Indexing Uncertain Data

Pankaj K. Agarwal	Siu-Wing Cheng	Yufei Tao	Ke Yi
Duke University	HKUST	CUHK	HKUST

PODS '09

Motivation

- Two sessions devoted to uncertain/probabilistic data management in each of SIGMOD'08, VLDB'08, and SIGMOD'09

Motivation

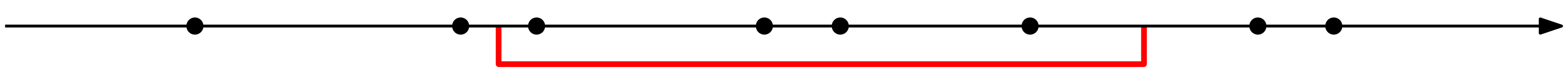
- Two sessions devoted to uncertain/probabilistic data management in each of SIGMOD'08, VLDB'08, and SIGMOD'09

- So, let's skip the cliché and just get to the problem

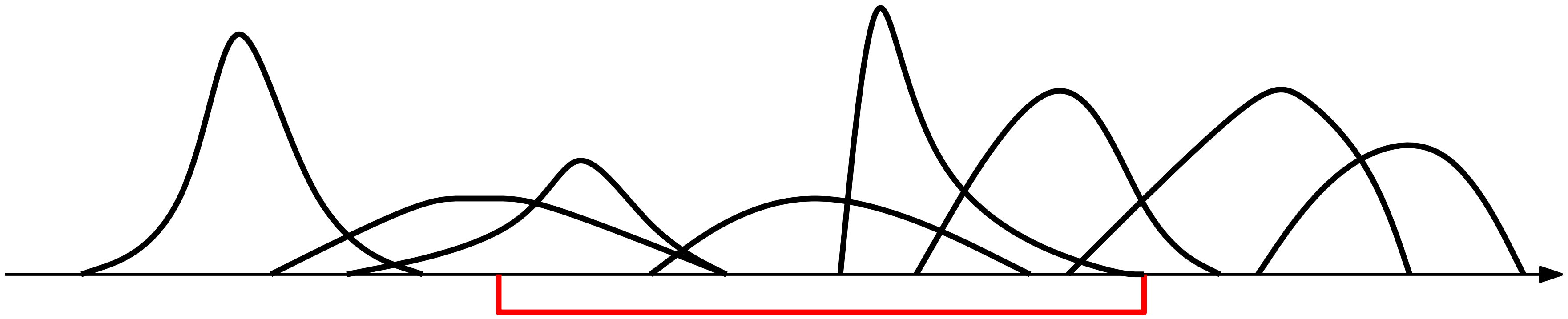
The Problem: Range Searching

- One of the very first and fundamental problems studied in querying uncertain data

The certain case:



The uncertain case:



Report all points that fall inside the range with probability $\geq \tau$

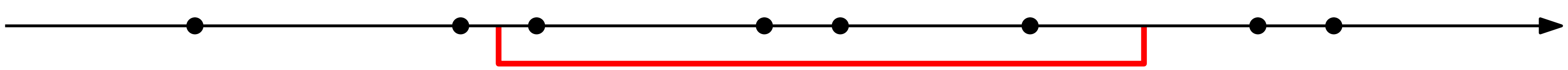
```
SELECT * FROM sensorreadings
WHERE Prob[20 <= temp <= 25] >= 0.5
```

The Problem: Range Searching

- One of the very first and fundamental problems studied in querying uncertain data

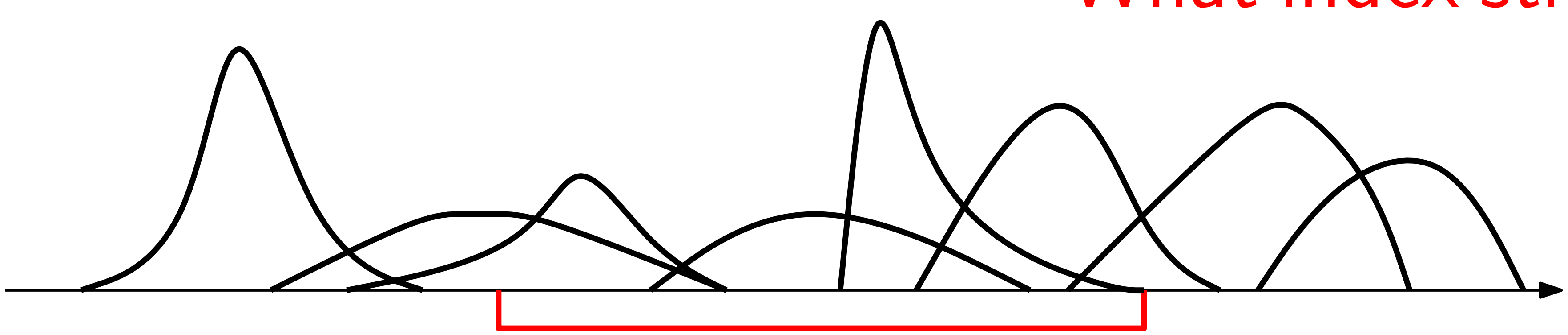
The certain case:

Binary tree (B-tree)



The uncertain case:

What index structure?



Report all points that fall inside the range with probability $\geq \tau$

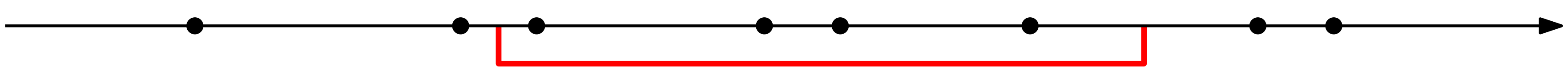
```
SELECT * FROM sensorreadings
WHERE Prob[20 <= temp <= 25] >= 0.5
```

The Problem: Range Searching

- One of the very first and fundamental problems studied in querying uncertain data

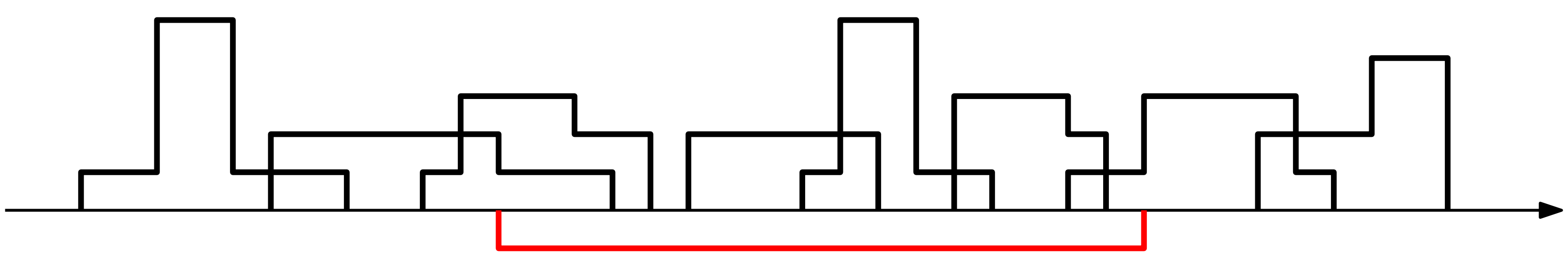
The certain case:

Binary tree (B-tree)



The uncertain case:

What index structure?



Report all points that fall inside the range with probability $\geq \tau$

Assume each pdf is piecewise constant (histogram)

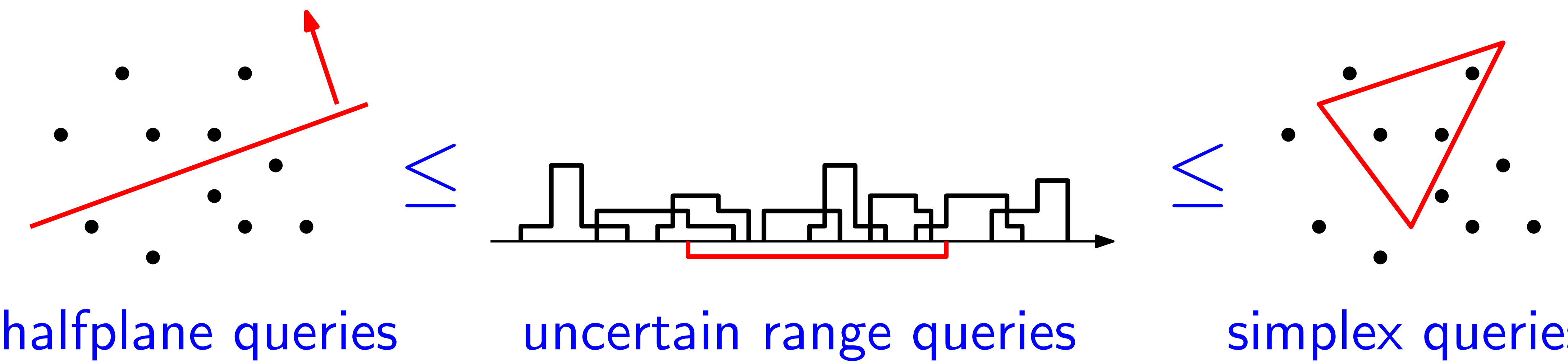
Will also talk about Gaussian and piecewise algebraic

Queries with a Fixed τ

- Previous results: [Cheng, Xia, Prabhakar, Shah, Vitter '04]
  - When each pdf is uniform (i.e., histogram with one piece)
  - An index with size $O(n\tau^{-1})$ and query time $O(\tau^{-1} \log n + k)$
    k: output size

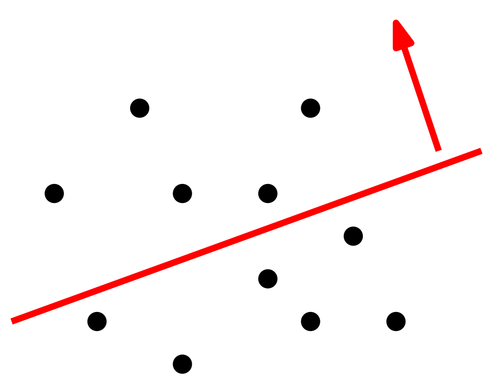
Queries with a Fixed τ

- Previous results: [Cheng, Xia, Prabhakar, Shah, Vitter '04]
  - When each pdf is uniform (i.e., histogram with one piece)
  - An index with size $O(n\tau^{-1})$ and query time $O(\tau^{-1}\log n + k)$
    k: output size
  - On the complexity side:

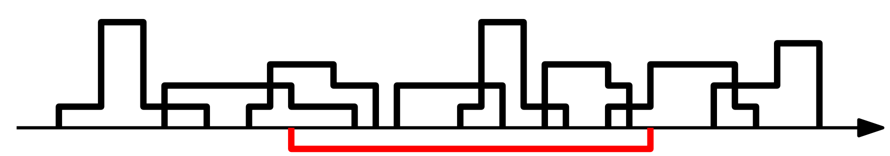


Queries with a Fixed τ

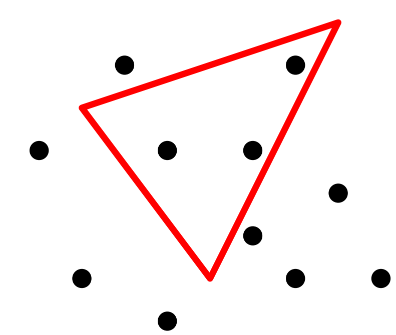
- Previous results: [Cheng, Xia, Prabhakar, Shah, Vitter '04]
  - When each pdf is uniform (i.e., histogram with one piece)
  - An index with size $O(n\tau^{-1})$ and query time $O(\tau^{-1} \log n + k)$
    k: output size
  - On the complexity side:



$\leq$



$\leq$



halfplane queries	uncertain range queries	simplex queries
$\Theta(\log n + k)$	?	$\Theta(\sqrt{n} + k)$
with $O(n)$ size	with $O(n)$ size	with $O(n)$ size

Queries with a Fixed τ

- Previous results: [Cheng, Xia, Prabhakar, Shah, Vitter '04]
  - When each pdf is uniform (i.e., histogram with one piece)
  - An index with size $O(n\tau^{-1})$ and query time $O(\tau^{-1}\log n + k)$
    k: output size
  - On the complexity side:

halfplane queries	uncertain range queries	simplex queries
$\Theta(\log n + k)$	?	$\Theta(\sqrt{n} + k)$
with $O(n)$ size	with $O(n)$ size	with $O(n)$ size

Queries with a Variable τ (given at query time)

