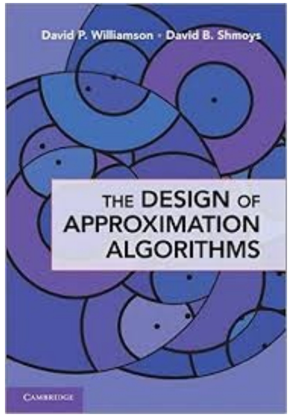
Approximation Algorithms for Maximum Independent Set of Rectangles

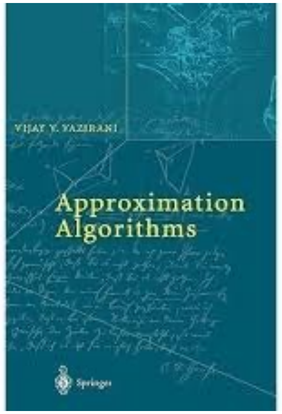
Arindam Khan
IISc Bangalore, India

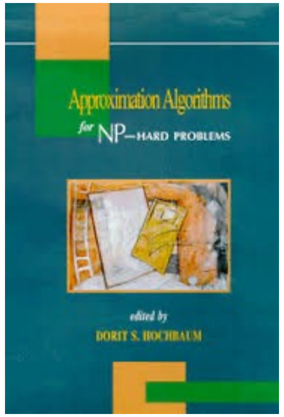
Joint work with Waldo Galvez (TU Munich),
Madhusudhan Reddy (IITKGP → CMU), Mathieu Mari (U Warsaw),
Tobias Momke (U Augsburg), Andreas Wiese (Vrije U.)

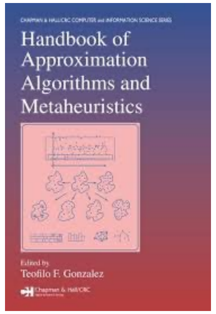
Approximation Algorithms for Maximum Independent Set of Rectangles

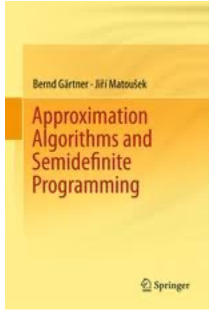
- Approximation algorithms are efficient algorithms that find near-optimal solution.
- For a minimization problem, an algorithm $\mathcal{A}$ is α-(absolute) approximation (α>1) if $\mathcal{A}\ (I) \leq \alpha\ OPT(I)$ ∀ input instances $I \in \mathcal{I}$.
- For a maximization problem, an algorithm $\mathcal{A}$ is α-(absolute) approximation (α>1) if OPT$(I) \leq\ \alpha\ \mathcal{A}\ (I)$ ∀ input instances $I \in \mathcal{I}$.
- For a minimization problem, an algorithm $\mathcal{A}$ is α-asymptotic approximation (α>1) if $\alpha = \limsup_{n\to\infty} \left\{ \sup_{I\in\mathcal{I}} \frac{\mathcal{A}(I)}{OPT(I)} \middle| OPT(I) = n \right\}$ ∀ input instances $I \in \mathcal{I}$.

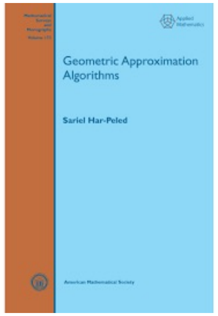


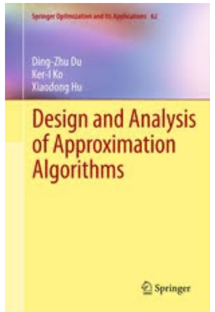












PTAS

- Polynomial Time Approximation Schemes (PTAS):
  If for every $\varepsilon > 0$, there exists a poly-time ($O(n^{f(\varepsilon)})$-time) algorithm A_ε such that $A_\varepsilon(I) \leq (1+\varepsilon)\, OPT(I)$.
- Efficient PTAS (EPTAS): if running time is $O(f(\varepsilon).n^c)$.
- Fully PTAS (FPTAS): if running time is $O((n/\varepsilon)^c)$.

- APX-hardness implies no PTAS.
- W[1]-hardness implies no EPTAS.
- Strong NP-hardness implies no FPTAS.

PTAS

- Asymptotic PTAS (APTAS): $A_\varepsilon(I) \ \leq \ (1+\varepsilon)\, OPT(I) + O(1)$.
- QuasiPTAS (QPTAS): $(1+\varepsilon)$-approximation in $n^{(\log n)^{O_\epsilon(1)}}$-time.
- PseudoPTAS (PPTAS): $(1+\varepsilon)$-approximation in $n^{O_\epsilon(1)}$-time, where n is the
  number of items and the numeric data is polynomially bounded in n.

- QPTAS implies not APX-hard unless $NP \subseteq DTIME\ (2^{poly(\log n)})$.
- So if a problem has QPTAS we expect it to have PTAS.

Approximation Algorithms for Maximum Independent Set of Rectangles

- We love rectangle.

Packing Problems: Placement of objects nonoverlappingly under some constraints

On Packing Squares with Equal Squares

P. Erdős

Stanford University and The Hungarian Academy of Sciences

AND

R. L. Graham

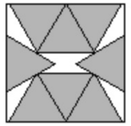
Bell Laboratories, Murray Hill, New Jersey

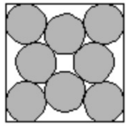
Communicated by the Managing Editors

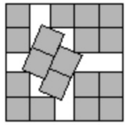
Received November 11, 1974

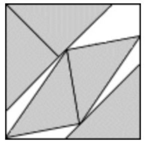
Erich's Packing Center

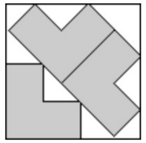
Packing Equal Copies


Triangles in Squares
updated 8/5/12


Circles in Squares
updated 10/9/10


Squares in Squares
updated 11/5/05


Tans in Squares
updated 3/7/08


L's in Squares
updated 5/4/12

PERFECTLY PACKING A SQUARE BY SQUARES OF NEARLY HARMONIC SIDELENGTH

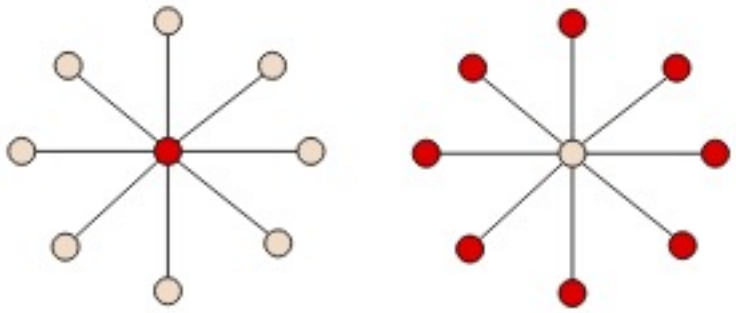
TERENCE TAO

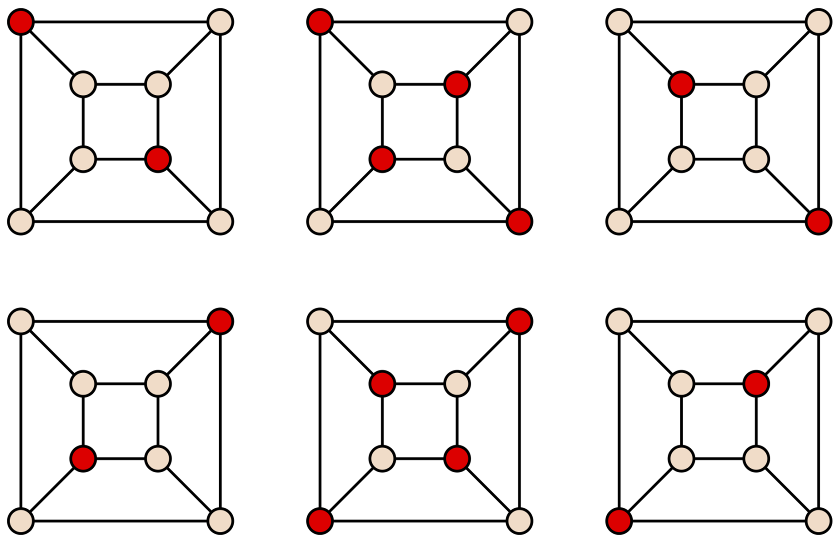
Abstract. A well known open problem of Meir and Moser asks if the squares of sidelength $1/n$ for $n \geq 1$ can be packed perfectly into a square of area $\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$. In this paper we show that for any $1/2 < t < 1$, and any n_0 that is sufficiently large depending on t, the squares of sidelength n^{-t} for $n \geq n_0$ can be packed perfectly into a square of area $\sum_{n=n_0}^{\infty} \frac{1}{n^{2t}}$. This was previously known for $1/2 < t < 2/3$ (in which case one can take $n_0 = 1$).

"I think packing problems are appealing to mathematicians and computer scientists because they seem very simple – just place these items into the container. Yet they tend to be extremely complicated to actually solve."
-- Erik Demaine (MIT).

Approximation Algorithms for Maximum Independent Set of Rectangles

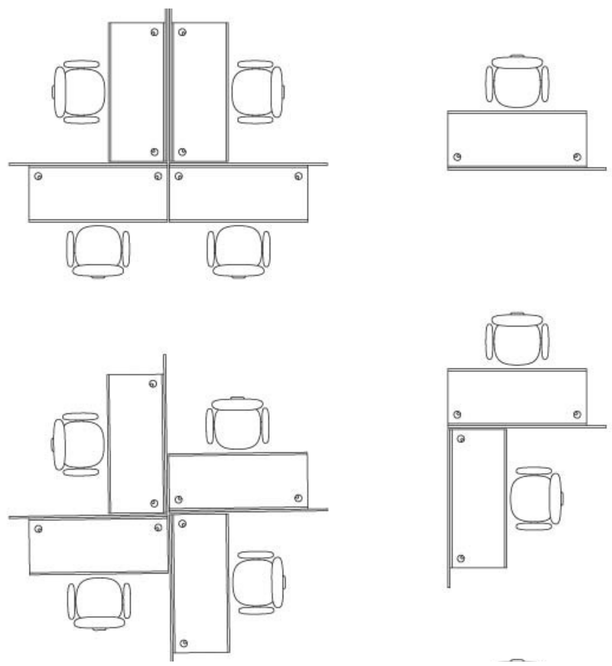
- Independent set is a set of vertices in a graph, s.t. no two vertices are adjacent.
- MIS: Find the maximum sized independent set.
- Classical NP-hard problem.
- Trivial to get n-approximation.
- $\tilde{O}(\frac{n}{\log^3 n})$-approximation [Feige'04],.
- NP-hard to get $n^{1-\epsilon}$-approximation, assuming $NP \nsubseteq ZPP$. [Hastad' 99].
- Hardness: $\Omega(\frac{n}{\exp(\log^{\frac{3}{4}+\epsilon} n)})$. [Khot-Ponnuswamy'06]





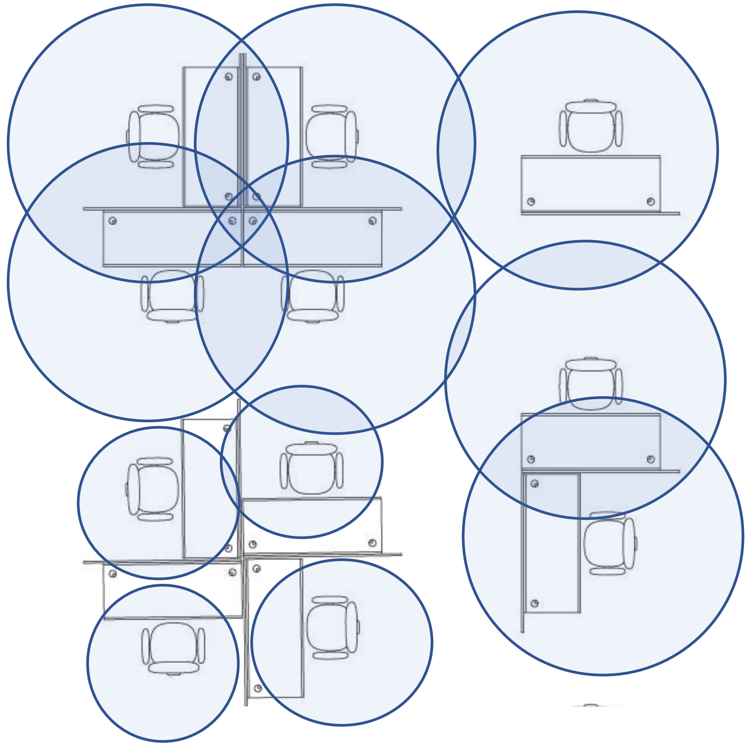
Geometric Intersection Graph

- Nodes correspond to geometric objects (e.g. polygons, spheres, …).
- There is an edge (u, v) if the objects corresponding to u and v overlap.



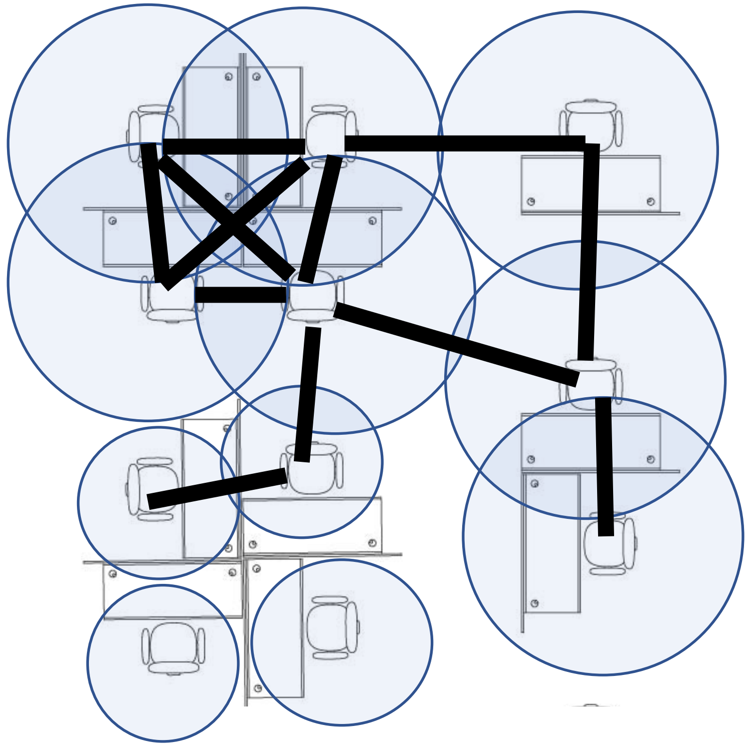
Geometric Intersection Graph

- Nodes correspond to geometric objects (e.g. polygons, spheres, …).
- There is an edge (u, v) if the objects corresponding to u and v overlap.



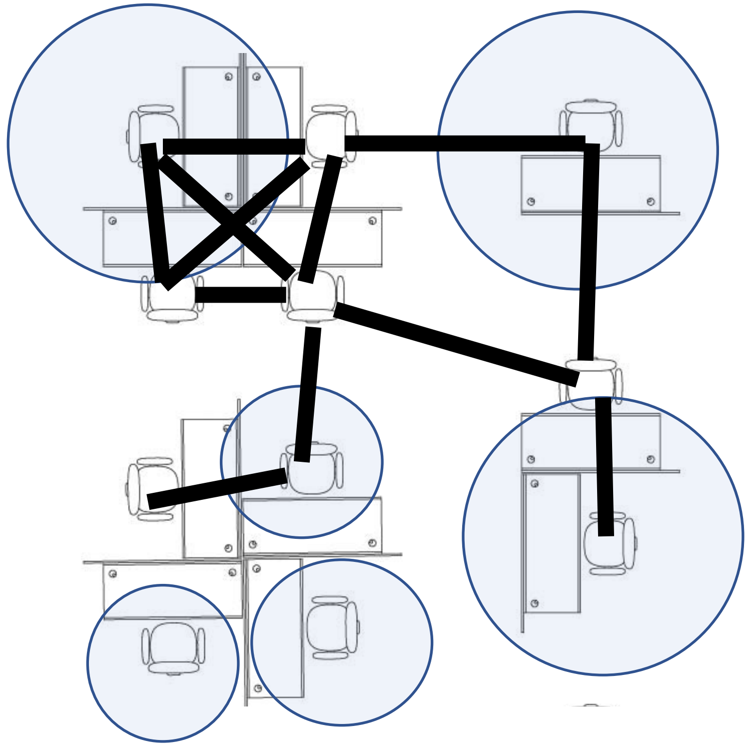
Geometric Intersection Graph

- Nodes correspond to geometric objects (e.g. polygons, spheres, …).
- There is an edge (u, v) if the objects corresponding to u and v overlap.



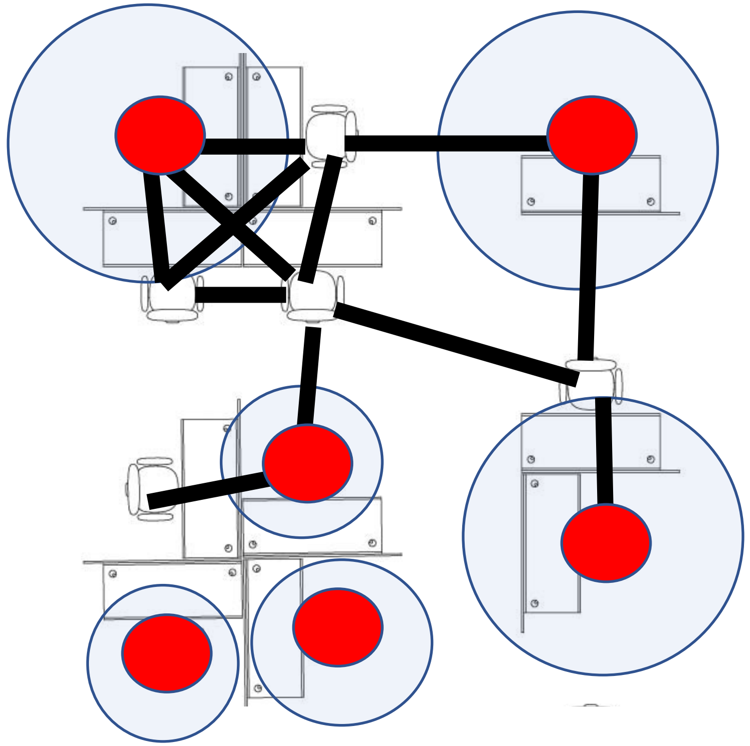
Geometric Intersection Graph

- Nodes correspond to geometric objects (e.g. polygons, spheres, …).
- There is an edge (u, v) if the objects corresponding to u and v overlap.
- Find the maximum independent set in geometric intersection graph.



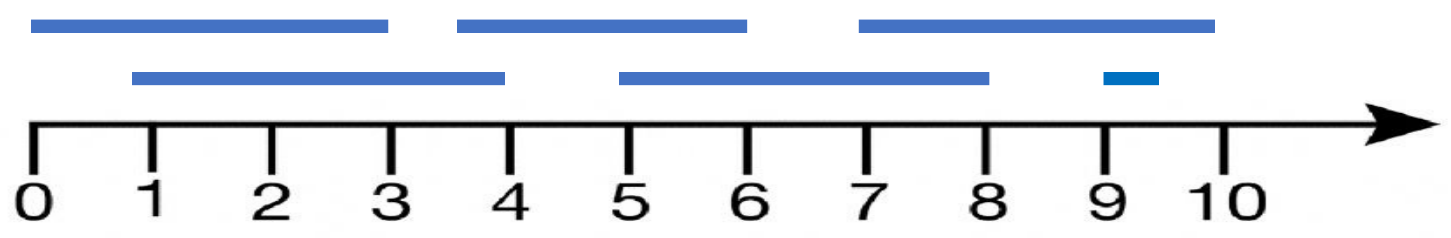
Geometric Intersection Graph

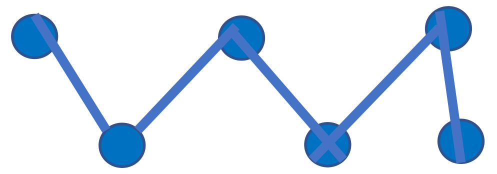
- Nodes correspond to geometric objects (e.g. polygons, spheres, …).
- There is an edge (u, v) if the objects corresponding to u and v overlap.
- Find the maximum independent set in geometric intersection graph.



MISI (Maximum Independent Set of Intervals)

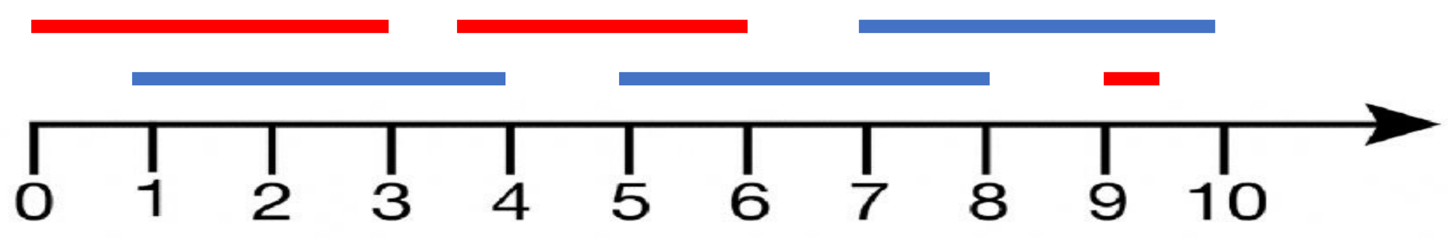
- Nodes corrs. to intervals.
- There is an edge (u, v) if the intervals corresponding to u and v overlap.
- Find the maximum independent set in geometric intersection graph (equivalently maximum cardinality nonoverlapping intervals).
- Polynomial time solvable for intersection graph of intervals.

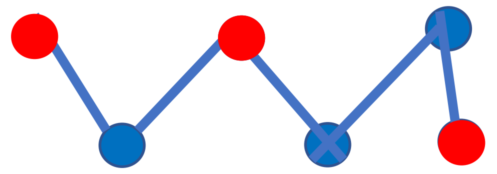




MISI (Maximum Independent Set of Intervals)

- Nodes corrs. to intervals.
- There is an edge (u, v) if the intervals corresponding to u and v overlap.
- Find the maximum independent set in geometric intersection graph (equivalently maximum cardinality nonoverlapping intervals).
- Polynomial time solvable for intersection graph of intervals.

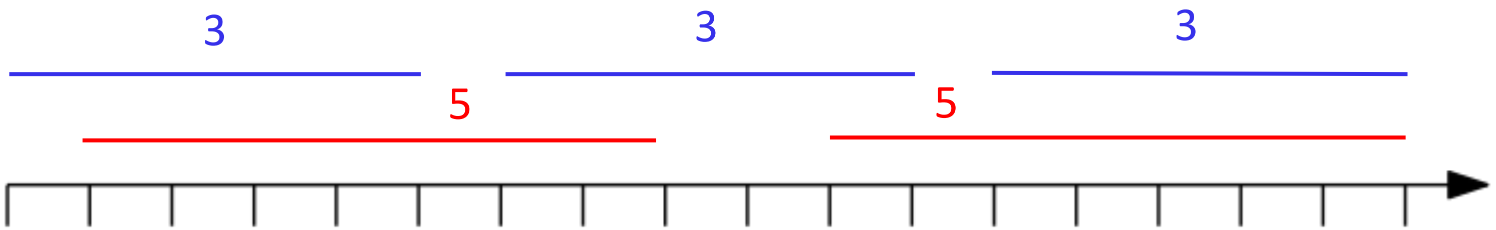




- Greedy: Earliest Finish Time (unweighted), DP (weighted).

MWISI (Maximum Independent Set of Intervals)

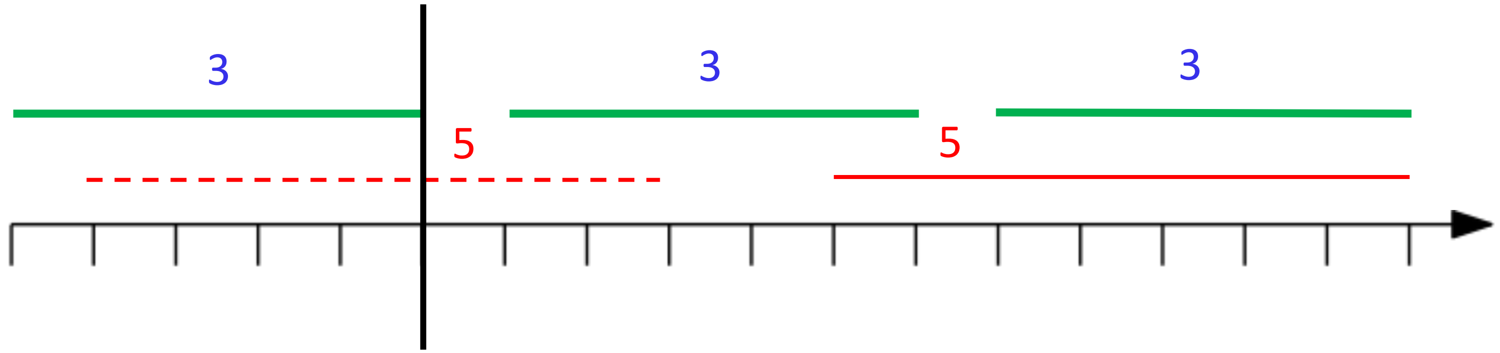
Question: How do you solve Max-Weight indep. set of intervals on a line?



- Divide and Conquer?

MWISI (Maximum Independent Set of Intervals)

Question: How do you solve Max-Weight indep. set of intervals on a line?

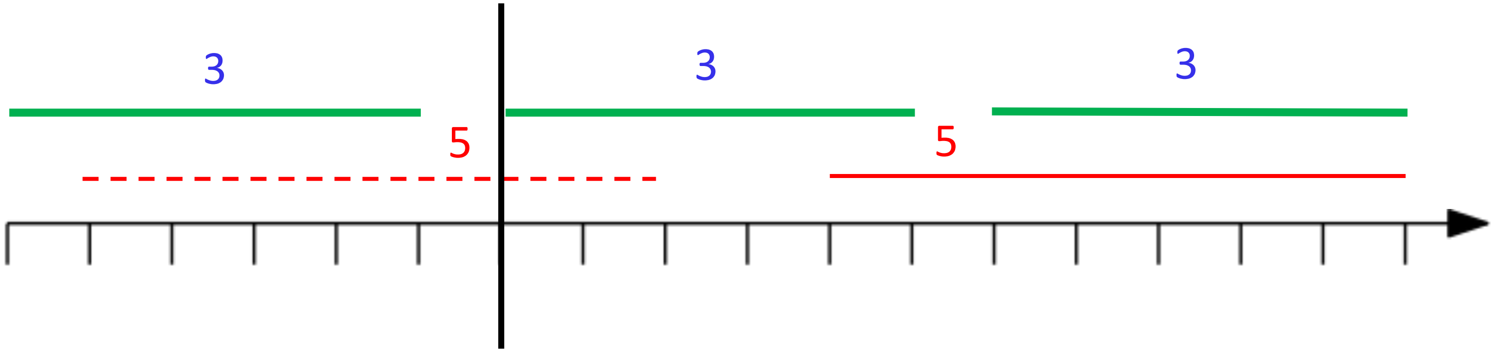


Cut, Solve recursively, Patch

- Divide and Conquer?

MWISI (Maximum Independent Set of Intervals)

Question: How do you solve Max-Weight indep. set of intervals on a line?

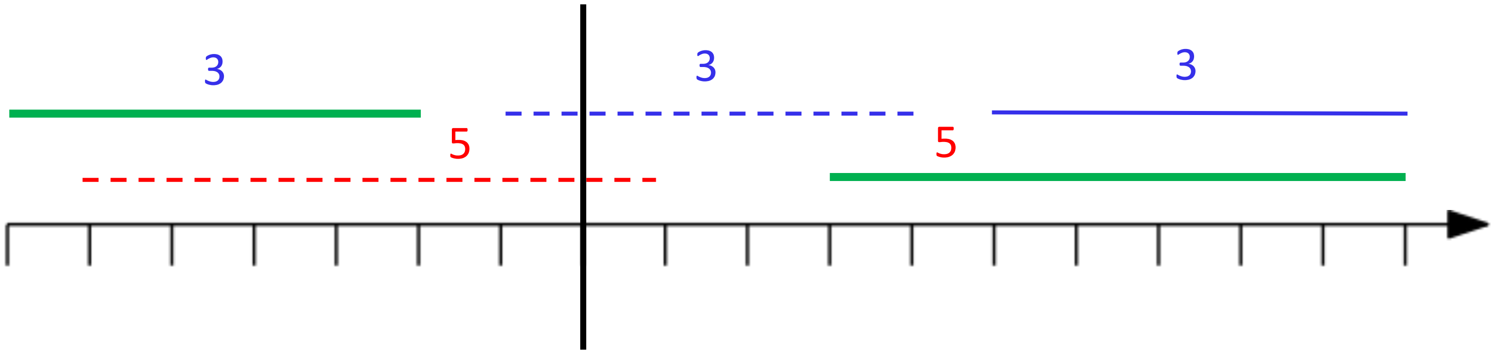


Cut, Solve recursively, Patch

- Divide and Conquer?

MWISI (Maximum Independent Set of Intervals)

Question: How do you solve Max-Weight indep. set of intervals on a line?

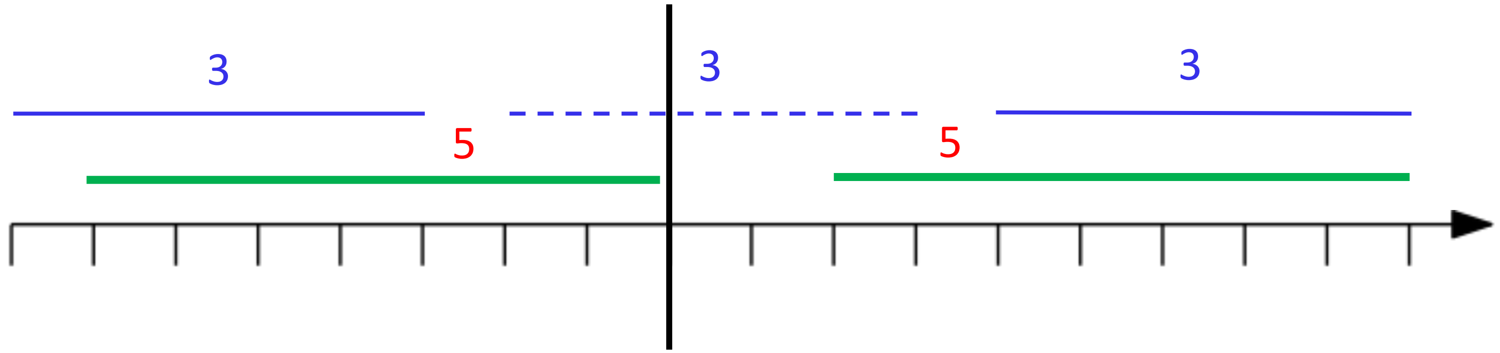


Cut, Solve recursively, Patch

- Divide and Conquer?

