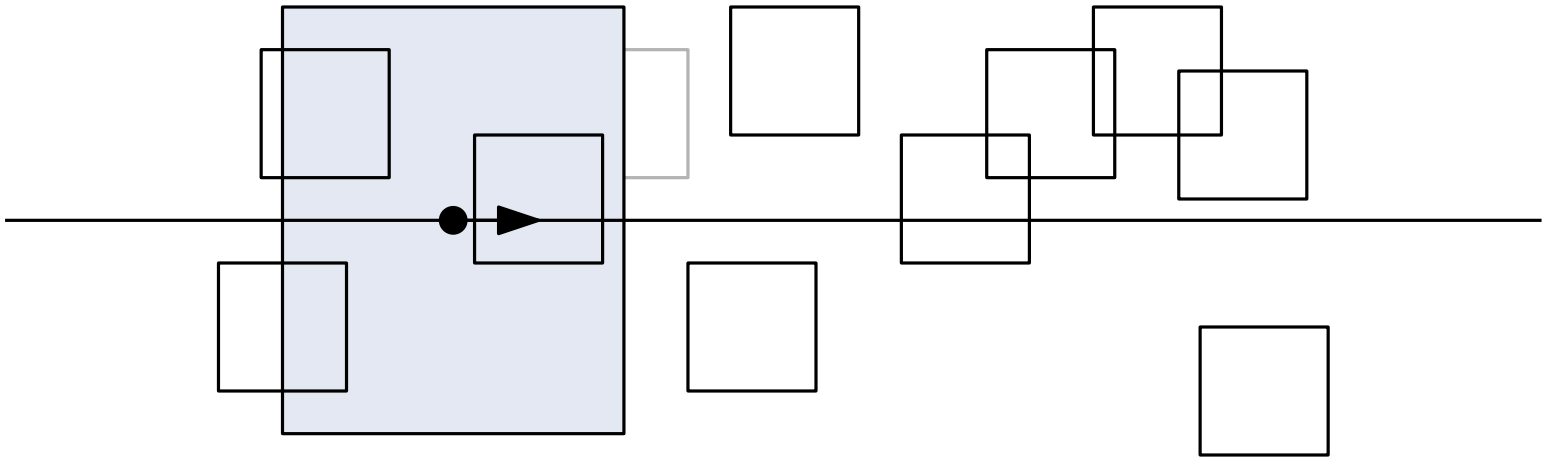
Annahme:

- Labels sind Quadrate (und in allgemeiner Lage).
- Trajektorie besteht aus Quadraten.

Greedy-Verfahren:

Wenn zwei Labels im Konflikt stehen, dann schalte dasjenige von beiden aktiv, das zuerst in den Viewport eintritt und das andere inaktiv.

Geradlinige Kamerafahrt



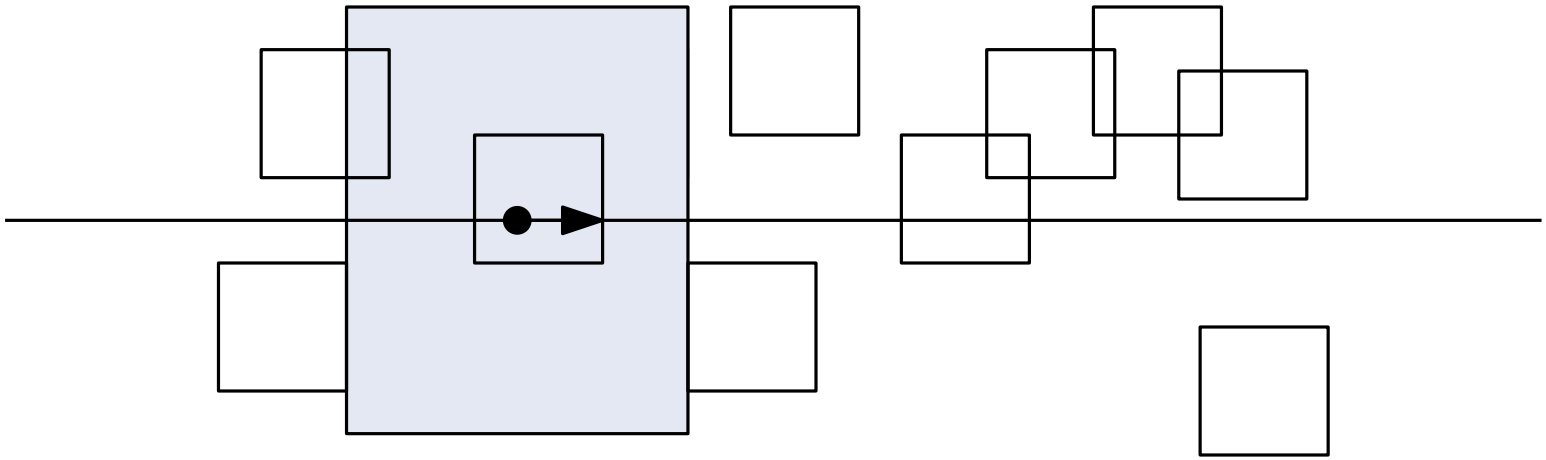
Annahme:

- Labels sind Quadrate (und in allgemeiner Lage).
- Trajektorie besteht aus Quadraten.

Greedy-Verfahren:

Wenn zwei Labels im Konflikt stehen, dann schalte dasjenige von beiden aktiv, das zuerst in den Viewport eintritt und das andere inaktiv.

Geradlinige Kamerafahrt



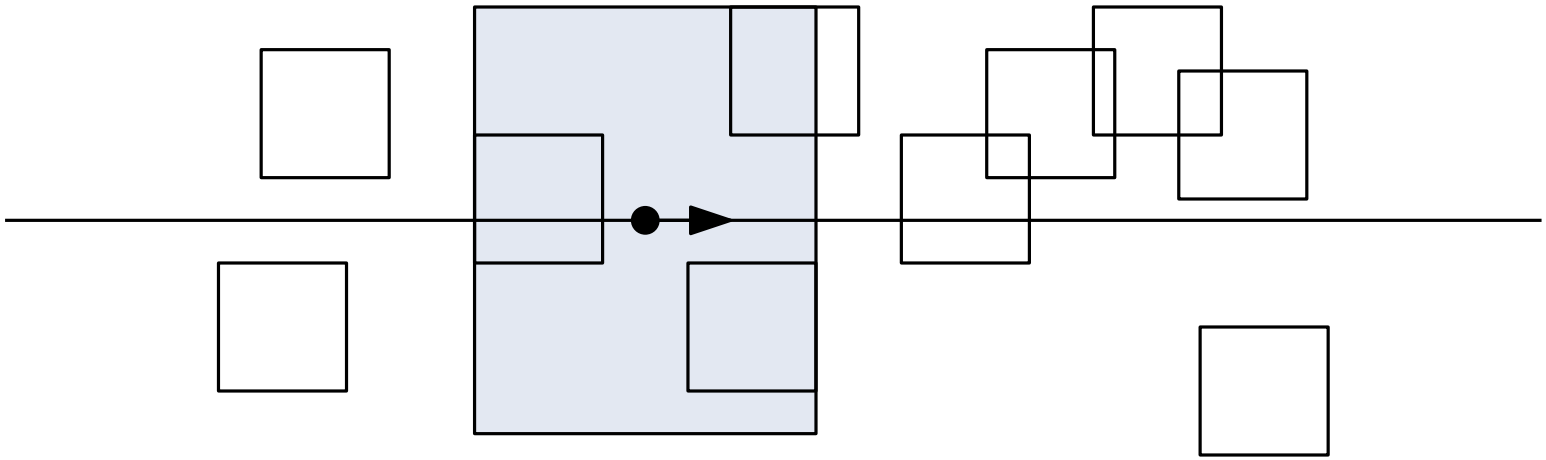
Annahme:

- Labels sind Quadrate (und in allgemeiner Lage).
- Trajektorie besteht aus Quadraten.

Greedy-Verfahren:

Wenn zwei Labels im Konflikt stehen, dann schalte dasjenige von beiden aktiv, das zuerst in den Viewport eintritt und das andere inaktiv.

Geradlinige Kamerafahrt



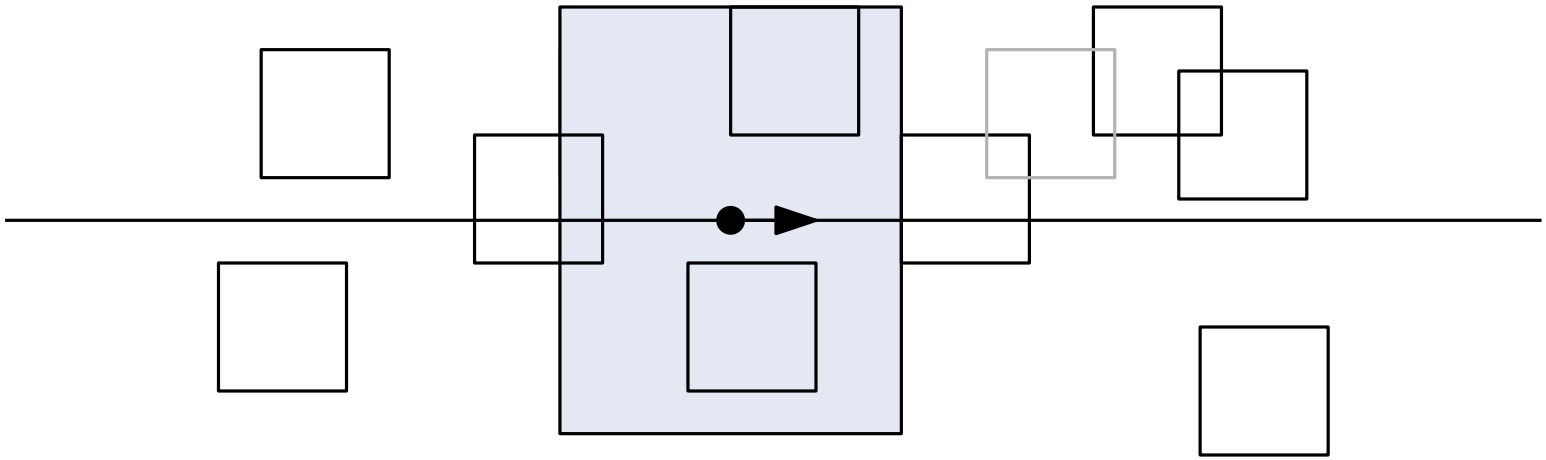
Annahme:

- Labels sind Quadrate (und in allgemeiner Lage).
- Trajektorie besteht aus Quadraten.

Greedy-Verfahren:

Wenn zwei Labels im Konflikt stehen, dann schalte dasjenige von beiden aktiv, das zuerst in den Viewport eintritt und das andere inaktiv.

Geradlinige Kamerafahrt



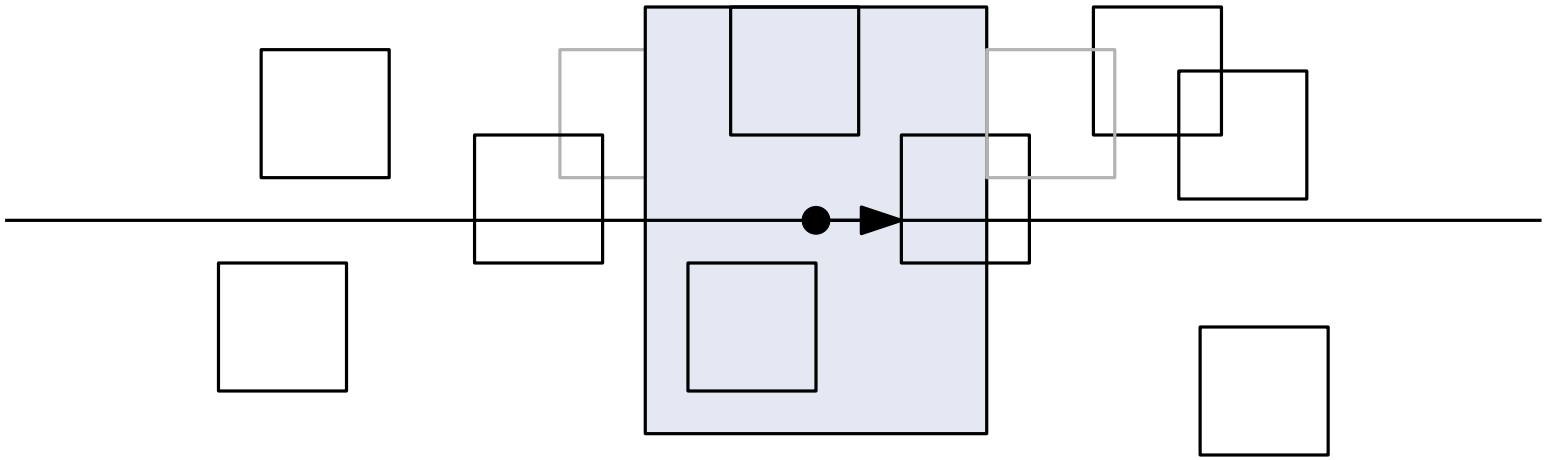
Annahme:

- Labels sind Quadrate (und in allgemeiner Lage).
- Trajektorie besteht aus Quadraten.

