A PTAS for the horizontal rectangle stabbing problem

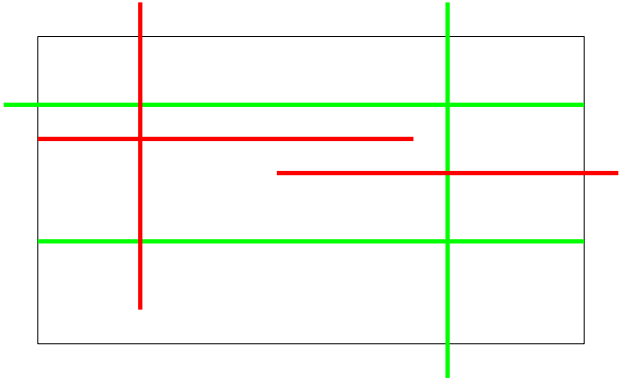
Aditya Subramanian

Indian Institute of Science, Bengaluru

IPCO 2022

Joint work with Arindam Khan (IISc Bengaluru) and Andreas Wiese (TU of Munich)

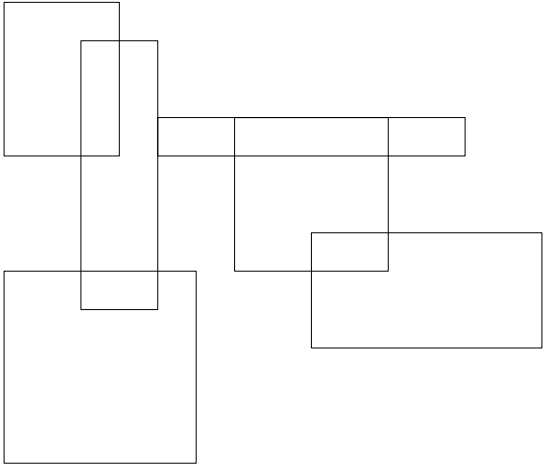
Rectangle Stabbing



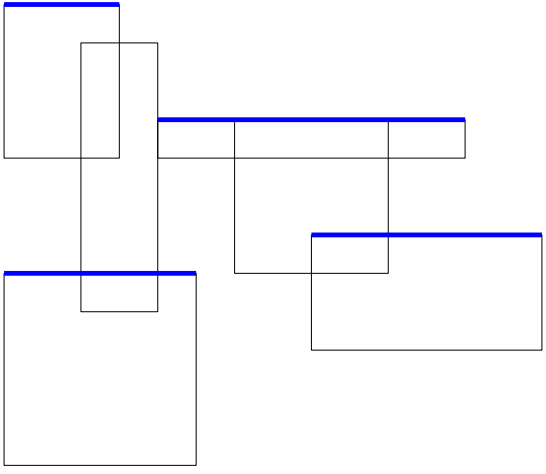
The Rectangle Stabbing Problem

Given: set of n axis-parallel rectangles $\mathcal{R}$ in the two-dimensional plane.
Goal: to compute a set of line segments $\mathcal{L}$ that stab all input rectangles.

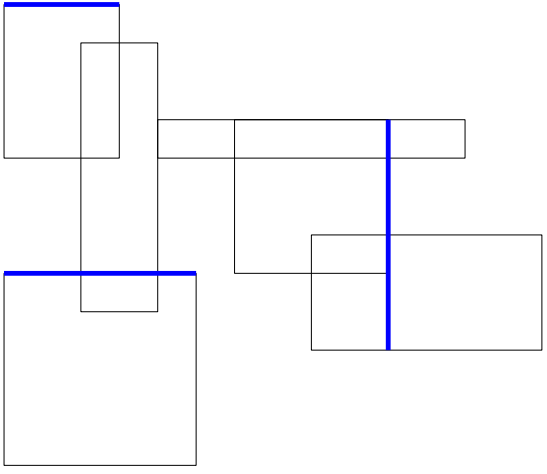
Horizontal Stabbing Problem



Horizontal Stabbing Problem

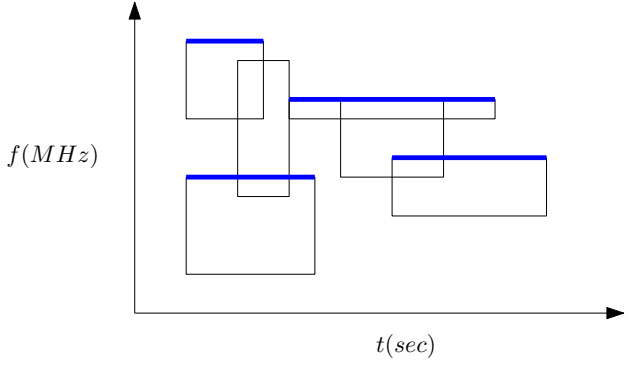


General Stabbing Problem



Motivation

- Can be viewed as a resource allocation problem.



