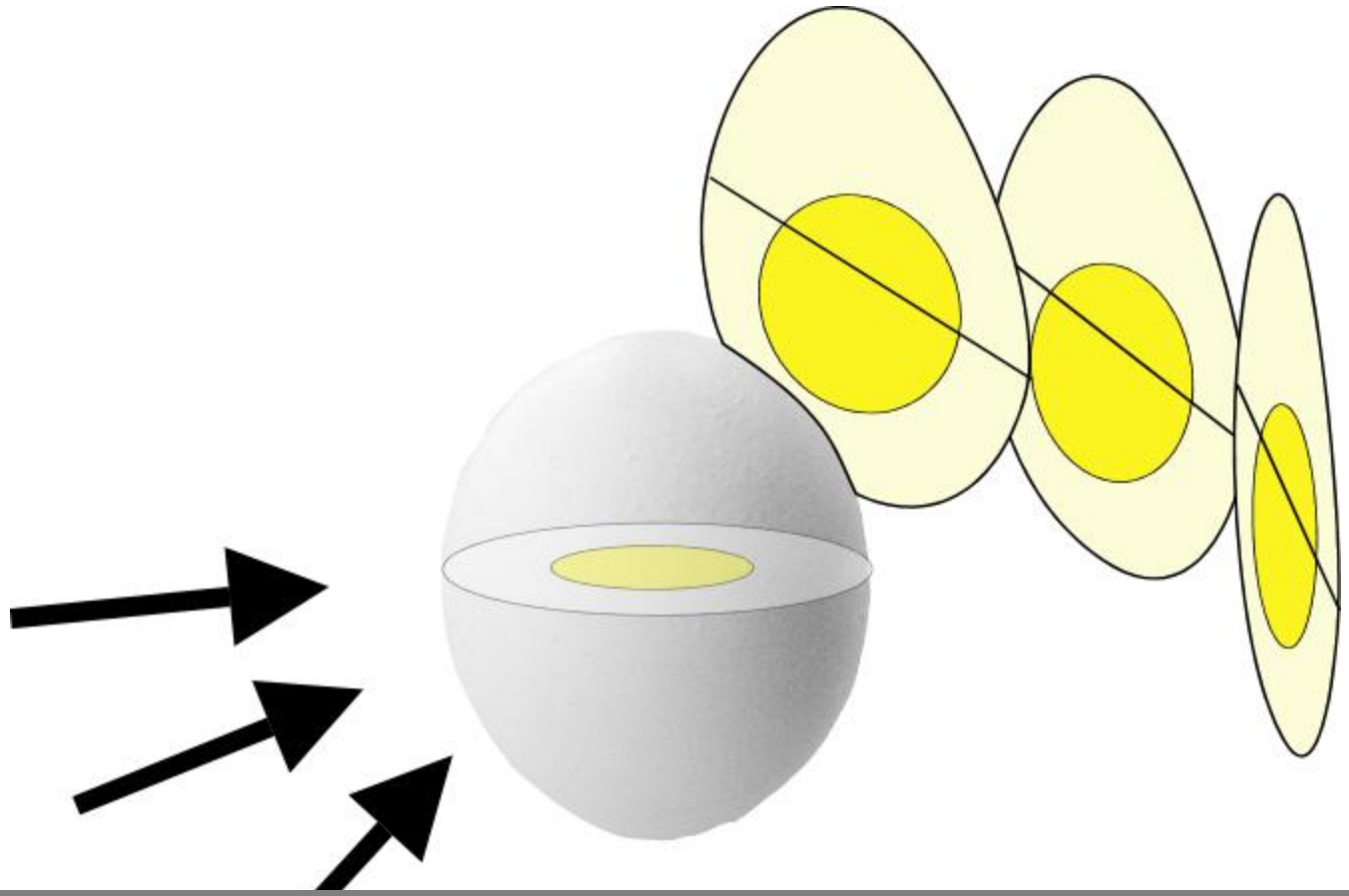
Geometric Tomography
with
Strip-based Model, Quad-trees and Greedy Algorithms

