Maximum Independent Set of Rectangles

Joint work with Orestis Konstantinides as part of the graduate course "Approximation Algorithms"

Alex Angelopoulos

M.P.L.A.

July 3, 2014

Outline

Introduction

The challenge

The algorithm

In the loop, step 1: initial partition

In the loop, step 2: removal of bad rectangles

In the loop, step 3: removal of large cliques

Epilogue

The problem

Input: A set $\mathcal{R}$ of axis-parallel rectangles

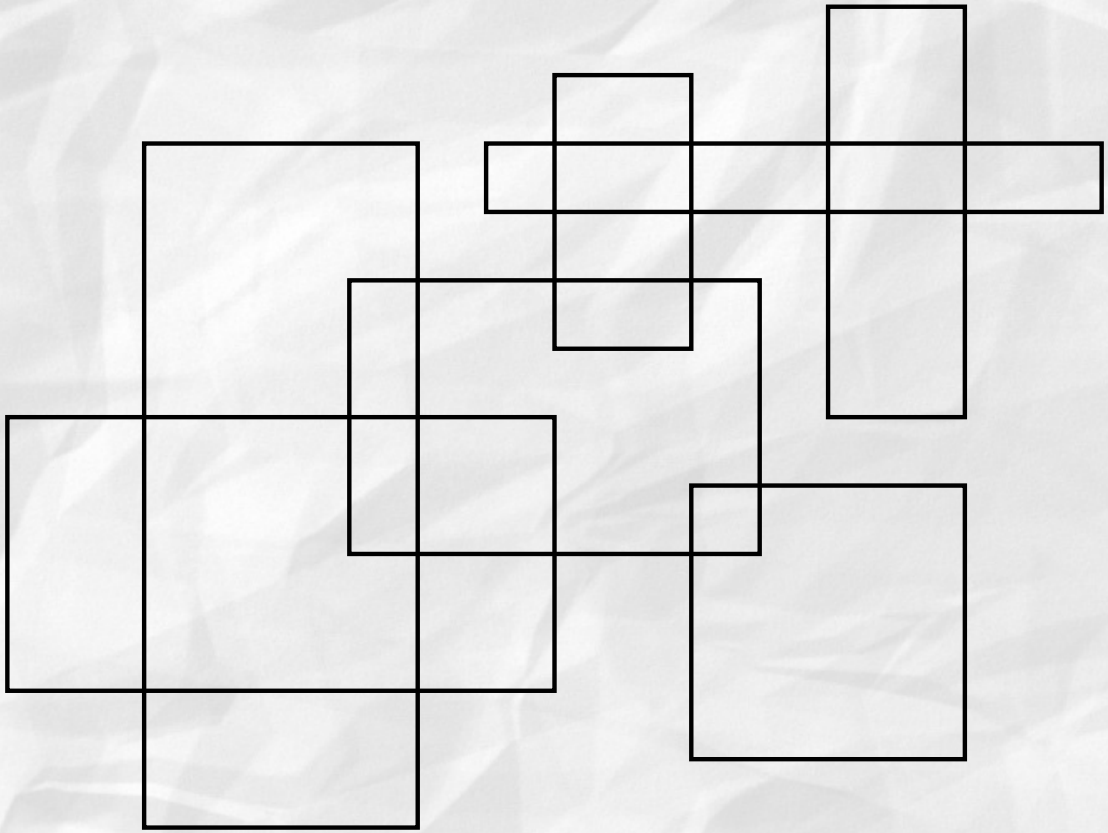


Figure: Instance of MISR

The problem

Input: A set $\mathcal{R}$ of axis-parallel rectangles
Output: The max. independent set of $\mathcal{R}$

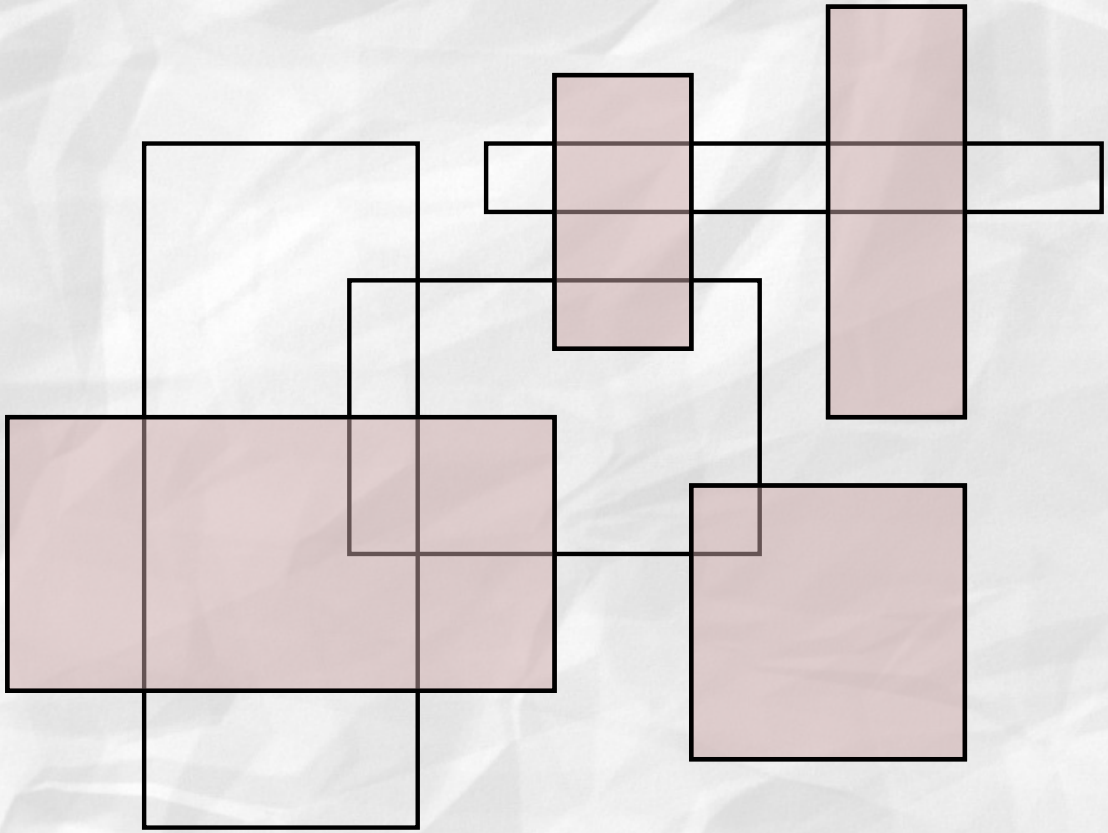


Figure: Instance of `MISR`

Existing results

- Agarwal et al. (1997), Khanna et al. (1998), Nielsen (2000): $\log n$-**approx. for** MISR
  - Agarwal et al. (1997): **PTAS** for rectangles with unit heights (improved running time by Chan (2004)
  - Chan (2003): **PTAS** for squares
  - Erlebach et al. (2005): **PTAS** for squares and rectangles with bounded aspect ratio
  - Lingas and Wahlen (2005): exact, in time $2^{\mathcal{O}\sqrt{n}\log n}$

Plus:

Theorem 1 (Lewin-Eytan et al. (2002)).

There is a parametrized $4q$-approximation for MISR *q being the cardinality of the maximum clique of the instance $\mathcal{R}$.*

Existing results

- Agarwal et al. (1997), Khanna et al. (1998), Nielsen (2000): $\log n$-**approx. for** MISR
- Agarwal et al. (1997): **PTAS** for rectangles with unit heights (improved running time by Chan (2004)
- Chan (2003): **PTAS** for squares
- Erlebach et al. (2005): **PTAS** for squares and rectangles with bounded aspect ratio
- Lingas and Wahlen (2005): exact, in time $2^{\mathcal{O}\sqrt{n}\log n}$

Plus:

Theorem 1 (Lewin-Eytan et al. (2002)).

There is a parametrized $4q$-approximation for MISR *q being the cardinality of the maximum clique of the instance $\mathcal{R}$.*

Existing results

- Agarwal et al. (1997), Khanna et al. (1998), Nielsen (2000): $\log n$-**approx. for** MISR
- Agarwal et al. (1997): **PTAS** for rectangles with unit heights (improved running time by Chan (2004)
- Chan (2003): **PTAS** for squares
- Erlebach et al. (2005): **PTAS** for squares and rectangles with bounded aspect ratio
- Lingas and Wahlen (2005): exact, in time $2^{\mathcal{O}\sqrt{n}\log n}$

Plus:

Theorem 1 (Lewin-Eytan et al. (2002)).

There is a parametrized $4q$-approximation for MISR q *being the cardinality of the maximum clique of the instance $\mathcal{R}$.*

The new result

Theorem (Chalermsook and Chuzhoy, 2009).

There is a randomized $\log \log n$*-approximation algorithm for* MISR

Basic notations:

- Instance: $\mathcal{R}$, a rectangle: R
- For the LP: z_R indicating the **participation** or not of R in the solution

Basic principles:

- Solve LP relaxation and form a canonical solution
- Design an iterative algorithm defining partitions of $\mathcal{R}$

Fun fact:

- MAXIMUM CLIQUE for a set of rectangles is in **P**

The new result

Theorem (Chalermsook and Chuzhoy, 2009).

There is a randomized $\log \log n$*-approximation algorithm for* MISR

Basic notations:

- Instance: $\mathcal{R}$, a rectangle: R
- For the LP: z_R indicating the **participation** or not of R in the solution

Basic principles:

- Solve LP relaxation and form a canonical solution
- Design an iterative algorithm defining partitions of $\mathcal{R}$

Fun fact:

- MAXIMUM CLIQUE for a set of rectangles is in **P**

The new result

Theorem (Chalermsook and Chuzhoy, 2009).

There is a randomized $\log \log n$*-approximation algorithm for* MISR

Basic notations:

- Instance: $\mathcal{R}$, a rectangle: R
- For the LP: z_R indicating the **participation** or not of R in the solution

Basic principles:

- Solve LP relaxation and form a canonical solution
- Design an iterative algorithm defining partitions of $\mathcal{R}$

Fun fact:

- Maximum Clique for a set of rectangles is in P

The new result

Theorem (Chalermsook and Chuzhoy, 2009).

There is a randomized $\log \log n$*-approximation algorithm for* MISR

Basic notations:

- Instance: $\mathcal{R}$, a rectangle: R
- For the LP: z_R indicating the **participation** or not of R in the solution

Basic principles:

- Solve LP relaxation and form a canonical solution
- Design an iterative algorithm defining partitions of $\mathcal{R}$

Fun fact:

- MAXIMUM CLIQUE for a set of rectangles is in **P**

LP (and relaxation)

Our LP is the following:

$$\max \sum_{R \in \mathcal{R}} z_R$$

$$\text{s.t.} \sum_{R : p \in \mathcal{R}} z_R \leq 1, \quad \forall p \in \mathcal{P}$$

$$z_R \geq 1, \quad \forall R \in \mathcal{R}$$

- $\mathcal{P}$ is the set of "interesting" points for our LP
- at most $(2n)^2$ points defined by our rectangles

LP (and relaxation)

Our LP is the following:

$$\max \sum_{R \in \mathcal{R}} z_R$$

$$\text{s.t.} \sum_{R : p \in \mathcal{R}} z_R \leq 1, \quad \forall \boldsymbol{p \in \mathcal{P}}$$

$$z_R \geq 1, \quad \forall R \in \mathcal{R}$$

- $\mathcal{P}$ is the set of "interesting" points for our LP
- at most $(2n)^2$ points defined by our rectangles

