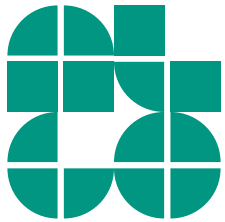
Vorlesung Algorithmische Kartografie

Beschriftung in dynamischen Karten: Rotieren

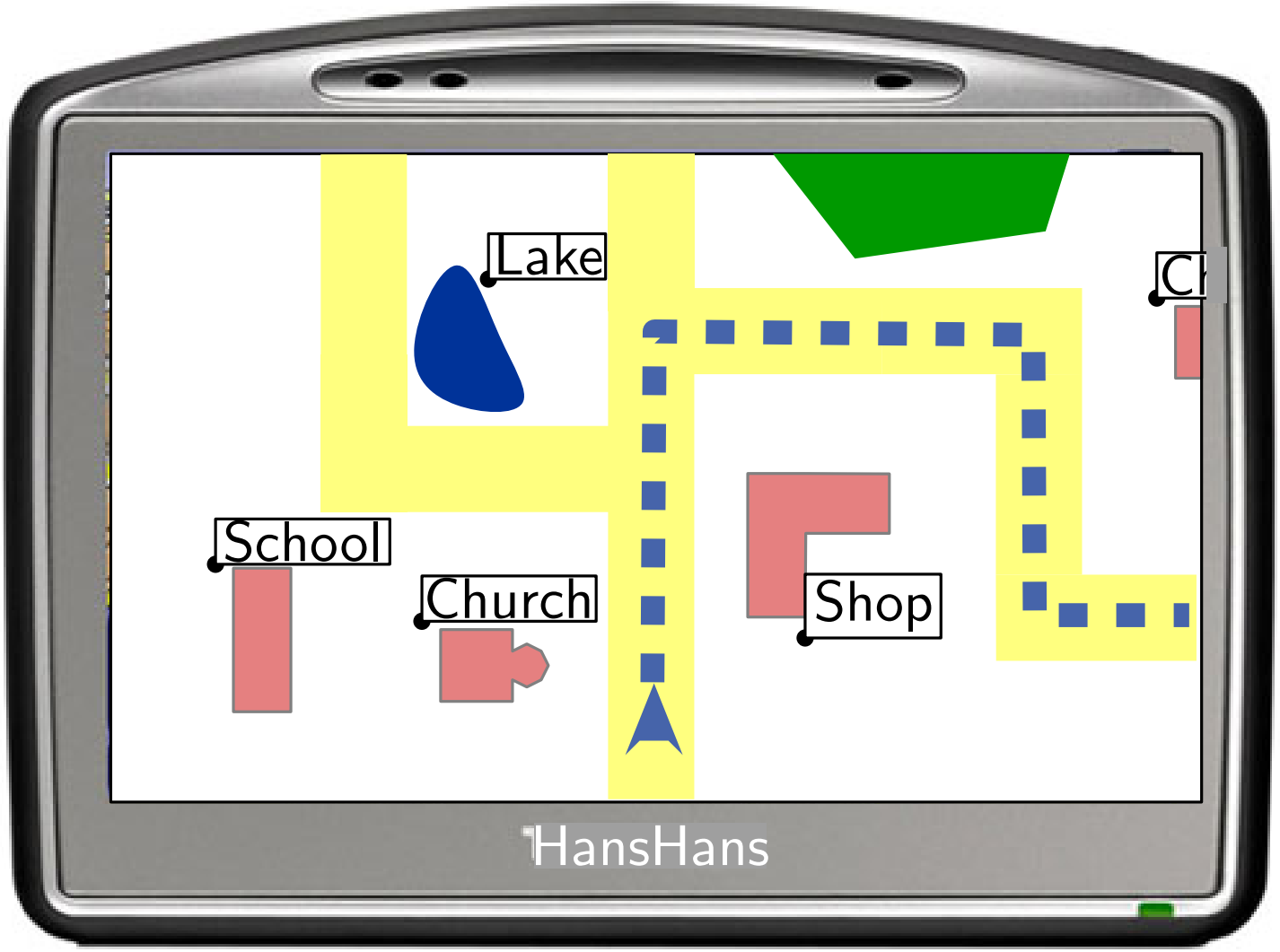
LEHRSTUHL FÜR ALGORITHMIK I · INSTITUT FÜR THEORETISCHE INFORMATIK · FAKULTÄT FÜR INFORMATIK