Greedy-Verfahren:

Wenn zwei Labels im Konflikt stehen, dann schalte dasjenige von beiden aktiv, das zuerst in den Viewport eintritt und das andere inaktiv.

Geradlinige Kamerafahrt



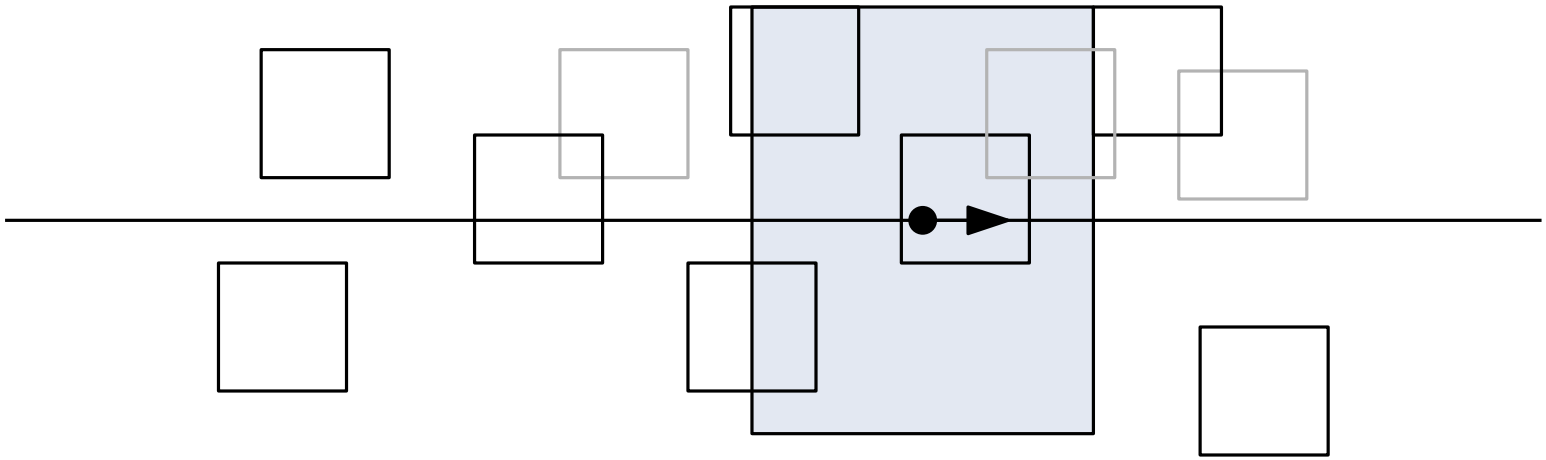
Annahme:

- Labels sind Quadrate (und in allgemeiner Lage).
- Trajektorie besteht aus Quadraten.

Greedy-Verfahren:

Wenn zwei Labels im Konflikt stehen, dann schalte dasjenige von beiden aktiv, das zuerst in den Viewport eintritt und das andere inaktiv.

Geradlinige Kamerafahrt



Annahme:

- Labels sind Quadrate (und in allgemeiner Lage).
- Trajektorie besteht aus Quadraten.

Greedy-Verfahren:

Wenn zwei Labels im Konflikt stehen, dann schalte dasjenige von beiden aktiv, das zuerst in den Viewport eintritt und das andere inaktiv.

Geradlinige Kamerafahrt

Warum ist das vorgeschlagene Verfahren eine $\frac{1}{2}$-Approximation?

Geradlinige Kamerafahrt

Warum ist das vorgeschlagene Verfahren eine $\frac{1}{2}$-Approximation?

$\mathcal{L}$: optimale Lösung

$\mathcal{L}'$: Greedy-Lösung

Geradlinige Kamerafahrt

Warum ist das vorgeschlagene Verfahren eine $\frac{1}{2}$-Approximation?

$\mathcal{L}$: optimale Lösung

$\mathcal{L}'$: Greedy-Lösung

1. Weise jedes Label $\ell \in \mathcal{L} \cap \mathcal{L}'$ sich selbst zu.

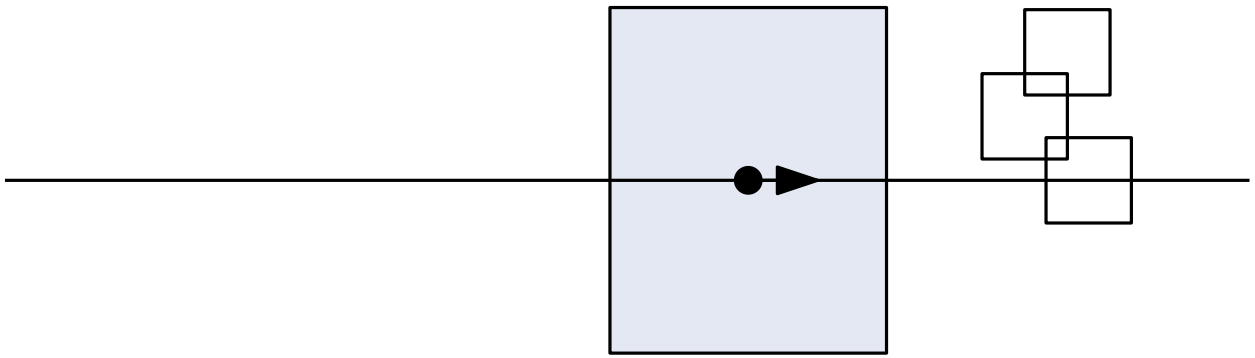
Geradlinige Kamerafahrt

Warum ist das vorgeschlagene Verfahren eine $\frac{1}{2}$-Approximation?

$\mathcal{L}$: optimale Lösung $\mathcal{L}'$: Greedy-Lösung

1. Weise jedes Label $\ell \in \mathcal{L} \cap \mathcal{L}'$ sich selbst zu.

2. Weise jedes Label $\ell \in \mathcal{L} \setminus \mathcal{L}'$ dem Label ℓ' zu, das am weitesten links liegt und ℓ schneidet:



Geradlinige Kamerafahrt

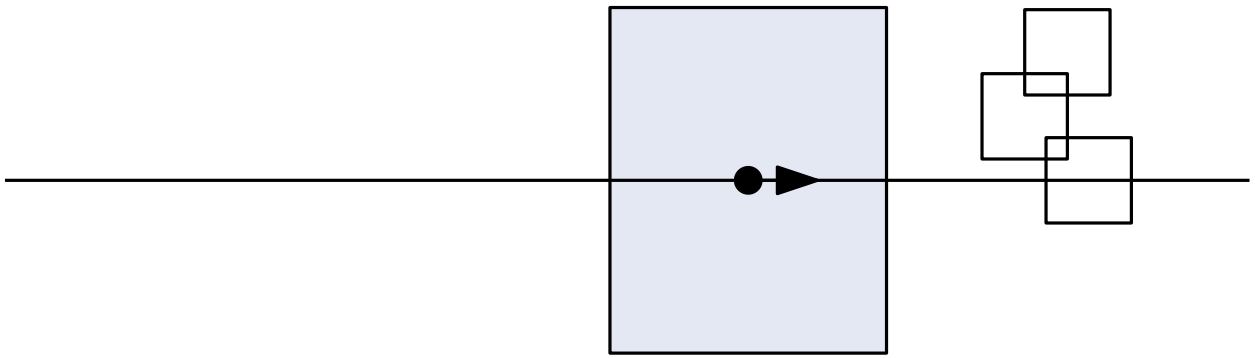
Warum ist das vorgeschlagene Verfahren eine $\frac{1}{2}$-Approximation?

$\mathcal{L}$: optimale Lösung $\mathcal{L}'$: Greedy-Lösung

1. Weise jedes Label $\ell \in \mathcal{L} \cap \mathcal{L}'$ sich selbst zu.

2. Weise jedes Label $\ell \in \mathcal{L} \setminus \mathcal{L}'$ dem Label ℓ' zu, das am weitesten links liegt und ℓ schneidet:

Wahl von ℓ':	Mittelpunkt von ℓ' liegt weiter links, als der von ℓ.
+Einheitsquadrate:	ℓ überdeckt mindestens eine rechte Ecke von ℓ', aber keine linke Ecke von ℓ.



Geradlinige Kamerafahrt

Warum ist das vorgeschlagene Verfahren eine $\frac{1}{2}$-Approximation?

$\mathcal{L}$: optimale Lösung $\mathcal{L}'$: Greedy-Lösung

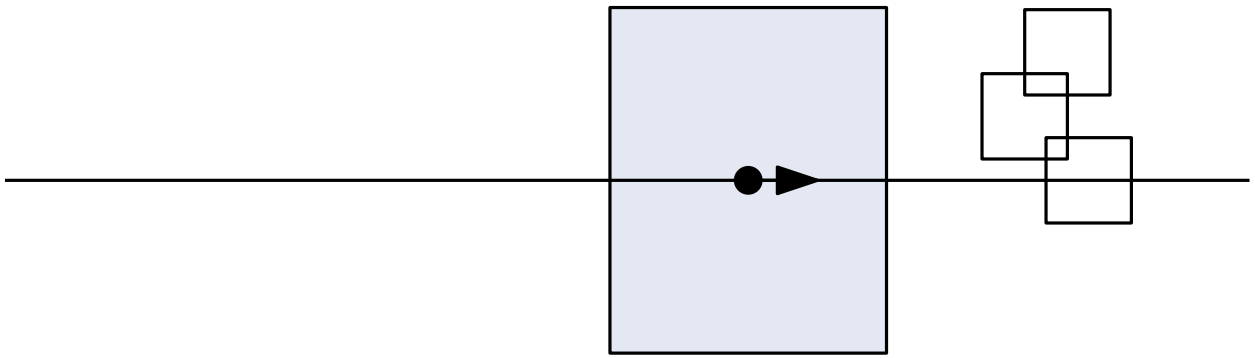
1. Weise jedes Label $\ell \in \mathcal{L} \cap \mathcal{L}'$ sich selbst zu.

2. Weise jedes Label $\ell \in \mathcal{L} \setminus \mathcal{L}'$ dem Label ℓ' zu, das am weitesten links liegt und ℓ schneidet:

Wahl von ℓ': Mittelpunkt von ℓ' liegt weiter links, als der von ℓ.

+Einheitsquadrate: ℓ überdeckt mindestens eine rechte Ecke von ℓ', aber keine linke Ecke von ℓ.

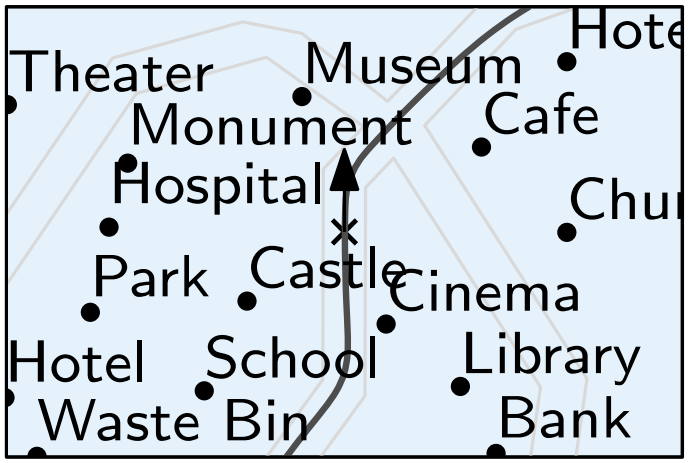
→ Maximal zwei Labels der optimalen Lösung können ℓ' überlappen.