"Interesting" points $\mathcal{P}$

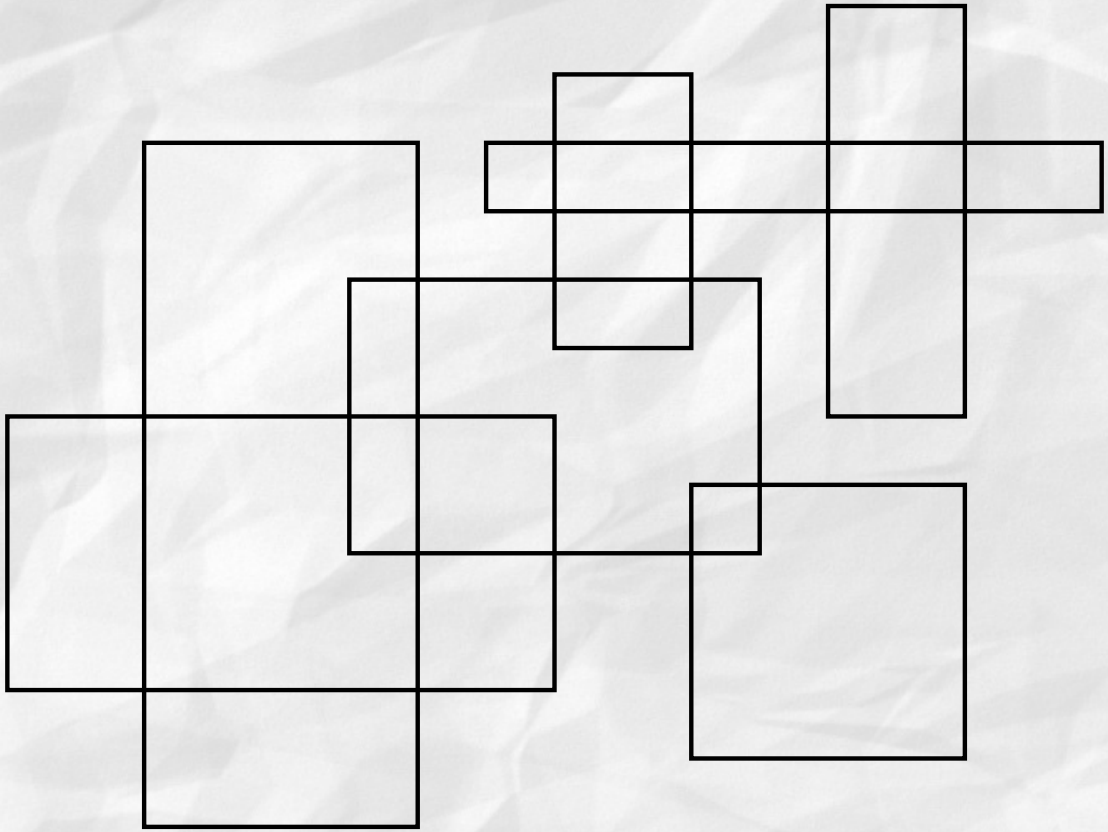


Figure: Set $\mathcal{P}$

"Interesting" points $\mathcal{P}$

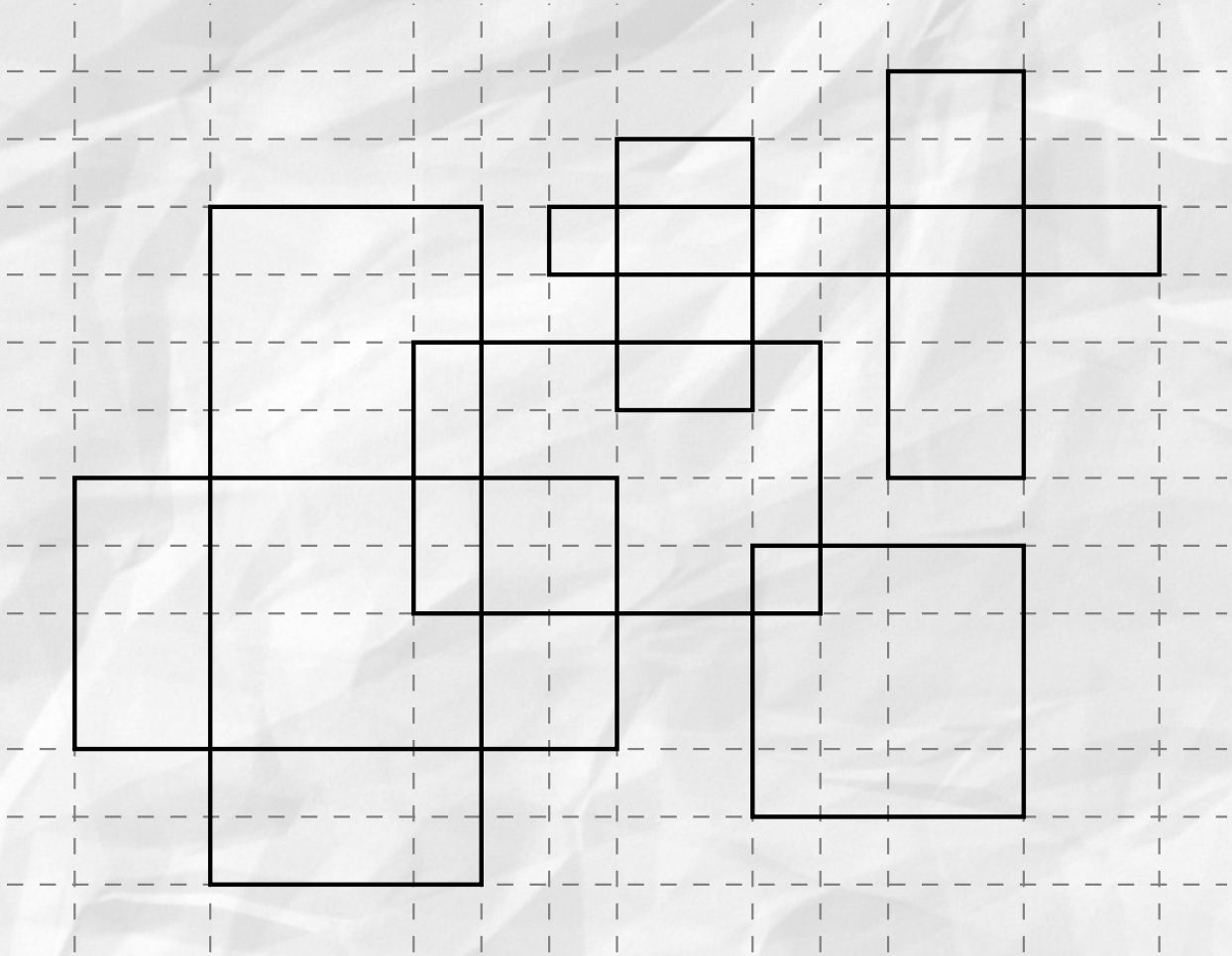


Figure: Set $\mathcal{P}$

"Interesting" points $\mathcal{P}$

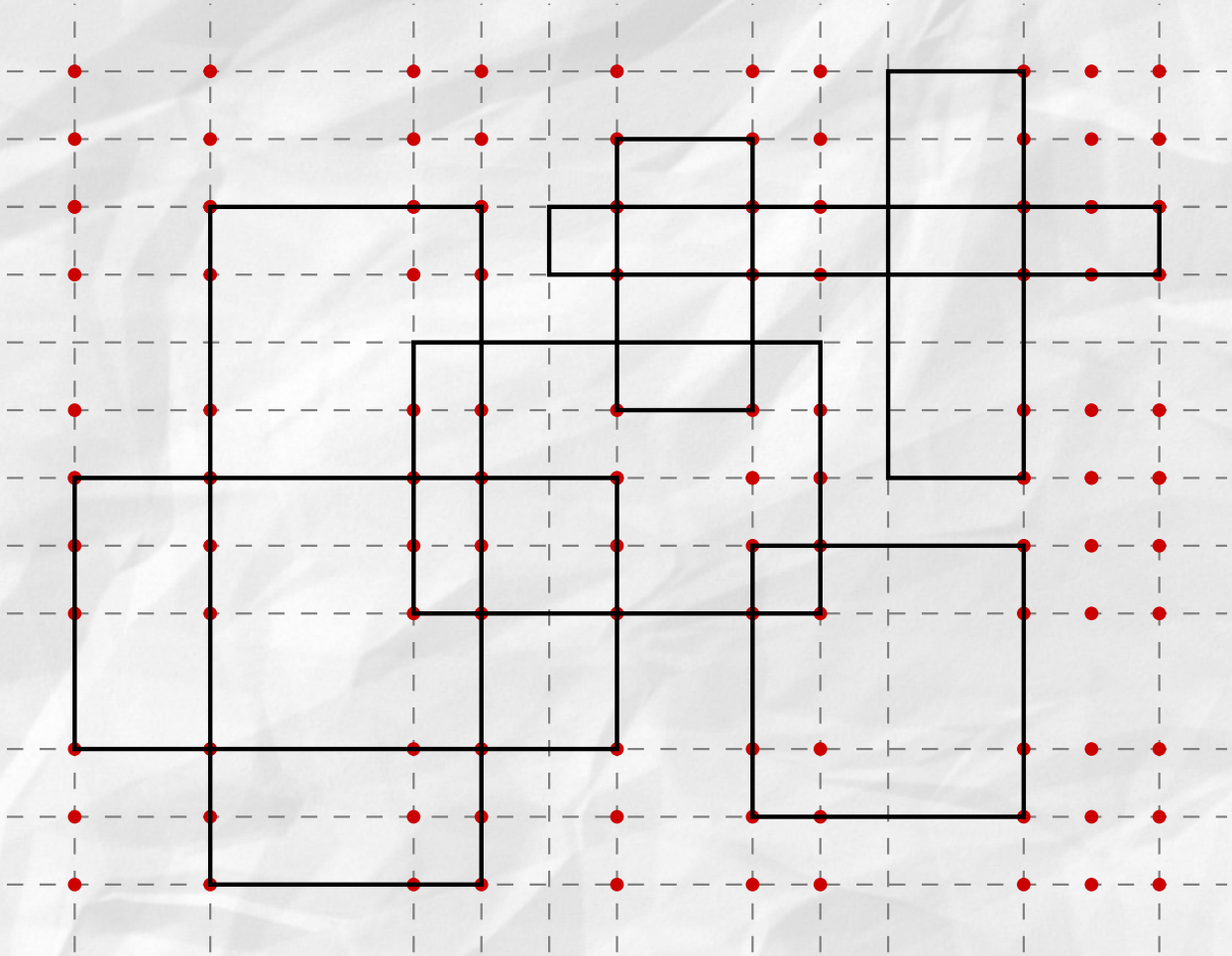


Figure: Set $\mathcal{P}$

The randomization

$$\max \sum_{R \in \mathcal{R}} z_R$$

$$\text{s.t.} \sum_{R: p \in \mathcal{R}} z_R \leq 1, \quad \forall p \in \mathcal{P}$$

$$z_R \geq 1, \quad \forall R \in \mathcal{R}$$

Lemma 2.

There is an efficient randomized algorithm that, given an optimal LP solution of cost OPT for $\mathcal{R}$, produces w.h.p. a feasible canonical solution $\mathcal{R}'$ taking $z_R = \{0, \frac{1}{M}\}$, $M = \Theta(\log n)$ with cost OPT' $\geq c \cdot$ OPT.

Immediate consequences: largest clique in $\mathcal{R}'$ is at most M

Aiming for an algorithm **not in APX**, *we maintain hereon the canonical solution as our instance (denote simply by $\mathcal{R}$).*

The randomization

$$\max \sum_{R \in \mathcal{R}} z_R$$

$$\text{s.t.} \sum_{R: p \in \mathcal{R}} z_R \leq 1, \quad \forall p \in \mathcal{P}$$

$$z_R \geq 1, \quad \forall R \in \mathcal{R}$$

Lemma 2.

There is an efficient randomized algorithm that, given an optimal LP solution of cost OPT for $\mathcal{R}$, produces w.h.p. a feasible canonical solution $\mathcal{R}'$ taking $z_R = \{0, \frac{1}{M}\}$, $M = \Theta(\log n)$ with cost OPT' $\geq c \cdot$ OPT.

Immediate consequences: largest clique in $\mathcal{R}'$ is at most M

Aiming for an algorithm **not in APX**, *we maintain hereon the canonical solution as our instance (denote simply by $\mathcal{R}$).*

The randomization

$$\max \sum_{R \in \mathcal{R}} z_R$$

$$\text{s.t.} \sum_{R: p \in \mathcal{R}} z_R \leq 1, \quad \forall p \in \mathcal{P}$$

$$z_R \geq 1, \quad \forall R \in \mathcal{R}$$

Lemma 2.

There is an efficient randomized algorithm that, given an optimal LP solution of cost OPT for $\mathcal{R}$, produces w.h.p. a feasible canonical solution $\mathcal{R}'$ taking $z_R = \{0, \frac{1}{M}\}$, $M = \boldsymbol{\Theta(\log n)}$ with cost OPT' $\geq c \cdot$ OPT.

Immediate consequences: largest clique in $\mathcal{R}'$ is at most M

Aiming for an algorithm **not in APX**, *we maintain hereon the canonical solution as our instance (denote simply by $\mathcal{R}$).*

Outline

Introduction

The challenge

The algorithm

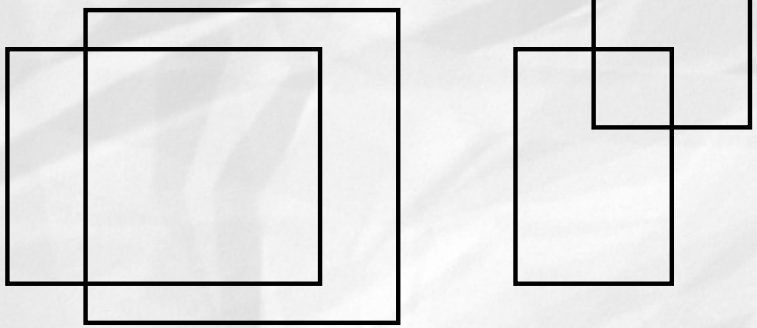
In the loop, step 1: initial partition

In the loop, step 2: removal of bad rectangles

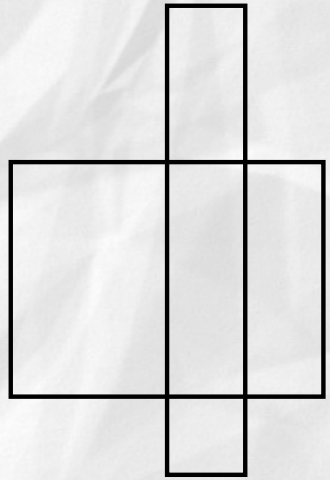
In the loop, step 3: removal of large cliques

Epilogue

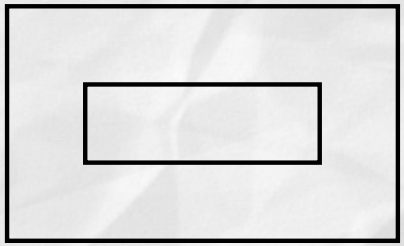
Types of intersection



(a) Corner intersections



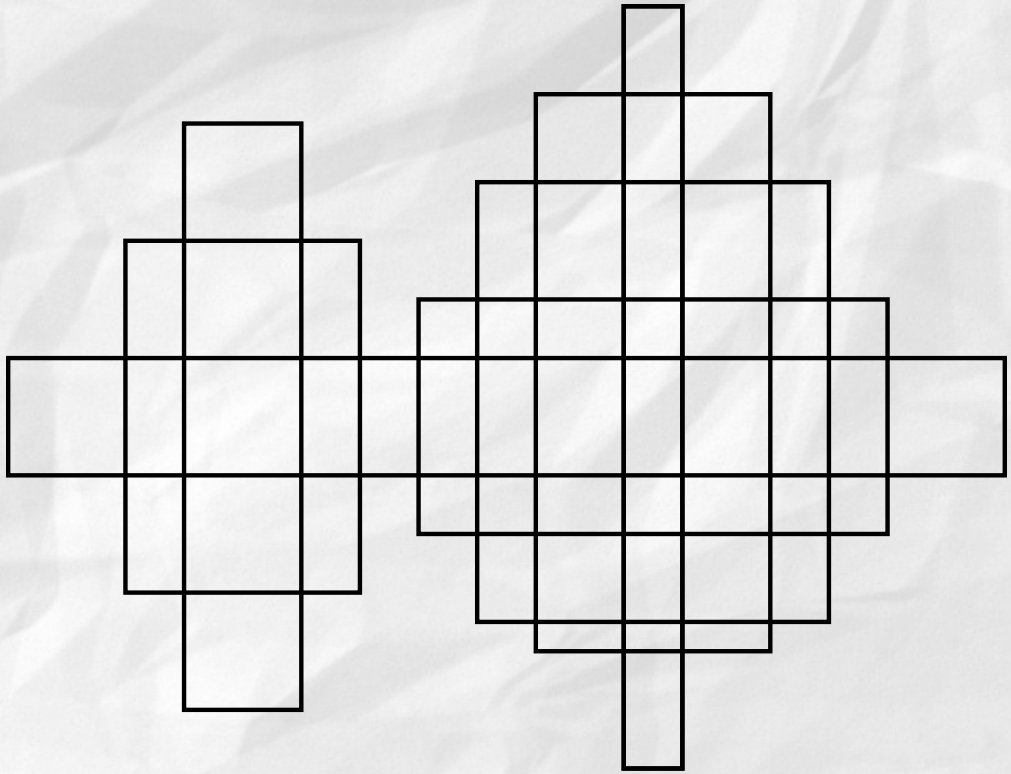
(b) Non-corner intersection



(c) Forget about that! (Why?)

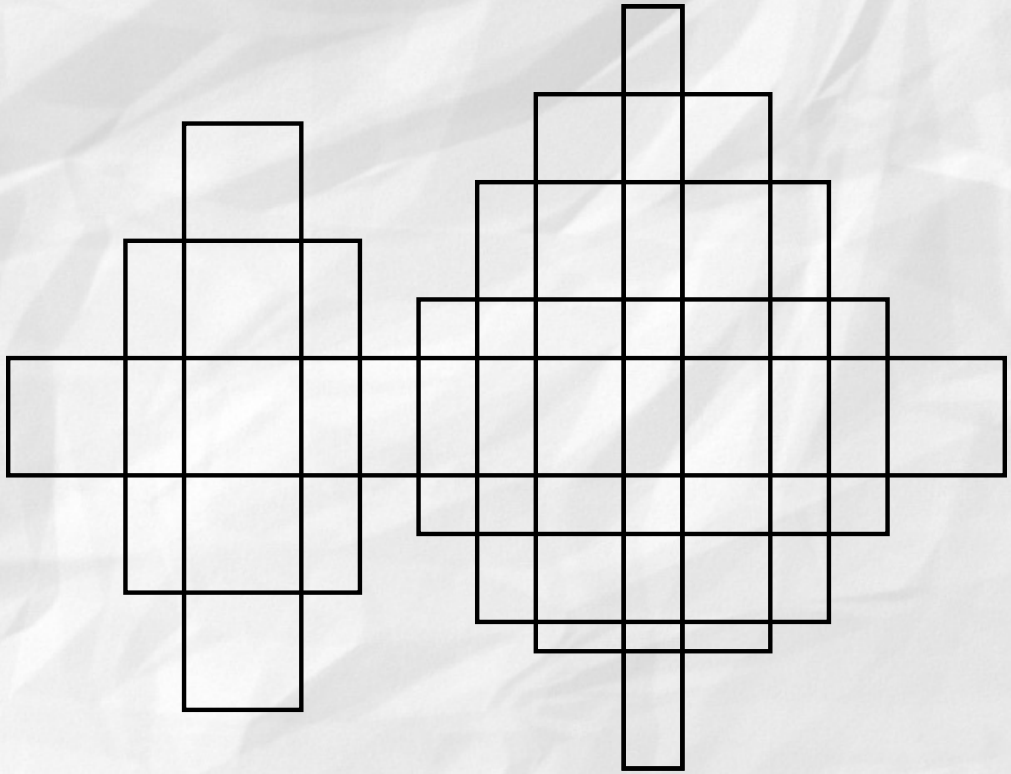
Treating corner and non-corner intersections

If no corner intersections allowed:



Treating corner and non-corner intersections

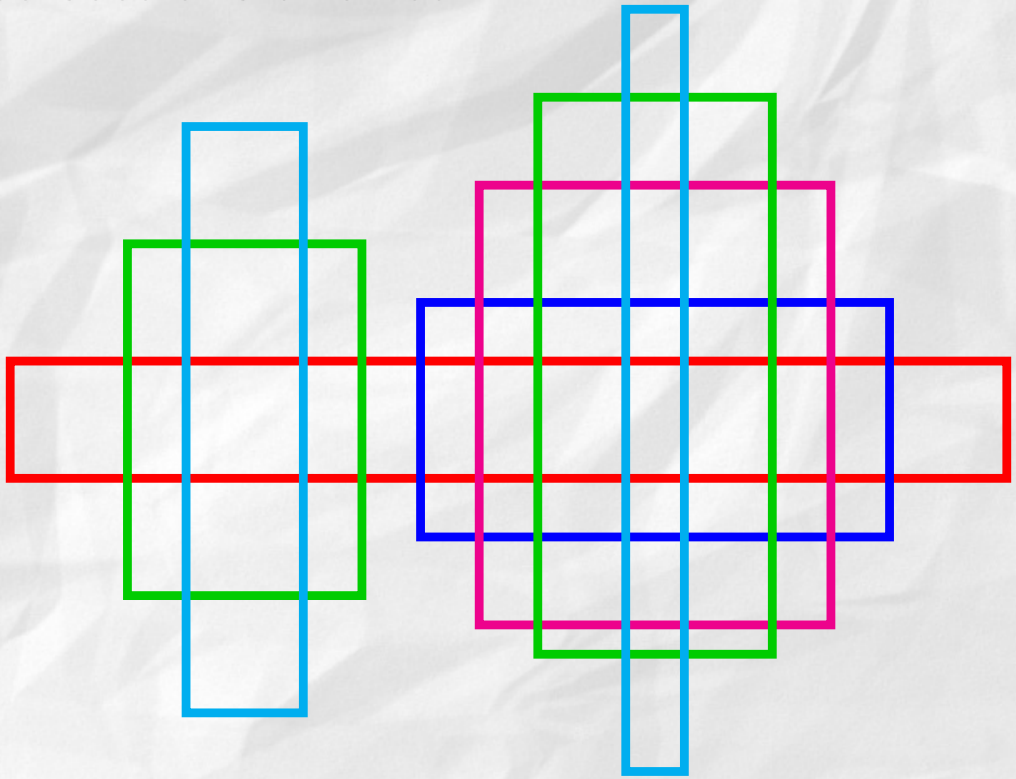
If no corner intersections allowed:



✔ Intersection graph is *perfect* (Lewin-Eytan et al., [8]). (Why?)