MWISI (Maximum Independent Set of Intervals)

Question: How do you solve Max-Weight indep. set of intervals on a line?

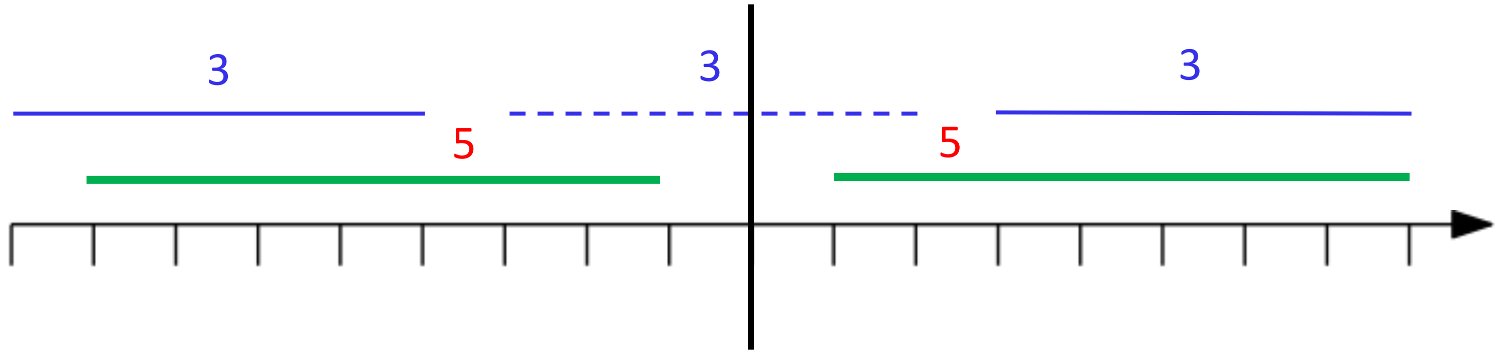


Cut, Solve recursively, Patch

- Divide and Conquer?

MWISI (Maximum Independent Set of Intervals)

Question: How do you solve Max-Weight indep. set of intervals on a line?

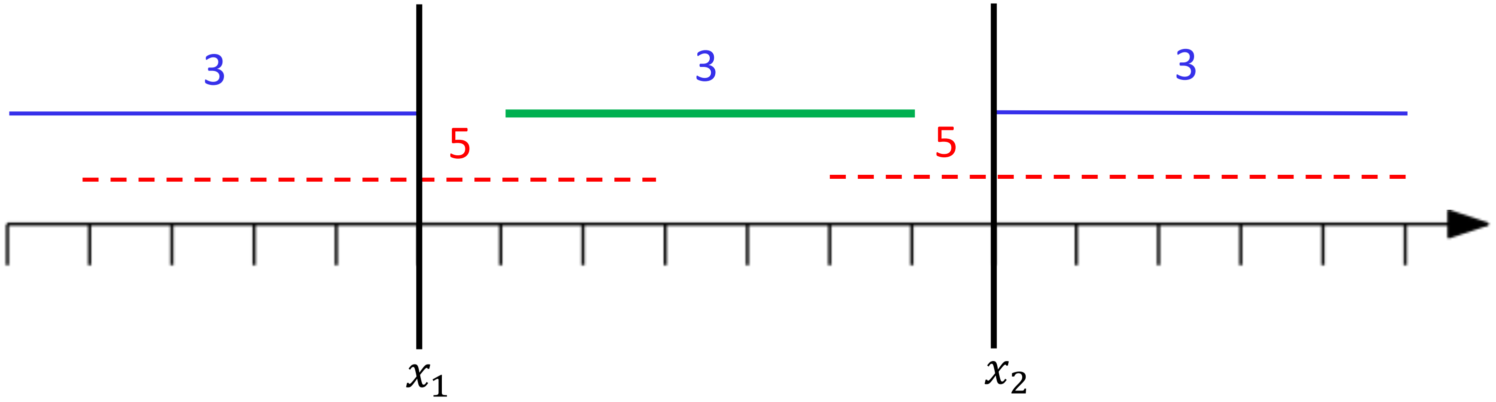


Cut, Solve recursively, Patch

- Divide and Conquer?
- Can be implemented as a dynamic program
- DP$[x_1, x_2]$ contains the optimal solution in the range $[x_1, x_2]$

MWISI (Maximum Independent Set of Intervals)

Question: How do you solve Max-Weight indep. set of intervals on a line?



Cut, Solve recursively, Patch

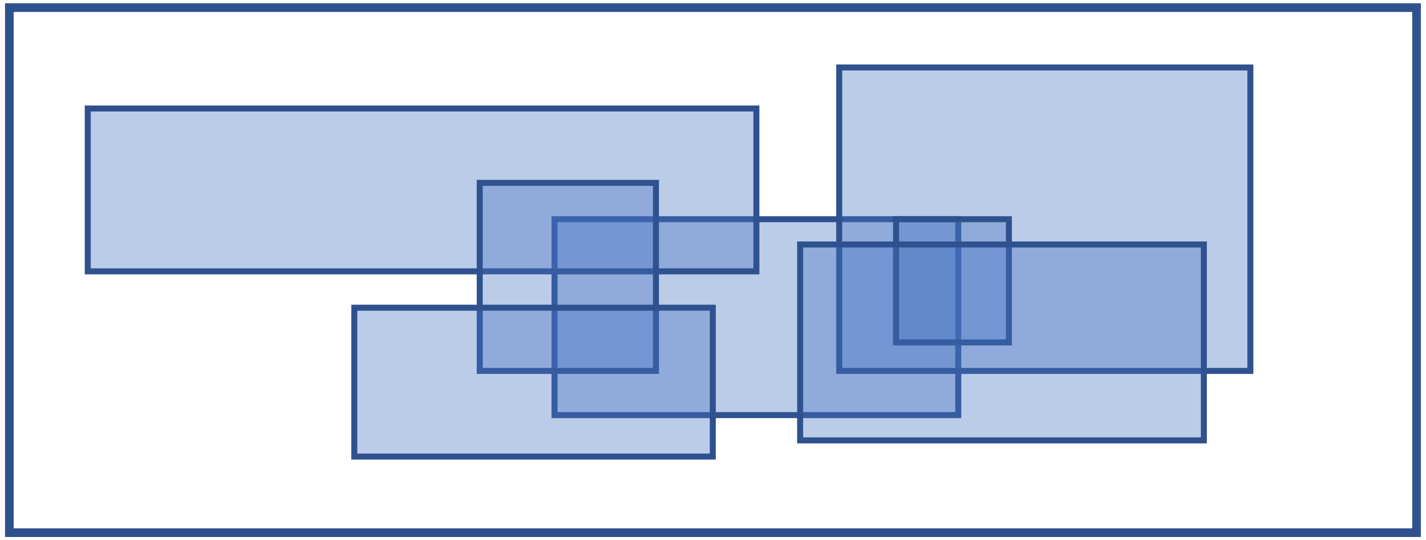
- Divide and Conquer?
- Can be implemented as a dynamic program
- DP[x_1, x_2] contains the optimal solution in the range [x_1, x_2]

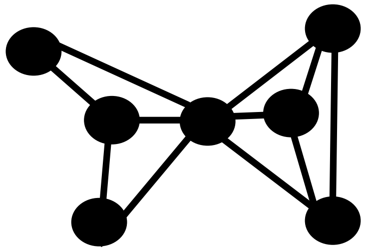
MISS (Maximum Independent Set of Squares)

- A simple greedy algorithm gives 4-approximation for unweighted case.
- Can be extended to disks and fat objects.
- PTAS for unit squares
  [Shifted grid: Hochbaum-Maass’85]
- PTAS for arbitrary squares (even with weights)
  [Shifted Hierarchical Decomposition: Erlebach-Jansen-Siedel’01, Quadtrees: Chan’03].
- PTAS for arbitrary squares [Geometric Separator Theorem: Smith-Wormald’98, Chan’03].
- PTAS for arbitrary squares [Local Search: Chan-HarPeled’09].

Maximum Independent Set of Rectangles (MISR)

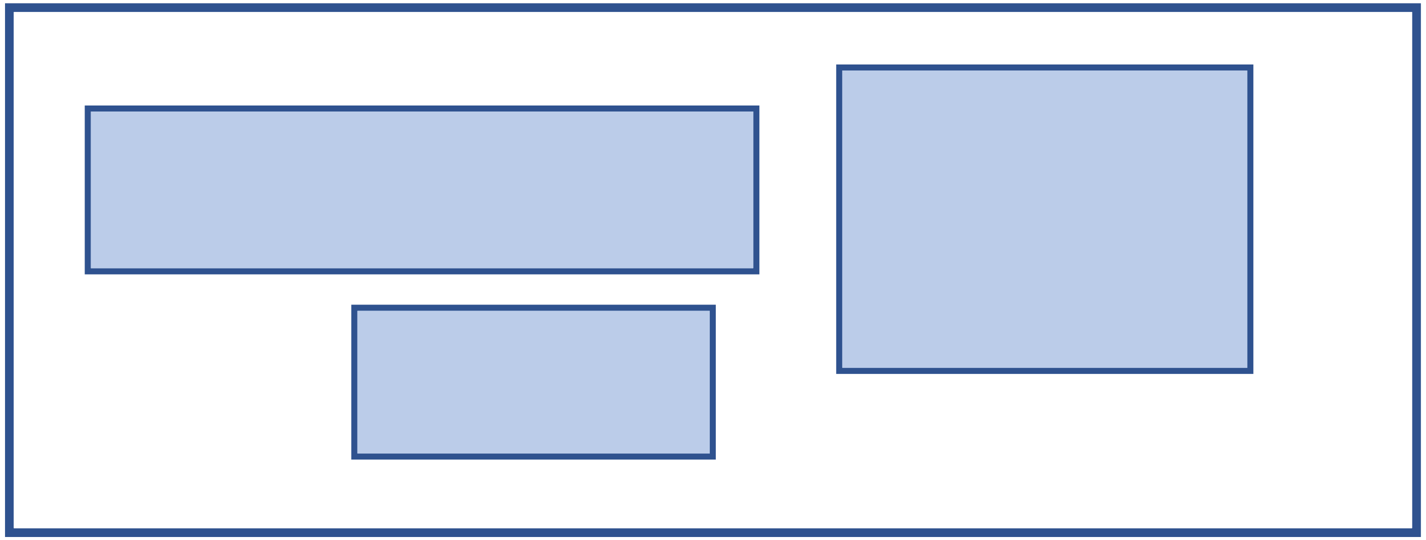
- Given: A set of n axis-parallel input rectangles.
- Goal: Find a set of non-overlapping rectangles of maximum cardinality.

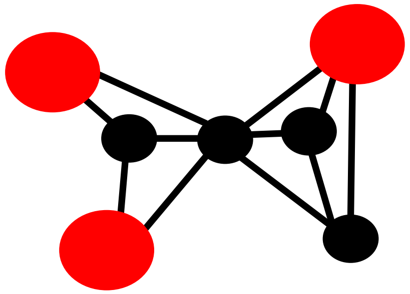




Maximum Independent Set of Rectangles (MISR)

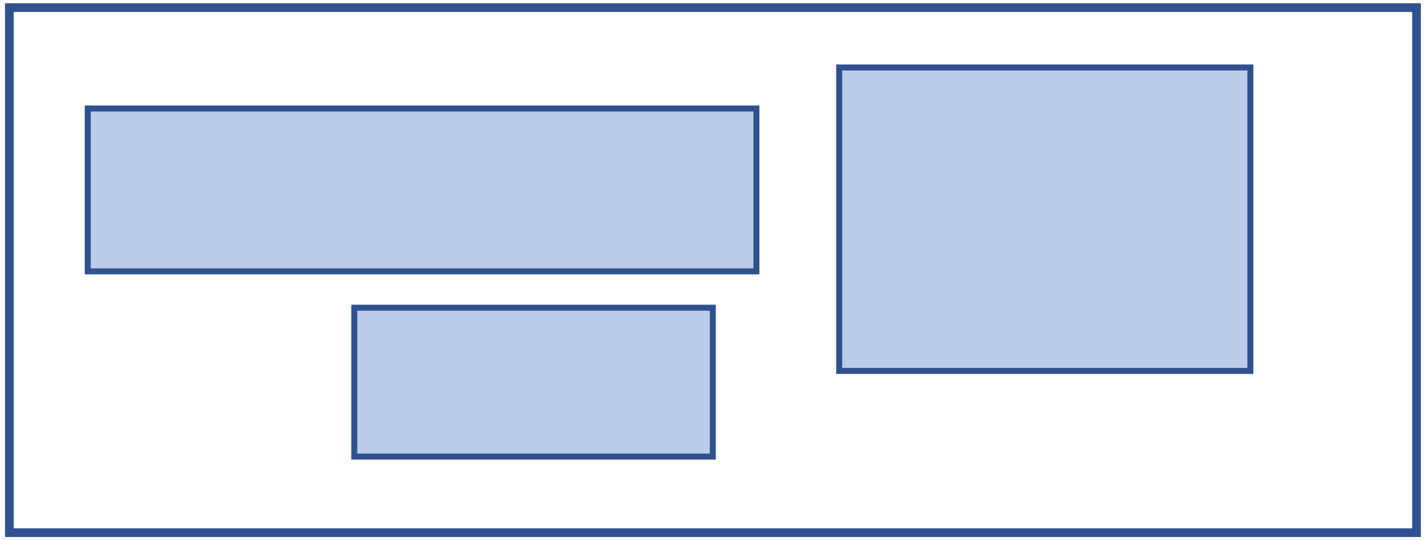
- Given: A set of n axis-parallel input rectangles.
- Goal: Find a set of non-overlapping rectangles of maximum cardinality.

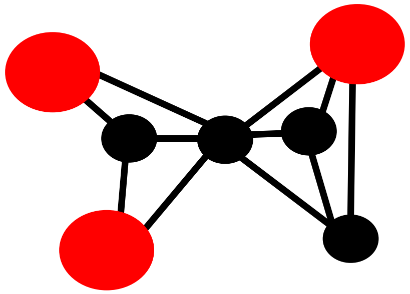




Maximum Independent Set of Rectangles (MISR)

- Given: A set of n axis-parallel input rectangles.
- Goal: Find a set of non-overlapping rectangles of maximum cardinality.





- Application: Map labelling, data mining, resource allocation.

MISR: Theoretical Importance

(Packing = axis-aligned nonoverlapping placement)	Bin-Packing Type	Knapsack Type
Rectangles movement	Pack all rectangles into minimum number of unit square bins	Pack maximum profit subset of rectangles into a unit square knapsack.
Vertically and Horizontally	**2-D Bin Packing** [1.405 BK'14] [No APTAS]	**2- Knapsack** [1.89 GGHKW'17][PTAS expected]
Vertically	(uniform) **round-SAP/round-UFP** [$2+\epsilon$ KKW'22] [No APTAS] (general) round-SAP/round-UFP [O(log log n) KKW'22] [No APTAS]	(uniform) **SAP** [1.969, MW'19] [PTAS expected] (general) $2+\epsilon$, MW'15] **UFP**: PTAS [GMW'22]
Not allowed	**Rectangle Coloring** [O(log ω) CW'21]	**MWISR** [O(log log n), CW'21]][PTAS expected]

MISR: Theoretical Importance

- Technique developed for MISR had found usage in many other intersecting packing/covering problems in approximation algorithms, combinatorial optimization, and computational geometry such as: 2D bin packing, 2D knapsack, strip packing, unsplittable flow on a path (UFP), storage allocation problem (SAP), round-UFP, round-SAP, rectangle coloring, geometric set cover, ...
- Deep connections with structural graph theory (chromatic number and clique number of χ-bounded graphs) and discrete/combinatorial geometry (Pach-Tardos Conjecture).

MISR: Tale of Approximability

- $O(\log n)$ [Khanna et al, SODA'98, Nielson TCS'00, K.,Reddy APPROX'20…]
- $\lceil \log_k n \rceil$ [Berman et al., SODA'01]
- $4q$ (q is the max clique size) [LNO, APPROX'02]
- W[1]-hard [Marx, ESA'05] even for squares.
- $O(\log \log n)$ [Chalermsook-Chuzhoy, SODA'09]
- QPTAS: $(1+\epsilon)$-approx. in $n^{poly(\log n/\epsilon)}$ time [Adamaszek-Wiese, FOCS'13]
- $(1+\epsilon)$-approx. in $n^{\mathrm{poly}(\log\log \mathrm{n}\,/\epsilon)}$ time [Chuzhoy-Ene FOCS'16]
- PTAS is expected but even O(1)-approximation was not known.

MISR: Tale of Approximability

- Mitchell [FOCS'21]: 10-approximation.
- Analysis is based on exhaustive case analysis, with sixty cases in total!
- Can we improved approximation algorithms, better runtime, and simpler analysis?

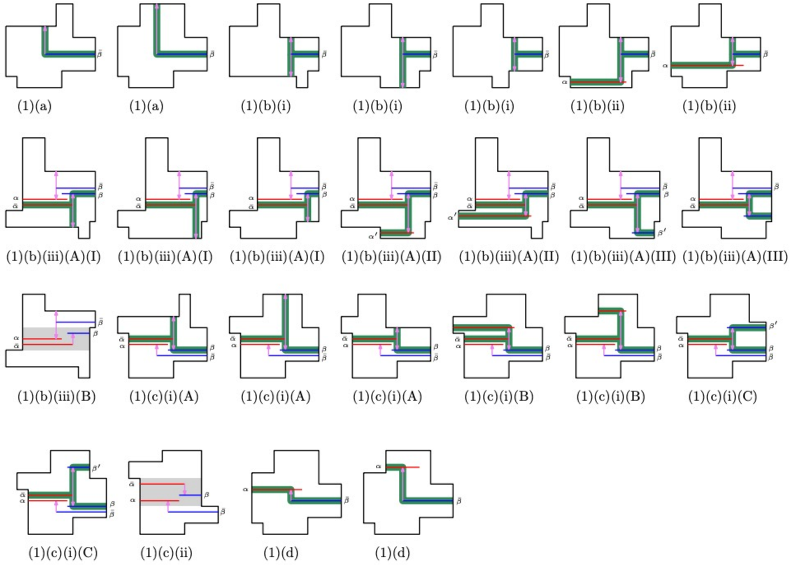


Figure 13: Case (1), and its subcases.

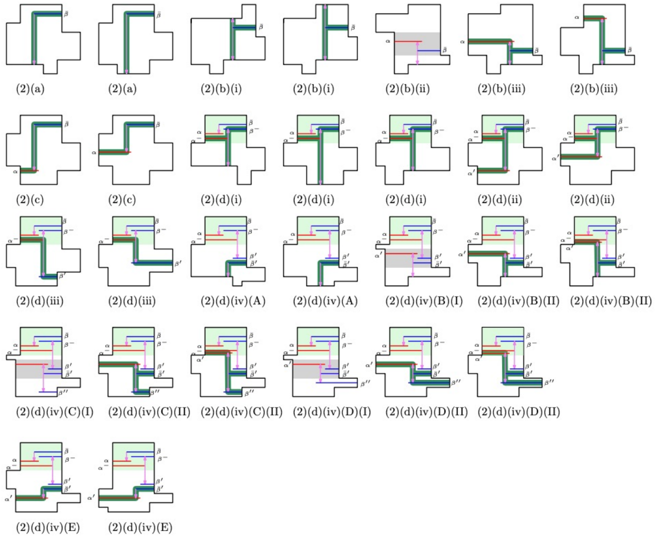


Figure 14: Case (2), and its subcases.

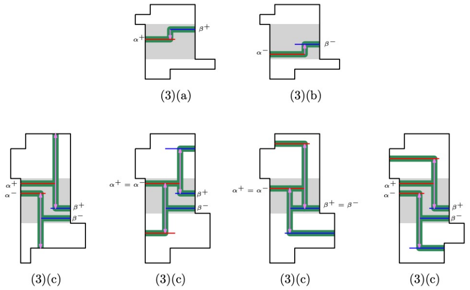


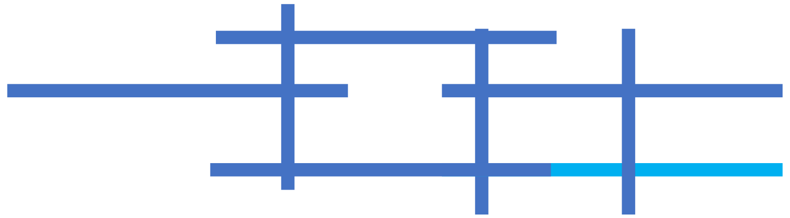
Figure 15: Case (3), and its subcases.

MISR: Our result

A polynomial time $(2+\epsilon)$-approximation.
[3-approx. in SODA’22]

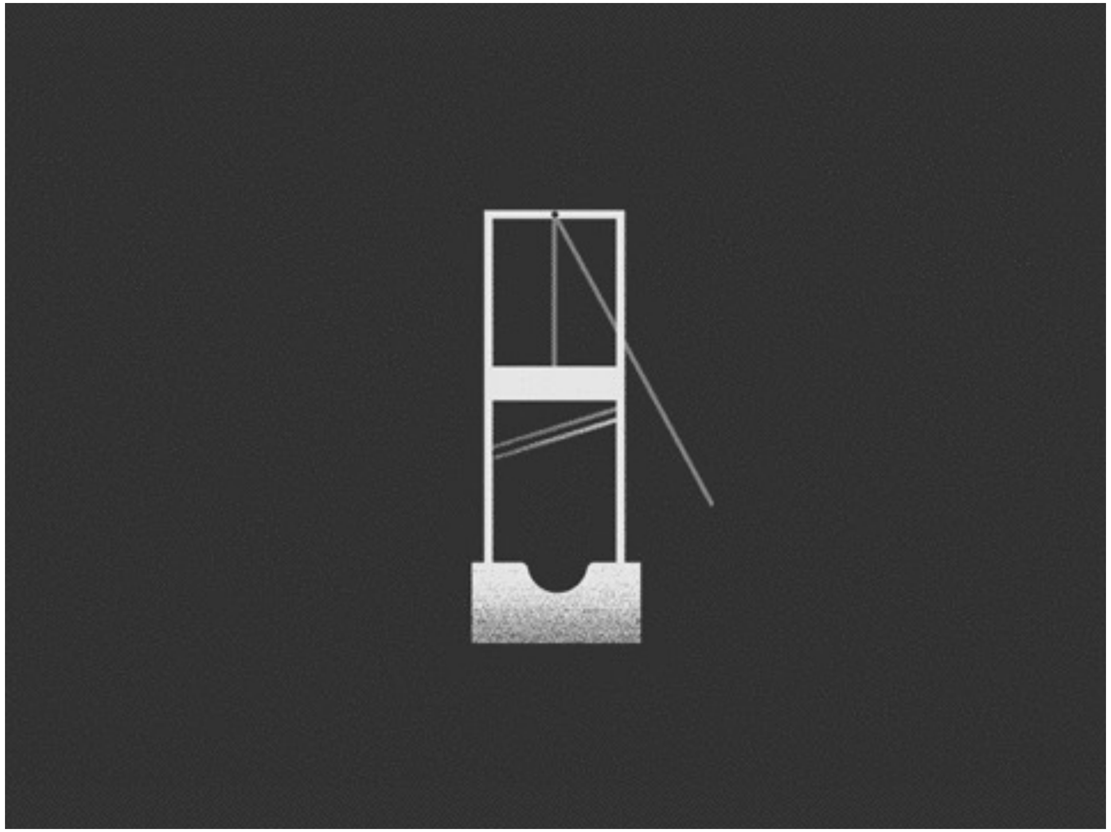
- Step 1. Show existence of a “good” structured solution:
  - Set of candidate solutions: C. -- size can be exponential.
  - Set of structured solution: S. -- need to make the size to be polynomial.
  - S is a good approximation of C: For any candidate solution $I \in C$ there is a structured solution $I' \subseteq I$ and $I' \in S$ such that $\alpha|I'| \geq |I|$.
- Step 2. DP-based algorithm that finds the best structured solution.

- Note: 2-approx. is best known even for axis-parallel line segments.



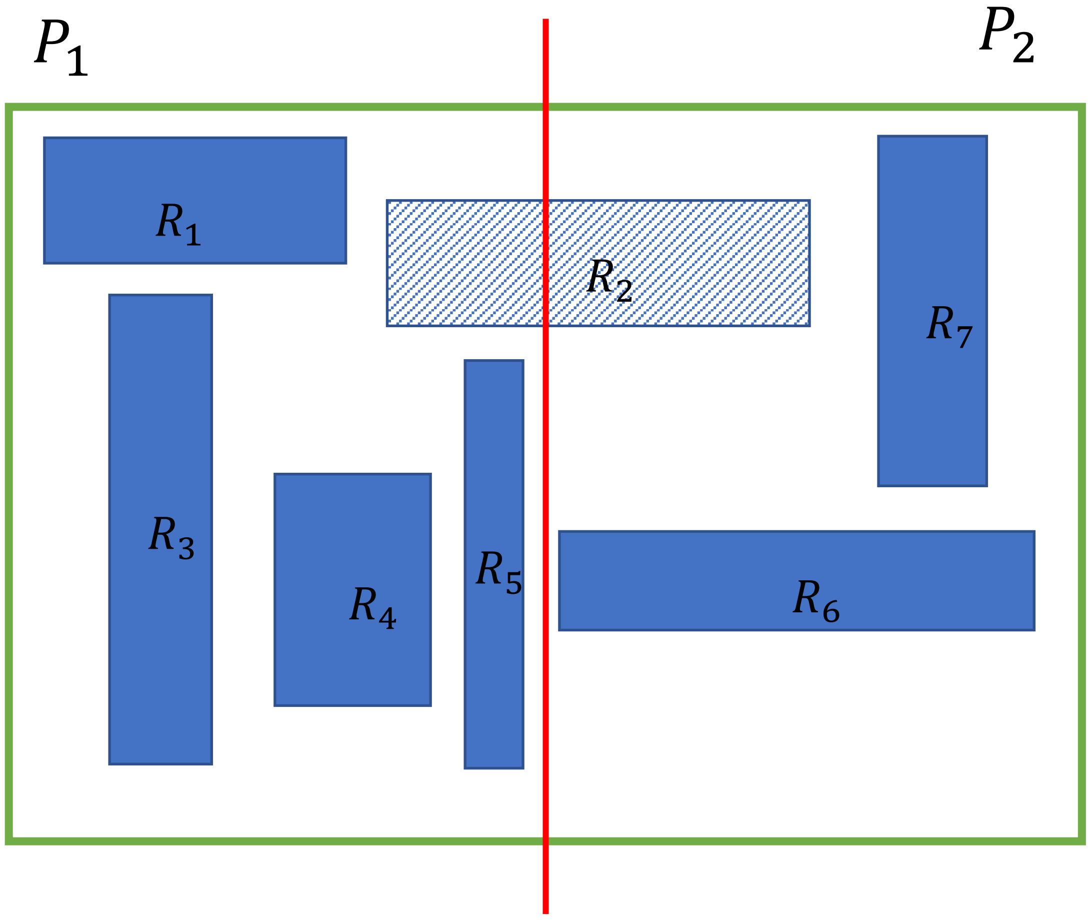
What is a structured solution?

Guillotine Cuts



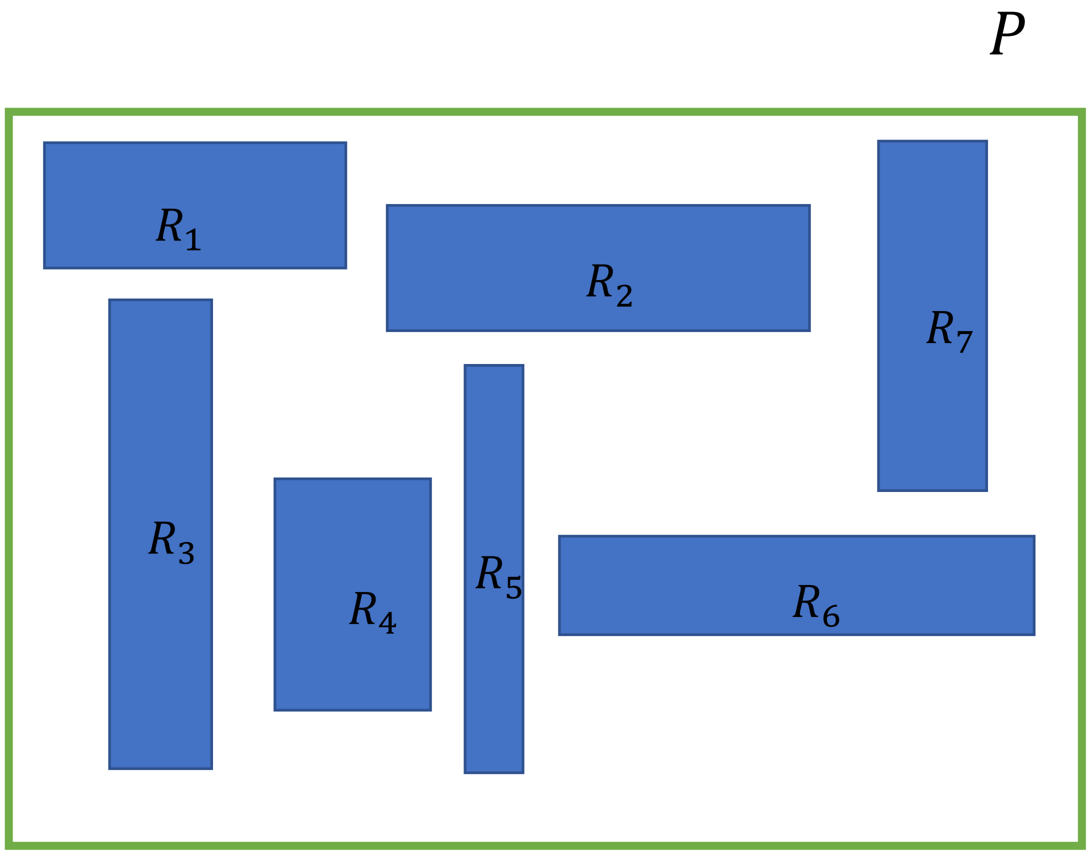
Guillotine Cuts

- Guillotine cuts: An end-to-end cut along a straight line to divide a (rectangular) piece into two smaller pieces.
- Practically very relevant.
- Cutting stock: Cut out some required geometric objects under some constraints, from a large source material such as glass, rubber, metal, wood or cloth.
- lower cost (minimal operation by machines) and simple usability (simple to program using column generation).