- Previous results: [Cheng, Xia, Prabhakar, Shah, Vitter '04]
  - Only heuristics are given, with worst-case query time $\Theta(n)$
- Other follow-up works are also heuristic
  - [Tao, Cheng, Xiao, Ngai, Kao, Prabhakar '05]
  - [Ljosa, Singh '07]

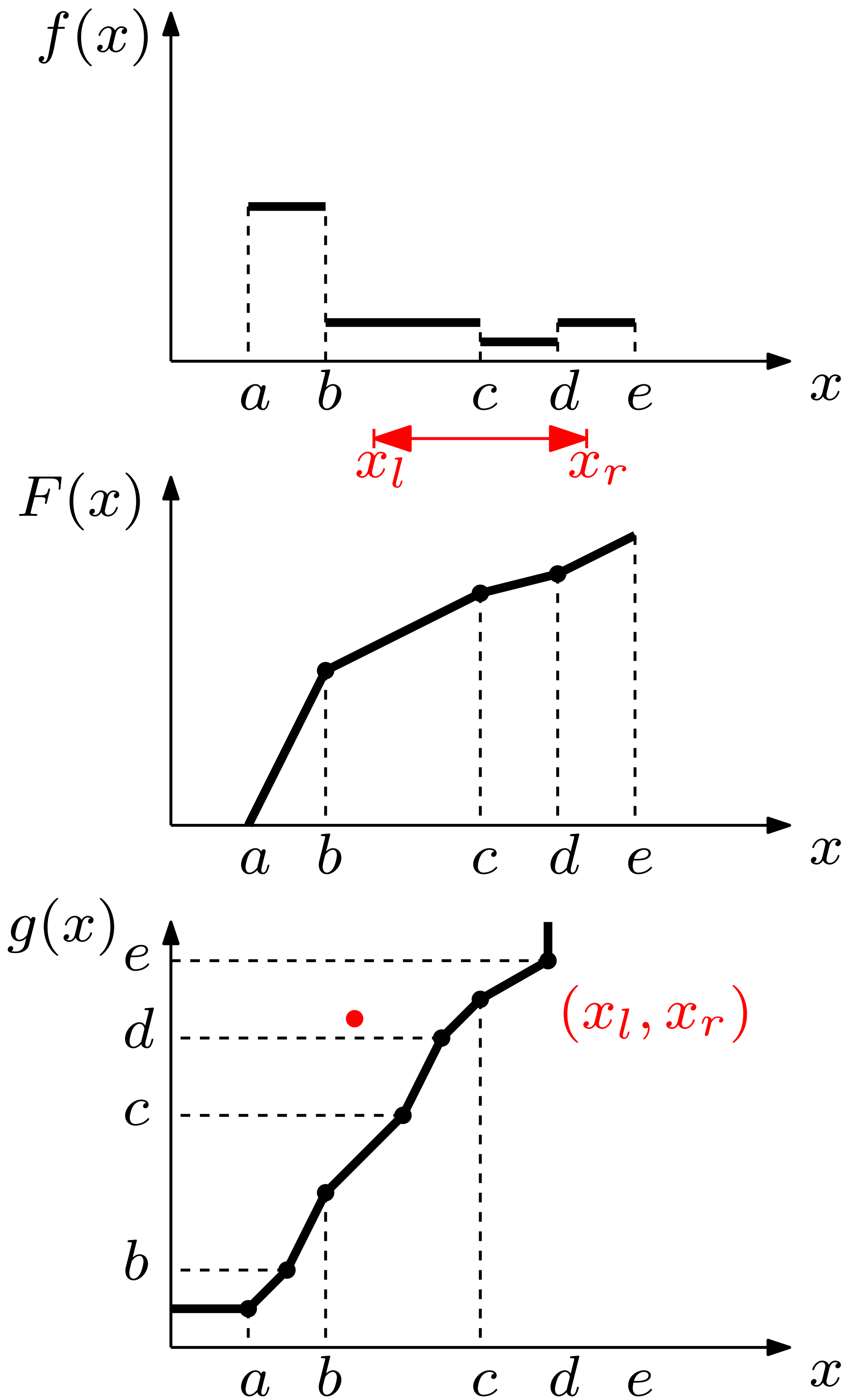
Queries with a Variable τ (given at query time)

- Previous results: [Cheng, Xia, Prabhakar, Shah, Vitter '04]
  - Only heuristics are given, with worst-case query time $\Theta(n)$
- Other follow-up works are also heuristic
  - [Tao, Cheng, Xiao, Ngai, Kao, Prabhakar '05]
  - [Ljosa, Singh '07]
- Our results
  - An index structure with size $O(n \log^2 n)$ and query $O(\log^3 n + k)$

Queries with a Variable τ (given at query time)

- Previous results: [Cheng, Xia, Prabhakar, Shah, Vitter '04]
  - Only heuristics are given, with worst-case query time $\Theta(n)$
- Other follow-up works are also heuristic
  - [Tao, Cheng, Xiao, Ngai, Kao, Prabhakar '05]
  - [Ljosa, Singh '07]
- Our results
  - An index structure with size $O(n \log^2 n)$ and query $O(\log^3 n + k)$
  - Can be made dynamic
  - Can be made I/O-efficient

Queries with a Fixed τ: A Geometric Reduction



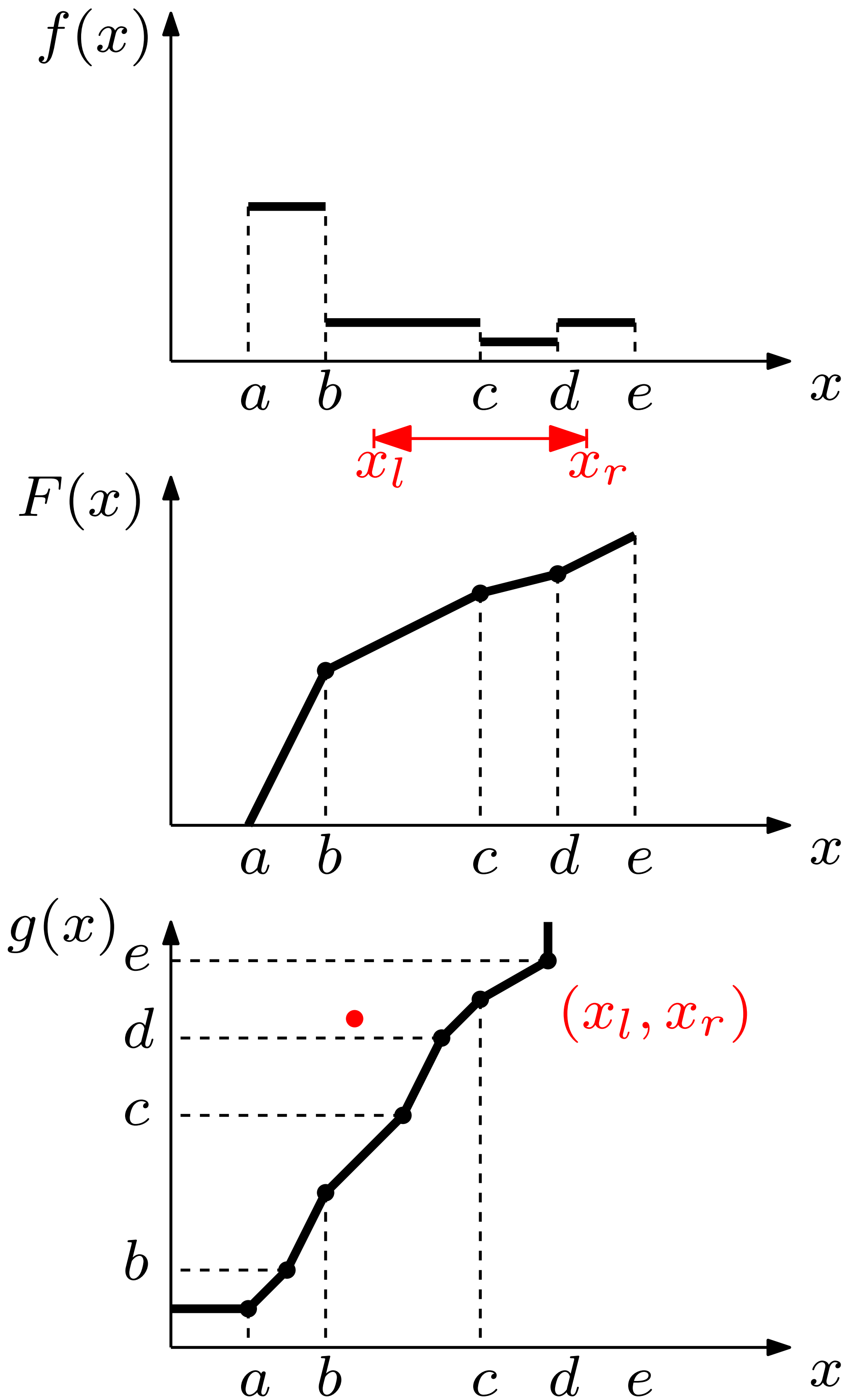
pdf

cdf

the threshold function

For any a, $g(a) :=$ the minimum b s.t. $F(b) - F(a) \geq \tau$;
If no such b exists, $g(a) := \infty$.

Queries with a Fixed τ: A Geometric Reduction



pdf

Properties of $g(x)$:

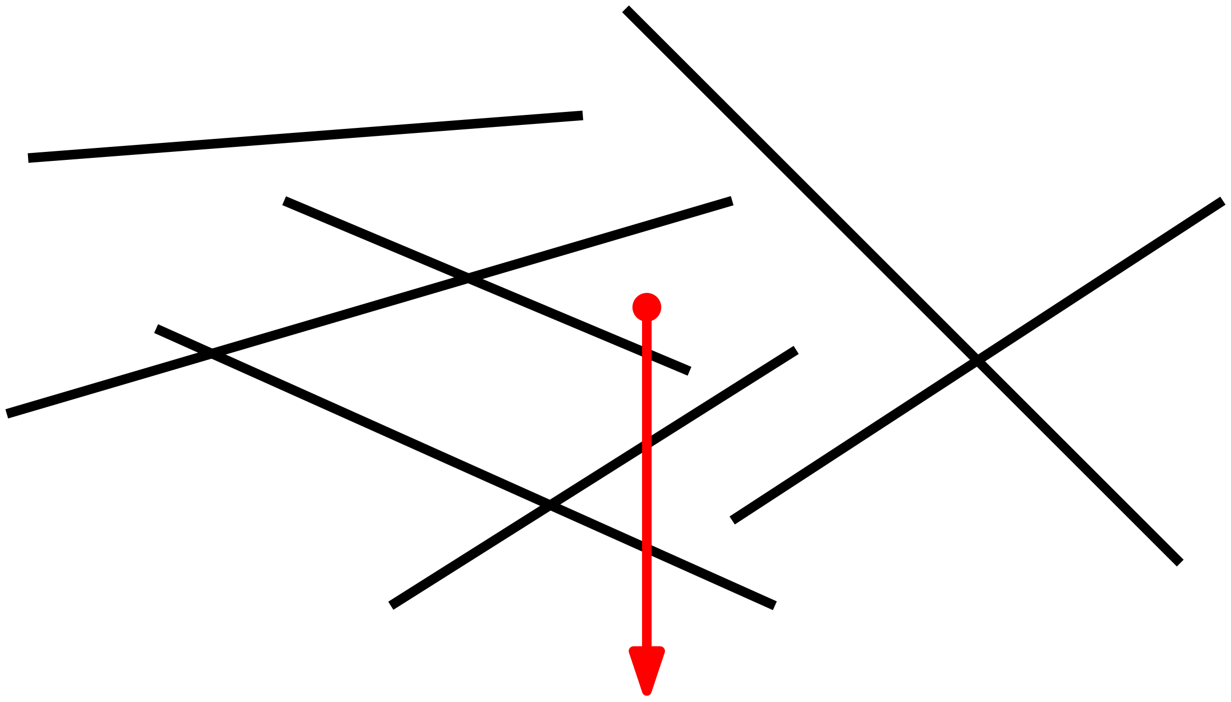
- $g(x)$ has complexity linear in the size of pdf
- $\Pr[p \in (x_l, x_r)] \geq \tau \Leftrightarrow x_r \geq g(x_l)$