Treating corner and non-corner intersections

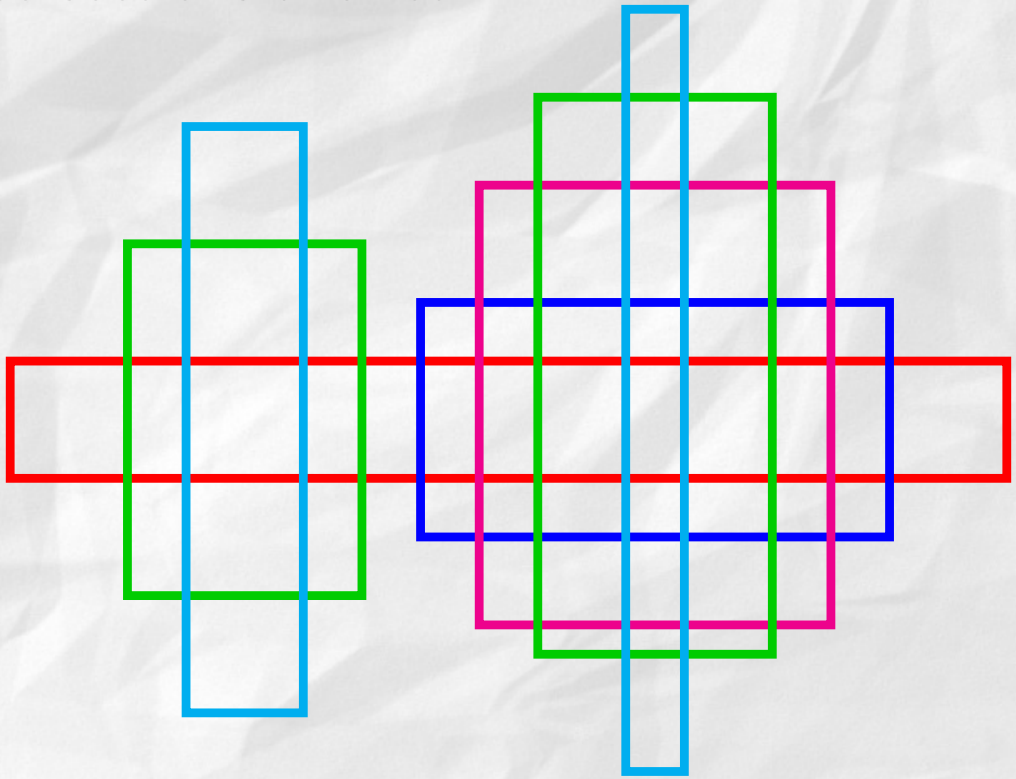
If no corner intersections allowed:



- ✔ Intersection graph is *perfect* (Lewin-Eytan et al., [8]). (Why?)
- ◆ Consider the sets $\boldsymbol{S_j}, 0 \leq j < M$: rectangles R which induce j-size cliques in between rectangles R' that have smaller width

Treating corner and non-corner intersections

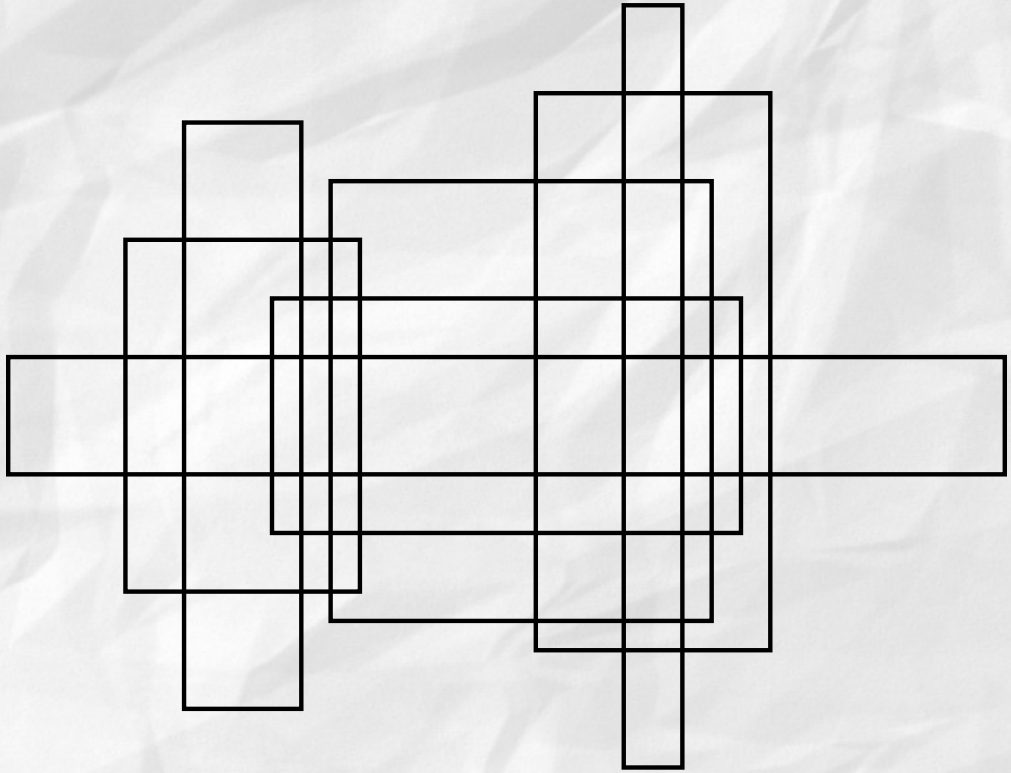
If no corner intersections allowed:



- ✔ Intersection graph is *perfect* (Lewin-Eytan et al., [8]). (Why?)
- ◆ Consider the sets $S_j, 0 \leq j < M$: rectangles R which induce j-size cliques in between rectangles R' that have smaller width
- ✔ **Every S_j is an independent set!**

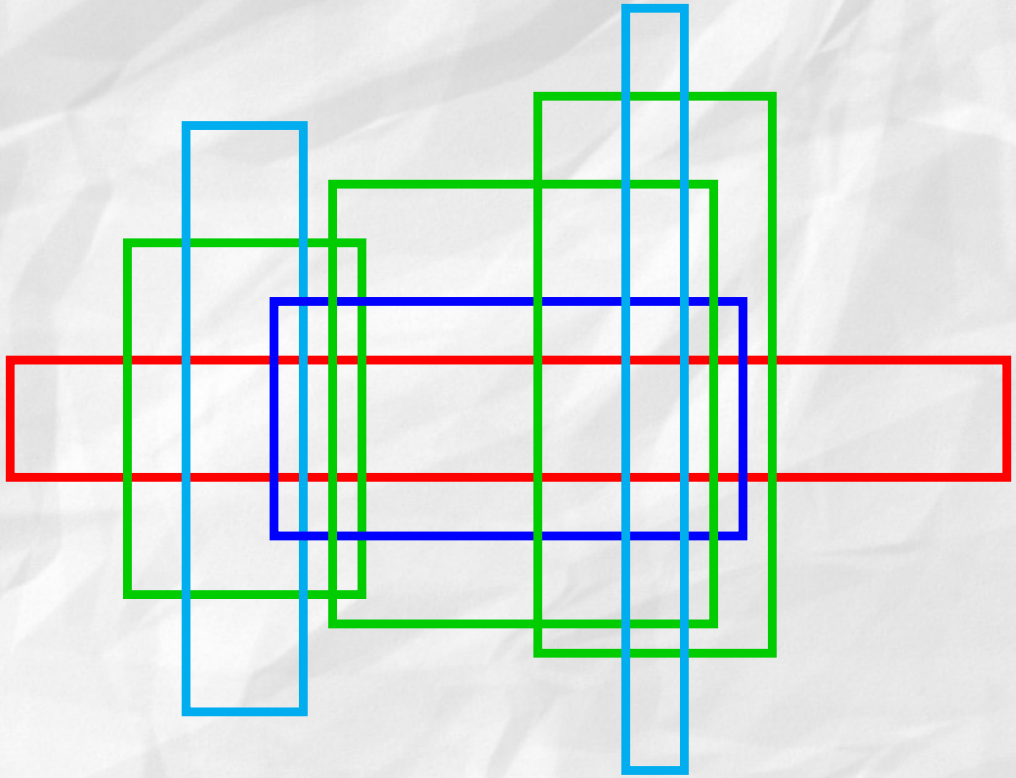
Treating corner and non-corner intersections

What if corner intersections exist?



Treating corner and non-corner intersections

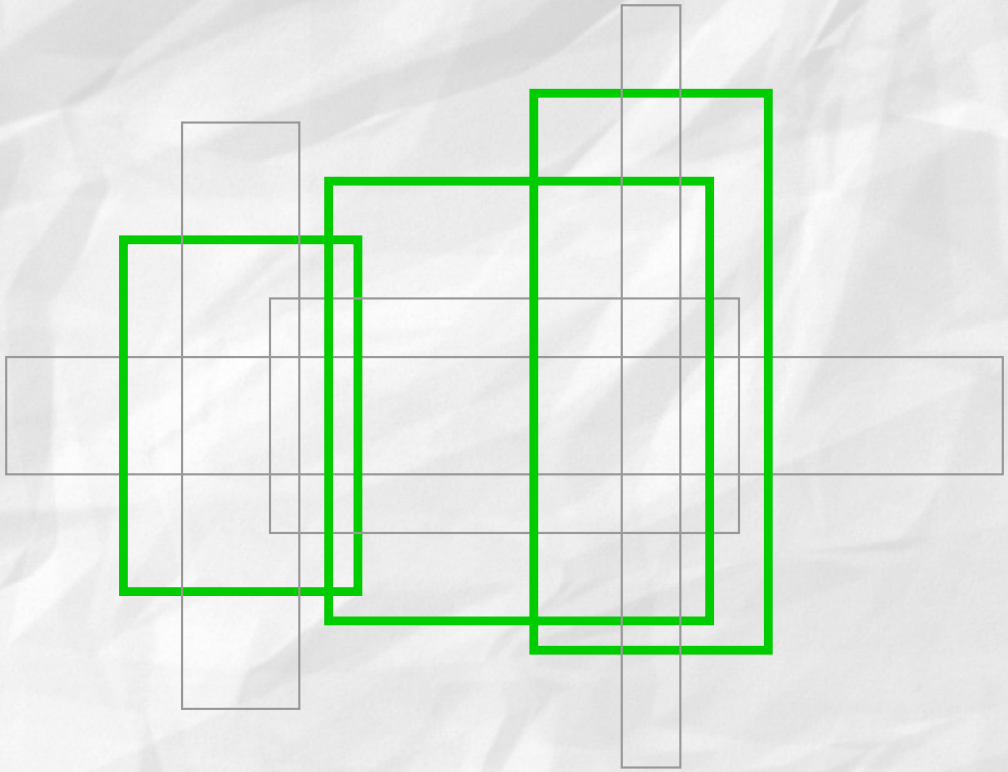
What if corner intersections exist?



- We maintain the idea but: $S_4 = \{R : \text{red}\}$, $S_3 = \emptyset$, $S_2 = \{R : \text{blue}\}$, $S_1 = \{R : \text{green}\}$, $S_0 = \{R : \text{cyan}\}$

Treating corner and non-corner intersections

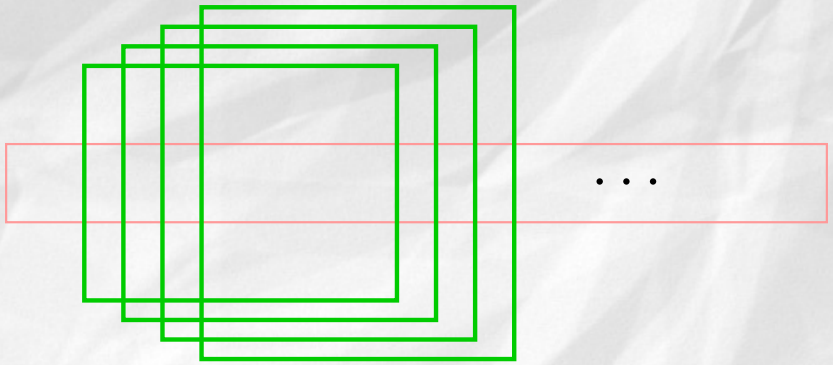
What if corner intersections exist?