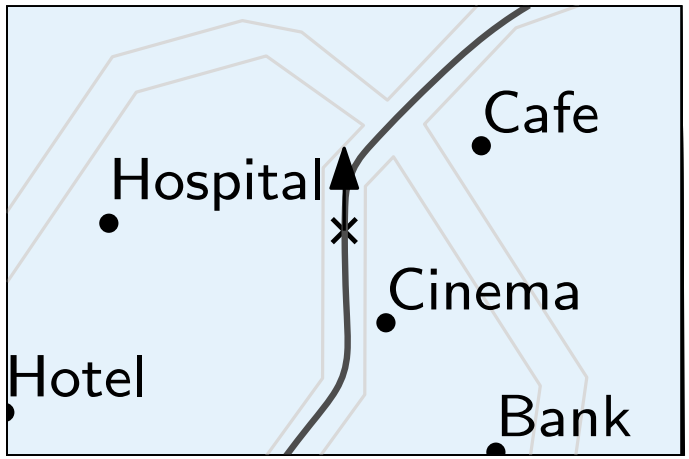


Problemdefinition

Problem k-RestrictedMaxTotal:
Gegeben: Menge I an Präsenzintervallen, Konflikte
Gesucht: $J \subseteq I$ sodass $\sum_{j \in J} \text{length}(j)$ maximal und J konfliktfrei ist, und sodass **es keinen Zeitpunkt t gibt, der von mehr als k Intervallen in J enthalten ist**.

Menge J an Intervallen ist **konfliktfrei** falls keine zwei Intervalle in J im Konflikt stehen.





$k = 5$

k-RestrictedMaxTotal

Karlsruhe Institute of Technology

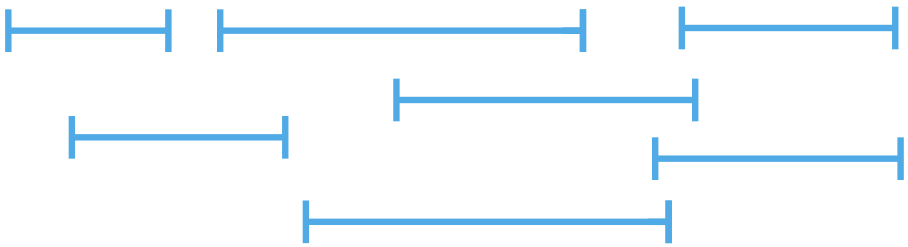
Theorem 4

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Case $\mathbf{k = 1}$: Problem ist äquivalent zu *maximum weighted independent set* in einem **Intervallgraph**.

$\Rightarrow O(n)$ Zeit falls Intervalle sortiert sind

[Hsiao et al., 1992]



k-RestrictedMaxTotal

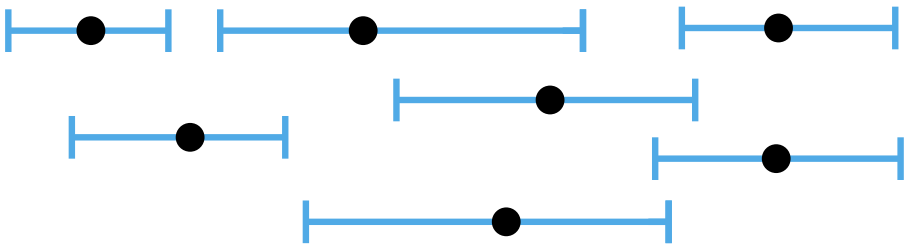
Theorem 4

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Case $\mathbf{k = 1}$: Problem ist äquivalent zu *maximum weighted independent set* in einem **Intervallgraph**.

$\Rightarrow O(n)$ Zeit falls Intervalle sortiert sind

[Hsiao et al., 1992]



k-RestrictedMaxTotal

Karlsruhe Institute of Technology

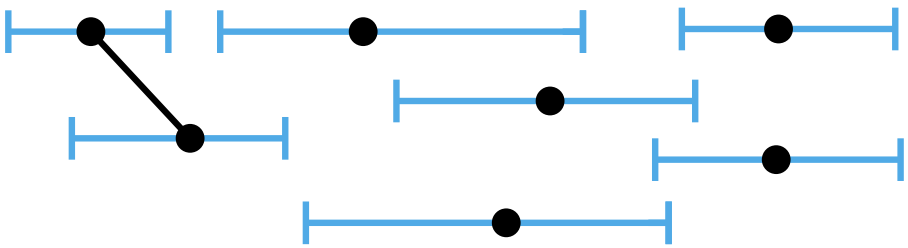
Theorem 4

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Case $\mathbf{k = 1}$: Problem ist äquivalent zu *maximum weighted independent set* in einem **Intervallgraph**.

$\Rightarrow O(n)$ Zeit falls Intervalle sortiert sind

[Hsiao et al., 1992]



k-RestrictedMaxTotal

Karlsruhe Institute of Technology

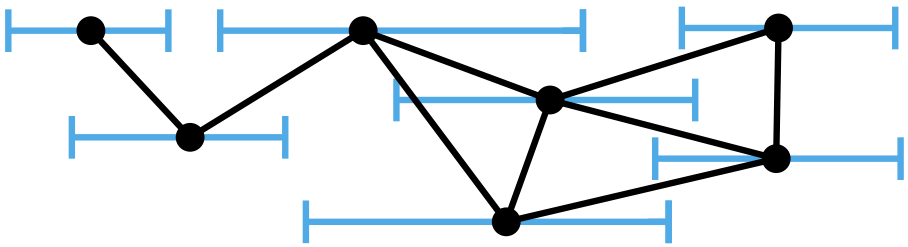
Theorem 4

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Case $\mathbf{k = 1}$: Problem ist äquivalent zu *maximum weighted independent set* in einem **Intervallgraph**.

$\Rightarrow O(n)$ Zeit falls Intervalle sortiert sind

[Hsiao et al., 1992]



k-RestrictedMaxTotal

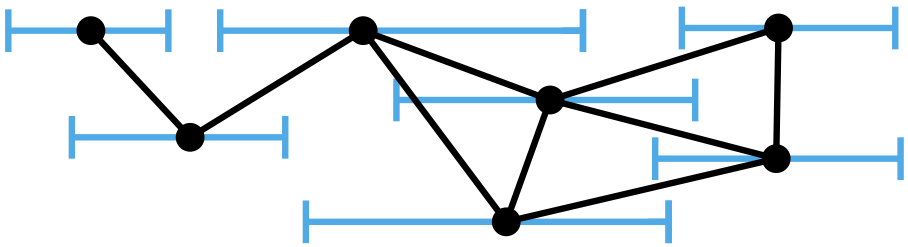
Theorem 4

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Case $\mathbf{k = 1}$: Problem ist äquivalent zu *maximum weighted independent set* in einem **Intervallgraph**.

$\Rightarrow O(n)$ Zeit falls Intervalle sortiert sind

[Hsiao et al., 1992]



Gewicht eines Knoten = Länge eines Intervals

k-RestrictedMaxTotal

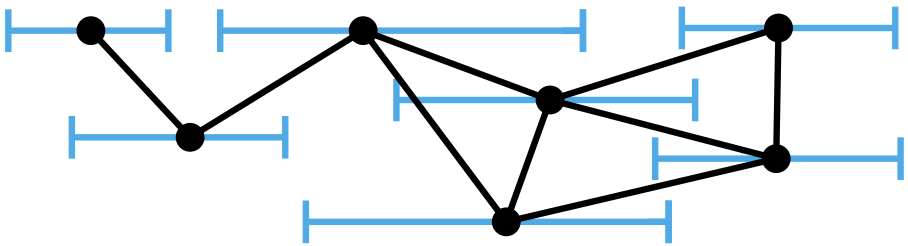
Theorem 4

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Case $\mathbf{k = 1}$: Problem ist äquivalent zu *maximum weighted independent set* in einem **Intervallgraph**.

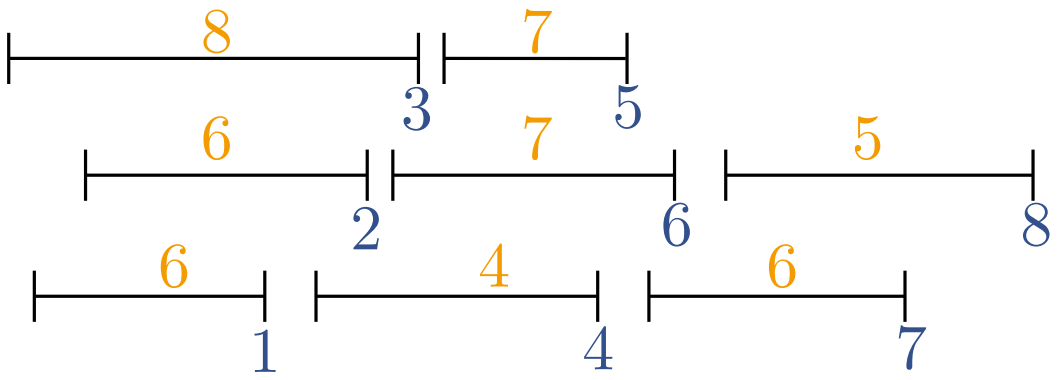
$\Rightarrow O(n)$ Zeit falls Intervalle sortiert sind

[Hsiao et al., 1992]



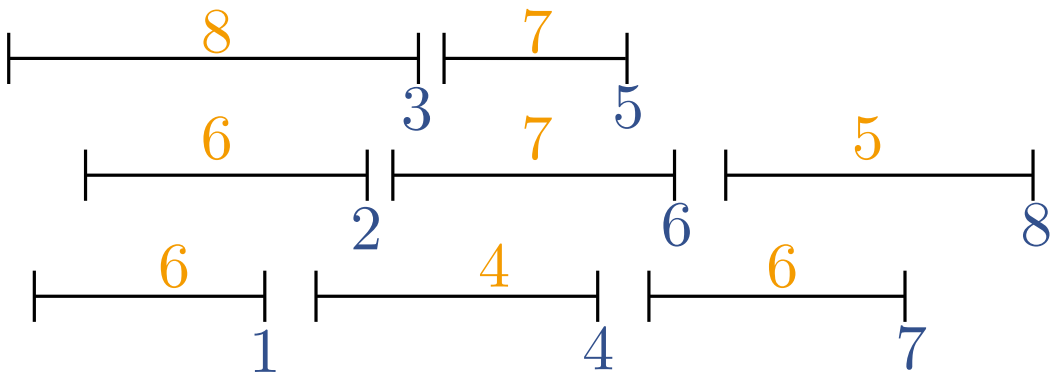
Gewicht eines Knoten = Länge eines Intervals

s. Übungsblatt



Index des Intervals sortiert nach rechtem Endpunkt.

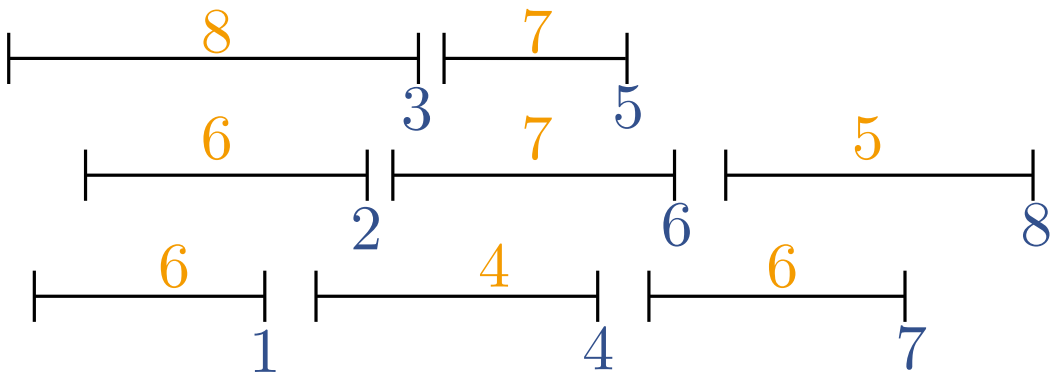
Gewicht des Intervals i: w(i)



Index des Intervals sortiert nach rechtem Endpunkt.