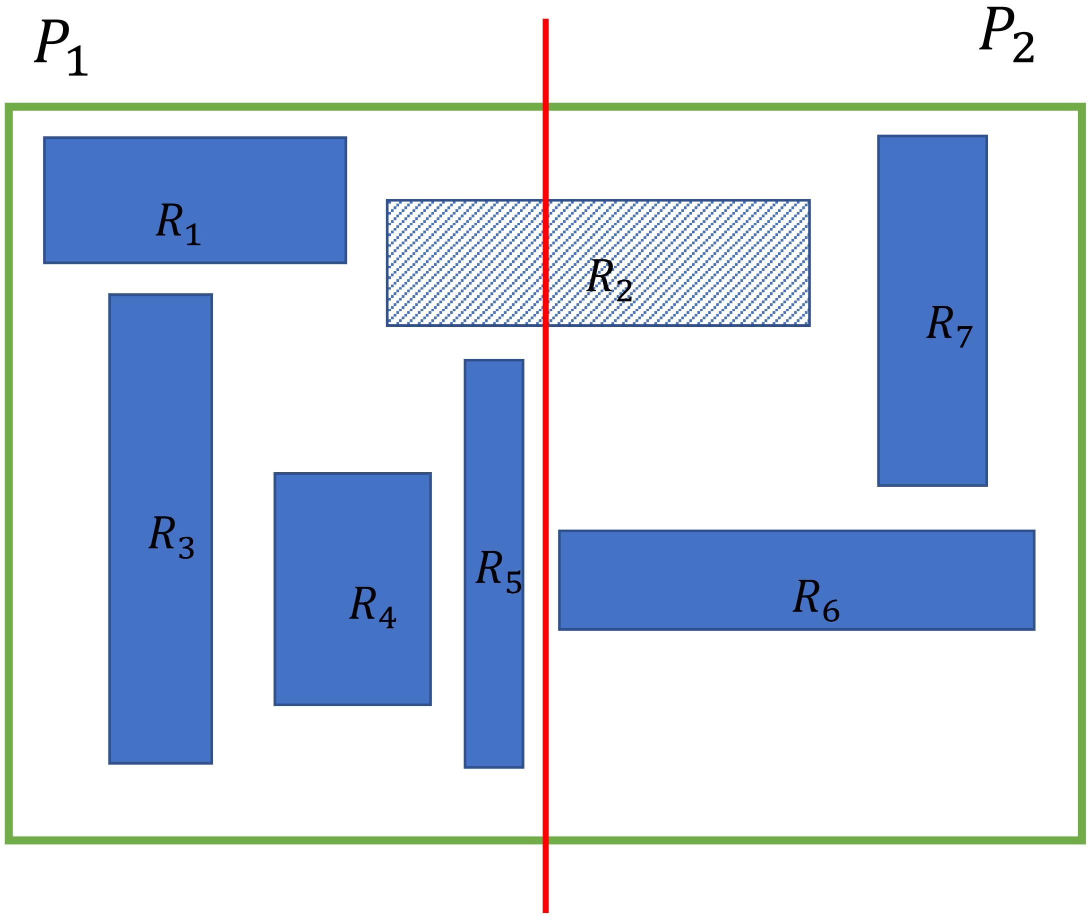
Guillotine Cuts

- Guillotine Cutting Sequence:
  - A series of guillotine cuts , each cut separating a sub-piece into two new sub-pieces.



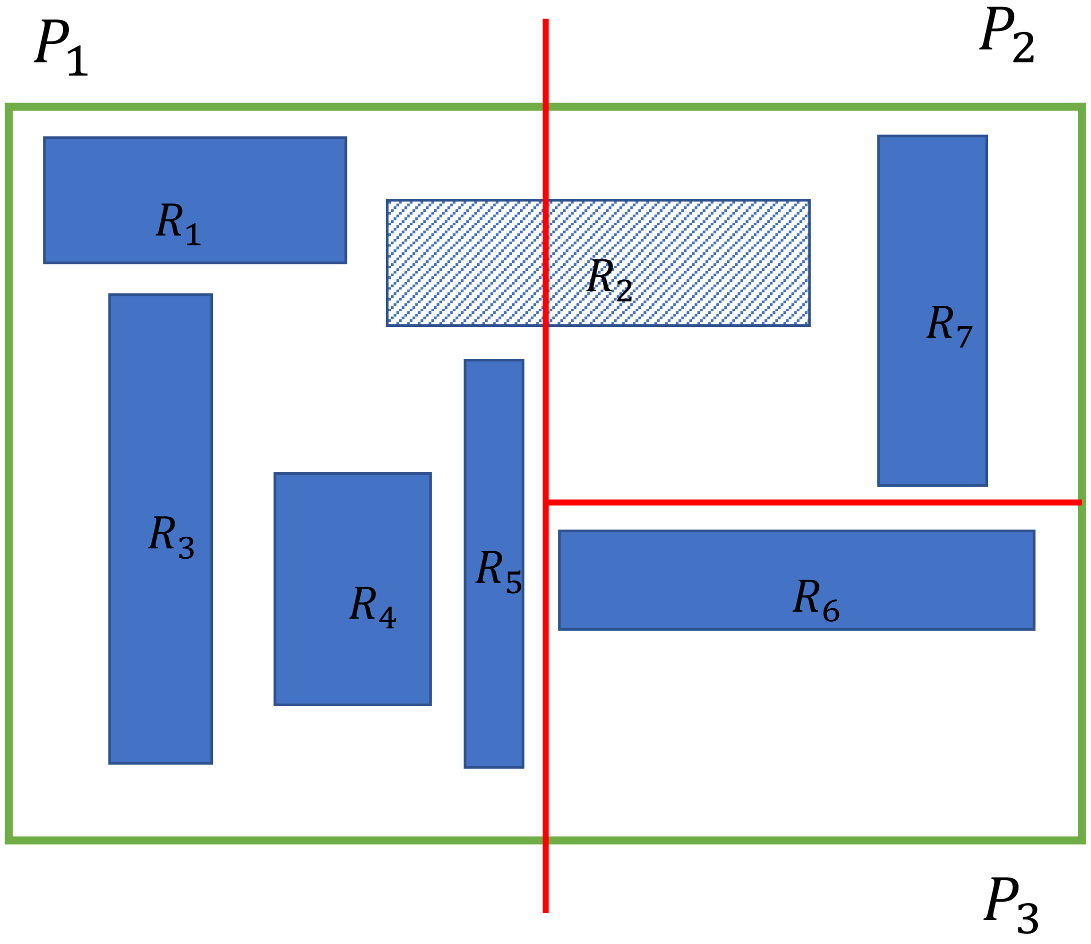
Guillotine Cuts

- Guillotine Cutting Sequence:
  - A series of guillotine cuts , each cut separating a sub-piece into two new sub-pieces.



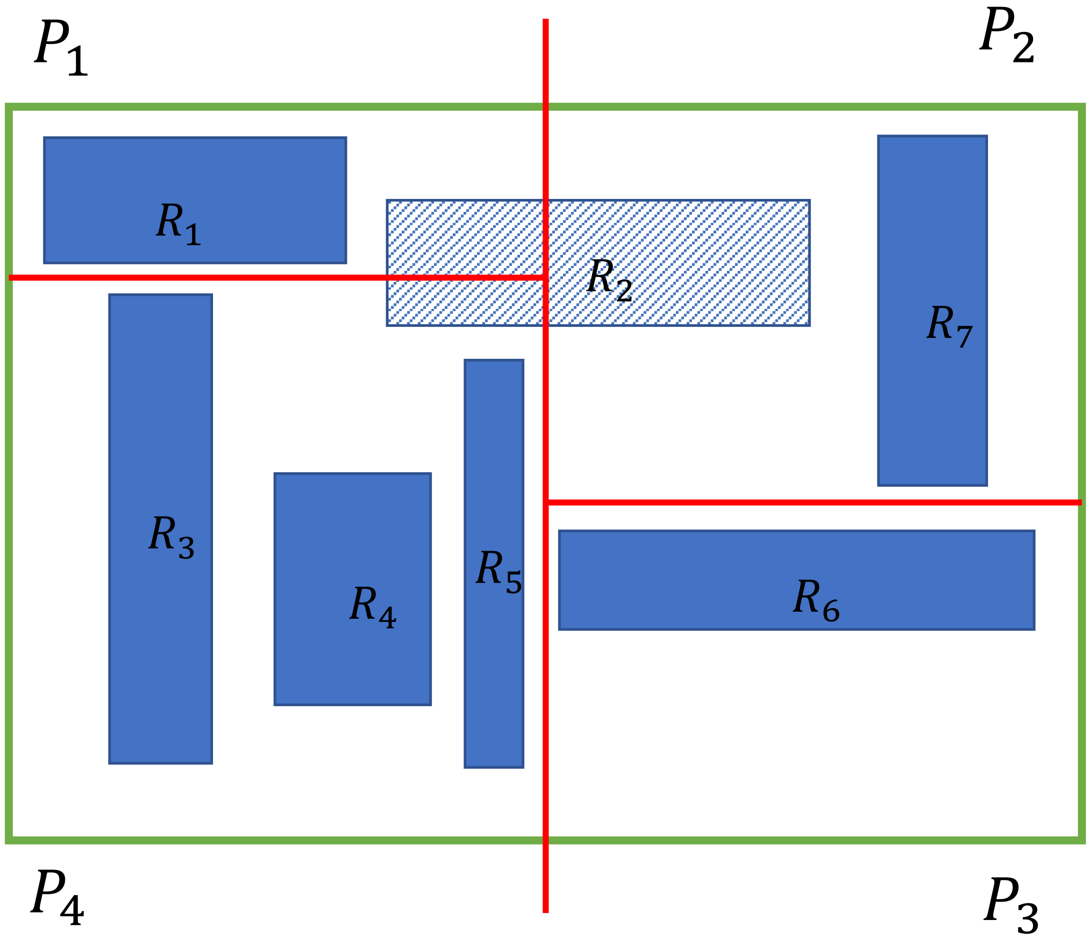
Guillotine Cuts

- Guillotine Cutting Sequence:
  - A series of guillotine cuts , each cut separating a sub-piece into two new sub-pieces.



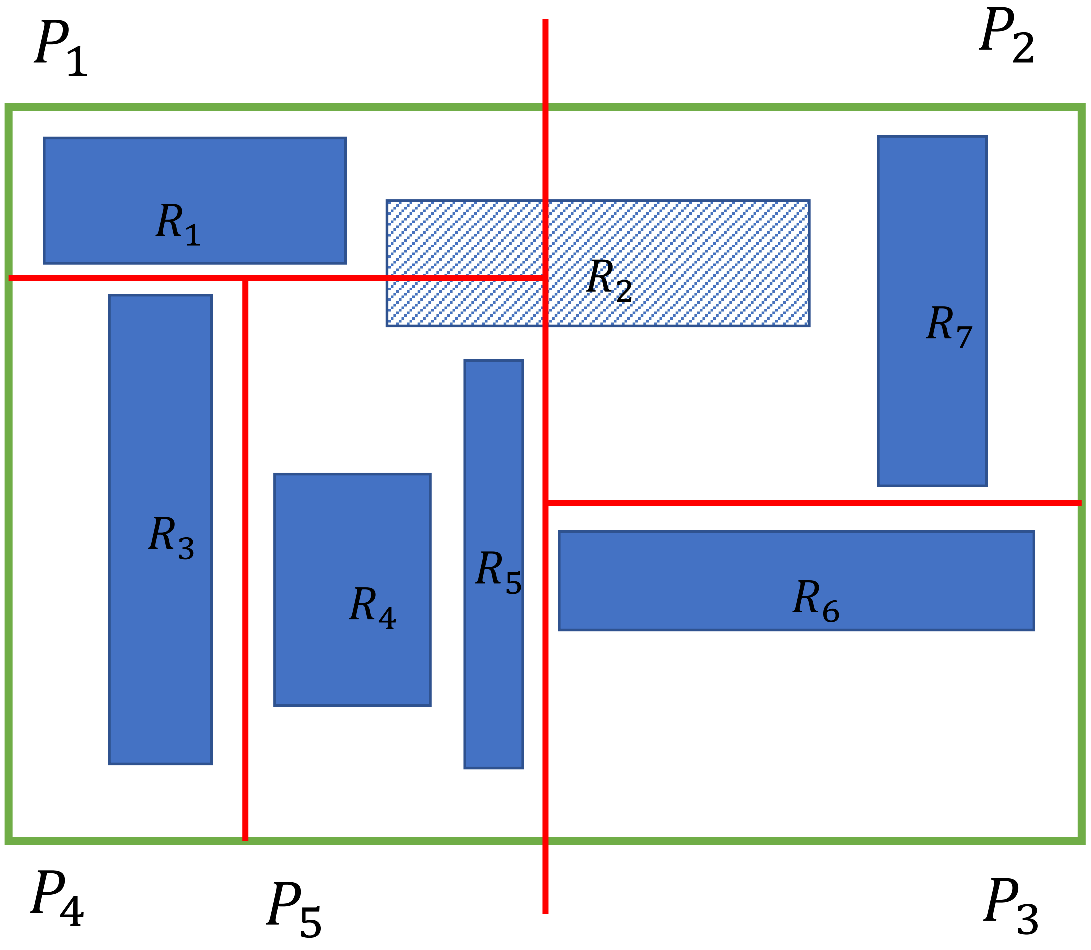
Guillotine Cuts

- Guillotine Cutting Sequence:
    - A series of guillotine cuts , each cut separating a sub-piece into two new sub-pieces.



Guillotine Cuts

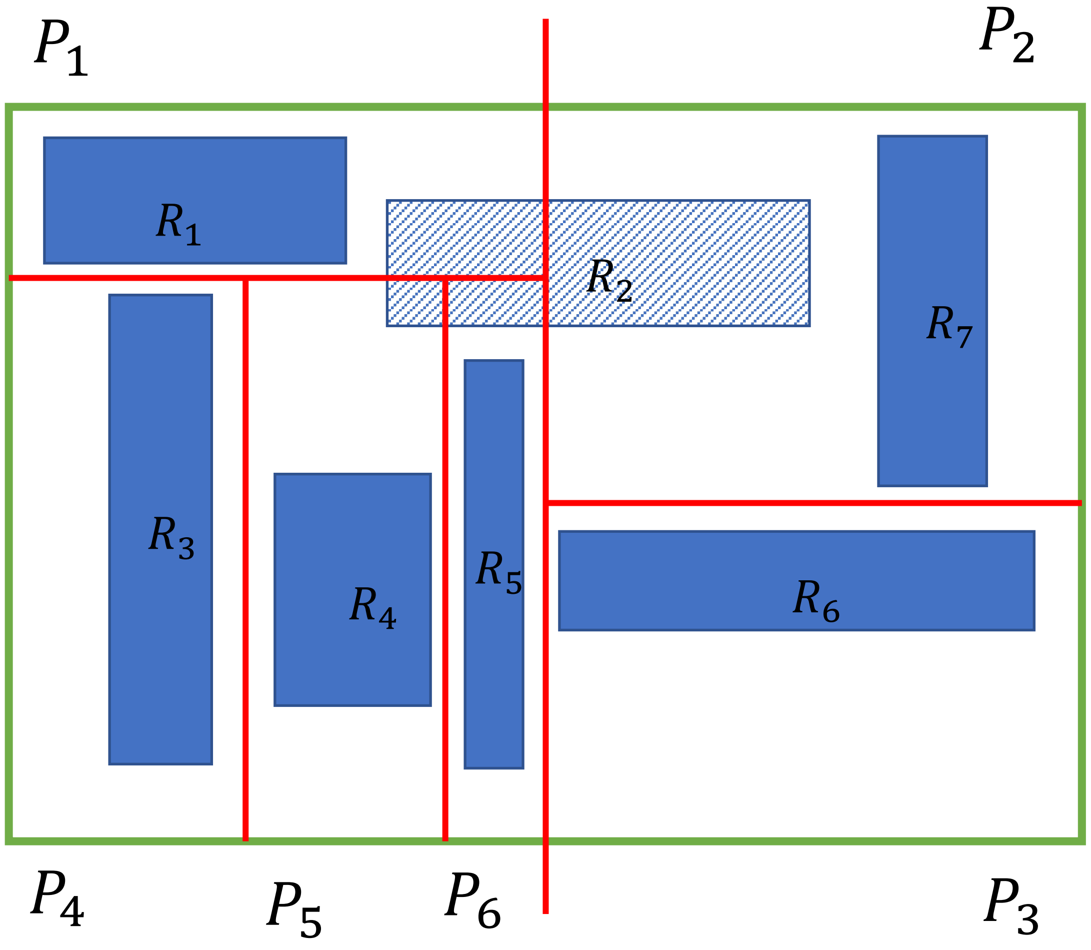
- Guillotine Cutting Sequence:
  - A series of guillotine cuts , each cut separating a sub-piece into two new sub-pieces.



Guillotine Cuts

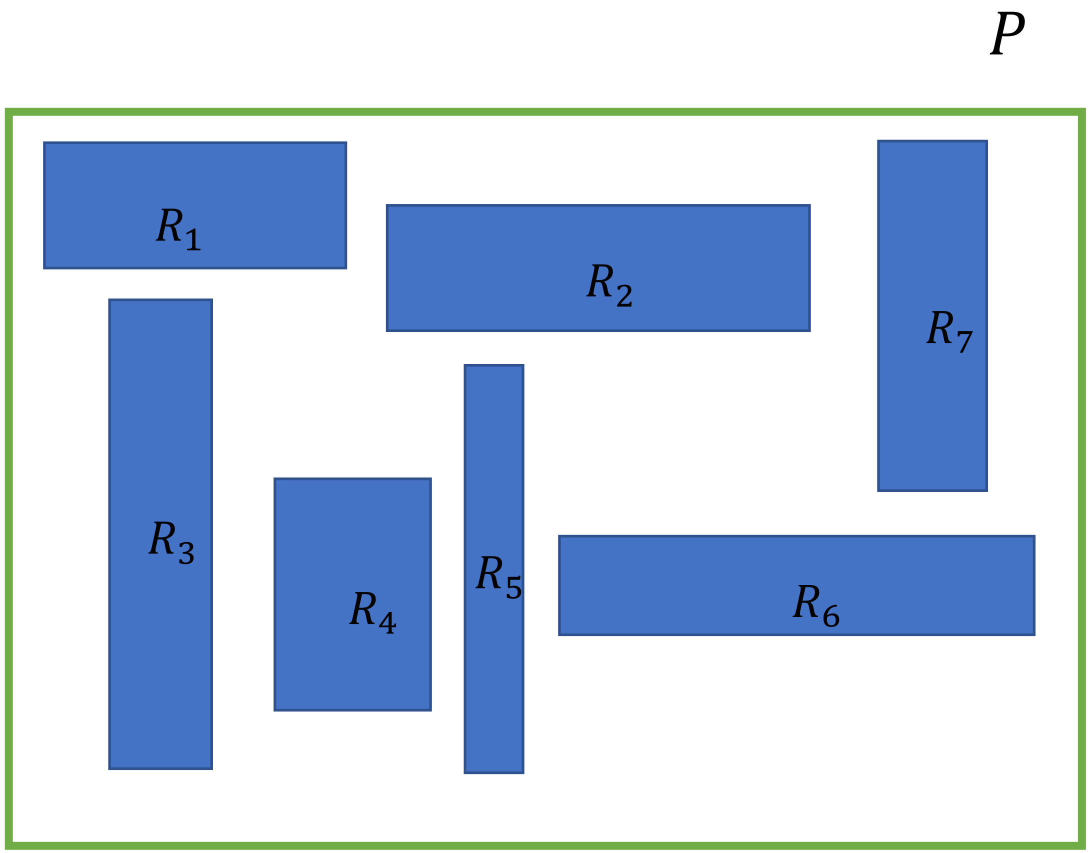
- Guillotine Cutting Sequence:
  - A series of guillotine cuts , each cut separating a sub-piece into two new sub-pieces.

- Guillotine cuts has connections with other packing problem
  such as bin packing [BLS FOCS'05], 2D Knapsack [KMSW SoCG'21], 2D Strip Packing [KLMS'22]…



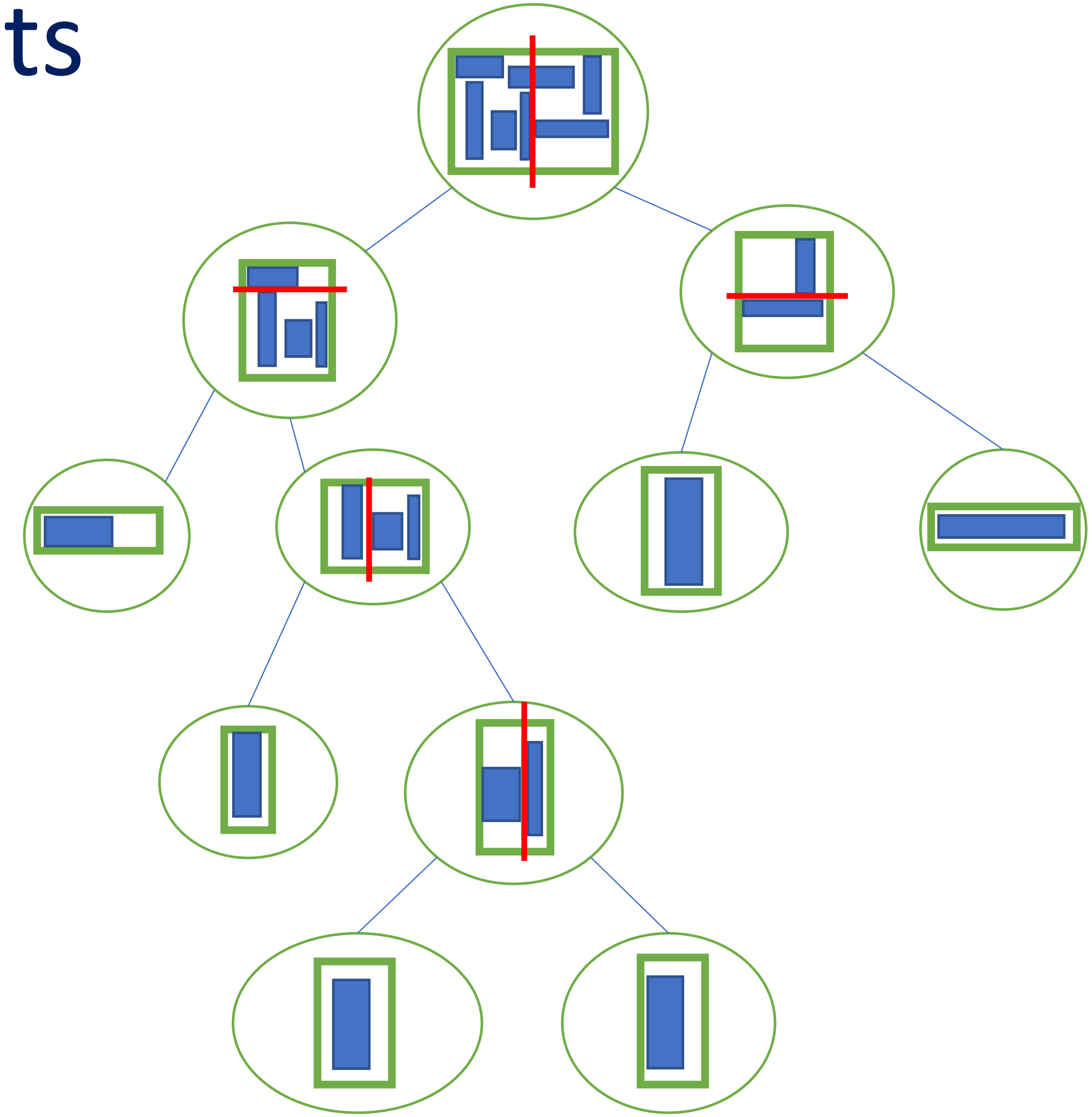
Guillotine Cuts

- A cutting sequence can be naturally imagined as a binary tree:
- Each node corresponds to a rectangular region.
- Each nonleaf node (corrs. to region P) contains two children (corrs. to P_1, P_2 obtained by guillotine cut from P).
- Each leaf node contains exactly one item.



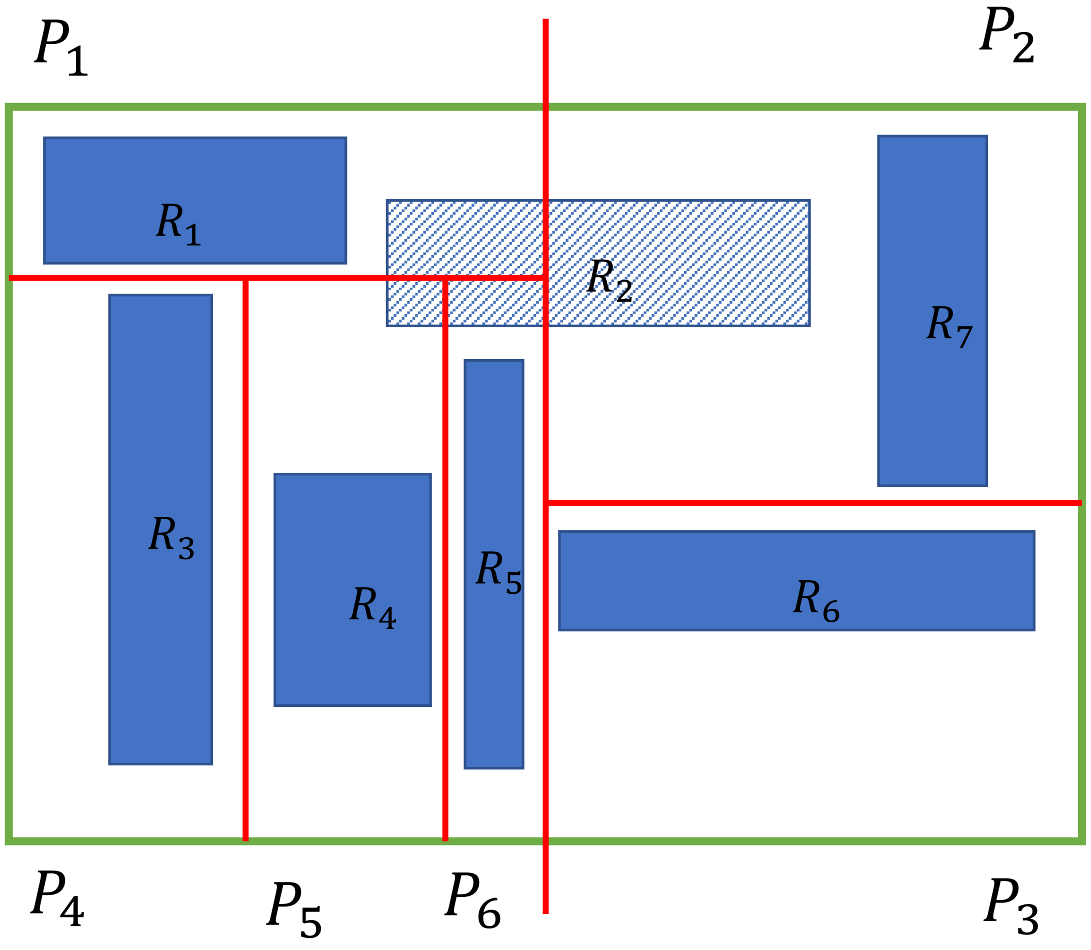
Guillotine Cuts

- A cutting sequence can be naturally imagined as a binary tree:
- Each node corresponds to a rectangular region.
- Each nonleaf node (corrs. to region P) contains two children (corrs. to P_1, P_2 obtained by a guillotine cut in P).
- Each leaf node contains exactly one item.



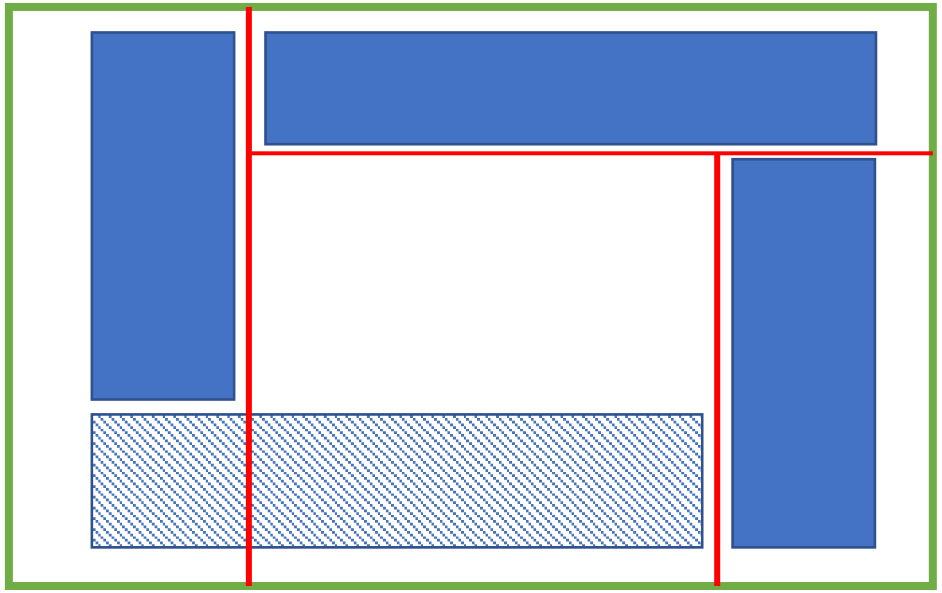
Guillotine Cuts

- A rectangle is extracted if it is not killed and is the only rectangle in its sub-piece.
- All rectangles except R_2 are extracted in this example.
- Given configuration is guillotine separable if all rectangles can be extracted using some cutting sequence.



Guillotine Cuts

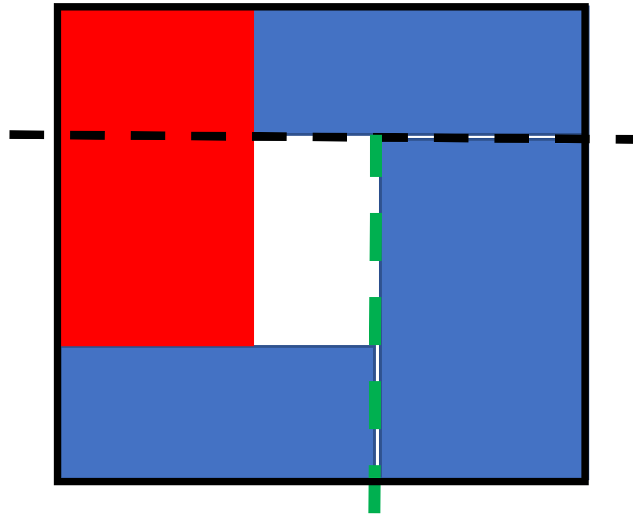
- Is it always possible to separate out all rectangles using a sequence of guillotine cuts?
- NO!