cdf

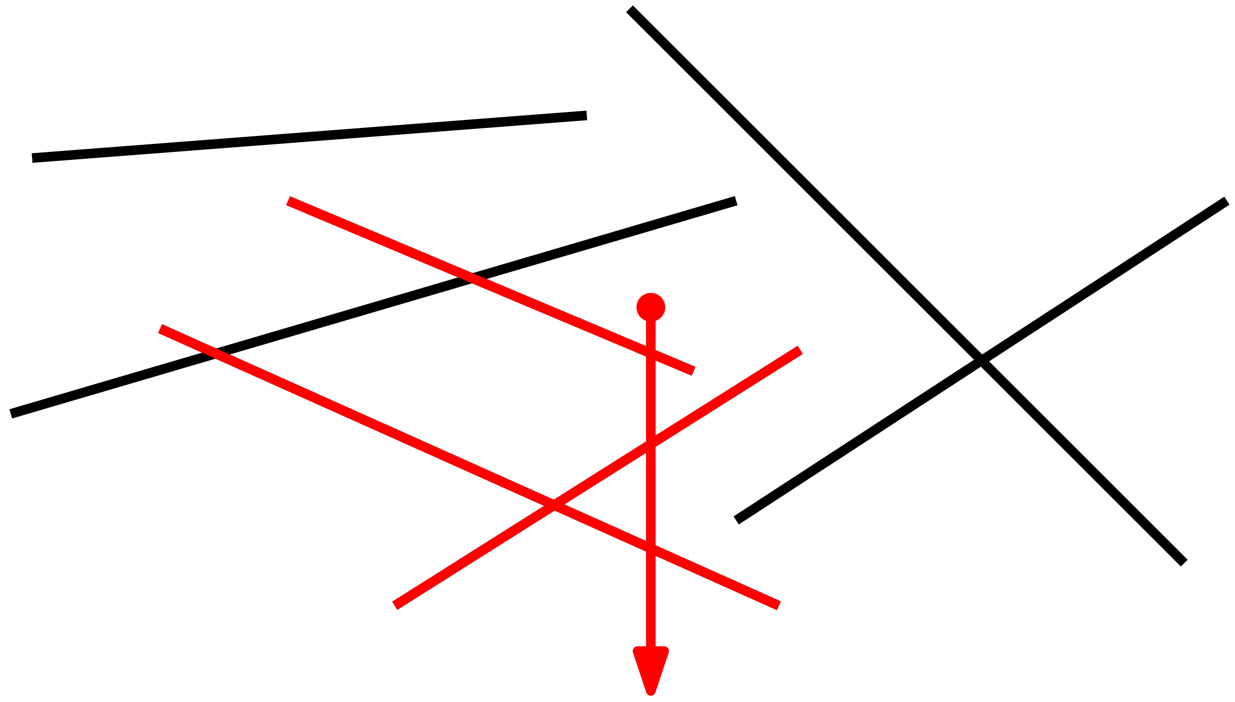
the threshold function

For any a, $g(a) :=$ the minimum b s.t. $F(b) - F(a) \geq \tau$;
If no such b exists, $g(a) := \infty$.