Gewicht des Intervals i: w(i)

Beobachtung: Gehört Interval i zur optimalen Lösung, dann trennt i Instanz in zwei unabhängige Teile auf.

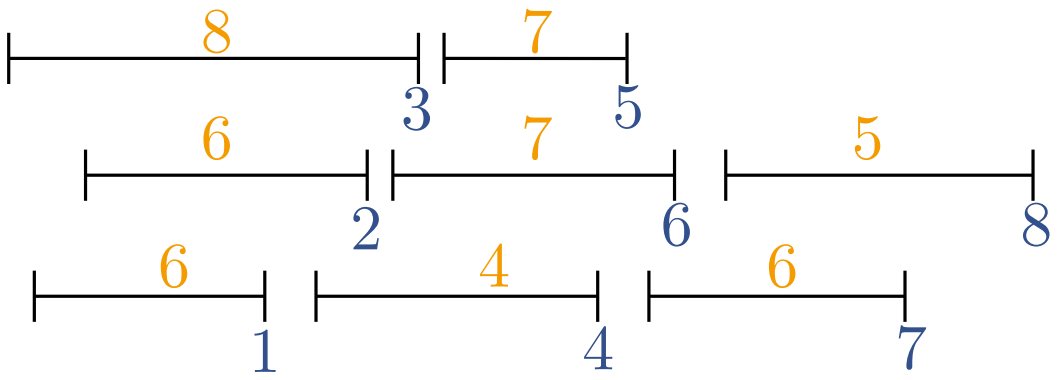


Index des Intervals sortiert nach rechtem Endpunkt.

Gewicht des Intervals i: w(i)

Beobachtung: Gehört Interval i zur optimalen Lösung, dann trennt i Instanz in zwei unabhängige Teile auf.

$T[i]=$ Wert der optimalen Lösung für Intervalle $1, \ldots, i$.



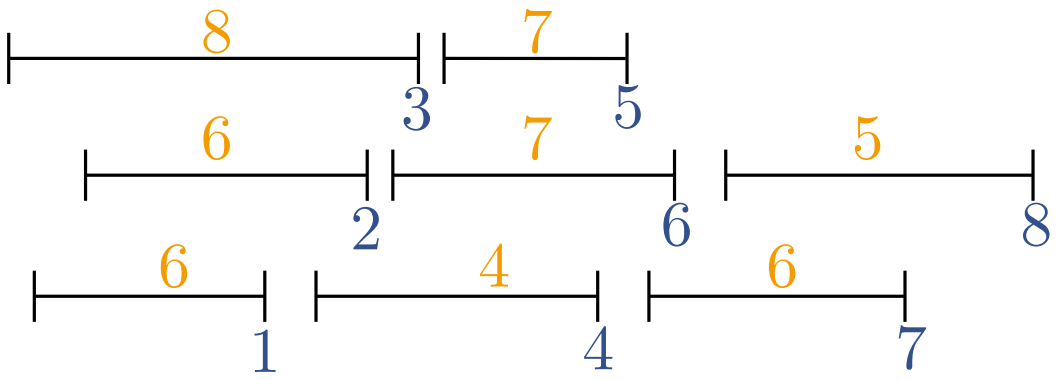
Index des Intervals sortiert nach rechtem Endpunkt.

Gewicht des Intervals i: w(i)

Beobachtung: Gehört Interval i zur optimalen Lösung, dann trennt i Instanz in zwei unabhängige Teile auf.

$T[i]=$ Wert der optimalen Lösung für Intervalle $1, \ldots, i$.

$T[0] := 0$



Index des Intervals sortiert nach rechtem Endpunkt.

Gewicht des Intervals i: w(i)

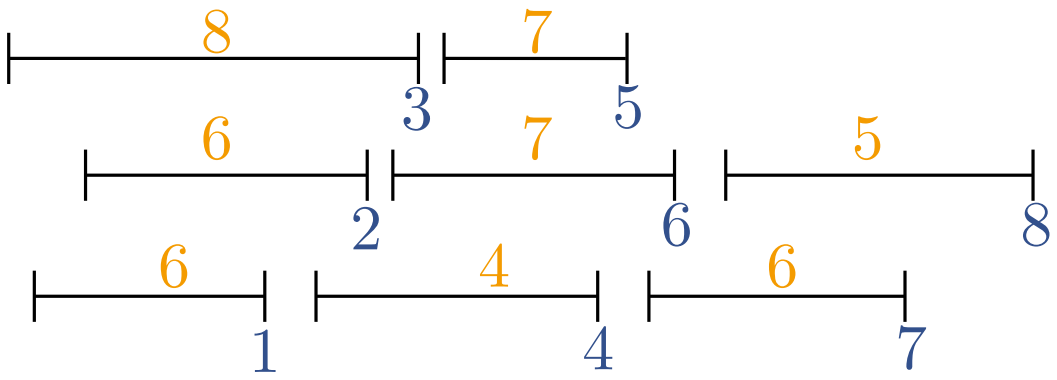
Beobachtung: Gehört Interval i zur optimalen Lösung, dann trennt i Instanz in zwei unabhängige Teile auf.

$T[i]$= Wert der optimalen Lösung für Intervalle $1, \ldots, i$.

$T[0] := 0$

$T[i] := \max\{T[i-1], w(i) + \max\{T[j] \mid j < i \text{ und } r(j) < l(i)\}\}$

$l(i)/r(i) =$ linker/rechter Endpunkt von Interval i



Index des Intervals sortiert nach rechtem Endpunkt.

Gewicht des Intervals i: w(i)

Beobachtung: Gehört Interval i zur optimalen Lösung, dann trennt i Instanz in zwei unabhängige Teile auf.

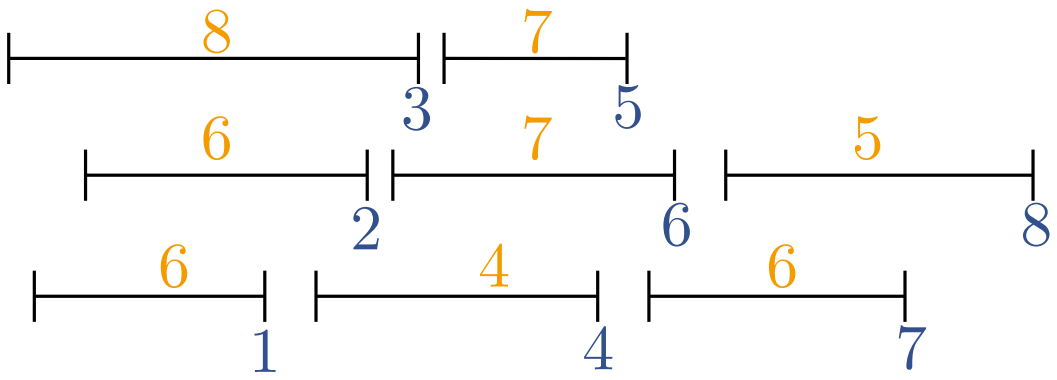
$T[i]$= Wert der optimalen Lösung für Intervalle $1, \ldots, i$.

$T[0] := 0$

$T[i] := \max\{T[i-1], w(i) + \max\{T[j] \mid j < i \text{ und } r(j) < l(i)\}\}$

Laufzeit?

$l(i)/r(i) =$ linker/rechter Endpunkt von Interval i



Index des Intervals sortiert nach rechtem Endpunkt.

Gewicht des Intervals i: w(i)

Beobachtung: Gehört Interval i zur optimalen Lösung, dann trennt i Instanz in zwei unabhängige Teile auf.

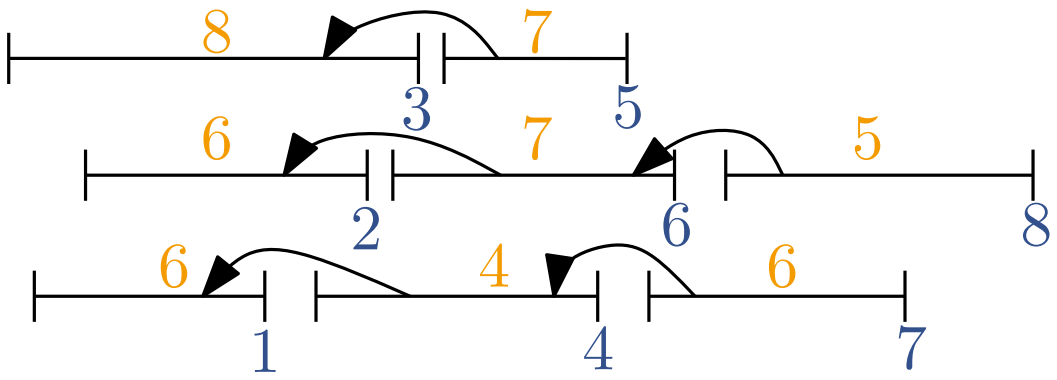
$T[i]$= Wert der optimalen Lösung für Intervalle $1, \ldots, i$.

$T[0] := 0$

$T[i] := \max\{T[i-1], w(i) + \max\{T[j] \mid j < i \text{ und } r(j) < l(i)\}\}$

$O(n^2)$: Wie kann Laufzeit auf $O(n)$ verbessert werden?

$l(i)/r(i) =$ linker/rechter Endpunkt von Interval i



Index des Intervals sortiert nach rechtem Endpunkt.

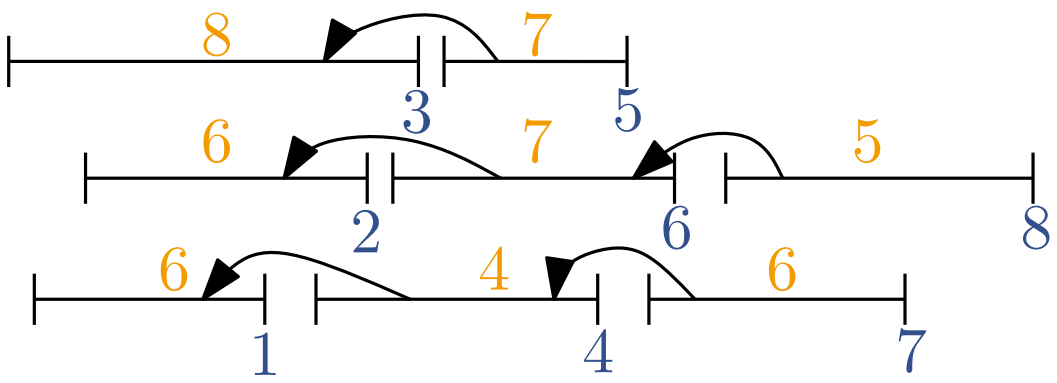
Gewicht des Intervals i: w(i)

Beobachtung: Gehört Interval i zur optimalen Lösung, dann trennt i Instanz in zwei unabhängige Teile auf.

$T[i]=$ Wert der optimalen Lösung für Intervalle $1, \ldots, i$.

Berechne für jedes Intervall i Vorgänger $pre(i)$ in $O(n)$ Zeit.

$pre(i) =$rechtestes Interval mit rechten Endpunkt links von Interval i.



Index des Intervals sortiert nach rechtem Endpunkt.