Not Guillotine separable

Guillotine separability of rectangles

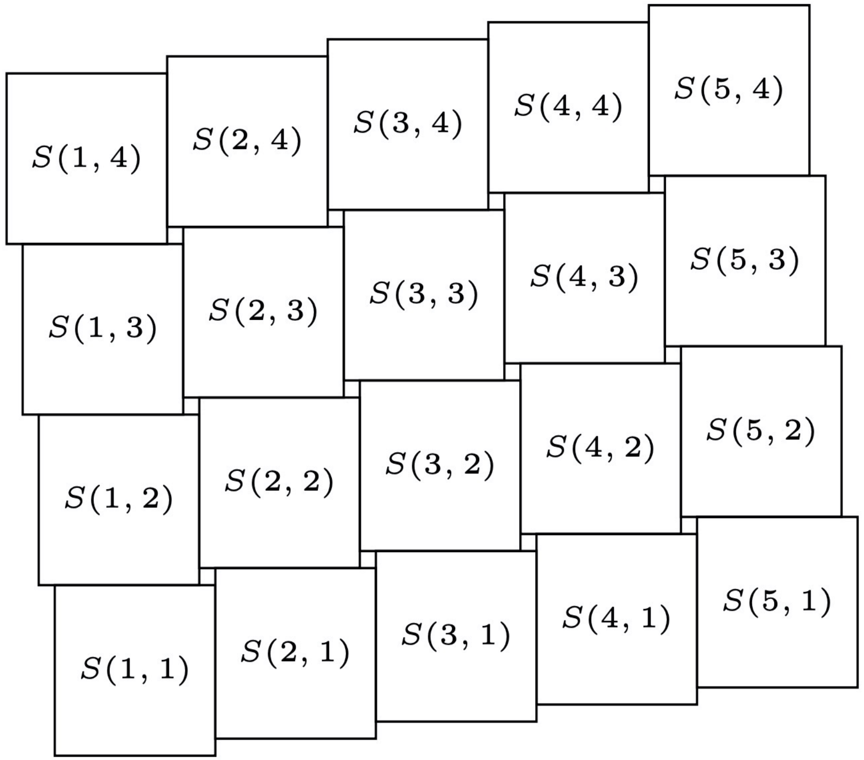
- Goal: Separate out **a constant fraction** of rectangles using a sequence of guillotine cuts?
- Pach-Tardos [SoCG'00] conjectured it to be true.
- BIG OPEN QUESTION in computational geometry, operations research and combinatorics.



Guillotine separability of rectangles

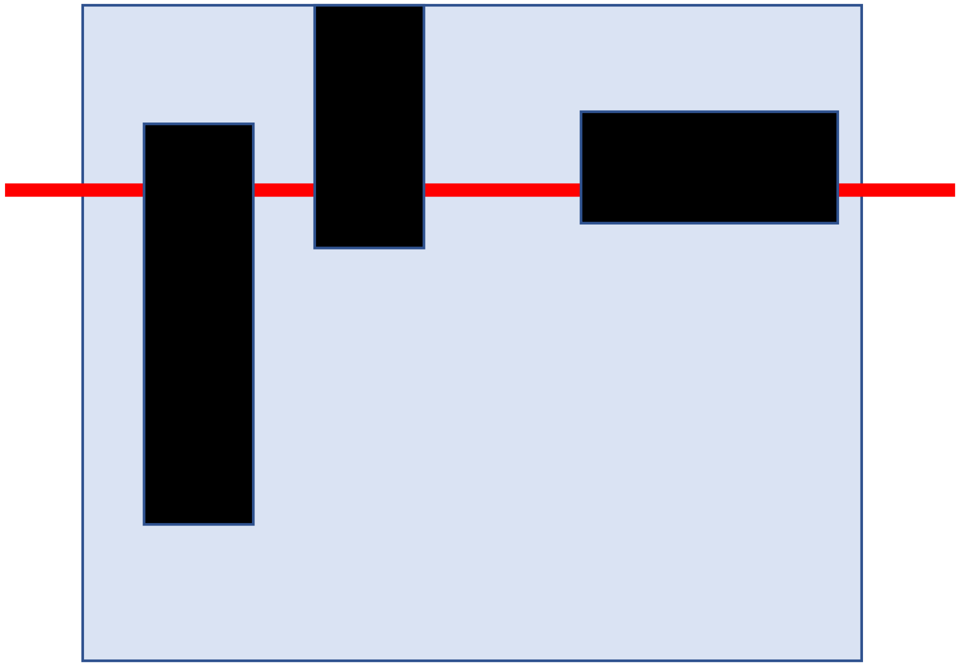
- Hardness: [Abed et al., APPROX'15]
  Given n rectangles, there are instance where we can not separate out > n/2 rectangles using a sequence of guillotine cuts.

- Algorithm: [K., Reddy, APPROX'20]
  We can separate out > n/(log n+1) rectangles using a sequence of guillotine cuts.



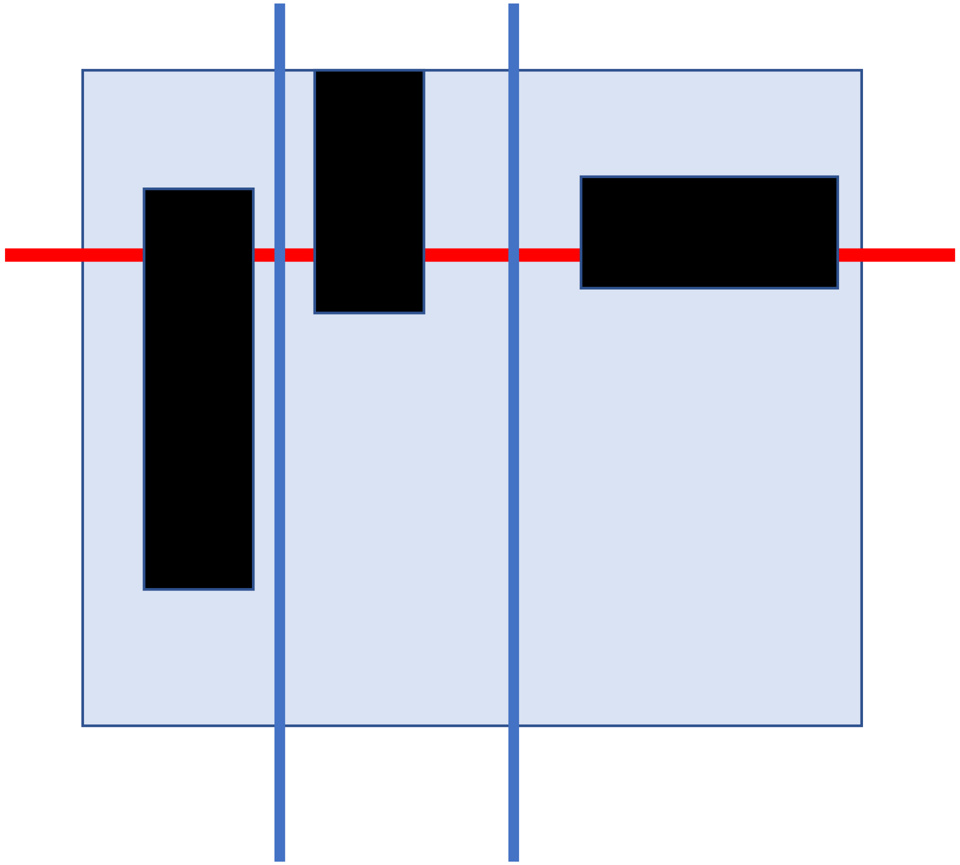
Extraction of $n/(\log n + 1)$ rectangles

- Observation 1: If all rectangles intersect a straight line then they are guillotine separable.



Extraction of $n/(\log n + 1)$ rectangles

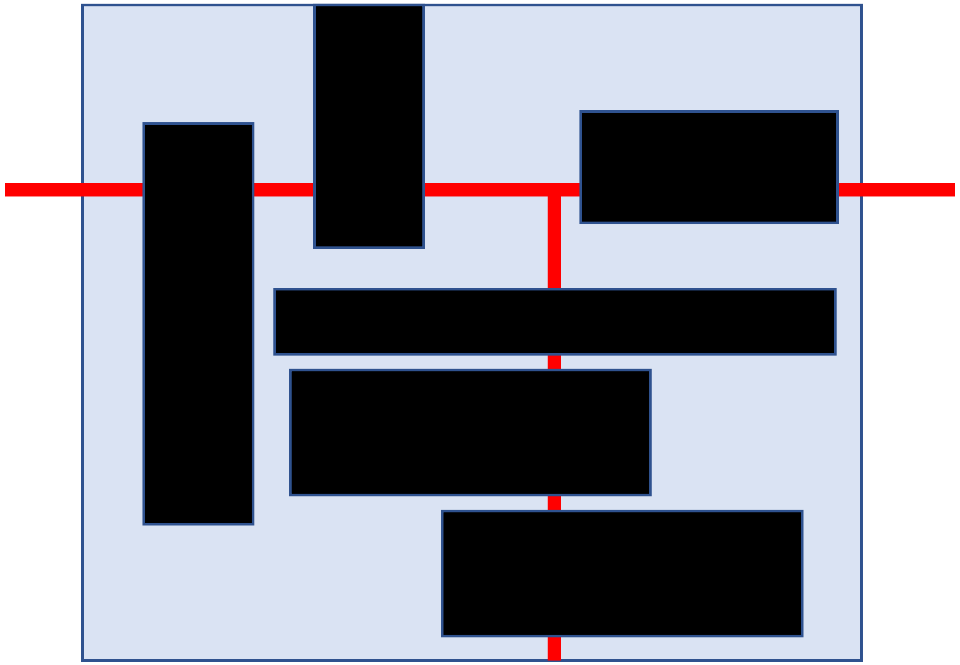
- Observation 1: If all rectangles intersect a straight line then they are guillotine separable.



Extraction of $n/(\log n + 1)$ rectangles

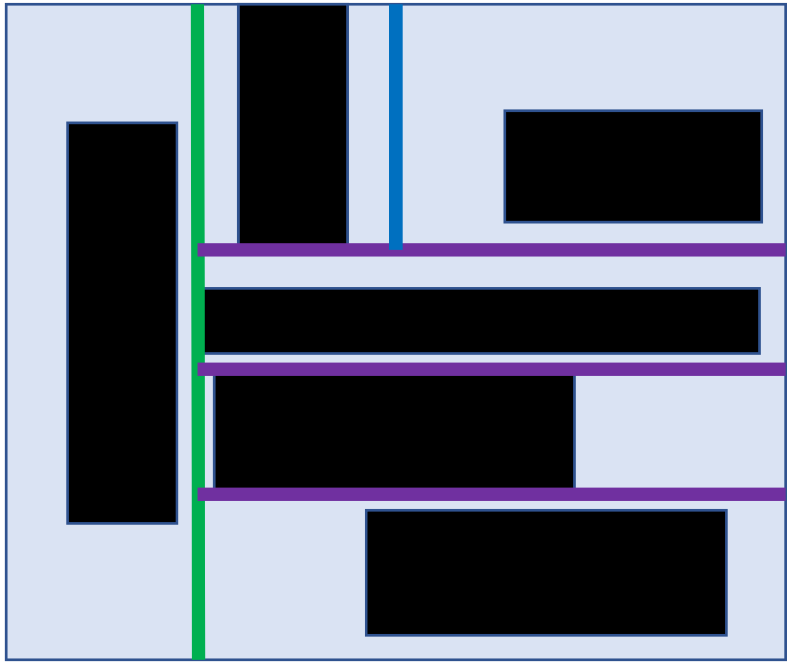
- Observation 1: If all rectangles intersect a straight line then they are guillotine separable.

- Observation 2: If all rectangles intersect a "T" then they are guillotine separable.



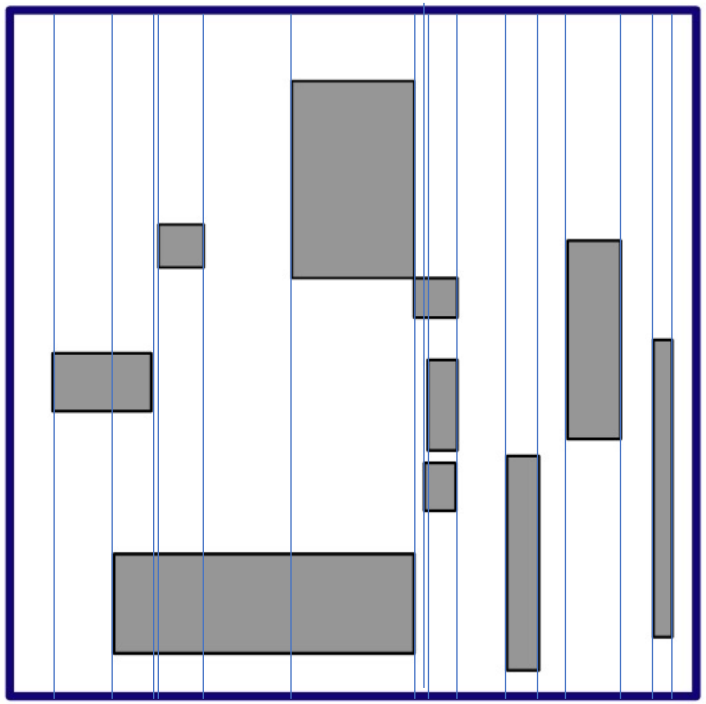
Extraction of $n/(\log n + 1)$ rectangles

- Observation 1: If all rectangles intersect a straight line then they are guillotine separable.
- Observation 2: If all rectangles intersect a "T" then they are guillotine separable.



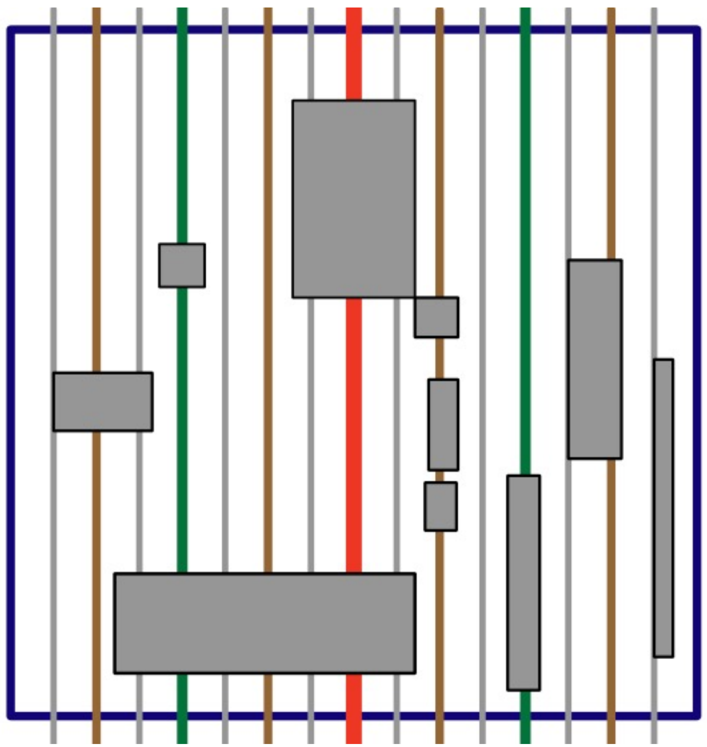
Extraction of $n/(\log n + 1)$ rectangles

- Assume: Rectangles are embedded in $[0,2n]\times[0,2n]$ grid with integral corners.

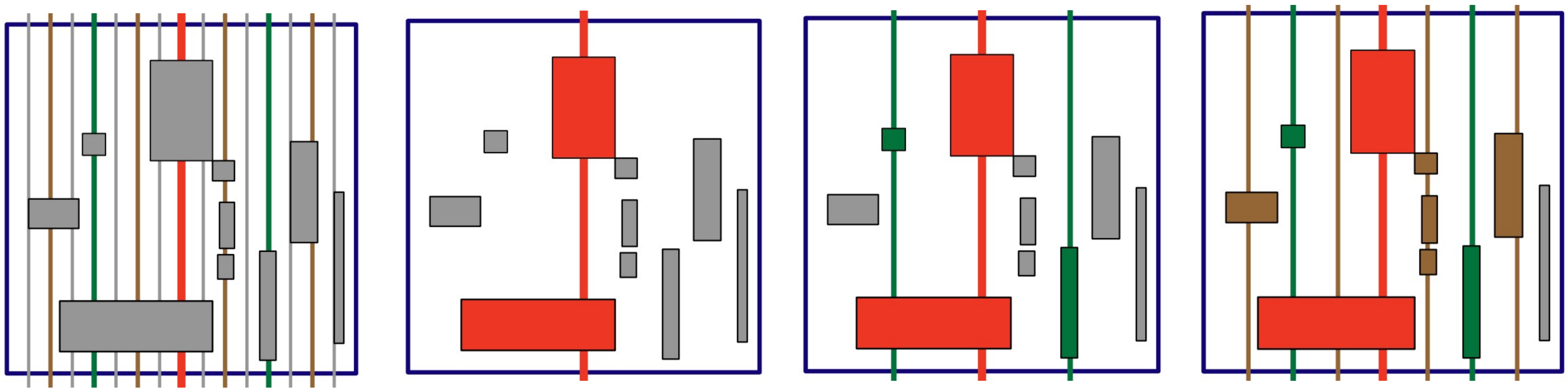


Extraction of $n/(\log n + 1)$ rectangles

- Assume: Rectangles are embedded in $[0,2n]\times[0,2n]$ grid with integral corners.
- Define pole lines: level 1 at n, level 2 at $n/2, 3n/2$, level 3 at $n/4, 3n/4, 5n/4, 7n/4$ ….
- There are $(\log n + 1)$ levels.
- The union of all poles of level 1 to i, divides the plane into 2^i equal partitions.
- Level of a rectangle is the smallest level i such that some level-i pole intersects the rectangle.

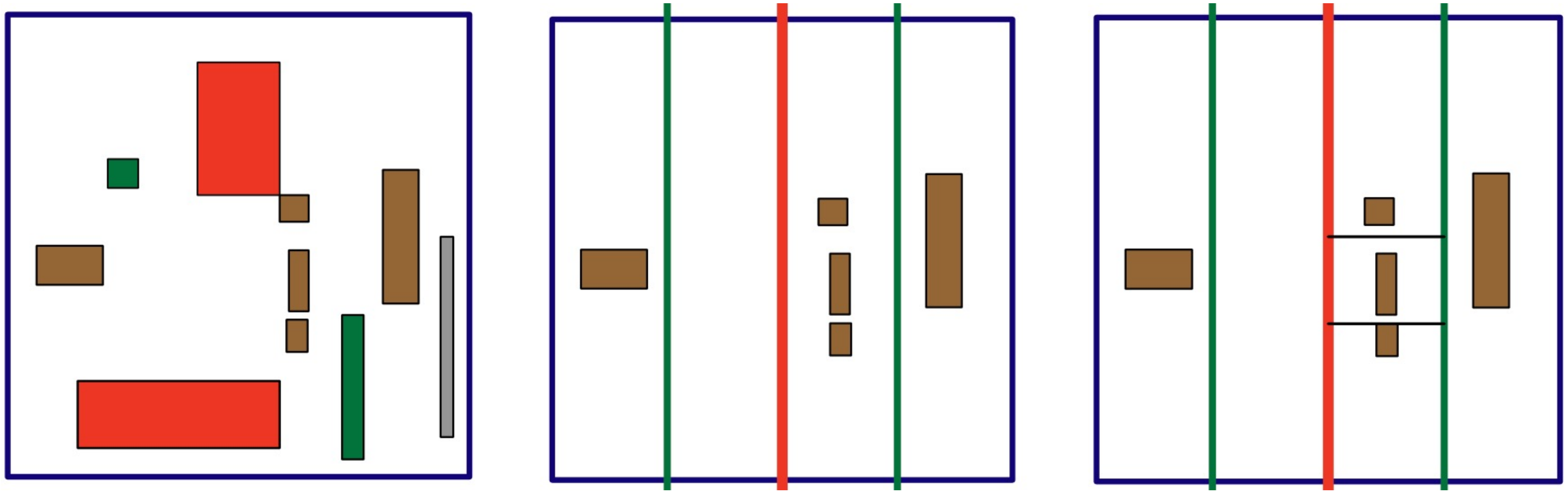


Extraction of $n/(\log n + 1)$ rectangles



- Pole lines: level 1 at $\frac{n}{2}$, level 2 at $\frac{n}{4}, \frac{3n}{4}$, level 3 at $\frac{n}{8}, \frac{3n}{8}, \frac{5n}{8}, \frac{7n}{8}$ ….
- Level of a rectangle is the smallest level i such that some level-i pole intersects the rectangle.

Extraction of $n/(\log n+1)$ rectangles



- We get a partition into $(\log n+1)$ color classes.
- Each color class is 2-stage guillotine separable.
- Take color of maximum cardinality.
- This gives guillotine separable $\geq \frac{n}{1+\log n}$ rectangles

Further improvement and Hardness.

- Using a similar decomposition using T-cuts (existentially), we can improve extraction factor from $n/\log_2(n+1)$ to $n/\log_3(n+2)$.
- Question:
  Can one obtain $n/(\log n)^{(1-\varepsilon)}$ using constant number of stages?

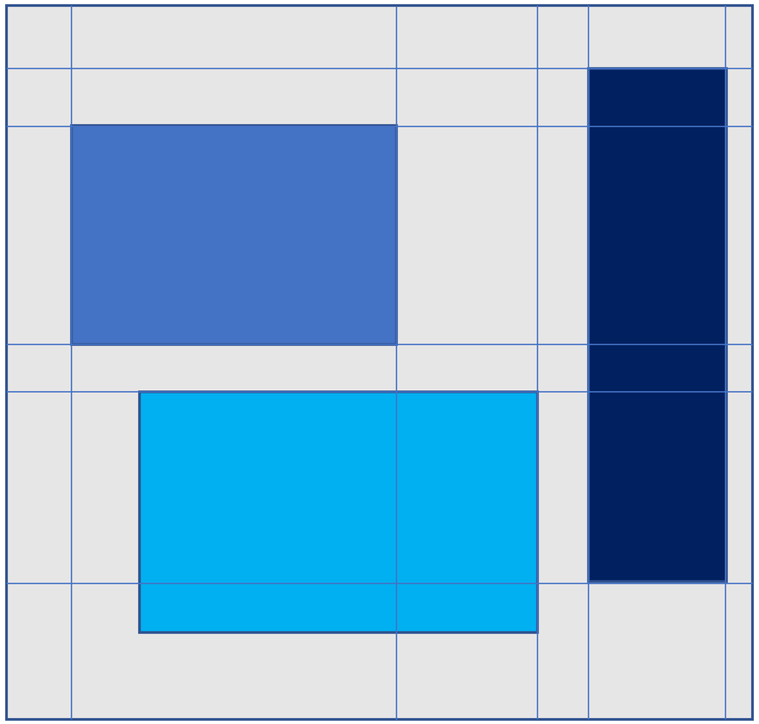
- Answer: No.
- In fact, to extract $\Omega(n)$ rectangles we need at least $\log n/\log\log n$ stages (and $\log\log n$ stages for the unweighted case).

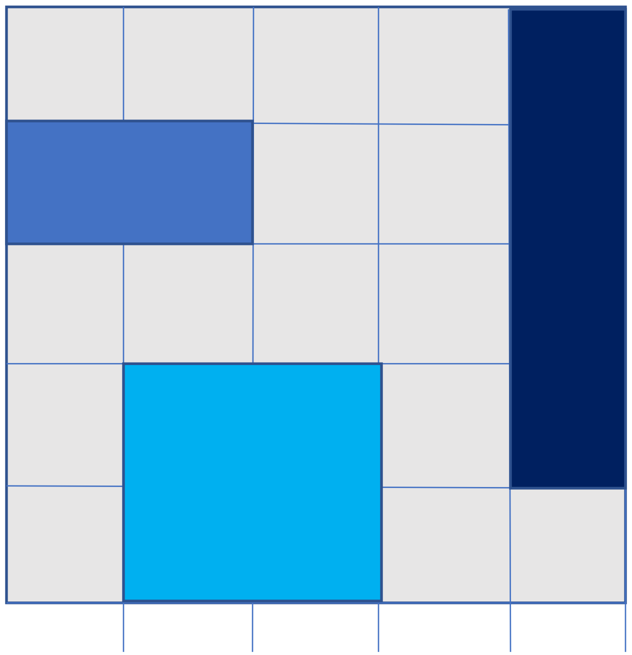
Connecting guillotine & MISR

- Assume an existential property (P1): For any embedding of n nonoverlapping rectangles, there are n/α fraction of rectangles separable by guillotine cuts.
- Assume optimal MISR solution is OPT.
- The rectangles in OPT are nonoverlapping, but may not be guillotine separable.
- From P1: $\frac{|OPT|}{\alpha}$ rectangles that are guillotine separable.
- We will show a Dynamic Program (DP) that returns optimal guillotine separable set for a MISR instance in $O(n^5)$ time.
- DP returns guillotine separable set R', $|R'| \geq \frac{|OPT|}{\alpha}$.
- Guillotine separable rectangles are structured and gives α-approximation for MISR. (Pach-Tardos Conjecture: $\alpha = 2$).

Processing before DP

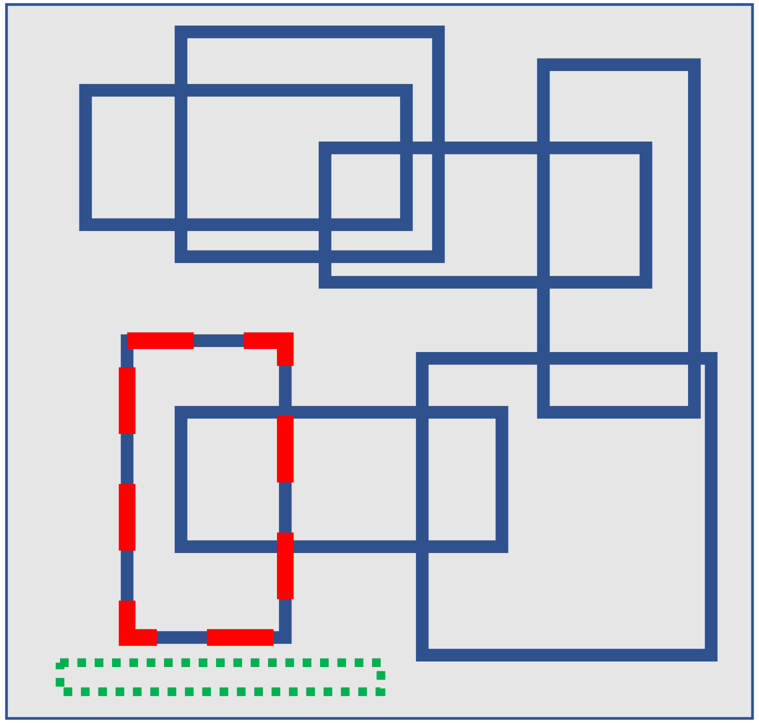
- All rectangle corners have integral coordinates in $[0,2n-1]\times[0,2n-1]$.





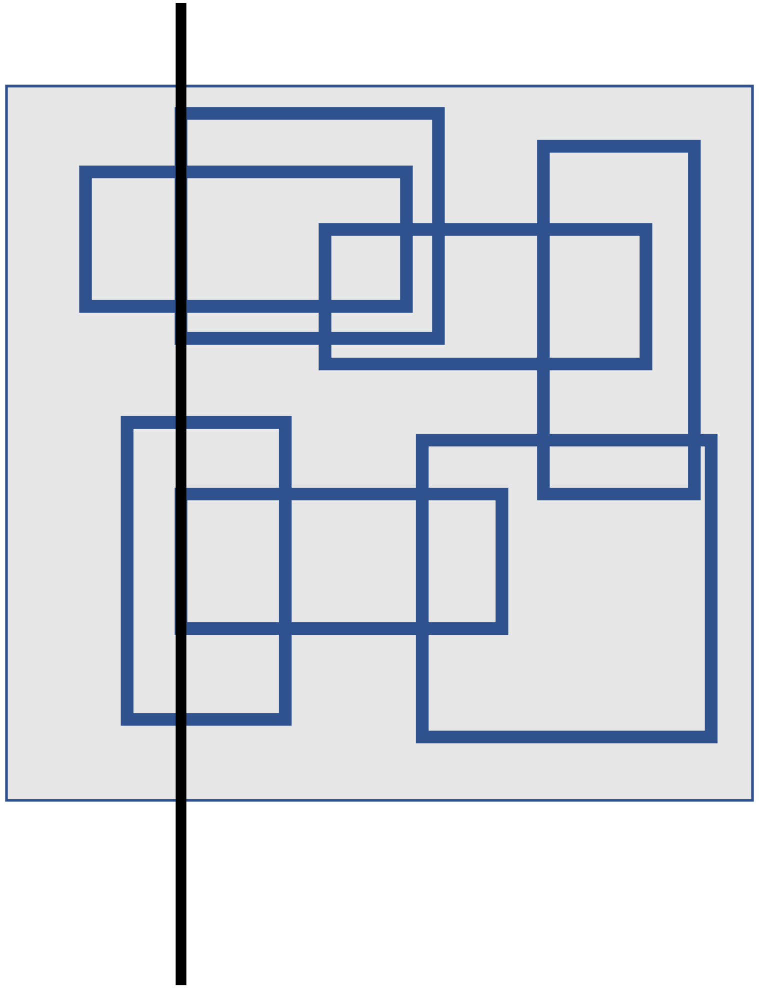
DP for optimal guillotine separable set for MISR instance.

- $DP[C]$ stores optimal guillotine separable set for MISR for rectangular region C.
- Base cases:
  If C coincide with a rectangle R, give $\{R\}$ as solution.
- If C contains no rectangle, give $\{\phi\}$ as solution.