Andreas Gemsa

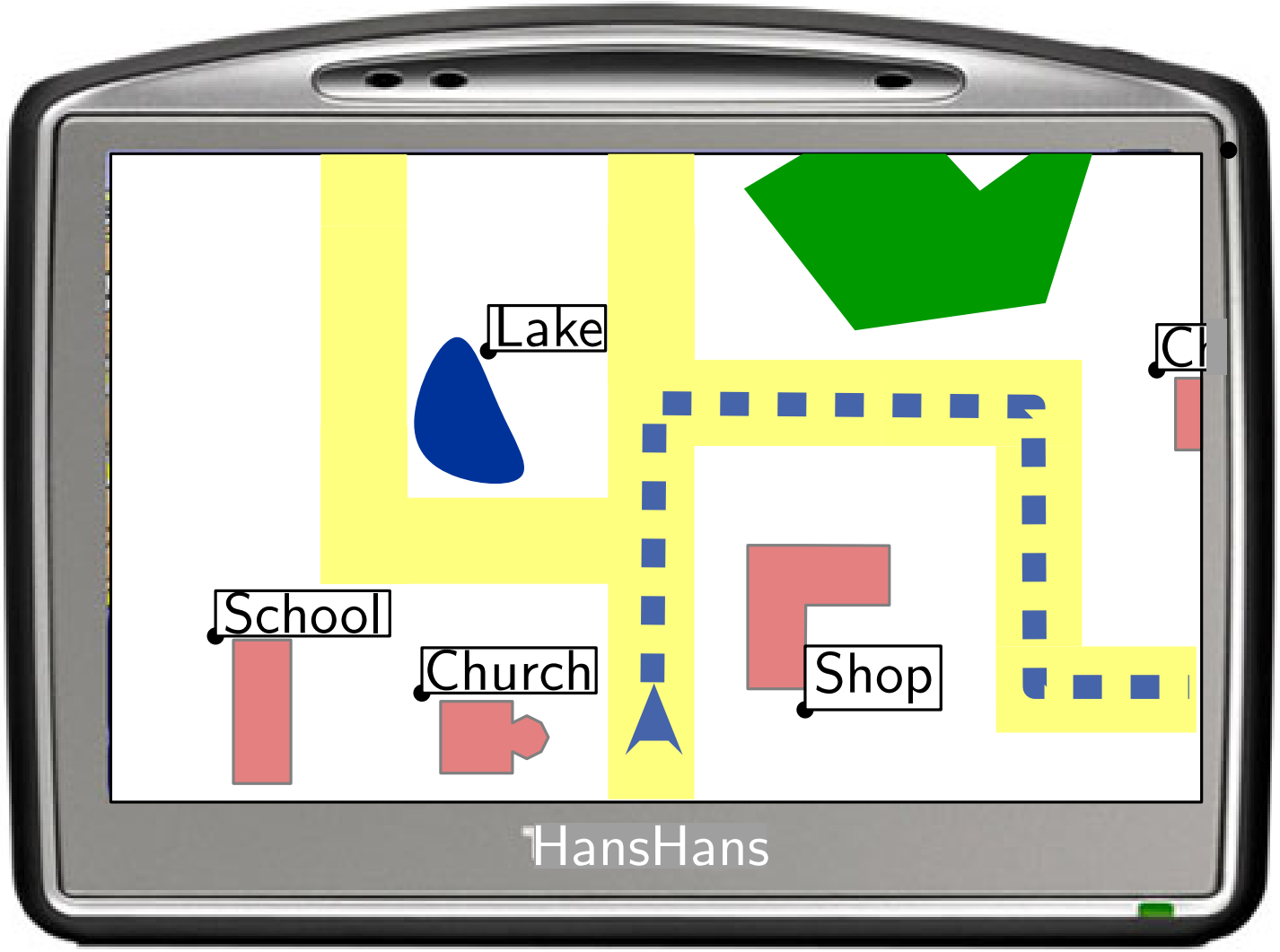
18.06.2013



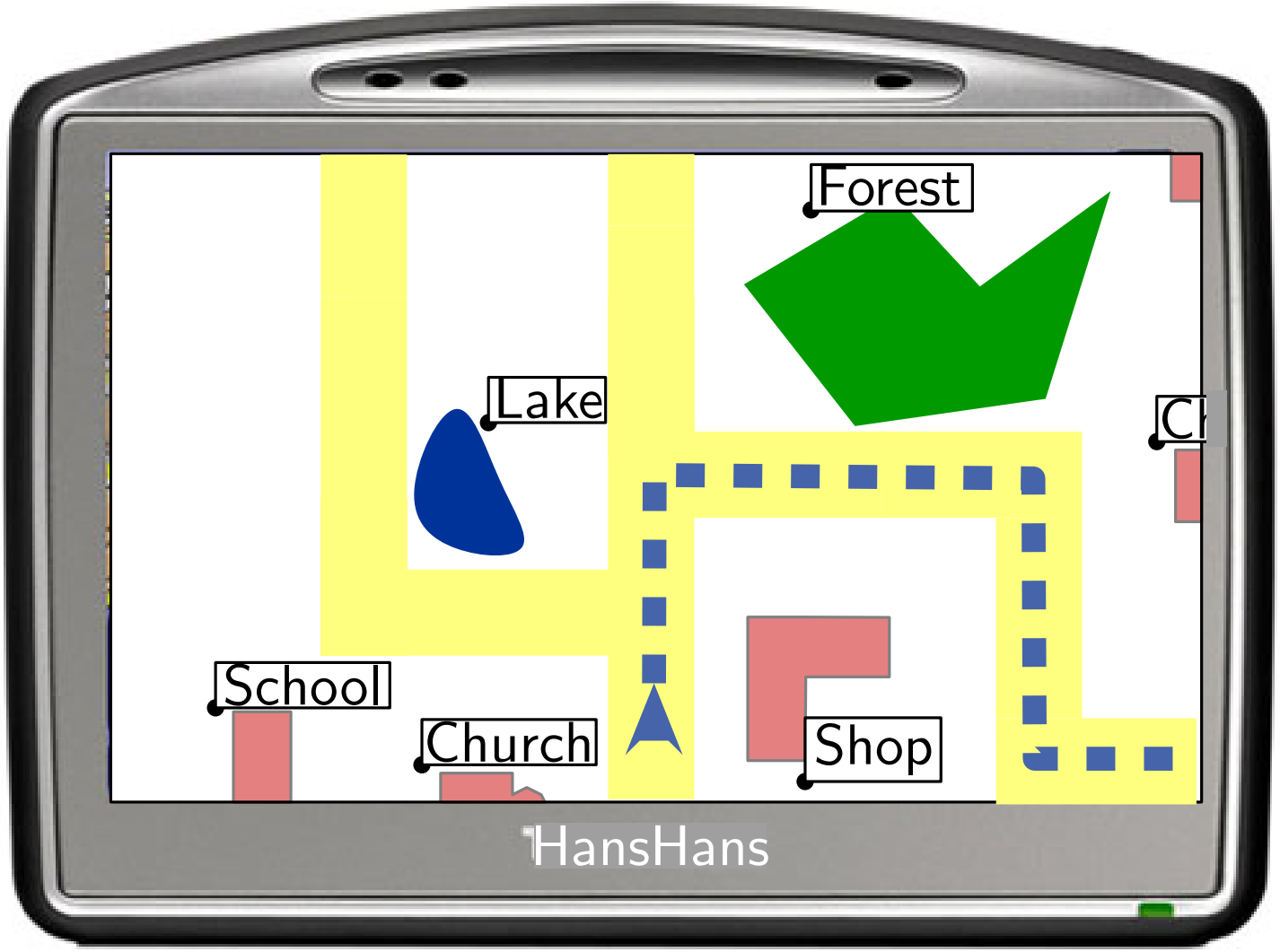
Motivation



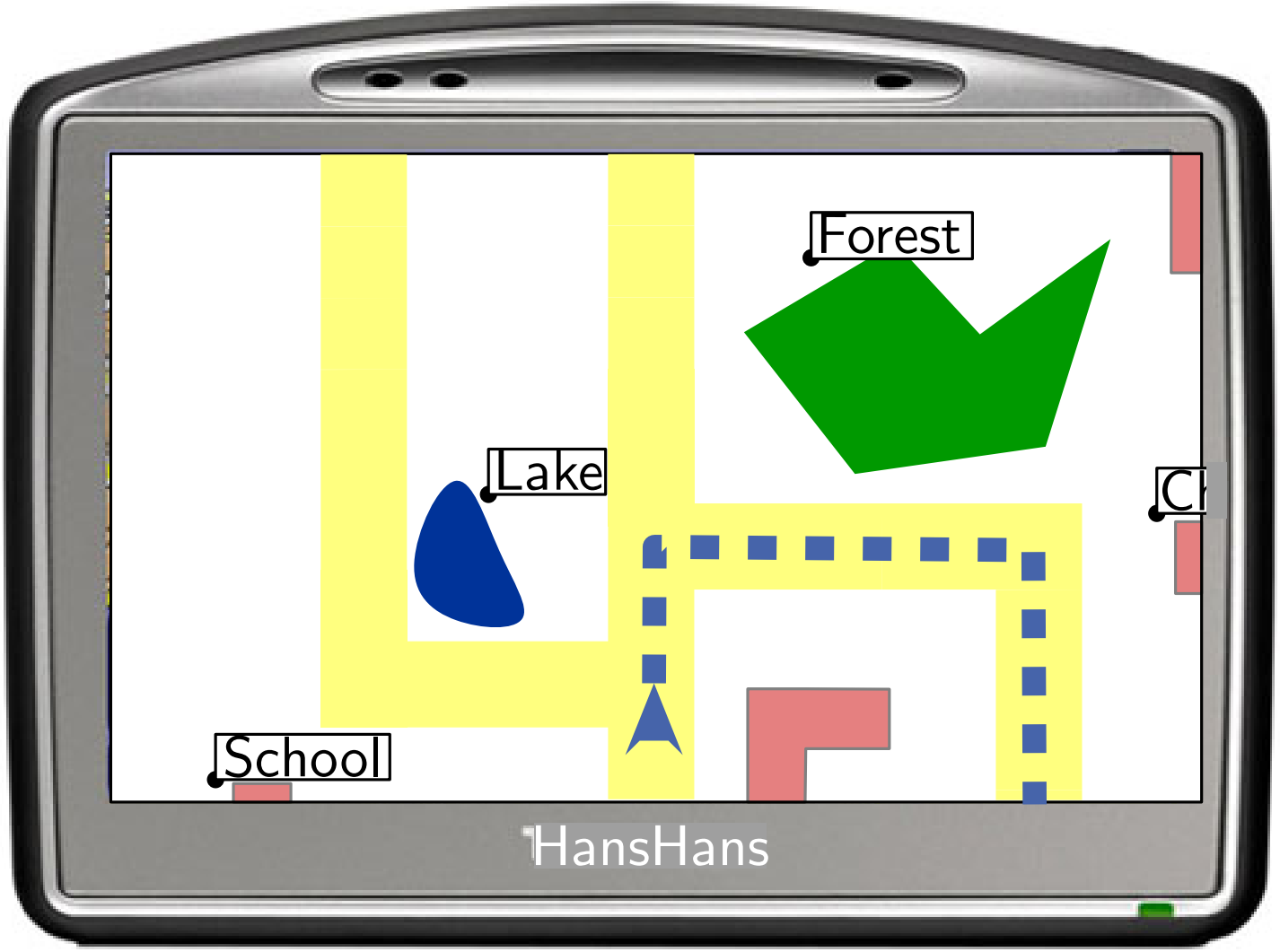
Motivation



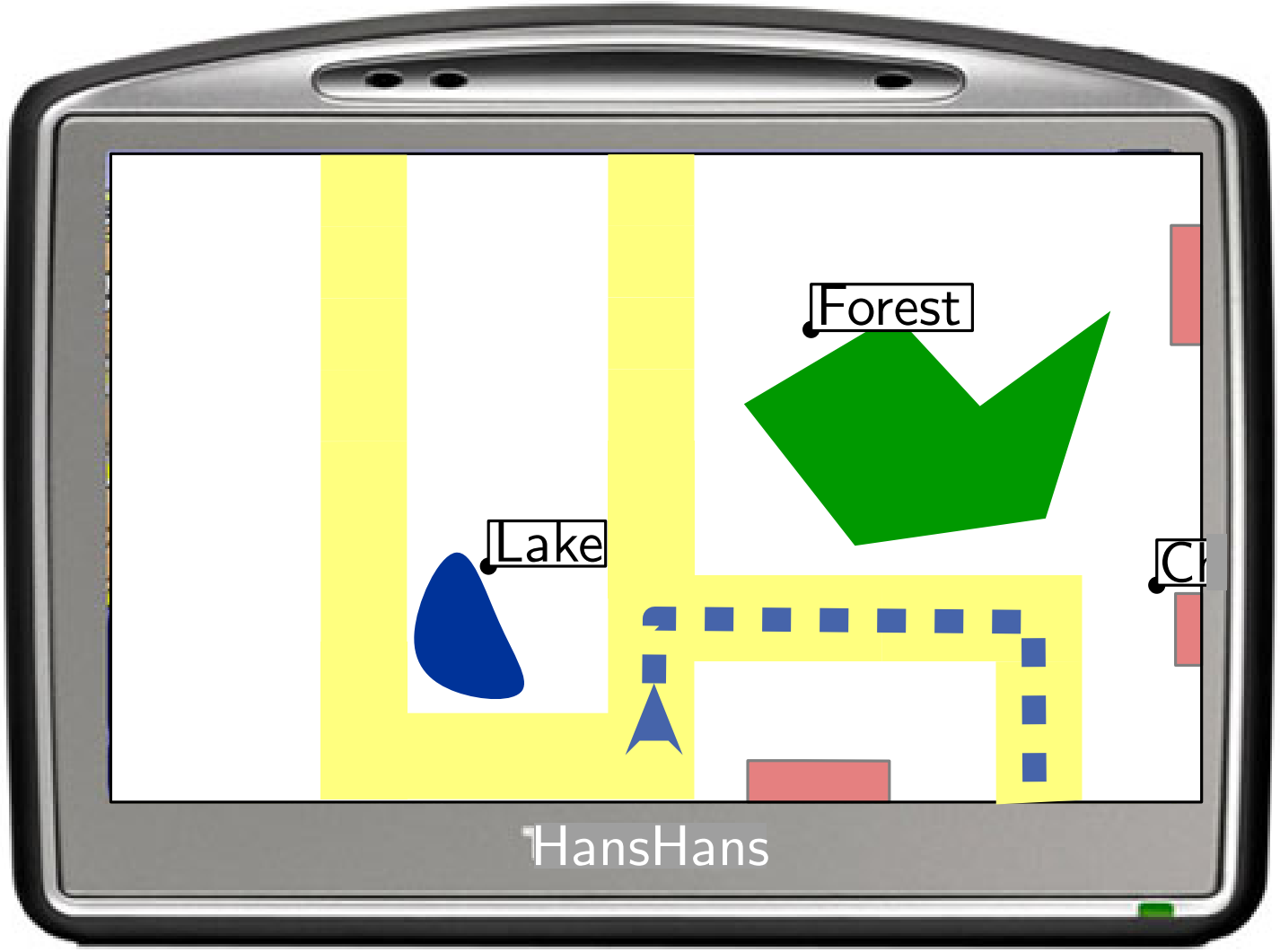
Motivation



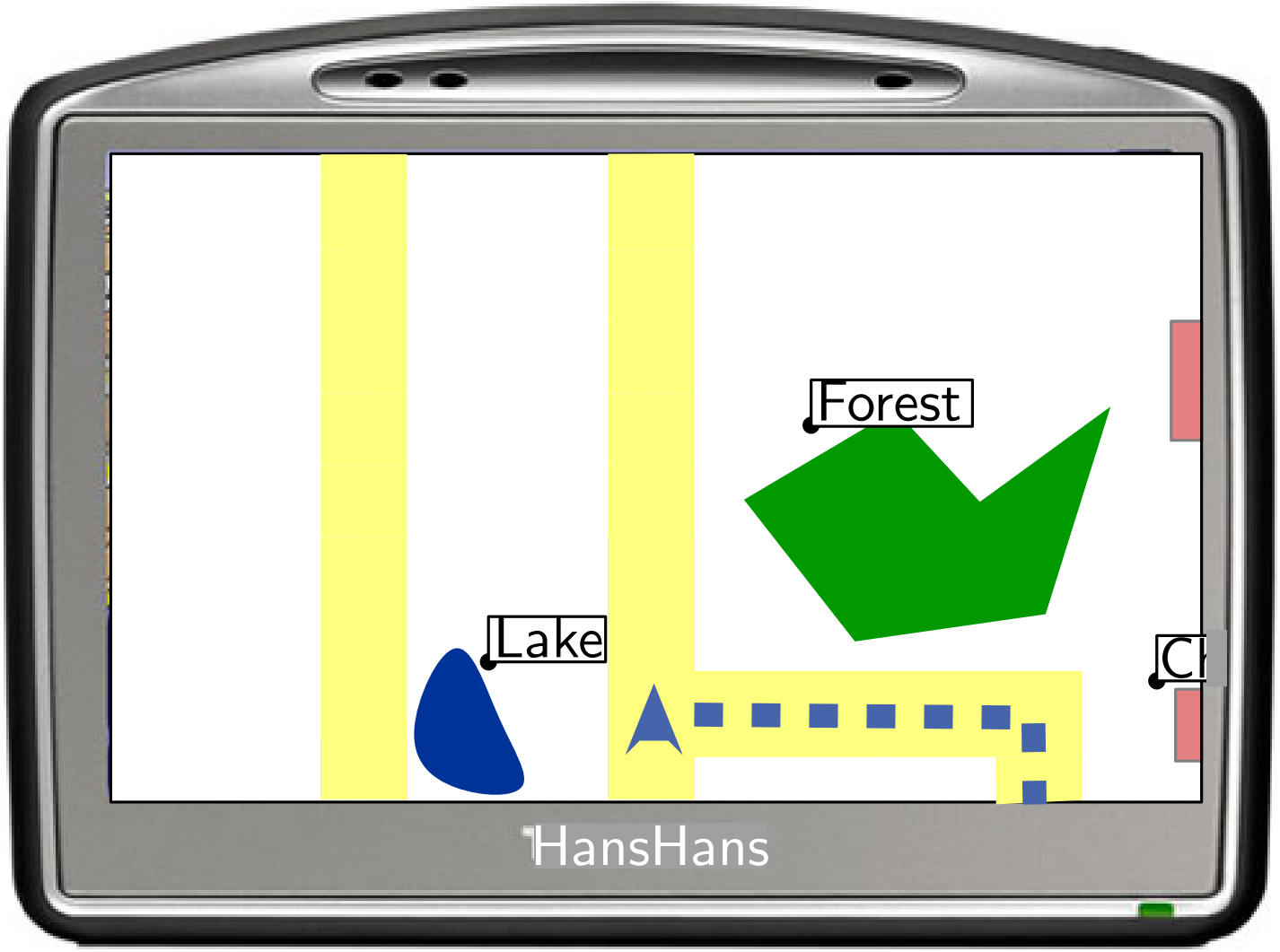
Motivation



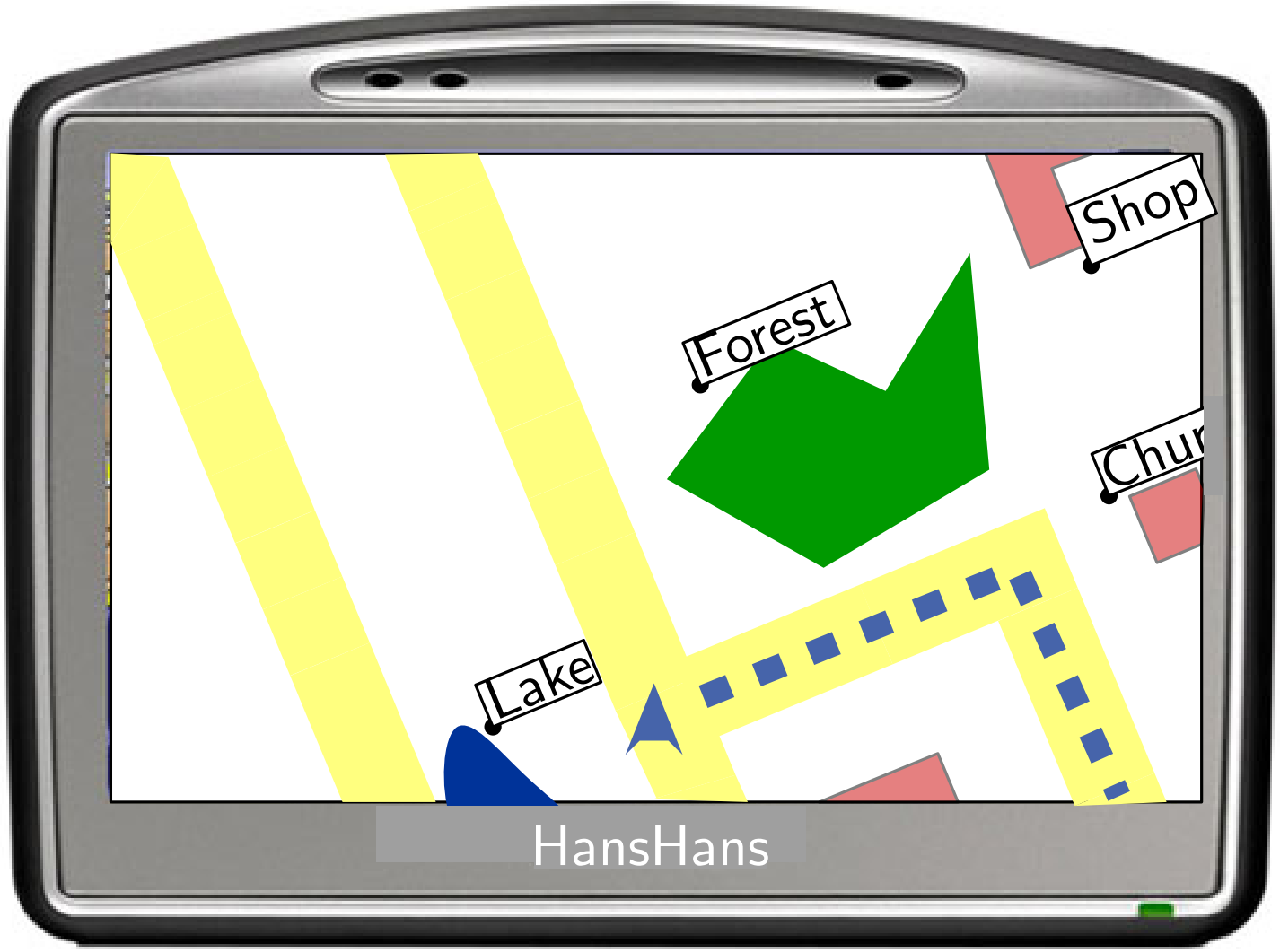
Motivation



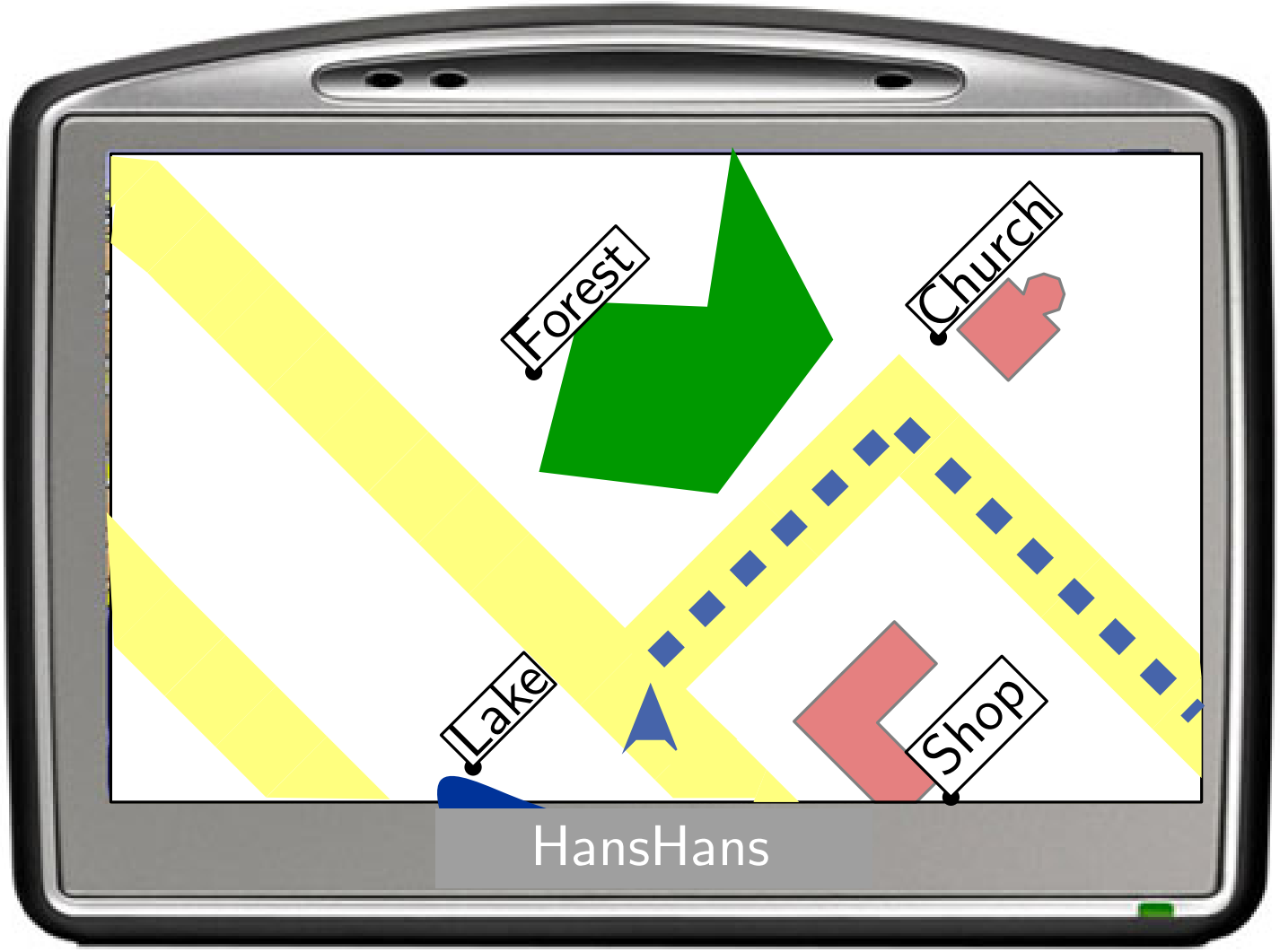
Motivation



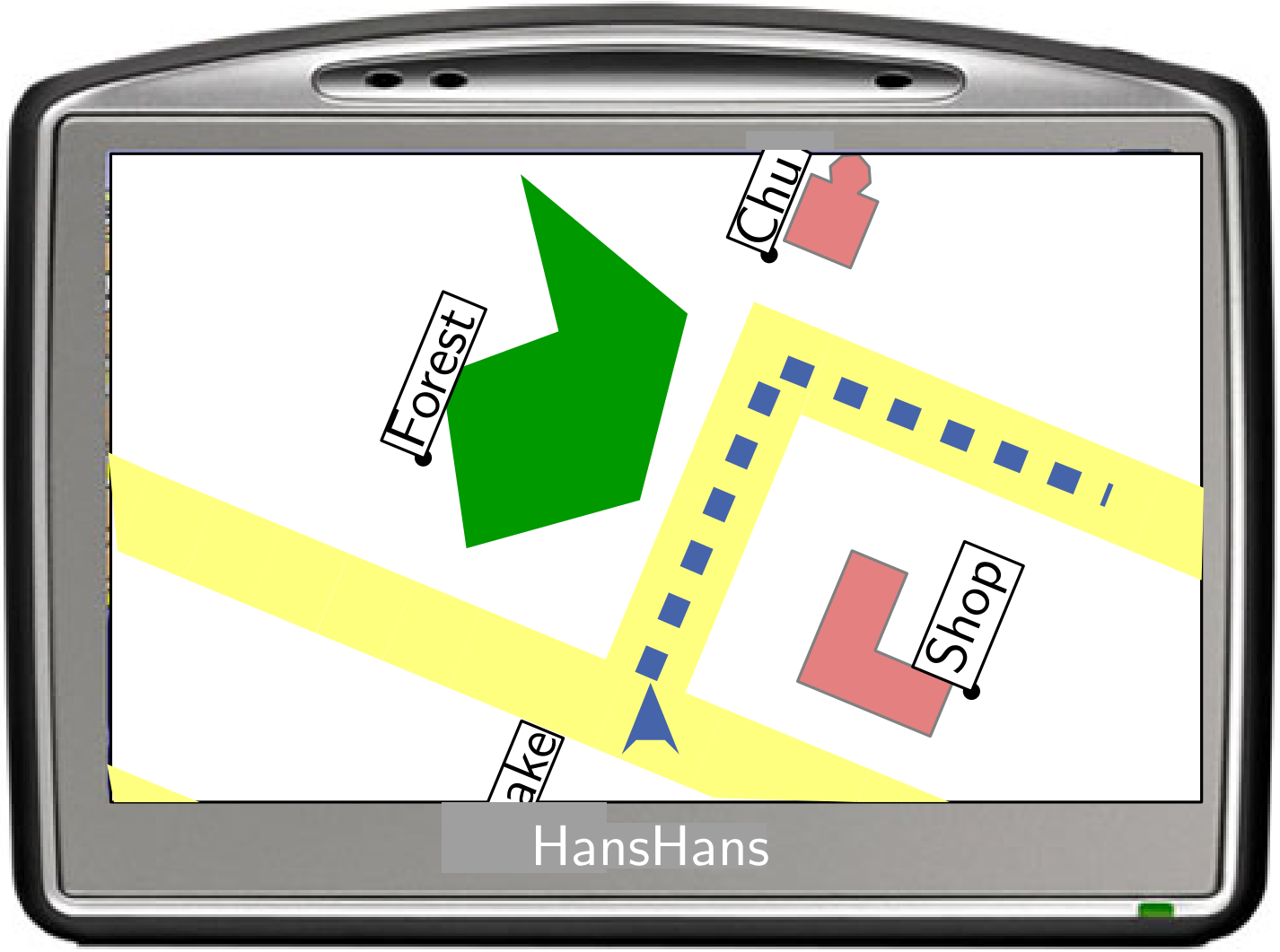
Motivation



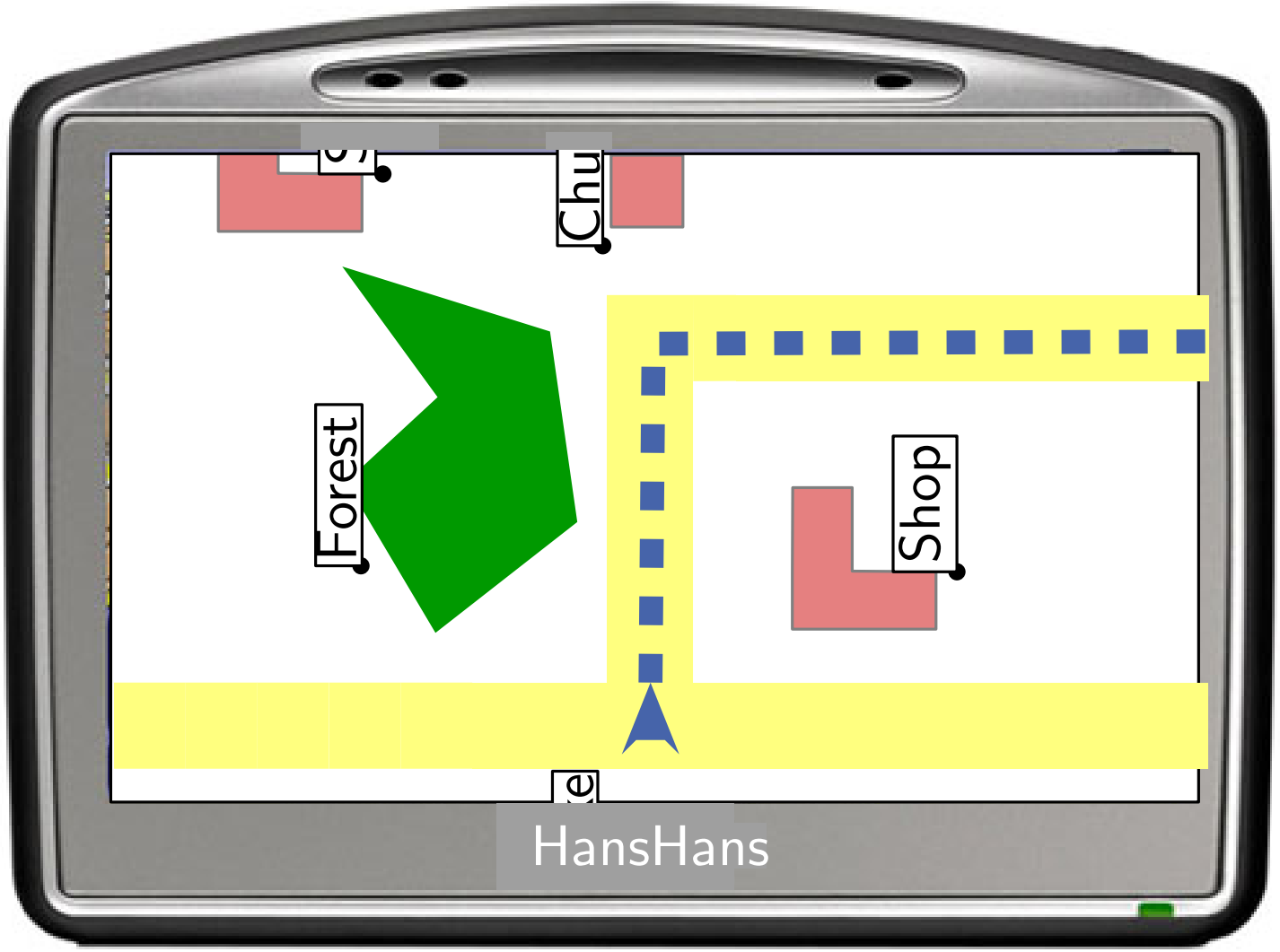
Motivation



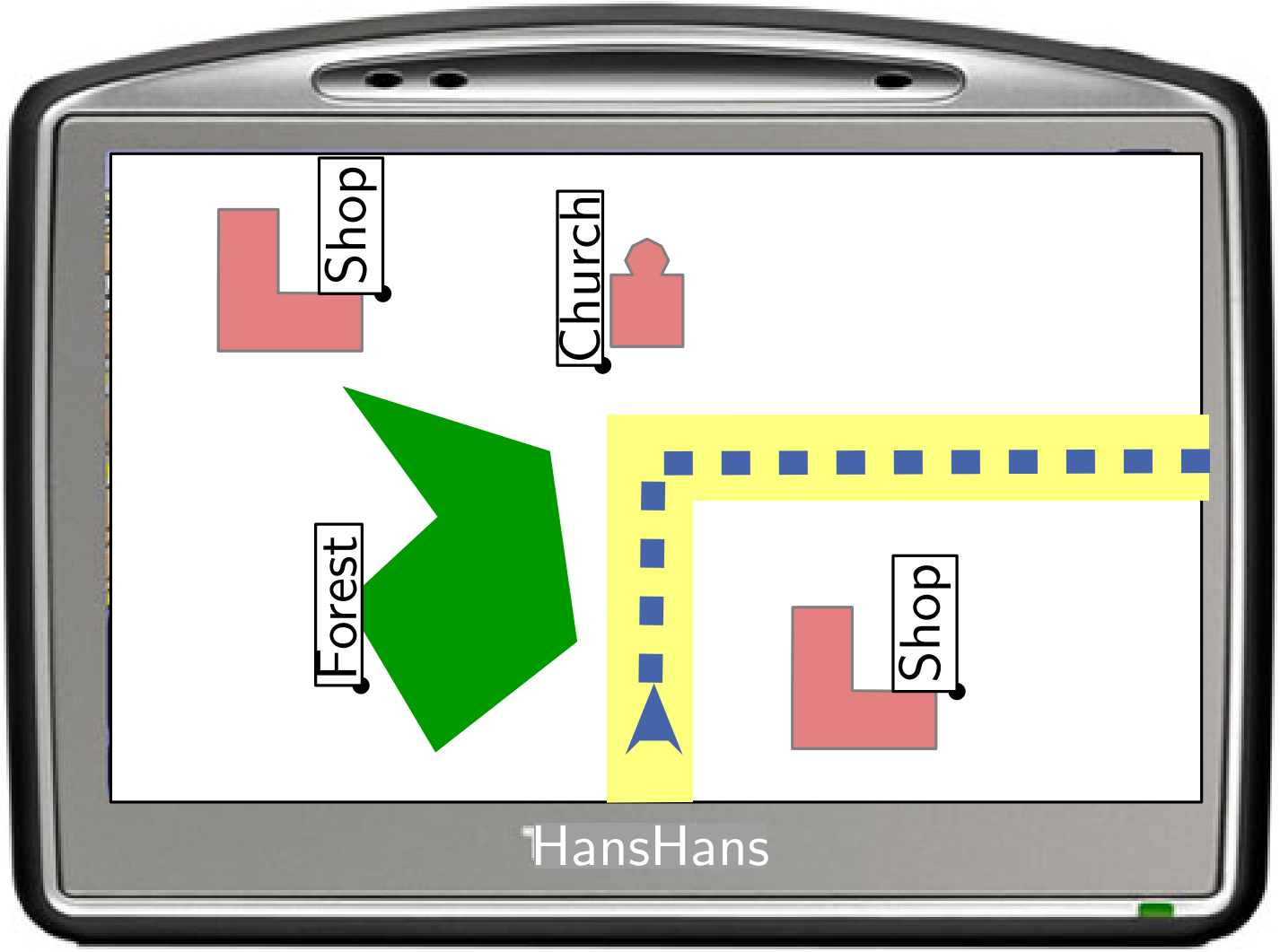
Motivation



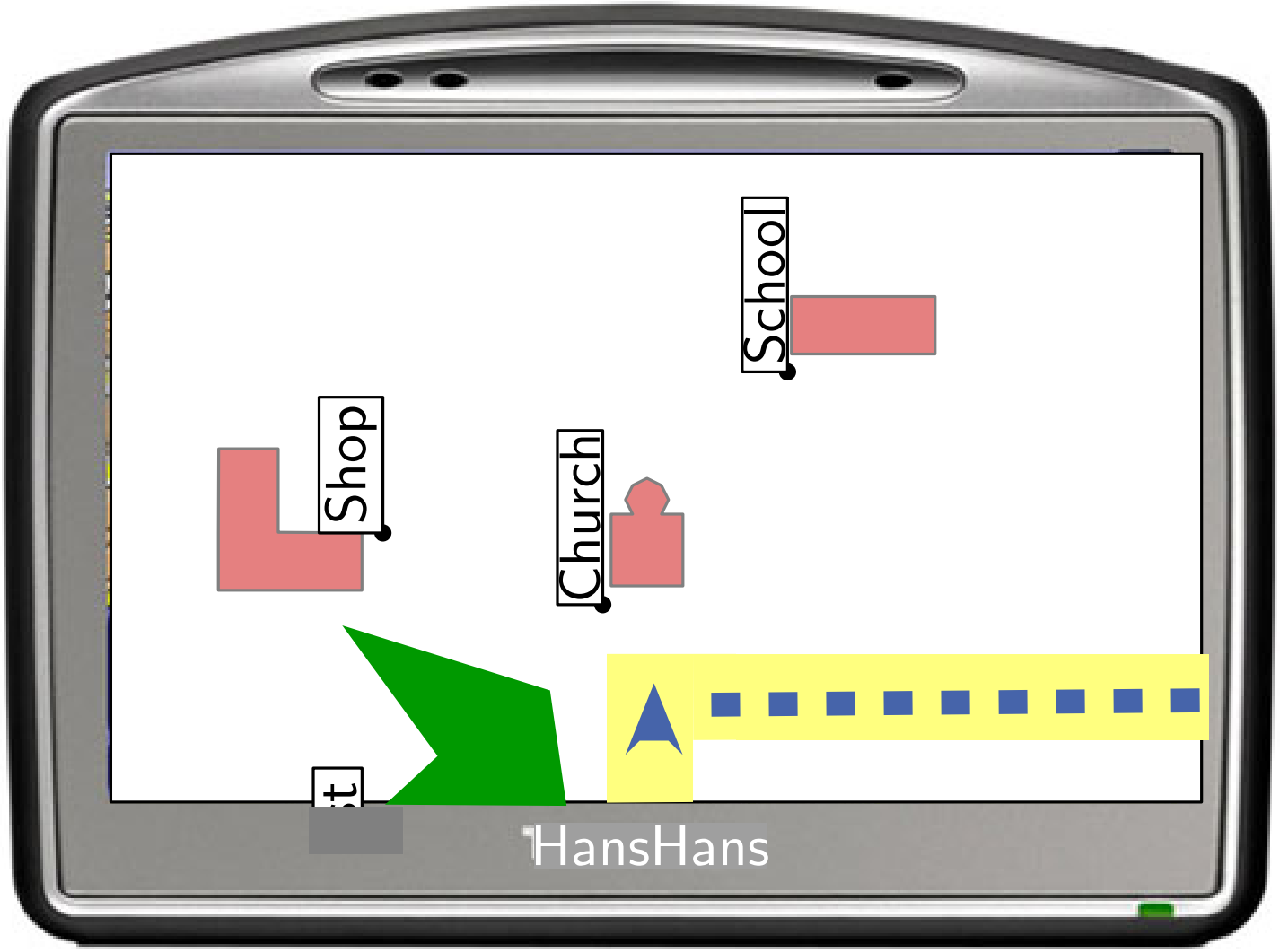
Motivation



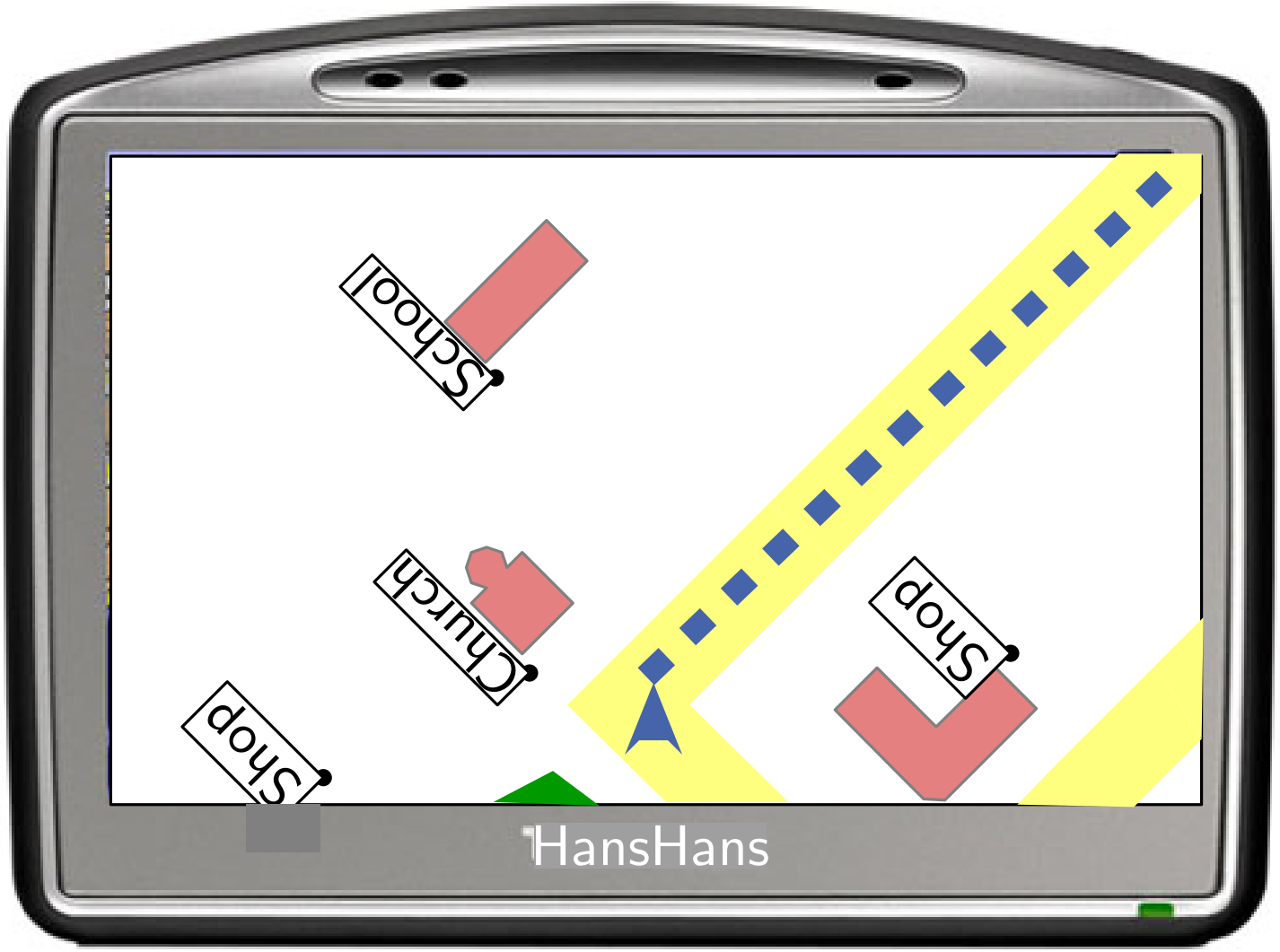
Motivation



Motivation



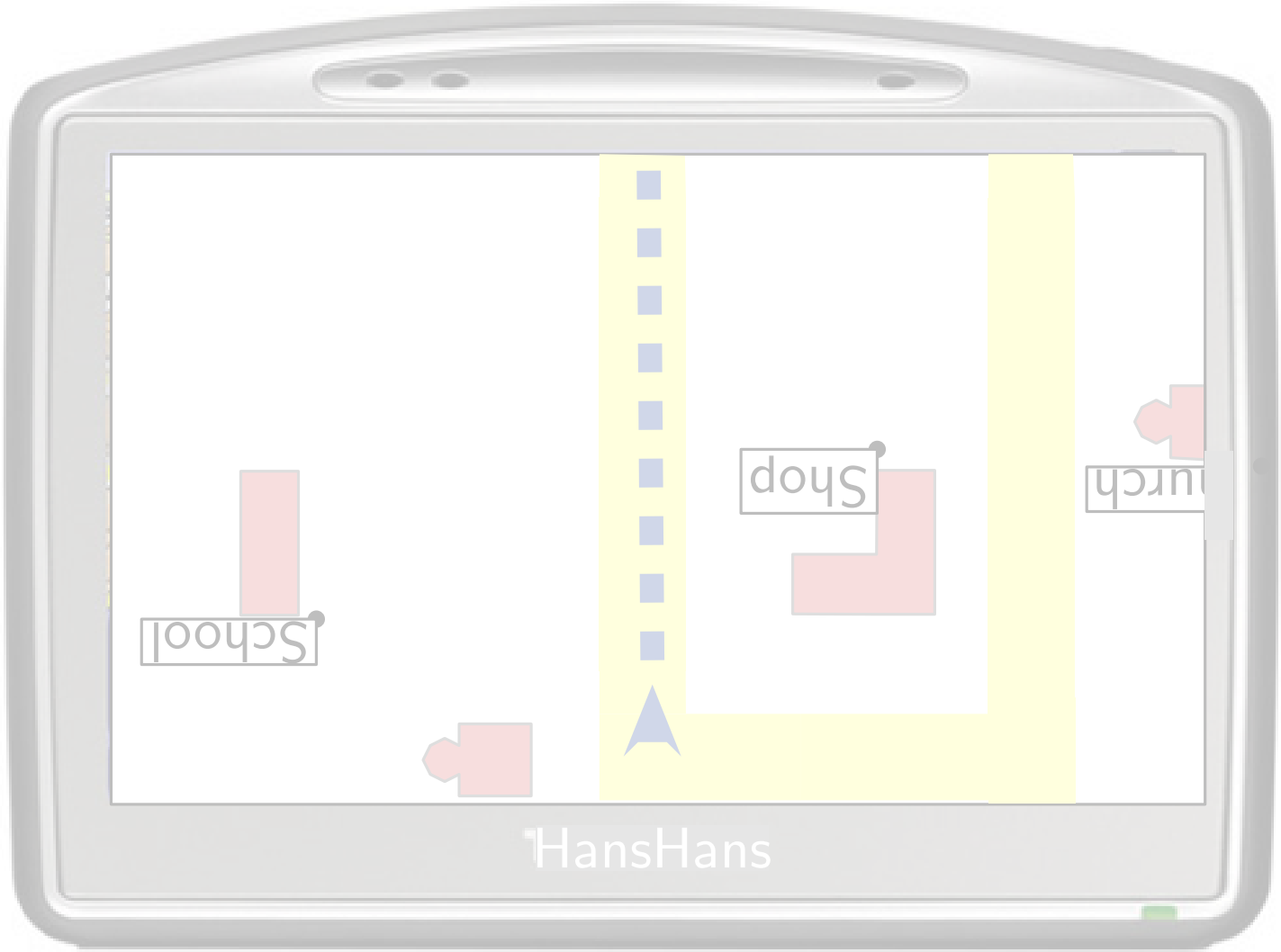
Motivation



Motivation

Motivation

Motivation



Motivation

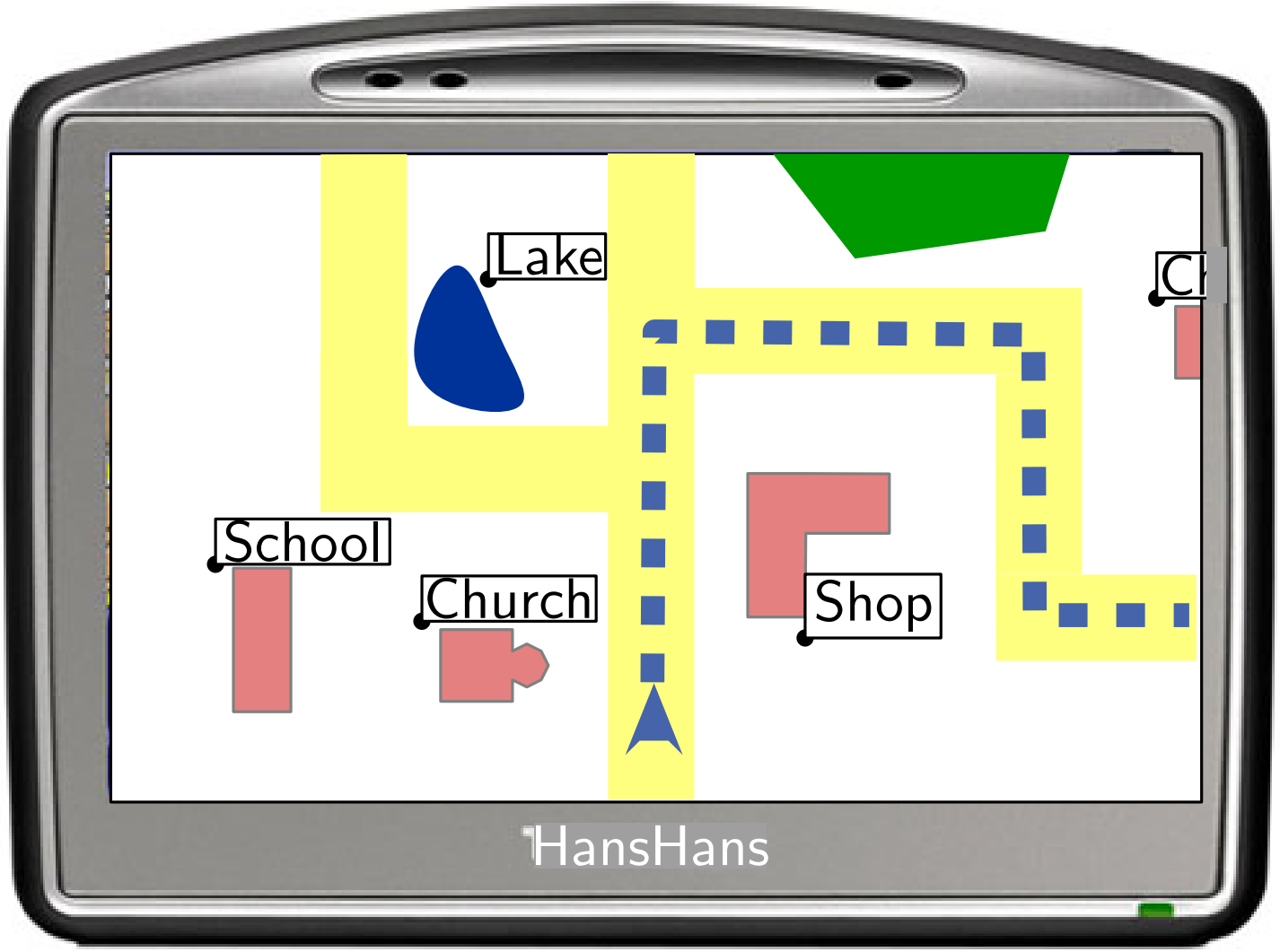
Motivation

Motivation

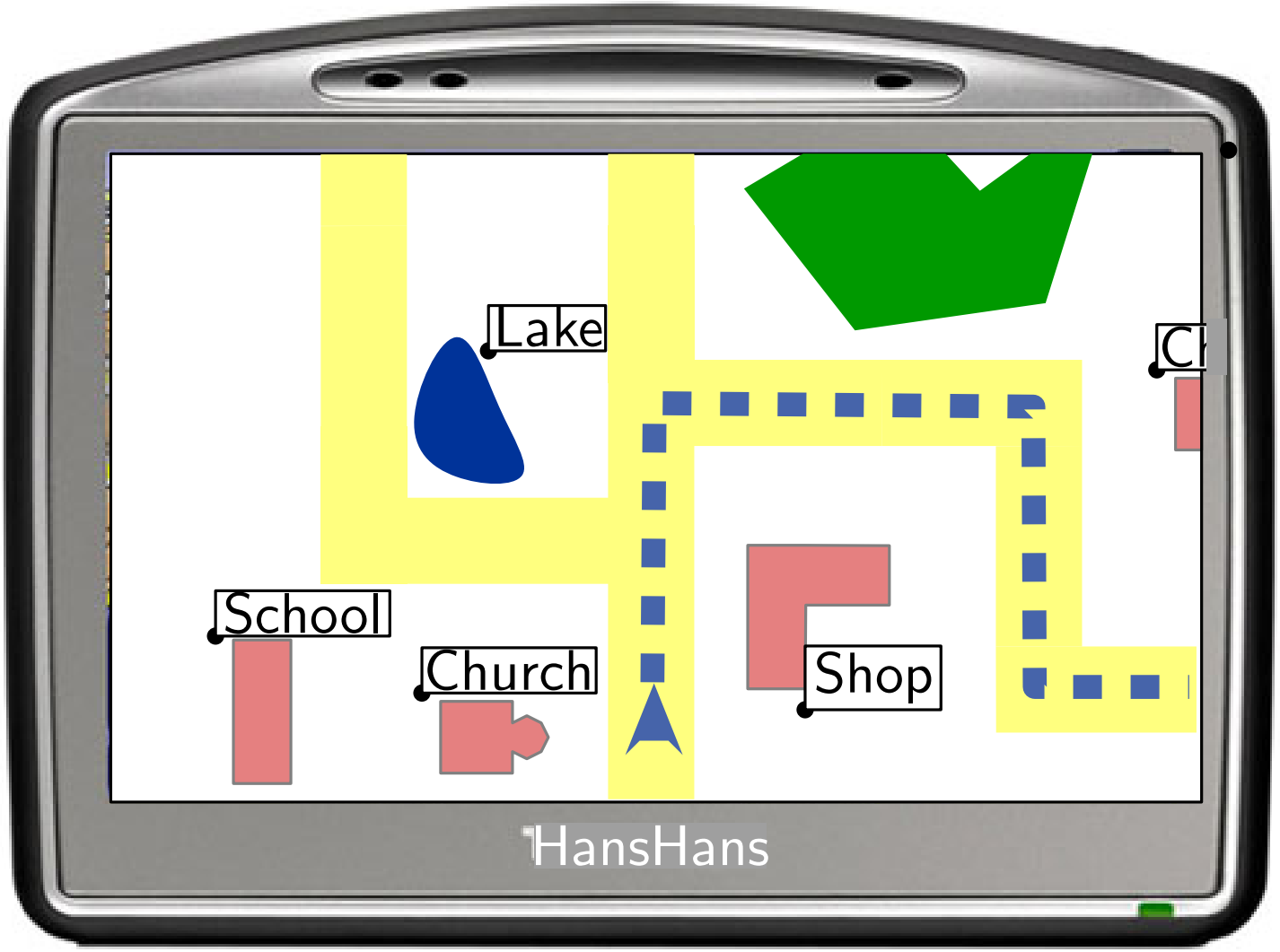
Motivation

Motivation

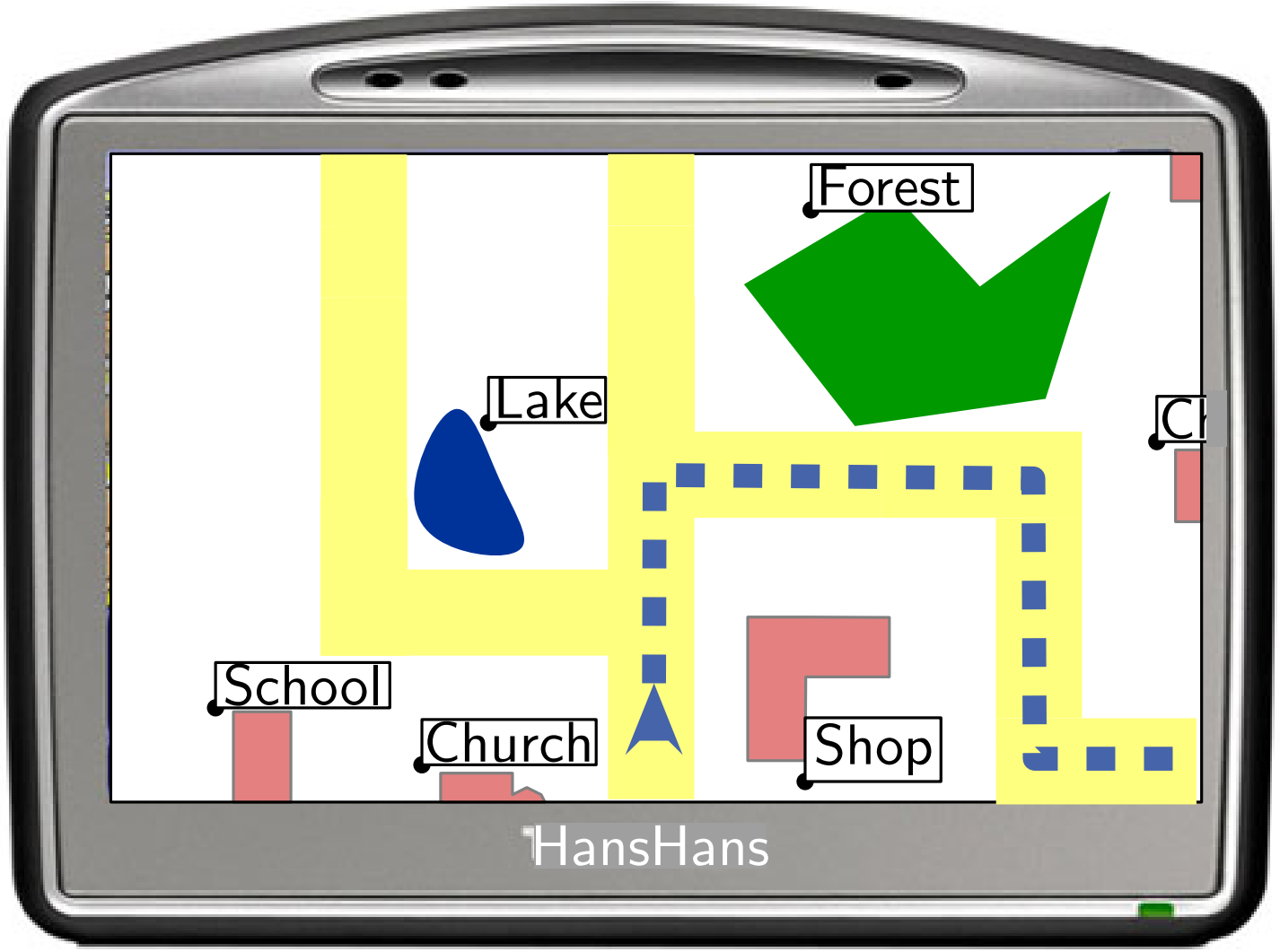