- We maintain the idea but: $S_4 = \{R : \text{red}\}$, $S_3 = \emptyset$, $S_2 = \{R : \text{blue}\}$, $S_1 = \{R : \text{green}\}$, $S_0 = \{R : \text{cyan}\}$
- **No guarantee that S_j is an independent set!**

Treating corner and non-corner intersections

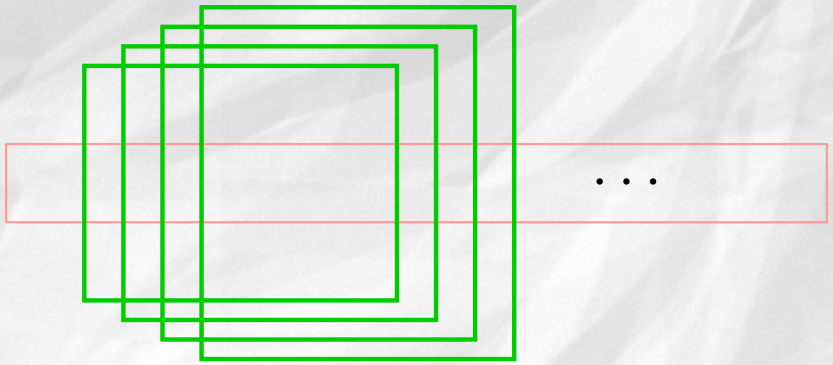
In fact:



- Cliques within S_j can be as large as possible

Treating corner and non-corner intersections

In fact:



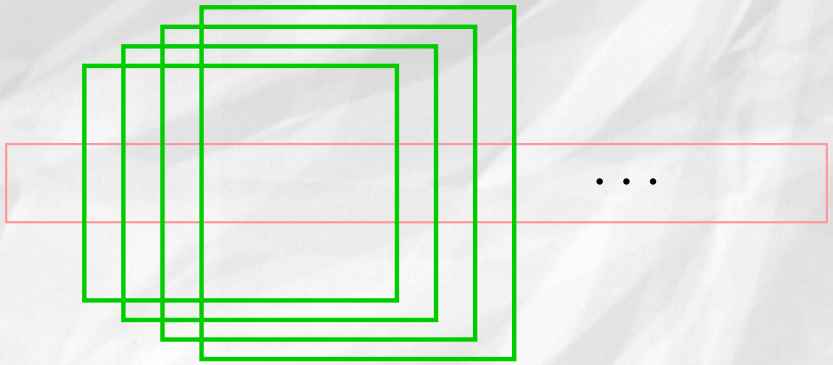
- Cliques within S_j can be as large as possible

Nevertheless:

- for all $R \in \mathcal{R}$, define $\boldsymbol{V(R)}$ to include all R' s.t. $R' \cap R$ is a non-corner intersection and $(w_{R'} < w_R)$

Treating corner and non-corner intersections

In fact:



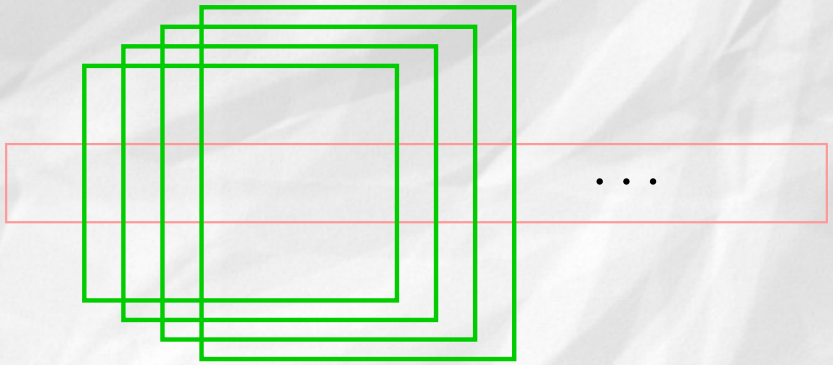
- Cliques within S_j can be as large as possible

Nevertheless:

- for all $R \in \mathcal{R}$, define $\boldsymbol{V(R)}$ to include all R' s.t. $R' \cap R$ is a non-corner intersection and $(w_{R'} < w_R)$
- algorithm will calculate $\boldsymbol{v(R)}$, the size of the maximum clique in $V(R)$

Treating corner and non-corner intersections

In fact:



- Cliques within S_j can be as large as possible

Nevertheless:

- for all $R \in \mathcal{R}$, define $\boldsymbol{V(R)}$ to include all R' s.t. $R' \cap R$ is a non-corner intersection and $(w_{R'} < w_R)$
- algorithm will calculate $\boldsymbol{v(R)}$, the size of the maximum clique in $V(R)$
- and use it to create partitions $\boldsymbol{S_j}$ *according to* $v(R)$

Outline

Introduction

The challenge

The algorithm

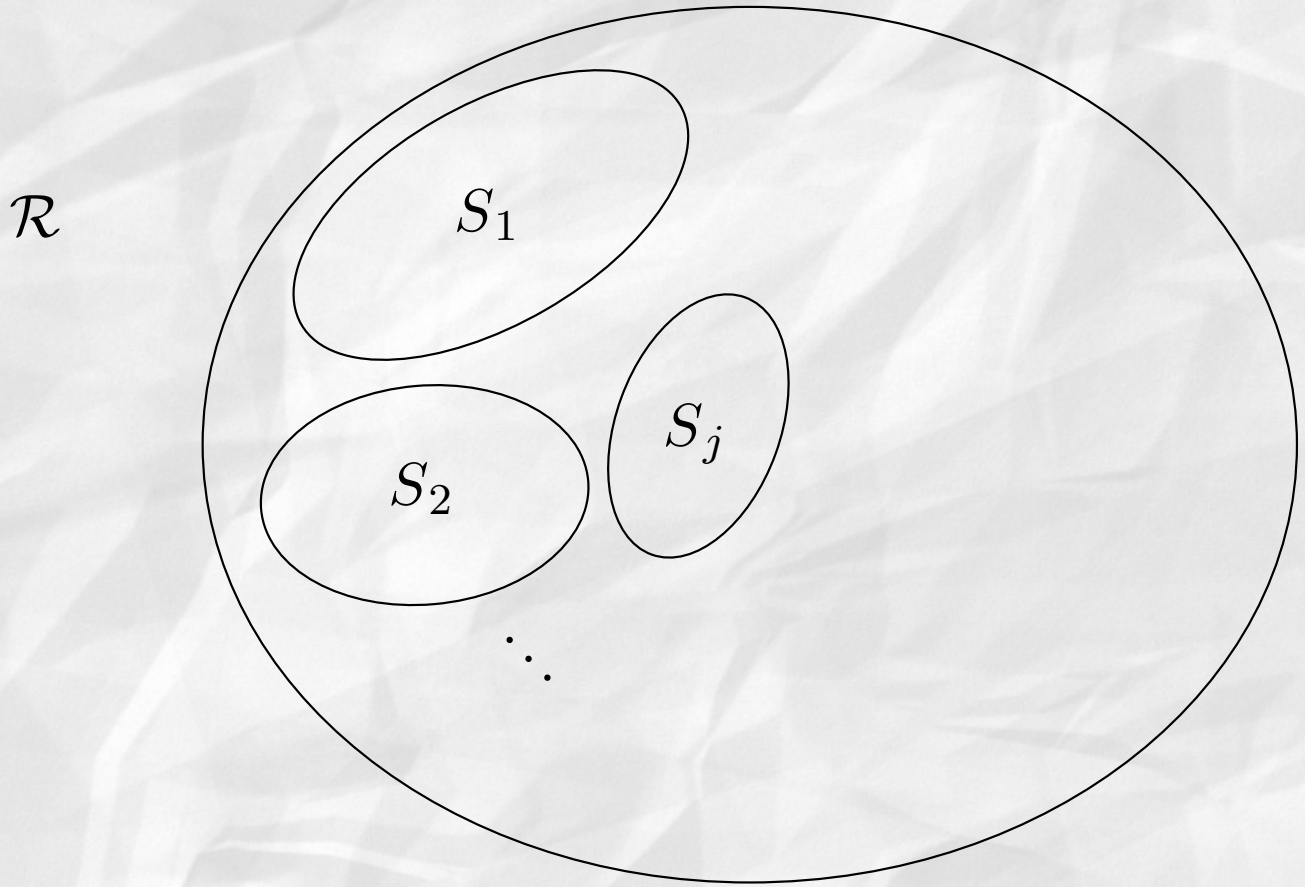
In the loop, step 1: initial partition

In the loop, step 2: removal of bad rectangles

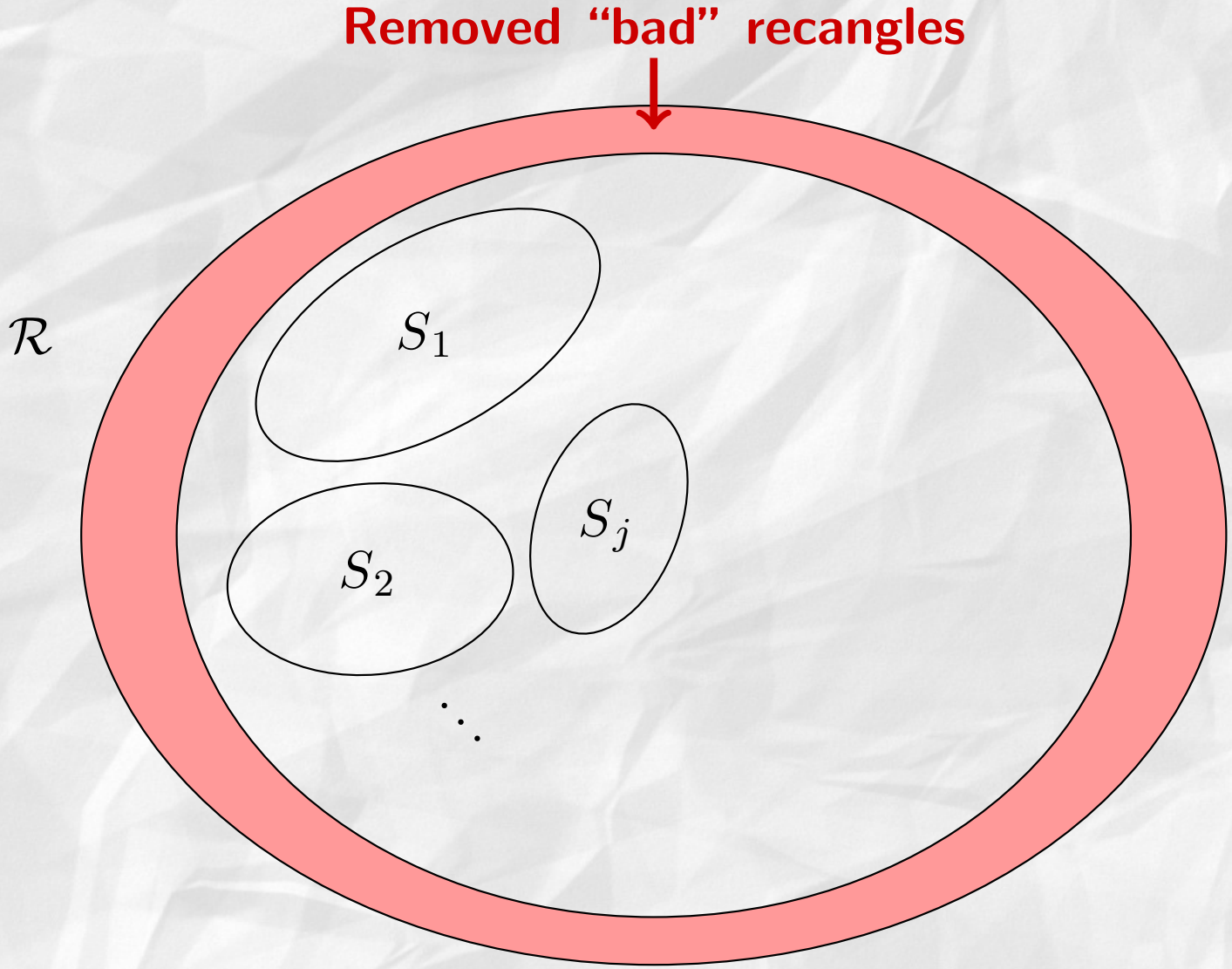
In the loop, step 3: removal of large cliques