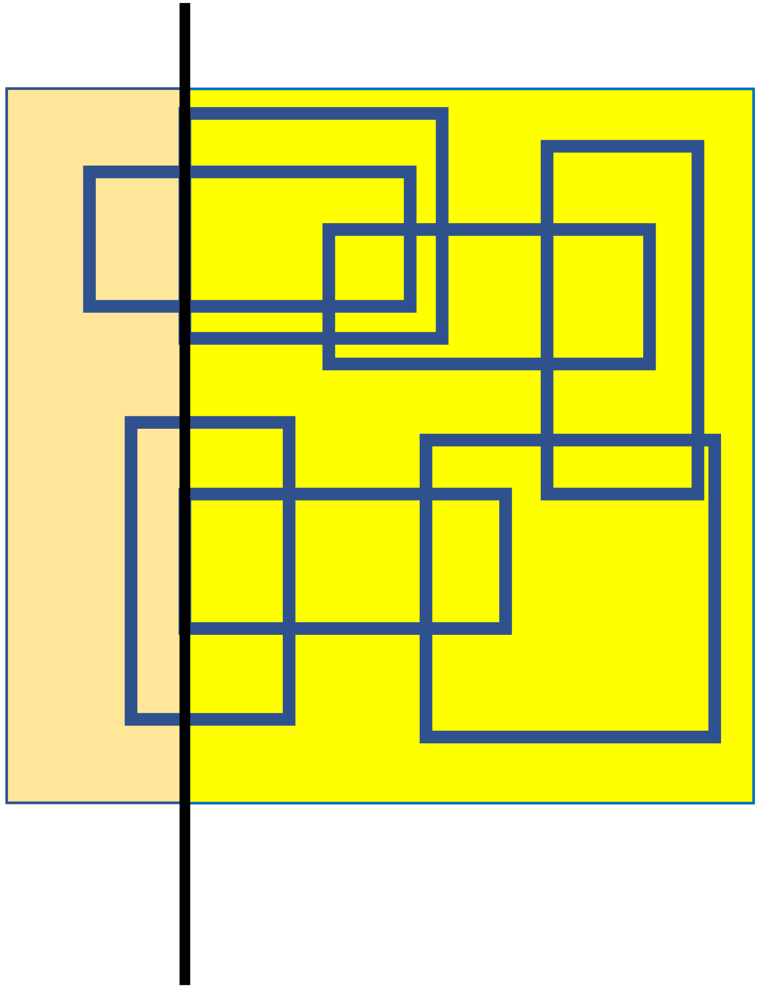
DP for optimal guillotine separable set for MISR instance.

- Recurrence: $DP[C]$ = $DP[C_1] \cup DP[C_2]$, where $|DP[C_1]| + |DP[C_2]|$ is maximum among all partitions C_1, C_2 of C by some guillotine cut.



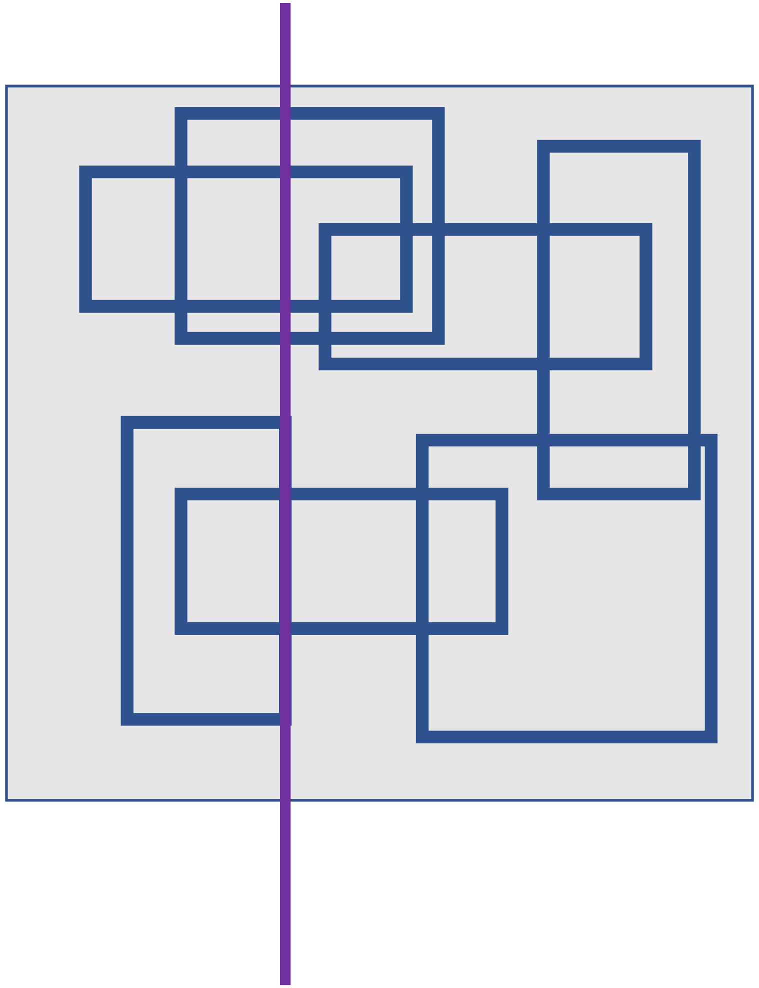
DP for optimal guillotine separable set for MISR instance.

- Recurrence: $DP[C]$ = $DP[C_1] \cup DP[C_2]$, where $|DP[C_1]| + |DP[C_2]|$ is maximum among all partitions C_1, C_2 of C by some guillotine cut.



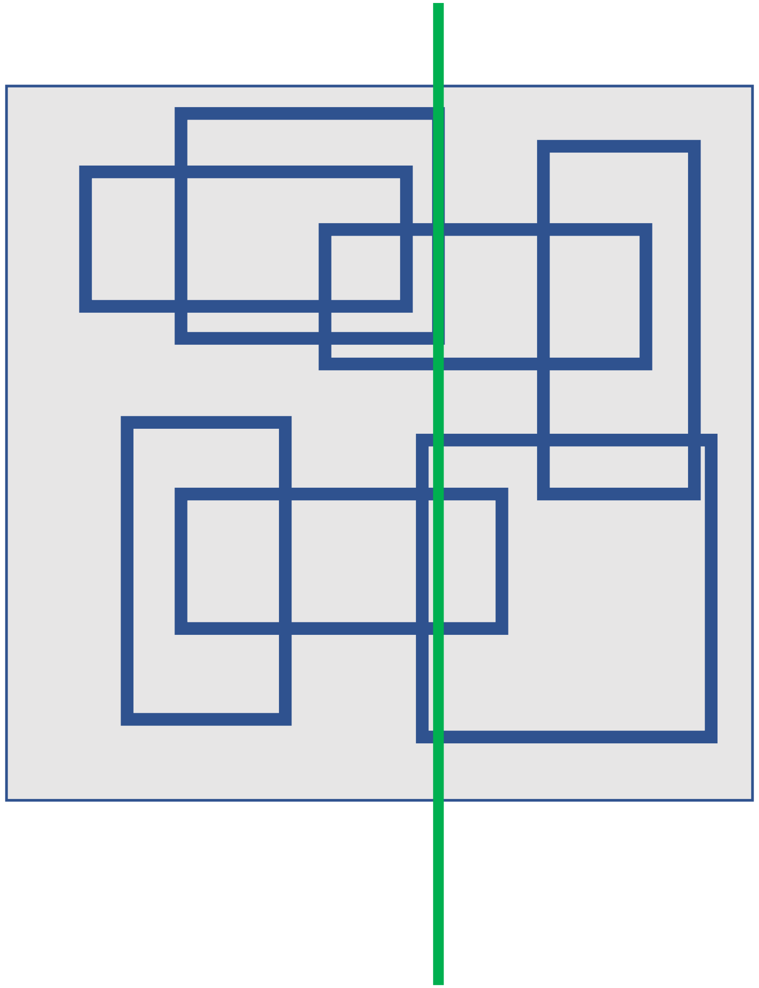
DP for optimal guillotine separable set for MISR instance.

- Recurrence: $DP[C]$ = $DP[C_1] \cup DP[C_2]$, where $|DP[C_1]| + |DP[C_2]|$ is maximum among all partitions C_1, C_2 of C by some guillotine cut.



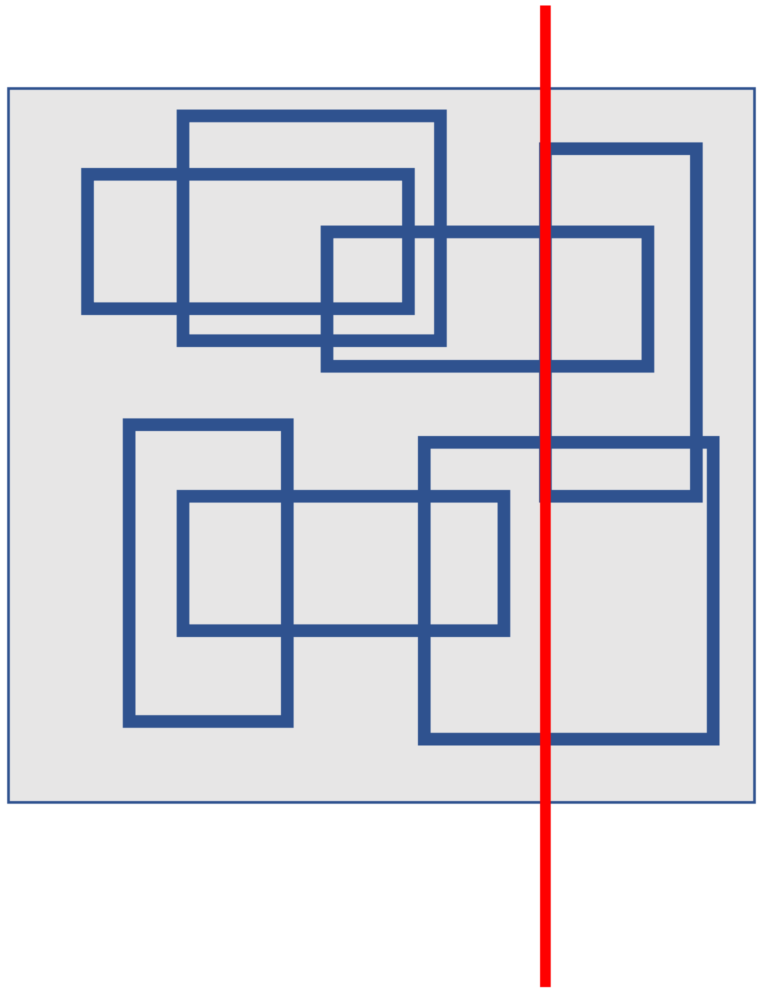
DP for optimal guillotine separable set for MISR instance.

- Recurrence: $DP[C]$ = $DP[C_1] \cup DP[C_2]$, where $|DP[C_1]| + |DP[C_2]|$ is maximum among all partitions C_1, C_2 of C by some guillotine cut.



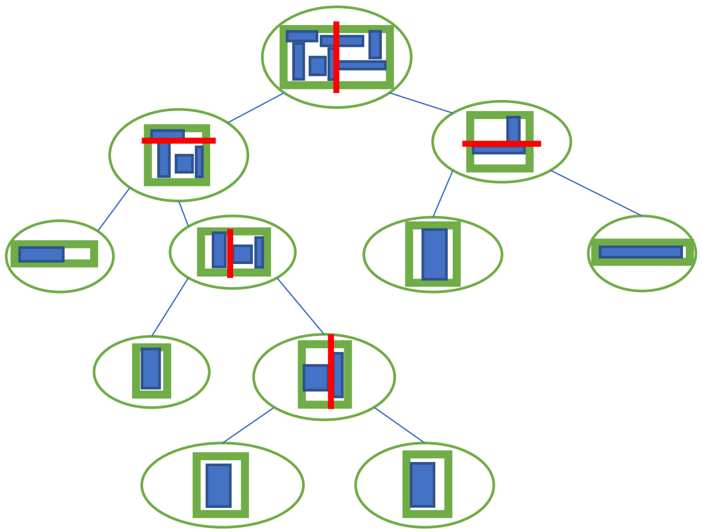
DP for optimal guillotine separable set for MISR instance.

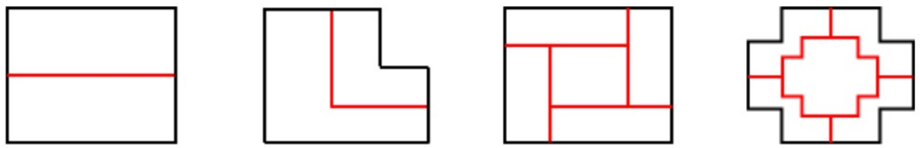
- Recurrence: $DP[C]$ = $DP[C_1] \cup DP[C_2]$, where $|DP[C_1]| + |DP[C_2]|$ is maximum among all partitions C_1, C_2 of C by some guillotine cut.
- There are $O(n)$ such cuts. $O(n^4)$ DP cells. $\Rightarrow O(n^5)$-algorithm.



Alternate Structured Solution?

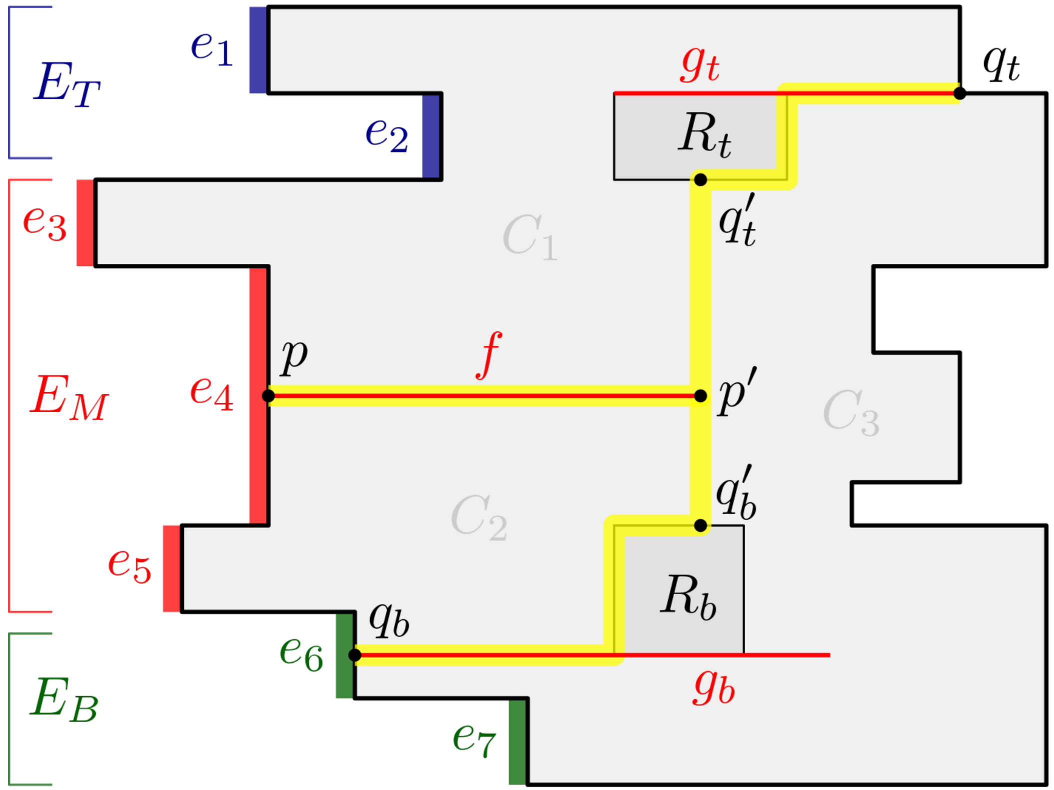
- Problem with Guillotine separability:
  We only know $\alpha \leq (\log n + 1)$.
- Can we generalize this idea?
  -- Instead of binary tree to k-ary tree.
  -- Instead of rectangular region allow orthogonal polygons with t sides.
  -- ($k \geq 2, t \geq 4$, are integers).
- Generalizes guillotine! By allowing more flexibility we might have a better approximation as well.
- For 6-approximation, we will use $k = 3, t = 26$.





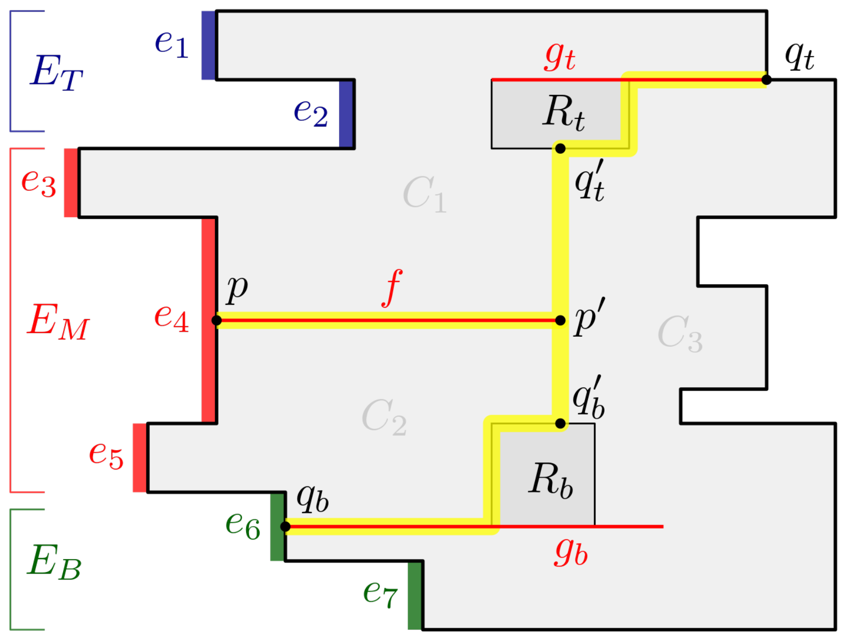
k-ary Partition into t-sided polygons

- Problem with Guillotine separability:
  We only know $\alpha \leq (\log n + 1)$.
- Can we generalize this idea?
  -- Instead of binary tree to k-ary tree.
  -- Instead of rectangular region allow orthogonal polygons with t sides.
  -- ($k \geq 2, t \geq 4$, are integers).
- Generalizes guillotine! By allowing more flexibility we might have a better approximation as well.
- For 6-approximation, we will use $k = 3, t = 26.$
- For $2 + \epsilon$-approximation, we will use $k = 2 + \epsilon, t = O_\epsilon(1).$



k-ary Partition into t-sided polygons

- Property of DP Table:
- Number of possible orthogonal polygons: $\leq (2n)^t$.
- Maximum number of segments in a possible cut for k-ary partition of a t-sided orthogonal polygon $\leq kt/2$.
- Number of possible cuts: $(2n)^{kt/2}$.
- DP Runtime = $O\left(n^{\frac{(k+2)t}{2}}\right)$.
- We will show such $k = 3, t = 26$ partitions give 6-approximation, implying time complexity $O(n^{65})$.



Recursive partitioning

- If there exists a recursive partition with k and t to be $O(1)$ then DP will find the best such solution.
- Main problem: How to show the existence of such a recursive partition?

-- cutting strategy [fences and cutting].

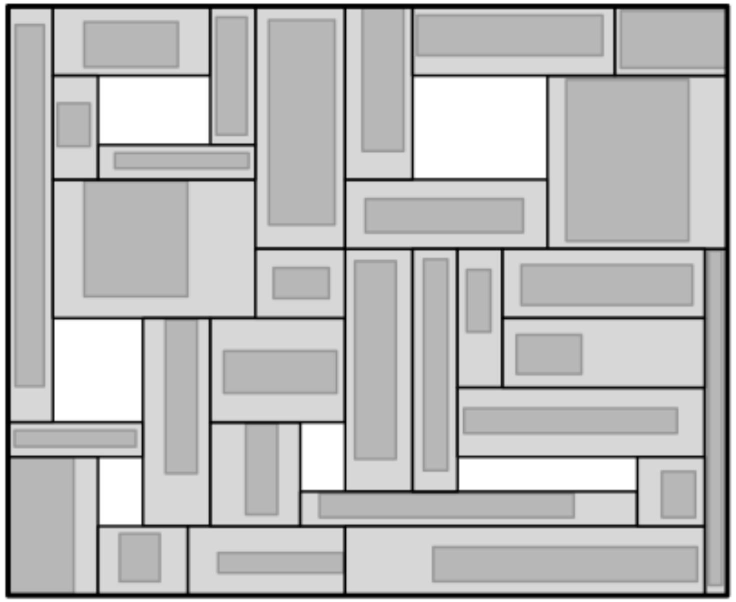
-- analysis [nesting and token counting].

Recursive partitioning

- Assume Maximality for $\mathcal{R}$

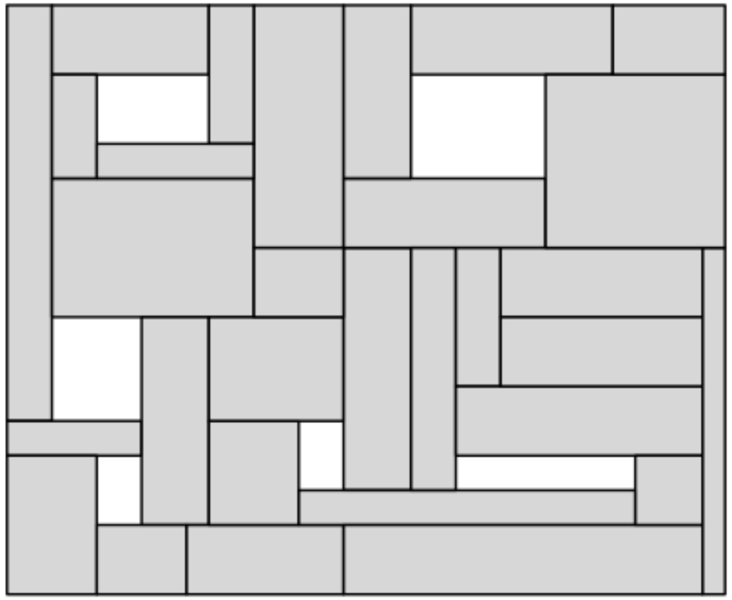
Recursive partitioning

- Assume Maximality for $\mathcal{R}$



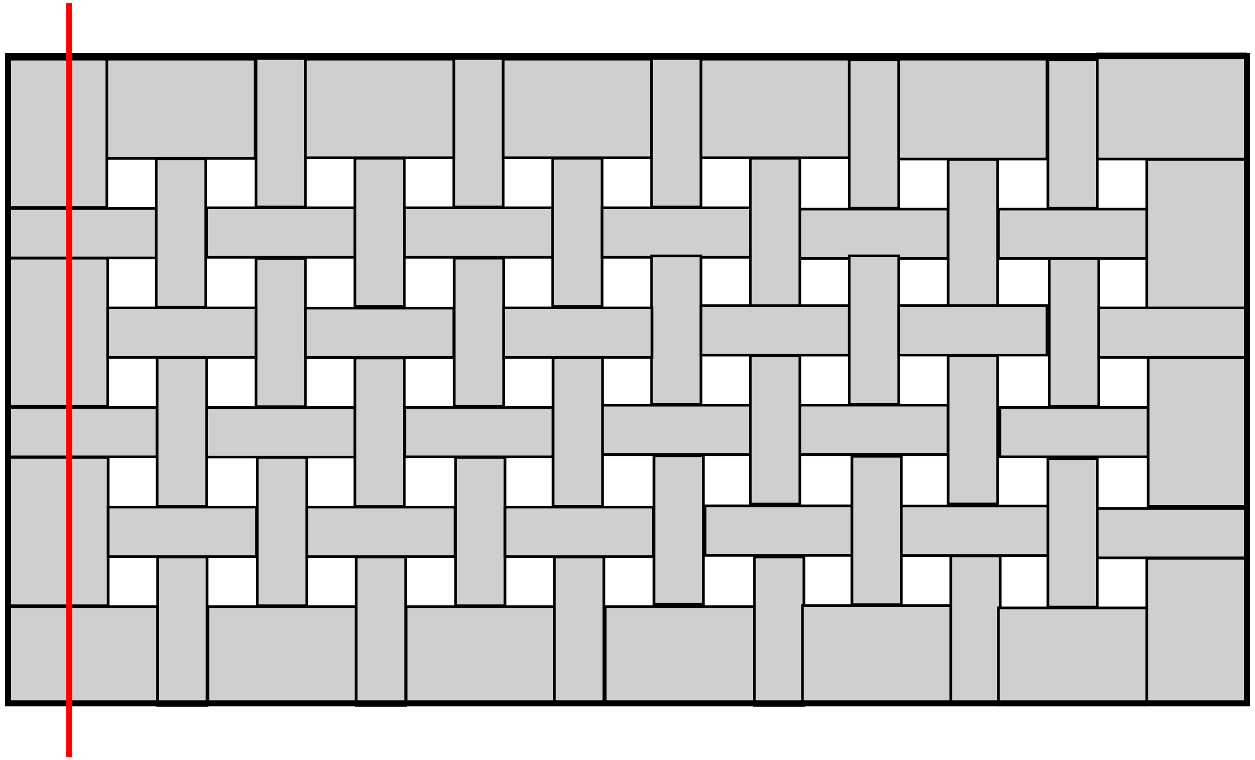
Recursive partitioning

- Assume Maximality for $\mathcal{R}$



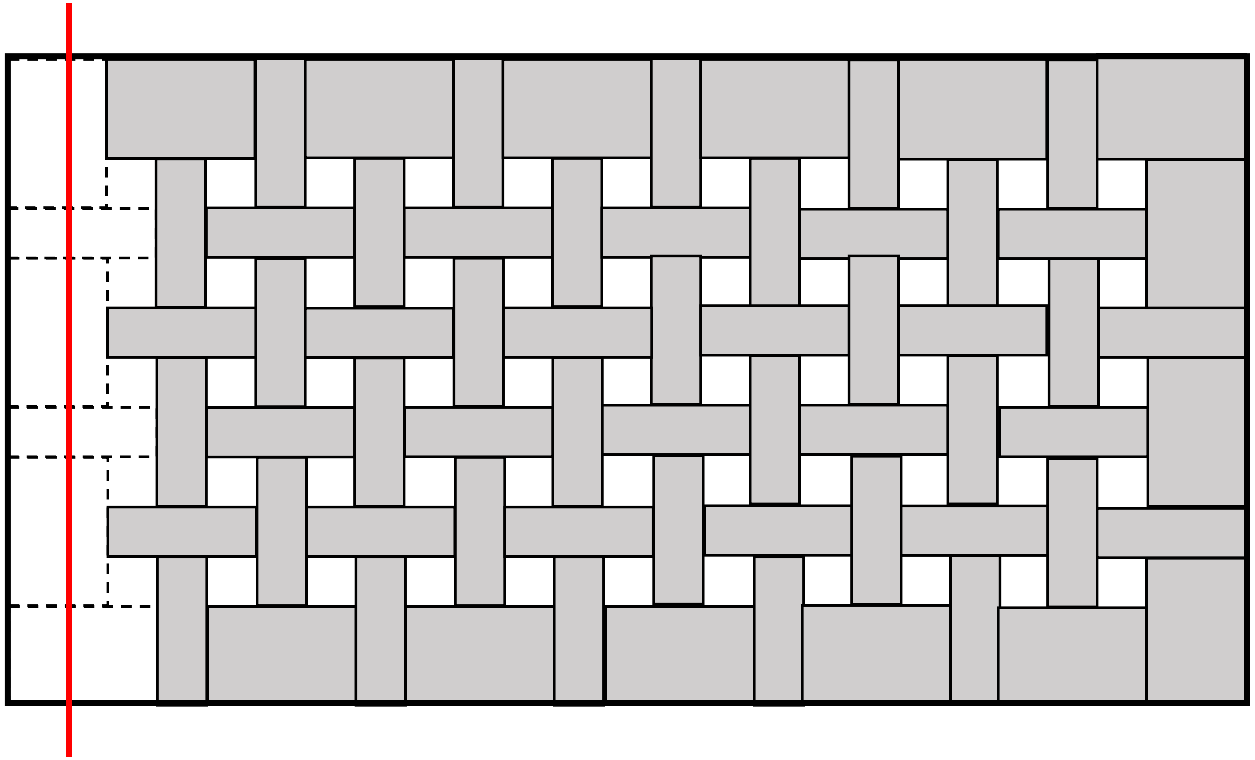
Recursive partitioning

- Fences and Cutting



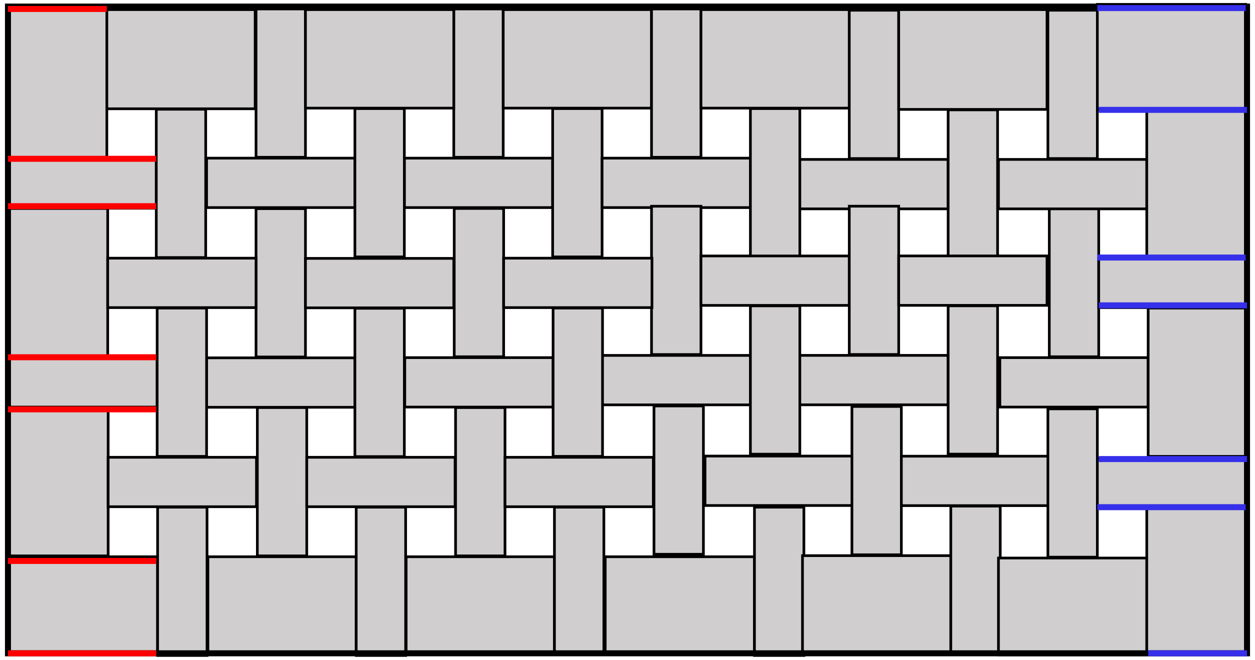
Recursive partitioning

- Fences and Cutting
  - Intuitively, better cuts are not near the boundary
  - We made no progress