Motivation



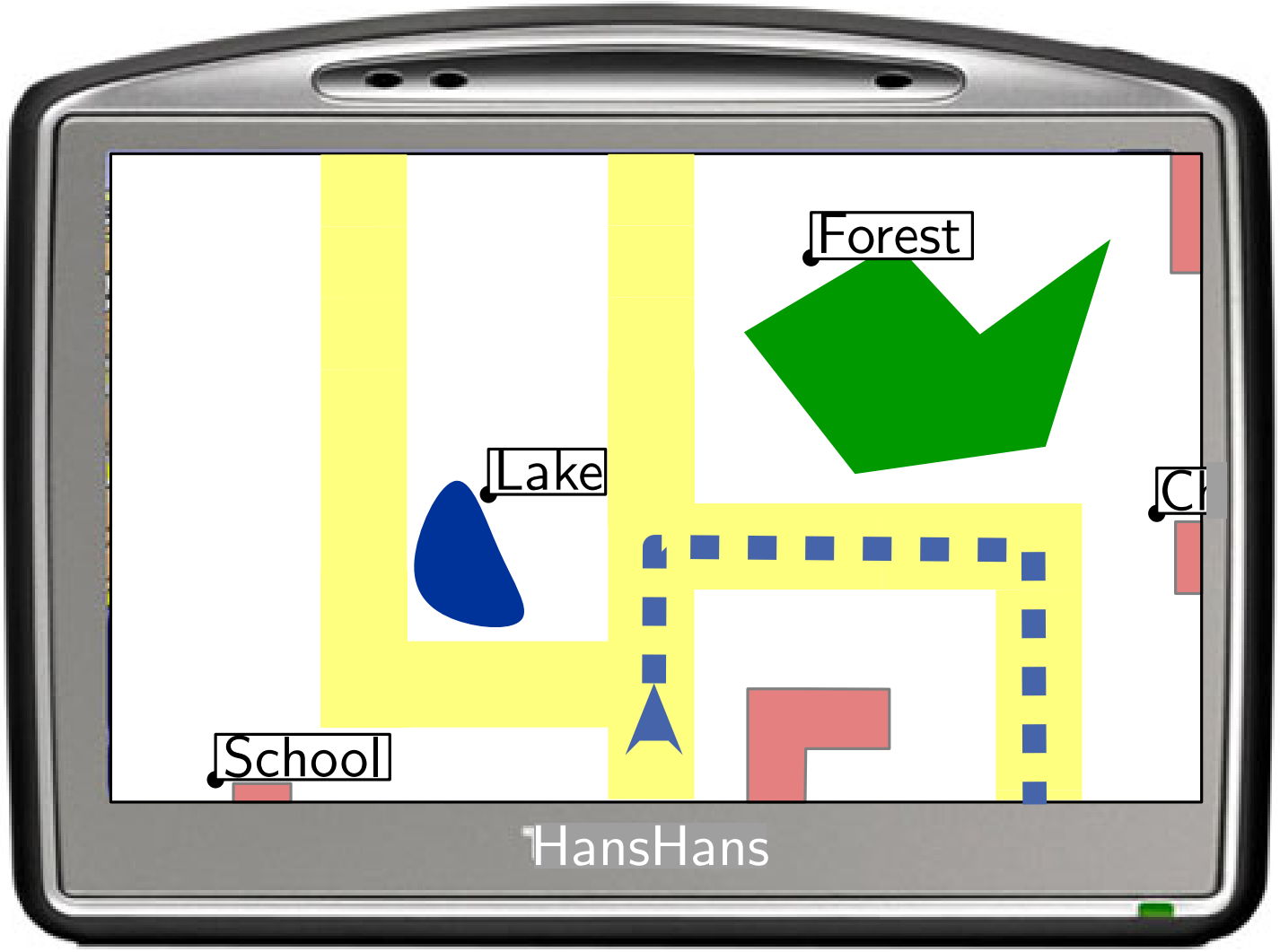
Motivation



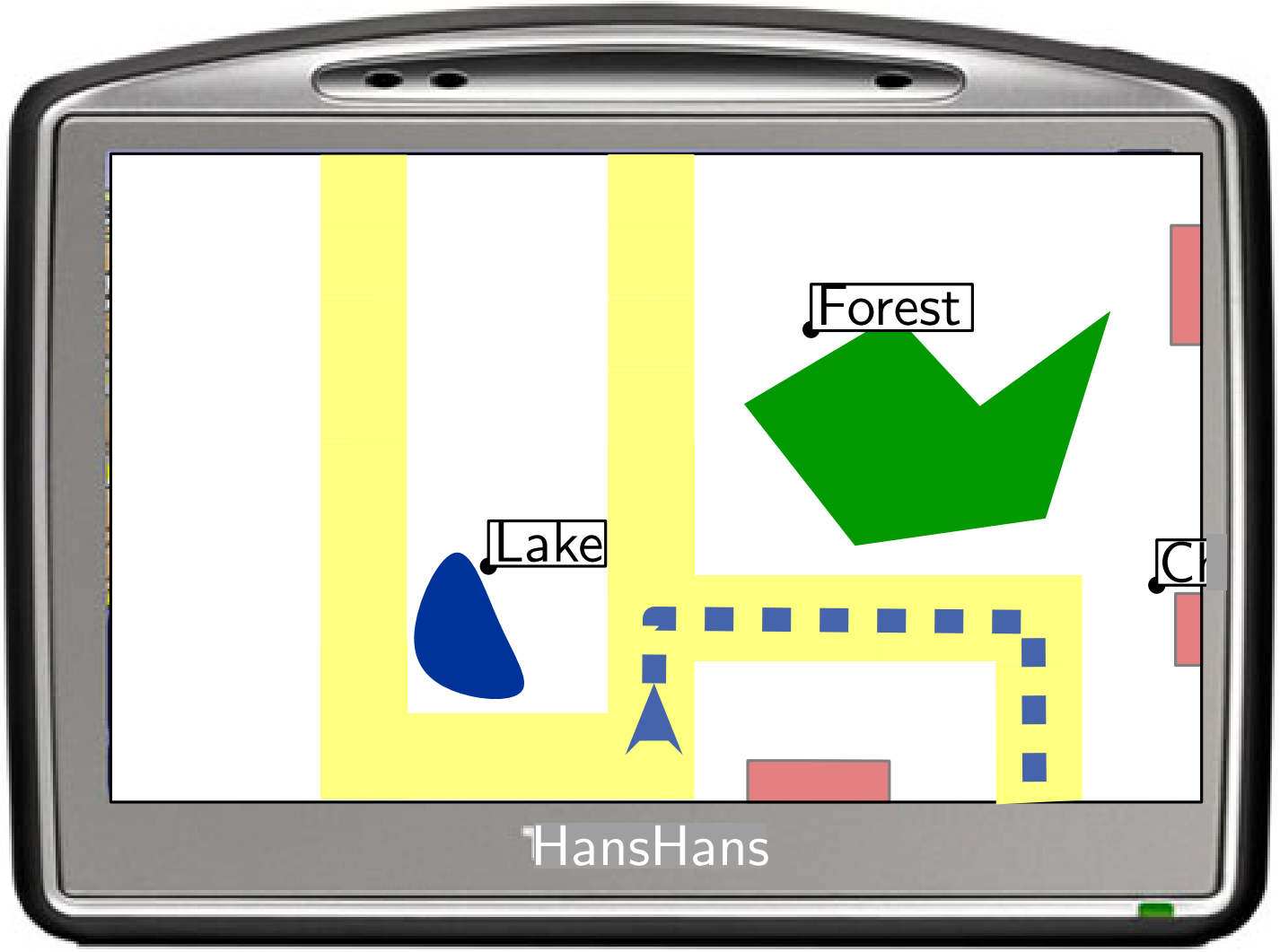
Motivation



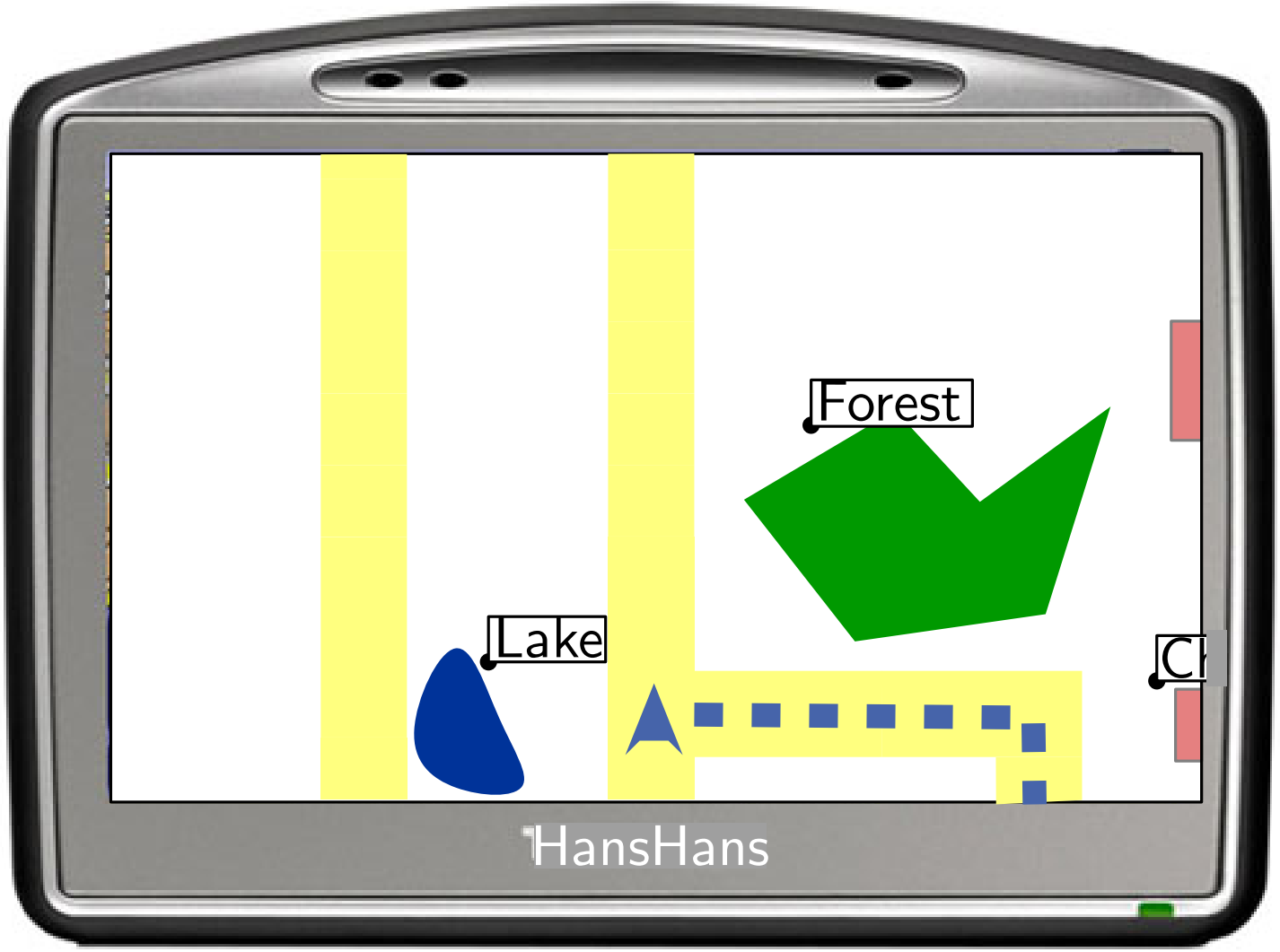
Motivation



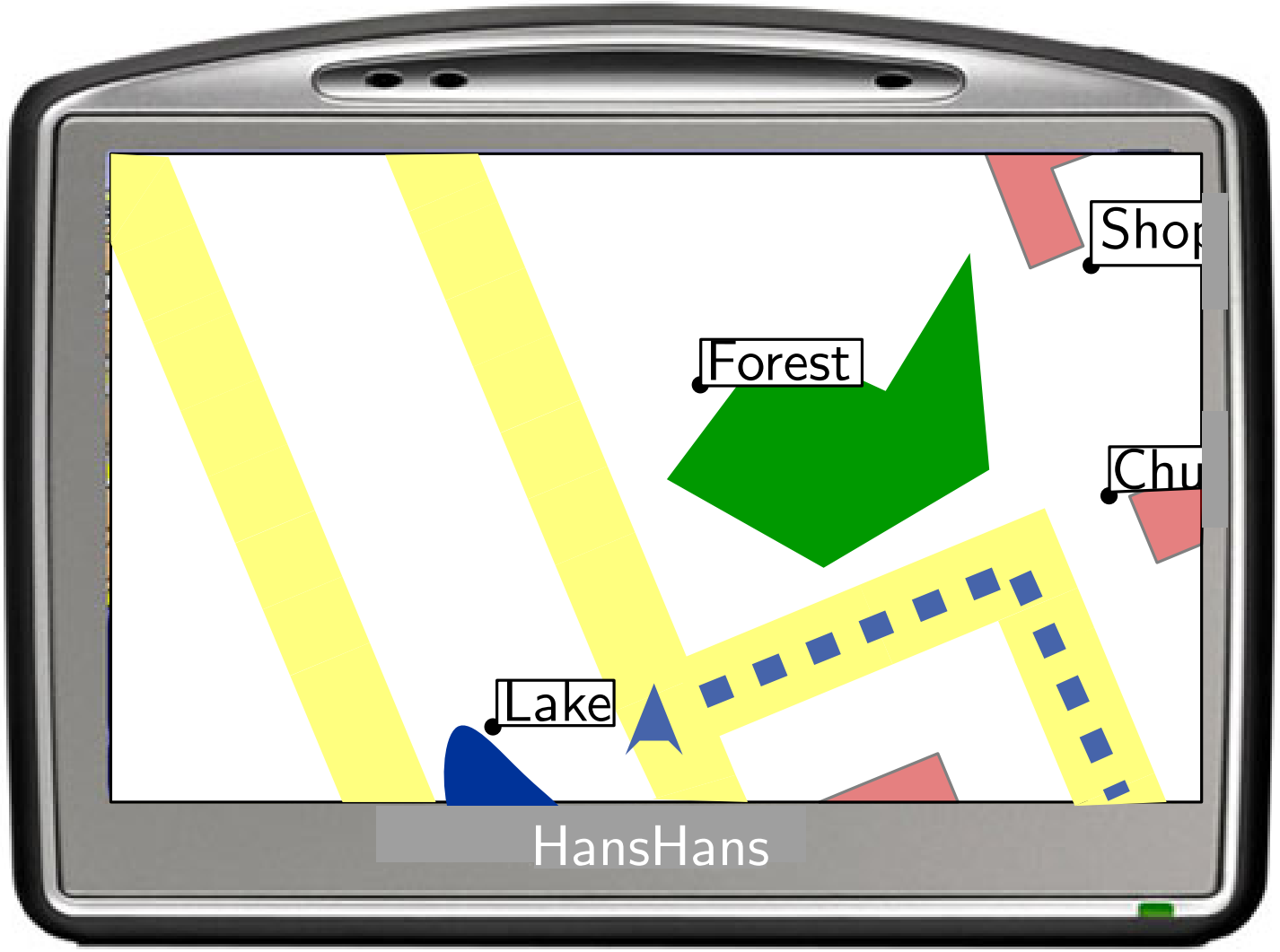
Motivation



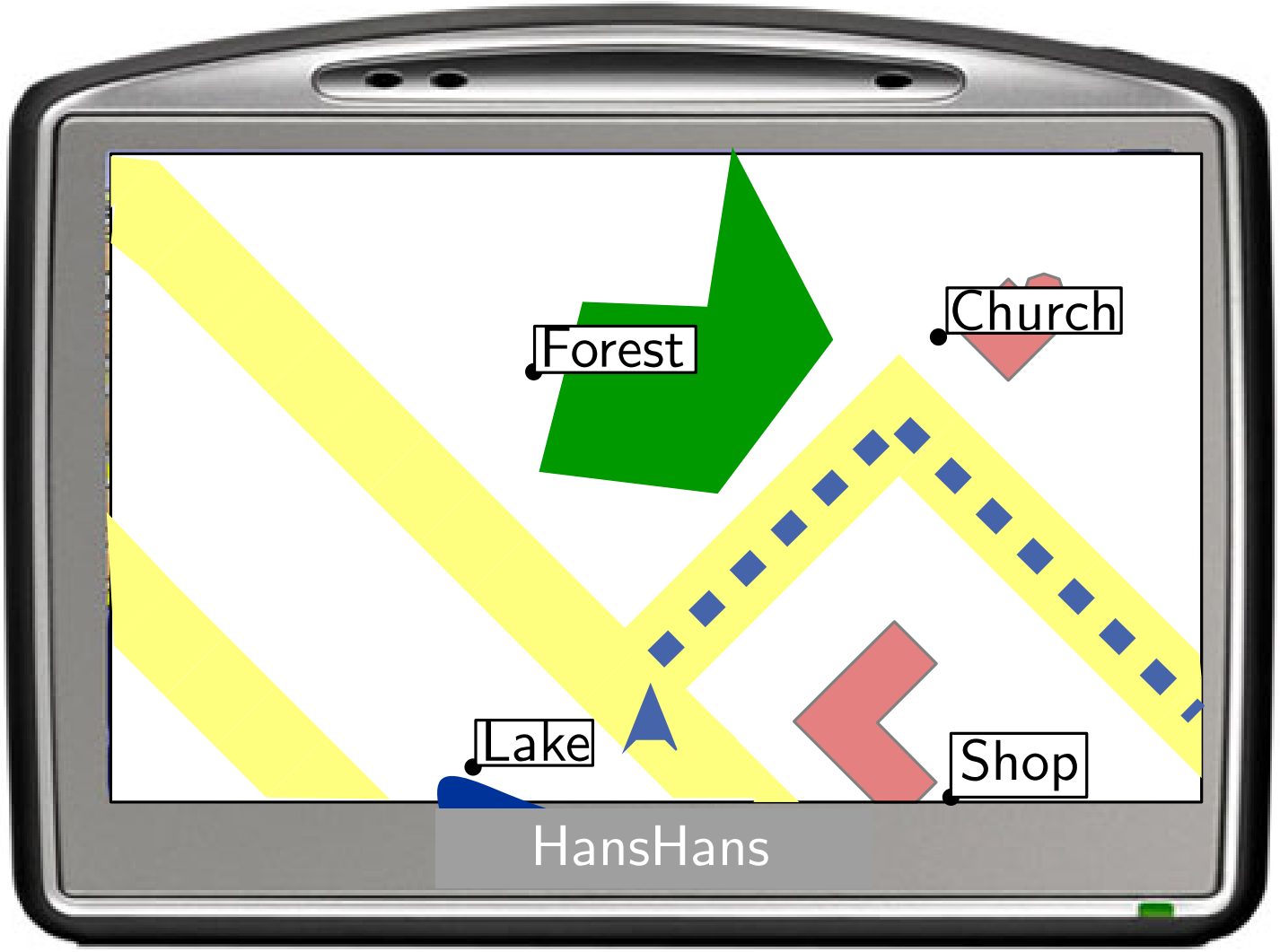
Motivation



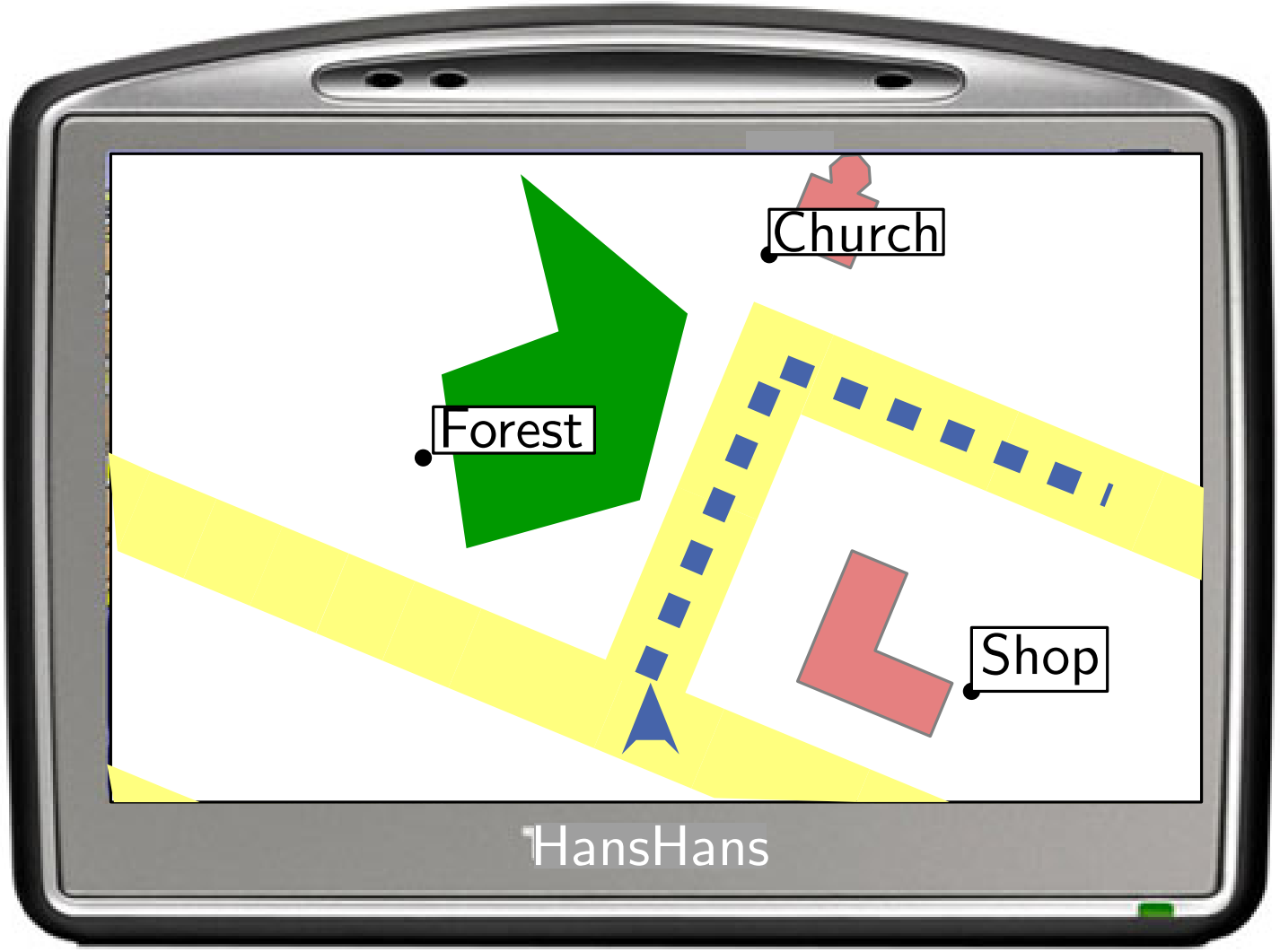
Motivation



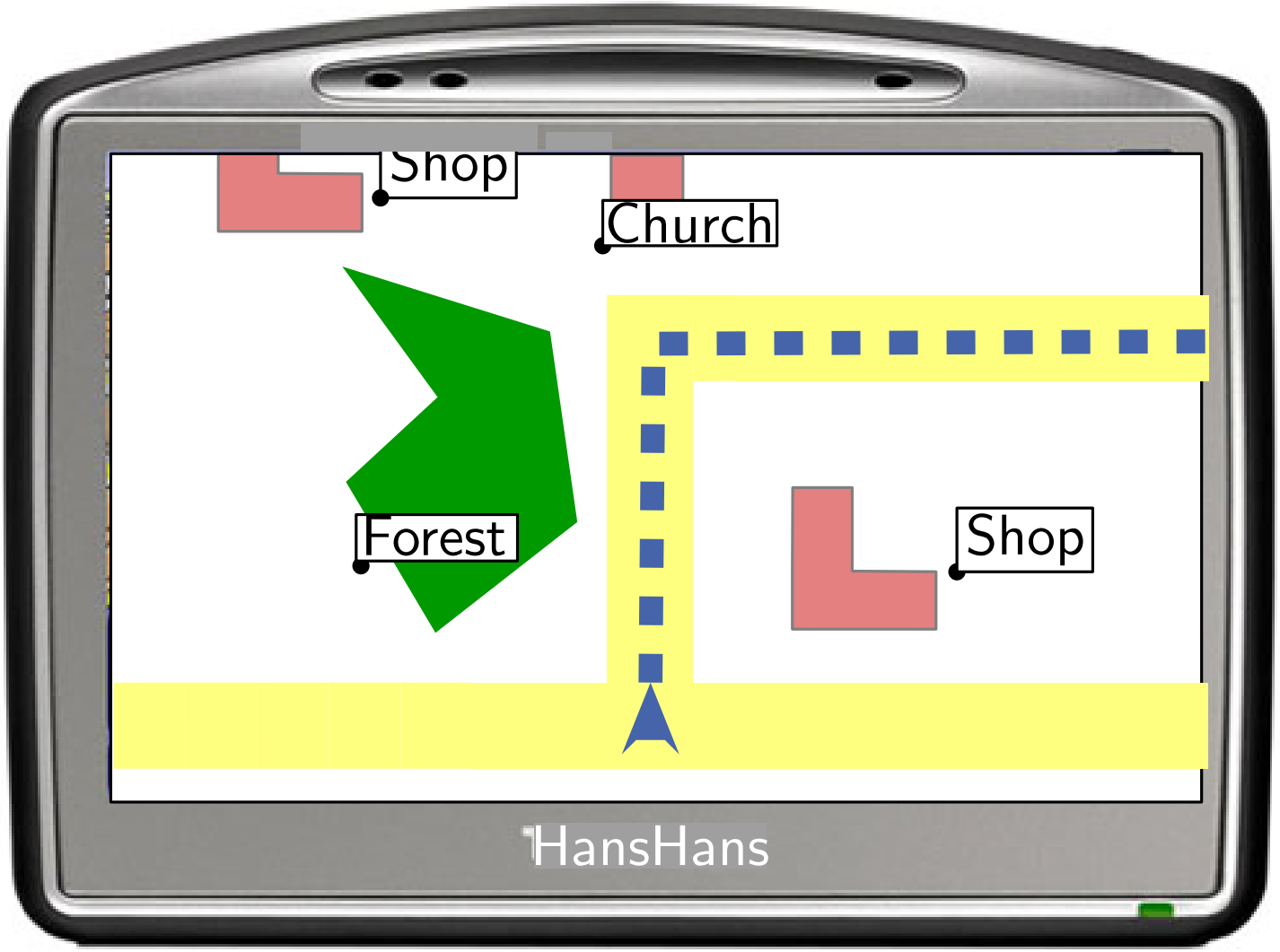
Motivation



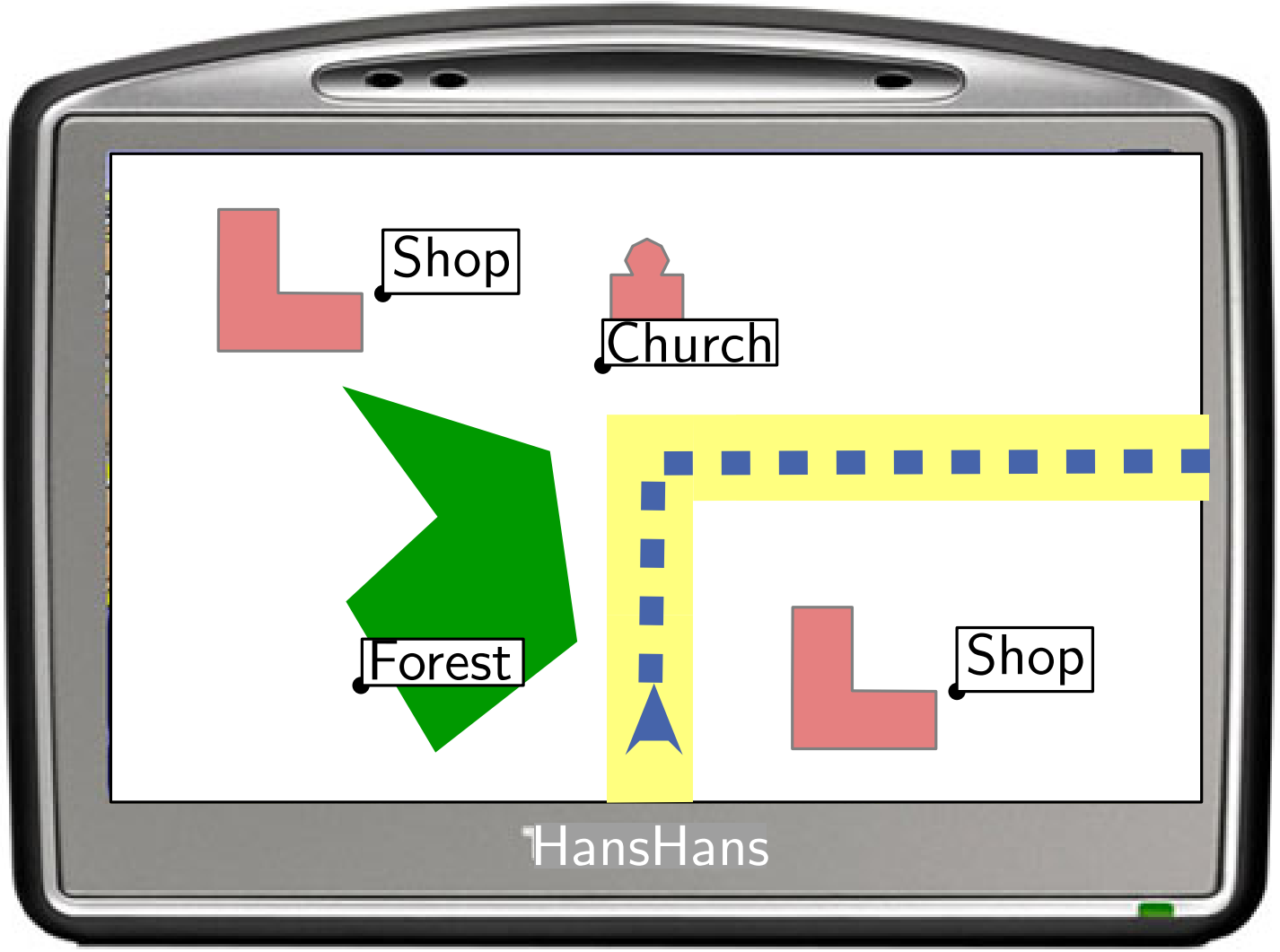
Motivation



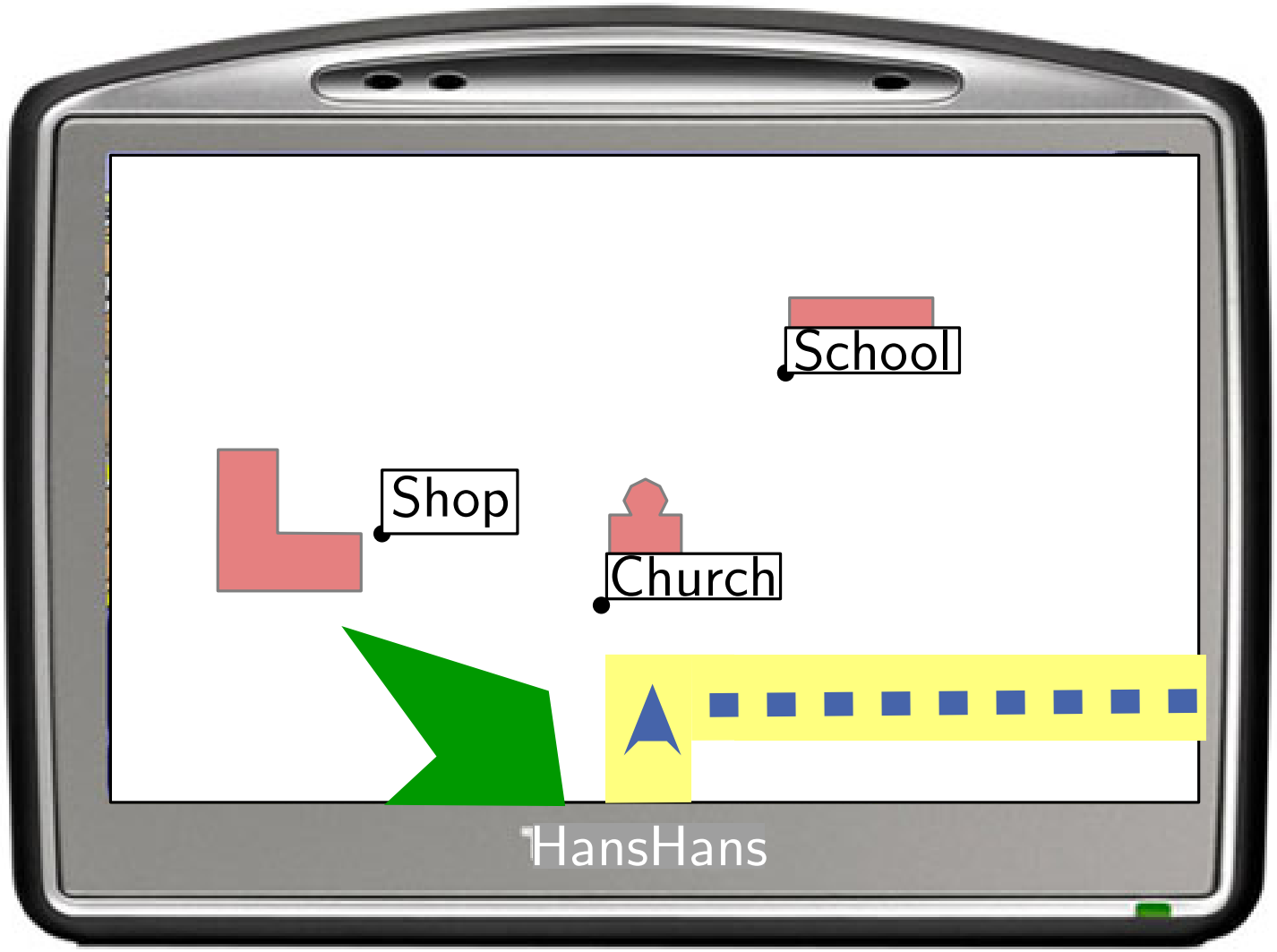
Motivation



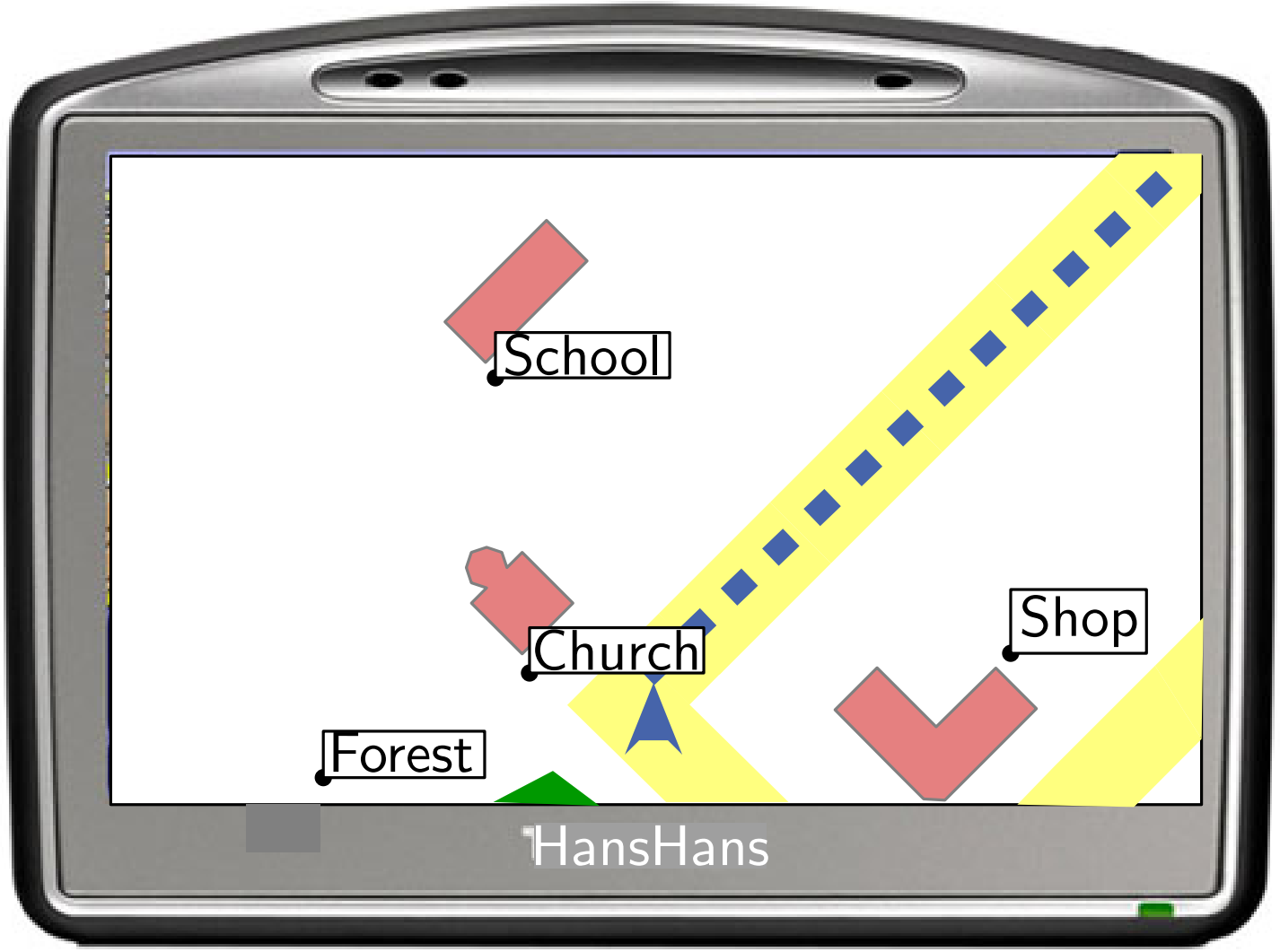
Motivation



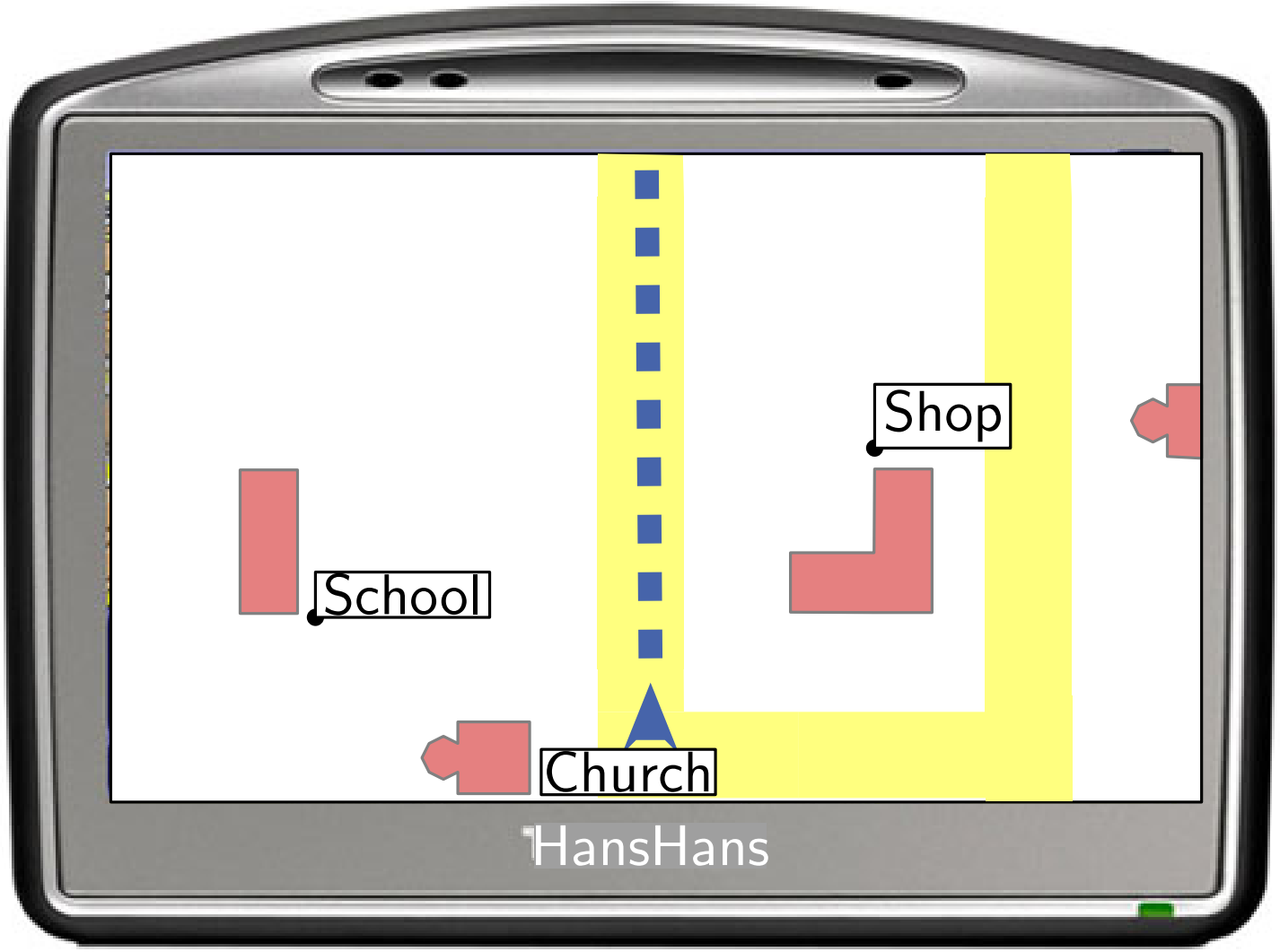
Motivation



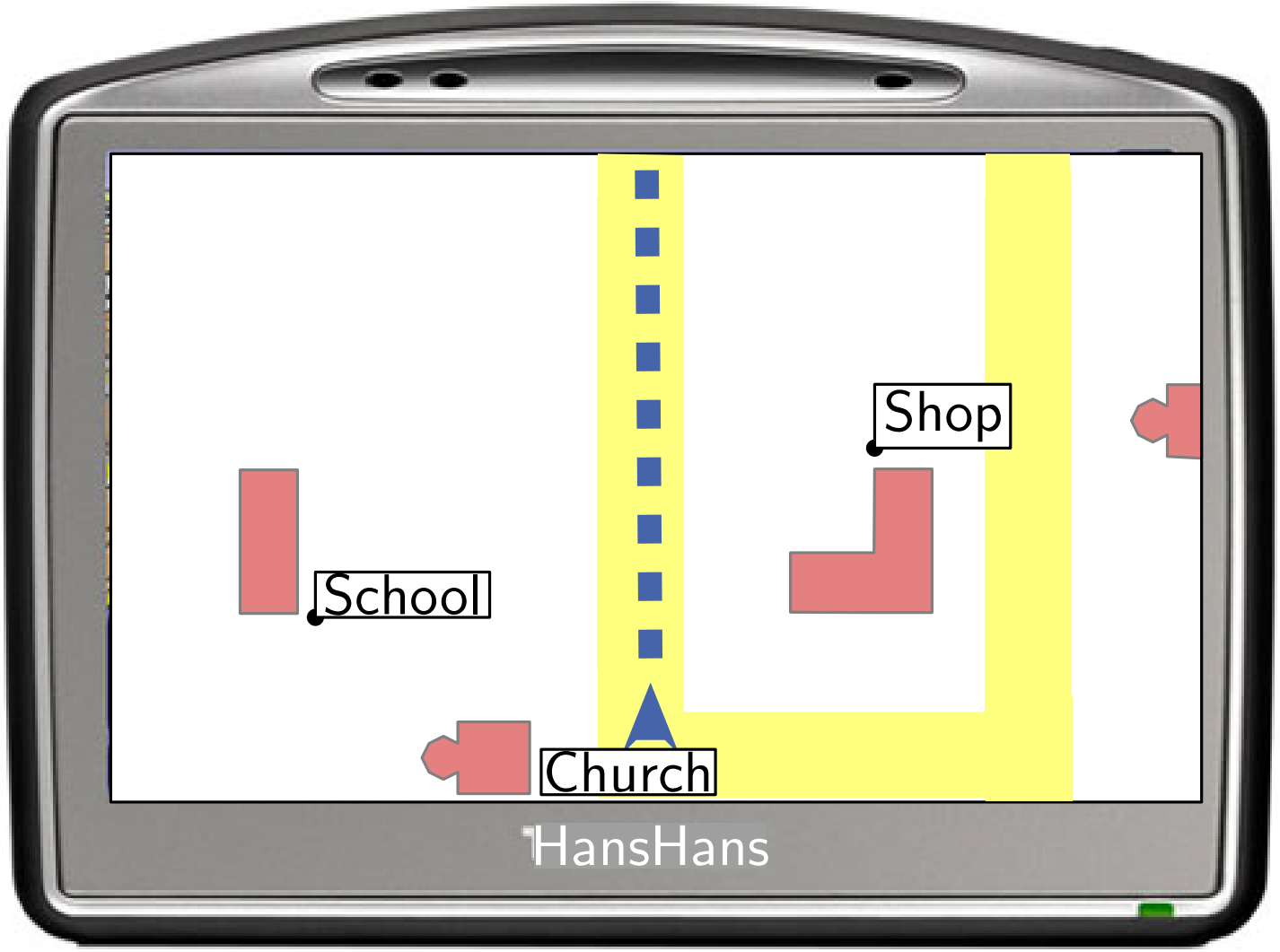
Motivation



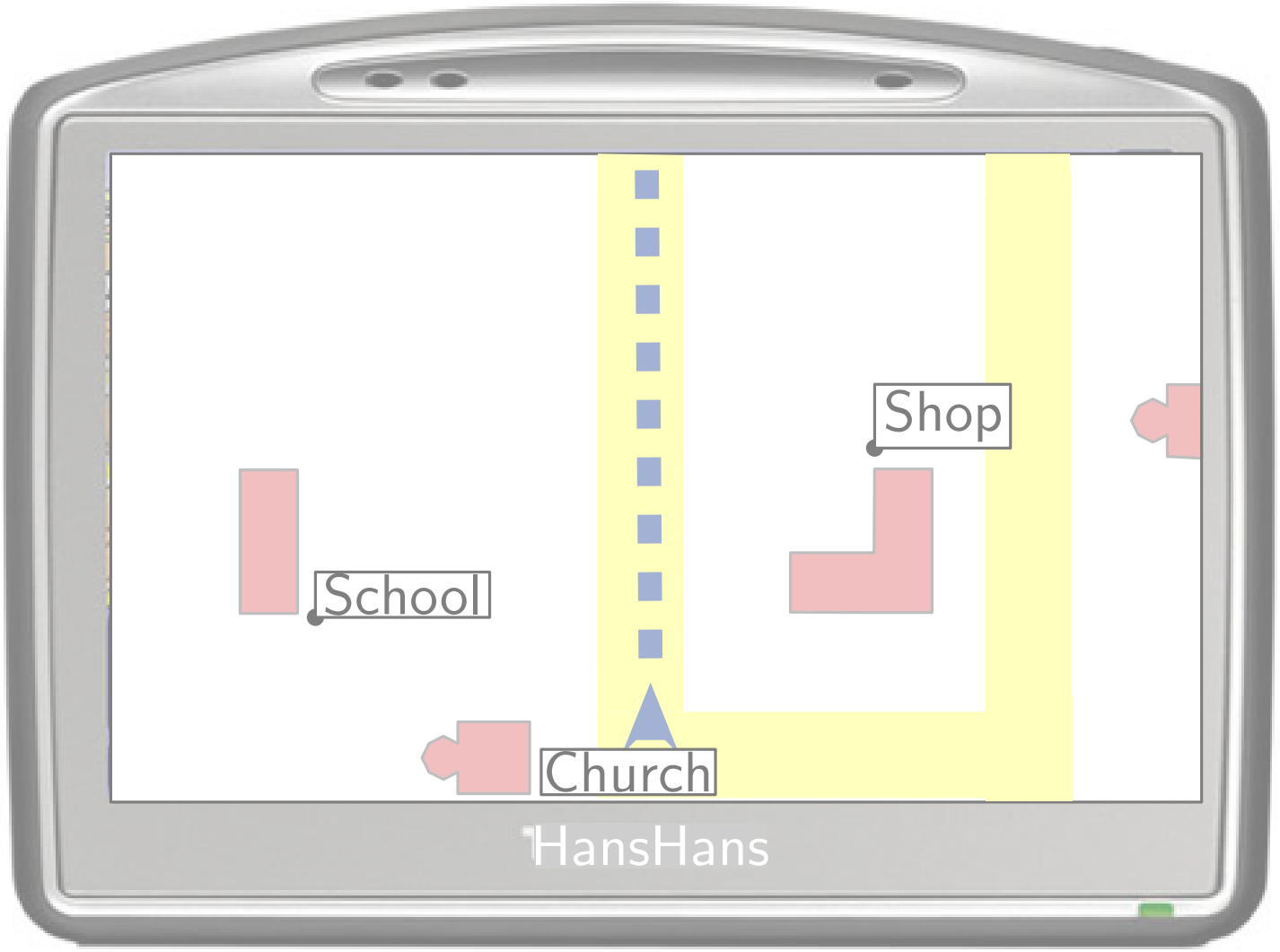
Motivation



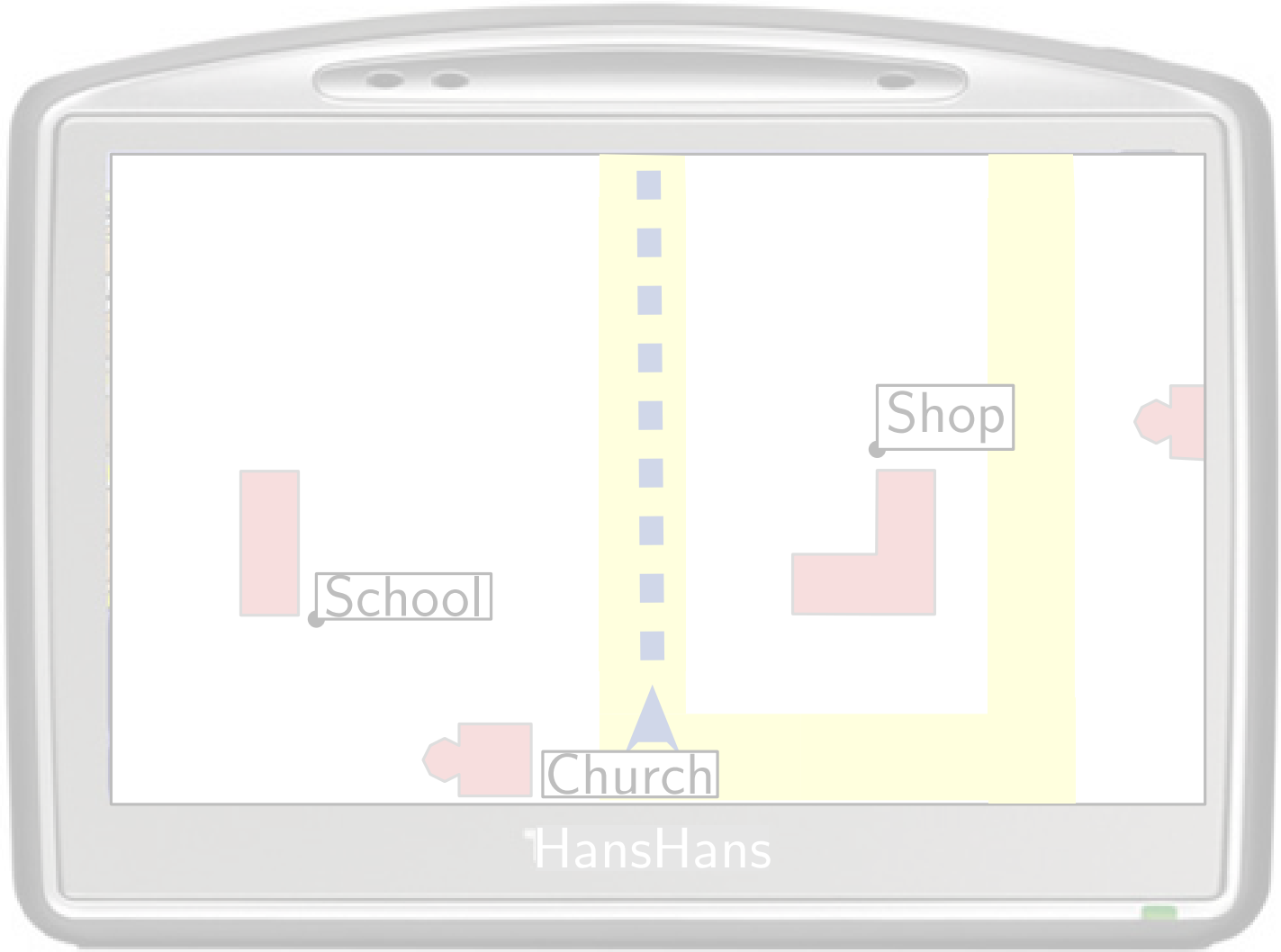
Motivation



Motivation



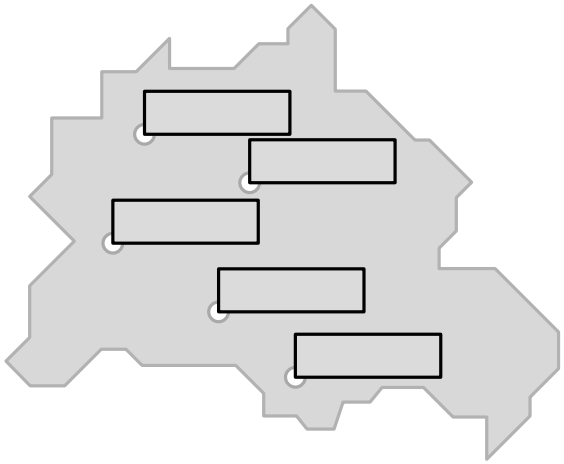
Motivation



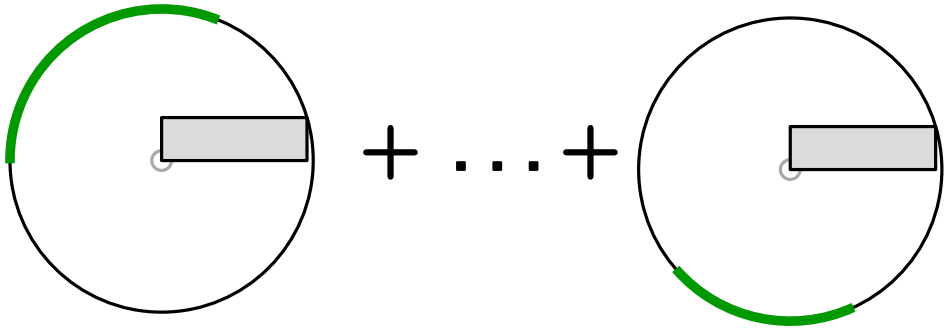
Motivation

Problem Statement