Queries with a Fixed τ: The Problem

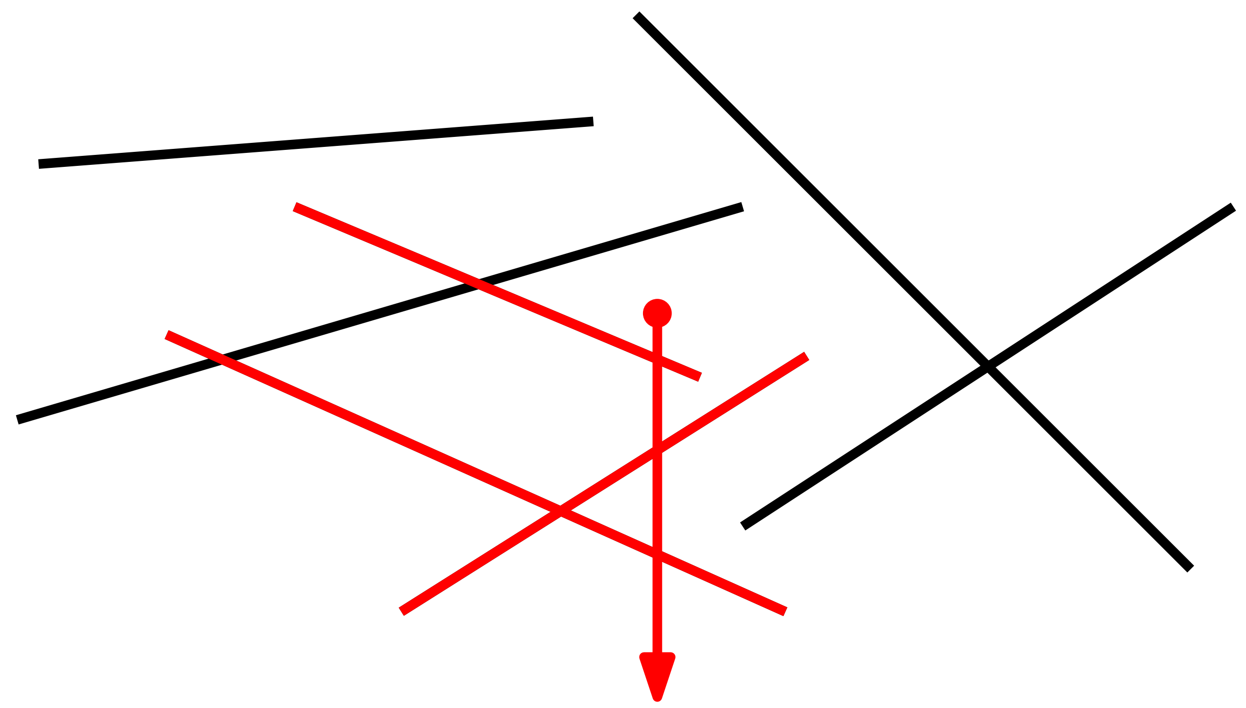


Queries with a Fixed τ: The Problem



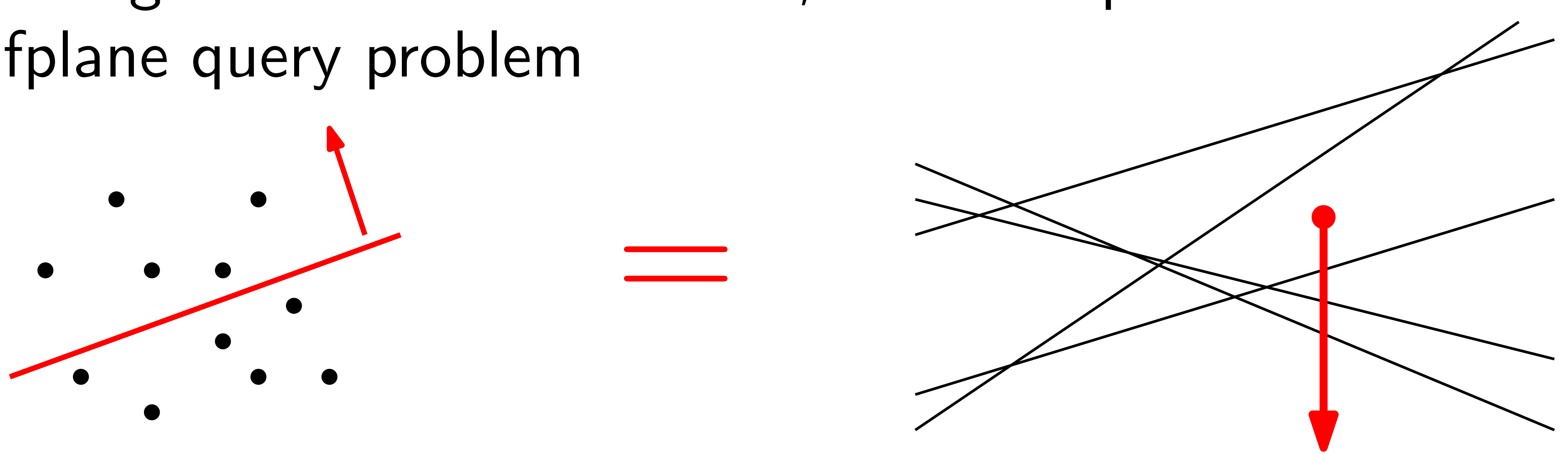
The segment-below-point problem

Queries with a Fixed τ: The Problem

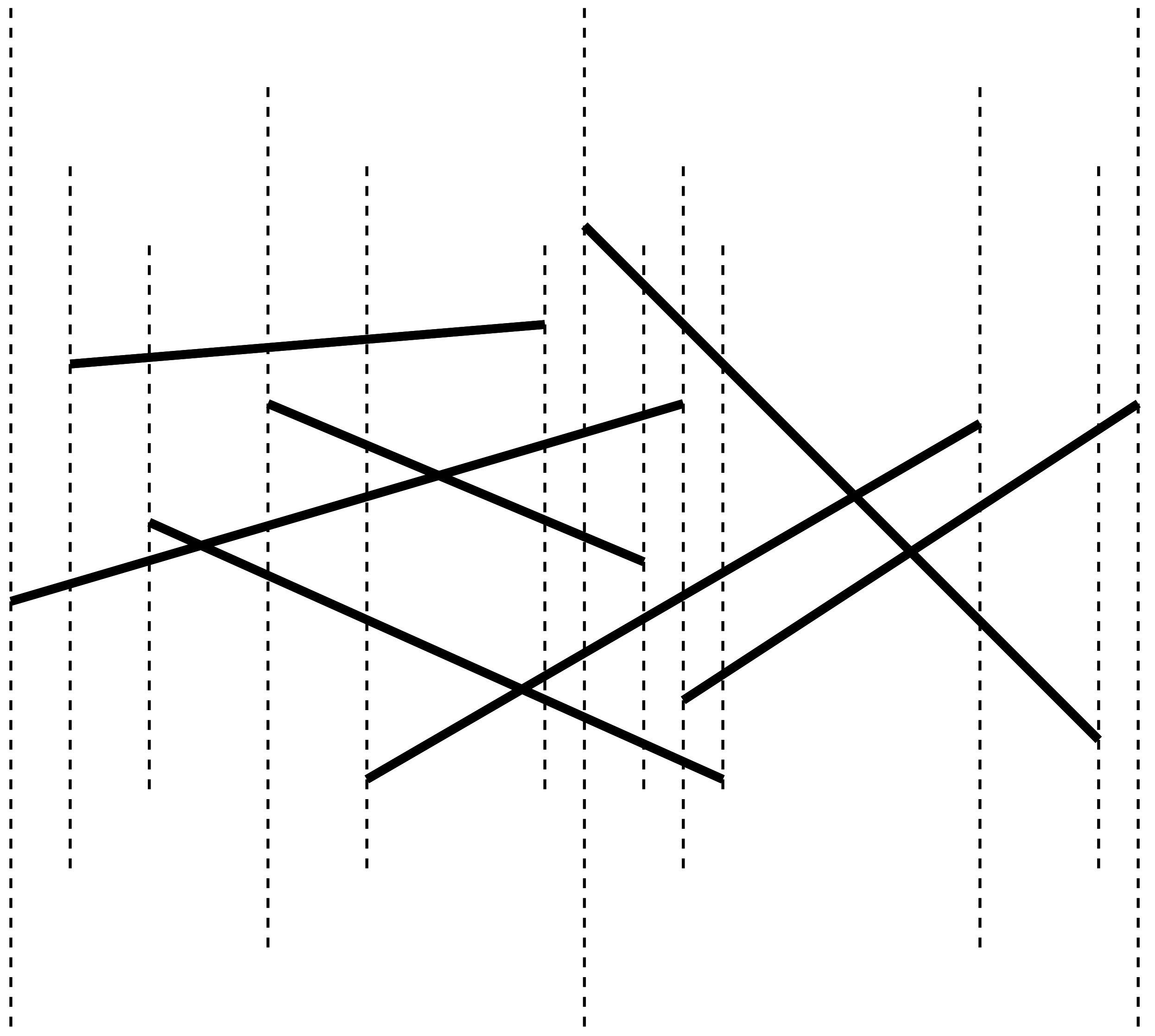


The segment-below-point problem

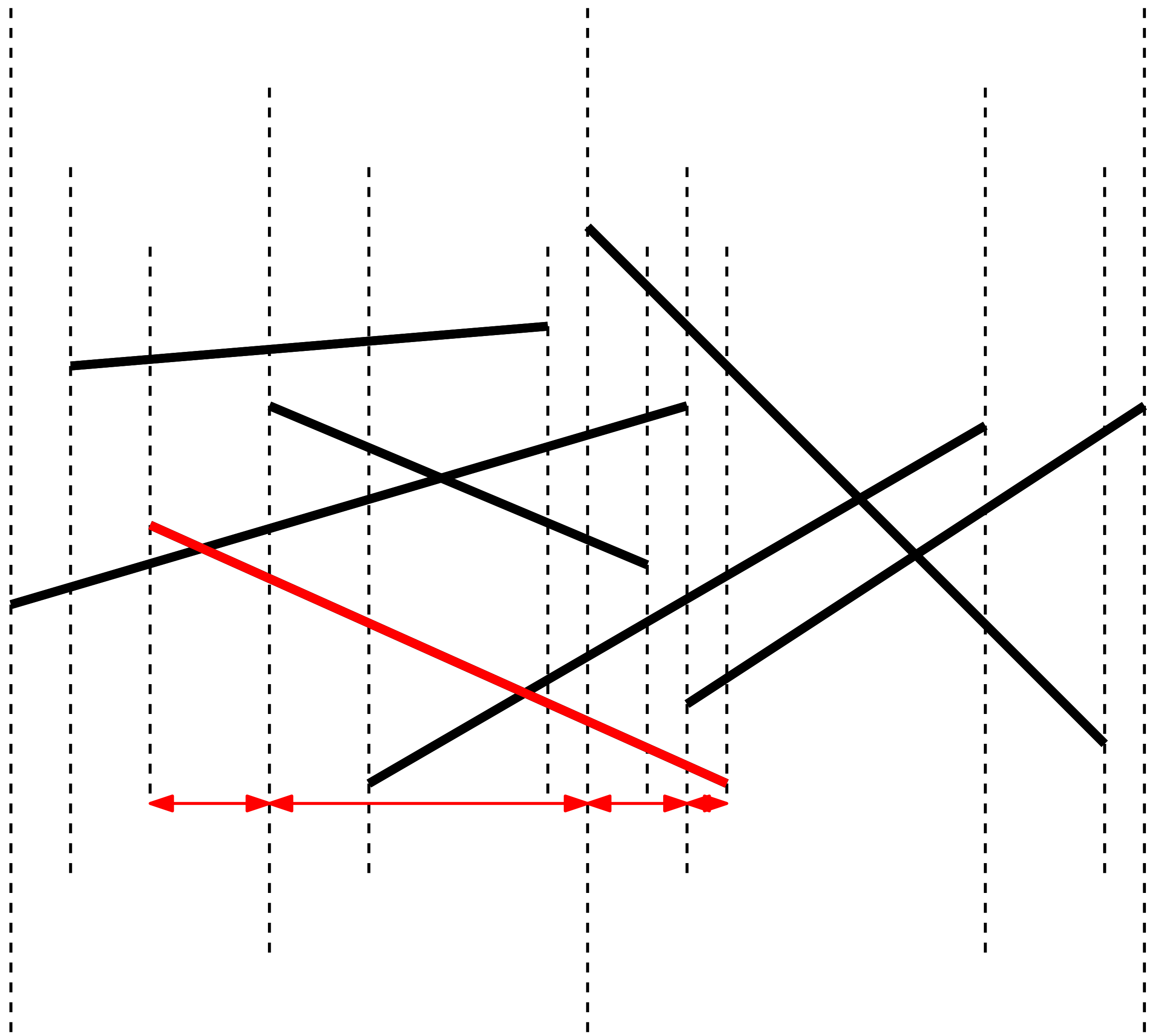
If all segments are infinite lines, then the problem becomes the halfplane query problem



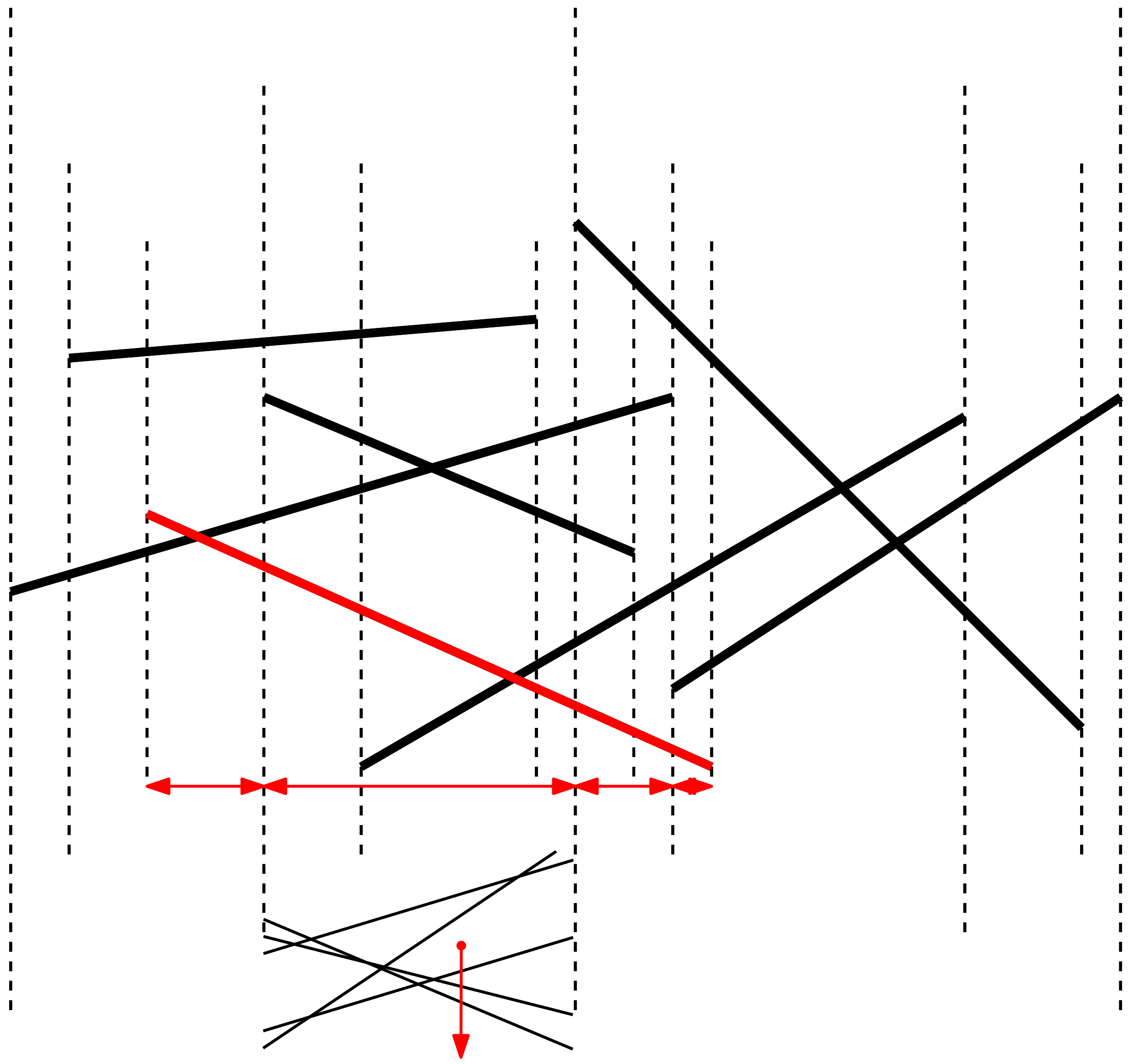
Queries with a Fixed τ: A First Attempt



Queries with a Fixed τ: A First Attempt

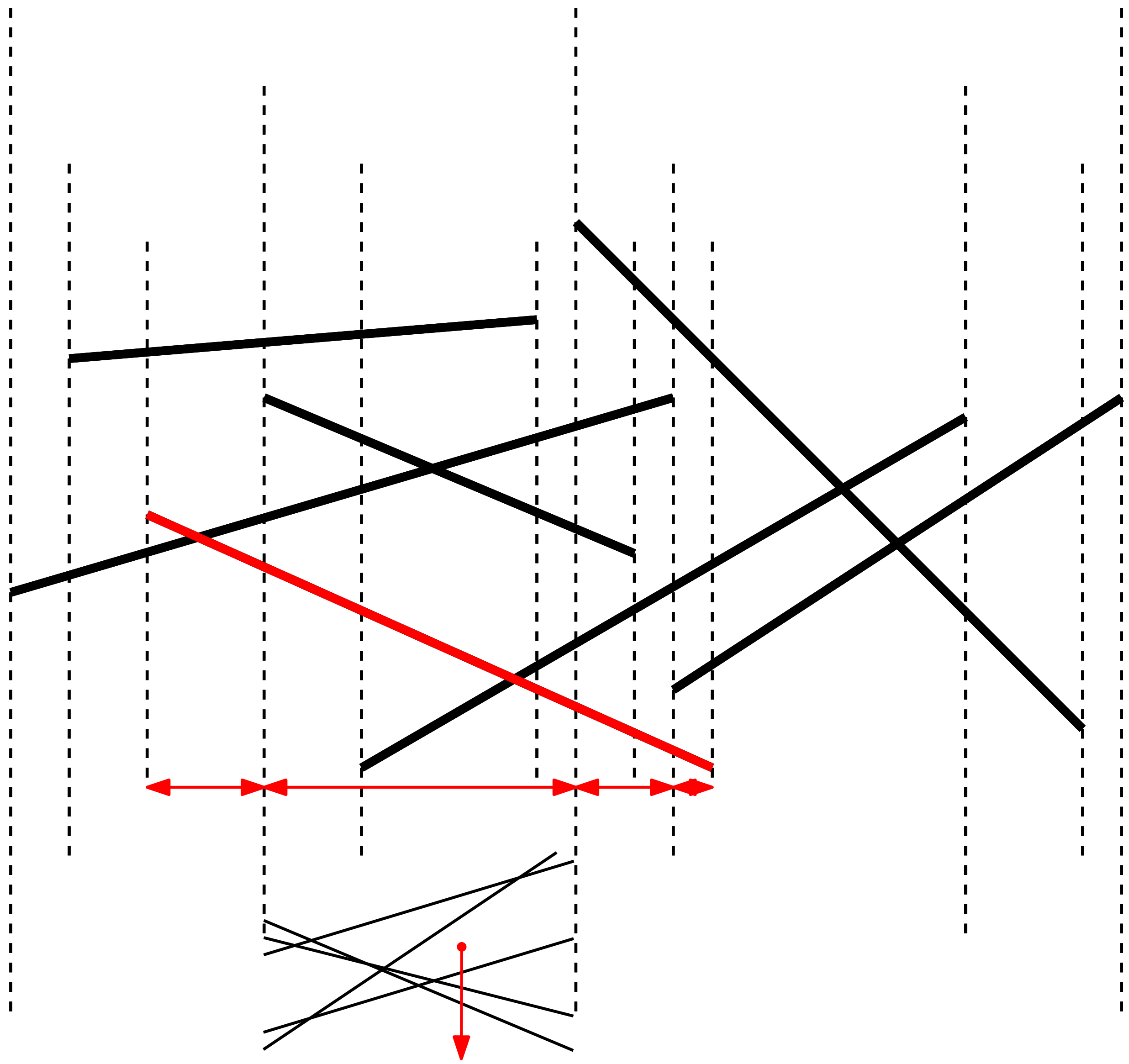


Queries with a Fixed τ: A First Attempt



Segment tree approach:
$O(n)$ canonical slabs;
Each segment decomposed into $O(\log n)$ slabs;
Build a halfplance structure for each slab

Queries with a Fixed τ: A First Attempt



Obtain a structure:
size $O(n \log n)$,
query $O(\log^2 n + k)$.