Gewicht des Intervals i: w(i)

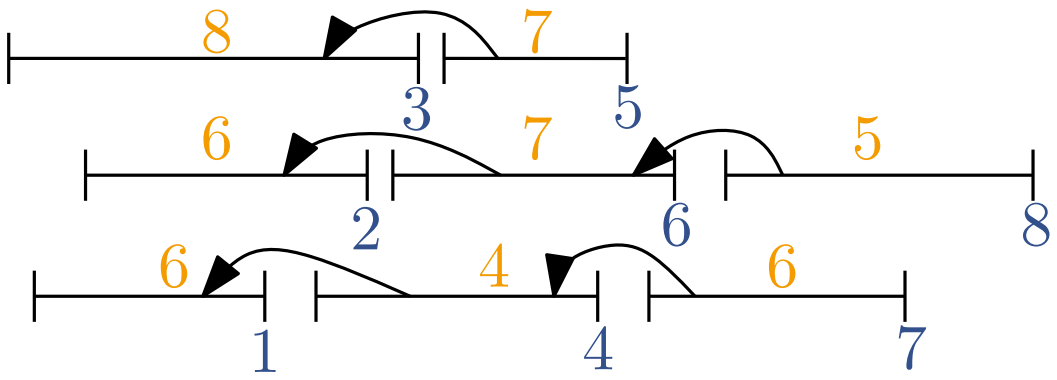
Beobachtung: Gehört Interval i zur optimalen Lösung, dann trennt i Instanz in zwei unabhängige Teile auf.

$T[i]=$ Wert der optimalen Lösung für Intervalle $1, \ldots, i$.

Berechne für jedes Intervall i Vorgänger $pre(i)$ in $O(n)$ Zeit.

$pre(i)$ =rechtestes Interval mit rechten Endpunkt links von Interval i.

$$T[0] := 0$$
$$T[i] := \max\{T[i-1], w(i) + T[pre(i)]\}$$



Index des Intervals sortiert nach rechtem Endpunkt.

Gewicht des Intervals i: w(i)

Beobachtung: Gehört Interval i zur optimalen Lösung, dann trennt i Instanz in zwei unabhängige Teile auf.

$T[i]=$ Wert der optimalen Lösung für Intervalle $1, \ldots, i$.

Berechne für jedes Intervall i Vorgänger $pre(i)$ in $O(n)$ Zeit.

$pre(i) =$rechtestes Interval mit rechten Endpunkt links von Interval i.

$$T[0] := 0$$
$$T[i] := \max\{T[i-1], w(i) + T[pre(i)]\}$$

Laufzeit $O(n)$.

Evaluation

Gut gefallen hat mir insbesondere:

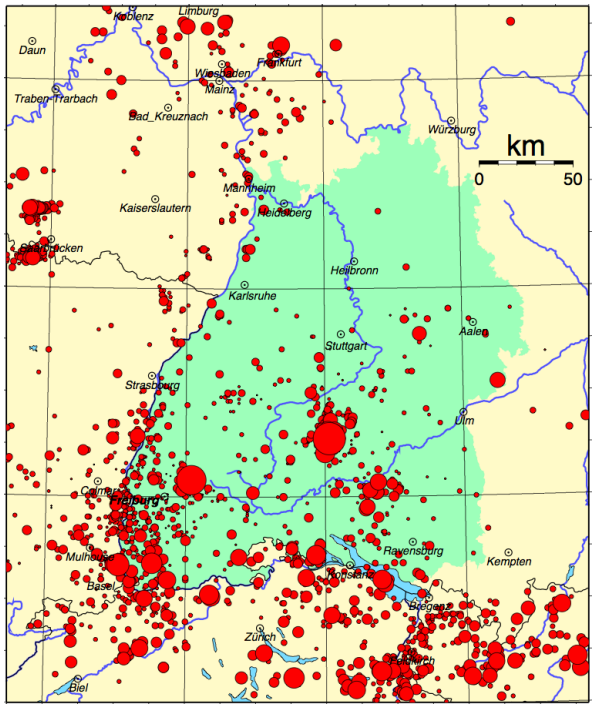
- Animationen/Illustrationen
- Folien
- Wiederholung $(2\times)$
- weiterführende Aufgaben
- Interaktivität

Fragen und Anregungen:

- Mehr Praxis-Tipps: Implementierungsdetails.
- Mehr Denkanstöße: Wo ist noch Forschungspotential.
- Wie sieht die Prüfung aus?

Proportional Symbol Maps für Punktdaten

[Cabello et al. 2010]



Problemstellung

Geg: Menge $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$ von Punkten, Menge $W = \{w_1, \ldots, w_n\} \subset \mathbb{R}^+$ von zugehörigen Werten

Ges: *gute* Visualisierung von P und W, wobei (p_i, w_i) Kreisscheibe D_i mit Mittelpunkt p_i und Fläche w_i def.

Problemstellung

Geg: Menge $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$ von Punkten, Menge $W = \{w_1, \ldots, w_n\} \subset \mathbb{R}^+$ von zugehörigen Werten

Ges: *gute* Visualisierung von P und W, wobei (p_i, w_i) Kreisscheibe D_i mit Mittelpunkt p_i und Fläche w_i def.

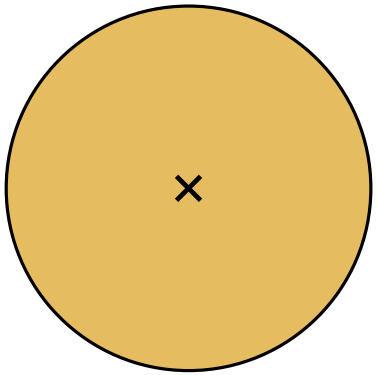
Wo ist dabei das Problem?

Problemstellung

Geg: Menge $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$ von Punkten, Menge $W = \{w_1, \ldots, w_n\} \subset \mathbb{R}^+$ von zugehörigen Werten

Ges: *gute* Visualisierung von P und W, wobei (p_i, w_i) Kreisscheibe D_i mit Mittelpunkt p_i und Fläche w_i def.

Wo ist dabei das Problem?

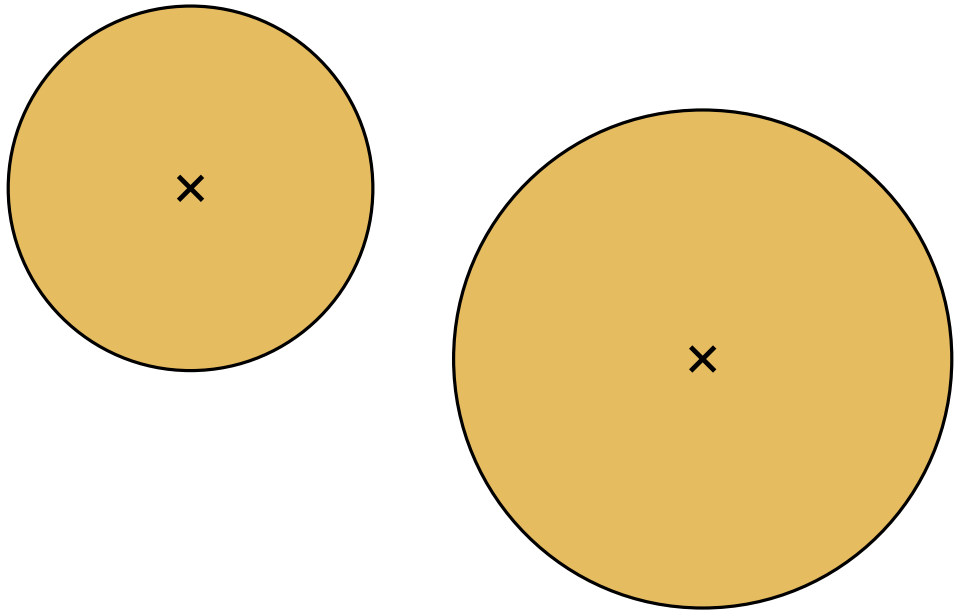


Problemstellung

Geg: Menge $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$ von Punkten, Menge $W = \{w_1, \ldots, w_n\} \subset \mathbb{R}^+$ von zugehörigen Werten

Ges: *gute* Visualisierung von P und W, wobei (p_i, w_i) Kreisscheibe D_i mit Mittelpunkt p_i und Fläche w_i def.

Wo ist dabei das Problem?

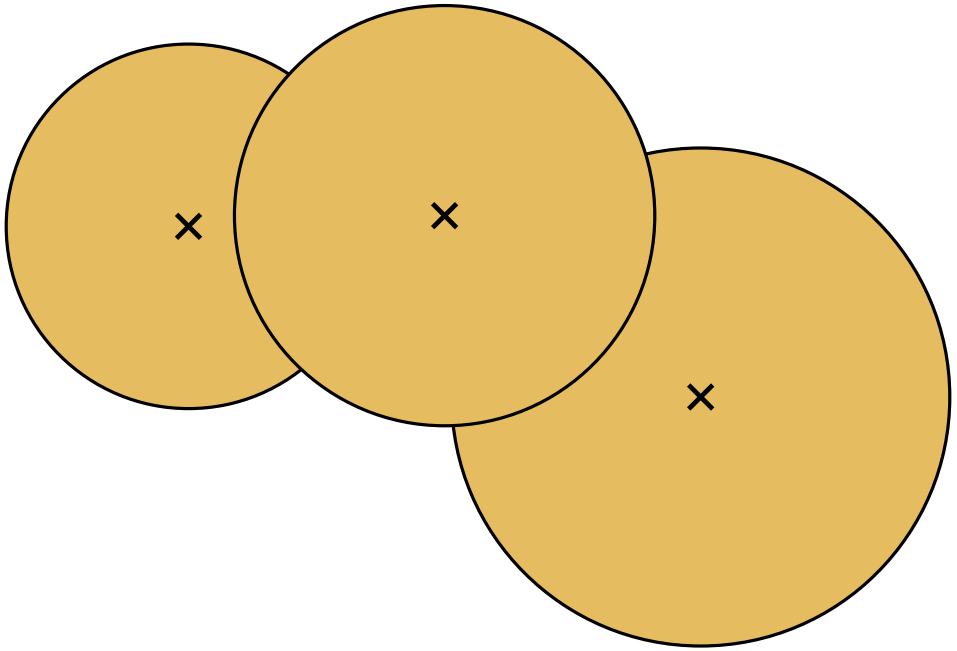


Problemstellung

Geg: Menge $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$ von Punkten, Menge $W = \{w_1, \ldots, w_n\} \subset \mathbb{R}^+$ von zugehörigen Werten

Ges: *gute* Visualisierung von P und W, wobei (p_i, w_i) Kreisscheibe D_i mit Mittelpunkt p_i und Fläche w_i def.

Wo ist dabei das Problem?

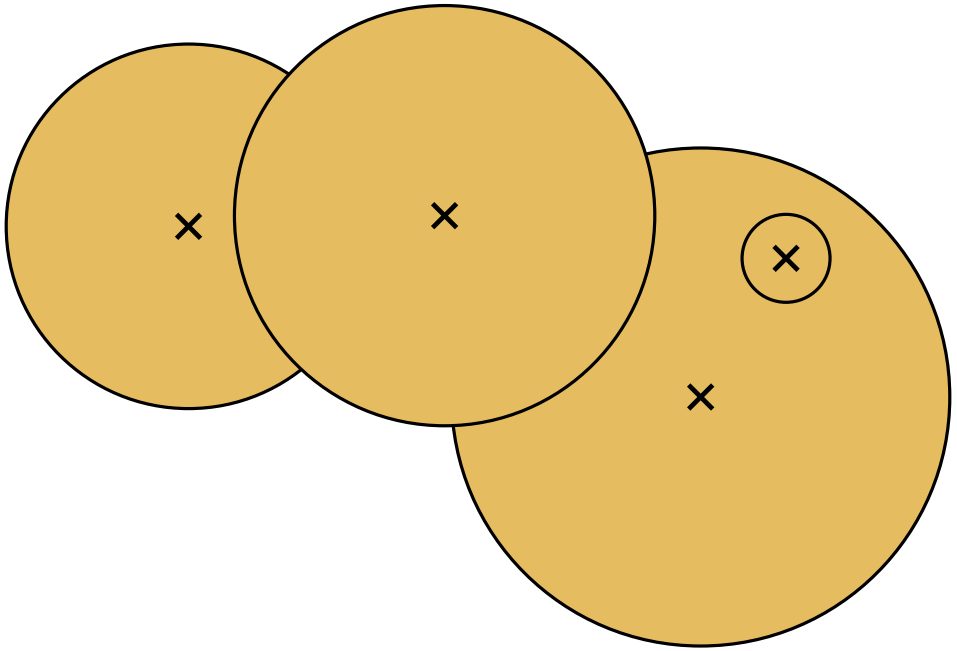


Problemstellung

Geg: Menge $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$ von Punkten, Menge $W = \{w_1, \ldots, w_n\} \subset \mathbb{R}^+$ von zugehörigen Werten

Ges: *gute* Visualisierung von P und W, wobei (p_i, w_i) Kreisscheibe D_i mit Mittelpunkt p_i und Fläche w_i def.