Input: A map M, points P, **valid** labeling $L(P)$



MaxTotal



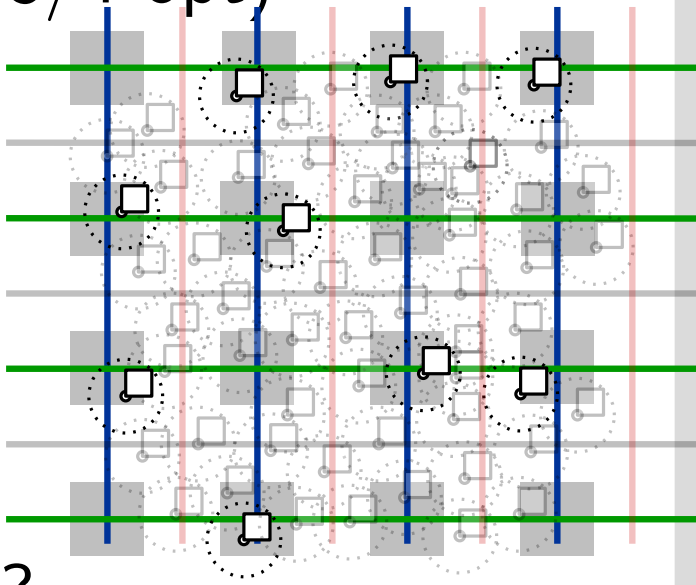
MaxTotal

Output: consistent rotation labeling that maximizes **the sum of all active ranges**

Summary I

Results:

- MaxTotal is NP-complete, MaxMin is NP-hard
- There is an EPTAS for MaxTotal
- EPTAS uses **line stabbing** (geometric problems)
- MaxMin is hard to approximate (within 3/4 opt)



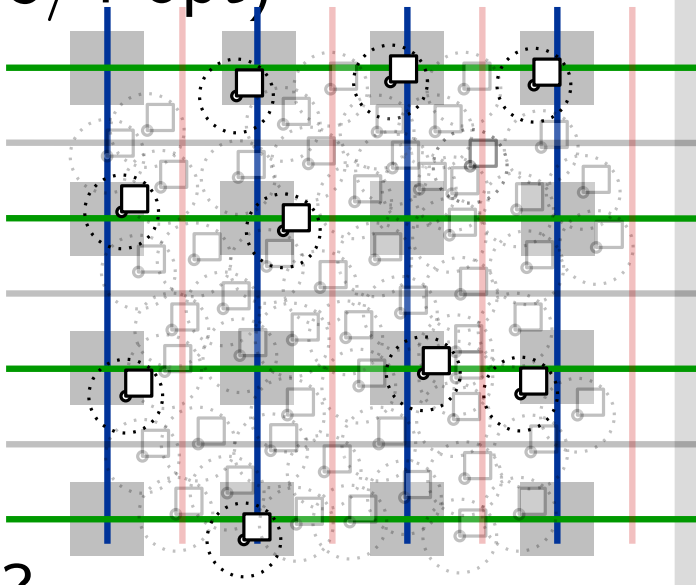
Open Problems:

- Approximation algorithm for MaxMin?
- Simpler/Better algorithms for MaxTotal?

Summary I

Results:

- MaxTotal is NP-complete, MaxMin is NP-hard
- There is an EPTAS for MaxTotal
- EPTAS uses **line stabbing** (geometric problems)
- MaxMin is hard to approximate (within 3/4 opt)



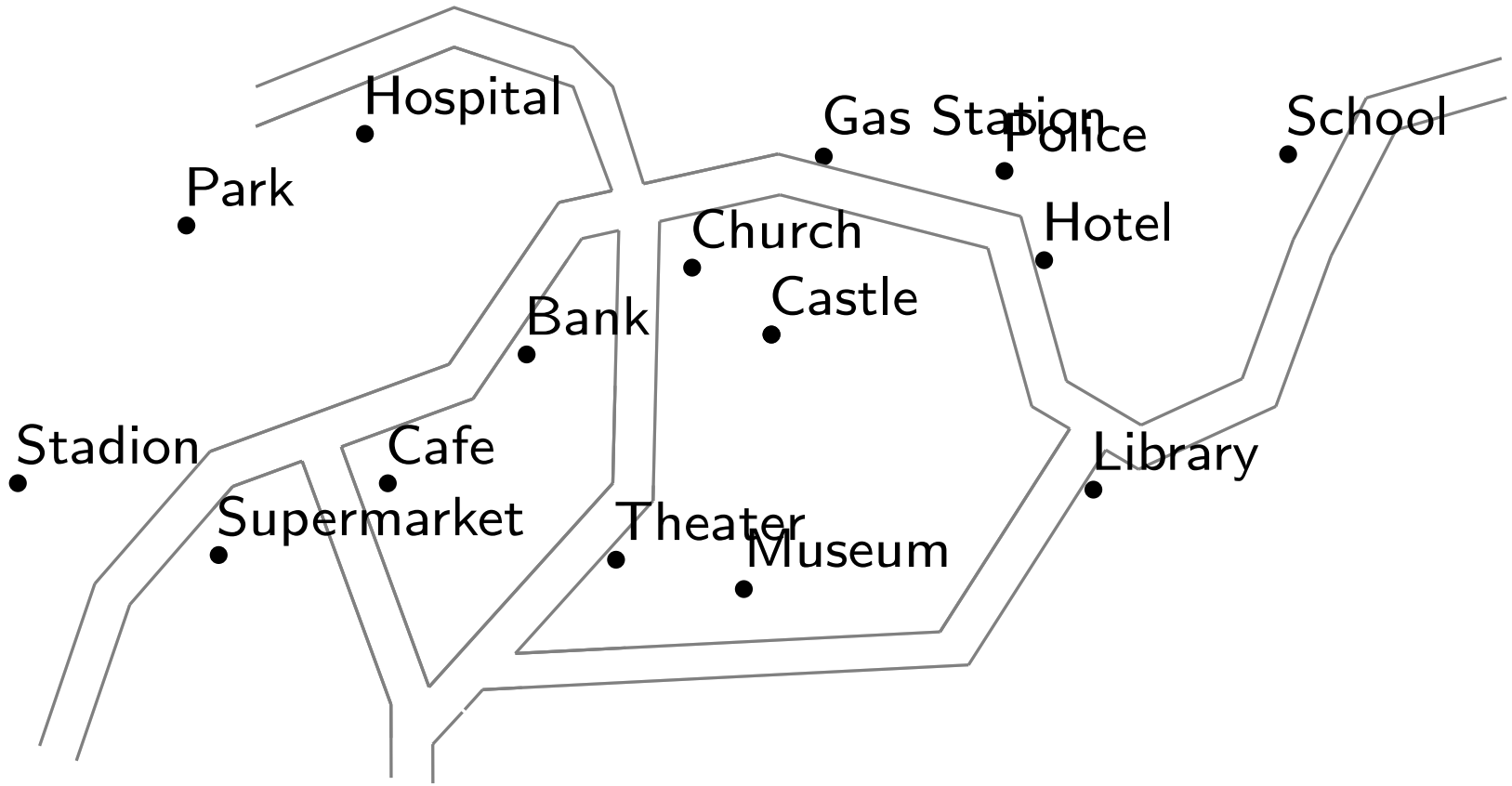
Open Problems:

- Approximation algorithm for MaxMin?
- Simpler/Better algorithms for MaxTotal?

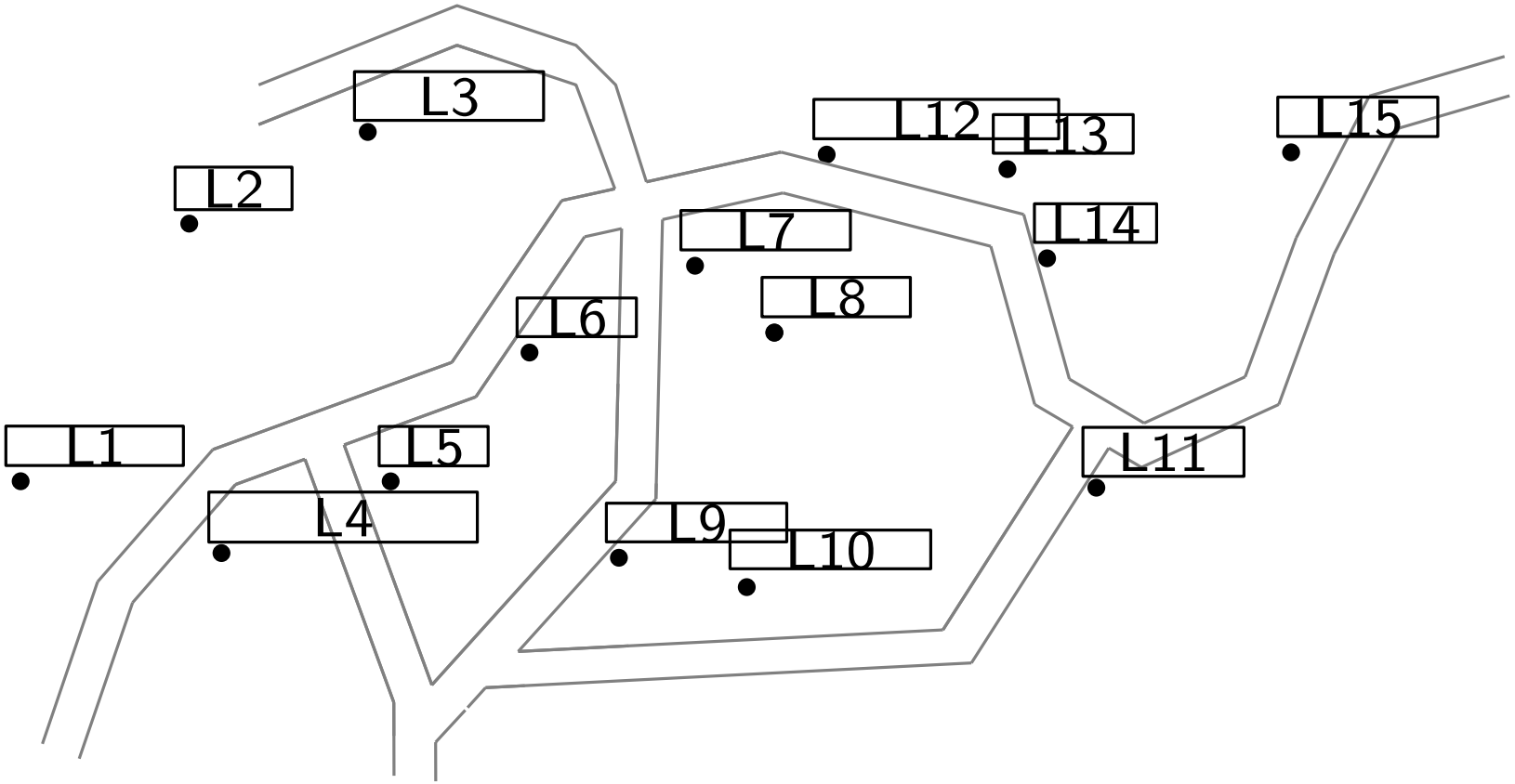
Did we actually solve the problem?