Query can be improved to $O(\log n + k)$ using fractional cascading

But hard to reduce size to linear

Segment tree approach:
$O(n)$ canonical slabs;
Each segment decomposed into $O(\log n)$ slabs;
Build a halfplance structure for each slab

Queries with a Fixed τ: Achieving Optimal

- Clever ideas and complicated techniques
  - Segment tree
  - Interval tree
  - Sampling
  - Compression
  - Fractional cascading
  - Boostrapping

Queries with a Fixed τ: Achieving Optimal

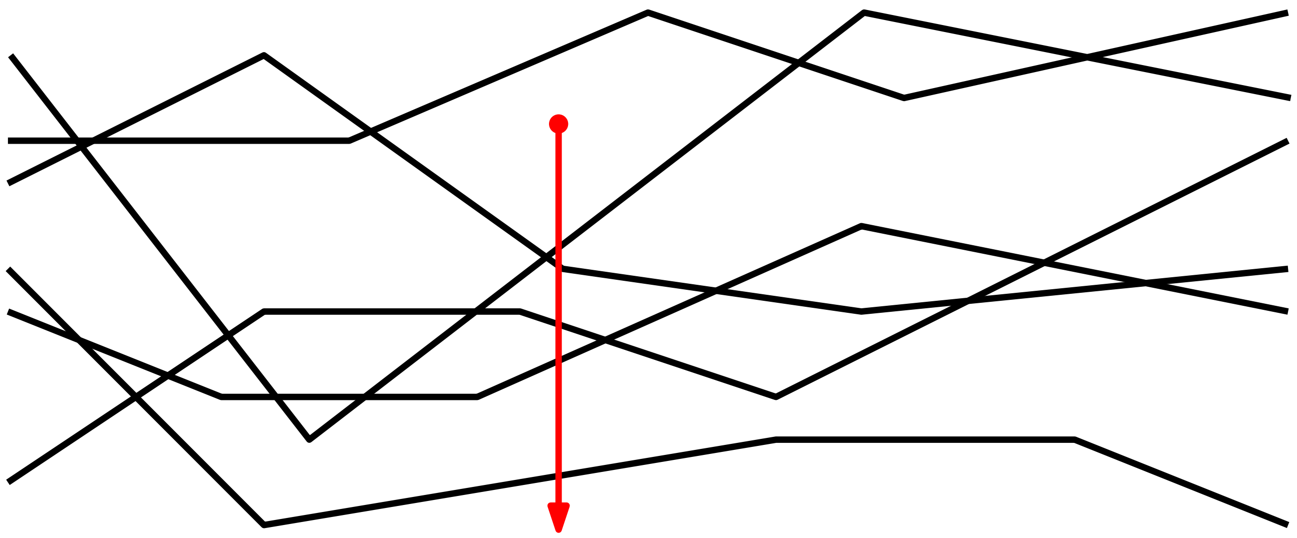
- Clever ideas and complicated techniques
  - Segment tree
  - Interval tree
  - Sampling
  - Compression
  - Fractional cascading
  - Boostrapping
- Nice theoretical result, but too complicated to implement

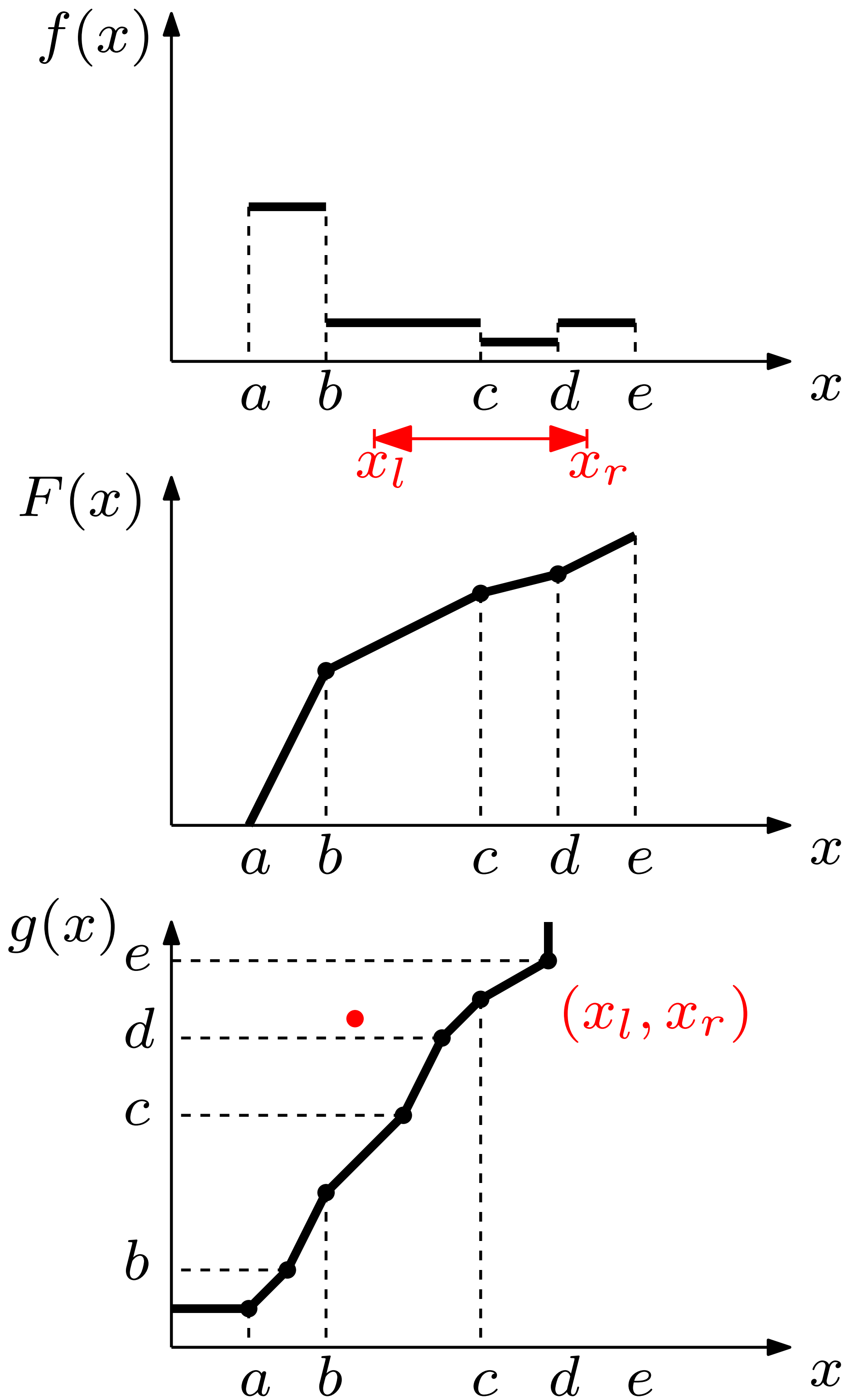
A Simpler and More General Structure

(although not optimal)

A Simpler and More General Structure

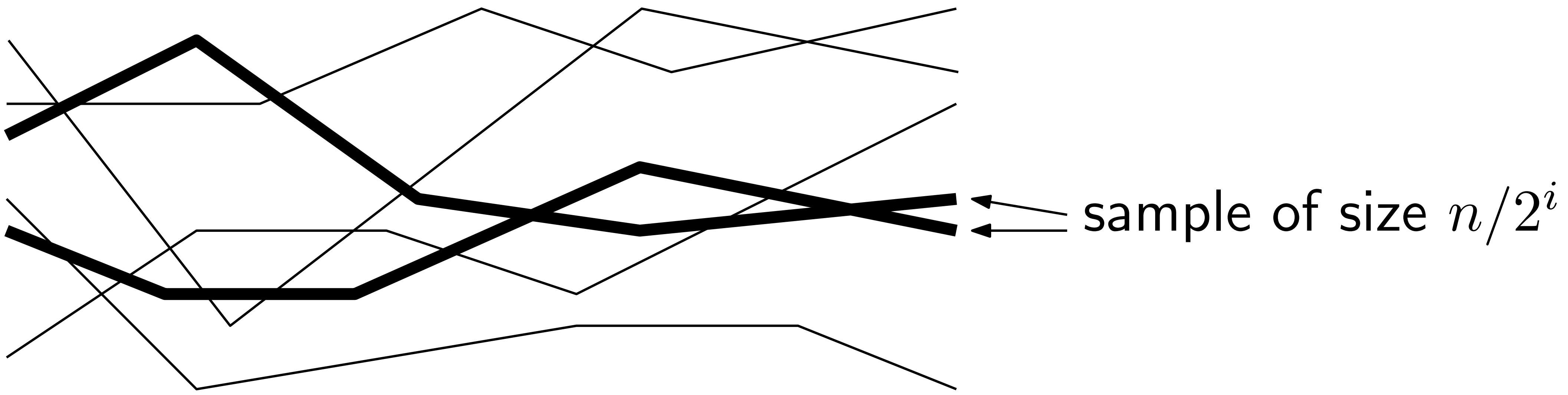
(although not optimal)



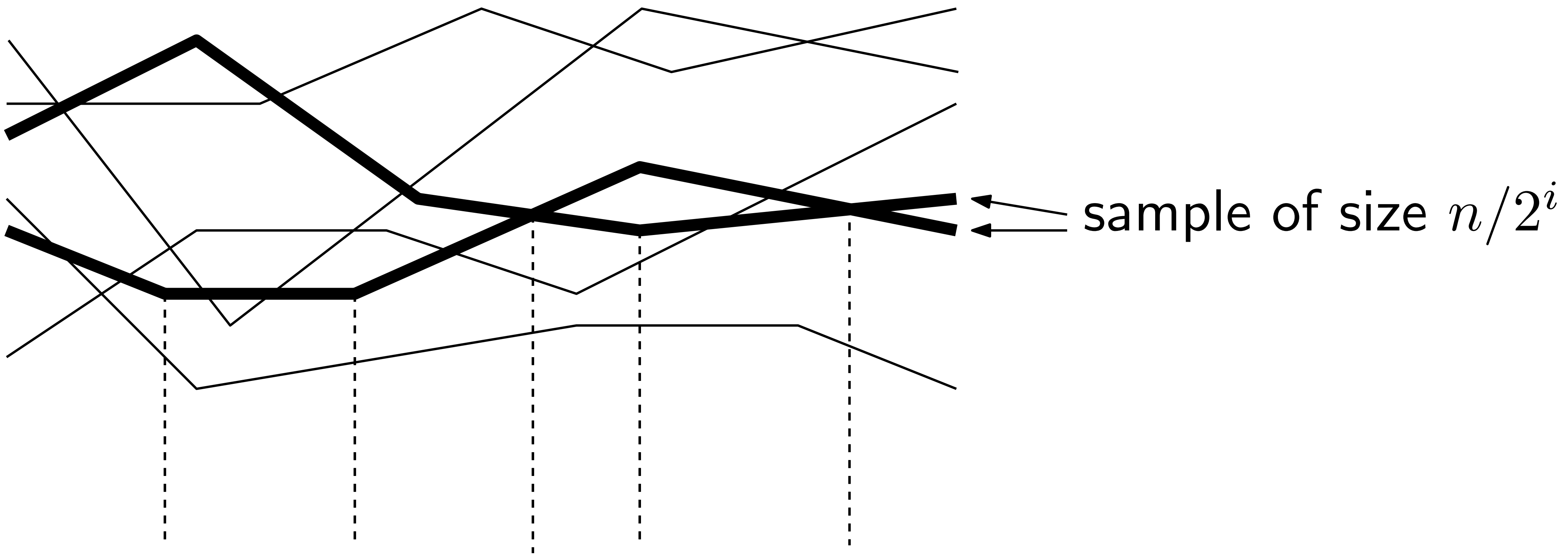


Report all polygonal chains below a query point

A Simpler and More General Structure

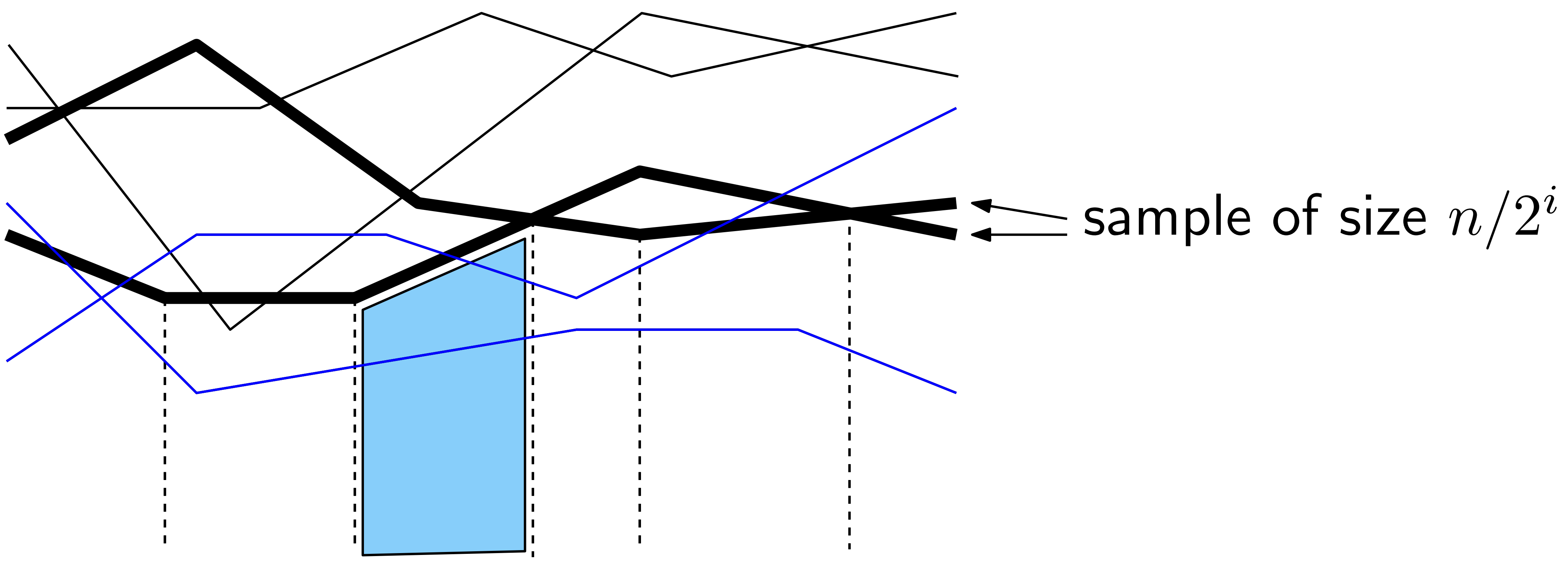


A Simpler and More General Structure



Compute the trapezoidal decomposition of the area bounded by the lower envelope of the sampled chains