Motivation

- Can be viewed as a resource allocation problem.
- Stabbing is a special case of weighted set cover.
- Also has applications in practical problems like VLSI circuit layout.

Past Work

- $O(1)$-**approximation** [CvDF$^+$18]: Chan, van Dijk, Fleszar, Spoerhase, and Wolff (ISAAC18) started the study of this problem and gave a constant factor approximation for general stabbing.
- $(1+\varepsilon)$-**approximation (**$n^{\text{poly}\log n}$ **time)** [EGSV21]: Recently, Eisenbrand, Gallato, Svensson, and Venzin presented a QPTAS for horizontal stabbing.

This leaves the open question of finding a PTAS for general and horizontal stabbing.

Our Results

- There is a polynomial time $(1+\varepsilon)$-approximation algorithm, for the horizontal stabbing problem.
- There is a polynomial time $(2+\varepsilon)$-approximation algorithm, for the general stabbing problem.
- There is a polynomial time $(1+\varepsilon)$-approximation algorithm, for the general stabbing problem, assuming that each input rectangle has height at least its width.
- There is a PTAS for the general square stabbing problem.

Outline of the talk

- Pre-processing
- Dynamic Programming Algorithm
- Construction of DP recursion tree

Pre-processing

Scale the widths so that $\varepsilon/n < w_i \leq 1$.
This also ensures that all the x-coordinates lie in the range $[0, n]$.

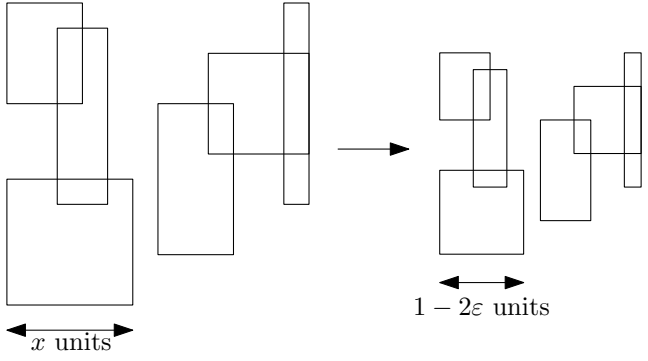


Figure: Preprocessing: Scaling

Pre-processing

Stretch the heights so that all y-coordinates lie in the range $[0, 4n^2]$.

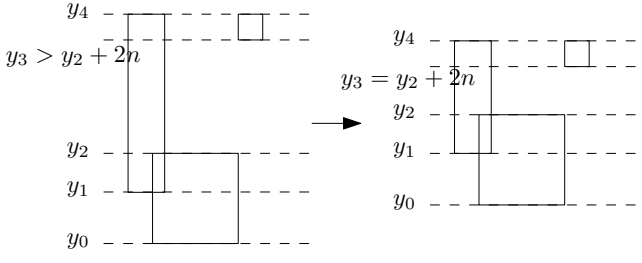


Figure: Preprocessing: Stretching

Pre-processing

Now, we discretize all the x and y-coordinates to multiples of ε/n.

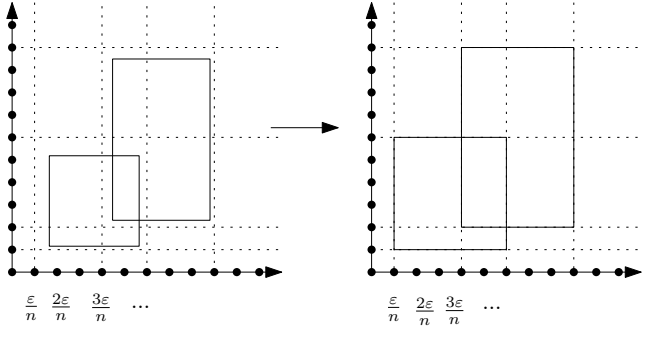


Figure: Preprocessing: Discretization

Pre-processing

Lemma

By losing a factor $(1 + O(\varepsilon))$ *in the approximation ratio, we can assume that the following properties hold:*

(i) *Width of every rectangle is in* $[\frac{\varepsilon}{n}, 1]$,

(ii) *All x-coordinates are discretized and within* $[0, n]$,

(iii) *All y-coordinates are discretized and within* $[0, 4n^2]$.

Dynamic Programming

Now we describe our DP based algorithm, which has a cell $\mathrm{DP}(S, \mathcal{L})$ for each combination of

- a rectangular region $S \subseteq [0, n] \times [0, 4n^2]$ with discretized coordinates.
- a 'small' set $\mathcal{L}$ of at most $3(1/\varepsilon)^3$ line segments with discretized coordinates.