Epilogue

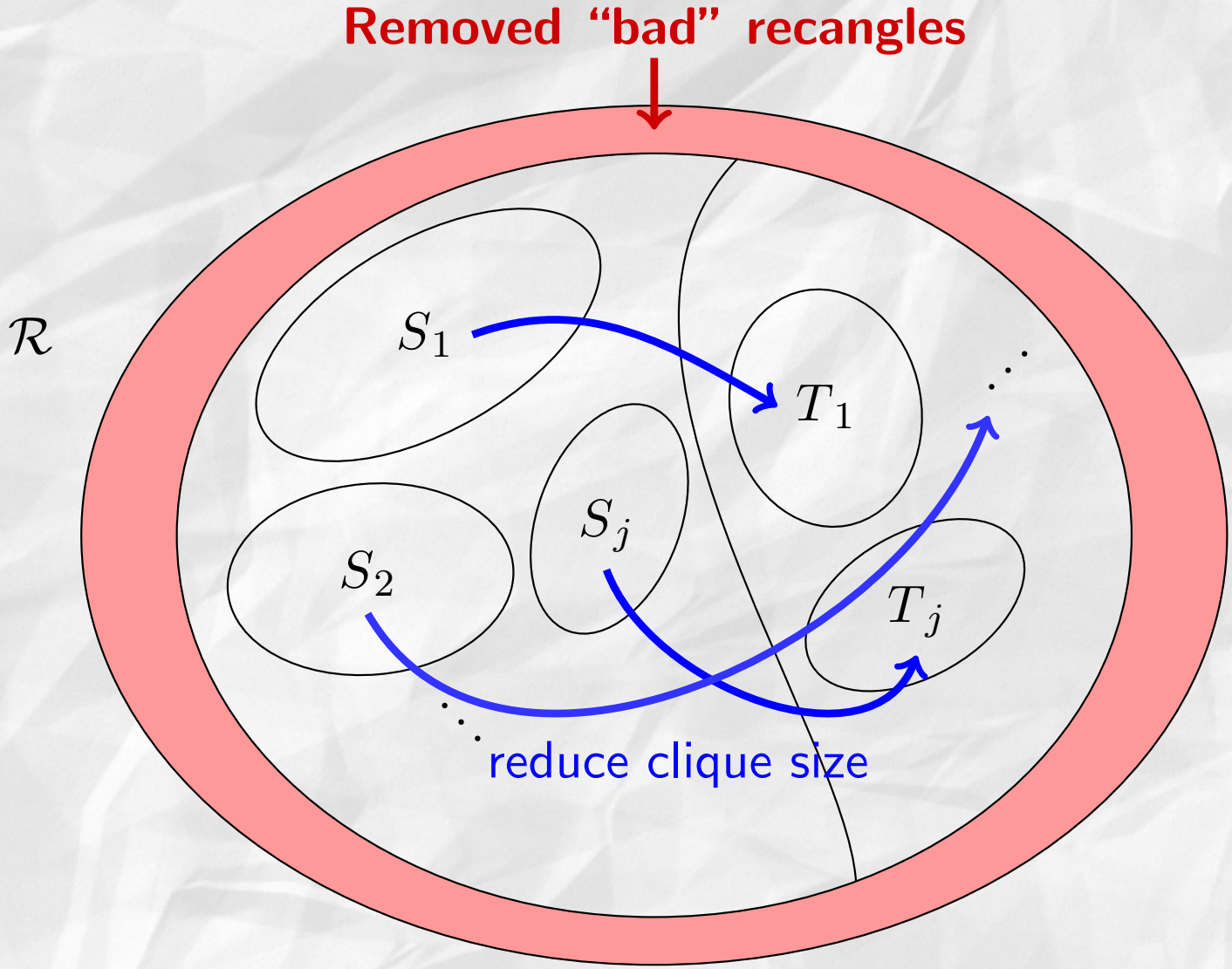
Partition overview



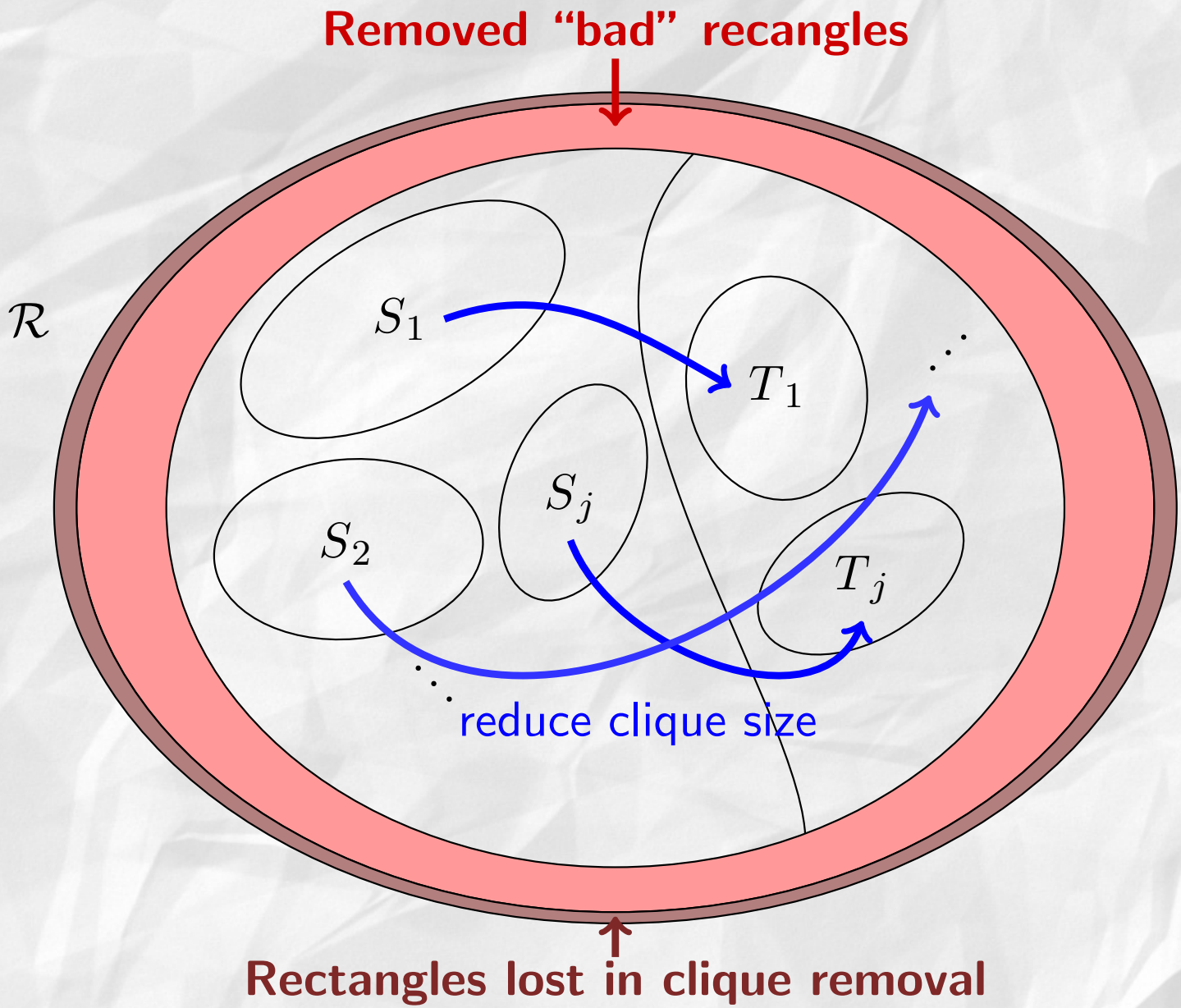
Partition overview



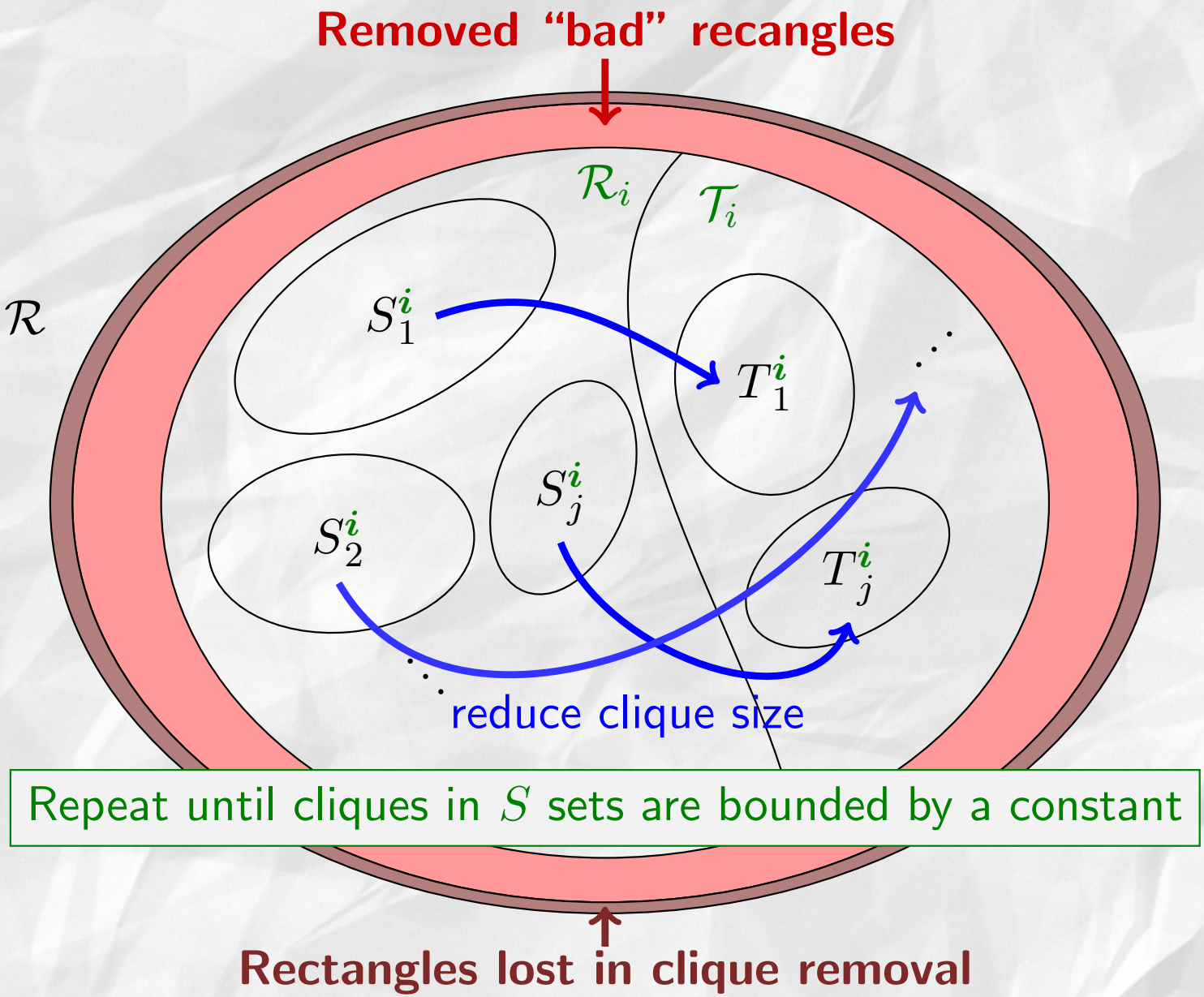
Partition overview



Partition overview



Partition overview



Some more notations

- $\mathcal{R}_i$, $\mathcal{T}_i$, S^i_j, $T^{i \leftarrow \text{iteration}}_{j \leftarrow \text{partition index}}$
- $\bigcup S^i_j = \mathcal{R}_i$, $\bigcup T^i_j = \mathcal{T}_i$
- Every $\mathcal{R}_i$, $\mathcal{T}_i$ is partitioned into $k_i = \beta^i$ subsets ($j \in \{1, ..., k_i\}$)
- $\boldsymbol{W_i}$: total weight of recangles in $\mathcal{R}_i$ **(LP weight, $1/M$ for each)**
- $\boldsymbol{C_i}$: total weight of recangles in $\mathcal{T}_i$ **(charged when moved to some T^i_j)**

Most basic:

$$R \in \boldsymbol{S^i_j} \Leftrightarrow \frac{(j-1)M}{\beta^i} \leq \boldsymbol{v(R)} < \frac{jM}{\beta^i}$$

Algorithm scheme

1. Run LP and create canonical solution $\mathcal{R}$
2. $\mathcal{R}_1 \leftarrow \mathcal{R}$, $\mathcal{T}_1 \leftarrow \emptyset$
3. $i \leftarrow 1$
4. **while** $(\beta^i < M/100)$ **do**
   a. create $\mathcal{R}_{i+1}$, $\mathcal{T}_{i+1}$ so that conditions **C1**–**C4** hold
   b. $i \leftarrow i + 1$
5. find $\max\{\boldsymbol{W_i}, \boldsymbol{C_i}\}$
6. **return** recovered **IS**

Algorithm scheme

1. Run LP and create canonical solution $\mathcal{R}$ [✔]
2. $\mathcal{R}_1 \leftarrow \mathcal{R}$, $\mathcal{T}_1 \leftarrow \emptyset$
3. $i \leftarrow 1$
4. **while** $(\beta^i < M/100)$ **do**
   a. create $\mathcal{R}_{i+1}$, $\mathcal{T}_{i+1}$ so that conditions **C1**–**C4** hold
   b. $i \leftarrow i + 1$
5. **find** $\max\{W_i, C_i\}$
6. **return recovered IS**

Algorithm's iteration invariants

Take $M = 64 \log n$, $\beta = 20$. Then if the following conditions hold, Proposition 3 is true.

C1. Within T_j^i rectangles do not intersect; an $R \in T_j^i$ does not intersect any $R' \in S_j^i \cup S_{j-1}^i$

C2. Cost of every rectangle in $\mathcal{T}_i$ is $c_R \leq 10^6 \log\log n / \beta^i$

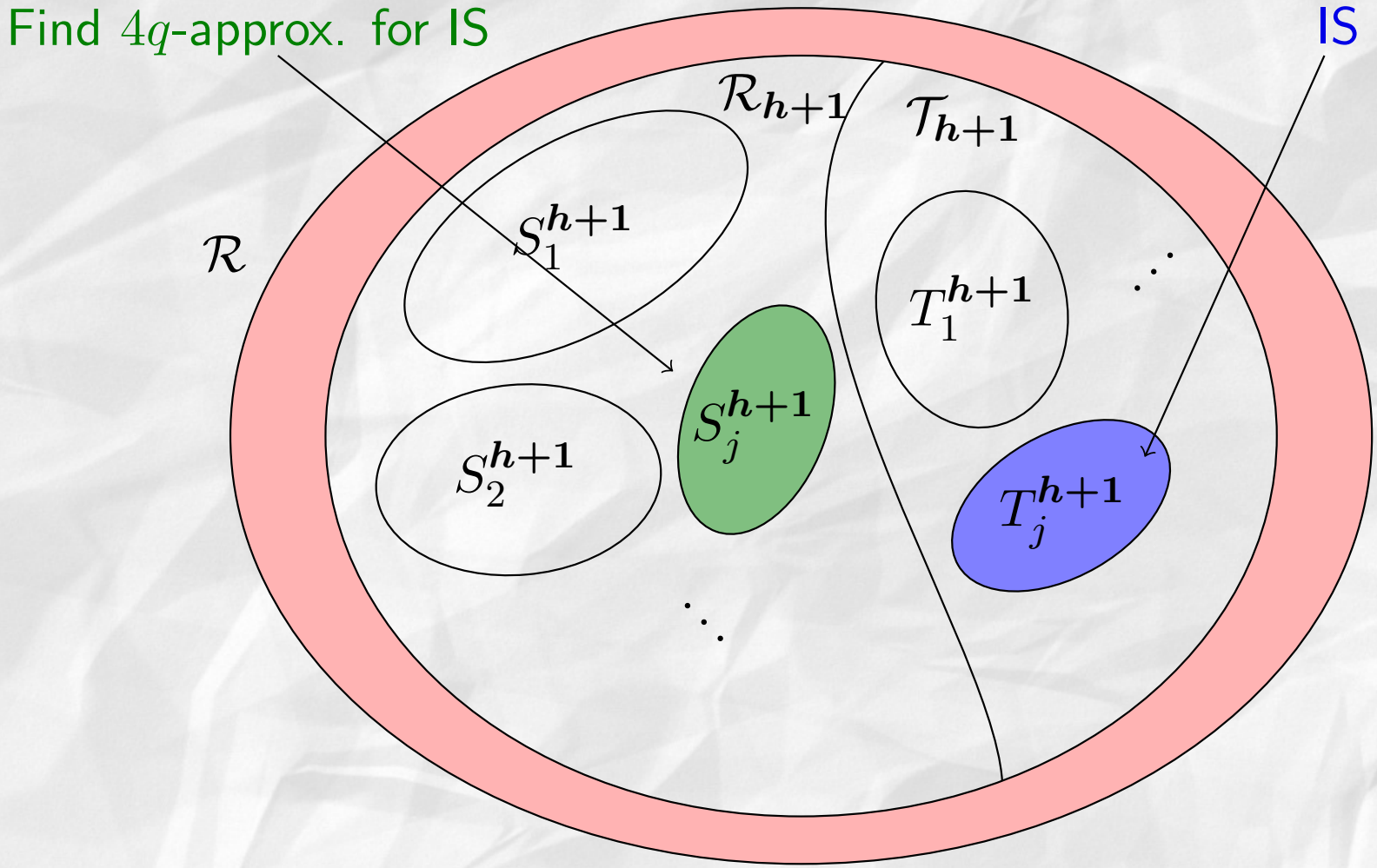
C3. $W_i + C_i \geq OPT(1 - \frac{1}{\log\log n})^i$

C4. The maximum clique of S_j^i is at most $100M/\beta^i$

Proposition 3.

There is an $h = \Theta(\log\log n)$ *such that the output* $\mathcal{R}_{h+1}, \mathcal{T}_{h+1}$ *quarantees the* $\log\log n$ *approximation for* `MISR`.

Partition overview



Algorithm's output is the desired

- **Stop when $\beta^h \geq M/100 \Rightarrow h+1 = \Theta(\log\log n)$**
- Take **C3**: $W_{h+1} + C_{h+1} \geq OPT(1 - \frac{1}{\log\log n})^{\Theta(\log\log n)} \geq \frac{OPT}{e}$
- That means $W_{h+1} + C_{h+1} = \boldsymbol{\Omega(OPT)}$, or, in other words, **not both W_{h+1} and C_{h+1} are asymptotically smaller than OPT**
- Case $W_{h+1} = \Omega(OPT)$:
  - $|\mathcal{R}_{h+1}| = \Omega(M \cdot OPT)$ (uniform rectangles' weight $1/M$)
  - pigeonhole principle: **there is a set** S_j^{h+1} with $\Omega(\frac{M \cdot OPT}{M/100}) = \boldsymbol{\Omega(OPT)}$ rectangles and
    - **bounded size clique** (**C4**)
    - apply Theorem 1 ($4q$-approx. where q is the largest clique)

...This is the good case, giving a constant factor approximation...

Algorithm's output is the desired

- **Stop when** $\beta^h \geq M/100 \Rightarrow h+1 = \Theta(\log\log n)$
- Take **C3**: $W_{h+1} + C_{h+1} \geq OPT(1 - \frac{1}{\log\log n})^{\Theta(\log\log n)} \geq \frac{OPT}{e}$
- That means $W_{h+1} + C_{h+1} = \boldsymbol{\Omega(OPT)}$, or, in other words, **not both** W_{h+1} **and** C_{h+1} **are asymptotically smaller than** OPT
- **Case** $W_{h+1} = \Omega(OPT)$**:**
  - $|\mathcal{R}_{h+1}| = \boldsymbol{\Omega(M \cdot OPT)}$ (uniform rectangles' weight $1/M$)
  - pigeonhole principle: **there is a set** S_j^{h+1} with $\Omega(\frac{M \cdot OPT}{M/100}) = \boldsymbol{\Omega(OPT)}$ rectangles and
  - **bounded size clique** (**C4**)
  - apply Theorem 1 ($4q$-approx. where q is the largest clique)

...This is the good case, giving a constant factor approximation...

Algorithm's output is the desired

- **Stop when** $\beta^h \geq M/100 \Rightarrow h+1 = \Theta(\log\log n)$
- Take **C3**: $W_{h+1} + C_{h+1} \geq OPT(1 - \frac{1}{\log\log n})^{\Theta(\log\log n)} \geq \frac{OPT}{e}$
- That means $W_{h+1} + C_{h+1} = \boldsymbol{\Omega(OPT)}$, or, in other words, **not both** W_{h+1} **and** C_{h+1} **are asymptotically smaller than** OPT
- **Case** $W_{h+1} = \Omega(OPT)$**:**
  - $|\mathcal{R}_{h+1}| = \boldsymbol{\Omega(M \cdot OPT)}$ (uniform rectangles' weight $1/M$)
  - pigeonhole principle: **there is a set** S_j^{h+1} with $\Omega(\frac{M \cdot OPT}{M/100}) = \boldsymbol{\Omega(OPT)}$ rectangles and
  - **bounded size clique** (**C4**)
  - apply Theorem 1 ($4q$-approx. where q is the largest clique)

...This is the good case, giving a constant factor approximation...

Algorithm's output is the desired

- **Stop when** $\beta^h \geq M/100 \Rightarrow h+1 = \Theta(\log\log n)$
- Take **C3**: $W_{h+1} + C_{h+1} \geq OPT(1 - \frac{1}{\log\log n})^{\Theta(\log\log n)} \geq \frac{OPT}{e}$
- That means $W_{h+1} + C_{h+1} = \boldsymbol{\Omega(OPT)}$, or, in other words, **not both** W_{h+1} **and** C_{h+1} **are asymptotically smaller than** OPT
- **Case** $W_{h+1} = \Omega(OPT)$**:**
  - $|\mathcal{R}_{h+1}| = \boldsymbol{\Omega(M \cdot OPT)}$ (uniform rectangles' weight $1/M$)
  - pigeonhole principle: **there is a set** S_j^{h+1} with $\Omega(\frac{M \cdot OPT}{M/100}) = \boldsymbol{\Omega(OPT)}$ rectangles and
  - **bounded size clique** (**C4**)
  - apply Theorem 1 ($4q$-approx. where q is the largest clique)

...This is the good case, giving a constant factor approximation...