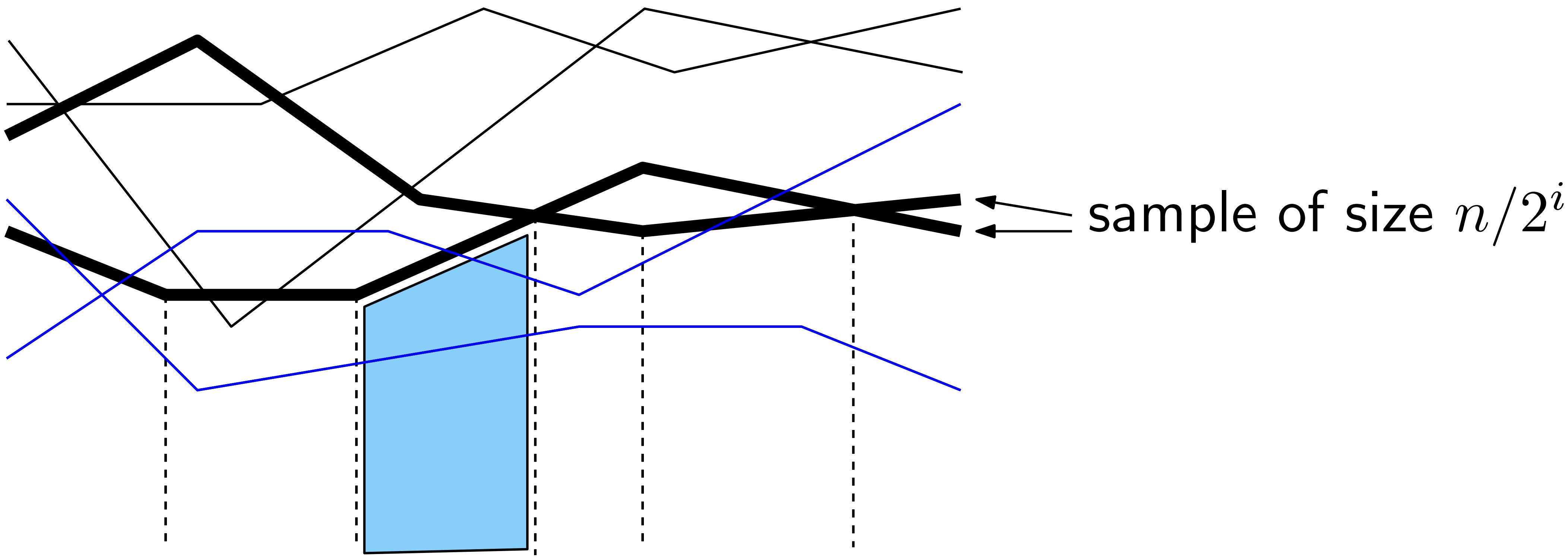
A Simpler and More General Structure



Compute the trapezoidal decomposition of the area bounded by the lower envelope of the sampled chains

Each trapezoid stores all the *conflicting* chains; can show that the expected size of each conflict list is $O(2^i)$

A Simpler and More General Structure

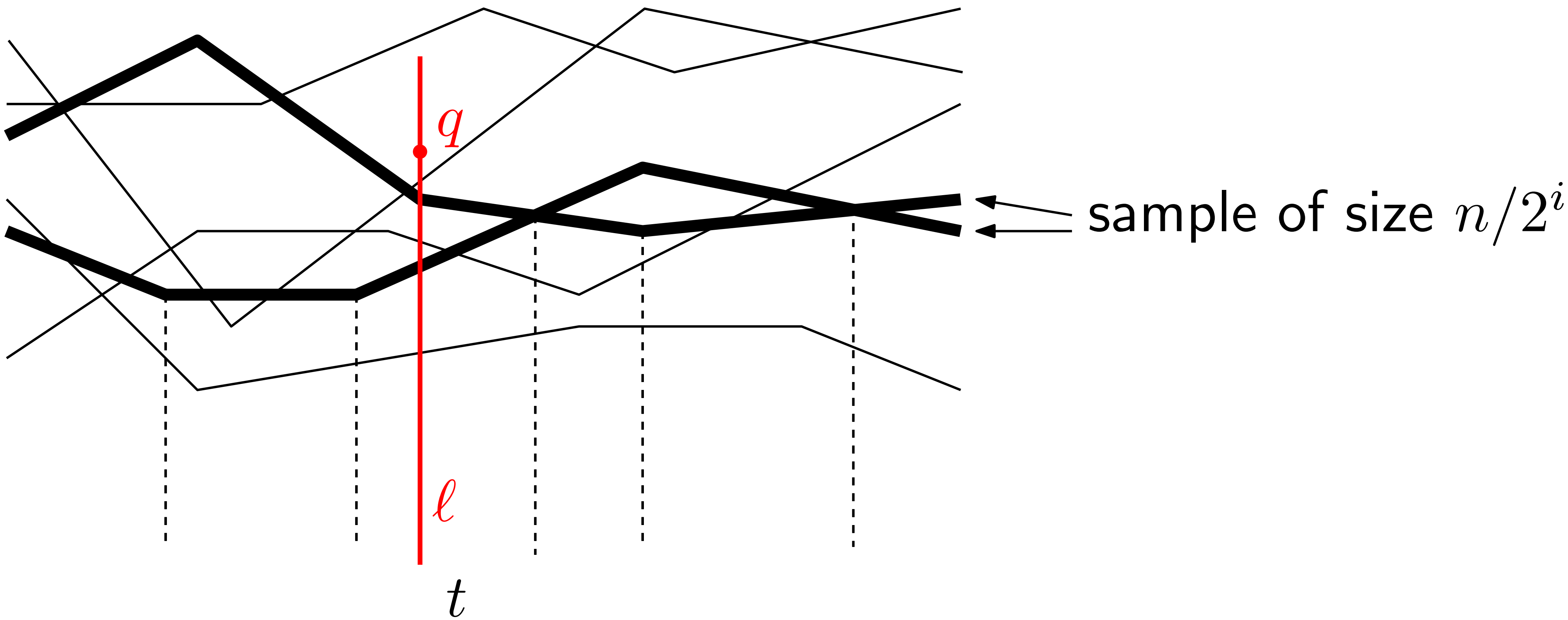


Compute the trapezoidal decomposition of the area bounded by the lower envelope of the sampled chains

Each trapezoid stores all the *conflicting* chains; can show that the expected size of each conflict list is $O(2^i)$

Do the above for $i = 1, 2, \ldots, \log n$

Querying the Structure



Try successive structures for $i = 1, 2, \ldots$:

locate the trapezoid t that ℓ intersects;

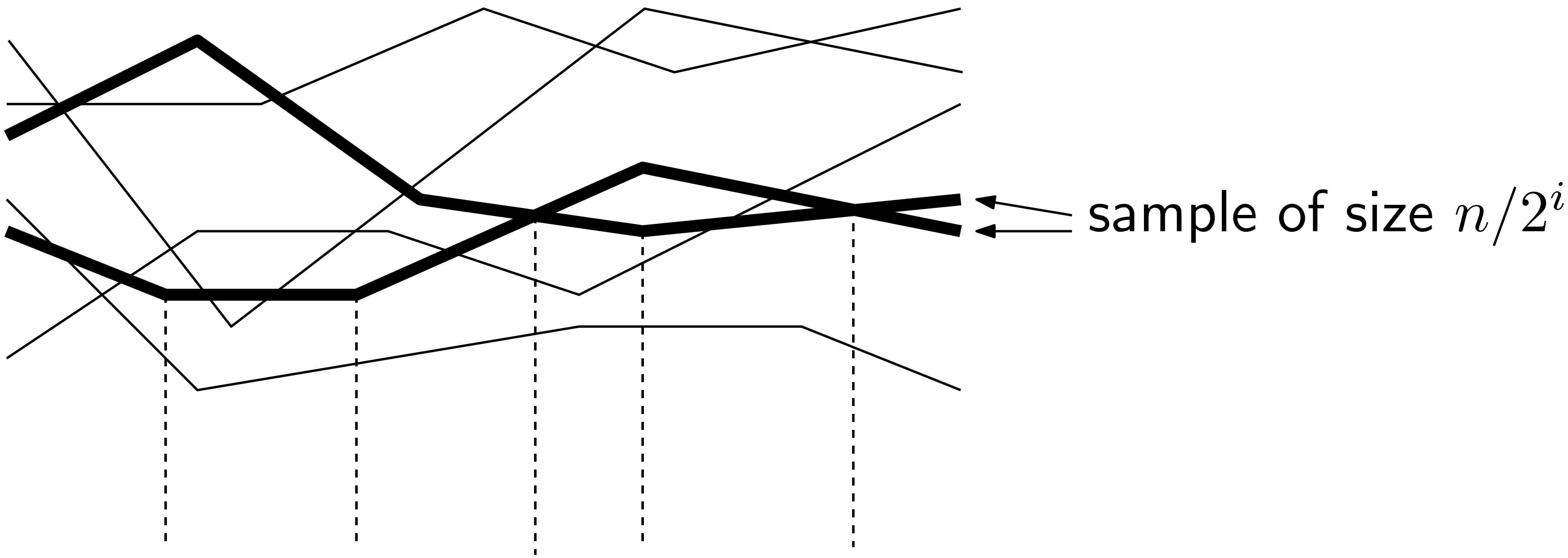
If $q \in t$

scan t's conflict list and report all chains below q;

stop;

Expected time: $O(\log n + k)$

Size of the Structure



size of each conflict list $= O(2^i)$
$\#$ trapezoids $=$ complexity of the lower envelope $= O(\frac{n}{2^i}\alpha(\frac{n}{2^i}))$
size of structure for each i: $O(n\alpha(n))$
total size: $O(n\alpha(n)\log n)$

$\alpha(n)$: inverse Ackermann function — extremely slow-growing

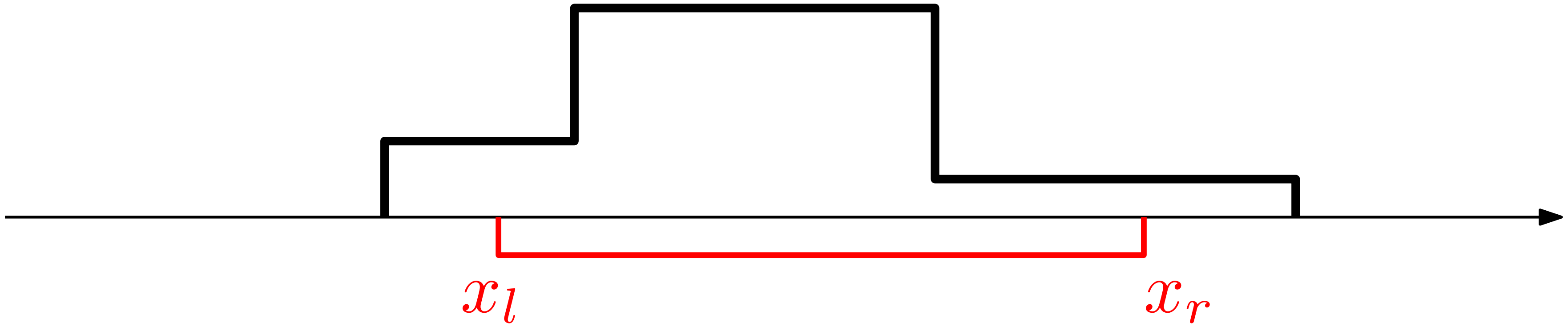
Supporting Other pdf's

- Suppose each pdf is piecewise algebraic
  - The threshold function $g(n)$ is also piecewise algebraic
- The structure and analysis remain the same, only the complexity of the lower envelope could change
- One can write out the piecewise form of $g(x)$, and determine the maximum number of intersections between any two different pieces, say c
- Query remains optimal $O(\log n + k)$
- Size becomes $O(\lambda_{c+2}(n) \log n)$ [Davenport, Schinzel '65]