This DP-cell corresponds to the subproblem of stabbing all rectangles contained in S which are not already stabbed by $\mathcal{L}$.

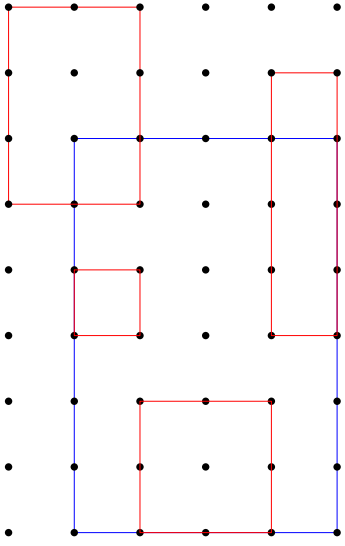


Figure: discretized cells and segments

Dynamic Programming

Now we describe our DP based algorithm, which has a cell $\mathrm{DP}(S, \mathcal{L})$ for each combination of

- a rectangular region $S \subseteq [0, n] \times [0, 4n^2]$ with discretized coordinates.
- a 'small' set $\mathcal{L}$ of at most $3(1/\varepsilon)^3$ line segments with discretized coordinates.

This DP-cell corresponds to the subproblem of stabbing all rectangles contained in S which are not already stabbed by $\mathcal{L}$.

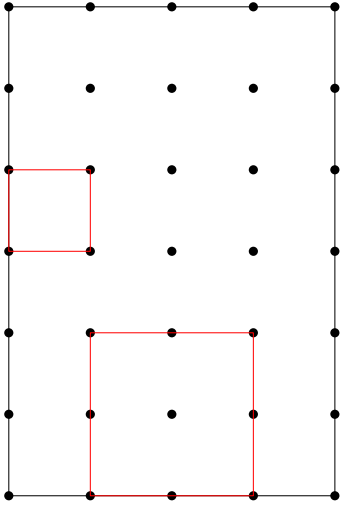


Figure: discretized cells and segments

Dynamic Programming

Now we describe our DP based algorithm, which has a cell $\mathrm{DP}(S, \mathcal{L})$ for each combination of

- a rectangular region $S \subseteq [0, n] \times [0, 4n^2]$ with discretized coordinates.
- a 'small' set $\mathcal{L}$ of at most $3(1/\varepsilon)^3$ line segments with discretized coordinates.

This DP-cell corresponds to the subproblem of stabbing all rectangles contained in S which are not already stabbed by $\mathcal{L}$.

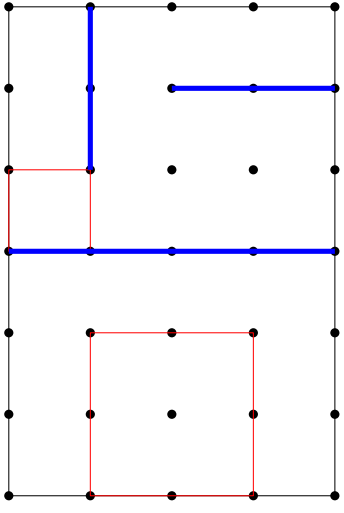


Figure: discretized cells and segments

Dynamic Programming

Now we describe our DP based algorithm, which has a cell $\mathrm{DP}(S, \mathcal{L})$ for each combination of

- a rectangular region $S \subseteq [0, n] \times [0, 4n^2]$ with discretized coordinates.
- a 'small' set $\mathcal{L}$ of at most $3(1/\varepsilon)^3$ line segments with discretized coordinates.

This DP-cell corresponds to the subproblem of stabbing all rectangles contained in S which are not already stabbed by $\mathcal{L}$.

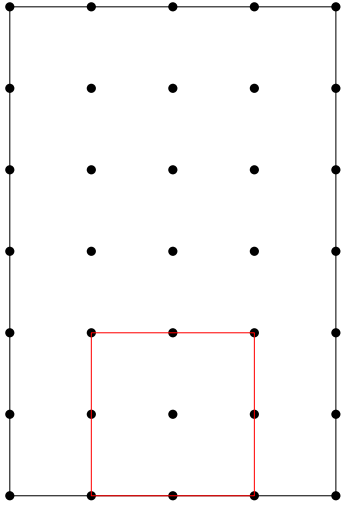
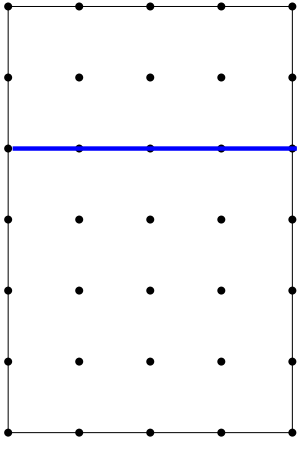