Trajectory-Based Dynamic Map Labeling

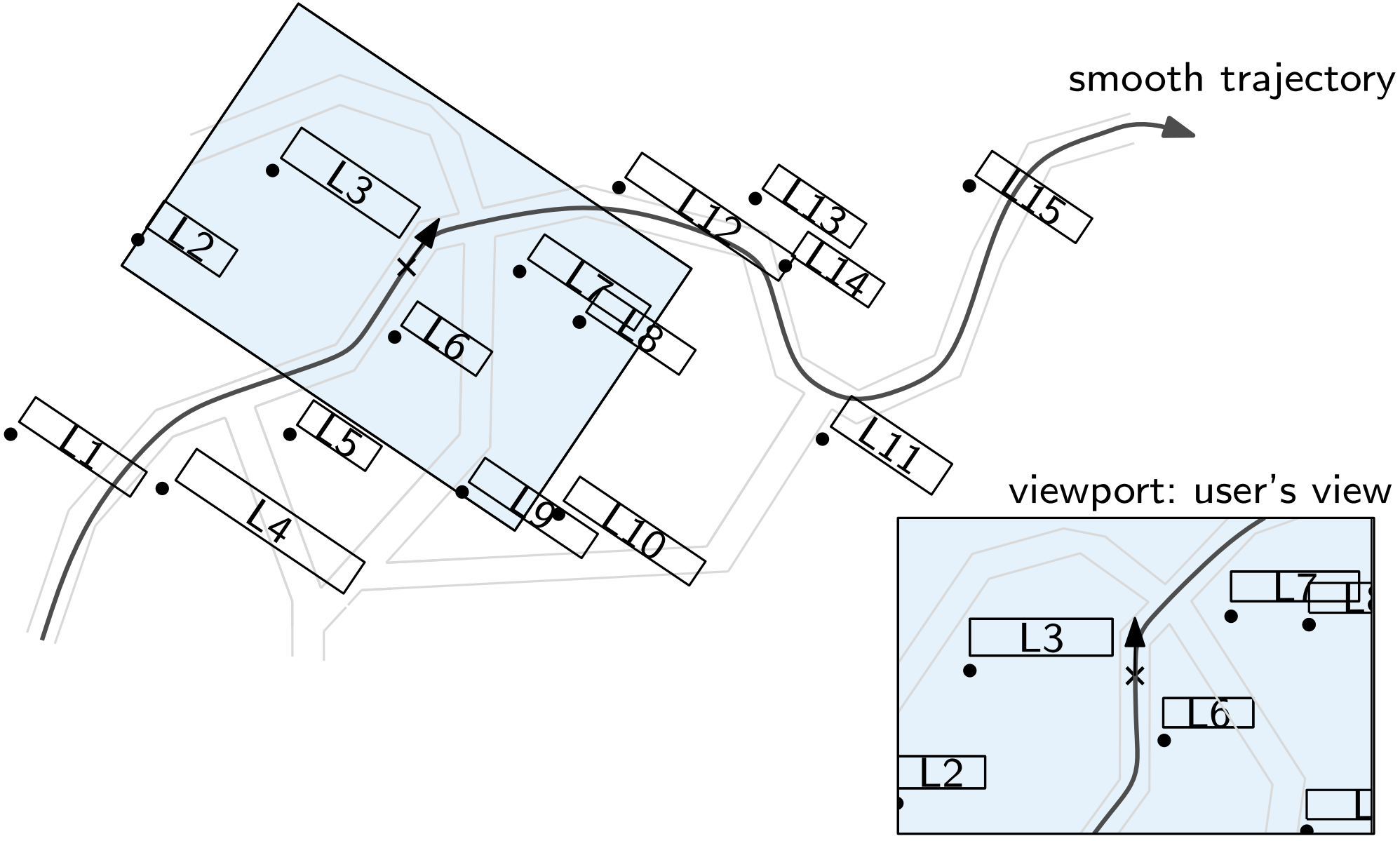
Trajectory-Based Dynamic Map Labeling



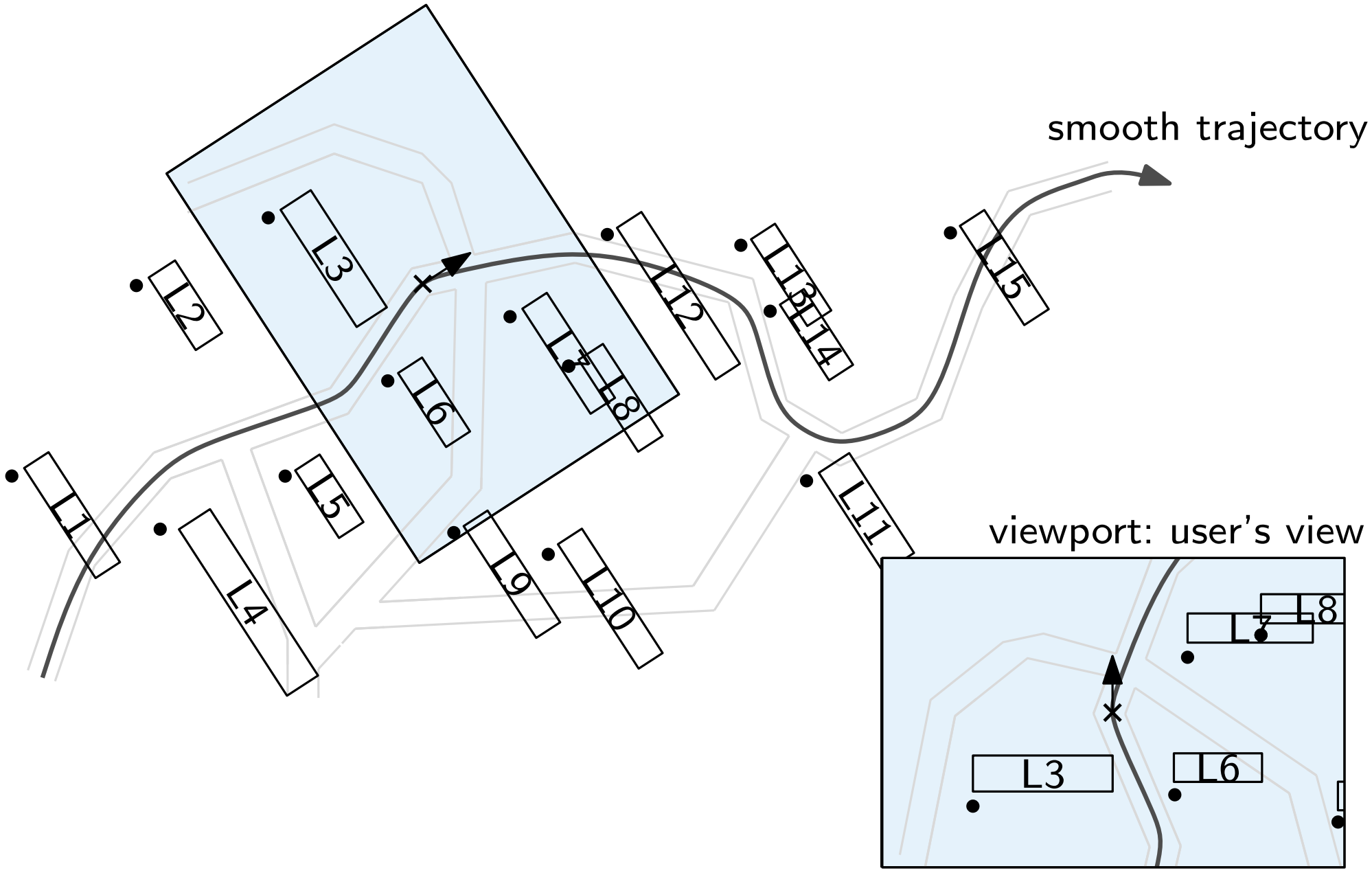
Trajectory-Based Dynamic Map Labeling



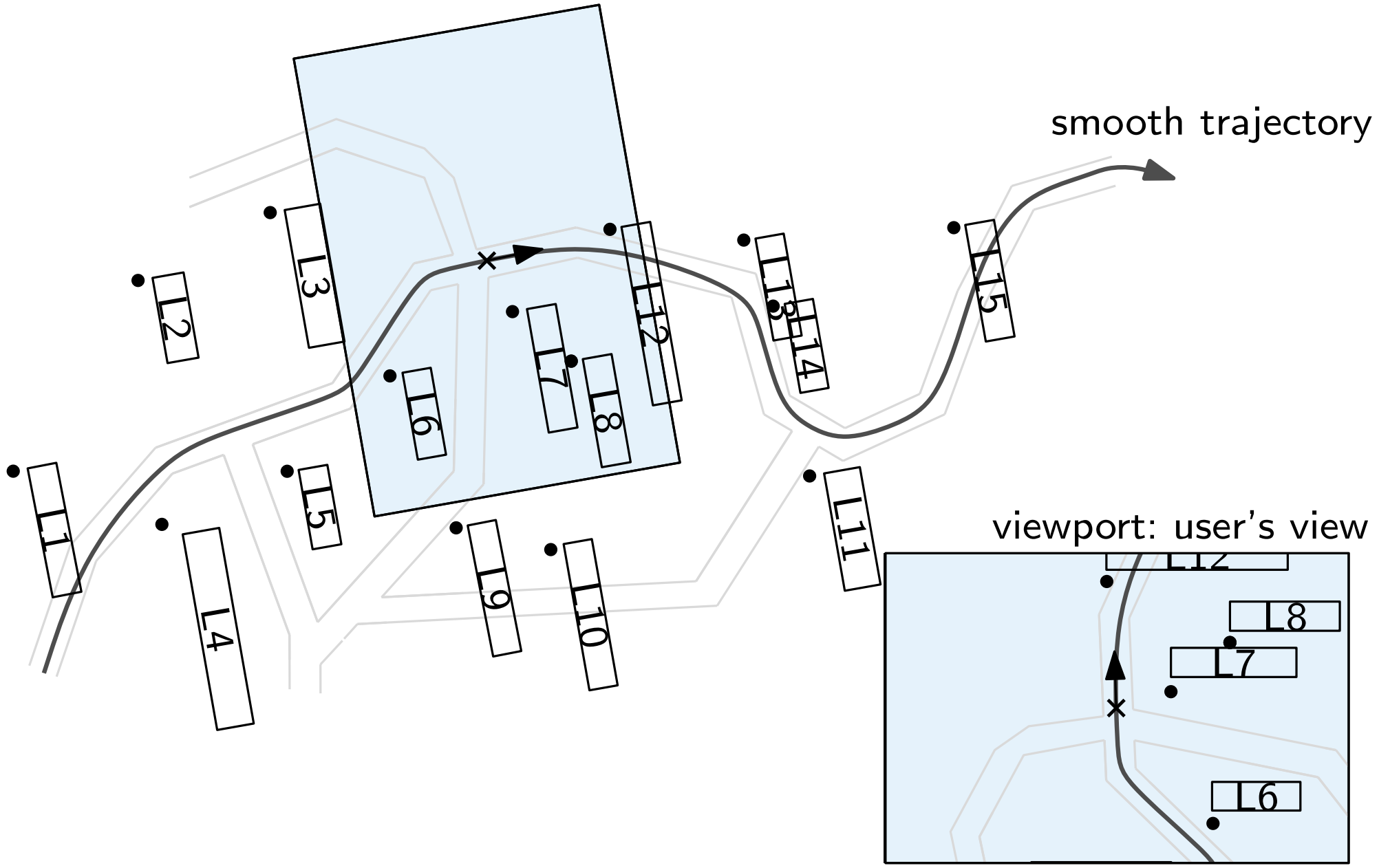
Trajectory-Based Dynamic Map Labeling



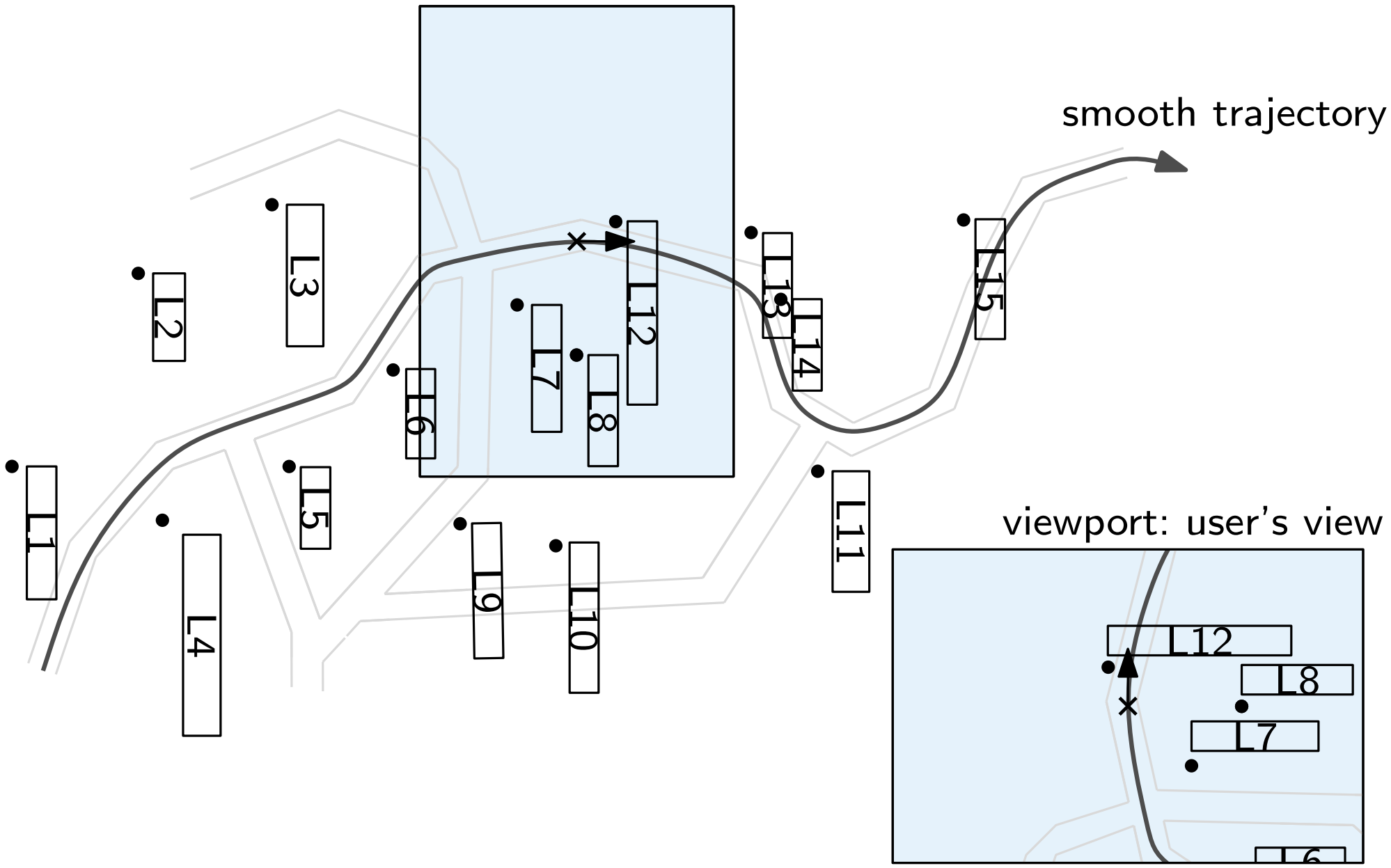
Trajectory-Based Dynamic Map Labeling



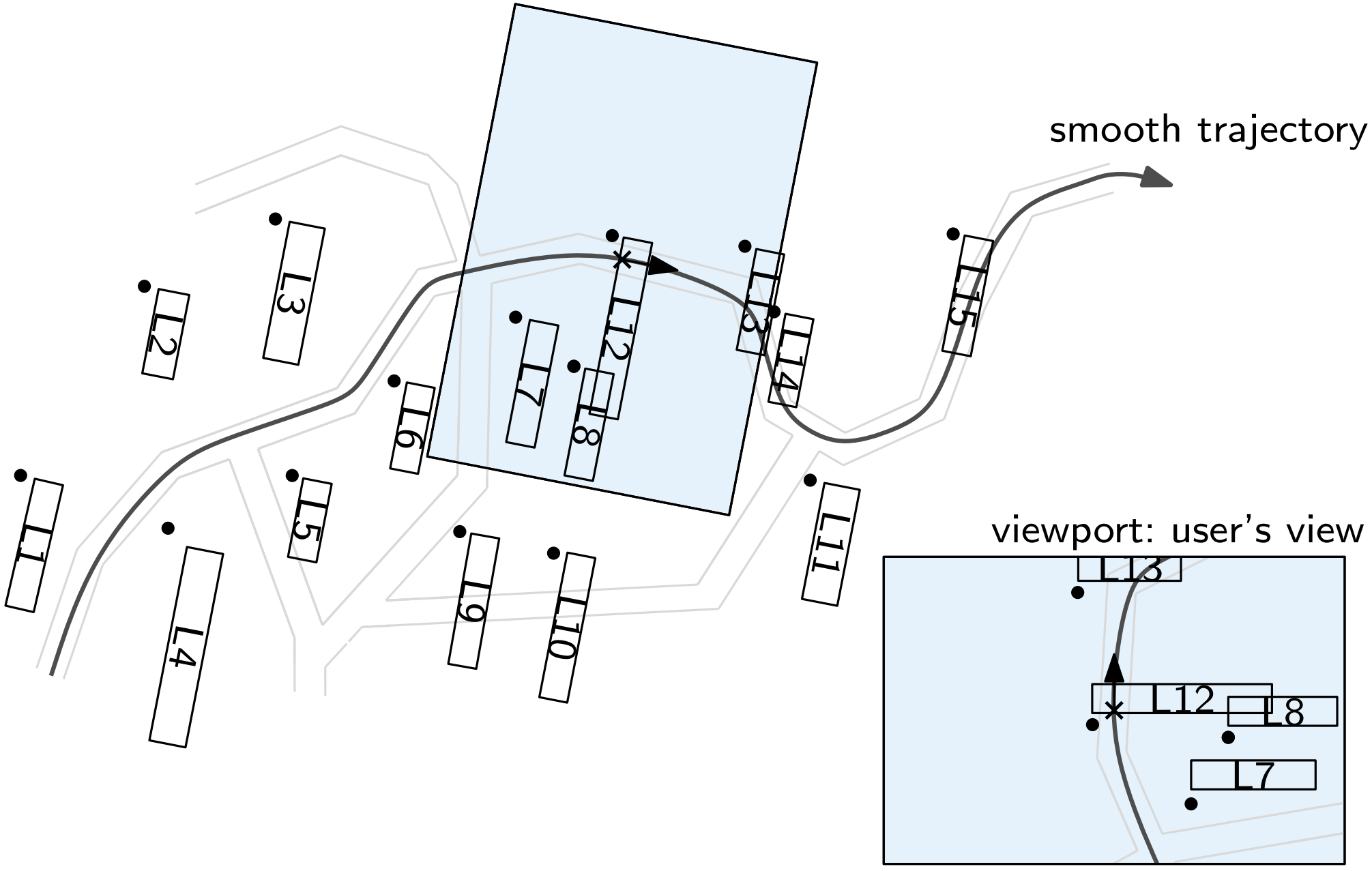
Trajectory-Based Dynamic Map Labeling



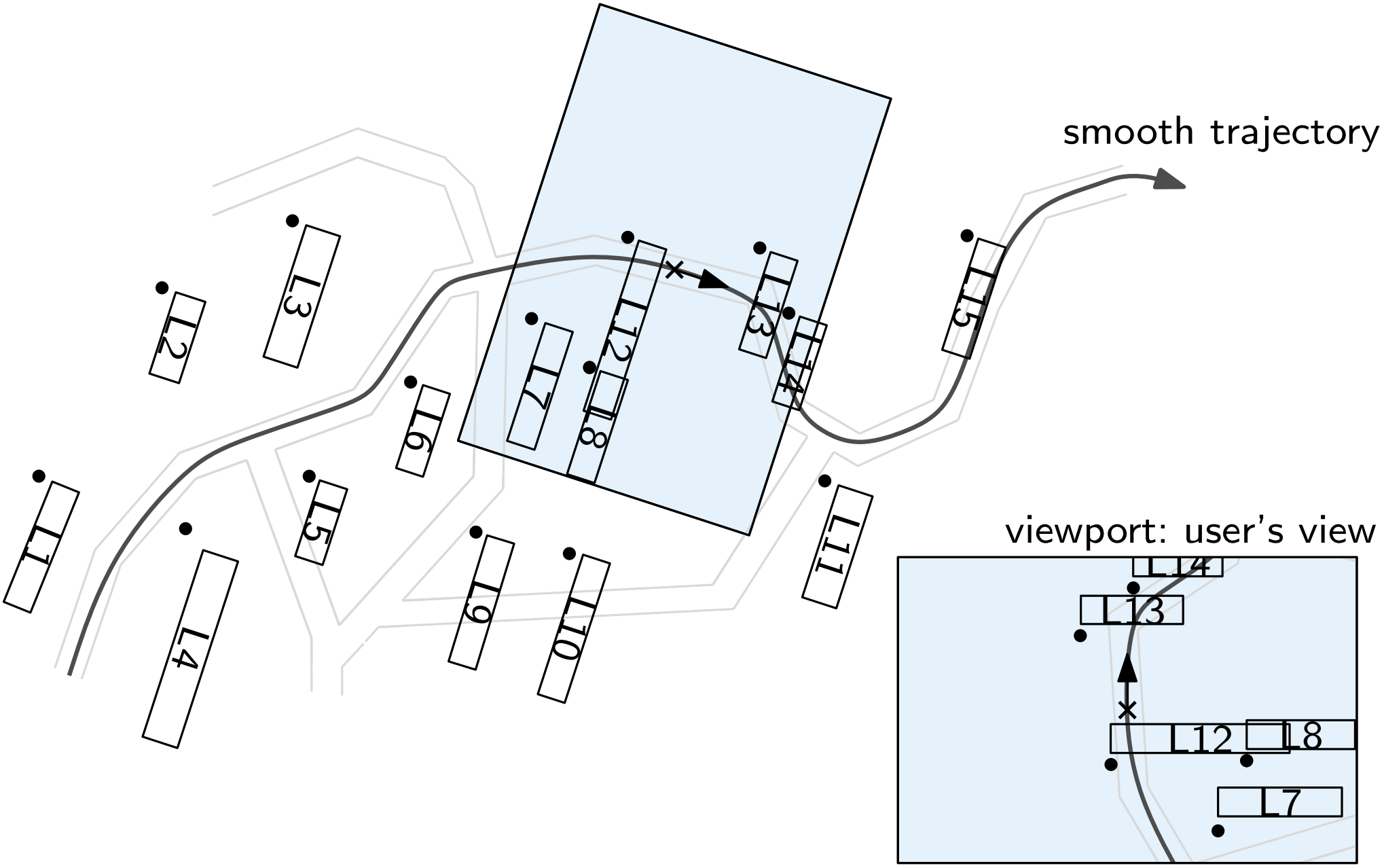
Trajectory-Based Dynamic Map Labeling



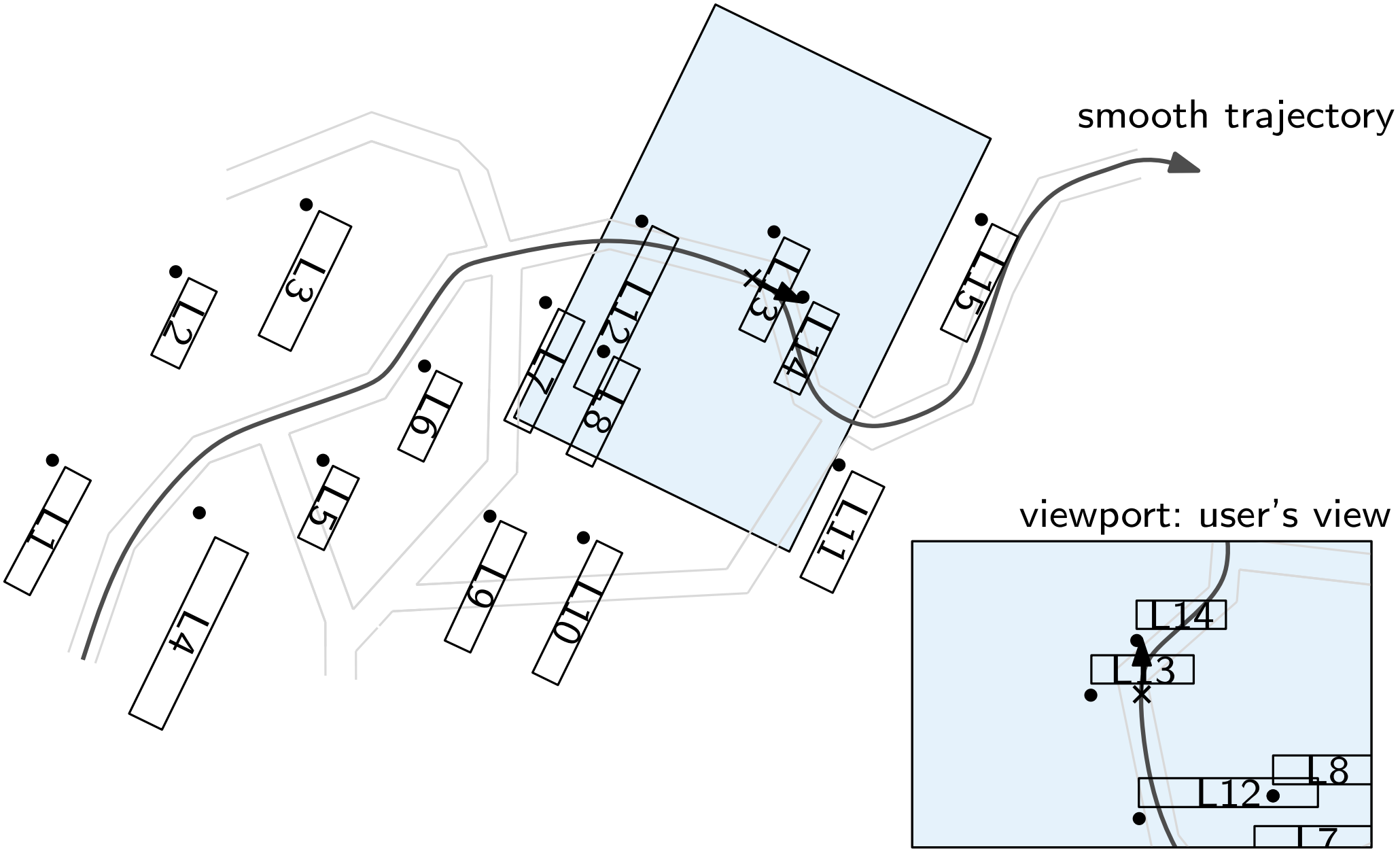
Trajectory-Based Dynamic Map Labeling



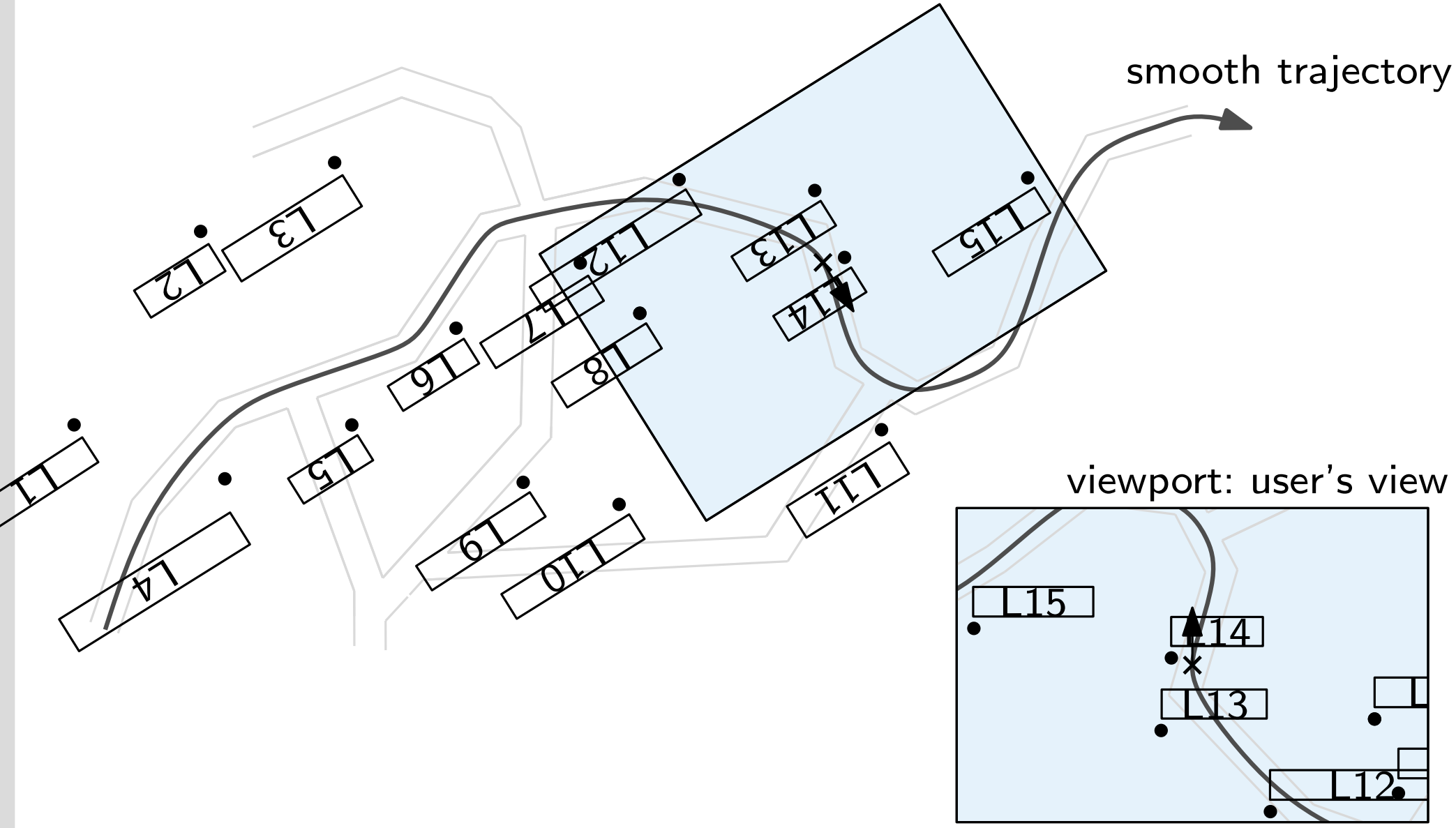
Trajectory-Based Dynamic Map Labeling



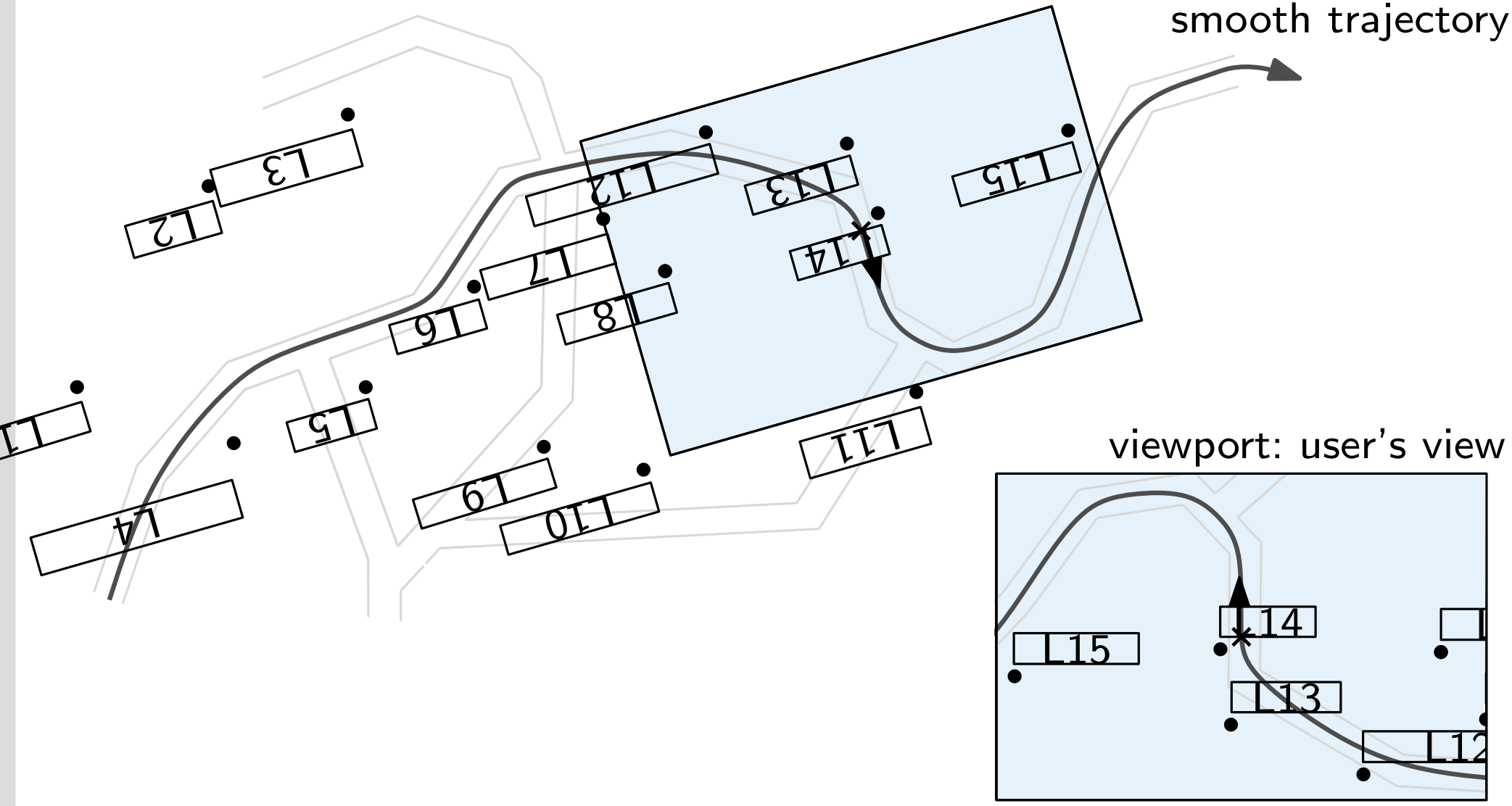
Trajectory-Based Dynamic Map Labeling



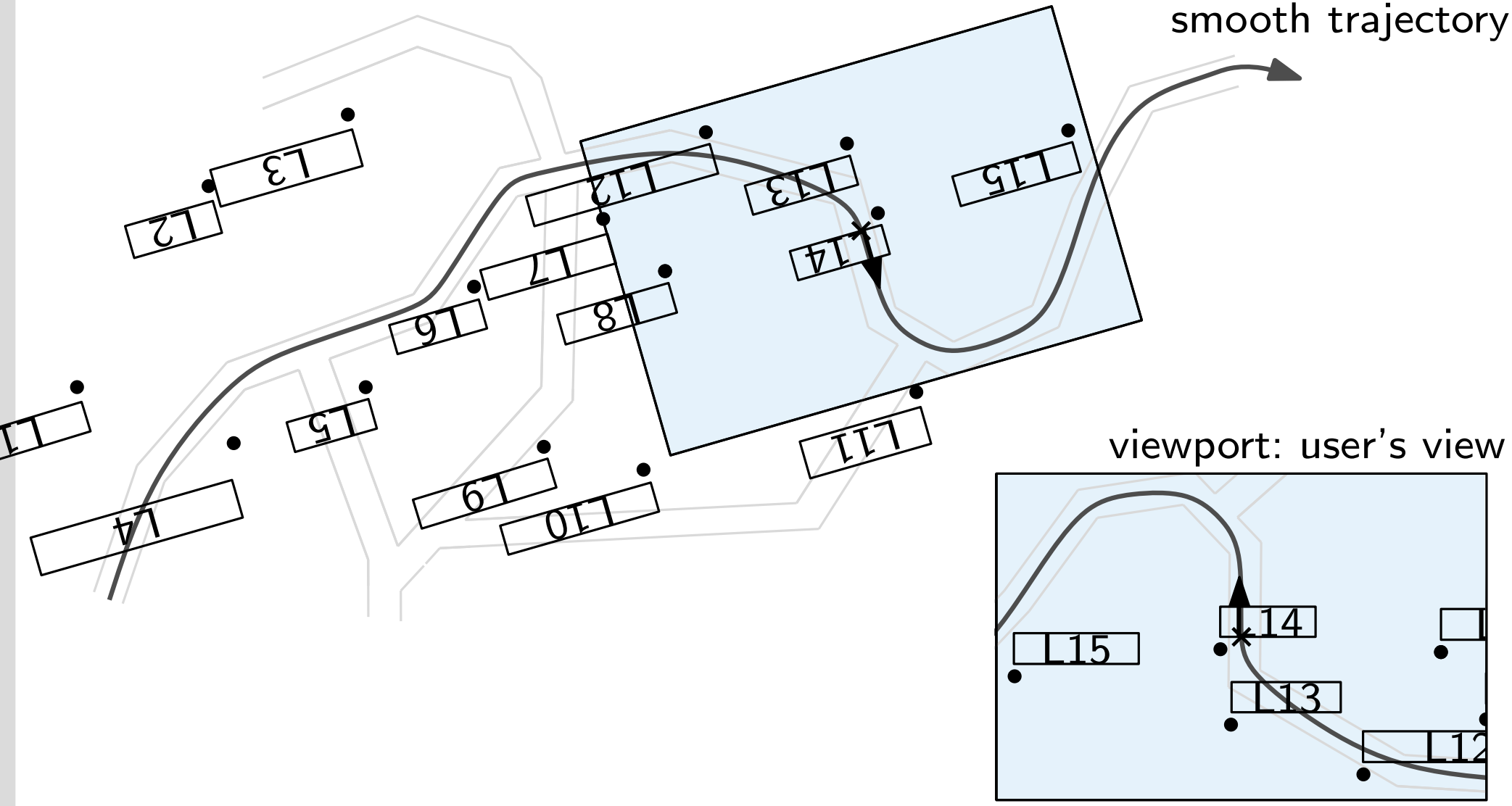
Trajectory-Based Dynamic Map Labeling



Trajectory-Based Dynamic Map Labeling

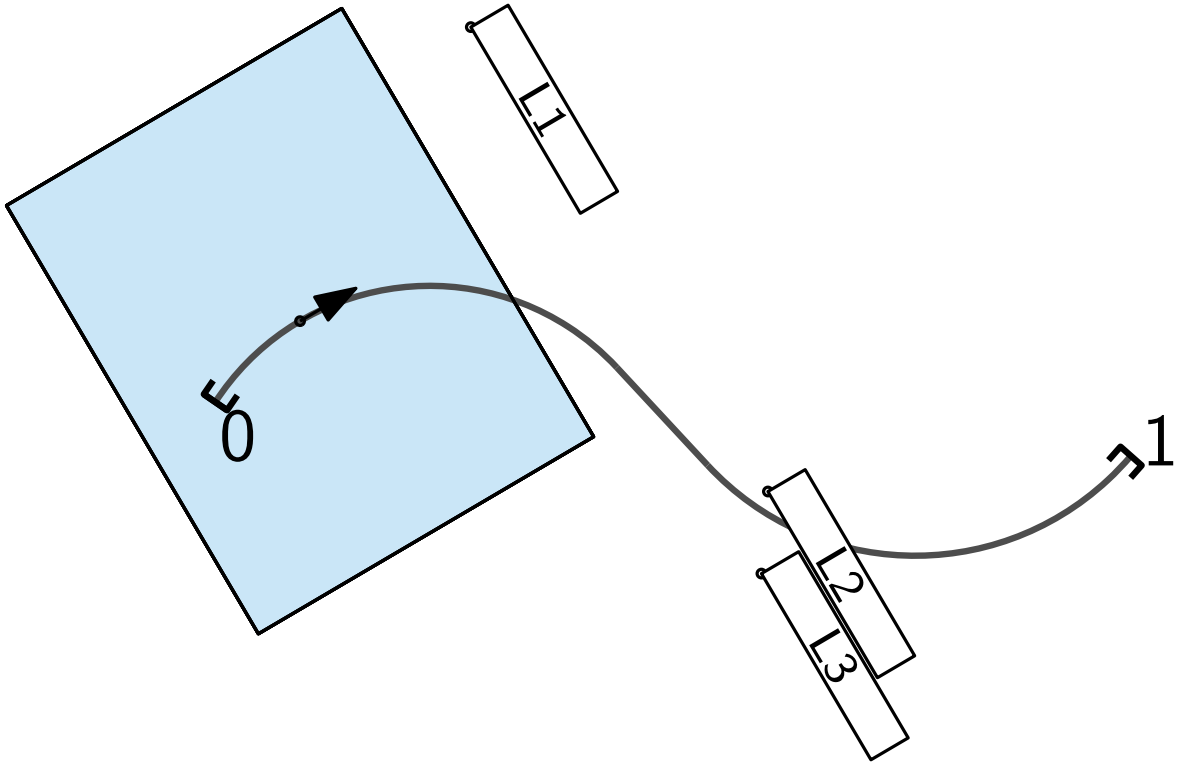


Trajectory-Based Dynamic Map Labeling

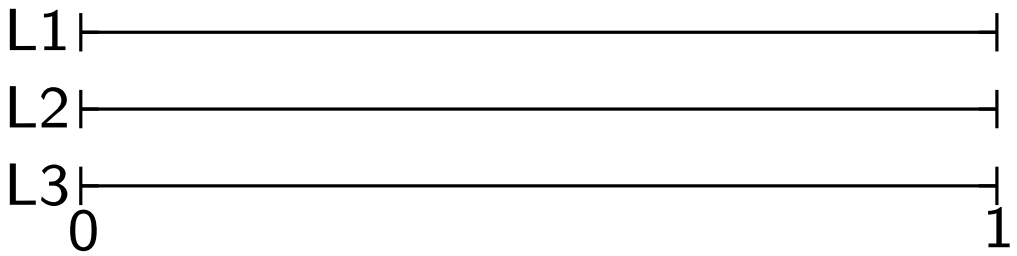


Mapping to Intervals

Label is **present** at time t if it intersects the viewport at time t

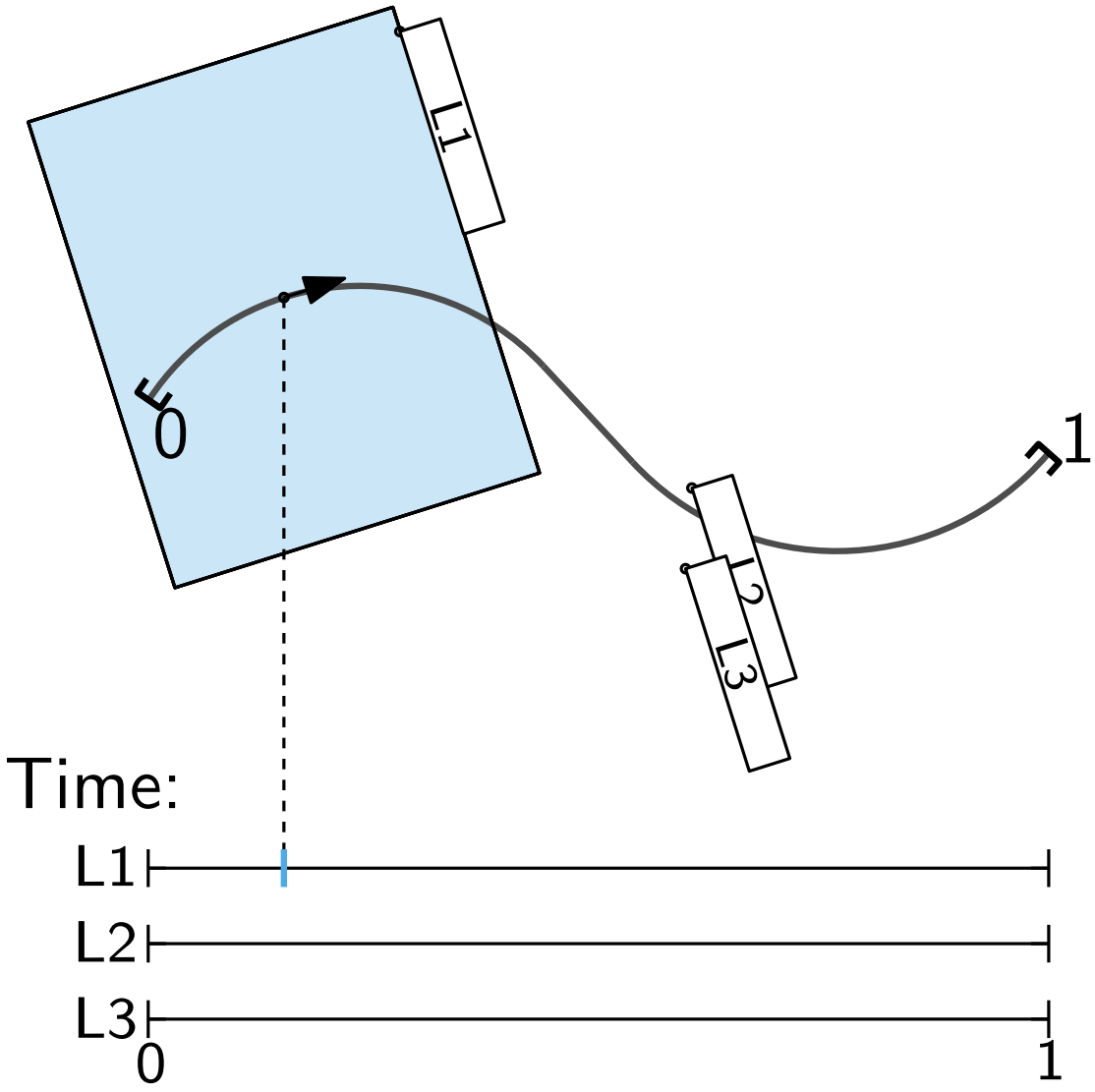


Time:



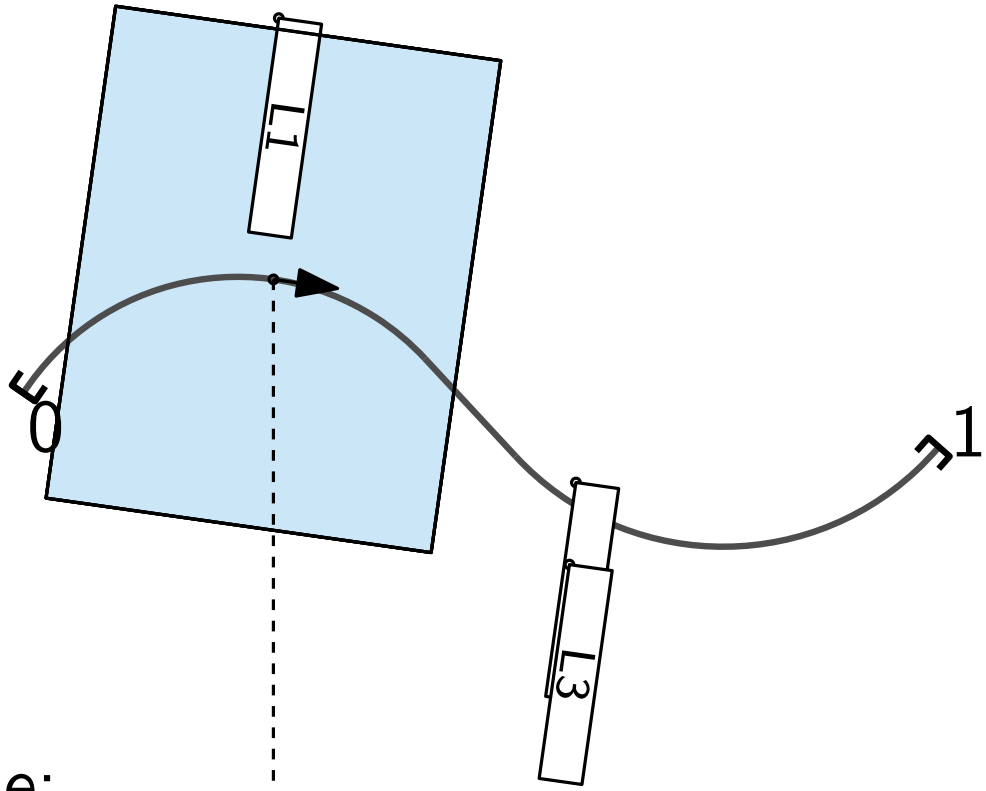
Mapping to Intervals

Label is **present** at time t if it intersects the viewport at time t

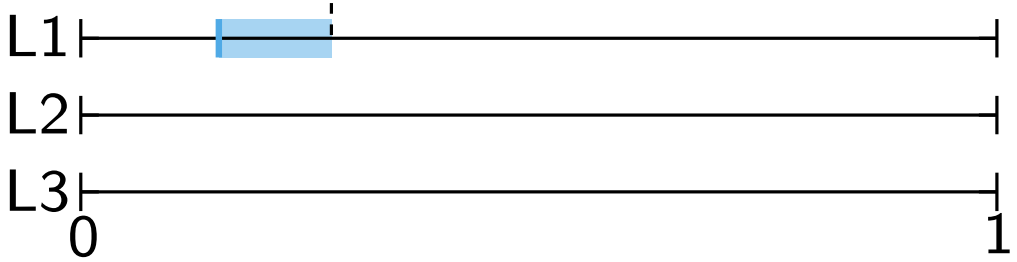


Mapping to Intervals

Label is **present** at time t if it intersects the viewport at time t

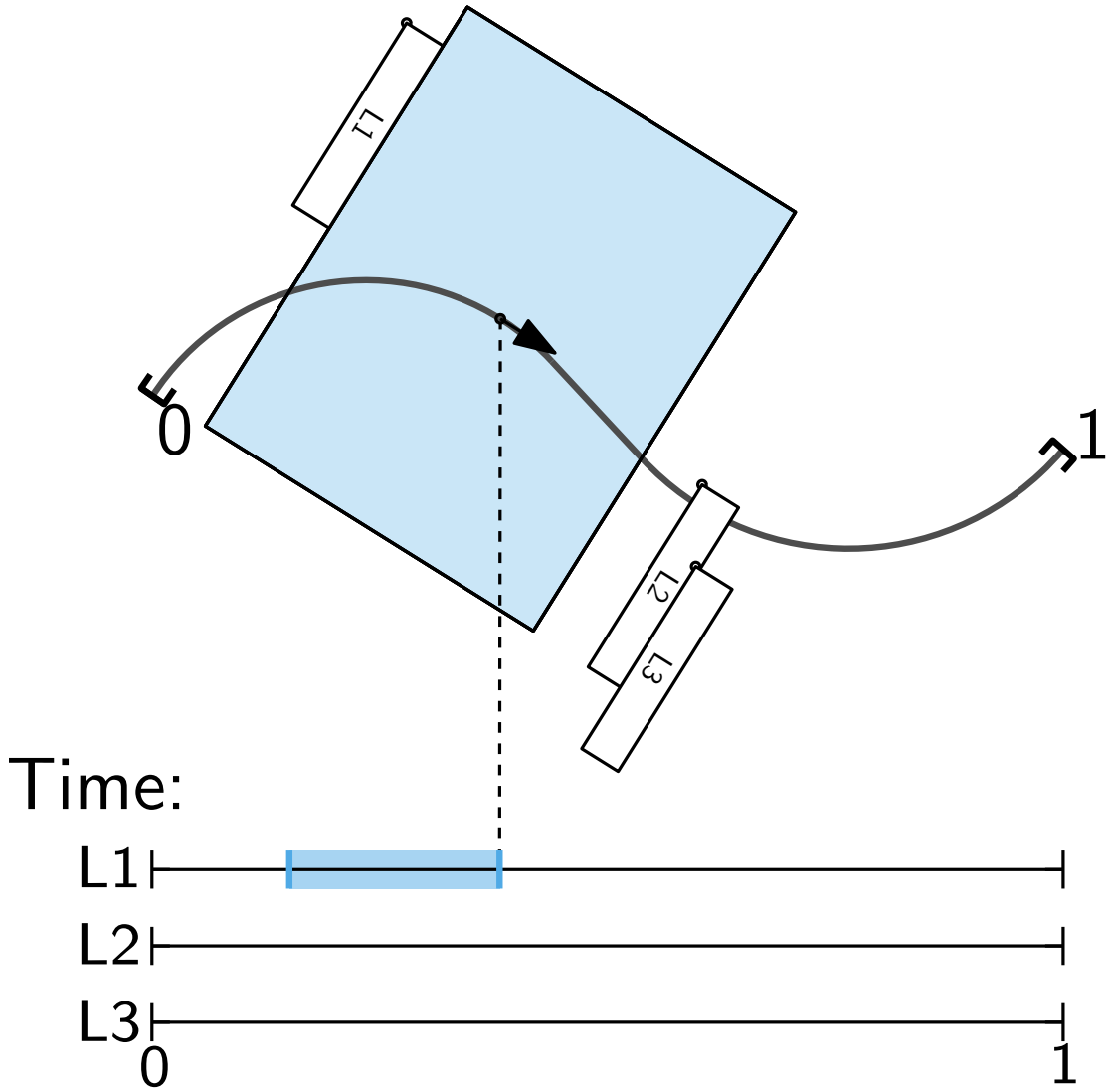


Time:



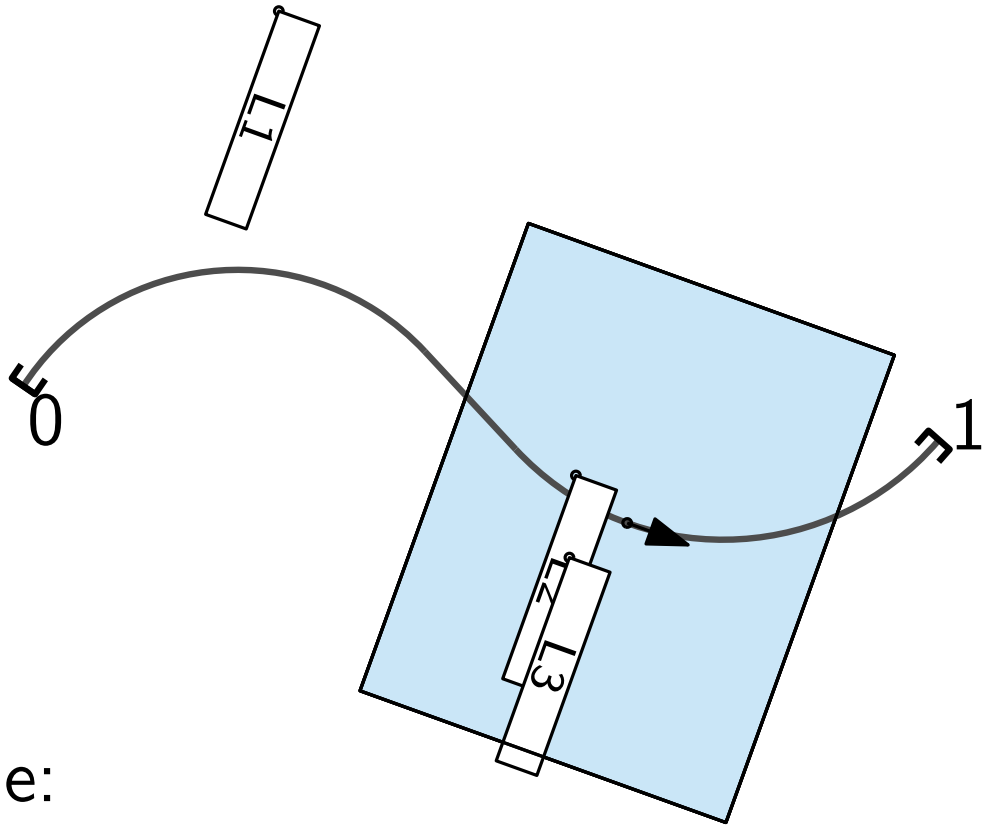
Mapping to Intervals

Label is **present** at time t if it intersects the viewport at time t

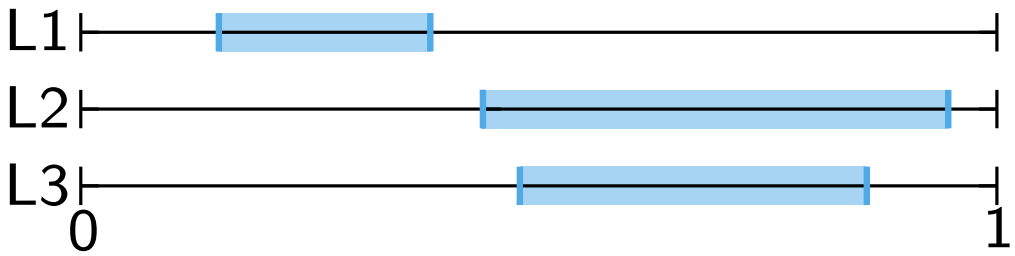


Mapping to Intervals

Label is **present** at time t if it intersects the viewport at time t



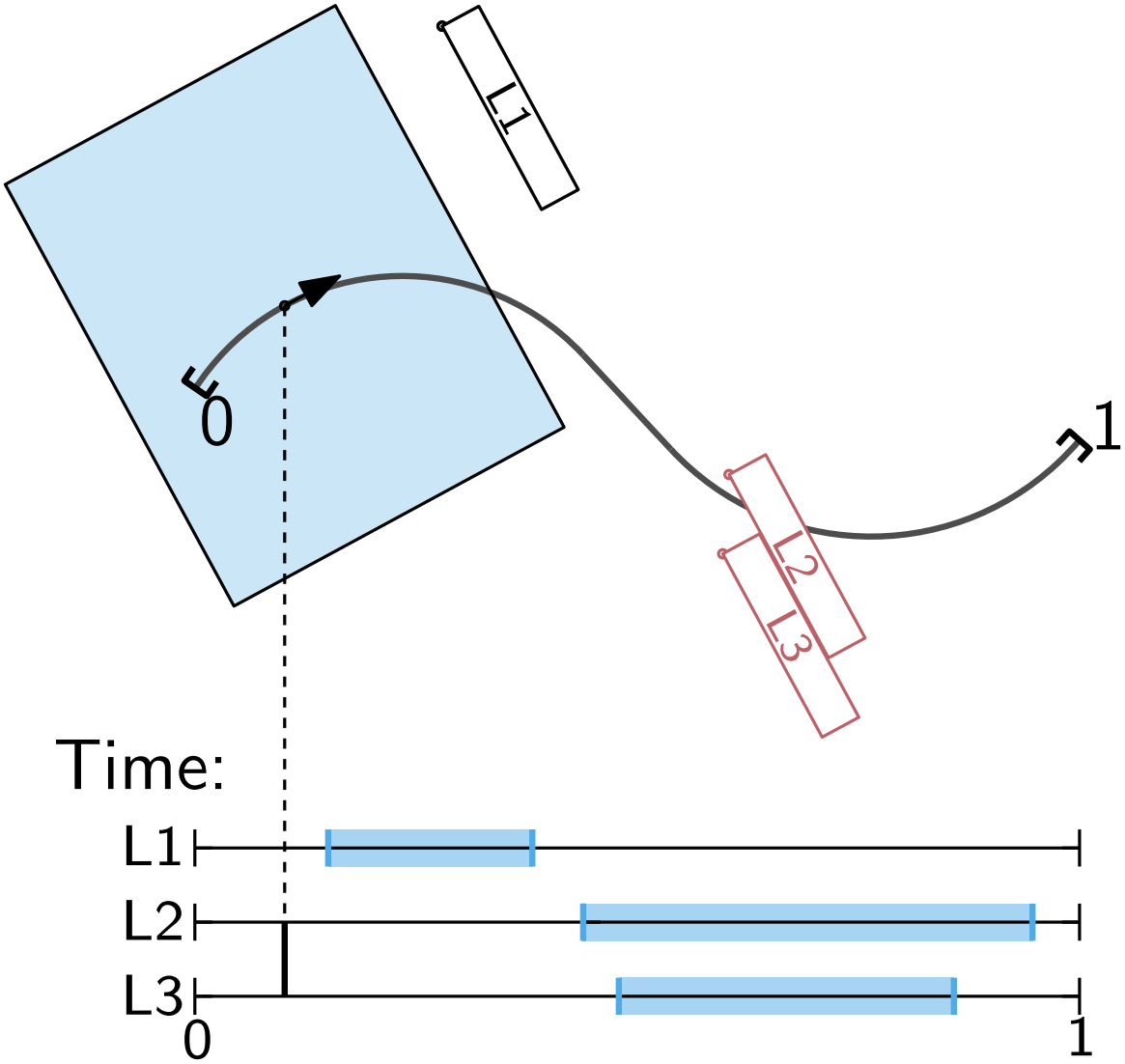
Time:



set of *presence intervals*

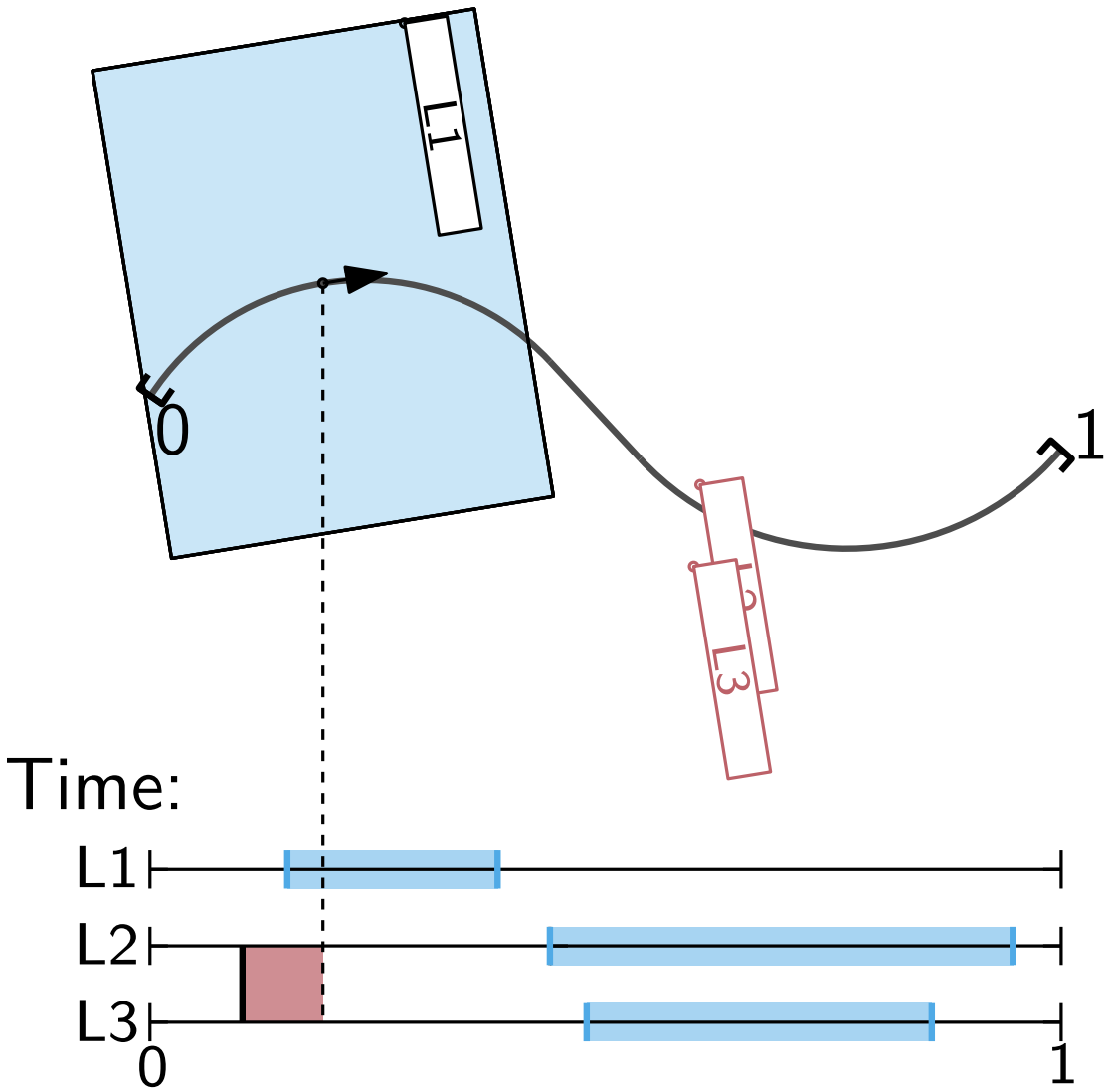
Mapping to Intervals

Two labels are in **conflict** at time t if they intersect at time t



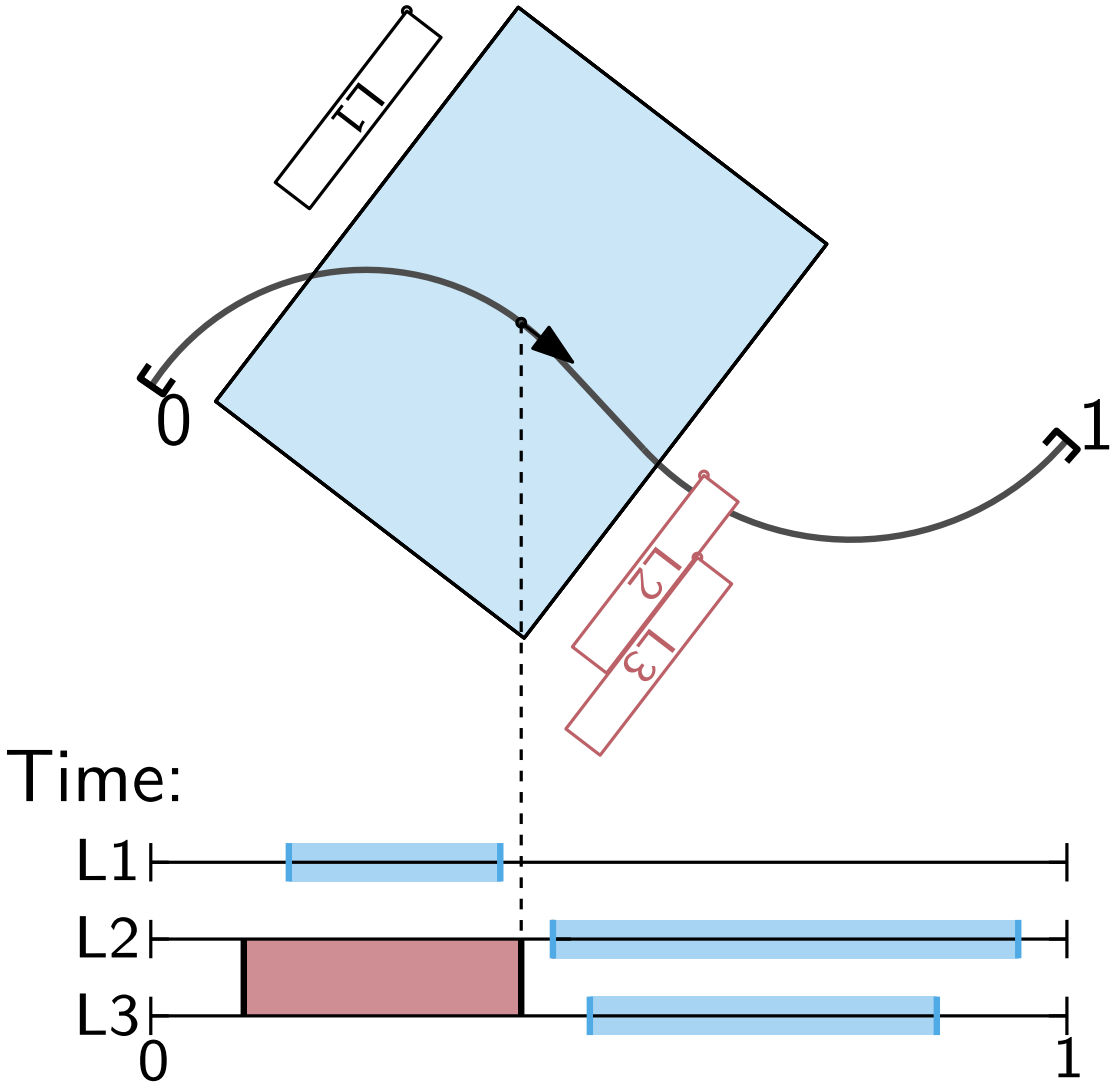
Mapping to Intervals

Two labels are in **conflict** at time t if they intersect at time t



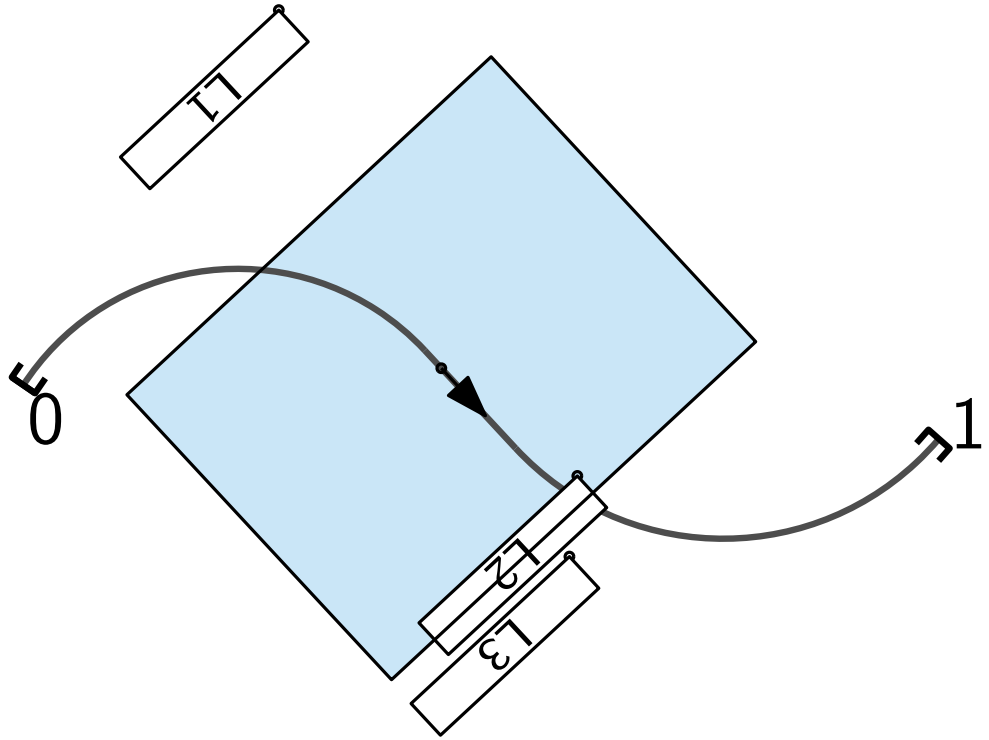
Mapping to Intervals

Two labels are in **conflict** at time t if they intersect at time t

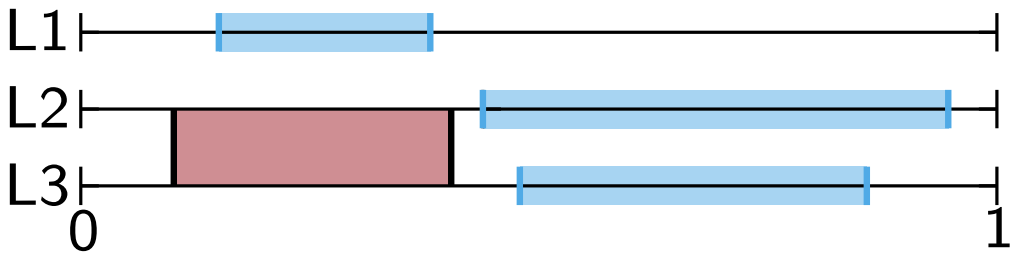


Mapping to Intervals

Two labels are in **conflict** at time t if they intersect at time t

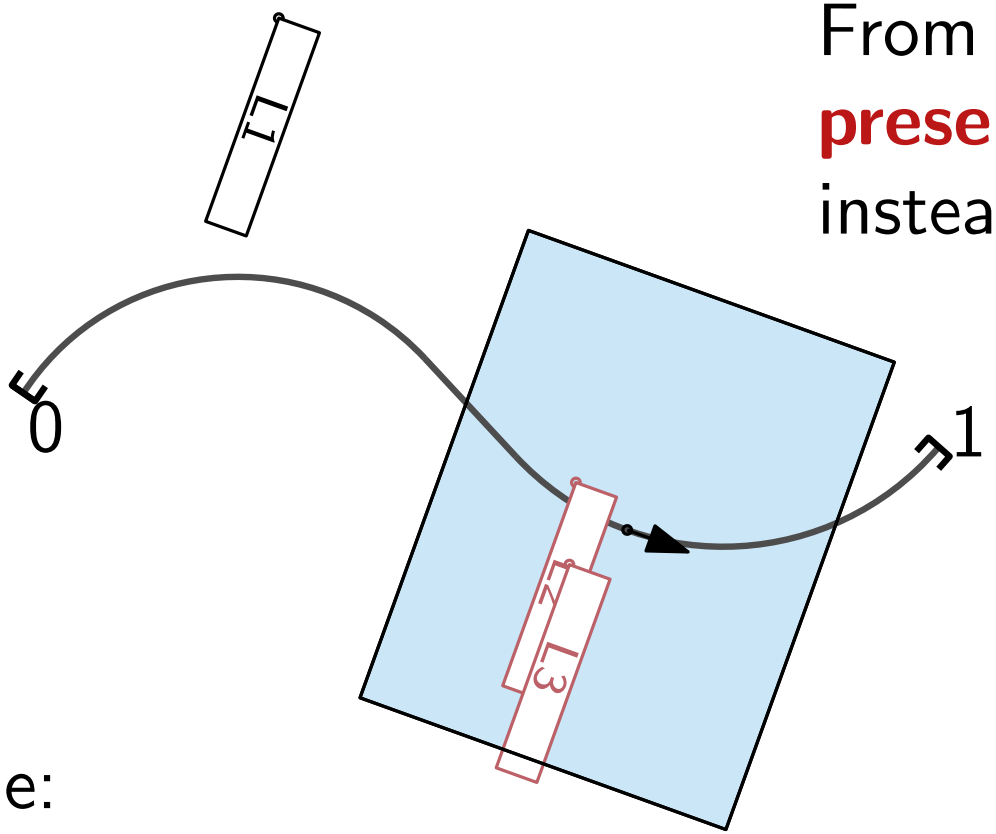


Time:



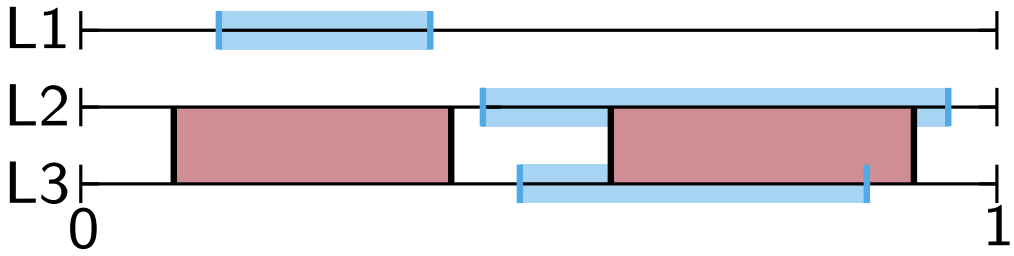
Mapping to Intervals

Two labels are in **conflict** at time t if they intersect at time t



From now on:
presence and conflict intervals
instead of labels.

Time:



set of **conflict intervals**

Problem Definition

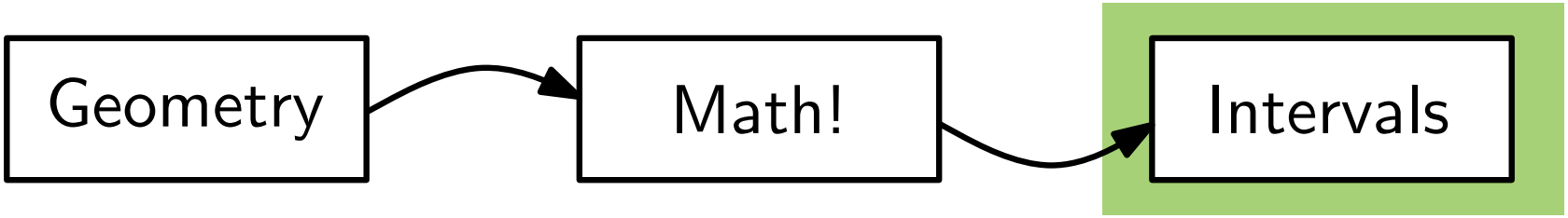
Overview

Input:
map m, set P of points, set L of corresponding labels

Output:
consistent rotation labeling that maximizes
the sum of all active ranges

Overview

Input:
map m, set P of points, set L of corresponding labels

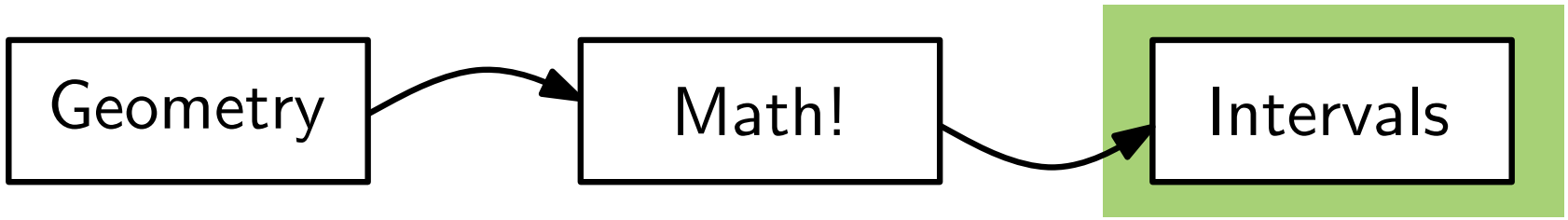


Output:
consistent rotation labeling that maximizes
the sum of all active ranges

Overview

Input:
map m, set P of points, set L of corresponding labels

Input':
set of presence intervals, conflicts



Output':
set of activity intervals

Output:
consistent rotation labeling that maximizes
the sum of all active ranges

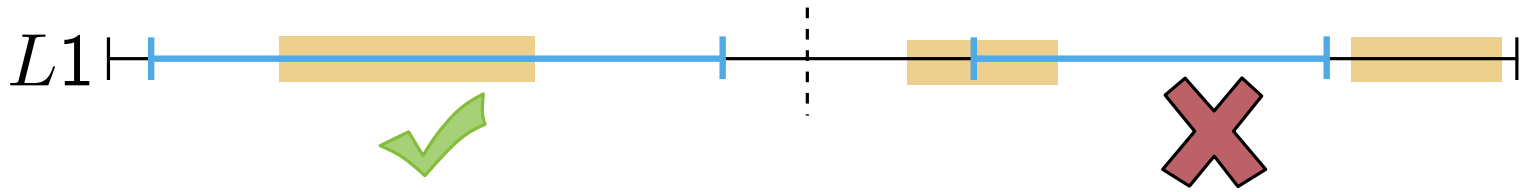
Activity of Labels

Label is **active** at time t if it is displayed at time t

Activity of Labels

Label is **active** at time t if it is displayed at time t

Only active if present

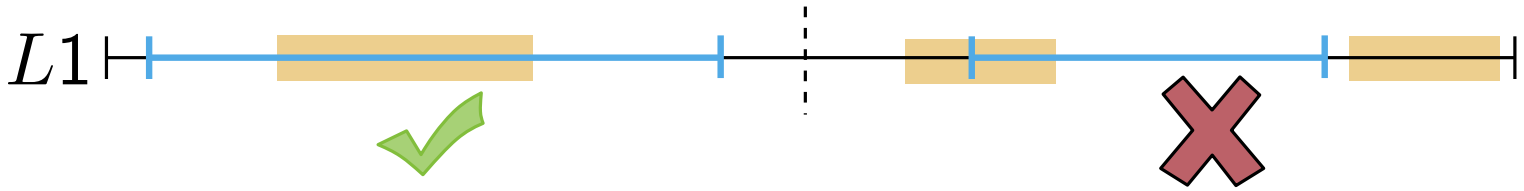


Label is present. Label is active. Labels are in conflict.

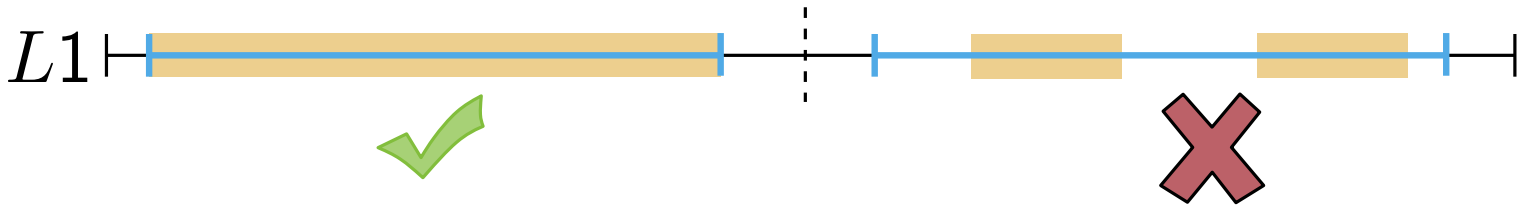
Activity of Labels

Label is **active** at time t if it is displayed at time t

Only active if present



Either active for the **full presence** intervall **or not** at all

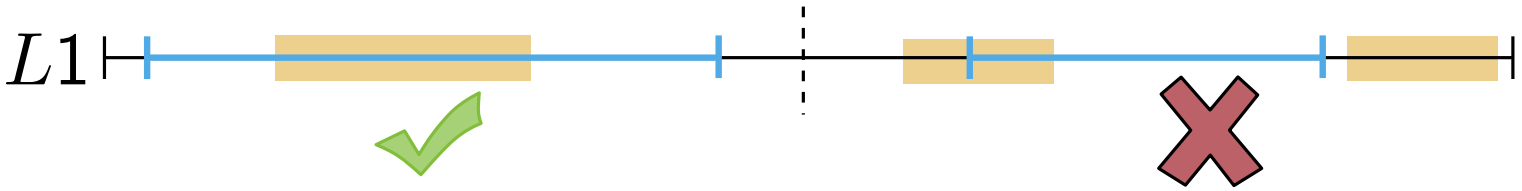


Label is present. Label is active. Labels are in conflict.

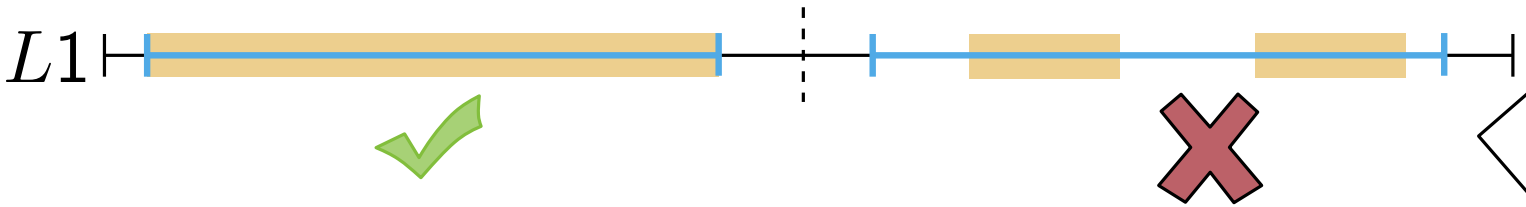
Activity of Labels

Label is **active** at time t if it is displayed at time t

Only active if present



Either active for the **full presence** intervall **or not** at all



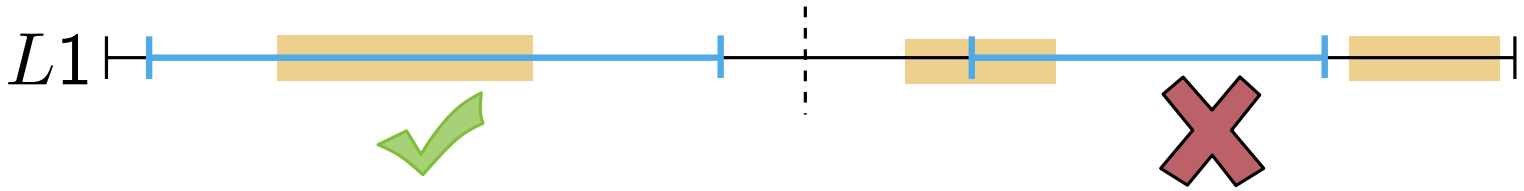
avoids flickering

Label is present. Label is active. Labels are in conflict.

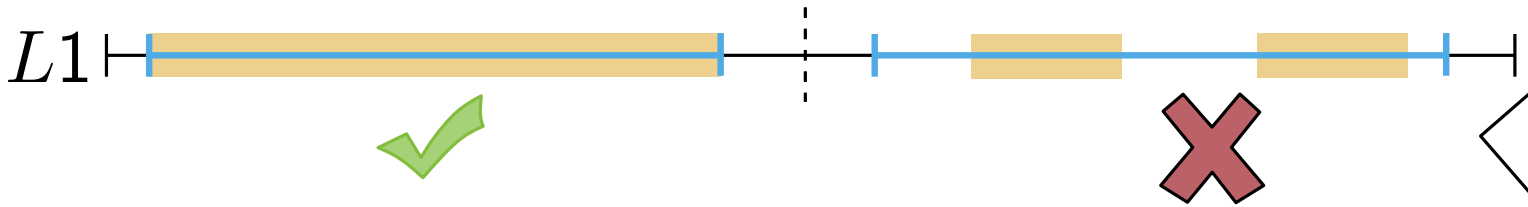
Activity of Labels

Label is **active** at time t if it is displayed at time t

Only active if present

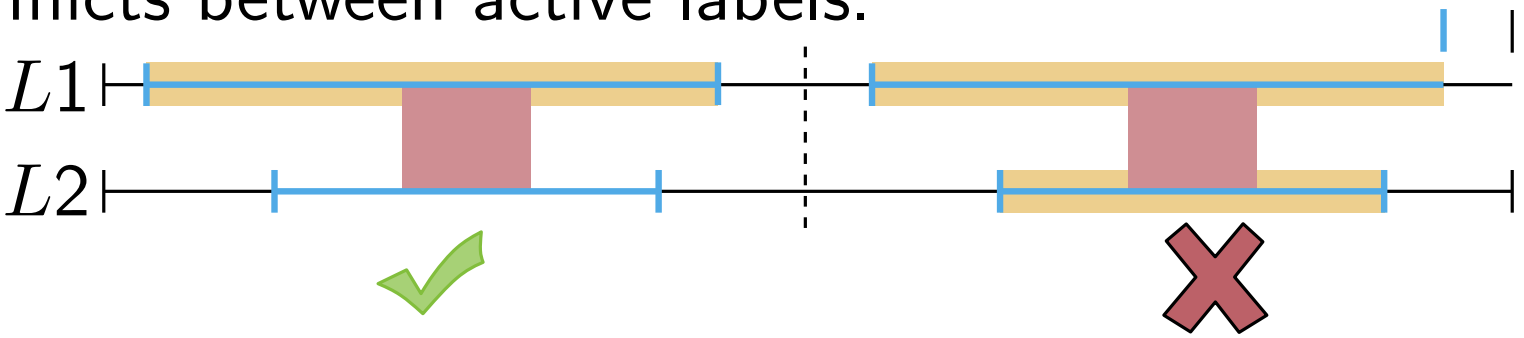


Either active for the **full presence** intervall **or not** at all



avoids flickering

No conflicts between active labels.