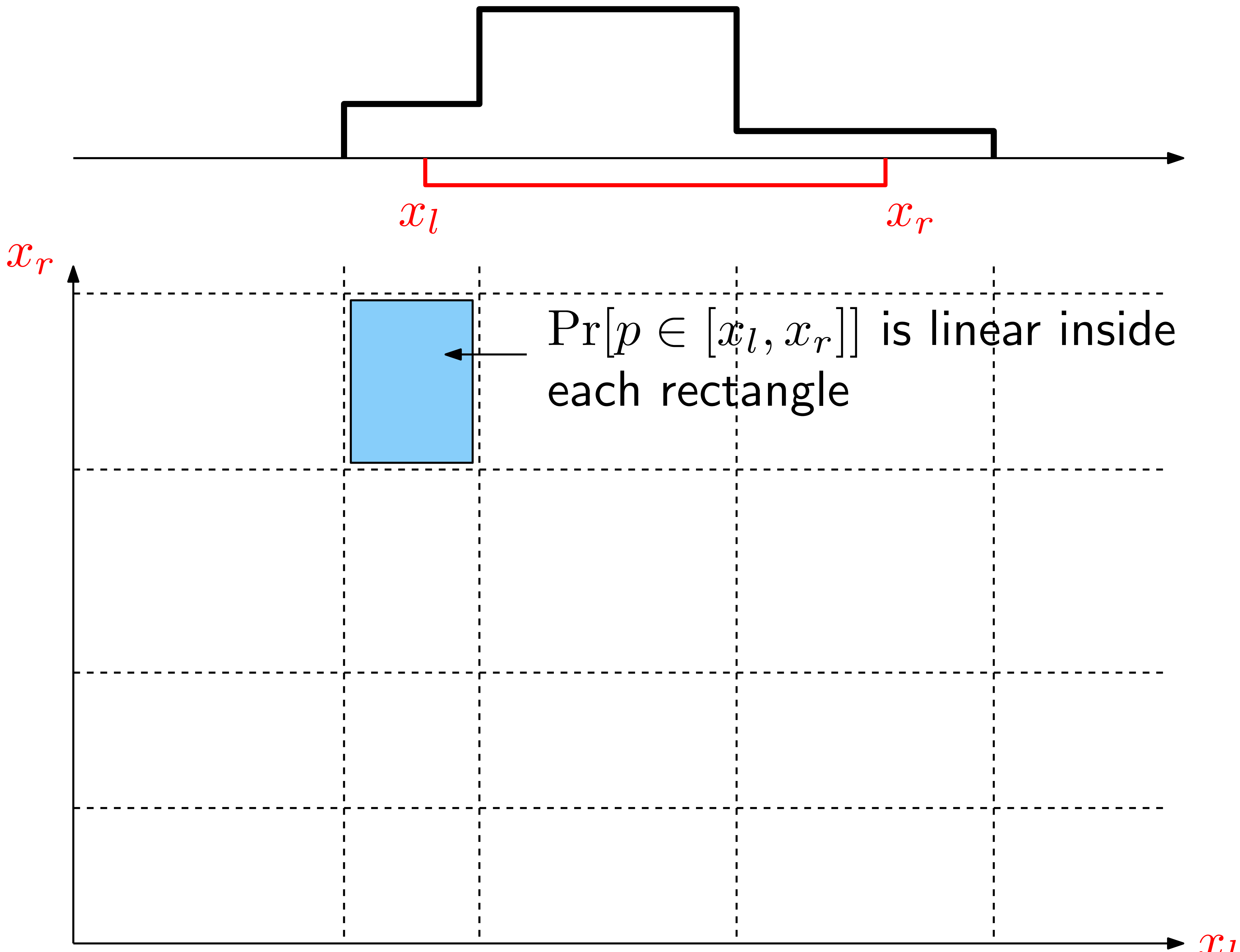
Supporting Other pdf's

- Suppose each pdf is piecewise algebraic
  - The threshold function $g(n)$ is also piecewise algebraic
- The structure and analysis remain the same, only the complexity of the lower envelope could change
- One can write out the piecewise form of $g(x)$, and determine the maximum number of intersections between any two different pieces, say c
- Query remains optimal $O(\log n + k)$
- Size becomes $O(\lambda_{c+2}(n) \log n)$ [Davenport, Schinzel '65]
  - $\lambda_c(n)$: the maximum length of (n, c) *Davenport-Schinzel sequences*
  - $\lambda_2(n) = \Theta(n), \lambda_3(n) = \Theta(n\alpha(n))$, $\lambda_4(n) = \Theta(n2^{\alpha(n)})$, $\lambda_{2t+2}(n) = n2^{(1/t!)\alpha^t(n)+\Theta(\alpha^{t-1}(n))}$ [Agarwal, Sharir '00]