Figure: discretized cells and segments

Dynamic Programming

We do the following operations to produces a set of candidate solutions:

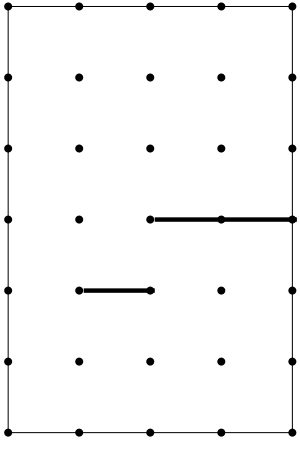
1. Trivial operation: If there is a line segment $\ell \in \mathcal{L}$ such that it divides S into 2 rectangular regions, then we add ℓ to the subproblem solution, and recurse on the 2 smaller rectangular cells.



Dynamic Programming

We do the following operations to produces a set of candidate solutions:

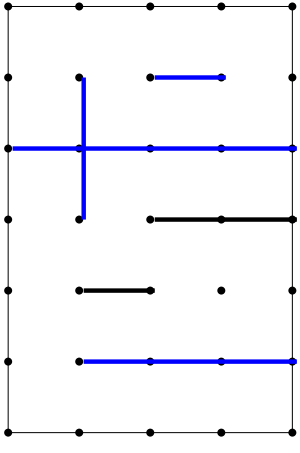
2. Add operation: Consider each 'small sized' set $\mathcal{L}'$, of discretized segments contained in S. We define the solution $\mathcal{L}' \cup \mathrm{SOL}(S, \mathcal{L} \cup \mathcal{L}')$ as a candidate solution.



Dynamic Programming

We do the following operations to produces a set of candidate solutions:

2. Add operation: Consider each 'small sized' set $\mathcal{L}'$, of discretized segments contained in S. We define the solution $\mathcal{L}' \cup \mathrm{SOL}(S, \mathcal{L} \cup \mathcal{L}')$ as a candidate solution.

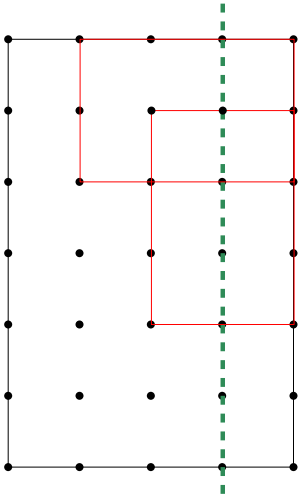


Dynamic Programming

We do the following operations to produces a set of candidate solutions:

3. Line operation: Consider each line ℓ with discretized vertical/horizontal coordinates that divides S into 2 rectangular regions, S_1 and S_2. If $\mathcal{R}_\ell$ are the rectangles (in S) stabbed by ℓ, do the following:

1. compute a $O(1)$-approximate solution $\mathcal{L}(\mathcal{R}_\ell)$, for rectangles in $\mathcal{R}_\ell$ [CvDF$^+$18].
2. produce the candidate solution as union of $\mathcal{L}(\mathcal{R}_\ell)$ and subproblem solutions of S_1 and S_2.

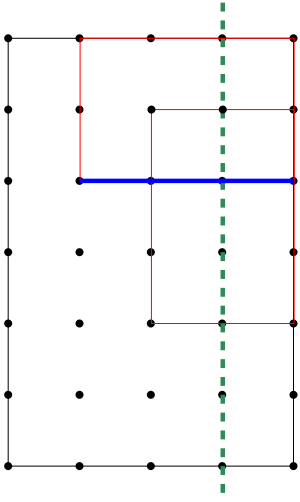


Dynamic Programming

We do the following operations to produces a set of candidate solutions:

3. Line operation: Consider each line ℓ with discretized vertical/horizontal coordinates that divides S into 2 rectangular regions, S_1 and S_2. If $\mathcal{R}_\ell$ are the rectangles (in S) stabbed by ℓ, do the following:

1. compute a $O(1)$-approximate solution $\mathcal{L}(\mathcal{R}_\ell)$, for rectangles in $\mathcal{R}_\ell$ [CvDF$^+$18].
2. produce the candidate solution as union of $\mathcal{L}(\mathcal{R}_\ell)$ and subproblem solutions of S_1 and S_2.