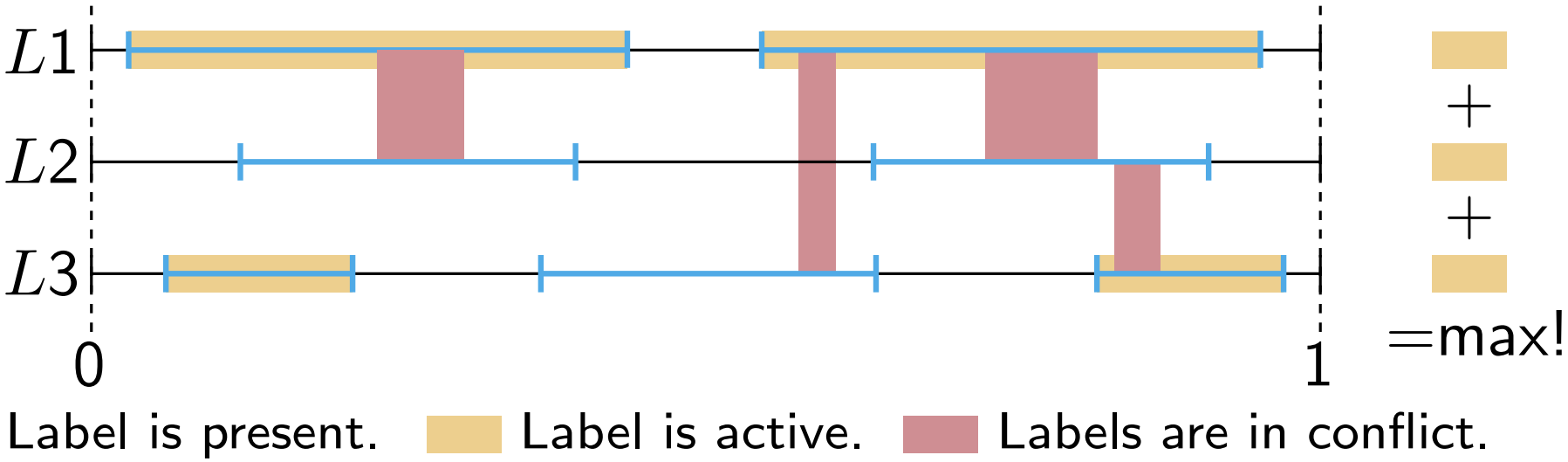


Label is present. Label is active. Labels are in conflict.

Problem Definition

Problem GeneralMaxTotal:
Given: Set I of presence intervals, conflicts
Find: Subset $J \subseteq I$ such that $\sum_{j \in J} \text{length}(j)$ is maximal, and J is conflict free

Set J of intervals is **conflict free** if no two in J intervals are in conflict



Problem Definition

Problem GeneralMaxTotal:
Given: Set I of presence intervals, conflicts
Find: Subset $J \subseteq I$ such that $\sum_{j \in J} \text{length}(j)$ is maximal, and J is conflict free

Set J of intervals is **conflict free** if no two in J intervals are in conflict

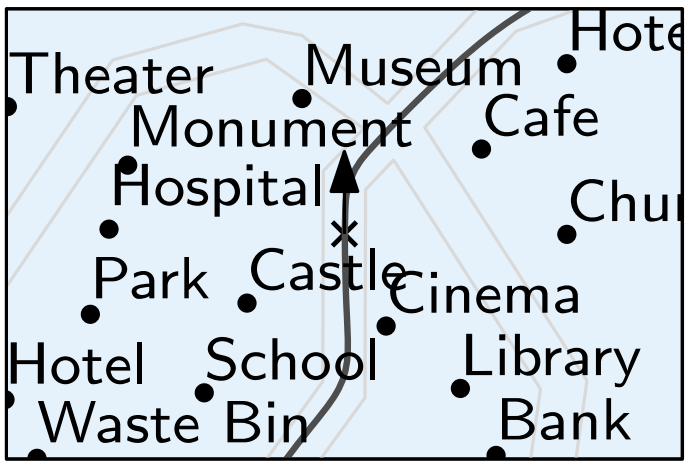
Theorem 5

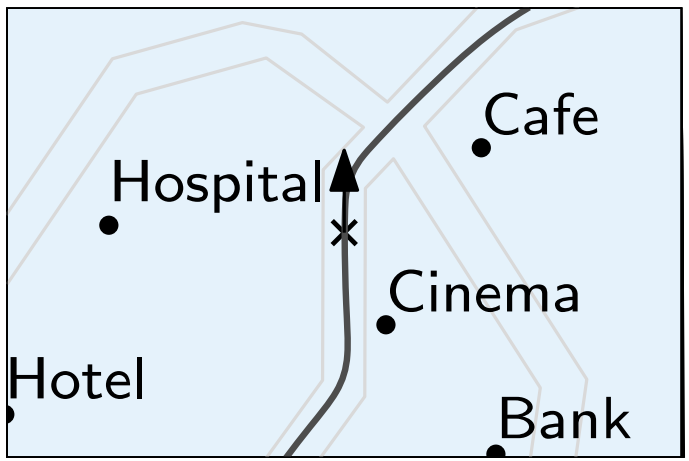
GeneralMaxTotal is NP-complete

Problem Definition

Problem k-RestrictedMaxTotal:
Given: Set I of presence intervals, conflicts
Find: Subset $J \subseteq I$ such that $\sum_{j \in J} \text{length}(j)$ is maximal, and J is conflict free, and **there is no t that is contained in more than k intervals in J**.

Set J of intervals is **conflict free** if no two in J intervals are in conflict





$k = 5$

k-RestrictedMaxTotal

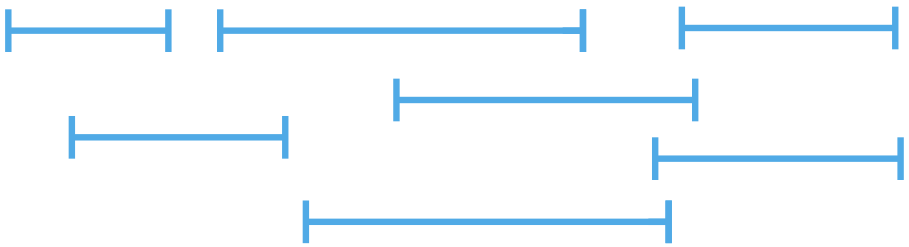
Theorem 4

k-RestrictedMaxTotal can be solved in polynomial time

Case $\mathbf{k = 1}$: Problem is equivalent to computing a maximum weighted independent set on an **interval graph**

$\Rightarrow O(n)$ time if intervals are sorted

[Hsiao et al., 1992]



k-RestrictedMaxTotal

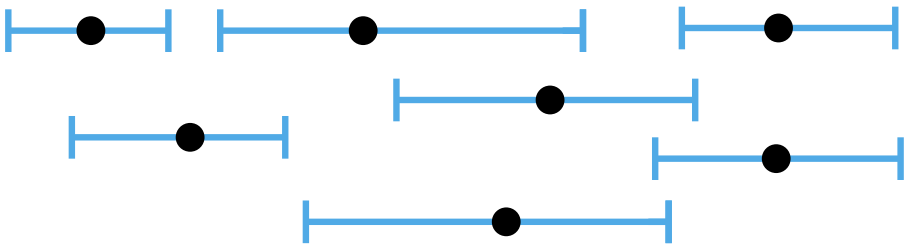
Theorem 4

k-RestrictedMaxTotal can be solved in polynomial time

Case $\mathbf{k = 1}$: Problem is equivalent to computing a maximum weighted independent set on an **interval graph**

$\Rightarrow O(n)$ time if intervals are sorted

[Hsiao et al., 1992]



k-RestrictedMaxTotal

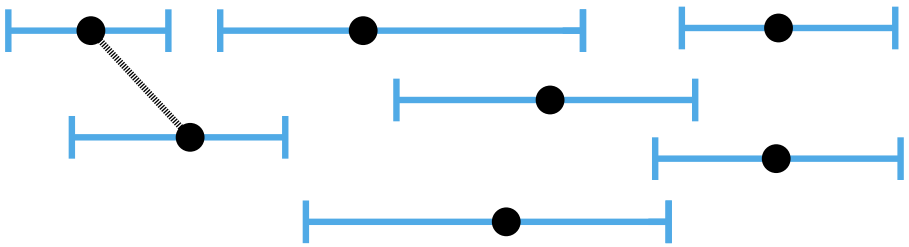
Theorem 4

k-RestrictedMaxTotal can be solved in polynomial time

Case $\mathbf{k = 1}$: Problem is equivalent to computing a maximum weighted independent set on an **interval graph**

$\Rightarrow O(n)$ time if intervals are sorted

[Hsiao et al., 1992]



k-RestrictedMaxTotal

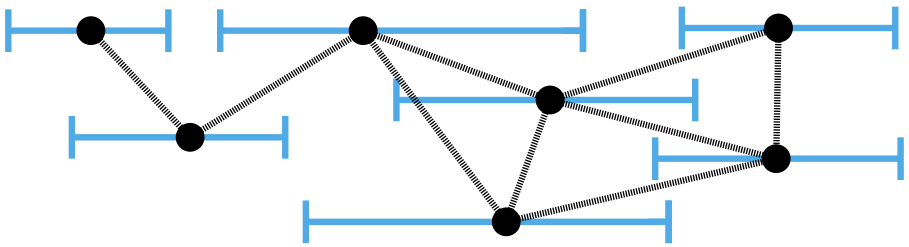
Theorem 4

k-RestrictedMaxTotal can be solved in polynomial time

Case $\mathbf{k = 1}$: Problem is equivalent to computing a maximum weighted independent set on an **interval graph**

$\Rightarrow O(n)$ time if intervals are sorted

[Hsiao et al., 1992]



k-RestrictedMaxTotal

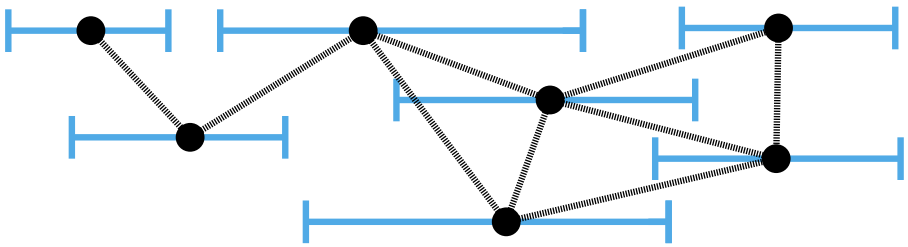
Theorem 4

k-RestrictedMaxTotal can be solved in polynomial time

Case $\mathbf{k = 1}$: Problem is equivalent to computing a maximum weighted independent set on an **interval graph**

$\Rightarrow O(n)$ time if intervals are sorted

[Hsiao et al., 1992]



weight of a vertex = length of its interval

k-RestrictedMaxTotal

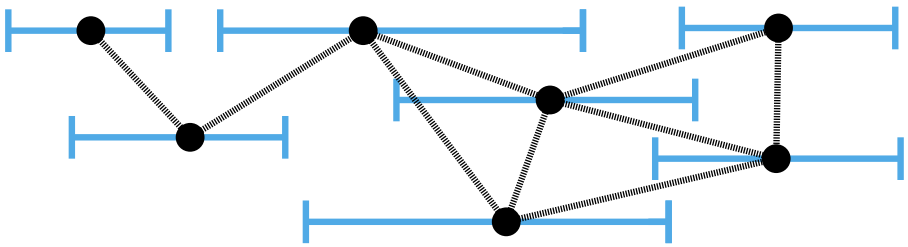
Theorem 4

k-RestrictedMaxTotal can be solved in polynomial time

Case $\mathbf{k = 1}$: Problem is equivalent to computing a maximum weighted independent set on an **interval graph**

$\Rightarrow O(n)$ time if intervals are sorted

[Hsiao et al., 1992]



weight of a vertex = length of its interval

s. Übungsblatt

k-RestrictedMaxTotal

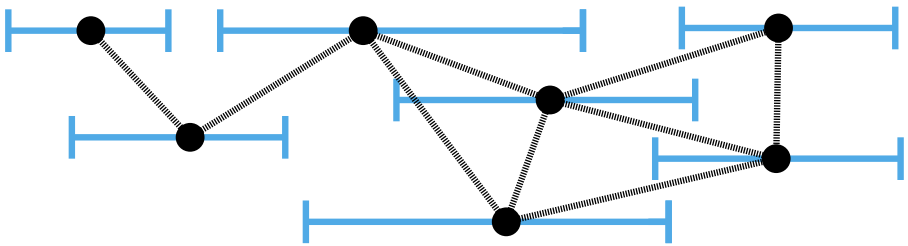
Theorem 4

k-RestrictedMaxTotal can be solved in polynomial time

Case $\mathbf{k = 1}$: Problem is equivalent to computing a maximum weighted independent set on an **interval graph**

$\Rightarrow O(n)$ time if intervals are sorted

[Hsiao et al., 1992]



weight of a vertex = length of its interval

s. Übungsblatt

Problem is related to **k-coloring** of interval graphs

There are $O(n + k)/O(kS(n))$ algorithms

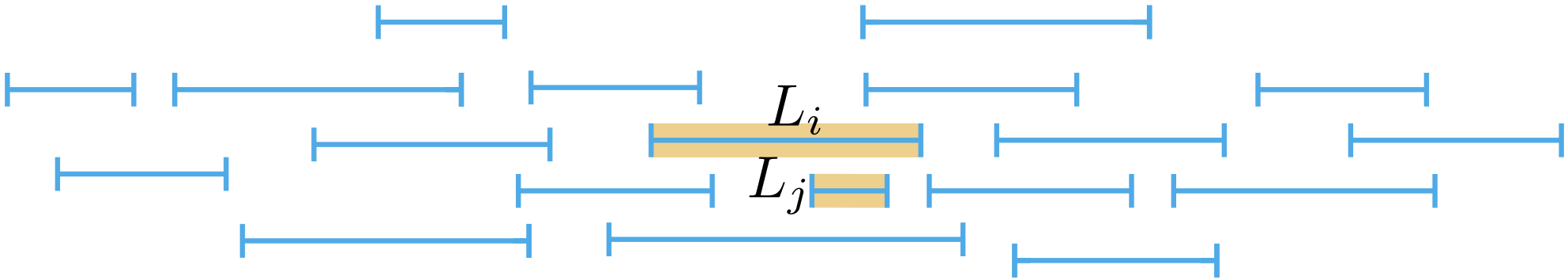
[Carlisle, Lloyd '93]

k-RestrictedMaxTotal

Theorem 4

k-RestrictedMaxTotal can be solved in polynomial time

Case $\mathbf{k = 2}$: Because of **conflicts** more difficult than $k = 1$

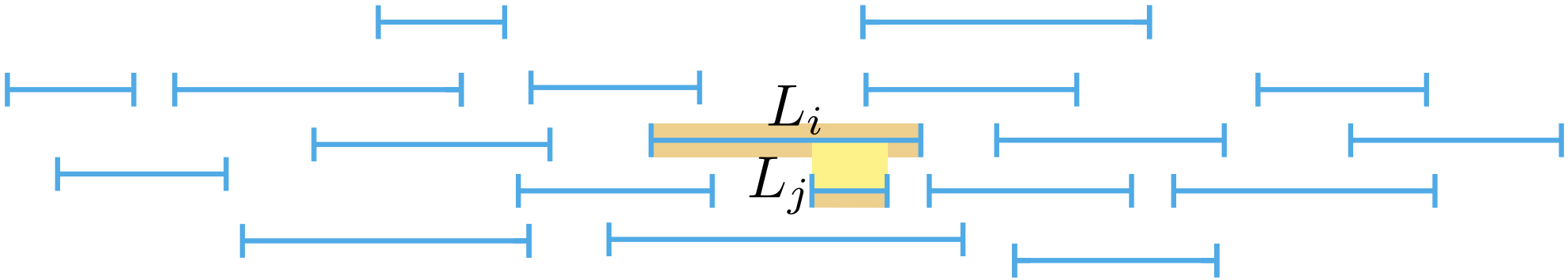


k-RestrictedMaxTotal

Theorem 4

k-RestrictedMaxTotal can be solved in polynomial time

Case $\mathbf{k = 2}$: Because of **conflicts** more difficult than $k = 1$

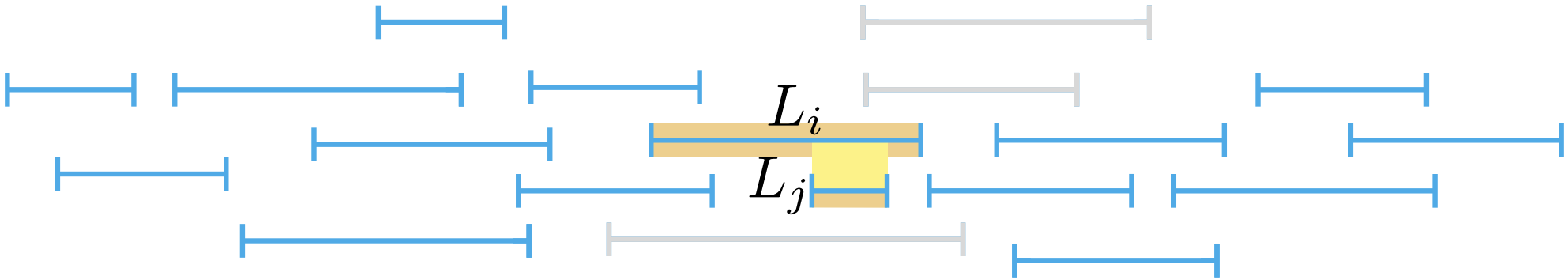


k-RestrictedMaxTotal

Theorem 4

k-RestrictedMaxTotal can be solved in polynomial time

Case $\mathbf{k = 2}$: Because of **conflicts** more difficult than $k = 1$

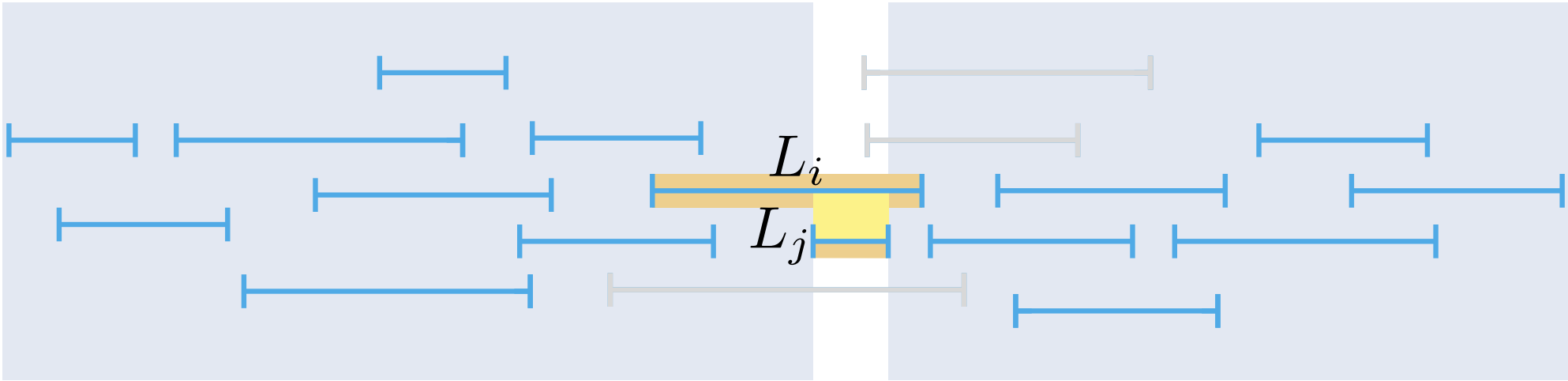


k-RestrictedMaxTotal

Theorem 4

k-RestrictedMaxTotal can be solved in polynomial time

Case $\mathbf{k = 2}$: Because of **conflicts** more difficult than $k = 1$



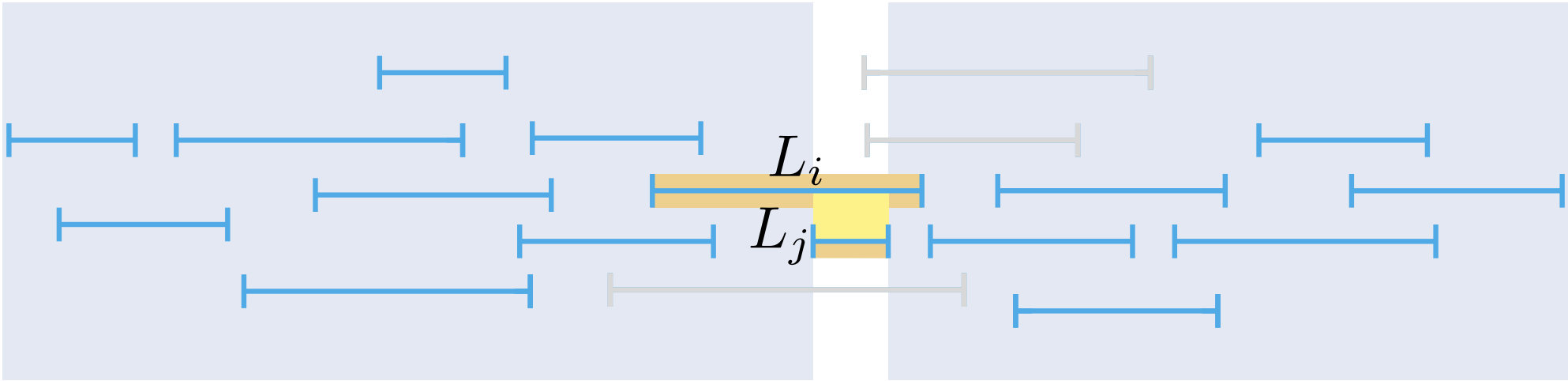
k-RestrictedMaxTotal

Theorem 4

k-RestrictedMaxTotal can be solved in polynomial time

Case $\mathbf{k = 2}$: Because of **conflicts** more difficult than $k = 1$

A **separating tuple** are two non-conflicting, but overlapping intervals



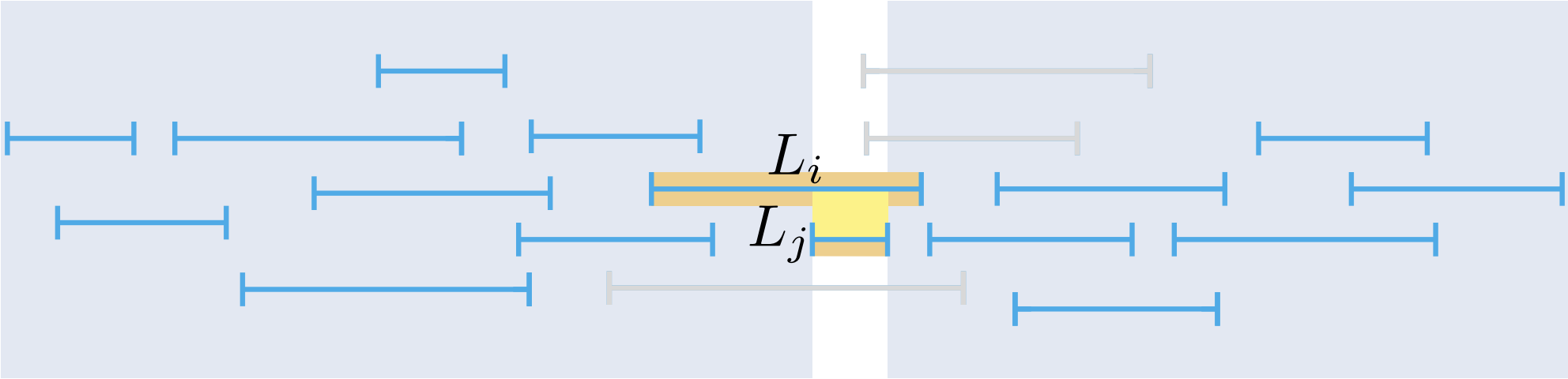
k-RestrictedMaxTotal

Theorem 4

k-RestrictedMaxTotal can be solved in polynomial time

Case $\mathbf{k = 2}$: Because of **conflicts** more difficult than $k = 1$

A **separating tuple** are two non-conflicting, but overlapping intervals



Observation:
a separating tuple splits the instance into two sub-instances

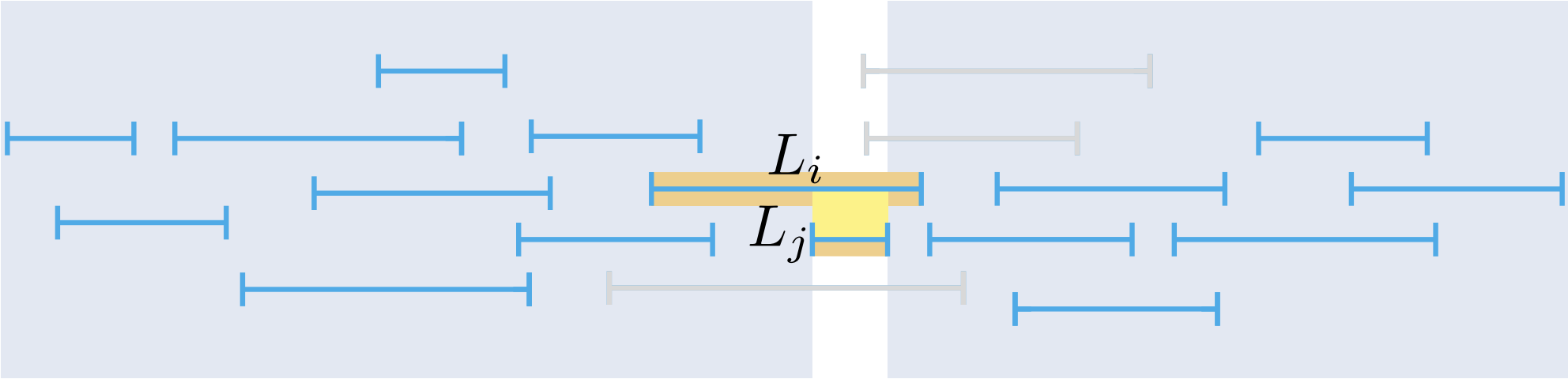
k-RestrictedMaxTotal

Theorem 4

k-RestrictedMaxTotal can be solved in polynomial time

Case $\mathbf{k = 2}$: Because of **conflicts** more difficult than $k = 1$

A **separating tuple** are two non-conflicting, but overlapping intervals



Observation:
a separating tuple splits the instance into two sub-instances

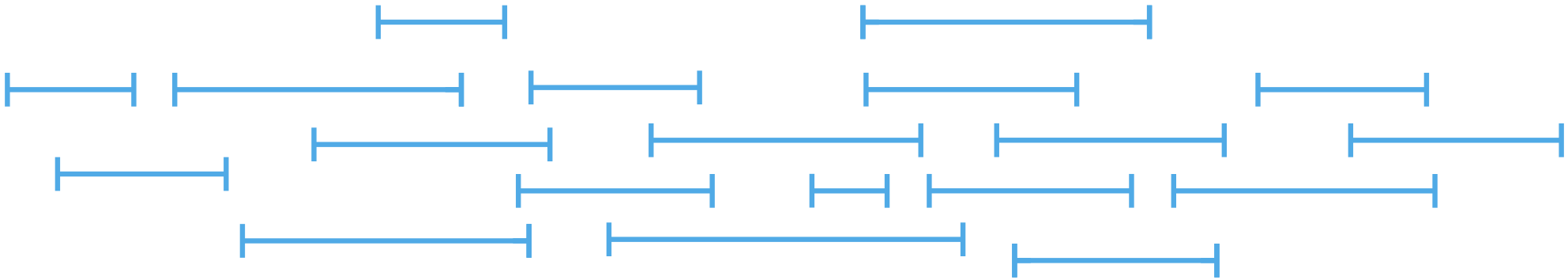
Dynamic Programming

k-RestrictedMaxTotal

Theorem 4

k-RestrictedMaxTotal can be solved in polynomial time

Case $\mathbf{k = 2}$:

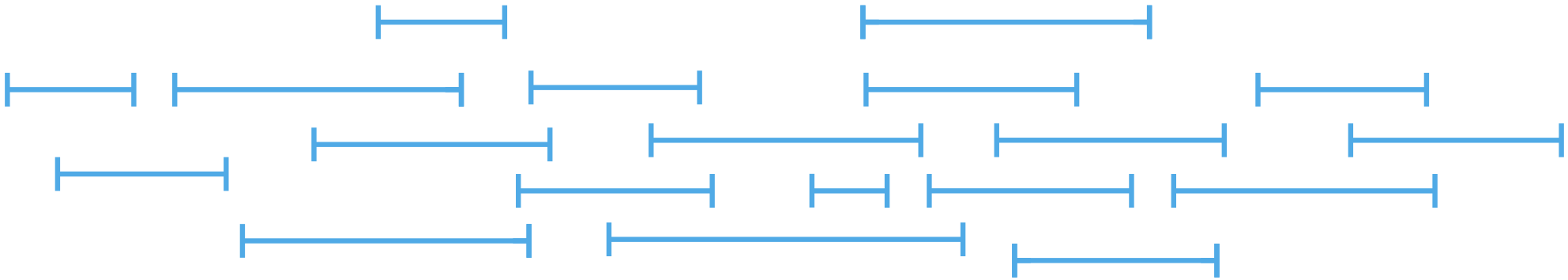


k-RestrictedMaxTotal

Theorem 4

k-RestrictedMaxTotal can be solved in polynomial time

Case $\mathbf{k = 2}$:



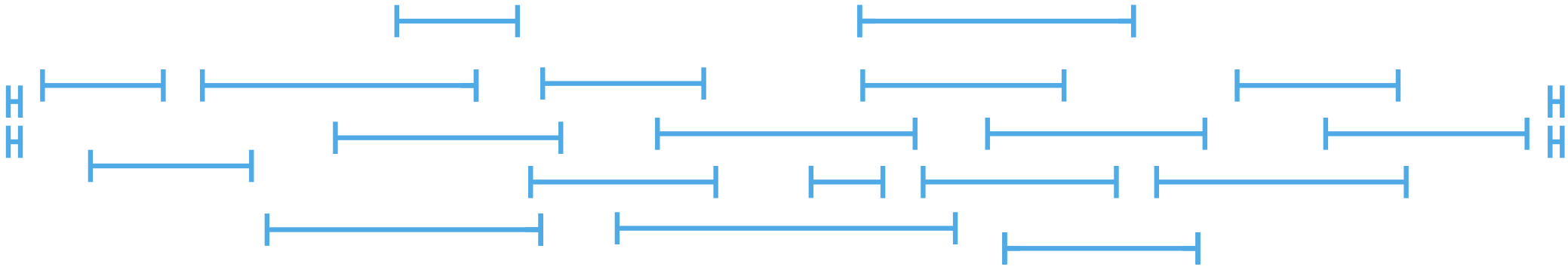
k-RestrictedMaxTotal

Theorem 4

k-RestrictedMaxTotal can be solved in polynomial time

Case $\mathbf{k = 2}$:

- add two dummy separating tuple



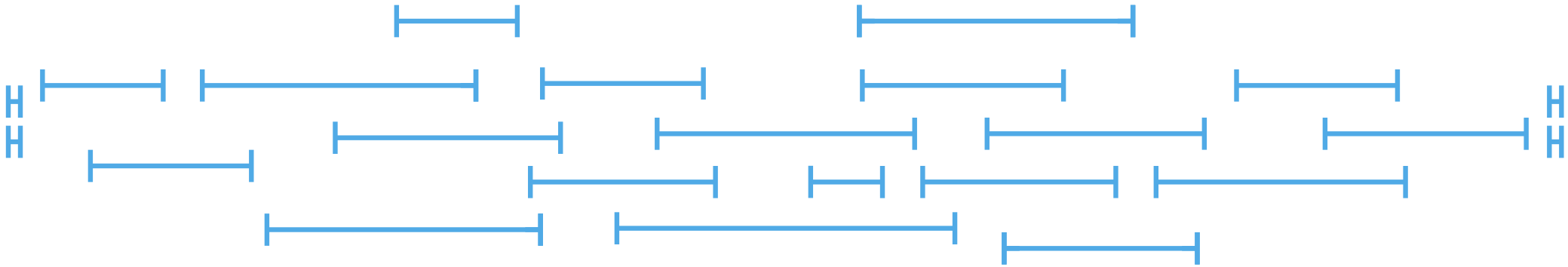
k-RestrictedMaxTotal

Theorem 4

k-RestrictedMaxTotal can be solved in polynomial time

Case $\mathbf{k = 2}$:

- add two dummy separating tuple
- enumerate all separating tuples s_i and sort them
- maintain one-dimensional table $\mathcal{T}$



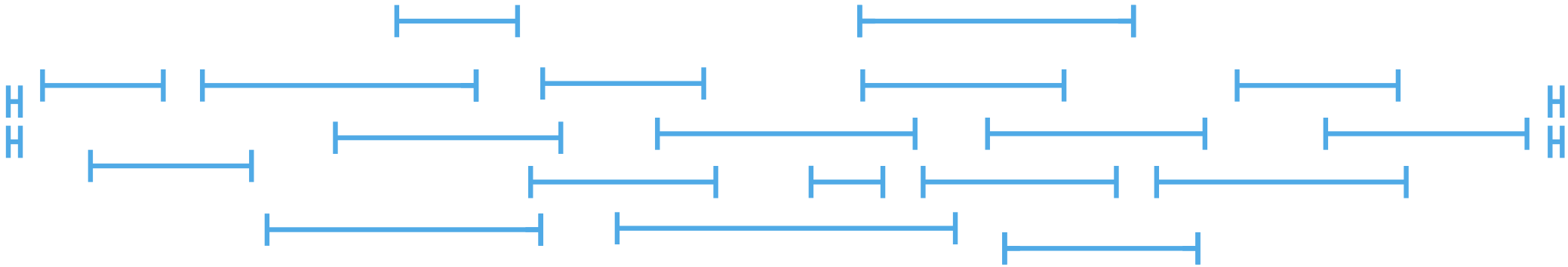
k-RestrictedMaxTotal

Theorem 4

k-RestrictedMaxTotal can be solved in polynomial time

Case $\mathbf{k = 2}$:

- add two dummy separating tuple
- enumerate all separating tuples s_i and sort them
- maintain one-dimensional table $\mathcal{T}$



$\mathcal{T}[i]$: optimal solution for all $s_j, j < i$ that contains s_i.