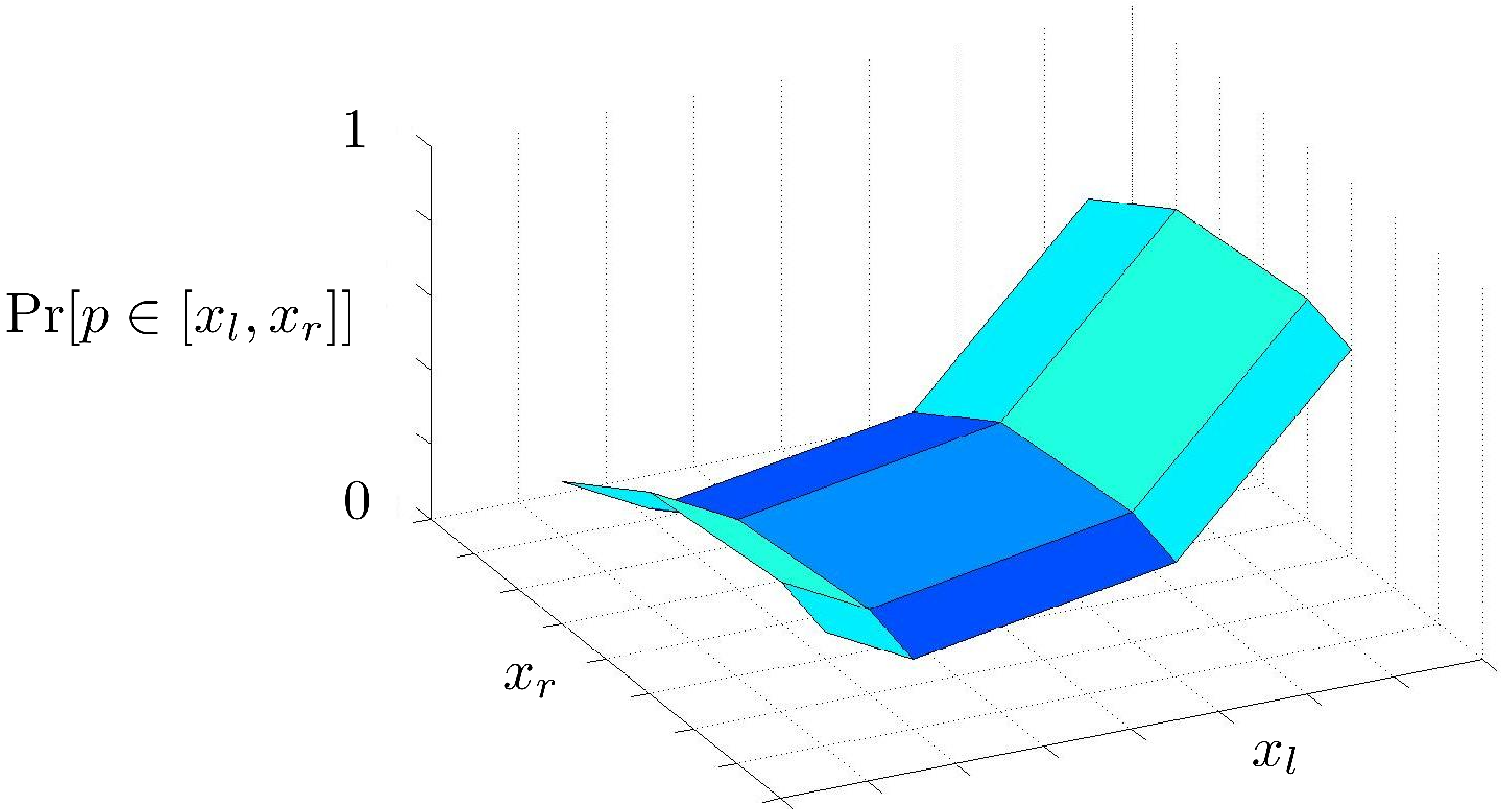
Queries with Variable τ



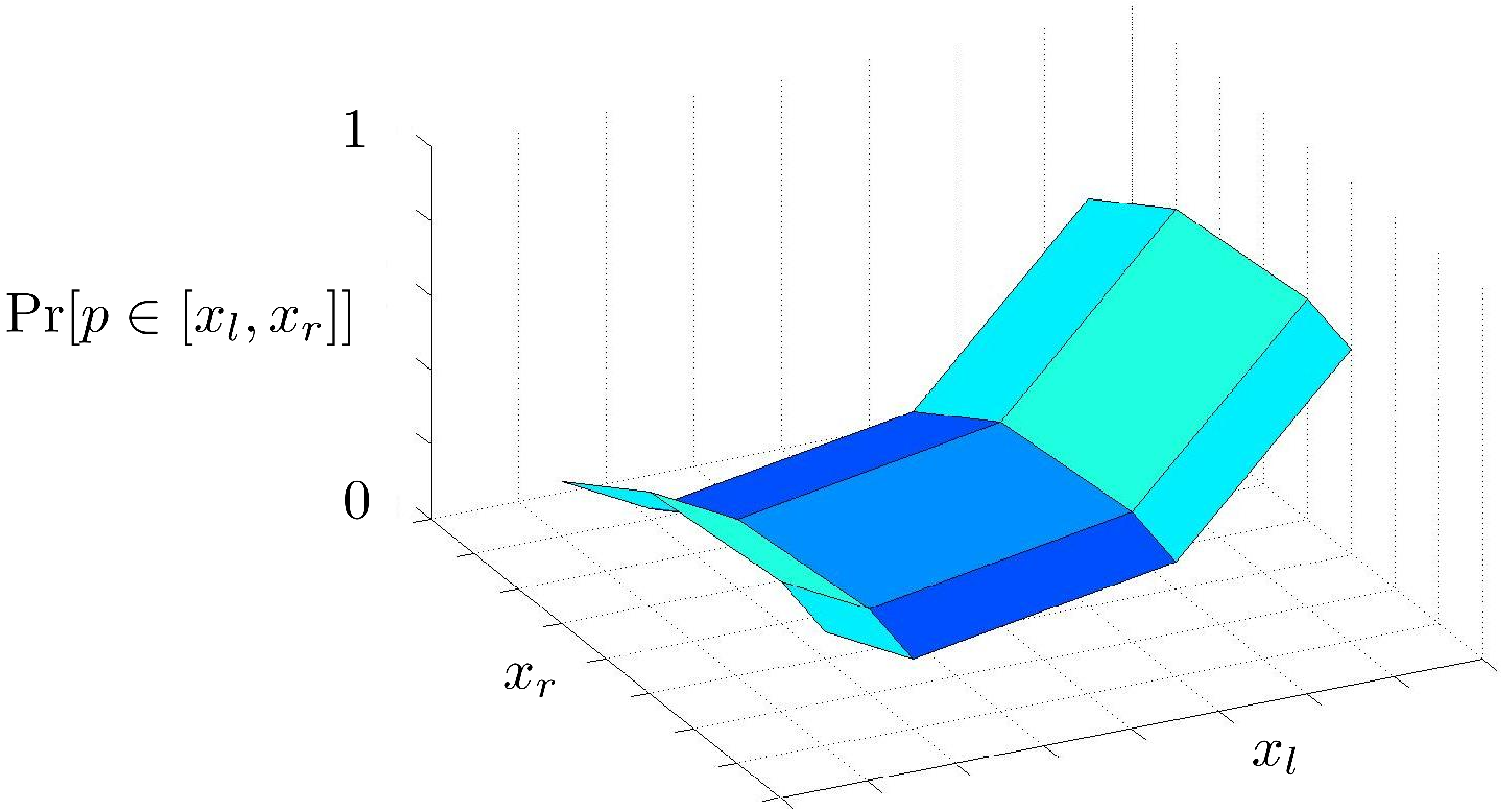
Queries with Variable τ



Queries with Variable τ

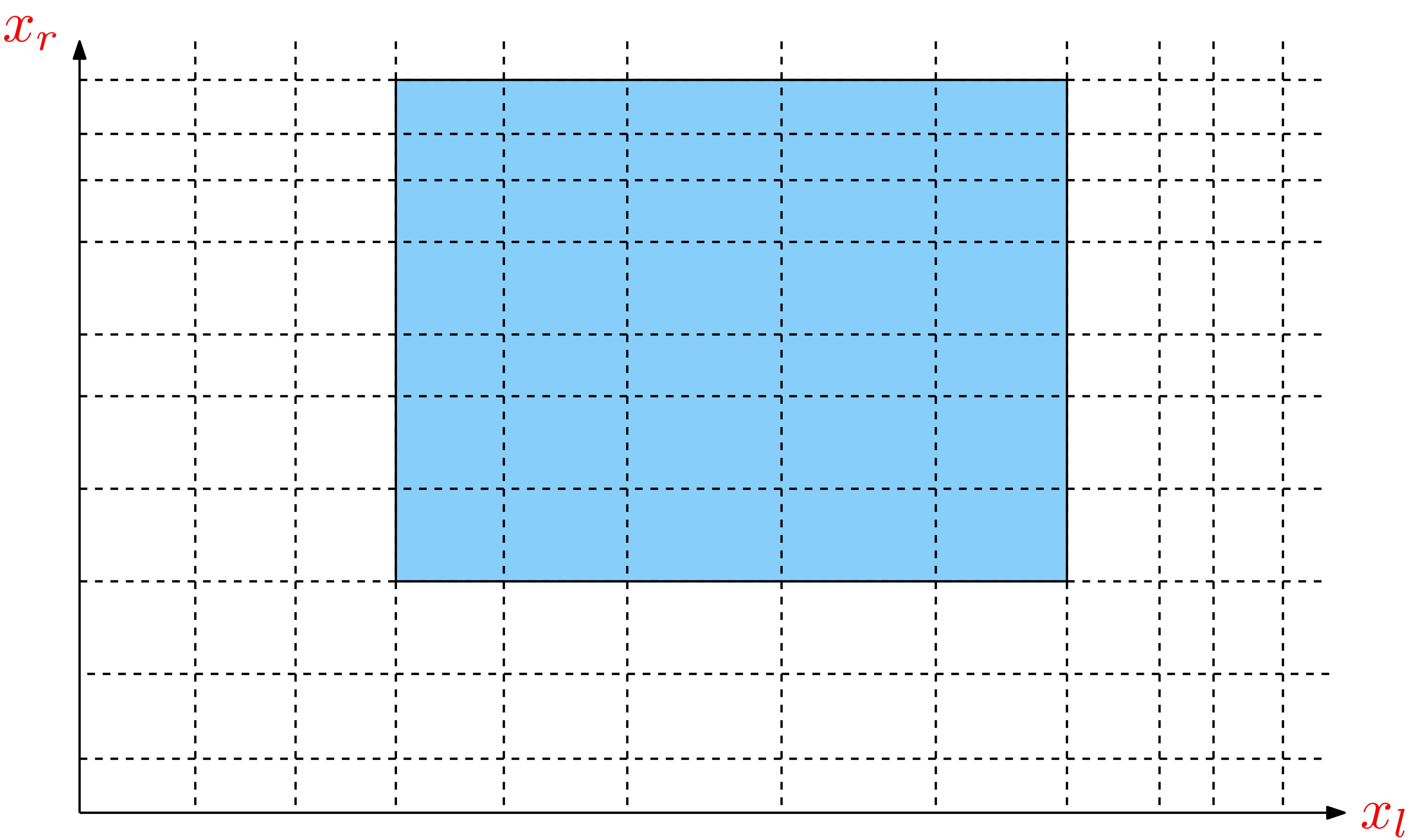


Queries with Variable τ

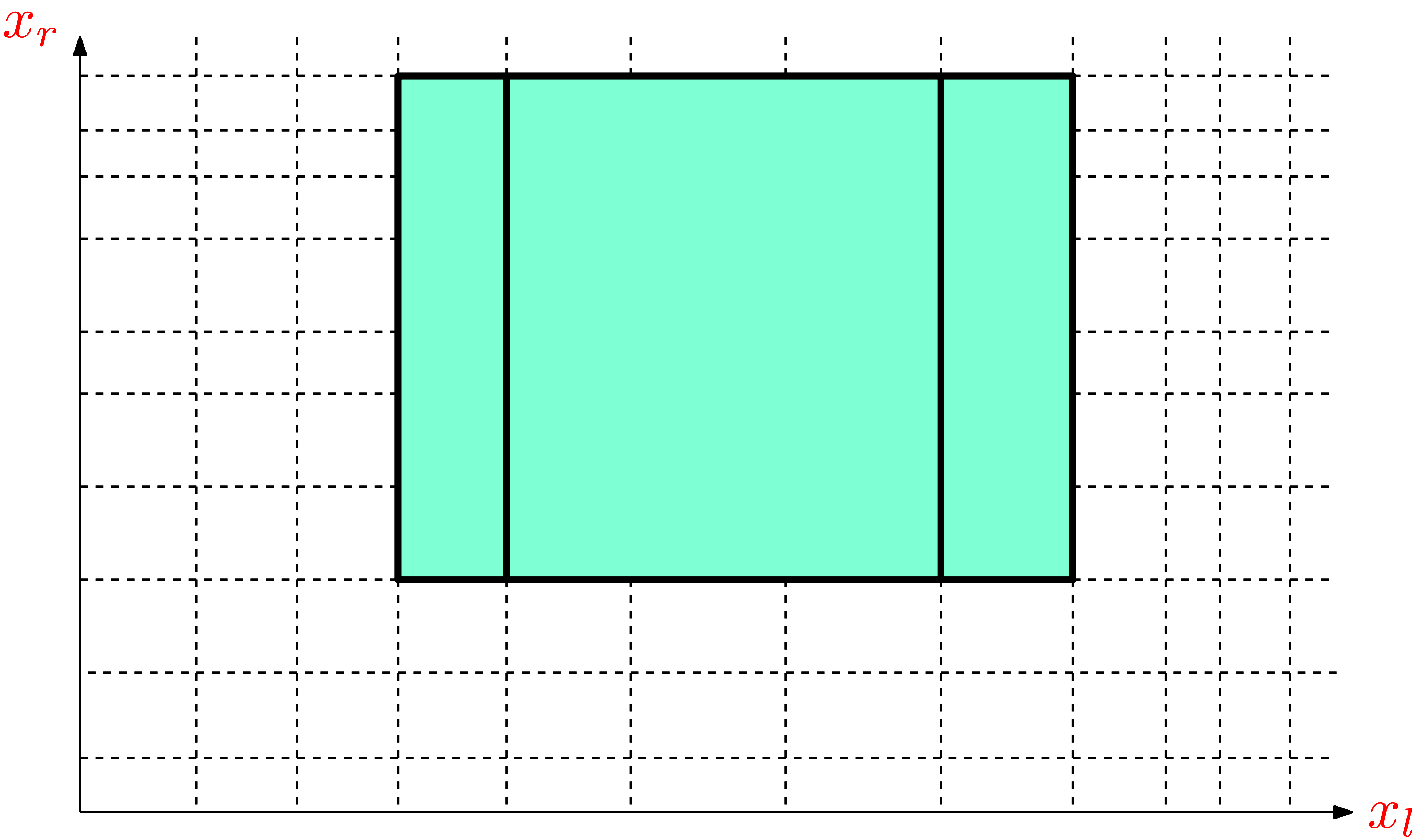


Problem: Indexing a collection of bivariate piecewise-linear functions where each piece spans an orthogonal rectangle, such that for a given query point q in 3D, we can report all functions below q

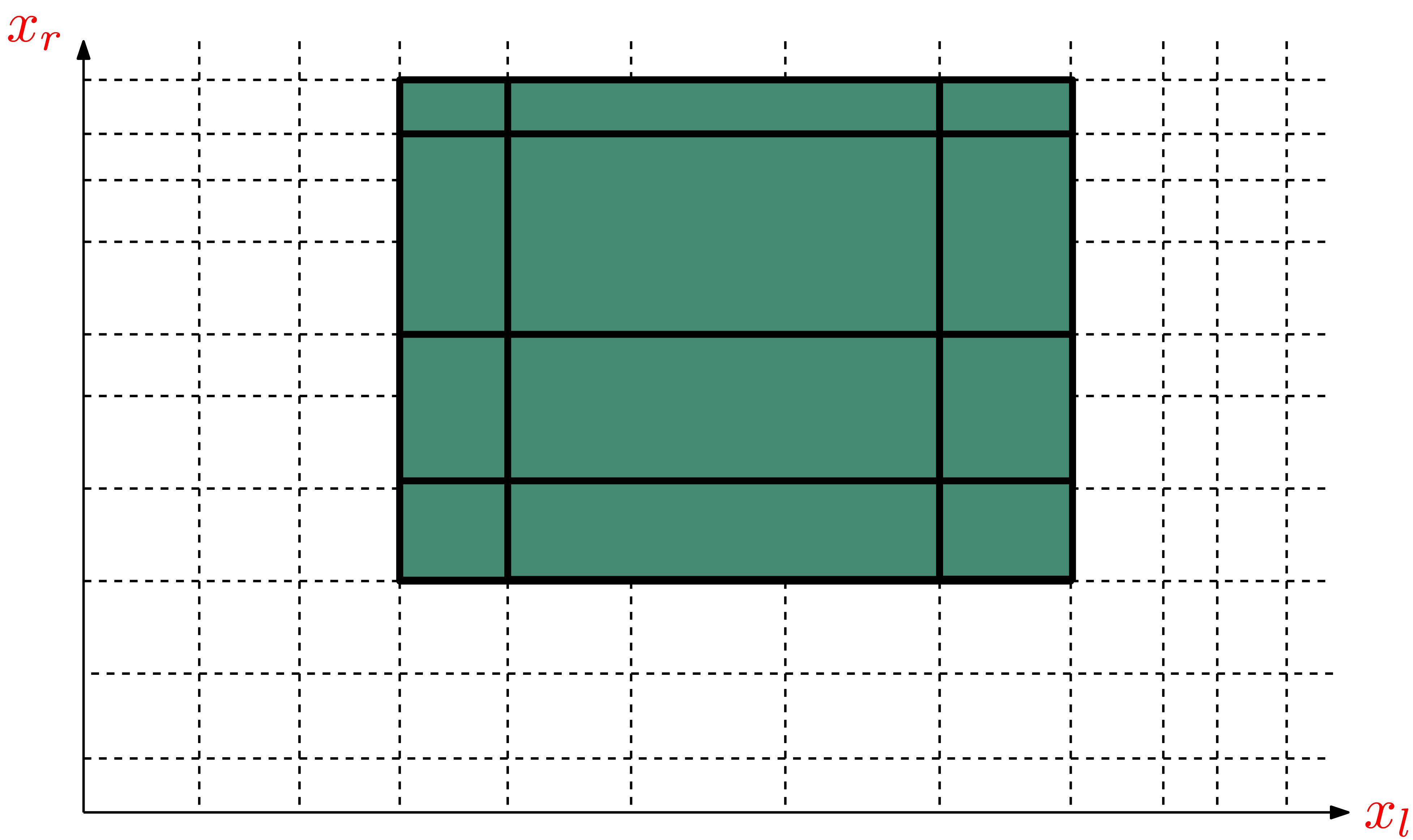
Queries with Variable τ



Queries with Variable τ



Queries with Variable τ



Each piece is decomposed into $O(\log^2 n)$ canonical rectangles

Queries with Variable τ

- For each canonical rectangle, build a 3D halfspace structure
  - Linear size, $O(\log n + k)$ query [Afshani, Chan '09]
- Total size of the index: $O(n \log^2 n)$

Queries with Variable τ

- For each canonical rectangle, build a 3D halfspace structure
  - Linear size, $O(\log n + k)$ query [Afshani, Chan '09]
- Total size of the index: $O(n \log^2 n)$
- Answering a query
  - The query point is covered by $O(\log n)$ canonical vertical slabs
  - Inside each canonical horizontal slabs, there are $O(\log n)$ canonical vertical slabs (rectangles) covering q
  - Need to query $O(\log^2 n)$ 3D halfspace structures
  - Total query time: $O(\log^3 n + k)$

Conclusions

- Queries with a fixed threshold
  - Can solve in linear space and optimal query time
  - A simpler and more general structure, might be of practical interests
- Queries with variable threshold
  - Can solve in $n \log^{O(1)} n$ size and $\log^{O(1)} n$ query

Conclusions

- Queries with a fixed threshold
  - Can solve in linear space and optimal query time
  - A simpler and more general structure, might be of practical interests
- Queries with variable threshold
  - Can solve in $n \log^{O(1)} n$ size and $\log^{O(1)} n$ query
- Problems to consider
  - Higher dimensions?
  - Range counting in uncertain db?
  - Nearest neighbors in uncertain db?