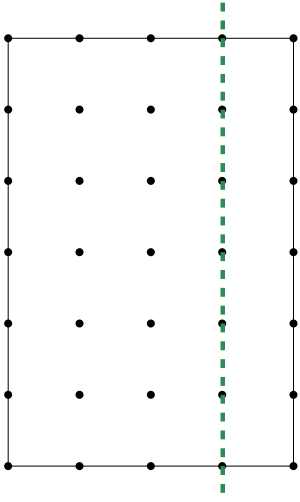


Dynamic Programming

We do the following operations to produces a set of candidate solutions:

3. Line operation: Consider each line ℓ with discretized vertical/horizontal coordinates that divides S into 2 rectangular regions, S_1 and S_2. If $\mathcal{R}_\ell$ are the rectangles (in S) stabbed by ℓ, do the following:

1. compute a $O(1)$-approximate solution $\mathcal{L}(\mathcal{R}_\ell)$, for rectangles in $\mathcal{R}_\ell$ [CvDF$^+$18].
2. produce the candidate solution as union of $\mathcal{L}(\mathcal{R}_\ell)$ and subproblem solutions of S_1 and S_2.



Runtime

- No. of grid points: $O_\varepsilon(\mathrm{poly}(n))$.

Runtime

- No. of grid points: $O_\varepsilon(\mathrm{poly}(n))$.
- No. of rectangular cells: $O_\varepsilon(\mathrm{poly}(n))$.

Runtime

- No. of grid points: $O_\varepsilon(\mathrm{poly}(n))$.
- No. of rectangular cells: $O_\varepsilon(\mathrm{poly}(n))$.
- No. of line segments: $O_\varepsilon(\mathrm{poly}(n))$.

Runtime

- No. of grid points: $O_\varepsilon(\mathrm{poly}(n))$.
- No. of rectangular cells: $O_\varepsilon(\mathrm{poly}(n))$.
- No. of line segments: $O_\varepsilon(\mathrm{poly}(n))$.
- No. of sets of line segments: $O_\varepsilon(\mathrm{poly}(n))$.

Runtime

- No. of grid points: $O_\varepsilon(\mathrm{poly}(n))$.
- No. of rectangular cells: $O_\varepsilon(\mathrm{poly}(n))$.
- No. of line segments: $O_\varepsilon(\mathrm{poly}(n))$.
- No. of sets of line segments: $O_\varepsilon(\mathrm{poly}(n))$.

- No. of DP cells: $O_\varepsilon(\mathrm{poly}(n))$.

Runtime

- No. of grid points: $O_\varepsilon(\mathrm{poly}(n))$.
- No. of rectangular cells: $O_\varepsilon(\mathrm{poly}(n))$.
- No. of line segments: $O_\varepsilon(\mathrm{poly}(n))$.
- No. of sets of line segments: $O_\varepsilon(\mathrm{poly}(n))$.

- No. of DP cells: $O_\varepsilon(\mathrm{poly}(n))$.
- Possible no. of line operations: $O_\varepsilon(\mathrm{poly}(n))$.

Runtime

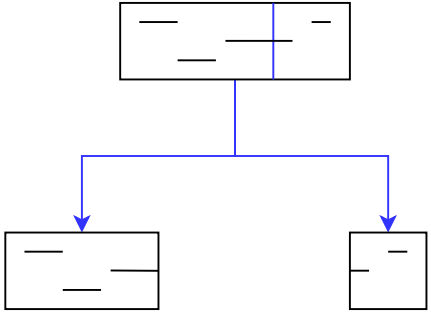
- No. of grid points: $O_\varepsilon(\mathrm{poly}(n))$.
- No. of rectangular cells: $O_\varepsilon(\mathrm{poly}(n))$.
- No. of line segments: $O_\varepsilon(\mathrm{poly}(n))$.
- No. of sets of line segments: $O_\varepsilon(\mathrm{poly}(n))$.

- No. of DP cells: $O_\varepsilon(\mathrm{poly}(n))$.
- Possible no. of line operations: $O_\varepsilon(\mathrm{poly}(n))$.
- Possible no. of add operations: $O_\varepsilon(\mathrm{poly}(n))$ (note that trivial operations on any segment can be charged to the corresponding add operation on the same segment).

This brings the total runtime of our algorithm to $O_\varepsilon(\mathrm{poly}(n))$ as required.

DP Decision Tree

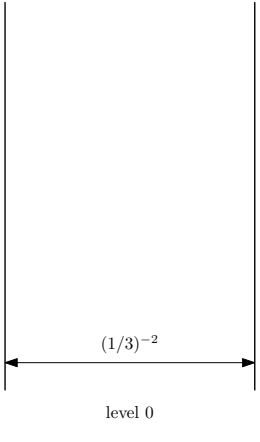
We define a DP decision tree, as a tree that describes any valid run of a recursive algorithm that breaks down a given subproblem into smaller subproblems using one of the 3 valid DP operations.