Recursive partitioning

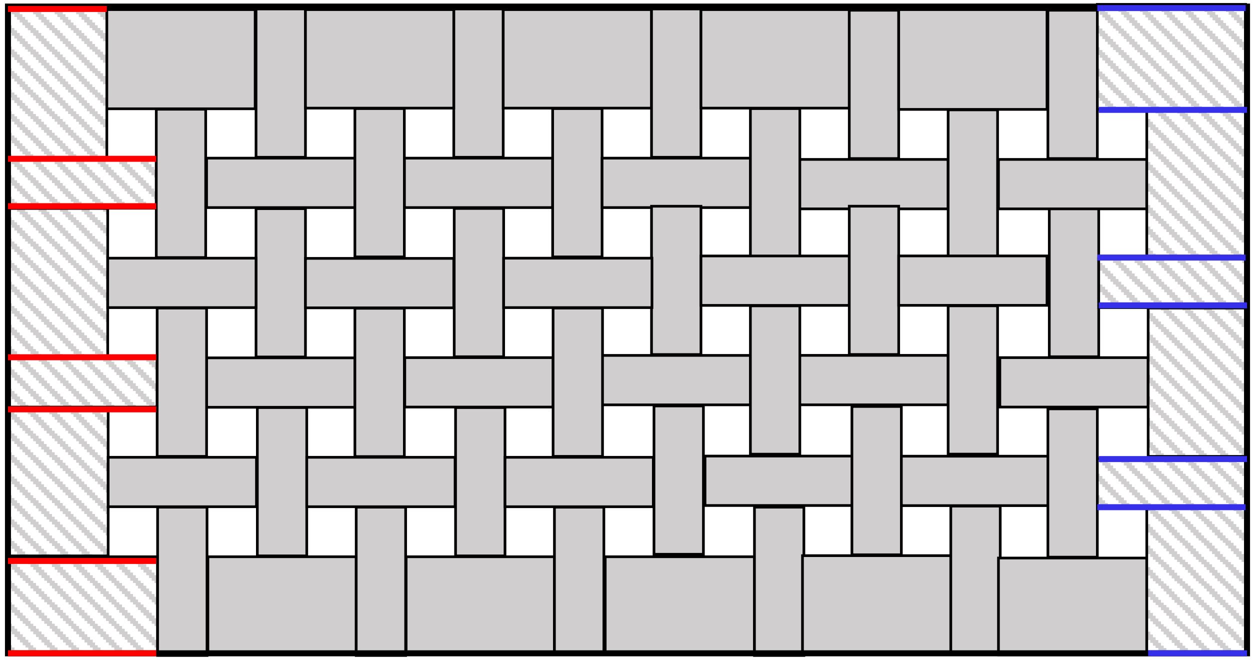
- Fences and Cutting
  - Intuitively, better cuts are not near the boundary
  - Block/Protect all boundary rectangles in every piece using fences



- Fences are horizontal line segments that join boundary of piece to boundary of a rectangle without intersecting any rectangles
- A rectangle is boundary rectangle in a piece if any of its horizontal edge is contained in a fence

Recursive partitioning

- Fences and Cutting
  - Intuitively, better cuts are not near the boundary
  - Block/Protect all boundary rectangles in every piece using fences



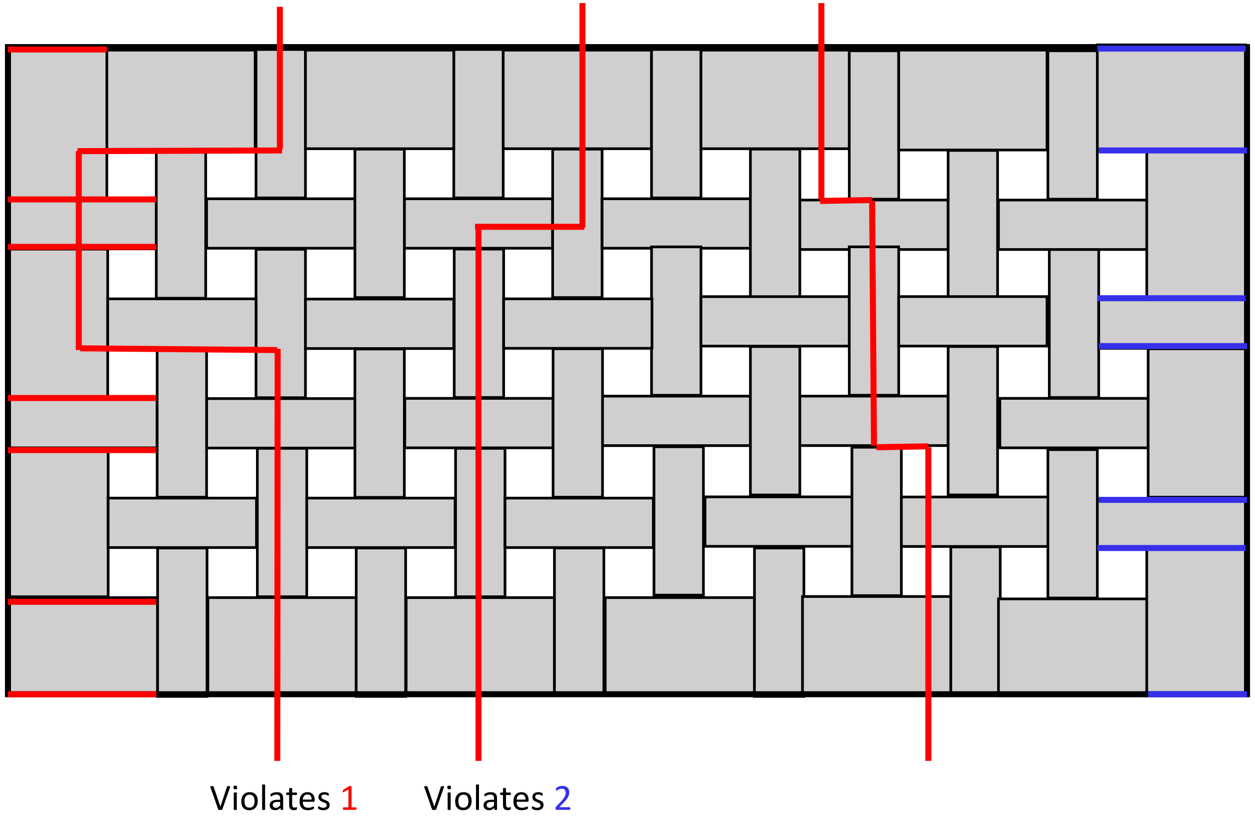
- Fences are horizontal line segments that join boundary of piece to boundary of a rectangle without intersecting any rectangles
- A rectangle is boundary rectangle in a piece if any of its horizontal edge is contained in a fence

Recursive partitioning

- Fences and Cutting
  - Intuitively, better cuts are not near the boundary
  - Block/Protect all boundary rectangles in every piece using fences

- Rules of cutting:

1. Vertical segments of cuts don’t pass through fences
2. Horizontal segments of cuts don’t intersect any rectangles



- Fences are horizontal line segments that join boundary of piece to boundary of a rectangle without intersecting any rectangles

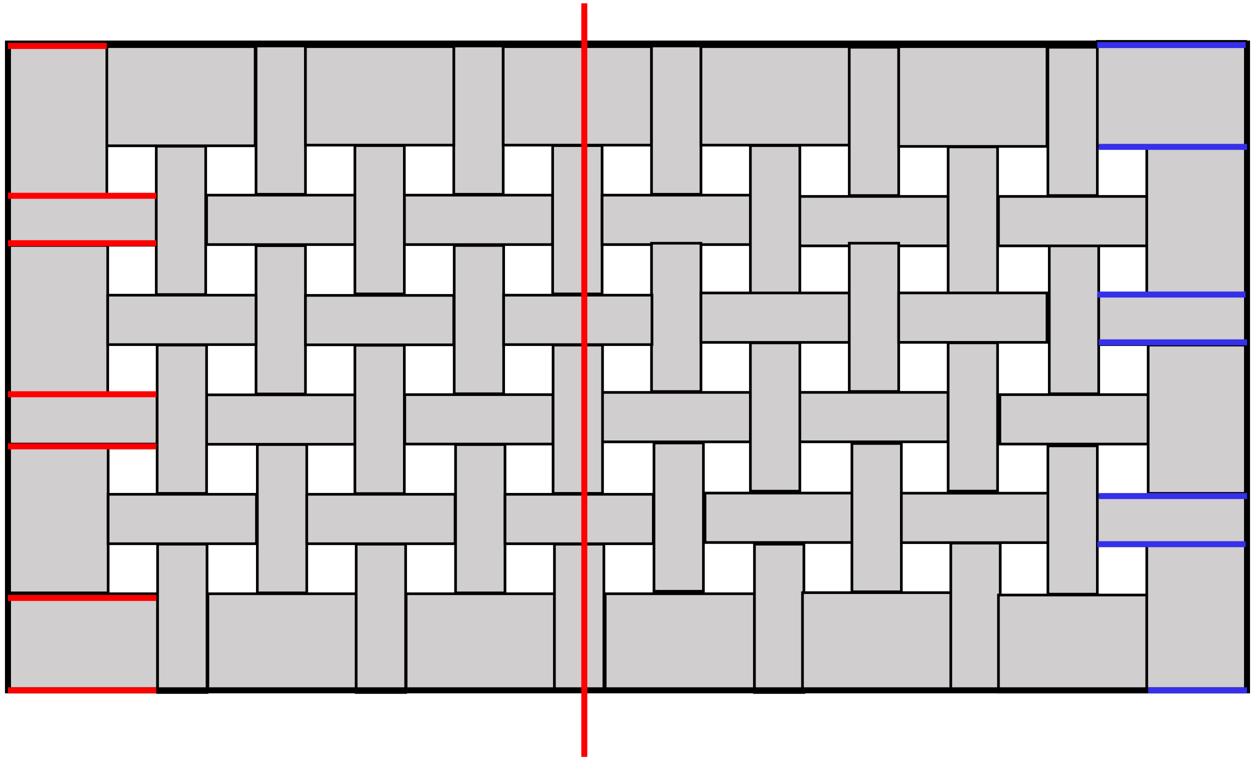
- A rectangle is boundary rectangle in a piece if any of its horizontal edge is contained in a fence

Recursive partitioning

- Fences and Cutting
  - Intuitively, better cuts are not near the boundary
  - Block/Protect all boundary rectangles in every piece using fences

- Rules of cutting:

1. Vertical segments of cuts don't pass through fences
2. Horizontal segments of cuts don't intersect any rectangles



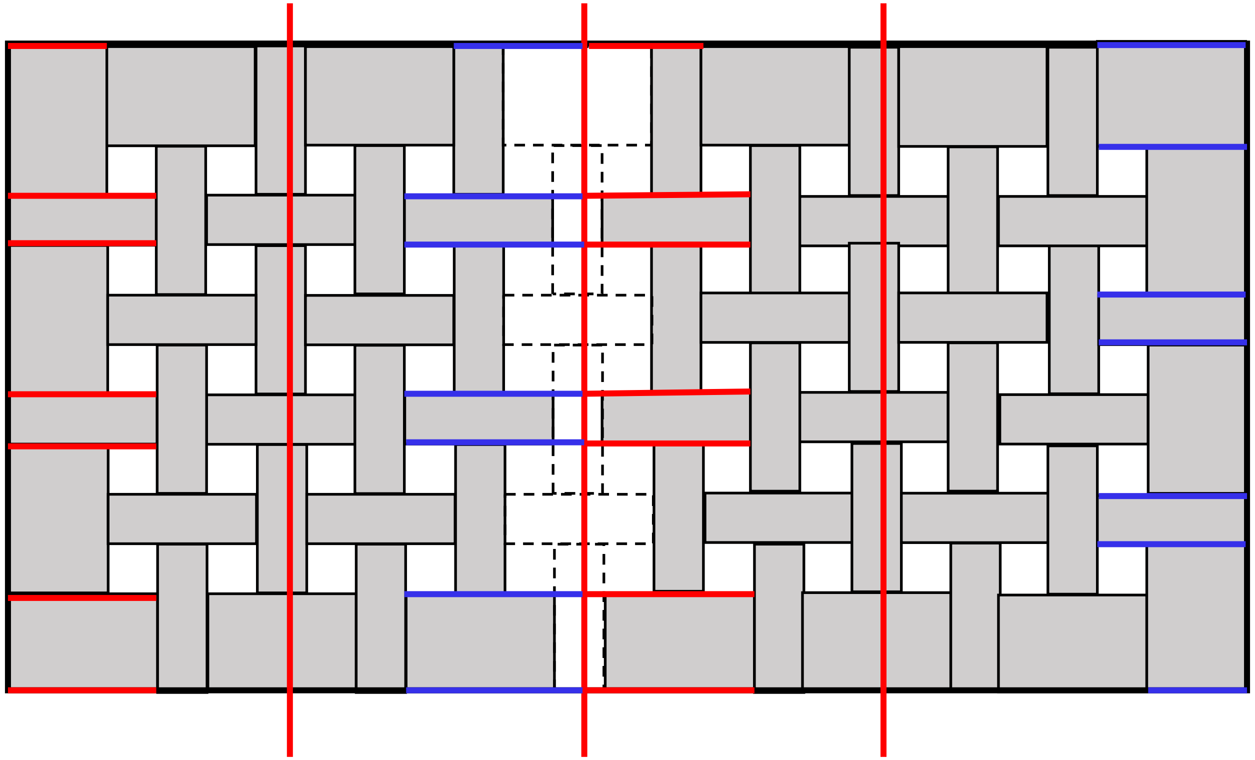
- Fences are horizontal line segments that join boundary of piece to boundary of a rectangle without intersecting any rectangles
- A rectangle is boundary rectangle in a piece if any of its horizontal edge is contained in a fence

Recursive partitioning

- Fences and Cutting
    - Intuitively, better cuts are not near the boundary
    - Block/Protect all boundary rectangles in every piece using fences

- Rules of cutting:

1. Vertical segments of cuts don't pass through fences
2. Horizontal segments of cuts don't intersect any rectangles



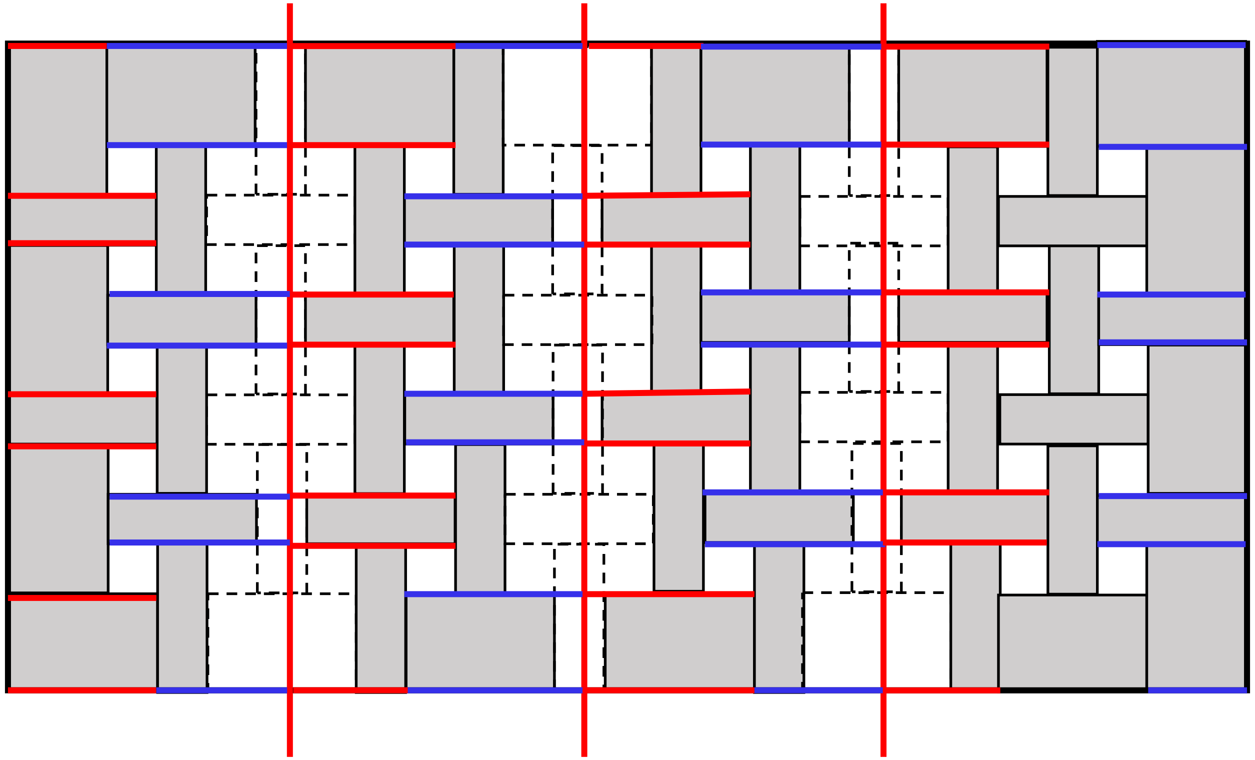
- Fences are horizontal line segments that join boundary of piece to boundary of a rectangle without intersecting any rectangles
- A rectangle is boundary rectangle in a piece if any of its horizontal edge is contained in a fence

Recursive partitioning

- Fences and Cutting
  - Intuitively, better cuts are not near the boundary
  - Block/Protect all boundary rectangles in every piece using fences

- Rules of cutting:

1. Vertical segments of cuts don’t pass through fences
2. Horizontal segments of cuts don’t intersect any rectangles



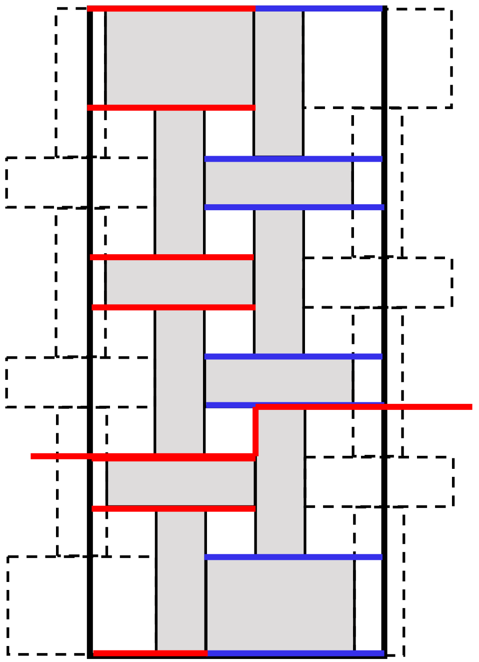
- Fences are horizontal line segments that join boundary of piece to boundary of a rectangle without intersecting any rectangles
- A rectangle is boundary rectangle in a piece if any of its horizontal edge is contained in a fence

Recursive partitioning

- Fences and Cutting
  - Intuitively, better cuts are not near the boundary
  - Block/Protect all boundary rectangles in every piece using fences

- Rules of cutting:

1. Vertical segments of cuts don’t pass through fences
2. Horizontal segments of cuts don’t intersect any rectangles



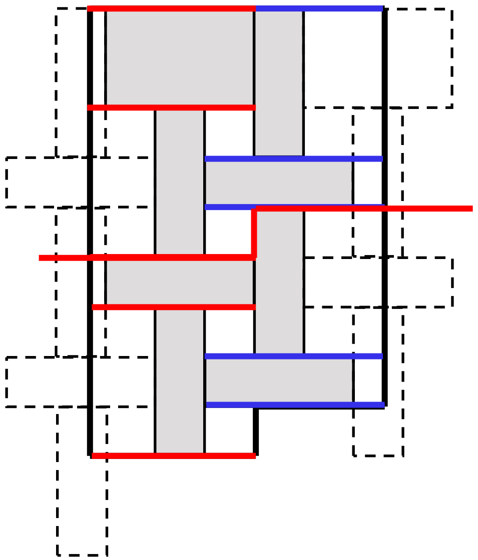
- Cannot make a straight vertical or horizontal cut without cutting any of the rectangles
- Use bends!

Recursive partitioning

- Fences and Cutting
  - Intuitively, better cuts are not near the boundary
  - Block/Protect all boundary rectangles in every piece using fences

- Rules of cutting:

1. Vertical segments of cuts don't pass through fences
2. Horizontal segments of cuts don't intersect any rectangles



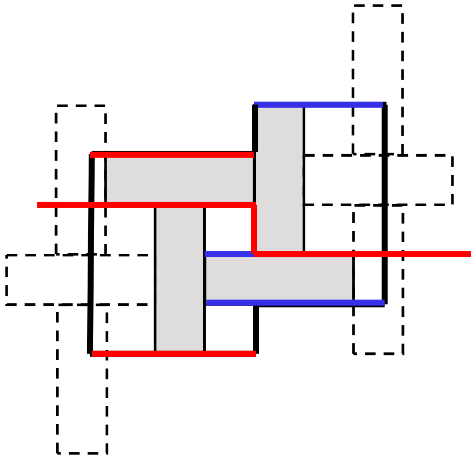
- Cannot make a straight vertical or horizontal cut without cutting any of the rectangles
- Use bends!

Recursive partitioning

- Fences and Cutting
  - Intuitively, better cuts are not near the boundary
  - Block/Protect all boundary rectangles in every piece using fences

- Rules of cutting:

1. Vertical segments of cuts don’t pass through fences
2. Horizontal segments of cuts don’t intersect any rectangles



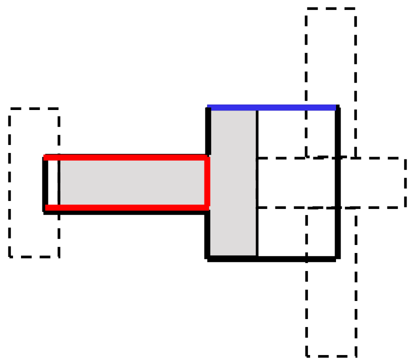
- Cannot make a straight vertical or horizontal cut without cutting any of the rectangles
- Use bends!

Recursive partitioning

- Fences and Cutting
  - Intuitively, better cuts are not near the boundary
  - Block/Protect all boundary rectangles in every piece using fences

- Rules of cutting:

1. Vertical segments of cuts don't pass through fences
2. Horizontal segments of cuts don't intersect any rectangles



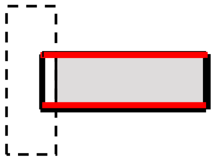
- Cannot make a straight vertical or horizontal cut without cutting any of the rectangles
- Use bends!

Recursive partitioning

- Fences and Cutting
  - Intuitively, better cuts are not near the boundary
  - Block/Protect all boundary rectangles in every piece using fences

- Rules of cutting:

1. Vertical segments of cuts don’t pass through fences
2. Horizontal segments of cuts don’t intersect any rectangles
3. Partition rule



- Cannot make a straight vertical or horizontal cut without cutting any of the rectangles
- Use bends!

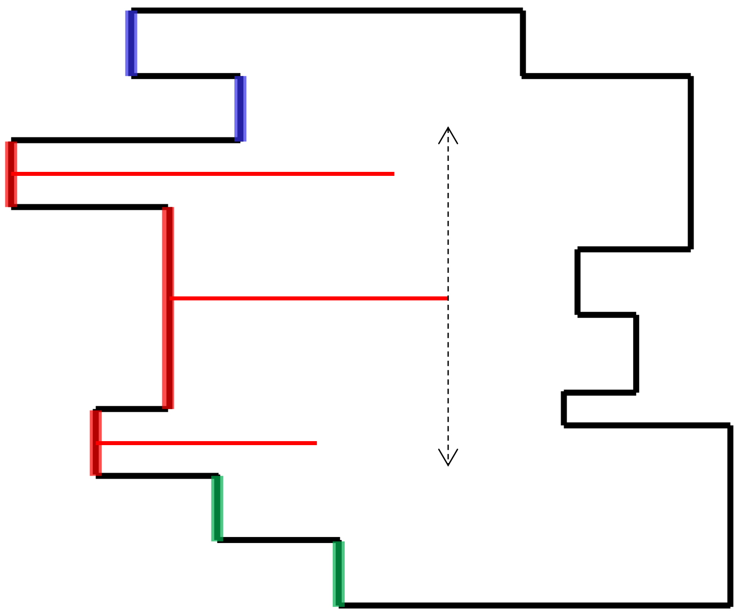
How do we make sure that the piece complexity doesn't go up in this process?

Recursive partitioning

- Fences and Cutting
    - Intuitively, better cuts are not near the boundary
    - Block/Protect all boundary rectangles in every piece using fences

- Rules of cutting:

1. Vertical segments of cuts don’t pass through fences
2. Horizontal segments of cuts don’t intersect any rectangles
3. Partition rule



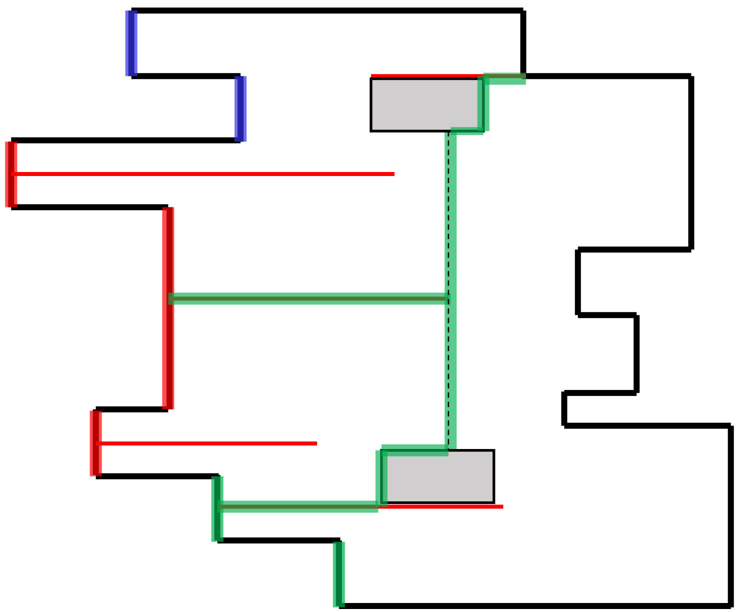
- Invariant: Assume the pieces are horizontally convex

Recursive partitioning

- Fences and Cutting
  - Intuitively, better cuts are not near the boundary
  - Block/Protect all boundary rectangles in every piece using fences

- Rules of cutting:

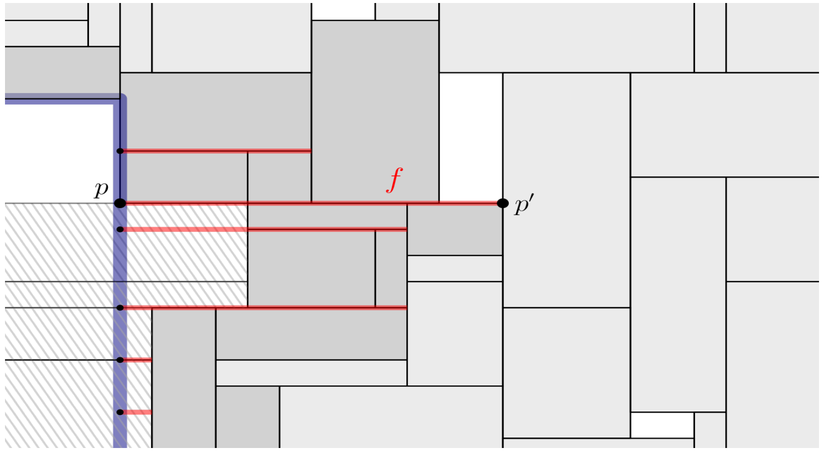
1. Vertical segments of cuts don't pass through fences
2. Horizontal segments of cuts don't intersect any rectangles
3. Partition rule

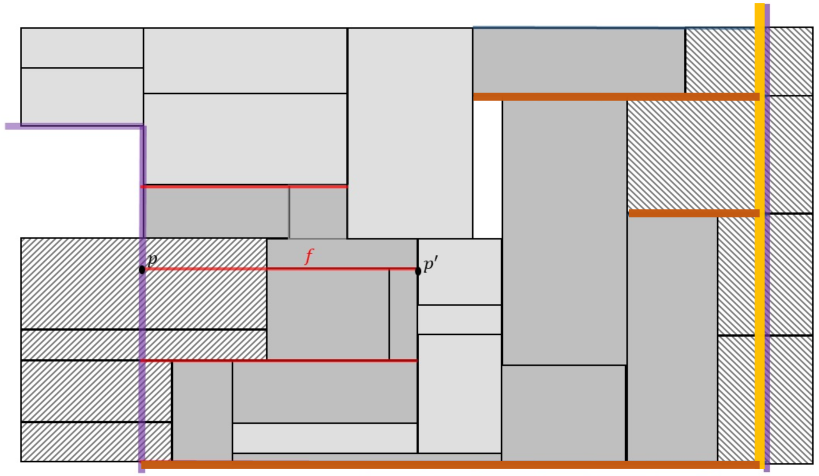


- Invariant: Assume the pieces are horizontally convex
- Invariant is maintained and piece complexity does not increase

Recursive partitioning

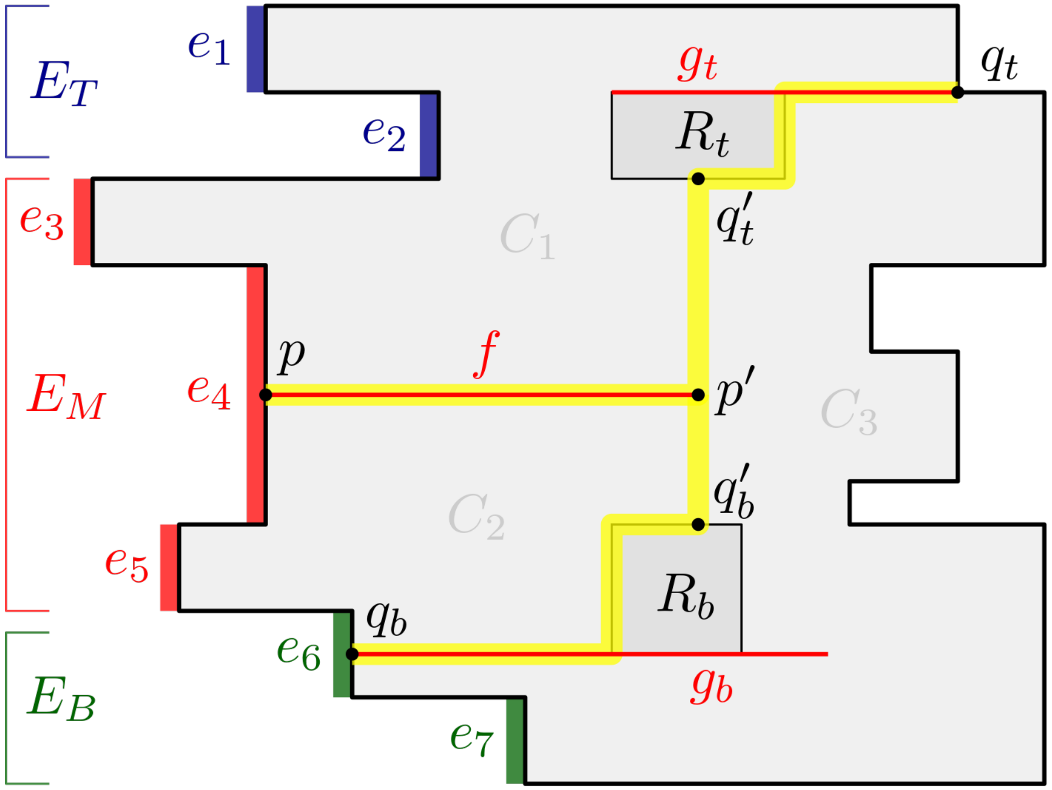
- Idea: Protect rectangles using fences.
- From each point p on left vertical edge of piece P shoot a horizontal ray towards right till it reaches point p' without intersecting any rectangle of $OPT(P)$ and p' is contained in the interior of the left side of a rectangle in $OPT(P)$ or reaches boundary.
- pp' is a line fence.
- Symmetrically, for points on right vertical edge shoot horizontal ray towards left.
- A rectangles is protected if its top or bottom edge is contained in some fence.