Wo ist dabei das Problem?

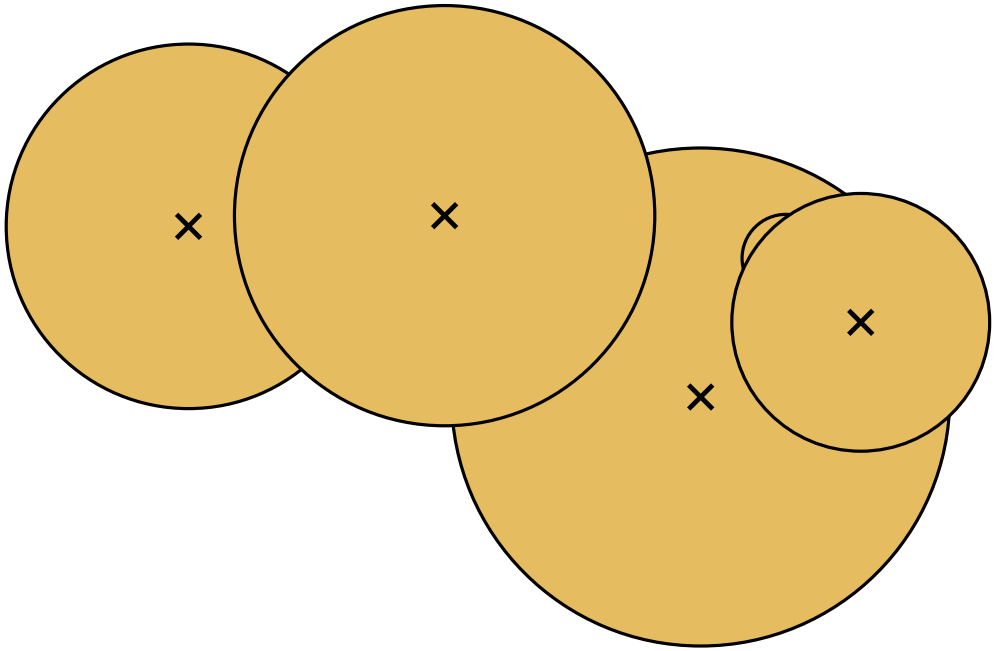


Problemstellung

Geg: Menge $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$ von Punkten, Menge $W = \{w_1, \ldots, w_n\} \subset \mathbb{R}^+$ von zugehörigen Werten

Ges: *gute* Visualisierung von P und W, wobei (p_i, w_i) Kreisscheibe D_i mit Mittelpunkt p_i und Fläche w_i def.

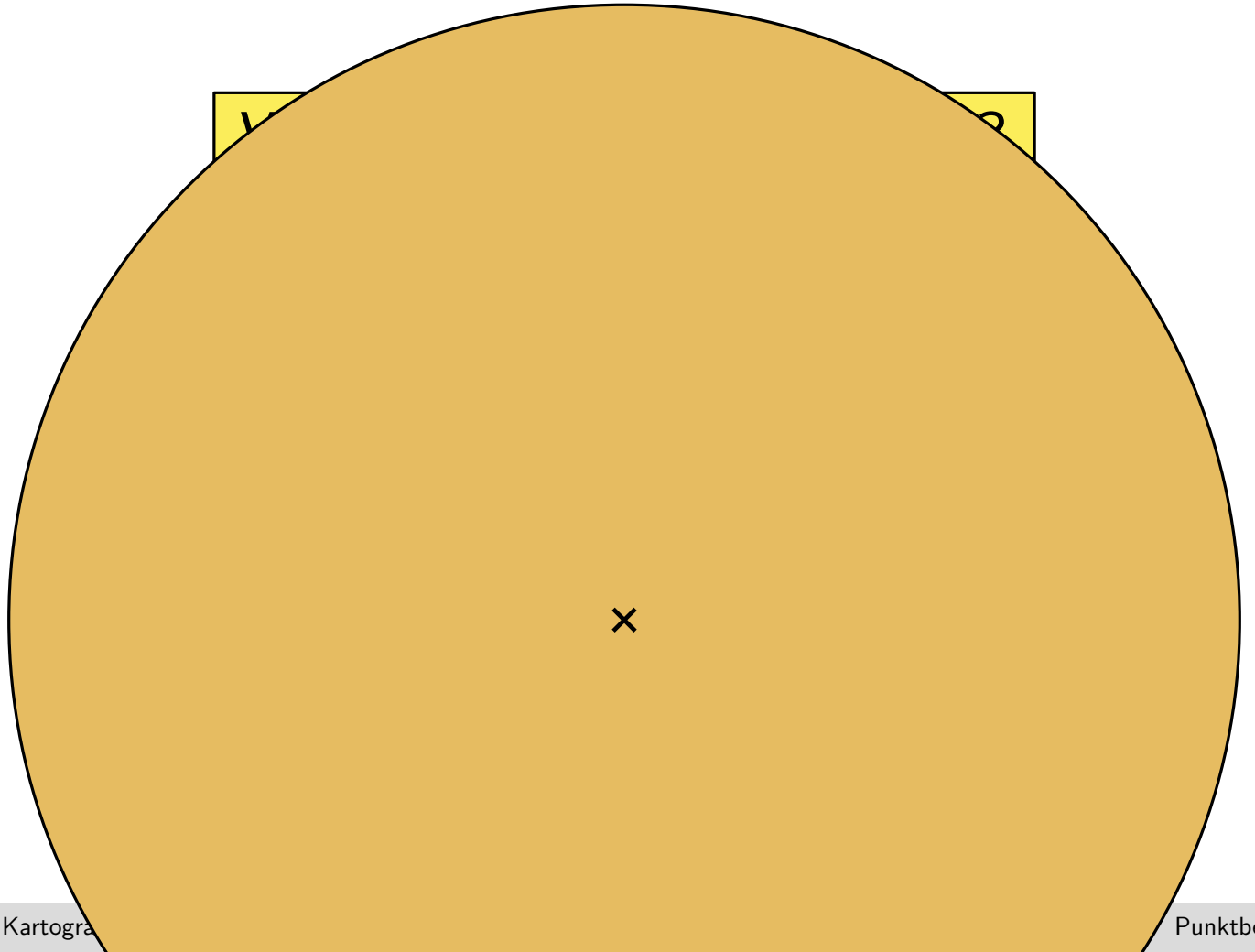
Wo ist dabei das Problem?



Problemstellung

Geg: Menge $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$ von Punkten, Menge $W = \{w_1, \ldots, w_n\} \subset \mathbb{R}^+$ von zugehörigen Werten

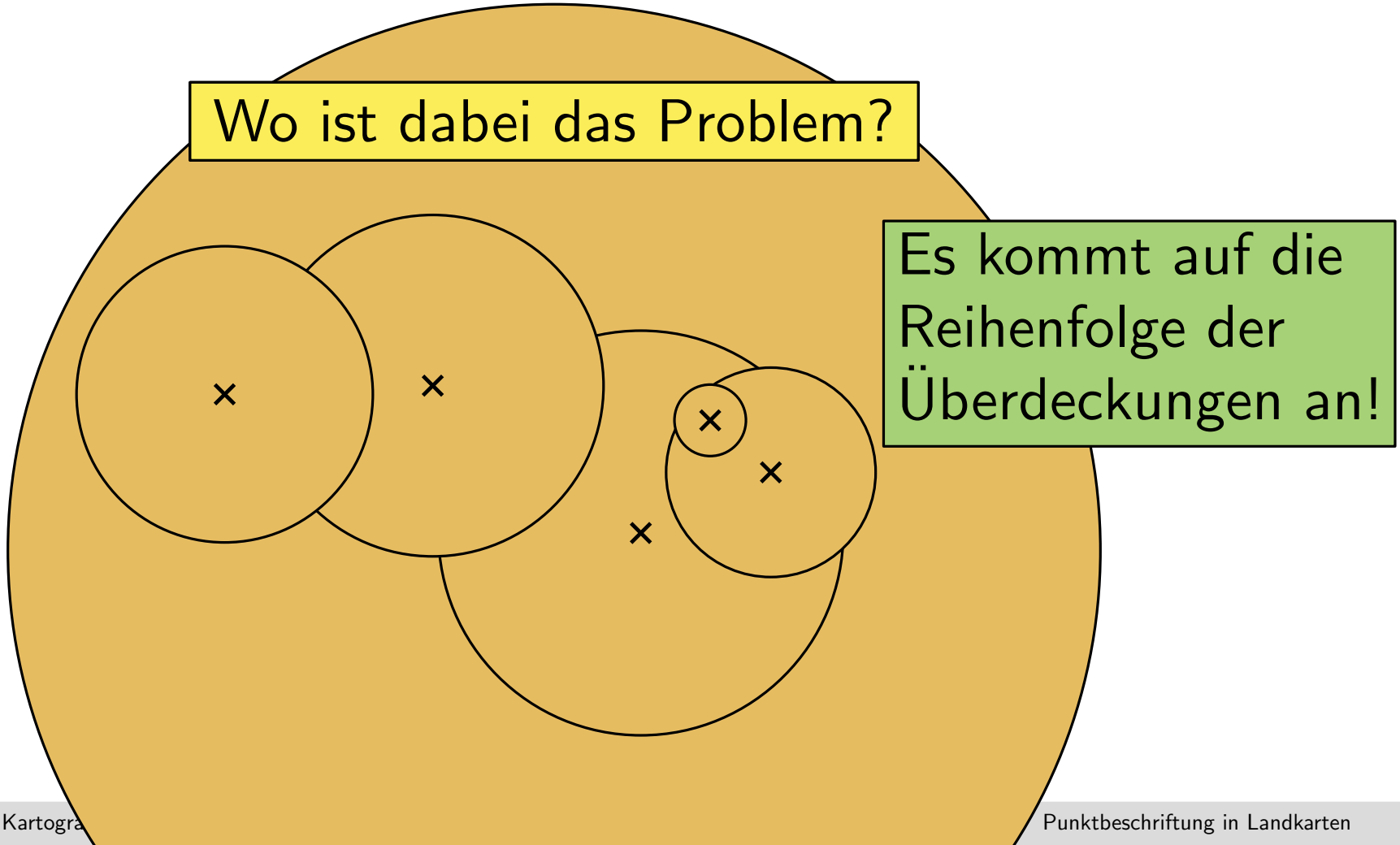
Ges: *gute* Visualisierung von P und W, wobei (p_i, w_i) Kreisscheibe D_i mit Mittelpunkt p_i und Fläche w_i def.



Problemstellung

Geg: Menge $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$ von Punkten, Menge $W = \{w_1, \ldots, w_n\} \subset \mathbb{R}^+$ von zugehörigen Werten

Ges: *gute* Visualisierung von P und W, wobei (p_i, w_i) Kreisscheibe D_i mit Mittelpunkt p_i und Fläche w_i def.



Problemstellung

Geg: Menge $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$ von Punkten, Menge $W = \{w_1, \ldots, w_n\} \subset \mathbb{R}^+$ von zugehörigen Werten

Ges: *gute* Visualisierung von P und W, wobei (p_i, w_i) Kreisscheibe D_i mit Mittelpunkt p_i und Fläche w_i def.

Qualitätskriterien:

Ideen?

Problemstellung

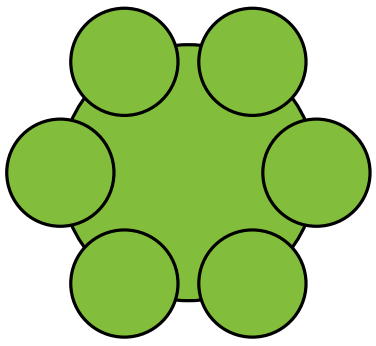
Geg: Menge $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$ von Punkten, Menge $W = \{w_1, \ldots, w_n\} \subset \mathbb{R}^+$ von zugehörigen Werten

Ges: *gute* Visualisierung von P und W, wobei (p_i, w_i) Kreisscheibe D_i mit Mittelpunkt p_i und Fläche w_i def.