Tree construction

We define now a DP-decision-tree for which $cost(\overline{\mathrm{SOL}}(S, \mathcal{L})) \leq (1+\varepsilon)\mathrm{OPT}$.

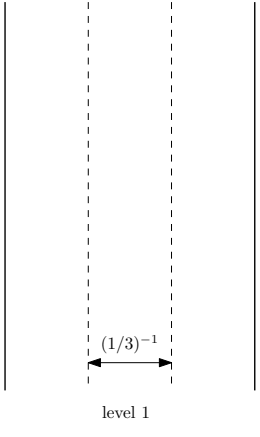
- We start by defining a hierarchical grid of vertical lines. The grid lines have levels. For each level $j \in \mathbb{N}_0$, there is a grid line $\{a + k \cdot \varepsilon^{j-2}\} \times \mathbb{R}$ for each $k \in \mathbb{N}$, and random offset a.



Tree construction

We define now a DP-decision-tree for which $cost(\overline{\mathrm{SOL}}(S, \mathcal{L})) \leq (1+\varepsilon)\mathrm{OPT}$.

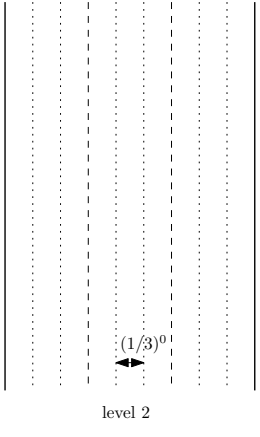
- We start by defining a hierarchical grid of vertical lines. The grid lines have levels. For each level $j \in \mathbb{N}_0$, there is a grid line $\{a + k \cdot \varepsilon^{j-2}\} \times \mathbb{R}$ for each $k \in \mathbb{N}$, and random offset a.



Tree construction

We define now a DP-decision-tree for which $cost(\overline{\mathrm{SOL}}(S, \mathcal{L})) \leq (1+\varepsilon)\mathrm{OPT}$.

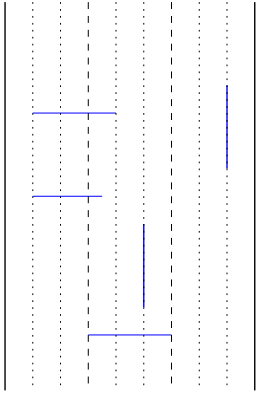
- We start by defining a hierarchical grid of vertical lines. The grid lines have levels. For each level $j \in \mathbb{N}_0$, there is a grid line $\{a + k \cdot \varepsilon^{j-2}\} \times \mathbb{R}$ for each $k \in \mathbb{N}$, and random offset a.



Tree construction

We define now a DP-decision-tree for which $cost(\overline{\mathrm{SOL}}(S, \mathcal{L})) \leq (1+\varepsilon)\mathrm{OPT}$.

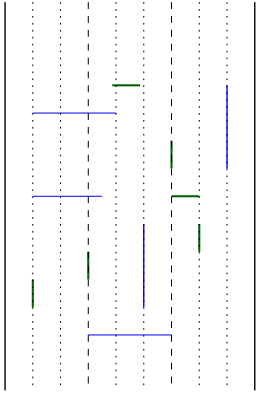
- We start by defining a hierarchical grid of vertical lines. The grid lines have levels. For each level $j \in \mathbb{N}_0$, there is a grid line $\{a + k \cdot \varepsilon^{j-2}\} \times \mathbb{R}$ for each $k \in \mathbb{N}$, and random offset a.
- We say that a line segment $\ell \in \mathrm{OPT}$ is of level j if $|\ell| \in (\varepsilon^j, \varepsilon^{j-1}]$.



Tree construction

We define now a DP-decision-tree for which $cost(\overline{\mathrm{SOL}}(S, \mathcal{L})) \leq (1+\varepsilon)\mathrm{OPT}$.

- We start by defining a hierarchical grid of vertical lines. The grid lines have levels. For each level $j \in \mathbb{N}_0$, there is a grid line $\{a + k \cdot \varepsilon^{j-2}\} \times \mathbb{R}$ for each $k \in \mathbb{N}$, and random offset a.
- We say that a line segment $\ell \in \mathrm{OPT}$ is of level j if $|\ell| \in (\varepsilon^j, \varepsilon^{j-1}]$.