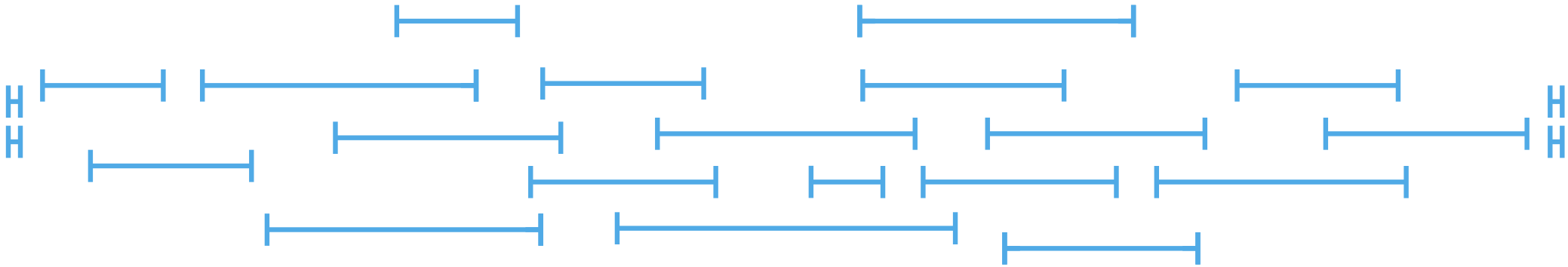
k-RestrictedMaxTotal

Theorem 4

k-RestrictedMaxTotal can be solved in polynomial time

Case $\mathbf{k = 2}$:

- add two dummy separating tuple
- enumerate all separating tuples s_i and sort them
- maintain one-dimensional table $\mathcal{T}$



$\mathcal{T}[i]$: optimal solution for all $s_j, j < i$ that contains s_i.

construct $\mathcal{T}$ from left to right by guessing **preceding** separating tuple

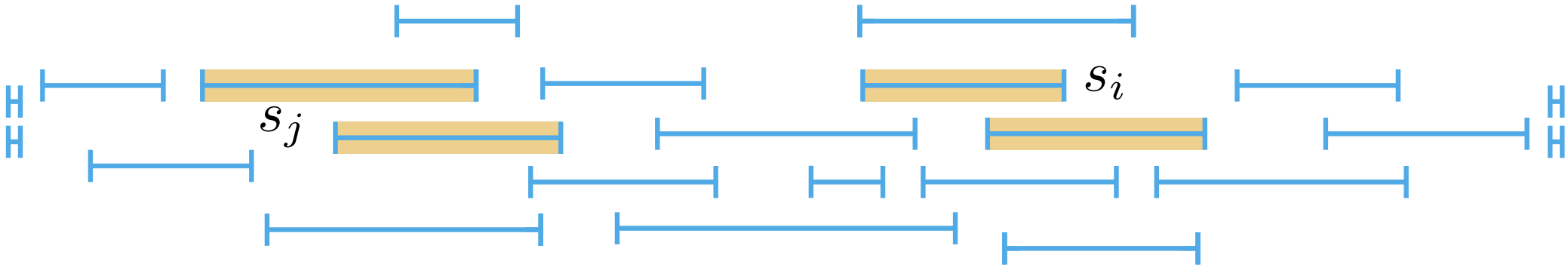
k-RestrictedMaxTotal

Theorem 4

k-RestrictedMaxTotal can be solved in polynomial time

Case $\mathbf{k = 2}$:

- add two dummy separating tuple
- enumerate all separating tuples s_i and sort them
- maintain one-dimensional table $\mathcal{T}$



$\mathcal{T}[i]$: optimal solution for all $s_j, j < i$ that contains s_i.

construct $\mathcal{T}$ from left to right by guessing **preceding** separating tuple

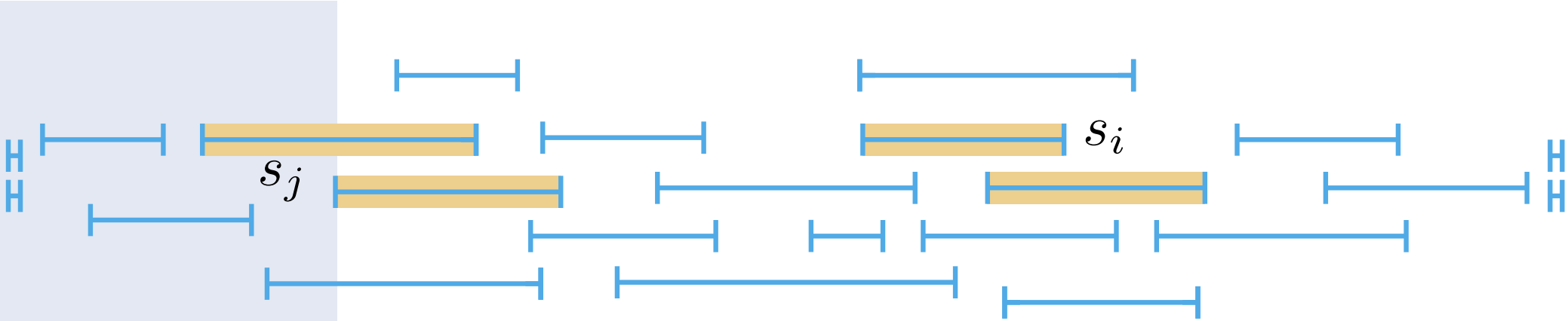
k-RestrictedMaxTotal

Theorem 4

k-RestrictedMaxTotal can be solved in polynomial time

Case $\mathbf{k = 2}$:

- add two dummy separating tuple
- enumerate all separating tuples s_i and sort them
- maintain one-dimensional table $\mathcal{T}$



$\mathcal{T}[i]$: optimal solution for all $s_j, j < i$ that contains s_i.

construct $\mathcal{T}$ from left to right by guessing **preceding** separating tuple

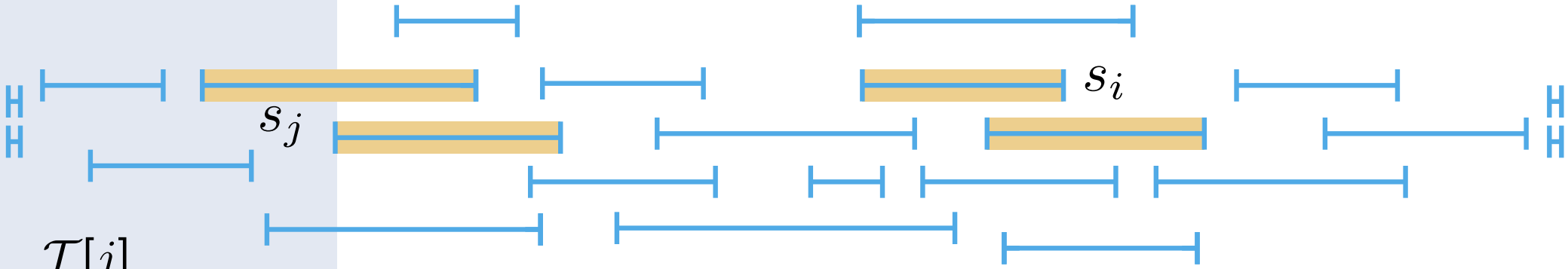
k-RestrictedMaxTotal

Theorem 4

k-RestrictedMaxTotal can be solved in polynomial time

Case $\mathbf{k = 2}$:

- add two dummy separating tuple
- enumerate all separating tuples s_i and sort them
- maintain one-dimensional table $\mathcal{T}$



$\mathcal{T}[i]$: optimal solution for all $s_j, j < i$ that contains s_i.

construct $\mathcal{T}$ from left to right by guessing **preceding** separating tuple

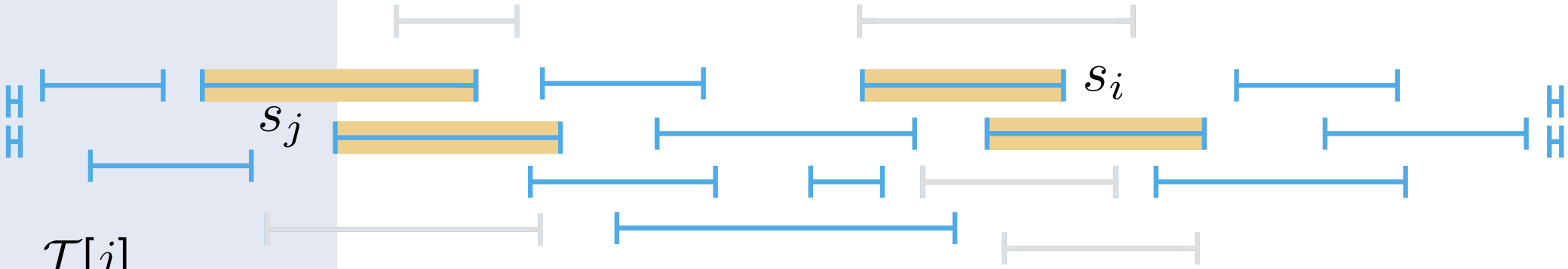
k-RestrictedMaxTotal

Theorem 4

k-RestrictedMaxTotal can be solved in polynomial time

Case $\mathbf{k = 2}$:

- add two dummy separating tuple
- enumerate all separating tuples s_i and sort them
- maintain one-dimensional table $\mathcal{T}$



$\mathcal{T}[i]$: optimal solution for all $s_j, j < i$ that contains s_i.

construct $\mathcal{T}$ from left to right by guessing **preceding** separating tuple

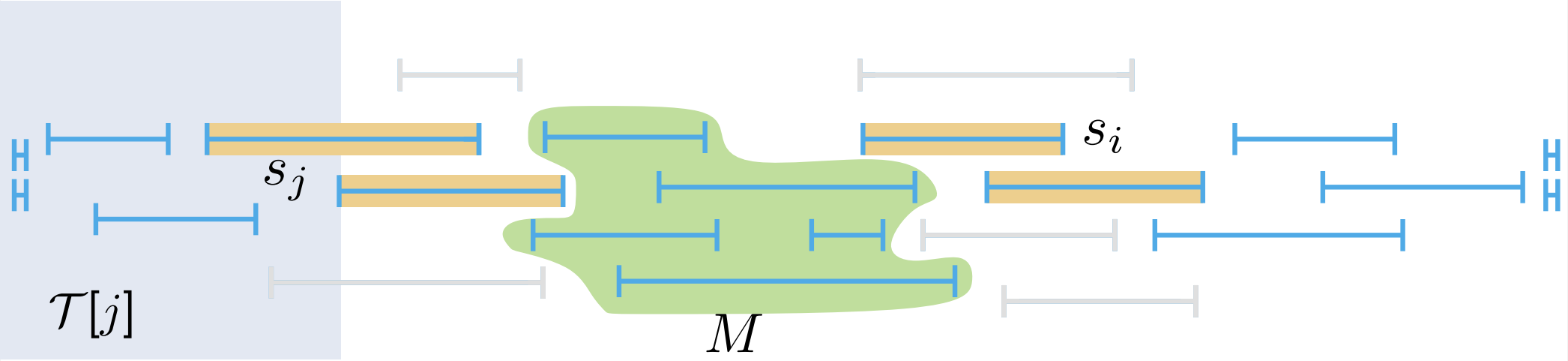
k-RestrictedMaxTotal

Theorem 4

k-RestrictedMaxTotal can be solved in polynomial time

Case $\mathbf{k = 2}$:

- add two dummy separating tuple
- enumerate all separating tuples s_i and sort them
- maintain one-dimensional table $\mathcal{T}$



$\mathcal{T}[i]$: optimal solution for all $s_j, j < i$ that contains s_i.

construct $\mathcal{T}$ from left to right by guessing **preceding** separating tuple

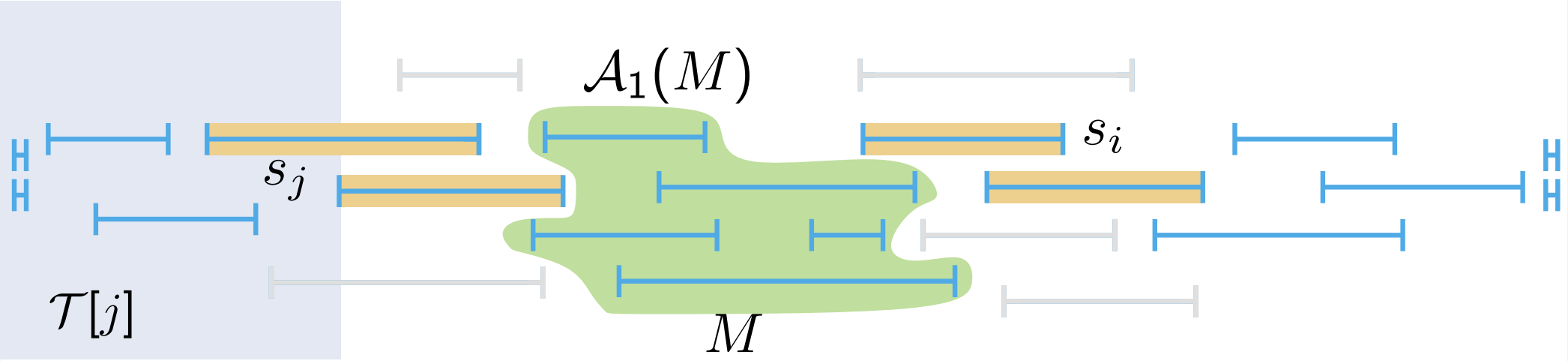
k-RestrictedMaxTotal

Theorem 4

k-RestrictedMaxTotal can be solved in polynomial time

Case $\mathbf{k = 2}$:

- add two dummy separating tuple
- enumerate all separating tuples s_i and sort them
- maintain one-dimensional table $\mathcal{T}$



$\mathcal{T}[i]$: optimal solution for all $s_j, j < i$ that contains s_i.

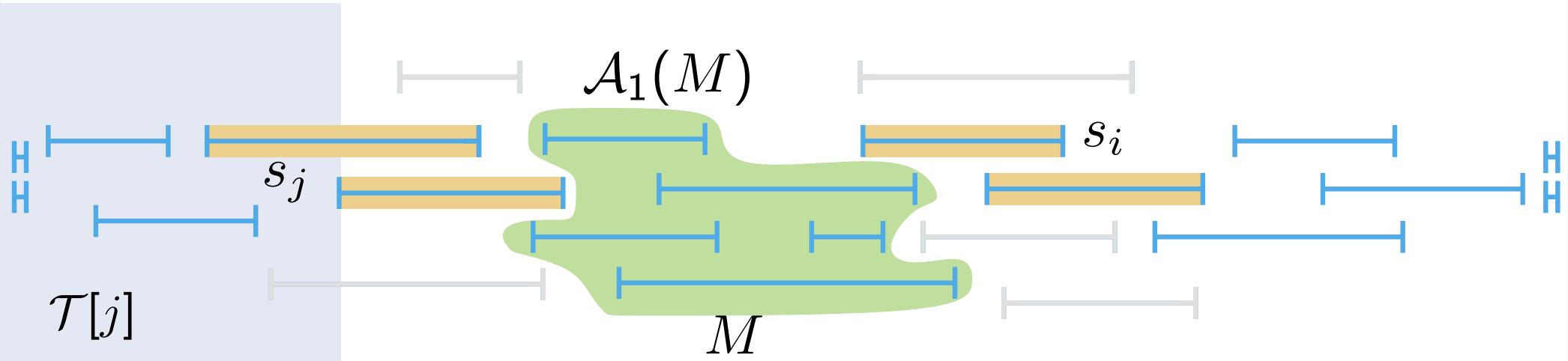
construct $\mathcal{T}$ from left to right by guessing **preceding** separating tuple

k-RestrictedMaxTotal

Theorem 4

k-RestrictedMaxTotal can be solved in polynomial time

Case $\mathbf{k = 2}$:

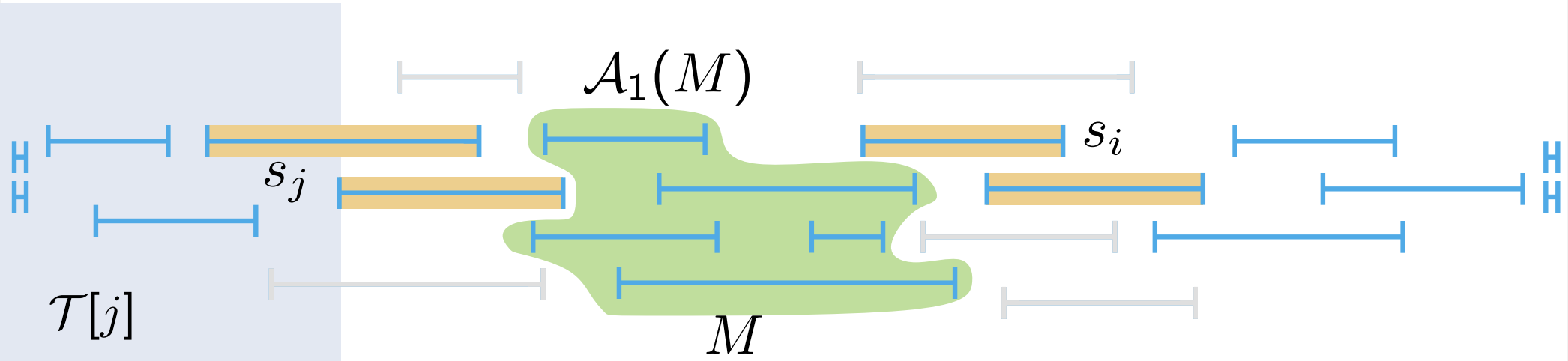


k-RestrictedMaxTotal

Theorem 4

k-RestrictedMaxTotal can be solved in polynomial time

Case $\mathbf{k = 2}$:



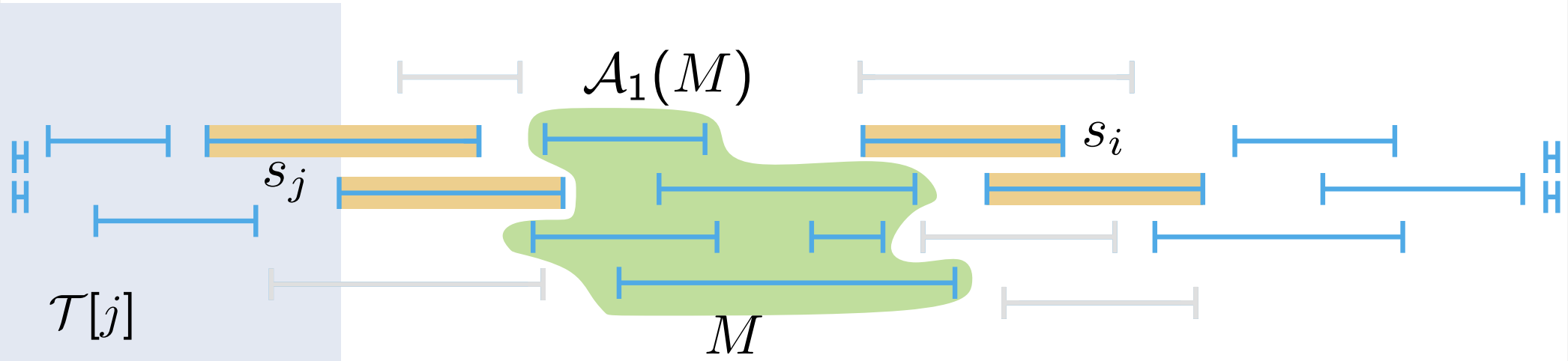
$$T[i] = \max_{j<i}\{T[j] + \mathcal{A}_1(M) + \text{length}(s_i) \mid s_i, s_j \text{ compatible } \}$$

k-RestrictedMaxTotal

Theorem 4

k-RestrictedMaxTotal can be solved in polynomial time

Case $\mathbf{k = 2}$:



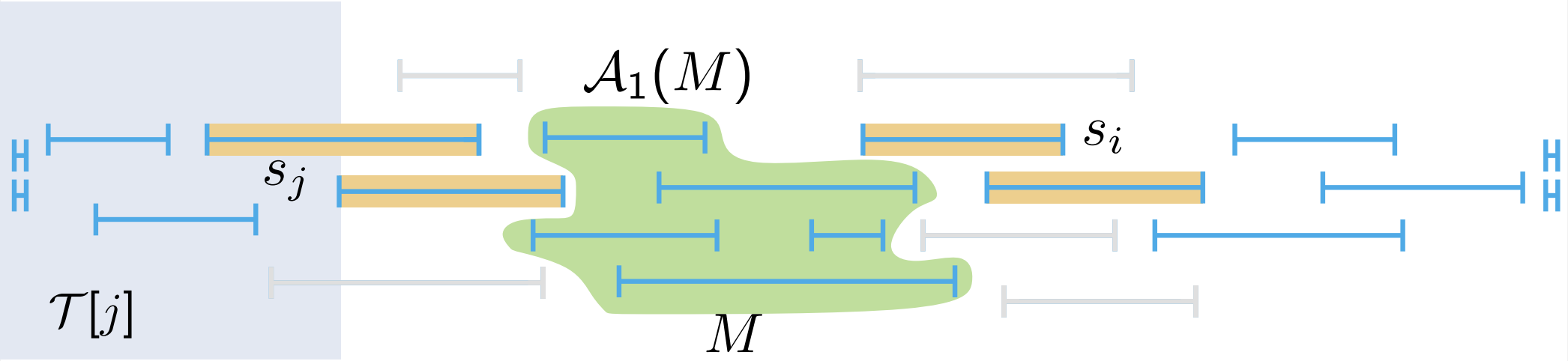
$$T[i] = \max_{j<i}\{\underbrace{T[j]}_{O(1)} + \underbrace{\mathcal{A}_1(M)}_{O(n)} + \underbrace{\text{length}(s_i)}_{O(1)} \mid s_i, s_j \text{ compatible } \}$$

k-RestrictedMaxTotal

Theorem 4

k-RestrictedMaxTotal can be solved in polynomial time

Case $\mathbf{k = 2}$:



$$T[i] = \max_{j<i}\{\underbrace{T[j]}_{O(1)} + \underbrace{\mathcal{A}_1(M)}_{O(n)} + \underbrace{\text{length}(s_i)}_{O(1)} \mid s_i, s_j \text{ compatible } \}$$

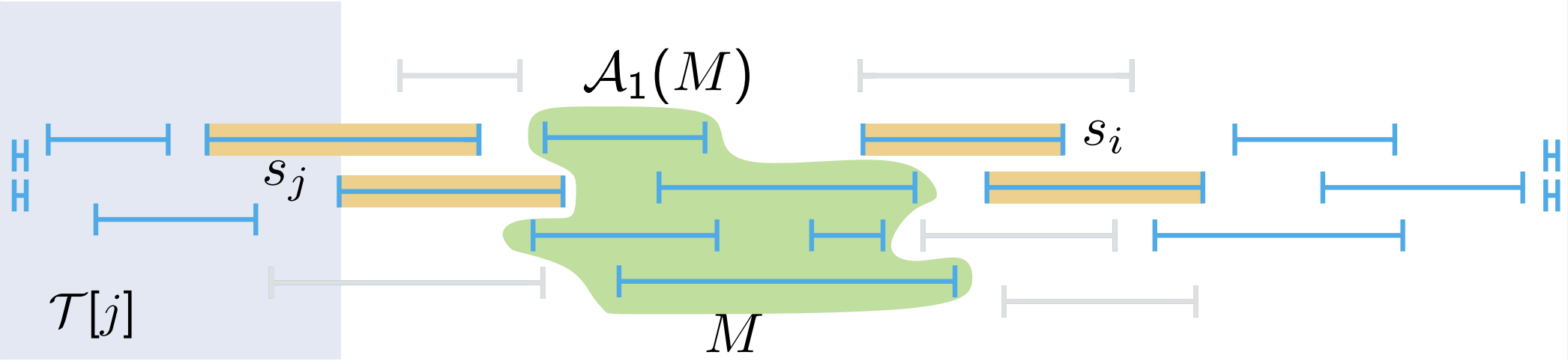
$O(\#\text{sep. tu.})$

k-RestrictedMaxTotal

Theorem 4

k-RestrictedMaxTotal can be solved in polynomial time

Case $\mathbf{k = 2}$:



$O(\#\text{sep. tu.})$

$$T[i] = \max_{j<i}\{T[j] + \mathcal{A}_1(M) + \text{length}(s_i) \mid s_i, s_j \text{ compatible } \}$$

$O(1)$ $O(n)$ $O(1)$

$O(\#\text{sep. tu.})$

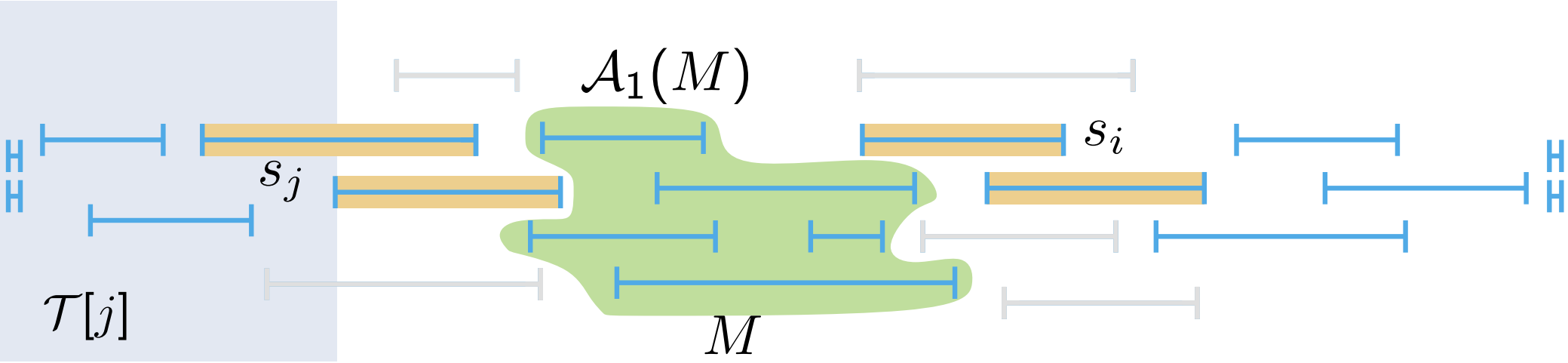
k-RestrictedMaxTotal

Theorem 4

k-RestrictedMaxTotal can be solved in polynomial time

Case $\mathbf{k = 2}$:

Time complexity: $O(n \cdot (\#\text{sep. tu.})^2)$



$$T[i] = \max_{j<i}\{\underbrace{T[j]}_{O(1)} + \underbrace{\mathcal{A}_1(M)}_{O(n)} + \underbrace{\text{length}(s_i)}_{O(1)} \mid s_i, s_j \text{ compatible } \}$$

$O(\#\text{sep. tu.})$

$O(\#\text{sep. tu.})$

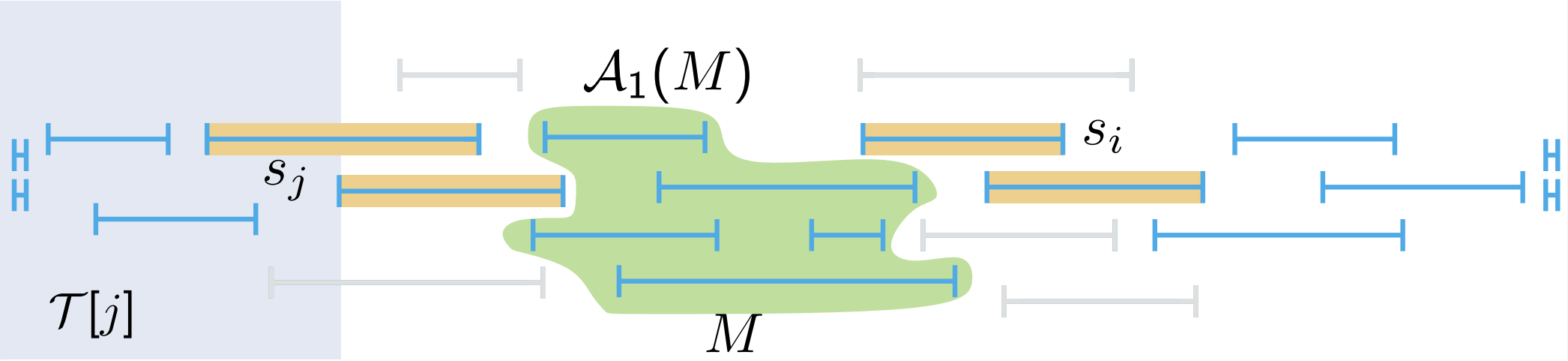
k-RestrictedMaxTotal

Theorem 4

k-RestrictedMaxTotal can be solved in polynomial time

Case $\mathbf{k = 2}$:

Time complexity: $O(n \cdot (\#\text{sep. tu.})^2) \quad = O(n^5)$



$$T[i] = \max_{j<i}\{\underbrace{T[j]}_{O(1)} + \underbrace{\mathcal{A}_1(M)}_{O(n)} + \underbrace{\text{length}(s_i)}_{O(1)} \mid s_i, s_j \text{ compatible } \}$$

$O(\#\text{sep. tu.})$

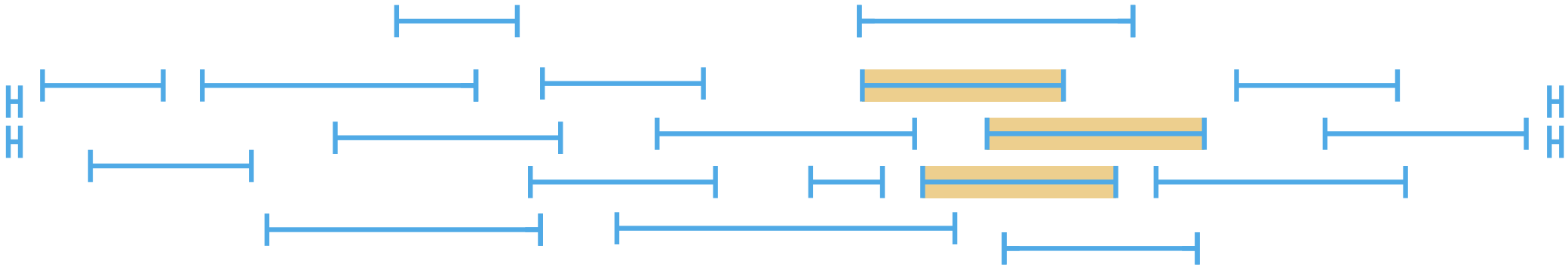
$O(\#\text{sep. tu.})$

k-RestrictedMaxTotal

Theorem 4

k-RestrictedMaxTotal can be solved in polynomial time

Case $\mathbf{k = 3}$:

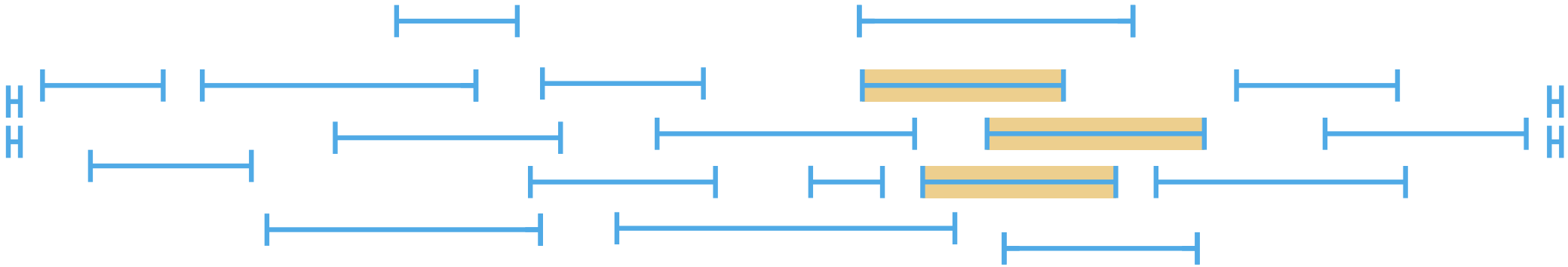


k-RestrictedMaxTotal

Theorem 4

k-RestrictedMaxTotal can be solved in polynomial time

Case $\mathbf{k = 3}$:



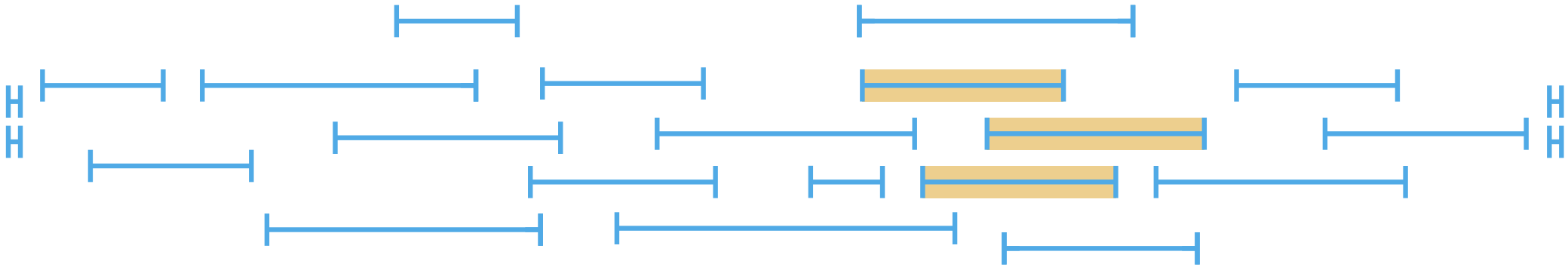
$$T[i] = \max_{j<i}\{T[j] + \mathcal{A}_2(M) + \text{length}(s_i) \mid s_i, s_j \text{ compatible } \}$$

k-RestrictedMaxTotal

Theorem 4

k-RestrictedMaxTotal can be solved in polynomial time

Case $\mathbf{k = 3}$:



Can be generalized to:

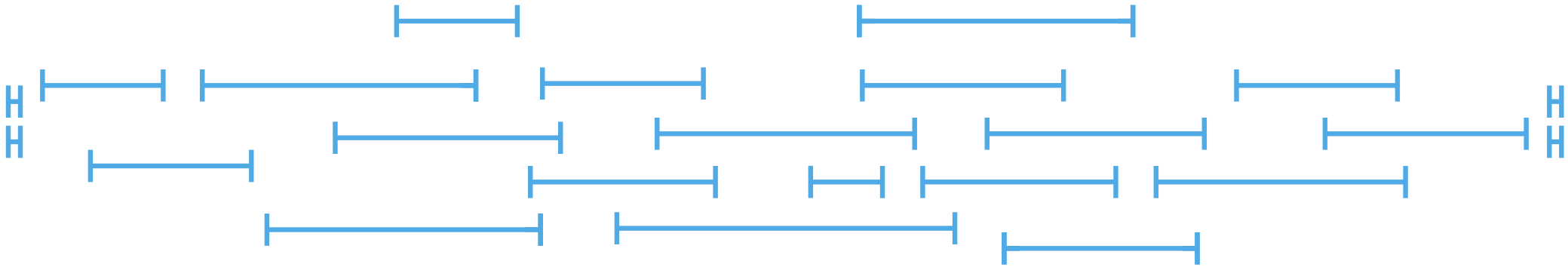
$T[i] = \max_{j<i}\{T[j] + \mathcal{A}_2(M) + \text{length}(s_i) \mid s_i, s_j \text{ compatible } \}$

k-RestrictedMaxTotal

Theorem 4

k-RestrictedMaxTotal can be solved in polynomial time

Case $\mathbf{k > 1}$:



Can be generalized to:

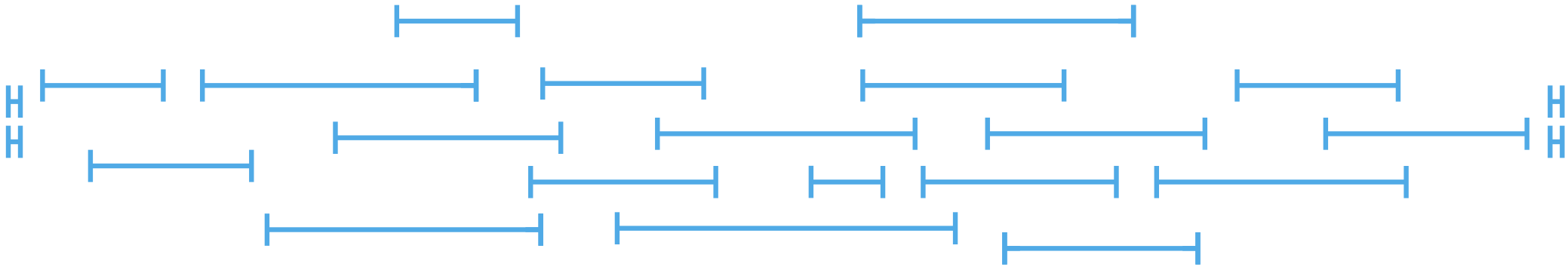
$T[i] = \max_{j<i}\{T[j] + \mathcal{A}_{k-1}(M) + \text{length}(s_i) \mid s_i, s_j \text{ compatible } \}$

k-RestrictedMaxTotal

Theorem 4

k-RestrictedMaxTotal can be solved in polynomial time

Case $\mathbf{k > 1}$:



Can be generalized to:

$T[i] = \max_{j<i}\{T[j] + \mathcal{A}_{k-1}(M) + \text{length}(s_i) \mid s_i, s_j \text{ compatible } \}$

Time complexity: $O(n^{k^2+k-1})$

Summary II

Our results:

- GeneralMaxTotal is $\mathcal{NP}$-complete
- k-RestrictedMaxTotal is polynomial solvable
- new labeling model is versatile
- algorithms **do not** rely on geometry

Open Problems:

- k-RestrictedMaxTotal is polynomial solvable but slow (how bad for real-world data sets?)
- trajectory-based labeling with **zooming**?