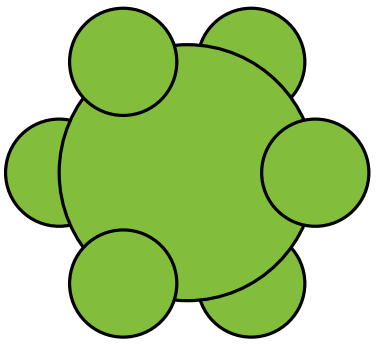
Qualitätskriterien:

relativ oder absolut?

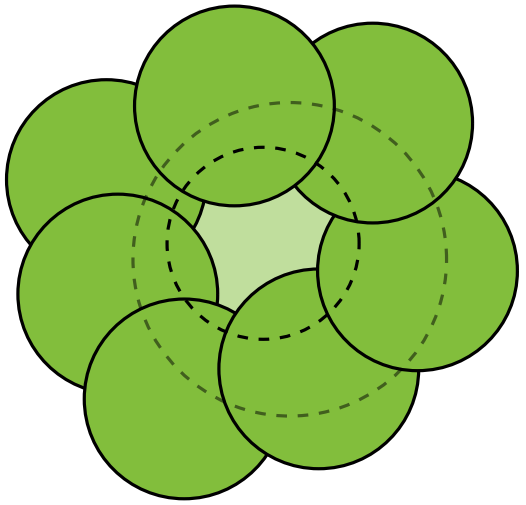
- maximiere die Länge des minimal sichtbaren Randes aller Scheiben
- maximiere die Gesamtlänge des sichtbaren Randes



MaxTotal



MaxMin



sichtbare Fläche problematisch

Problemstellung

Geg: Menge $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$ von Punkten, Menge $W = \{w_1, \ldots, w_n\} \subset \mathbb{R}^+$ von zugehörigen Werten

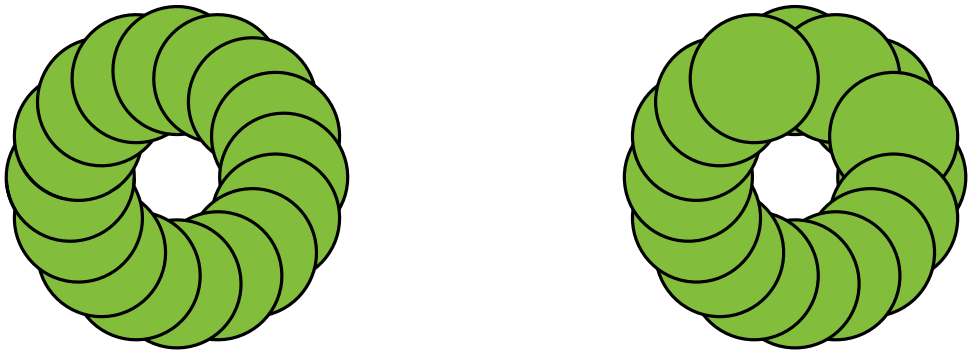
Ges: *gute* Visualisierung von P und W, wobei (p_i, w_i) Kreisscheibe D_i mit Mittelpunkt p_i und Fläche w_i def.

Qualitätskriterien:

- maximiere die Länge des minimal sichtbaren Randes aller Scheiben
- maximiere die Gesamtlänge des sichtbaren Randes

Modelle:

- Physisch realisierbares Modell: mit Münzen nachbaubar
- Stacking Modell: Totalordnung der Scheiben



Greedy-Algorithmus MaxMin Stacking

Input: Menge von Kreisscheiben $\mathcal{D} = \{D_1, \ldots, D_n\}$
Output: Permutation π als optimale Stacking Ordnung

---

initialisiere $\mathcal{S} \leftarrow \mathcal{D}$
for $i = 1, \ldots, n$ **do**
  foreach $D_j \in \mathcal{S}$ **do**
    vis$(D_j) \leftarrow$ sichtb. Rand von D_j bzgl. $\mathcal{S}$ falls $\pi(i) = j$
  $D_k \leftarrow \arg\max_{D_j \in \mathcal{S}}$ vis(D_j)
  set $\pi(i) \leftarrow k$
  $\mathcal{S} \leftarrow \mathcal{S} \setminus \{D_k\}$
return π

MaxMin Stacking: Ergebnis

Satz 2: Gegeben n Kreisscheiben in der Ebene berechnet der Greedy-Algorithmus eine Stacking Ordnung, die den minimal sichtbaren Rand aller Kreisscheiben maximiert. Der Algorithmus lässt sich in $O(n^2 \log n)$ Laufzeit implementieren.

MaxMin Stacking: Ergebnis

Satz 2: Gegeben n Kreisscheiben in der Ebene berechnet der Greedy-Algorithmus eine Stacking Ordnung, die den minimal sichtbaren Rand aller Kreisscheiben maximiert. Der Algorithmus lässt sich in $O(n^2 \log n)$ Laufzeit implementieren.

Beweis:

- Korrektheit: Vergleiche Lösung $\mathcal{A}$ des Algorithmus mit optimaler Lösung $\mathcal{O}$ und nutze Austauschargument
- Laufzeit: nutze Variante von Segment-Bäumen
  [de Berg et al. '08, Kap. 10.3]

Austauschargument

Gegeben: Kreisscheiben $D_1, \cdots, D_n$

Austauschargument

Gegeben: Kreisscheiben $D_1, \cdots, D_n$

Greedy-Lösung $\mathcal{L}'$: D_{i_1} D_{i_2} D_{i_3} D_{i_4} D_{i_5} D_{i_6}

Optimale Lösung $\mathcal{L}$: D_{i_1} D_{i_2} D_{i_5} D_{i_6} D_{i_4} D_{i_3}

Austauschargument

Gegeben: Kreisscheiben $D_1, \cdots, D_n$

Greedy-Lösung $\mathcal{L}'$: D_{i_1} D_{i_2} D_{i_3} D_{i_4} D_{i_5} D_{i_6} $j = i_3$

Optimale Lösung $\mathcal{L}$: D_{i_1} D_{i_2} D_{i_5} D_{i_6} D_{i_4} D_{i_3} $i = i_5$

1. Betrachte erste Stelle in $\mathcal{L}$ und $\mathcal{L}'$ die sich unterscheidet:
$D_i \in \mathcal{L}$, $D_j \in \mathcal{L}'$ mit $i \neq j$ aber an selber Position.

Austauschargument

Gegeben: Kreisscheiben $D_1, \cdots, D_n$

Greedy-Lösung $\mathcal{L}'$:	D_{i_1}	D_{i_2}	D_{i_3}	D_{i_4}	D_{i_5}	D_{i_6}	$j = i_3$
Optimale Lösung $\mathcal{L}$:	D_{i_1}	D_{i_2}	D_{i_5}	D_{i_6}	D_{i_4}	D_{i_3}	$i = i_5$
Neue Lösung $\mathcal{L}''$:	D_{i_1}	D_{i_2}	D_{i_3}	D_{i_5}	D_{i_6}	D_{i_4}	

1. Betrachte erste Stelle in $\mathcal{L}$ und $\mathcal{L}'$ die sich unterscheidet:
$D_i \in \mathcal{L}$, $D_j \in \mathcal{L}'$ mit $i \neq j$ aber an selber Position.

2. Verschiebe D_j vor D_i in $\mathcal{L}$. $\longrightarrow$ $vis_{\mathcal{L}''}(D_k) \geq vis_{\mathcal{L}}(D_k)$ für $k \neq j$

Austauschargument

Gegeben: Kreisscheiben $D_1, \cdots, D_n$

Greedy-Lösung $\mathcal{L}'$:	D_{i_1}	D_{i_2}	D_{i_3}	D_{i_4}	D_{i_5}	D_{i_6}	$j = i_3$
Optimale Lösung $\mathcal{L}$:	D_{i_1}	D_{i_2}	D_{i_5}	D_{i_6}	D_{i_4}	D_{i_3}	$i = i_5$
Neue Lösung $\mathcal{L}''$:	D_{i_1}	D_{i_2}	D_{i_3}	D_{i_5}	D_{i_6}	D_{i_4}	

1. Betrachte erste Stelle in $\mathcal{L}$ und $\mathcal{L}'$ die sich unterscheidet:
$D_i \in \mathcal{L}$, $D_j \in \mathcal{L}'$ mit $i \neq j$ aber an selber Position.

2. Verschiebe D_j vor D_i in $\mathcal{L}$. $\longrightarrow$ $vis_{\mathcal{L}''}(D_k) \geq vis_{\mathcal{L}}(D_k)$ für $k \neq j$

Zeige: $vis_{\mathcal{L}''}(D_j) \geq \min\limits_{1 \leq k \leq n} vis_{\mathcal{L}}(D_k)$

Austauschargument

Gegeben: Kreisscheiben $D_1, \cdots, D_n$

Greedy-Lösung $\mathcal{L}'$:	D_{i_1}	D_{i_2}	D_{i_3}	D_{i_4}	D_{i_5}	D_{i_6}	$j = i_3$
Optimale Lösung $\mathcal{L}$:	D_{i_1}	D_{i_2}	D_{i_5}	D_{i_6}	D_{i_4}	D_{i_3}	$i = i_5$
Neue Lösung $\mathcal{L}''$:	D_{i_1}	D_{i_2}	D_{i_3}	D_{i_5}	D_{i_6}	D_{i_4}	

1. Betrachte erste Stelle in $\mathcal{L}$ und $\mathcal{L}'$ die sich unterscheidet:
$D_i \in \mathcal{L}$, $D_j \in \mathcal{L}'$ mit $i \neq j$ aber an selber Position.

2. Verschiebe D_j vor D_i in $\mathcal{L}$. $\longrightarrow$ $vis_{\mathcal{L}''}(D_k) \geq vis_{\mathcal{L}}(D_k)$ für $k \neq j$

Zeige: $vis_{\mathcal{L}''}(D_j) \geq \min\limits_{1 \leq k \leq n} vis_{\mathcal{L}}(D_k)$

3. Nach Def.: $vis_{\mathcal{L}'}(D_j) \geq vis_{\mathcal{L}}(D_i)$

Austauschargument

Gegeben: Kreisscheiben $D_1, \cdots, D_n$

Greedy-Lösung $\mathcal{L}'$:	D_{i_1}	D_{i_2}	D_{i_3}	D_{i_4}	D_{i_5}	D_{i_6}	$j = i_3$
Optimale Lösung $\mathcal{L}$:	D_{i_1}	D_{i_2}	D_{i_5}	D_{i_6}	D_{i_4}	D_{i_3}	$i = i_5$
Neue Lösung $\mathcal{L}''$:	D_{i_1}	D_{i_2}	D_{i_3}	D_{i_5}	D_{i_6}	D_{i_4}	

1. Betrachte erste Stelle in $\mathcal{L}$ und $\mathcal{L}'$ die sich unterscheidet:
$D_i \in \mathcal{L}$, $D_j \in \mathcal{L}'$ mit $i \neq j$ aber an selber Position.