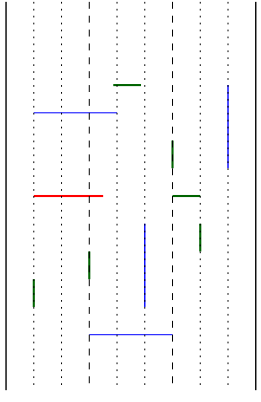


level 1

Tree construction

We define now a DP-decision-tree for which $cost(\overline{\mathrm{SOL}}(S, \mathcal{L})) \leq (1+\varepsilon)\mathrm{OPT}$.

- We start by defining a hierarchical grid of vertical lines. The grid lines have levels. For each level $j \in \mathbb{N}_0$, there is a grid line $\{a + k \cdot \varepsilon^{j-2}\} \times \mathbb{R}$ for each $k \in \mathbb{N}$, and random offset a.
- We say that a line segment $\ell \in \mathrm{OPT}$ is of level j if $|\ell| \in (\varepsilon^j, \varepsilon^{j-1}]$.
- We say that a line segment of some level j is well-aligned if both its end-points lie on a vertical(/imaginary horizontal) grid line of level $j+3$.



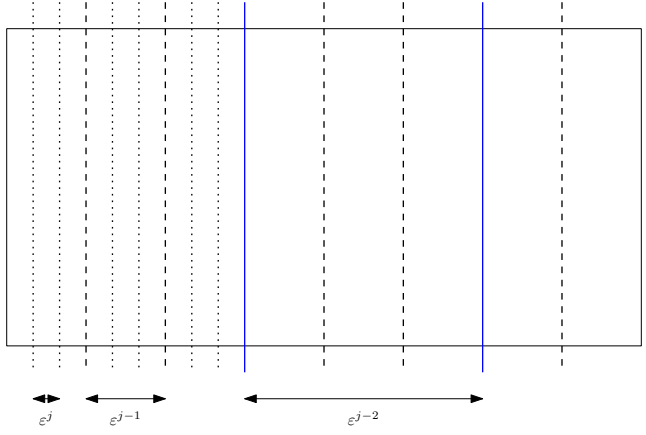
Tree construction

Lemma

By losing a factor $1 + O(\varepsilon)$*, we can assume that each line segment* $\ell \in \mathrm{OPT}$ *is well-aligned.*

Tree construction
(at level j)

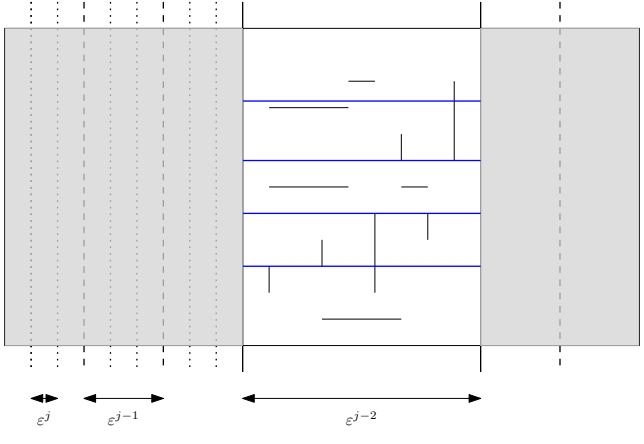
At some level j, we do line operations along the grid lines of that level.



Tree construction

(at level j)

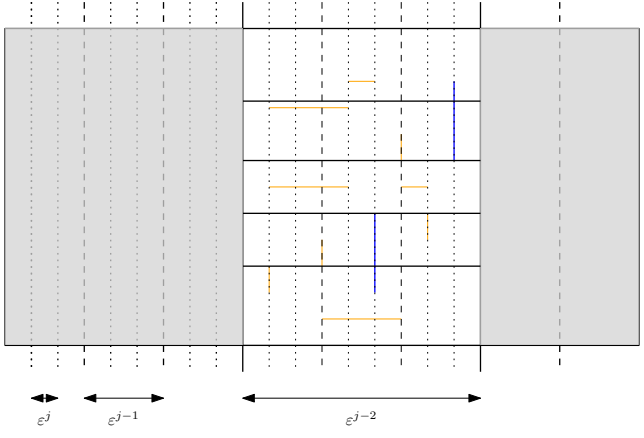
In a subproblem thus created, if there are more than ε^{-3} end points of segments from OPT of level j, do a horizontal line operation to divide into further cells.