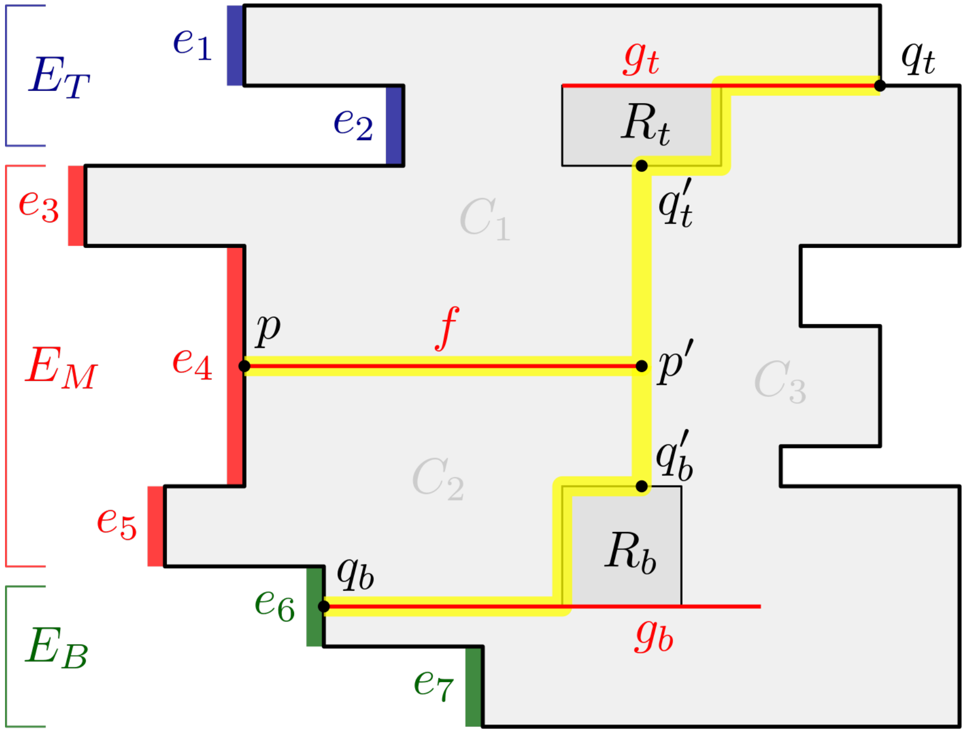
Line-partitioning lemma

- Given a horizontally convex polygon P with at most 26 sides, there exits a cut C s.t.
  1. C has at most 8 line segments.
  2. C divides P into at most 3 axis-parallel polygons (each with at most 26 sides).
  3. There is only a single line segment ℓ that can intersect some rectangles in $OPT(P)$.
  4. However, ℓ does not intersect any protected rectangles (so ℓ does not cross any fence).



Proof of Line-partitioning lemma

- P has ≤ 26 edges, i.e. $v \leq 13$ vertical edges.
- Say there are $\geq \lceil v/2 \rceil$ left vertical edges.
- Divide them into 3 groups E_T, E_M, E_B s. t. $|E_T|, |E_B| \geq v/6, |E_M| \geq \lceil v/6 \rceil$
- Let fence f from E_M have the rightmost p′.
- Shoot vertical rays to top and bottom till it hits a protected rectangle to create ℓ.
- Cut C is formed by line fences on top and bottom and boundary of the protected rectangle alongwith ℓ.



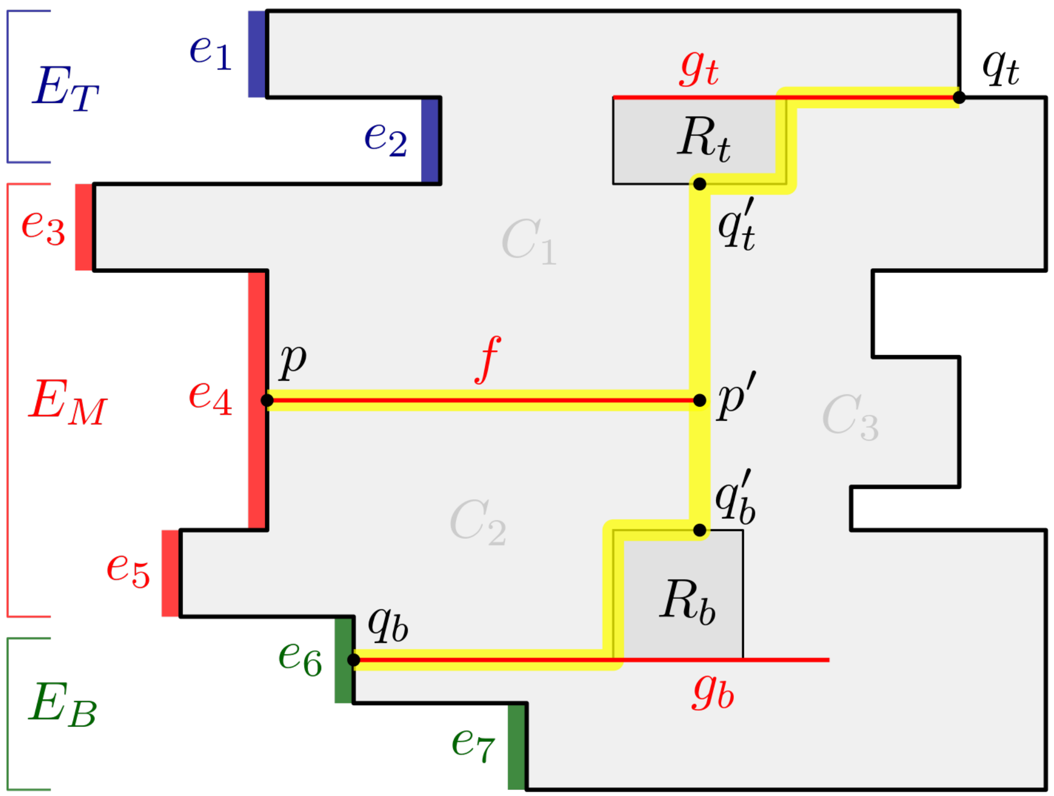
Proof of Line-partitioning lemma

Easy to check the following properties:

1. C has at most 8 line segments.
[implies $O(n^{(26+8)}) = O(n^{34})$ runtime.]

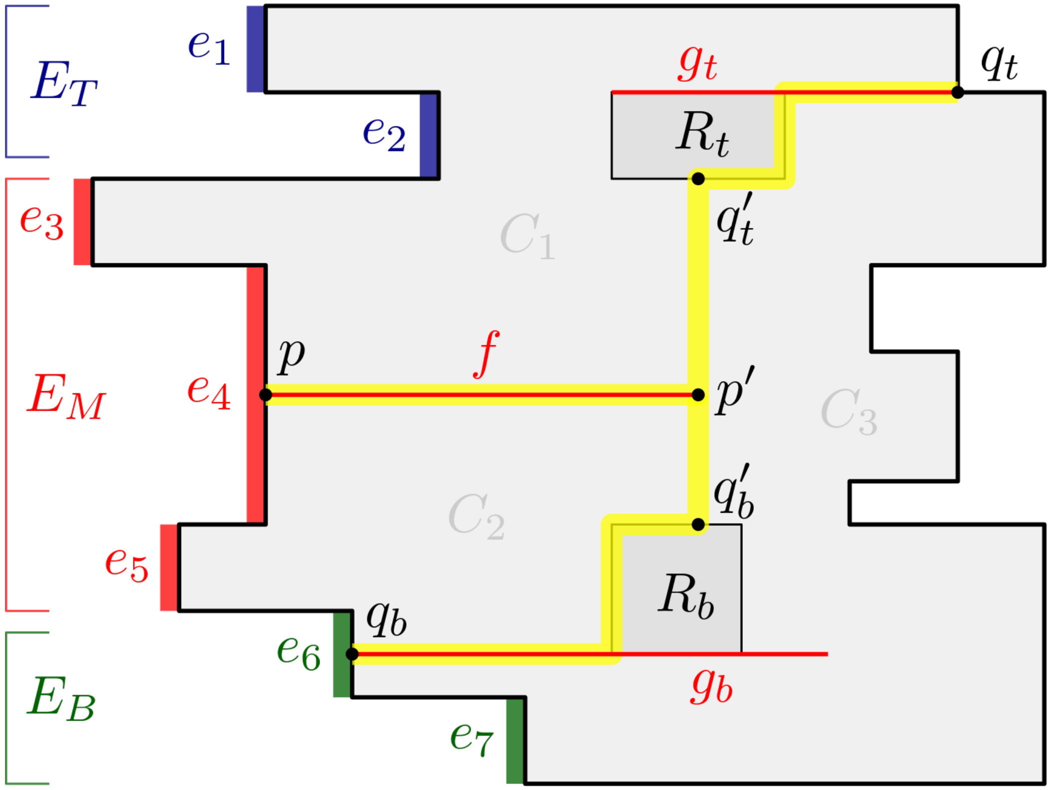
3. There is only a single line segment ℓ that can intersect some rectangles in $OPT(P)$.

4. However, ℓ does not intersect any protected rectangles (so ℓ does not cross any fence).



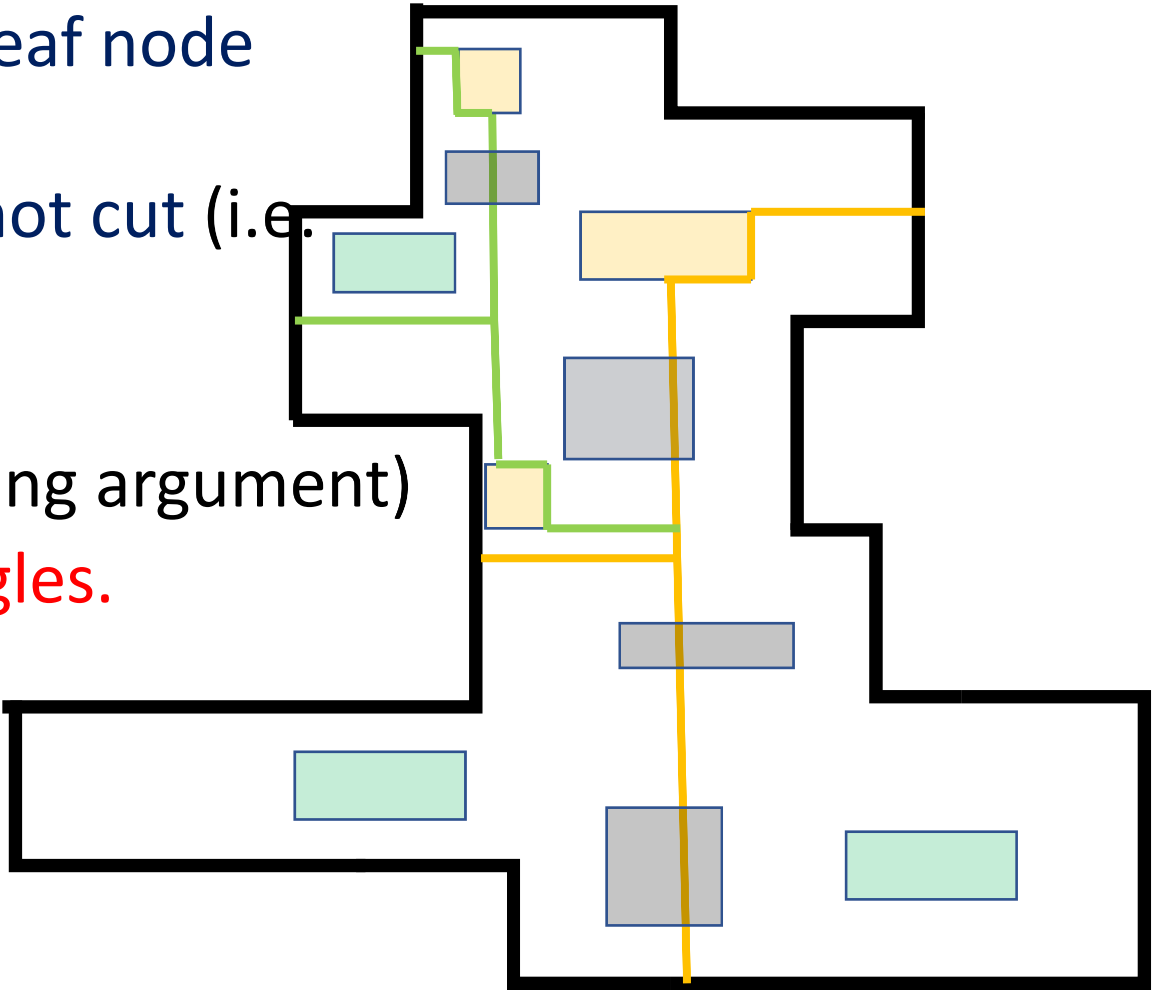
Proof of Line-partitioning lemma

- Property 2. C divides P into at most three axis-parallel polygons (each with at most 26 sides).
- Boundary of C_1 (resp. C_2) is disjoint from E_B (resp. E_T).
- Number of vertical edges in C_1 is $\leq v - \left\lfloor \frac{v}{6} \right\rfloor + 2 = \left\lceil \frac{5v}{6} \right\rceil + 2 \leq \left\lceil \frac{5 \times 13}{6} \right\rceil + 2 = 13$.
- Boundary of C_3 is disjoint from E_M.
- Number of vertical edges in C_3 is $\leq v - \left\lceil \frac{5v}{6} \right\rceil + 3 = \left\lfloor \frac{5v}{6} \right\rfloor + 3 \leq \left\lfloor \frac{5 \times 13}{6} \right\rfloor + 3 = 13$.



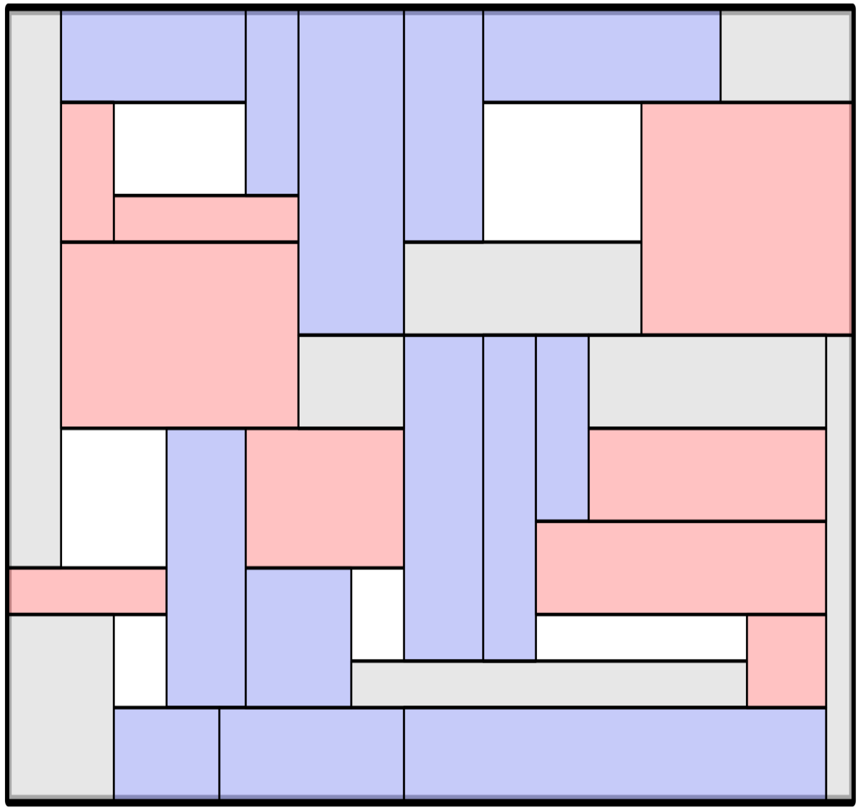
Recursive Partitioning

- We continue the partitioning till each leaf node contains one node.
- We return $\mathcal{R}$, all rectangles that were not cut (i.e. belong to some leaf).
- Need to show $|\mathcal{R}| \geq |OPT|/6$
- High level idea: (Charging/token counting argument)
- Killed rectangles save sufficient rectangles.
- Each killed rectangle that is not horizontally nested will distribute one token to some of its neighbors that it can see.

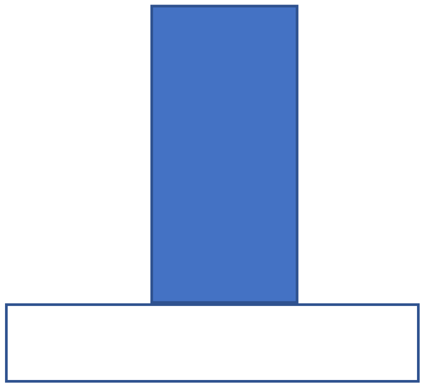


Proof of Approximation Guarantee

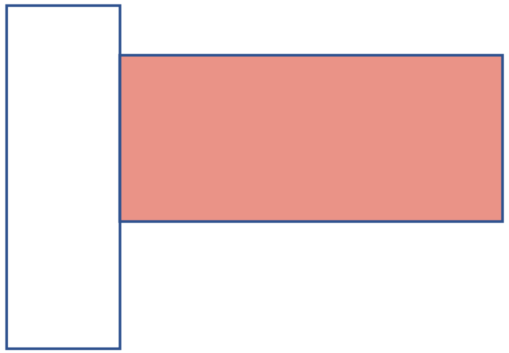
- Assumption:
  All rectangles in OPT are maximal.
- Nesting Relationship:
  A rectangle is vertically nested or blue (resp. horizontally nested or red) if its top or bottom edge (resp. right or left edge) is contained in the interior of some other rectangle or interior of a boundary edge.
- Observation: A rectangles is either red or blue or none (grey).
- Wlog assume at most half rectangles are red.



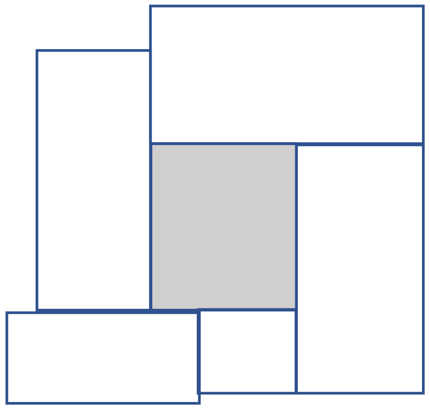
Proof of Approximation Guarantee



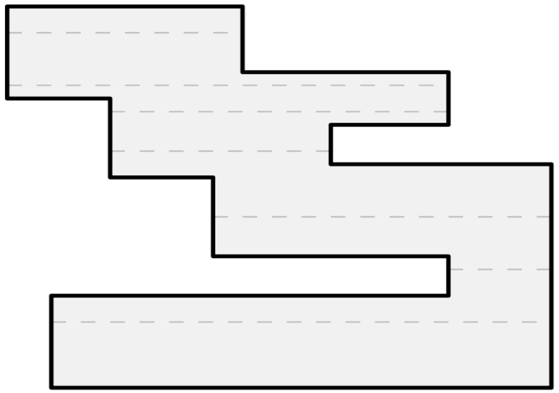
Vertically nested



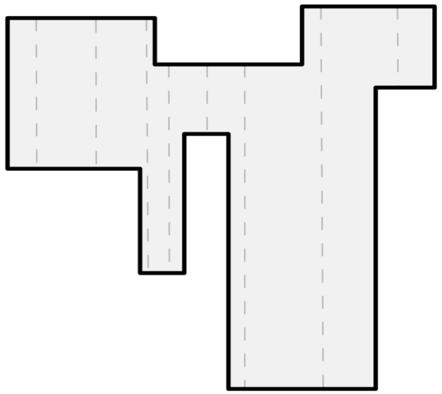
Horizontally nested



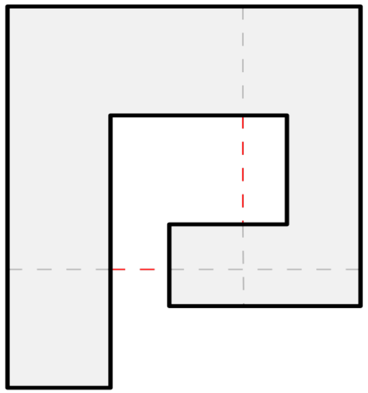
Neither



horizontally convex

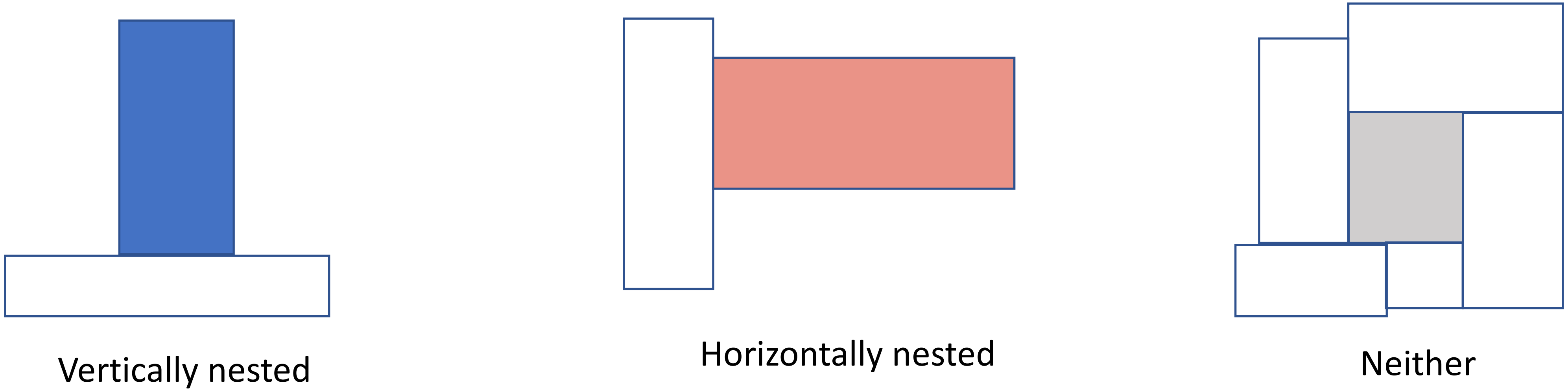


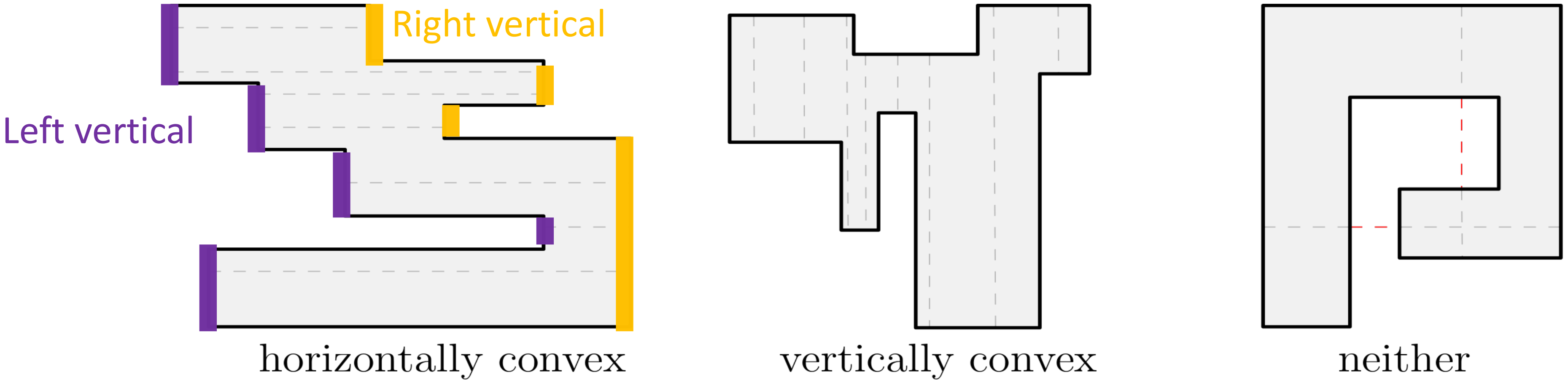
vertically convex



neither

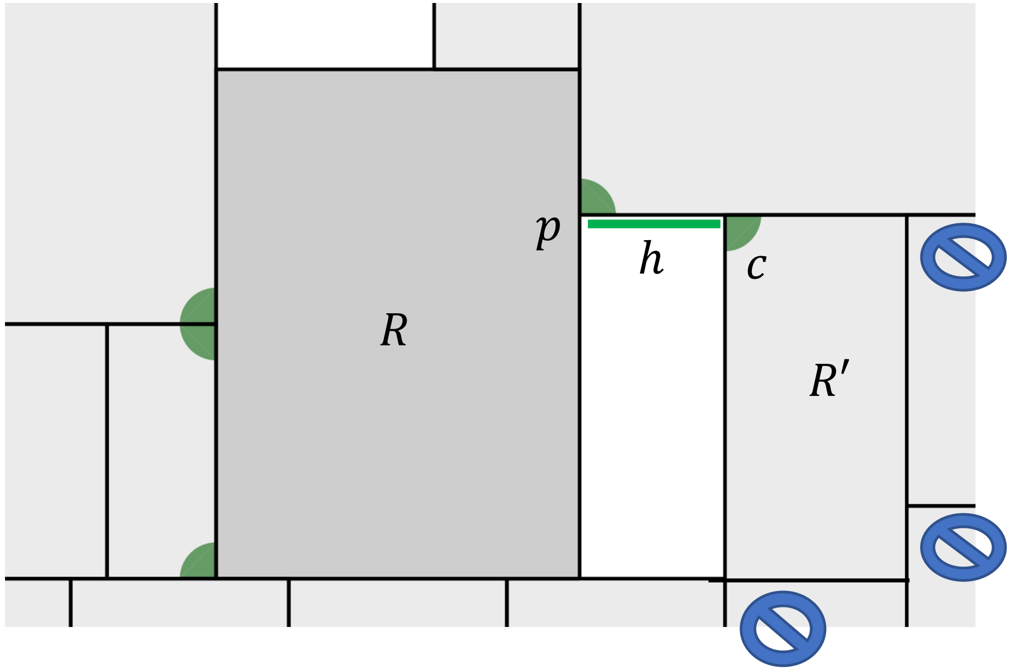
Proof of Approximation Guarantee

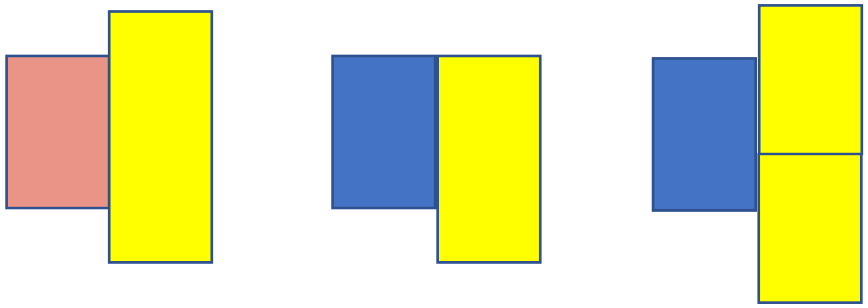




Proof of Approximation Guarantee

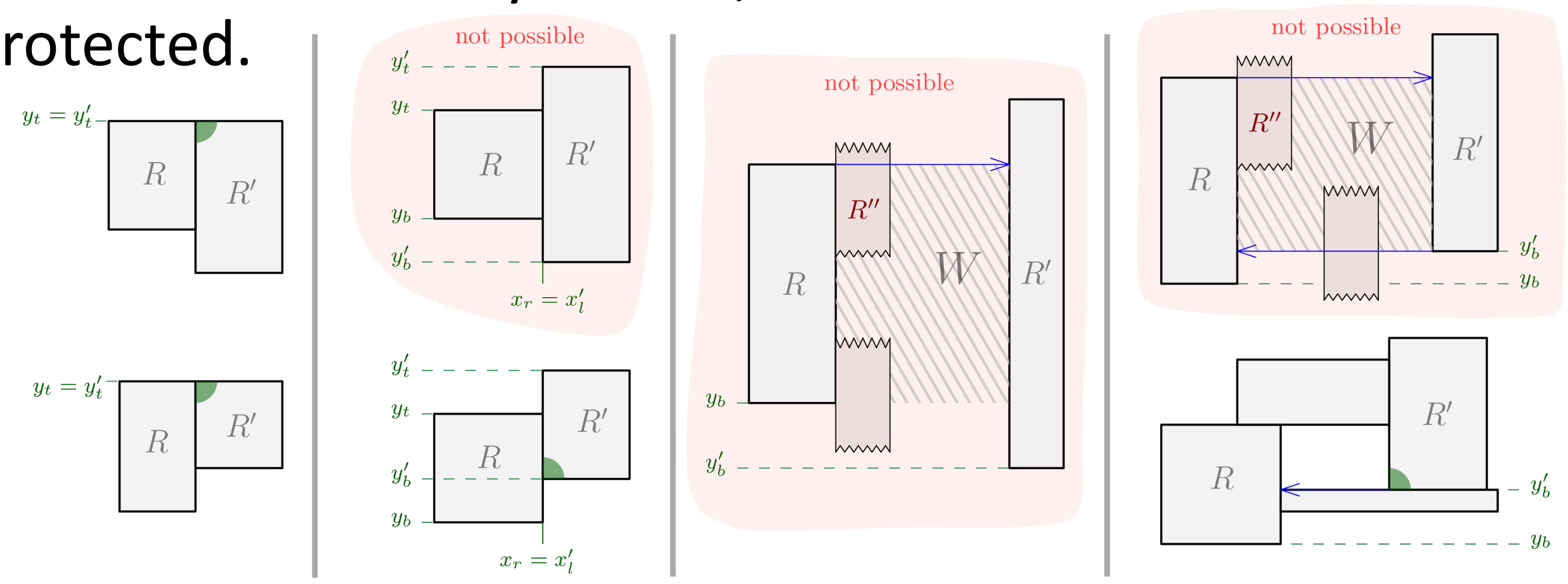
- Rectangle R sees the top-left corner c of rectangle $R' \in OPT$ on its right if there is a line segment h joining a point p on R with c s.t. h does not intersect any rectangle in OPT, h does not contain the top edge of any rectangle in OPT, and p is not the bottom-right corner of R.
- If R is hor. nested, it does not see any corner in the nested sides.
- If R is ver. nested or grey, it sees at least one corner on each side.