Algorithm's output is the desired

- **Case $C_{h+1} = \Omega(OPT)$:**
  - pigeonhole principle: at least one T_j^{h+1} has charge $\Omega(\frac{OPT}{M/100}) = \boldsymbol{\Omega(\frac{OPT}{M})}$
  - this very T_j^{h+1} forms an independent set (C1) and
  - by (C2): $|T_j^{h+1}| \geq \frac{\Omega(\frac{OPT}{M})}{10^6 \log\log n/\beta^{h+1}} = \boldsymbol{\Omega(\frac{OPT}{\log\log n})}$

...This is the worst case, giving the $\log\log n$-approximation.

Algorithm's output is the desired

- **Case $C_{h+1} = \Omega(OPT)$:**
  - pigeonhole principle: at least one T_j^{h+1} has charge $\Omega(\frac{OPT}{M/100}) = \boldsymbol{\Omega(\frac{OPT}{M})}$
  - this very T_j^{h+1} forms an independent set (**C1**) and
  - by (**C2**): $|T_j^{h+1}| \geq \frac{\Omega(\frac{OPT}{M})}{10^6 \log\log n/\beta^{h+1}} = \Omega(\frac{OPT}{\log\log n})$

...This is the worst case, giving the $\log\log n$*-approximation.*

Algorithm's output is the desired

- **Case $C_{h+1} = \Omega(OPT)$:**
  - pigeonhole principle: at least one T_j^{h+1} has charge $\Omega(\frac{OPT}{M/100}) = \boldsymbol{\Omega(\frac{OPT}{M})}$
  - this very T_j^{h+1} forms an independent set (**C1**) and
  - by (**C2**): $|T_j^{h+1}| \geq \frac{\Omega(\frac{OPT}{M})}{10^6 \log\log n/\beta^{h+1}} = \boldsymbol{\Omega(\frac{OPT}{\log\log n})}$

...This is the worst case, giving the $\log\log n$-approximation.

Algorithm's output is the desired

- **Case $C_{h+1} = \Omega(OPT)$:**
  - pigeonhole principle: at least one T_j^{h+1} has charge $\Omega(\frac{OPT}{M/100}) = \boldsymbol{\Omega(\frac{OPT}{M})}$
  - this very T_j^{h+1} forms an independent set (**C1**) and
  - by (**C2**): $|T_j^{h+1}| \geq \frac{\Omega(\frac{OPT}{M})}{10^6 \log\log n/\beta^{h+1}} = \boldsymbol{\Omega(\frac{OPT}{\log\log n})}$

...This is the worst case, giving the $\boldsymbol{\log\log n}$*-approximation.*

Algorithm scheme

1. Run LP and create canonical solution $\mathcal{R}$
2. $\mathcal{R}_1 \leftarrow \mathcal{R}$, $\mathcal{T}_1 \leftarrow \emptyset$
3. $i \leftarrow 1$
4. **while** ($\beta^i < M/100$) **do**
   a. **create** $\mathcal{R}_{i+1}$, $\mathcal{T}_{i+1}$ **so that conditions C1–C4 hold**
   b. $i \leftarrow i + 1$
5. find $\max\{\boldsymbol{W_i}, \boldsymbol{C_i}\}$
6. **return** recovered **IS**

Outline

Introduction

The challenge

The algorithm

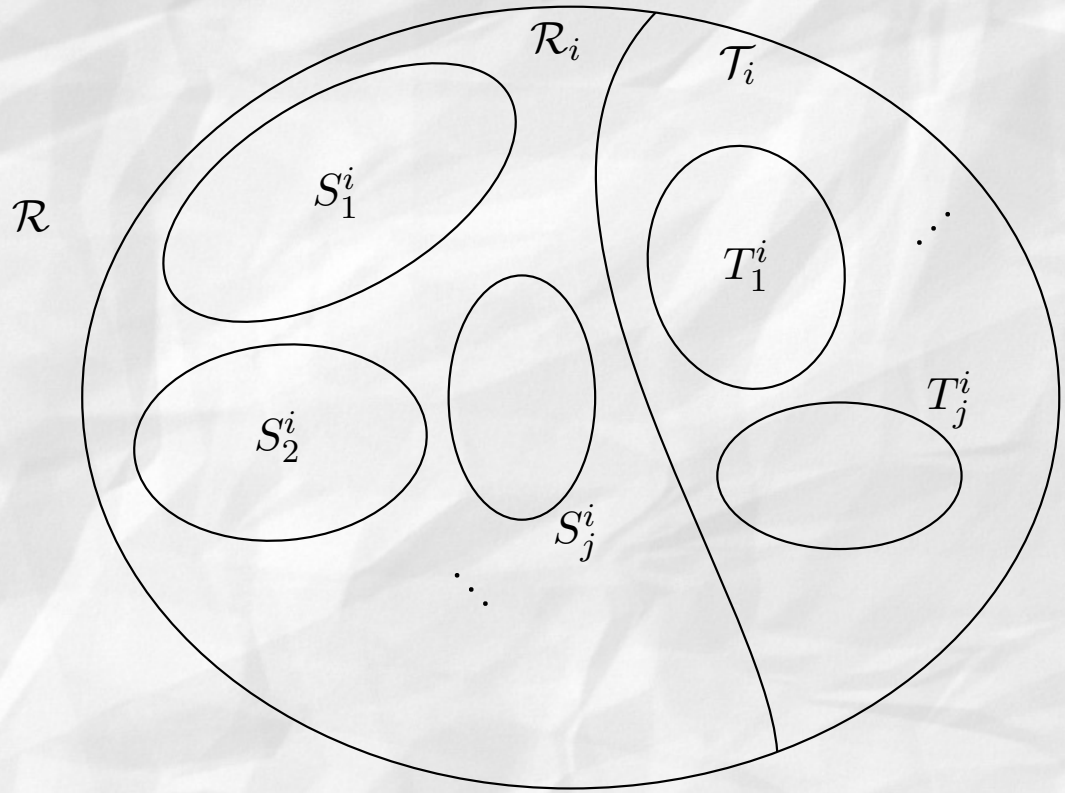
In the loop, step 1: initial partition

In the loop, step 2: removal of bad rectangles

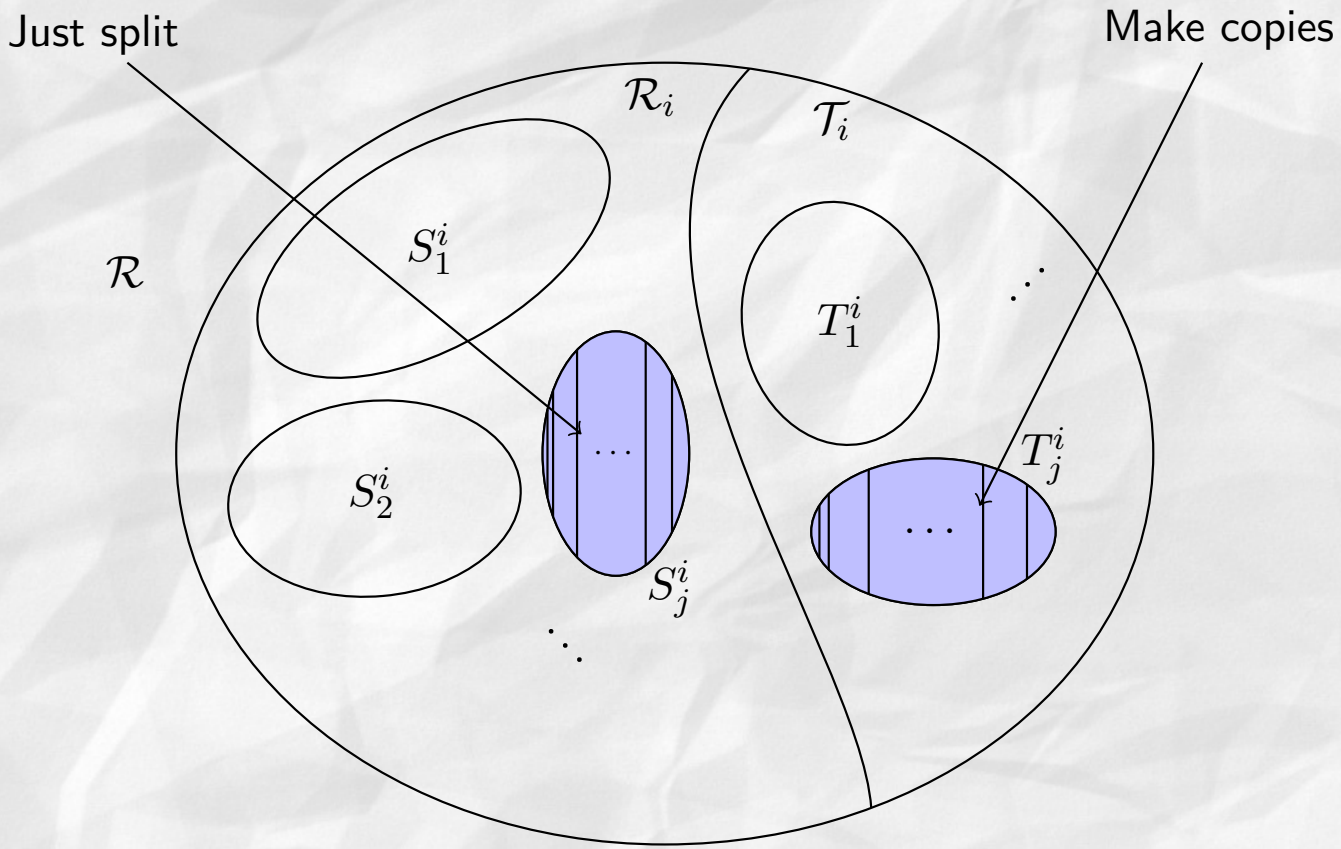
In the loop, step 3: removal of large cliques

Epilogue

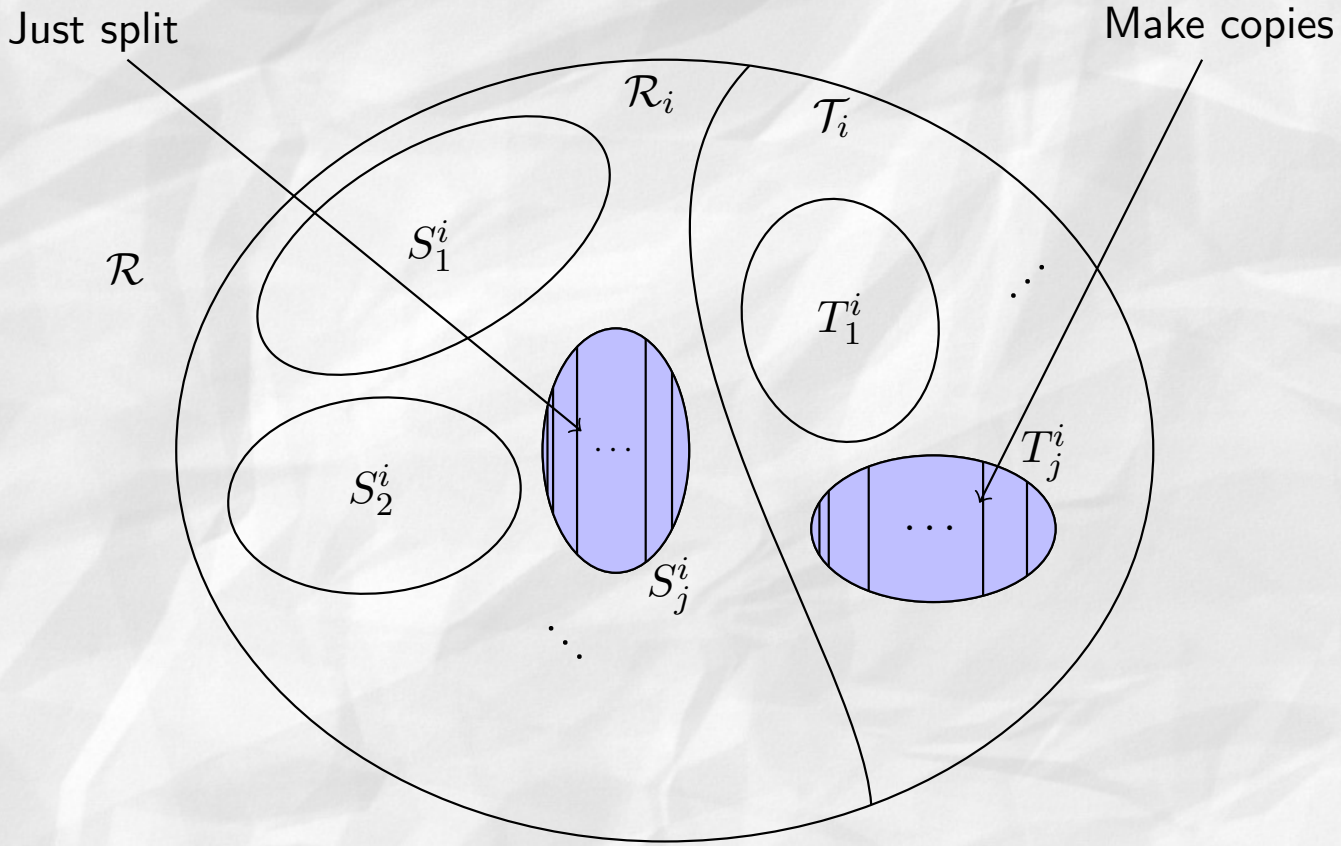
Refine partitions



Refine partitions



Refine partitions



✔Easy to see that **C1**–**C3** hold.

Outline

Introduction

The challenge

The algorithm

In the loop, step 1: initial partition

In the loop, step 2: removal of bad rectangles

In the loop, step 3: removal of large cliques

Epilogue

Bad rectangles

Getting closer to the core of the work, we need the following:

Claim 4 (Deriving from [8]).

For any set of rectangles $\mathcal{H}$ with max. clique size q, the number of **(γ, q)-bad** *rectangles, i.e.* **those which have corner intersections with at least γq recangles in $\mathcal{H}$,** *is at most* $\frac{4}{\gamma}|\mathcal{H}|$.

- It suffices to consider parameters $\gamma = 12 \log\log n$ and $q = 300M/\beta^i$ for rectangles in S_j^{i+1} and set set $\mathcal{H}$ to be $S_j^{i+1} \cup S_{j+1}^{i+1} \cup S_{j+2}^{i+1}$.
- We then remove at most $\frac{|S_j^{i+1} \cup S_{j+1}^{i+1} \cup S_{j+2}^{i+1}|}{3 \log\log n}$ from each S_j^{i+1} (Claim 4).
- Summing for all j, the numerator becomes at most $3|\mathcal{R}_{i+1}|$;
- **C1**,**C2** are not affected while **C3 still holds**: C_{i+1} is unchanged, W_{i+1} was decreased by a factor of $\log\log n$ at most.

Bad rectangles

Getting closer to the core of the work, we need the following:

Claim 4 (Deriving from [8]).

For any set of rectangles $\mathcal{H}$ with max. clique size q, the number of **(γ, q)-bad** *rectangles, i.e.* **those which have corner intersections with at least γq recangles in $\mathcal{H}$,** *is at most* $\frac{4}{\gamma}|\mathcal{H}|$.

- It suffices to consider parameters $\gamma = 12 \log\log n$ and $q = 300M/\beta^i$ for rectangles in S_j^{i+1} and set set $\mathcal{H}$ to be $S_j^{i+1} \cup S_{j+1}^{i+1} \cup S_{j+2}^{i+1}$.
- We then remove at most $\frac{|S_j^{i+1} \cup S_{j+1}^{i+1} \cup S_{j+2}^{i+1}|}{3 \log\log n}$ from each S_j^{i+1} (Claim 4).
- Summing for all j, the numerator becomes at most $3|\mathcal{R}_{i+1}|$;
- **C1**,**C2** are not affected while **C3 still holds**: C_{i+1} is unchanged, W_{i+1} was decreased by a factor of $\log\log n$ at most.

Bad rectangles

Getting closer to the core of the work, we need the following:

Claim 4 (Deriving from [8]).