Tree construction

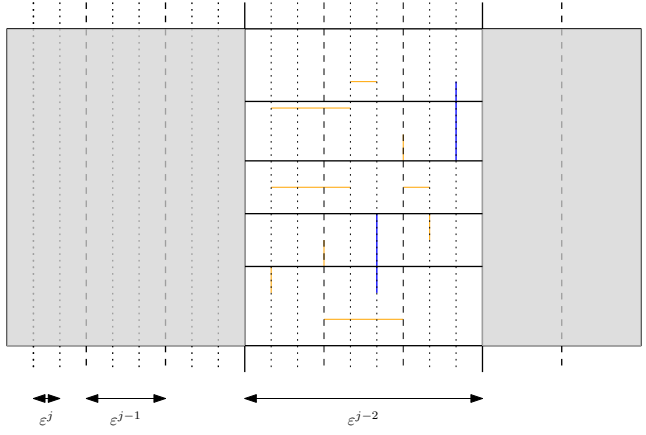
(at level j)

This is where the magic happens! Do trivial operations, and line operations on segments from level $j-2$.



Tree construction
(at level j)

Do appropriate add operations.



Random offset gives good solution

Lemma

There is a choice for the offset a, such that the solution $\overline{\mathrm{SOL}}([0, n] \times [0, 4n^2], \emptyset)$ in T has a cost of at most $(1 + O(\varepsilon))\mathrm{OPT}$.

Theorem

There is a polynomial time $(1 + \varepsilon)$-approximation algorithm, for the general stabbing problem, assuming that each input rectangle has height greater than its width.

Results

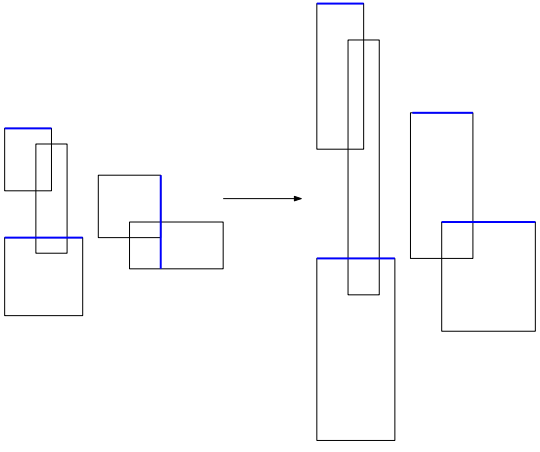
Corollary

There is a PTAS for the general square stabbing problem.

Results

Corollary

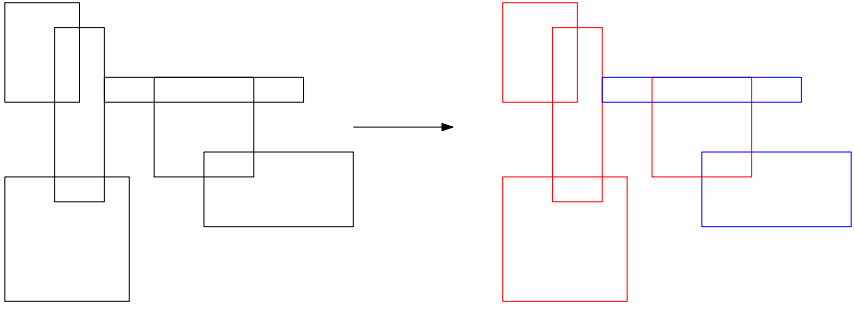
There is a polynomial time $(1+\varepsilon)$-approximation algorithm, for the horizontal stabbing problem.



Results

Corollary

There is a polynomial time $(2+\varepsilon)$-approximation algorithm, for the general stabbing problem.



Open Questions

- Is there a PTAS possible for the general rectangle stabbing problem? Can ideas from our DP be reused?
- Does Cardinality or Constrained Stabbing admit a constant factor approximation?

PTAS for δ-large Rectangles

- Preprocess the input (as before) incurring a factor $(1 + \varepsilon)$ loss.
- Do vertical line operations along level 0 grid lines, and horizontal line operations to divide instance into smaller cells.
- There are at most $O_{\varepsilon,\delta}(1)$ 'large' segments in a cell. Guess them by enumeration.
- The remaining instance can now be partitioned into two disjoint instances with $h_i \geq w_i$ and $w_i > h_i$.

PTAS for δ-large Rectangles

Theorem

For Gen-Stabbing *with δ-large rectangles, there is a $(1+\varepsilon)$-approximation algorithm with a running time of $(n/\varepsilon)^{O(1/\delta\varepsilon^3)}$.*


Timothy M. Chan, Thomas C. van Dijk, Krzysztof Fleszar, Joachim Spoerhase, and Alexander Wolff.
Stabbing rectangles by line segments - how decomposition reduces the shallow-cell complexity.
In Wen-Lian Hsu, Der-Tsai Lee, and Chung-Shou Liao, editors, *ISAAC*, volume 123, pages 61:1–61:13, 2018.


Friedrich Eisenbrand, Martina Gallato, Ola Svensson, and Moritz Venzin.
A QPTAS for stabbing rectangles.
CoRR, abs/2107.06571, 2021.