Proof of Approximation Guarantee

- Claim 1: If R is not horizontally nested, then it sees at least one corner on each side, or is protected.



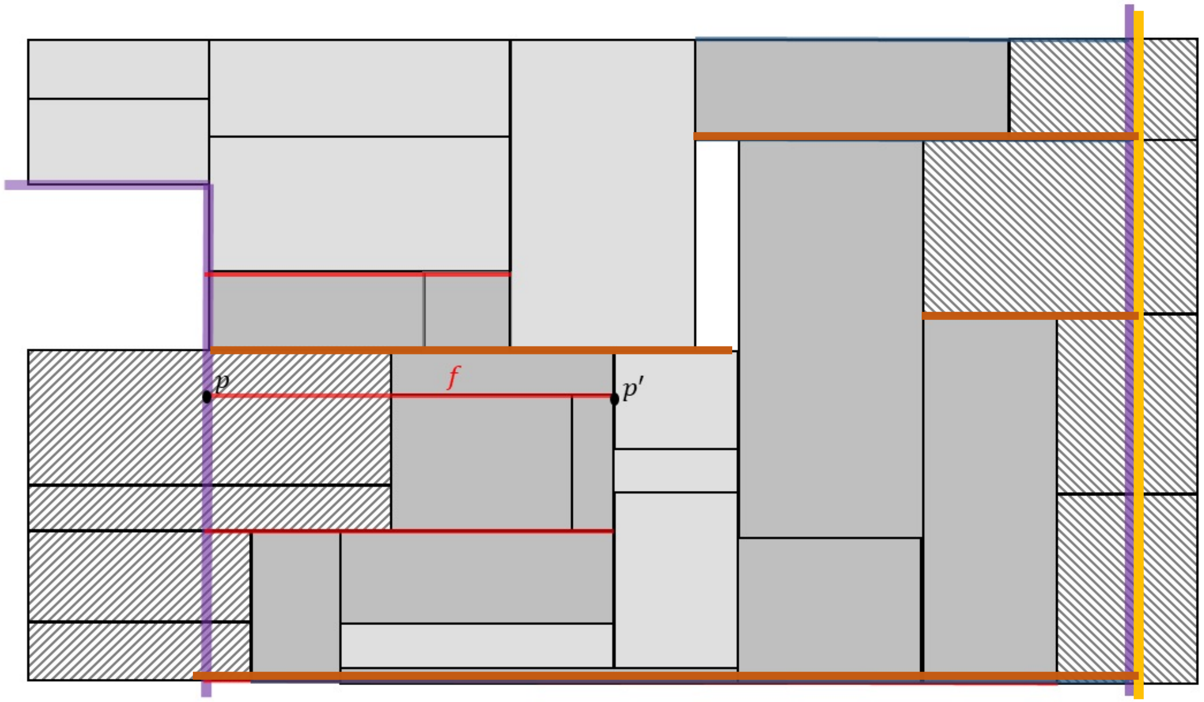
case 1. case 2.a. case 2.b.i. case 2.b.ii.

- Charging argument: For each R that is not horizontally nested and intersected by ℓ, we assign a (fractional) charge of ½ to a corner that R sees on left and ½ to a corner that R sees on right.

Proof of Approximation Guarantee

- Claim 2: R' receives a charge to at least one of its corners $\Rightarrow R'$ is protected.

- Claim 3: Each corner of R' is charged at most once.



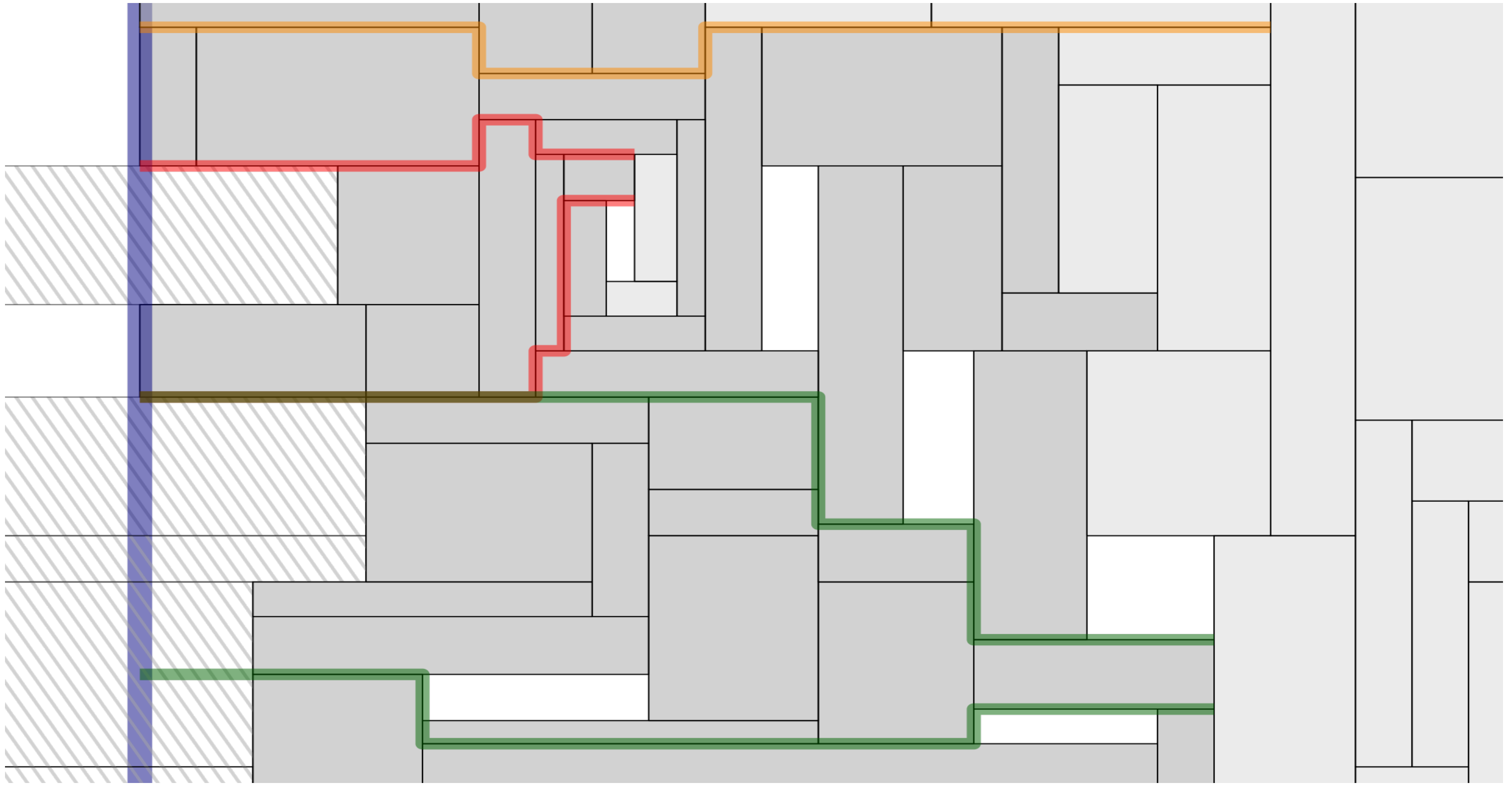
Proof of Approximation Guarantee

- Claim 2: R' receives a charge to at least one of its corners $\Rightarrow$ R' is protected.
- Claim 3: Each corner of R' is charged at most once.
- Lemma: $|\mathcal{R}'| \geq |OPT|/6$.
  - Proof:
  - We lose a factor 2 by removing horizontally nested rectangles and consider only rectangles that are not horizontally nested.
  - Any $R' \in \mathcal{R}$ receives charge at most $\frac{1}{2} \times 4 = 2$ from not horizontally nested rectangles.
  - Let k be the number of not horizontally nested rectangles that are cut by vertical lines. If they save t rectangles in $\mathcal{R}$, then $t \geq \frac{k}{2}$ i.e. $\frac{\mathrm{t}}{\mathrm{k+t}} \geq 1/3$.
  - Total loss $= \frac{1}{2} \times \frac{1}{3} = \frac{1}{6}$.

Proof of Approximation Guarantee

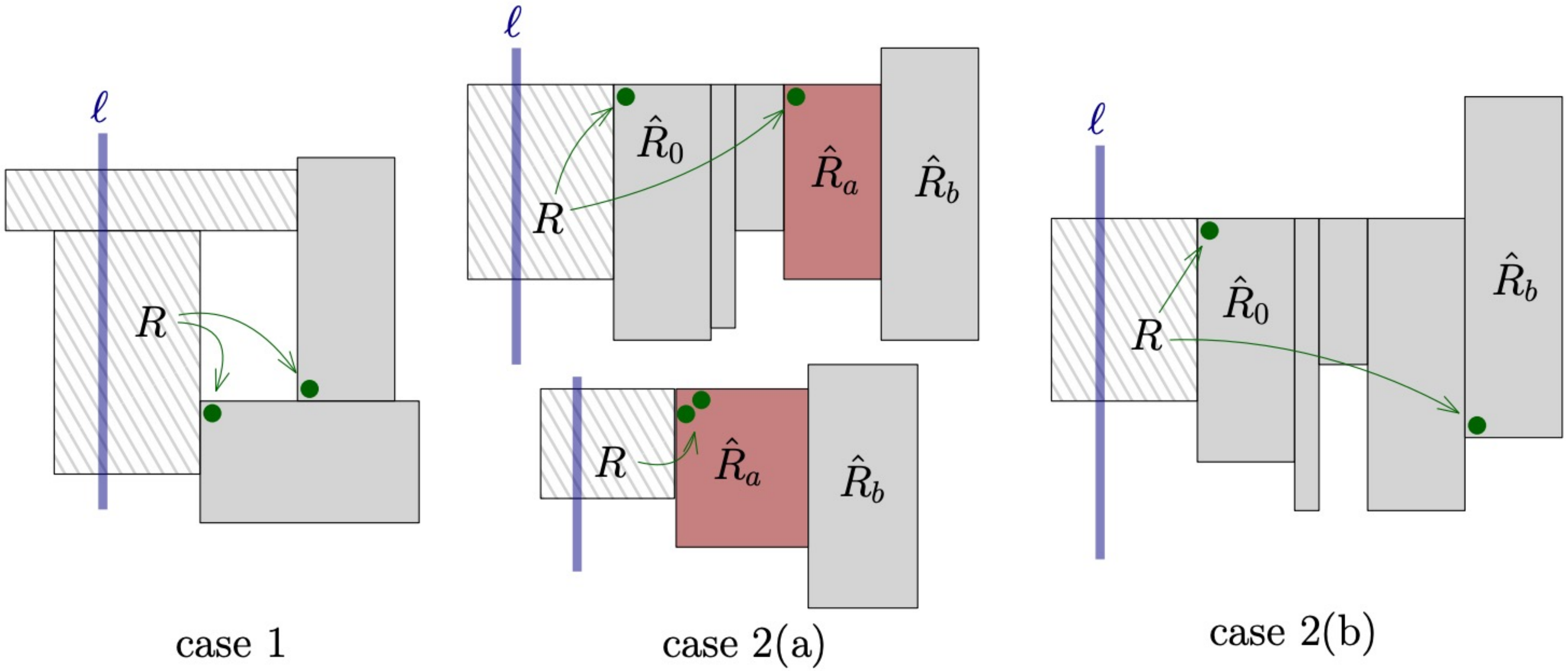
- Claim 2: R' receives a charge to at least one of its corners $\Rightarrow$ R' is protected.
- Claim 3: Only two corners of R' are charged (and at most once).
- Lemma: $|\mathcal{R}'| \geq |OPT|/4.$
  - Proof:
  - We lose a factor 2 by removing horizontally nested rectangles and consider only rectangles that are not horizontally nested.
  - Any $R' \in \mathcal{R}$ receives charge at most $\frac{1}{2} \times 2 = 1$ from not horizontally nested rectangles.
  - Let k be the number of not horizontally nested rectangles that are cut by vertical lines. If they save t rectangles in $\mathcal{R}$, then $t \geq k$ i.e. $\frac{\mathrm{t}}{\mathrm{k+t}} \geq 1/2.$
  - Total loss $= \frac{1}{2} \times \frac{1}{2} = \frac{1}{4}.$

4-approximation



- Fences with more bends (x-monotone curves) ensure corners for only one side get tokens.
- However, bounds for partitioning lemma gets worse: $t = 30\tau + 18$ for τ-fences.

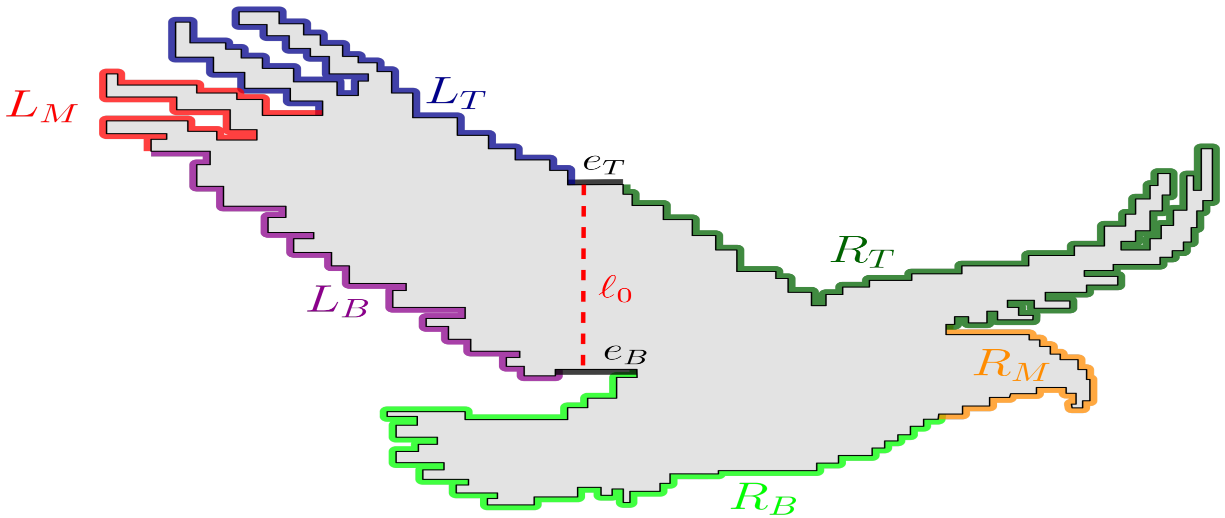
3-approximation

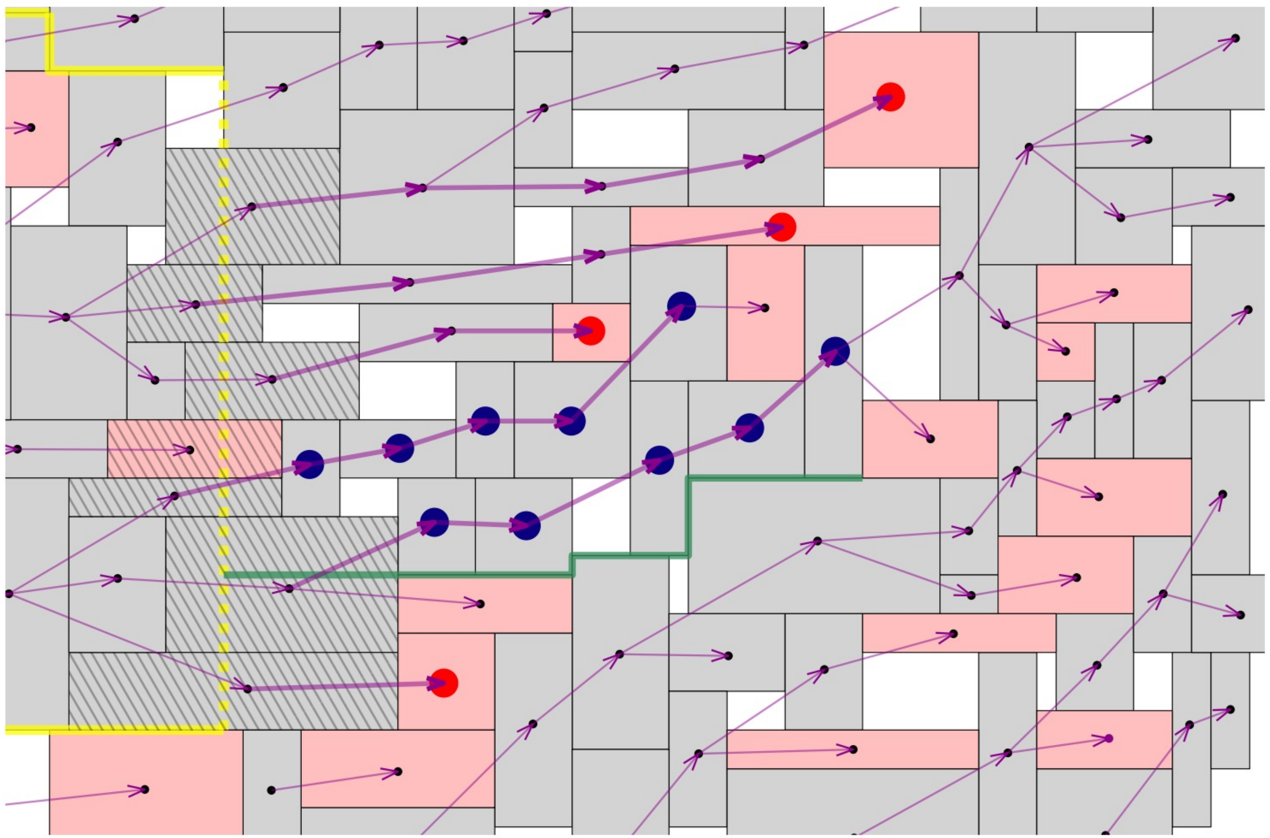


- More sophisticated charging!
- Charge additional corners (may not be seen) or charge hor. nested rectangles.
- Corners of nonhor-nested (resp hor-nested) gets at most 1/4 (resp. 1/2) tokens.

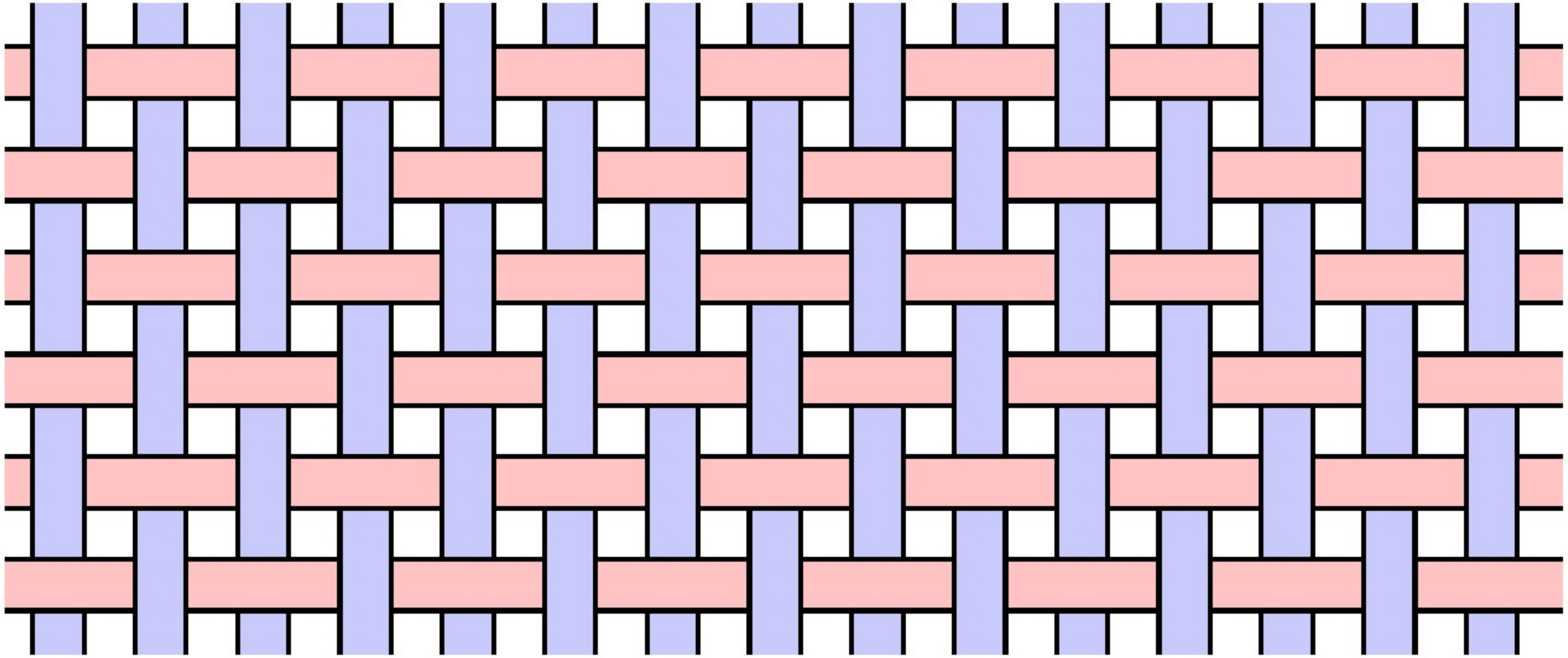
$(2 + \epsilon)$-approximation

- Even more general polygons and even more general fence to accommodate more sophisticated charging argument.





Tight Example



Weighted MISR

- Our techniques don't extend to weighted case.
- Best Approximation: O(log log n) [Chalermsook et al., SODA'21]
- Showed: $\chi\ is\ O(\omega \log \omega)$, where χ is chromatic number and ω is the clique number and using LP rounding.

$$\max \sum_{i \in V} w_i x_i \ : \ \sum_{v \in Q} x_v \leq 1 \ \text{ for every clique } Q \in G,\ 0 \leq x_i \leq 1.$$

- Conjecture: $\chi\ is\ O(\omega)$
- The conjecture, if true, will give a O(1)-approximation.

Open Problems

- PTAS for MISR.
- Obtaining $(2-\epsilon)$-approximation for axis-parallel line segments.
- O(1)-approximation for maximum weighted independent set of rectangles.
- Resolution of $\chi(G) = O\big(\omega(G)\big)$ conjecture for rectangle intersection graphs.
- Resolution of Pach-Tardos conjecture.
- Extension to higher dimensions.
- Obtaining O(1)-approximation (even poly(n)-approx) for arbitrary line segments. [Present best: n^{ϵ} : Fox-Pach SODA'11]

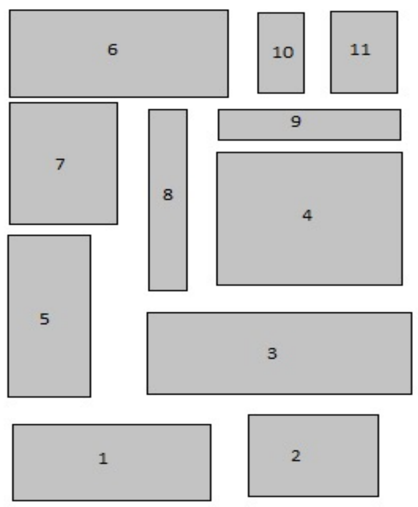
Thank you!

2-D Geometric Bin Packing

- **Given:** Collection of rectangles (by width, height),
- **Goal:** Pack them into minimum number of unit square bins.

- Orthogonal Packing: rectangles packed parallel to bin edges.

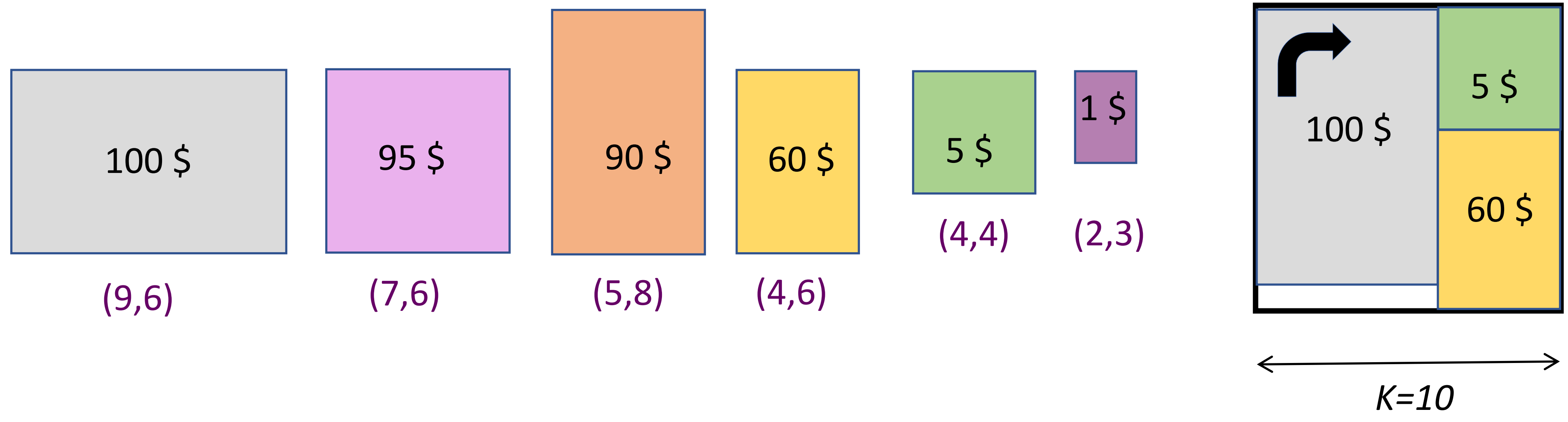
- With 90 degree *rotations* and *without rotations*.



Geometric Knapsack: (2-D)

- Input :
  - Rectangles $I := \{R_1, R_2, \ldots, R_n\}$; Each R_i has integral width and height (w_i, h_i) and profit p_i.
  - A Square $K \times K$ knapsack.

- Goal : Find an axis-parallel non-overlapping packing of a subset of input rectangles into the knapsack that maximizes the total profit.



Variant 2: (2DKR)

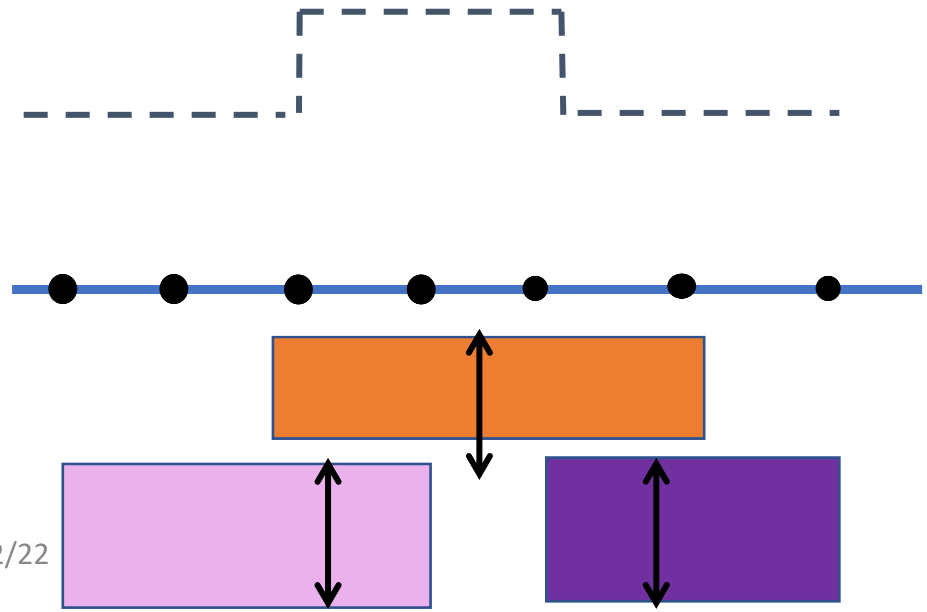
90 degree rotations are allowed!

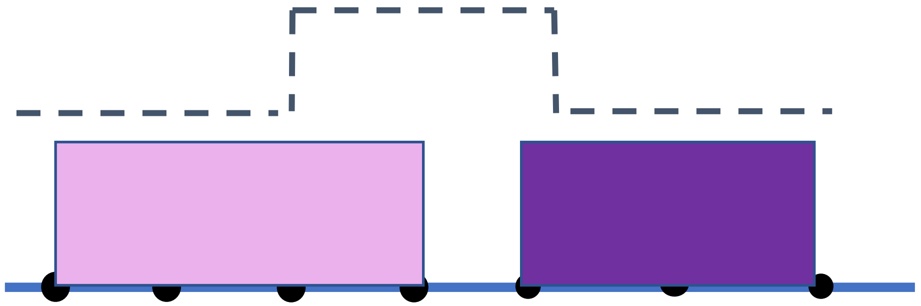
OPT=165

SAP

- Input: A path with edge capacities and a set of tasks (rectangles) that are specified by start and end vertices (fixed starting coordinate and width), demands (heights) and profits.

- Goal: Select a subset of tasks that can be drawn as non-overlapping rectangles underneath the capacity profile.





UFP (sliced version of SAP)

- Input: A path with edge capacities and a set of tasks (rectangles) that are specified by start and end vertices (fixed starting coordinate and width), demands (heights) and profits.

- Goal: Select a subset of tasks such that total demand of selected tasks at any edge is less than the edge capacity.