For any set of rectangles $\mathcal{H}$ with max. clique size q, the number of **(γ, q)-bad** *rectangles, i.e.* **those which have corner intersections with at least γq recangles in $\mathcal{H}$,** *is at most* $\frac{4}{\gamma}|\mathcal{H}|$.

- It suffices to consider parameters $\gamma = 12 \log\log n$ and $q = 300M/\beta^i$ for rectangles in S_j^{i+1} and set set $\mathcal{H}$ to be $S_j^{i+1} \cup S_{j+1}^{i+1} \cup S_{j+2}^{i+1}$.
- We then remove at most $\frac{|S_j^{i+1} \cup S_{j+1}^{i+1} \cup S_{j+2}^{i+1}|}{3 \log\log n}$ from each S_j^{i+1} (Claim 4).
- Summing for all j, the numerator becomes at most $3|\mathcal{R}_{i+1}|$;
- **C1**,**C2** are not affected while **C3 still holds**: C_{i+1} is unchanged, W_{i+1} was decreased by a factor of $\log\log n$ at most.

Bad rectangles

Getting closer to the core of the work, we need the following:

Claim 4 (Deriving from [8]).

For any set of rectangles $\mathcal{H}$ with max. clique size q, the number of **(γ, q)-bad** *rectangles, i.e.* **those which have corner intersections with at least γq recangles in $\mathcal{H}$,** *is at most* $\frac{4}{\gamma}|\mathcal{H}|$.

- It suffices to consider parameters $\gamma = 12 \log\log n$ and $q = 300M/\beta^i$ for rectangles in S_j^{i+1} and set set $\mathcal{H}$ to be $S_j^{i+1} \cup S_{j+1}^{i+1} \cup S_{j+2}^{i+1}$.
- We then remove at most $\frac{|S_j^{i+1} \cup S_{j+1}^{i+1} \cup S_{j+2}^{i+1}|}{3 \log\log n}$ from each S_j^{i+1} (Claim 4).
- Summing for all j, the numerator becomes at most $3|\mathcal{R}_{i+1}|$;
- **C1**,**C2** are not affected while **C3 still holds**: C_{i+1} is unchanged, W_{i+1} was decreased by a factor of $\log\log n$ at most.

Outline

Introduction

The challenge

The algorithm

In the loop, step 1: initial partition

In the loop, step 2: removal of bad rectangles

In the loop, step 3: removal of large cliques

Epilogue

α-cover

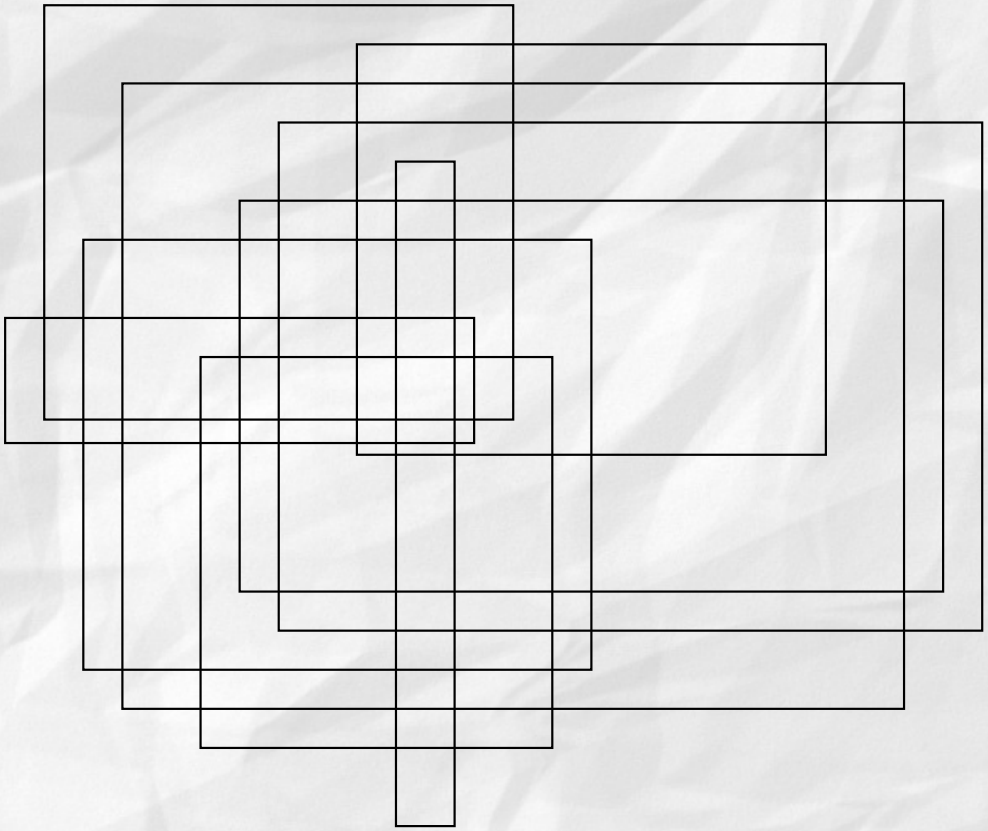


Figure:

α-cover

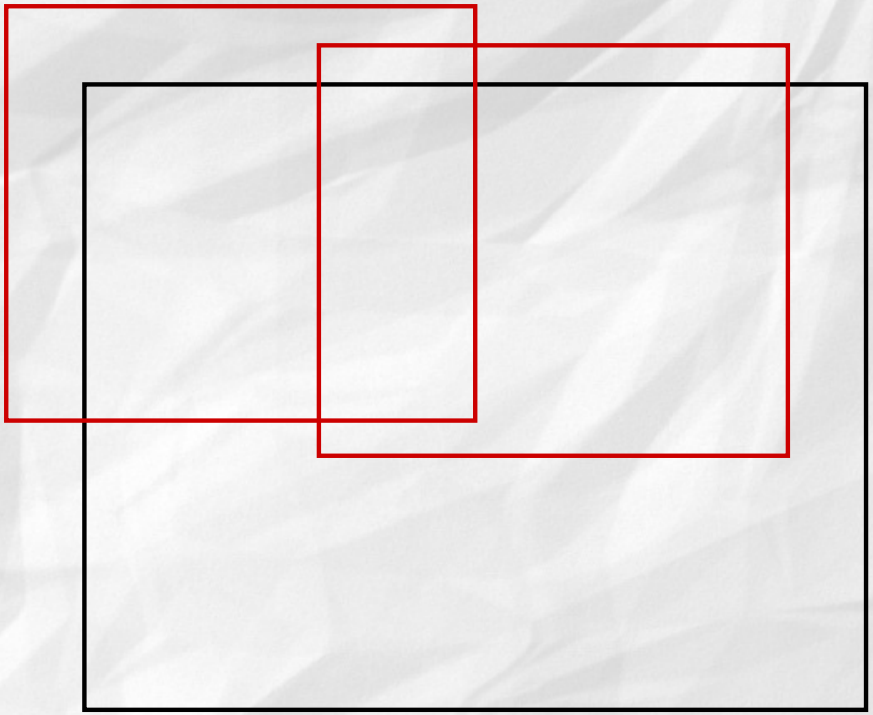


Figure:

α-cover

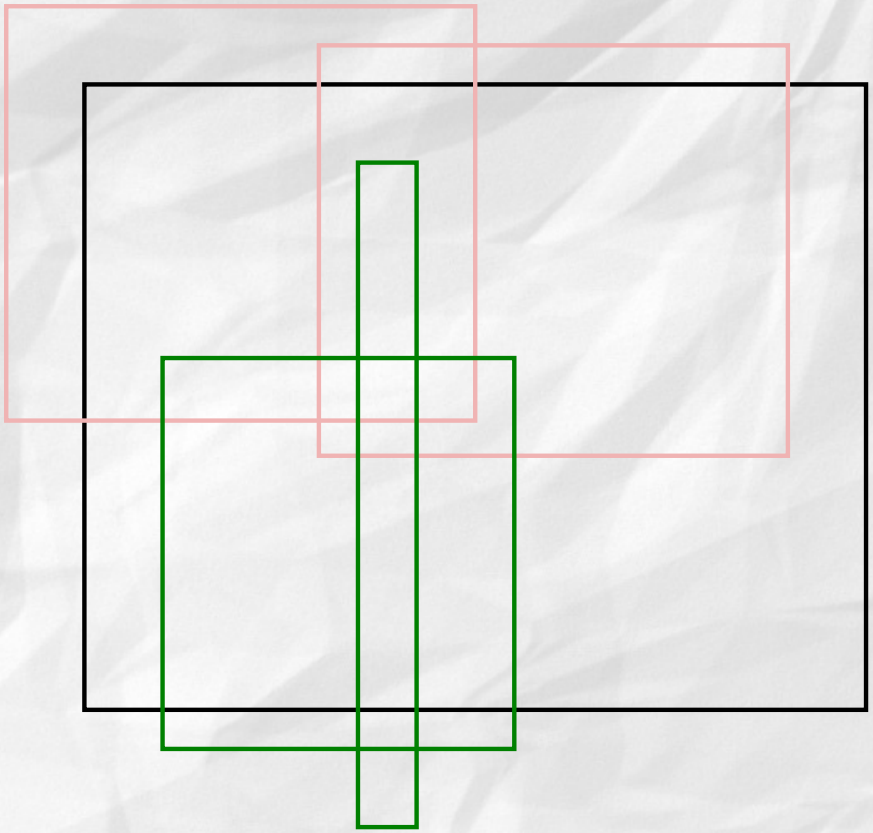


Figure:

α-cover

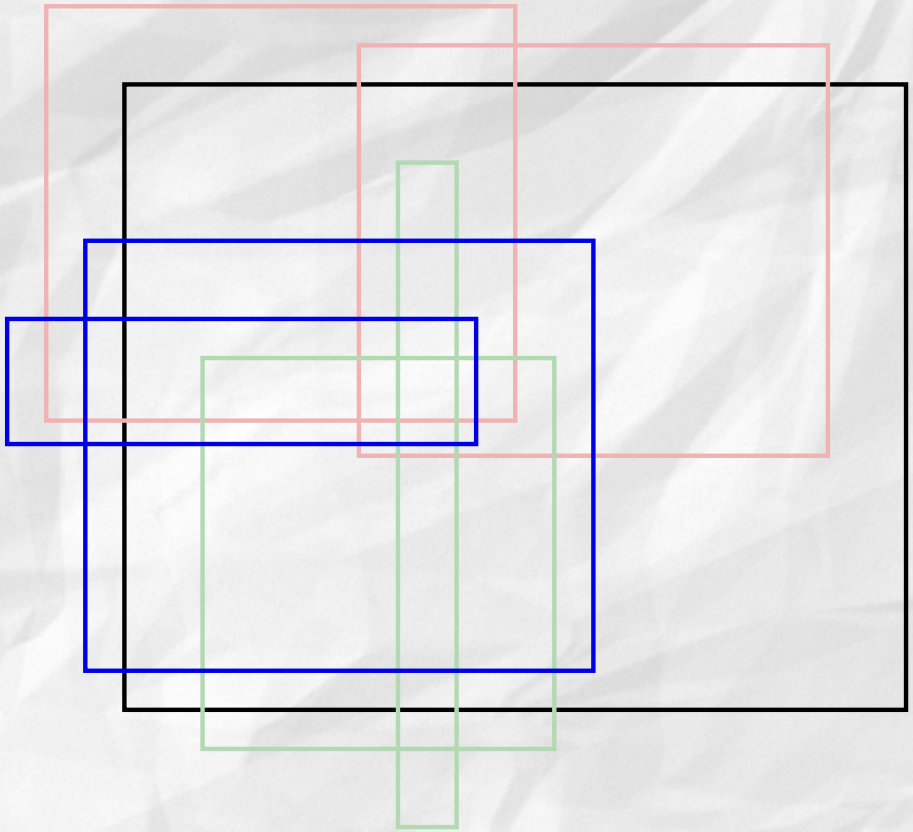


Figure:

α-cover

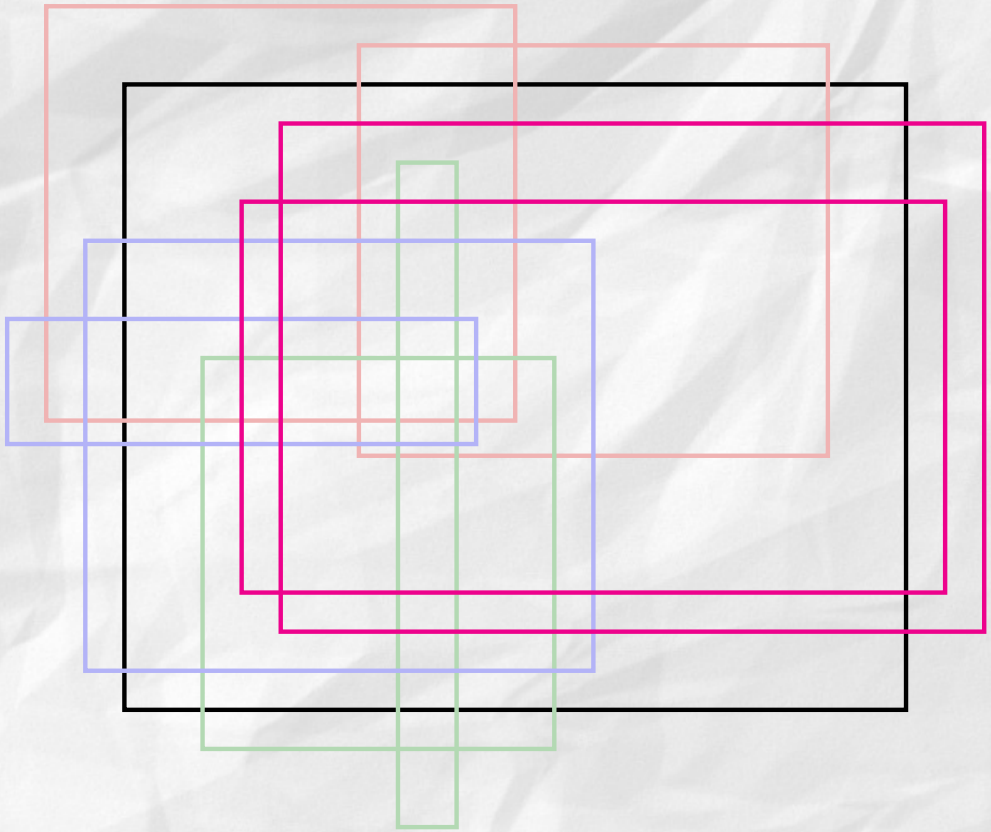


Figure:

α-cover

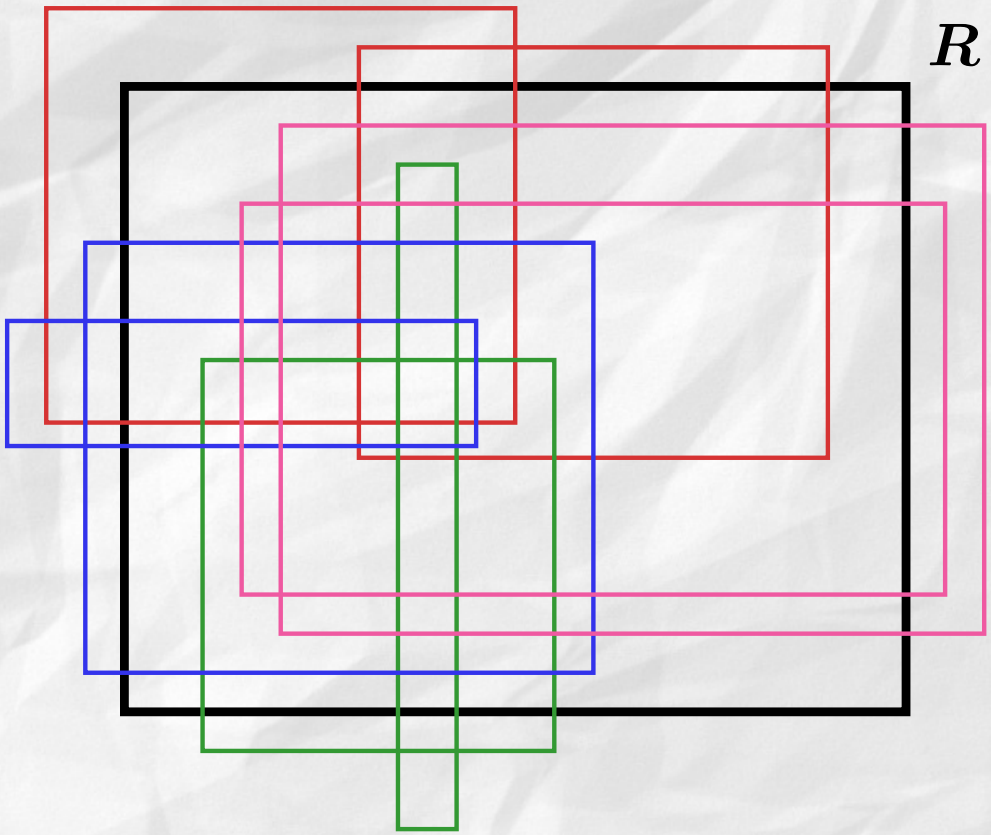


Figure: α-cover, $\alpha(R) = 2$

α-cover

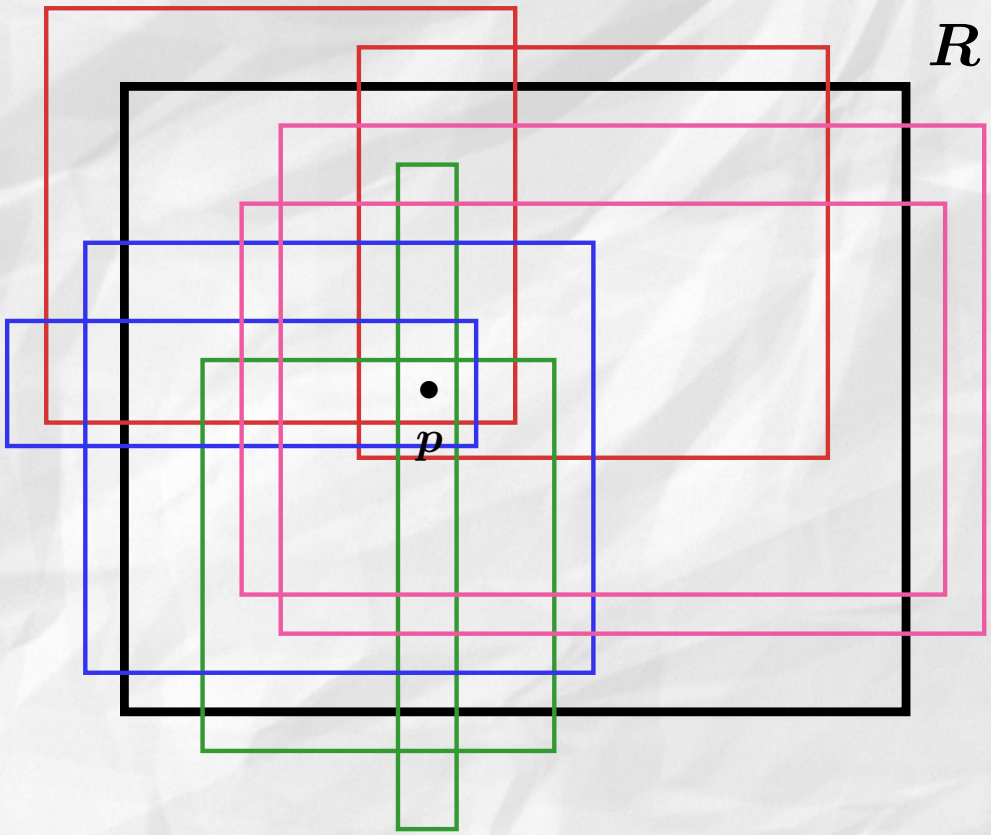


Figure: α-cover, $\alpha(R) = 2$

Claim 5.

If S_j contains a clique of size c then $\exists R \in S_j$ s.t. $\alpha(R) \geq \lfloor \frac{c}{4} \rfloor + 1$

Round(R) for removing the cliques

1. **Move** $\boldsymbol{R}$ to T_{j+2}
2. Remove all rectanges in $S_j \cup S_{j+1} \cup S_{j+2}$ containing $\boldsymbol{p}$
3. Remove all rectanges in $S_j \cup S_{j+1} \cup S_{j+2}$ having **corner intersections** with at least one of $\{R, R_l^1, R_r^1, R_l^2, R_r^2\}$
4. **Charge** R in T_{j+2} with the sum of the weights removed.

Observe:

- C3 **and C4** now hold!

Round(R) for removing the cliques

1. **Move $\boldsymbol{R}$** to T_{j+2}
2. Remove all rectanges in $S_j \cup S_{j+1} \cup S_{j+2}$ containing $\boldsymbol{p}$
3. Remove all rectanges in $S_j \cup S_{j+1} \cup S_{j+2}$ having **corner intersections** with at least one of $\{R, R_l^1, R_r^1, R_l^2, R_r^2\}$
4. **Charge** R in T_{j+2} with the sum of the weights removed.

Observe:

- C3 **and C4** now hold!

Checking condition C2

C2. Cost of every rectangle in $\mathcal{T}_i$ is $\boldsymbol{c_R \leq 10^6 \log\log n/\beta^i}$

- For C2, remember we have removed our bad rectangles, therefore the rectangles removed in Step 3 of *Round(R)* now intersect with at most $\gamma q = \Theta(\log\log n) \cdot q$ rectangles.

  Rectangles removed in Step 2 of *Round(R)* where contained in bounded-size clique.

Checking condition C2

C2. Cost of every rectangle in $\mathcal{T}_i$ is $\boldsymbol{c_R \leq 10^6 \log\log n/\beta^i}$

- For **C2**, remember we have removed our bad rectangles, therefore the rectangles removed in Step 3 of *Round(R)* now intersect with at most $\gamma q = \Theta(\log\log n) \cdot q$ rectangles.

  Rectangles removed in Step 2 of *Round(R)* where contained in bounded-size clique.

Checking condition C1

C1. Within T_j^i rectangles do not intersect; an $R \in T_j^i$ does not intersect any $R' \in S_j^i \cup S_{j-1}^i$

- Before *Round(R)*, **C1** holds. It is enough to prove:

Lemma 6.

Whenever rectangle $R \in S_j$ is added to T_{j+2} due to Round(R), it does not intersect any rectangle in $T_{j+2} \cup S_{j+2} \cup S_{j+1}$. **Moreover, there is no rectangle $R' \in S_j$ that intersects R in a non-corner manner, and whose width is greater than the width of R.**

Claim 7.

Let R' be any rectangle intersecting R in a non-corner manner, such that the width of R' is greater than that of R. Then $v(R') \geq v(R) + \alpha(R)/4 \geq v(R) + 5M/k$. It follows that $R' \notin S_j \cup S_{j+1} \cup S_{j+2}$.

Checking condition C1

C1. Within T_j^i rectangles do not intersect; an $R \in T_j^i$ does not intersect any $R' \in S_j^i \cup S_{j-1}^i$

- Before *Round(R)*, **C1** holds. It is enough to prove:

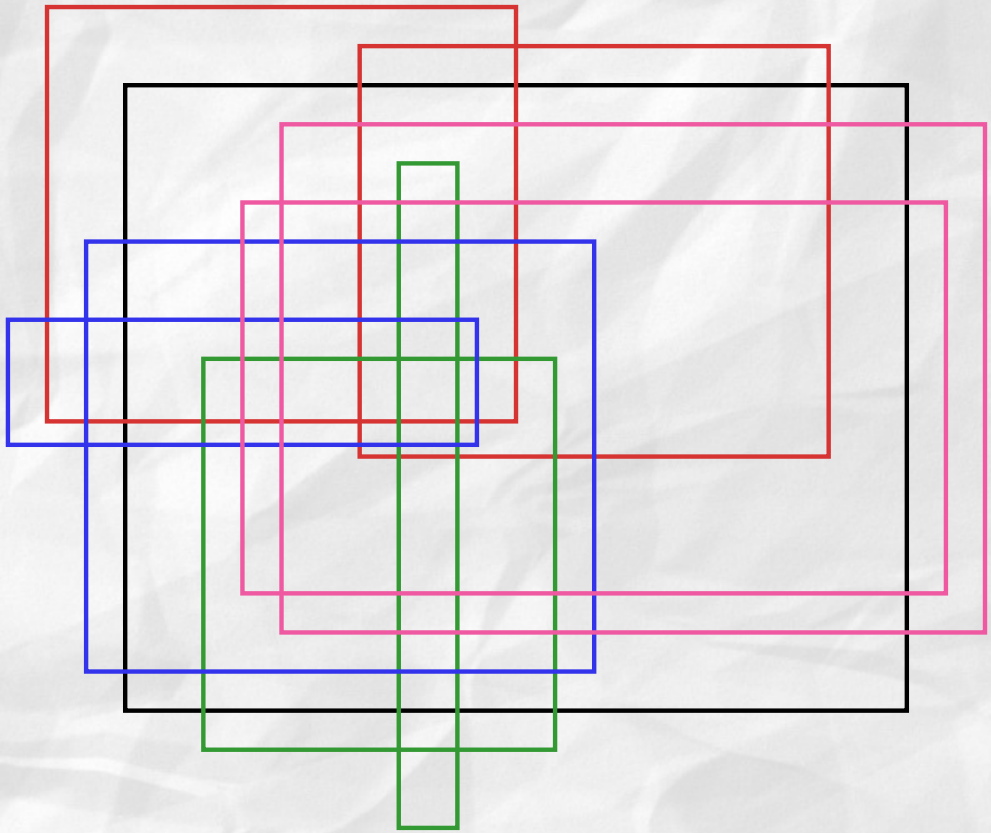
Lemma 6.

Whenever rectangle $R \in S_j$ is added to T_{j+2} due to Round(R), it does not intersect any rectangle in $T_{j+2} \cup S_{j+2} \cup S_{j+1}$. **Moreover, there is no rectangle $R' \in S_j$ that intersects R in a non-corner manner, and whose width is greater than the width of R.**

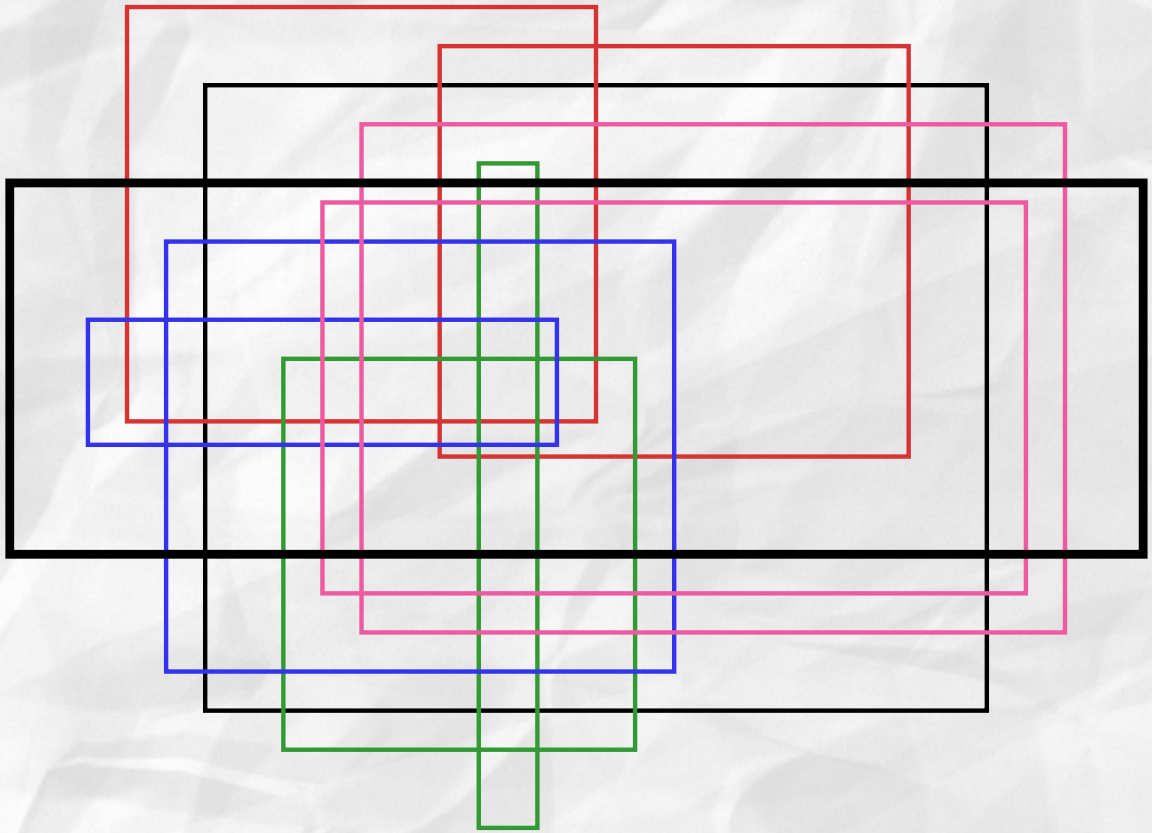
Claim 7.

Let R' be any rectangle intersecting R in a non-corner manner, such that the width of R' is greater than that of R. Then $v(R') \geq v(R) + \alpha(R)/4 \geq v(R) + 5M/k$. It follows that $R' \notin S_j \cup S_{j+1} \cup S_{j+2}$.

Checking condition C1



Checking condition C1



Outline

Introduction

The challenge

The algorithm

In the loop, step 1: initial partition

In the loop, step 2: removal of bad rectangles

In the loop, step 3: removal of large cliques

Epilogue

Generalization in higher dimensions

Theorem 8 (Chalermsook and Chuzhoy (2009)).

If there is a factor $f(n)$-approximation algorithm for $\texttt{MISB}_d$, then there is an $O(f(n)\log n)$-approximation algorithm for $\texttt{MISB}_{d+1}$.

It follows that there is an $O((\log n)^{d-2}\log\log n)$-approximation algorithm for MISB *on d dimensions.*

This improves the $O((\log n)^{d-1})$-approximation given by Berman et al., [2].

But:

- Approximation status remains open (**PTAS?**)
- Integrality gap for LP

Generalization in higher dimensions

Theorem 8 (Chalermsook and Chuzhoy (2009)).

If there is a factor $f(n)$-approximation algorithm for $\texttt{MISB}_d$, then there is an $O(f(n)\log n)$-approximation algorithm for $\texttt{MISB}_{d+1}$.

It follows that there is an $O((\log n)^{d-2}\log\log n)$-approximation algorithm for MISB *on d dimensions.*

This improves the $O((\log n)^{d-1})$-approximation given by Berman et al., [2].

But:

- Approximation status remains open (**PTAS?**)
- Integrality gap for LP

The end

Thank you!

Bibliography I

[1] P. K. Agarwal, M. J. van Kreveld, and S. Suri. Label placement by maximum independent set in rectangles. In *CCCG*, 1997.

[2] P. Berman, B. DasGupta, S. Muthukrishnan, and S. Ramaswami. Improved approximation algorithms for rectangle tiling and packing. In *SODA*, pages 427–436, 2001.

[3] P. Chalermsook and J. Chuzhoy. Maximum independent set of rectangles. In *SODA*, pages 892–901, 2009.

[4] T. M. Chan. Polynomial-time approximation schemes for packing and piercing fat objects. *J. Algorithms*, 46(2):178–189, 2003.

[5] T. M. Chan. A note on maximum independent sets in rectangle intersection graphs. *Inf. Process. Lett.*, 89(1):19–23, 2004.

Bibliography II

[6] T. Erlebach, K. Jansen, and E. Seidel. Polynomial-time approximation schemes for geometric intersection graphs. *SIAM J. Comput.*, 34(6):1302–1323, 2005.

[7] S. Khanna, S. Muthukrishnan, and M. Paterson. On approximating rectangle tiling and packing. In *SODA*, pages 384–393, 1998.

[8] L. Lewin-Eytan, J. Naor, and A. Orda. Routing and admission control in networks with advance reservations. In *APPROX*, pages 215–228, 2002.

[9] A. Lingas and M. Wahlen. A note on maximum independent set and related problems on box graphs. *Inf. Process. Lett.*, 93(4): 169–171, 2005.

[10] F. Nielsen. Fast stabbing of boxes in high dimensions. *Theor. Comput. Sci.*, 246(1-2):53–72, 2000.