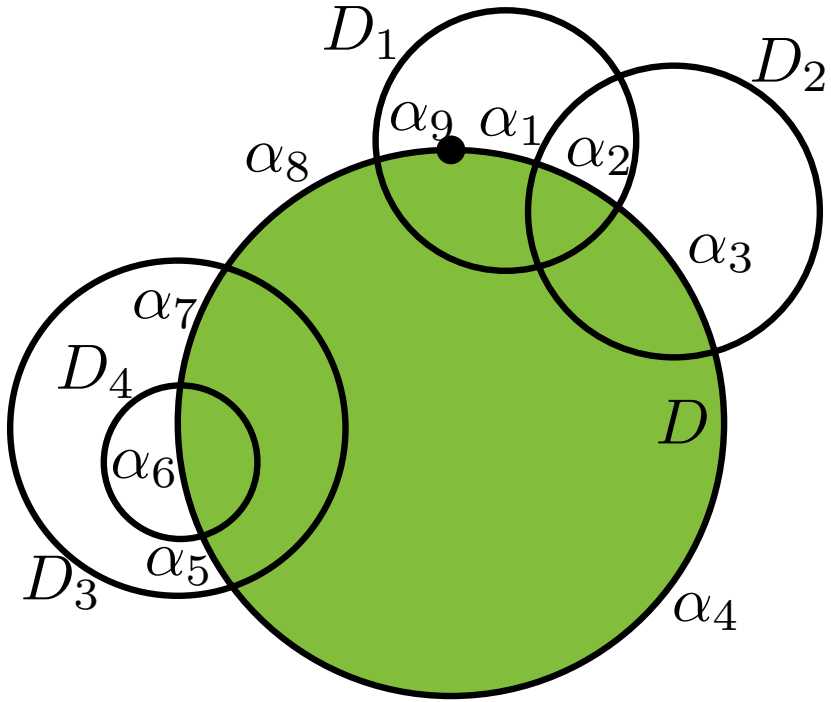
2. Verschiebe D_j vor D_i in $\mathcal{L}$. $\longrightarrow$ $vis_{\mathcal{L}''}(D_k) \geq vis_{\mathcal{L}}(D_k)$ für $k \neq j$

Zeige: $vis_{\mathcal{L}''}(D_j) \geq \min\limits_{1 \leq k \leq n} vis_{\mathcal{L}}(D_k)$

3. Nach Def.: $vis_{\mathcal{L}'}(D_j) \geq vis_{\mathcal{L}}(D_i)$ $\longrightarrow$ $vis_{\mathcal{L}''}(D_j) \geq vis_{\mathcal{L}}(D_i)$

Segment-Bäume für Kreisränder

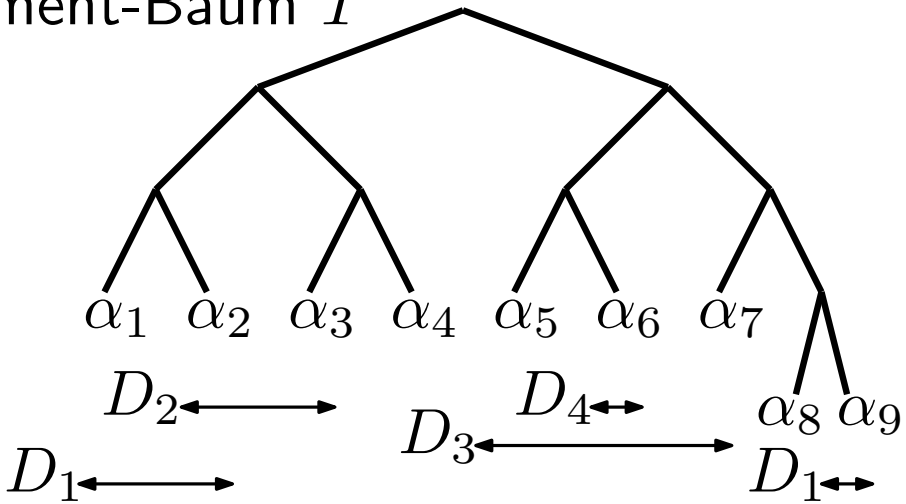
- betrachte Rand eines Kreises D von oben im UZS als lineares Intervall
- definiere Elementarintervalle aus Schnitten mit anderen Kreisen in $\mathcal{D}$
- jeder Kreis $D' \neq D$ schneidet D in 0, 1 oder 2 Intervallen

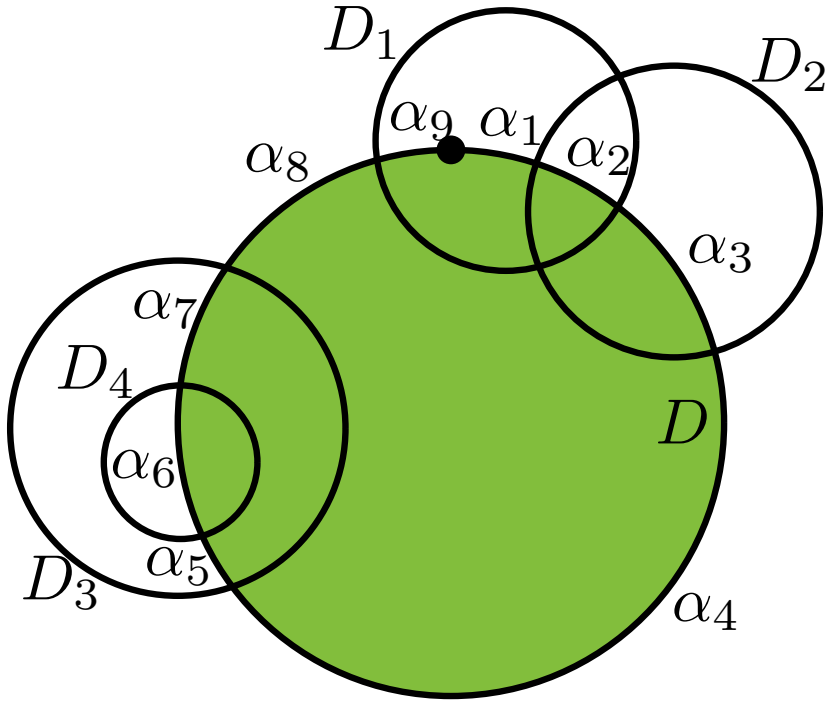


Segment-Bäume für Kreisränder

- betrachte Rand eines Kreises D von oben im UZS als lineares Intervall
- definiere Elementarintervalle aus Schnitten mit anderen Kreisen in $\mathcal{D}$
- jeder Kreis $D' \neq D$ schneidet D in 0, 1 oder 2 Intervallen

Segment-Baum T



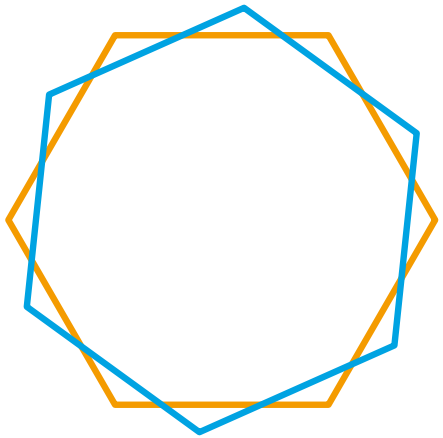


Aufgabe

Ist das Verfahren auch für konvexe Polygone möglich?

Aufgabe

Ist das Verfahren auch für konvexe Polygone möglich?

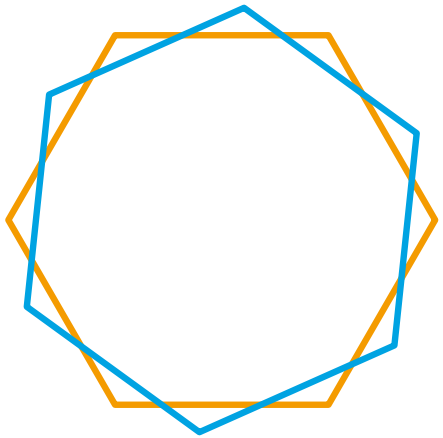


- Austauschargument bleibt erhalten.
- Unterteilung in Intervalle ist ebenfalls möglich.

Komplexität steigt entsprechend um $O(n)$, da $\Theta(n)$ viele Schnittpunkte möglich sind.

Aufgabe

Ist das Verfahren auch für konvexe Polygone möglich?



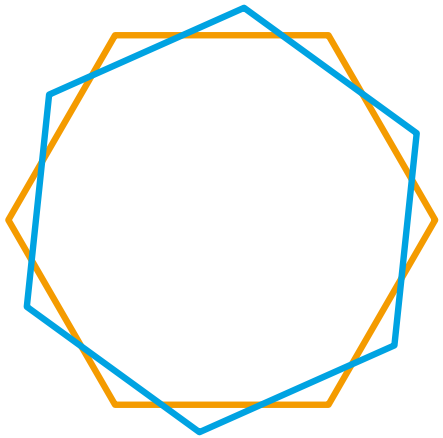
- Austauschargument bleibt erhalten.
- Unterteilung in Intervalle ist ebenfalls möglich.

Komplexität steigt entsprechend um $O(n)$, da $\Theta(n)$ viele Schnittpunkte möglich sind.

Mögliche Verbesserung: Betrache *Pseudo-Disks*: Polygone die maximal $O(1)$ viele Schnittpunkte haben dürfen.

Aufgabe

Ist das Verfahren auch für konvexe Polygone möglich?



- Austauschargument bleibt erhalten.
- Unterteilung in Intervalle ist ebenfalls möglich.

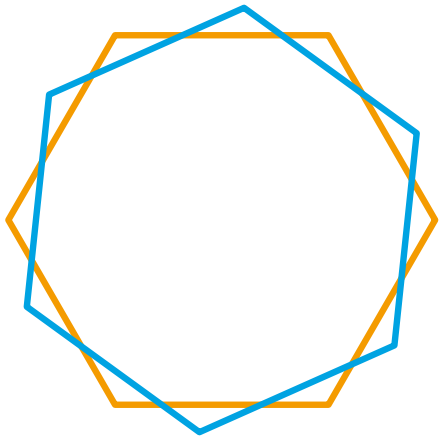
Komplexität steigt entsprechend um $O(n)$, da $\Theta(n)$ viele Schnittpunkte möglich sind.

Mögliche Verbesserung: Betrache *Pseudo-Disks*: Polygone die maximal $O(1)$ viele Schnittpunkte haben dürfen.

Ist das Verfahren auch für allgemeine Polyone möglich?

Aufgabe

Ist das Verfahren auch für konvexe Polygone möglich?

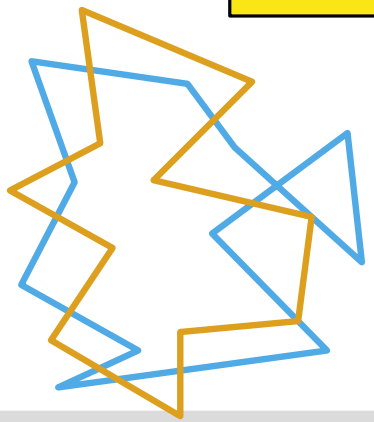


- Austauschargument bleibt erhalten.
- Unterteilung in Intervalle ist ebenfalls möglich.

Komplexität steigt entsprechend um $O(n)$, da $\Theta(n)$ viele Schnittpunkte möglich sind.

Mögliche Verbesserung: Betrache *Pseudo-Disks*: Polygone die maximal $O(1)$ viele Schnittpunkte haben dürfen.

Ist das Verfahren auch für allgemeine Polyone möglich?



- Austauschargument bleibt erhalten.
- Unterteilung in Intervalle ist ebenfalls möglich.

Komplexität steigt entsprechend um $O(n^2)$, da $\Theta(n^2)$ viele Schnittpunkte möglich sind.

Aufgabe

Was wenn minimal sichtbare Fläche maximiert werden soll?

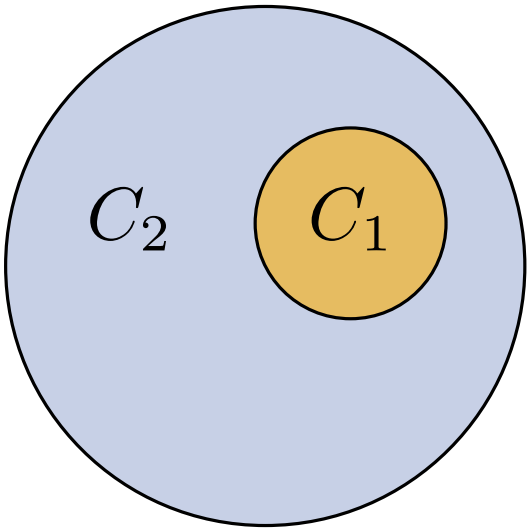
Aufgabe

Was wenn minimal sichtbare Fläche maximiert werden soll?

- Austauschargument bleibt erhalten.
- Unterteilung in Intervalle ist nicht mehr möglich.

→ naive Variante ist machbar.

→ Beschleunigung nicht.



C_1 beeinflusst sichtbare Fläche von C_2, indzuiert allerdings keine Intervall auf dem Kreisrand von C_1.