Yan Gerard

Groupe de Travail de Géométrie Discrète, *June 13th 2013, Paris*

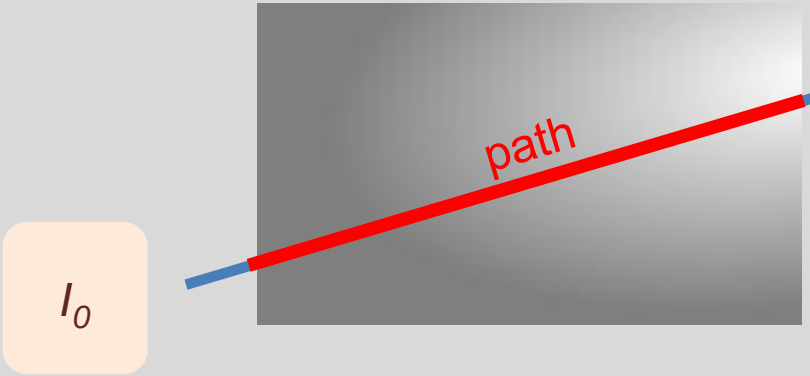
Geometric Tomography
with
Strip-based Model, Quad-trees and Greedy Algorithms

The different materials absorb differently the X-rays which cross it



The intensity of the X-ray measured behind the object provides an information about the lengths of the different materials which have been crossed by the X-ray.

The intensity of the X-ray measured behind the object provides an information about the lengths of the different materials which have been crossed by the X-ray.



$$I_{measured} = I_0 \int_{path} \exp(-f(x,y))\, dl$$

$f(x,y)$ is the absorbance coefficient
it characterizes the material at point (x,y)

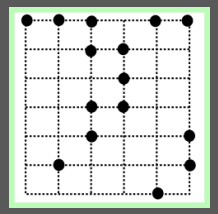
The image of $f(x,y)$ is interesting for many applications (medicine, non destructive control, geology…)

Geometric Tomography
with
Strip-based Model, Quad-trees and Greedy Algorithms

Computerized Tomography deals with the reconstruction
of a continuous function $f : [0 , 1]^2 \rightarrow [0 , 1]$
on a continuous domain

Geometric Tomography deals with the reconstruction
of a binary function $f : [0 , 1]^2 \rightarrow \{0 , 1\}$
on a continuous domain *namely a subset of* $[0 , 1]^2$

Discrete Tomography deals with the reconstruction
of a binary function *f : Lattice* $\rightarrow \{0 , 1\}$
on a discrete domain *namely a lattice set*

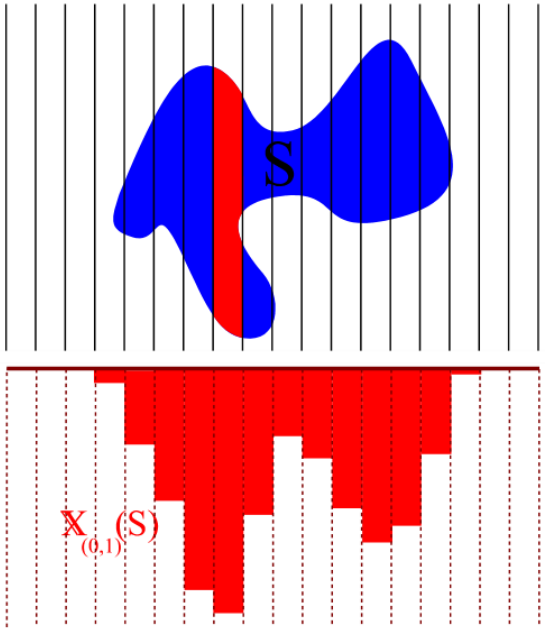


Geometric Tomography
with
Strip-based Model, Quad-trees and Greedy Algorithms

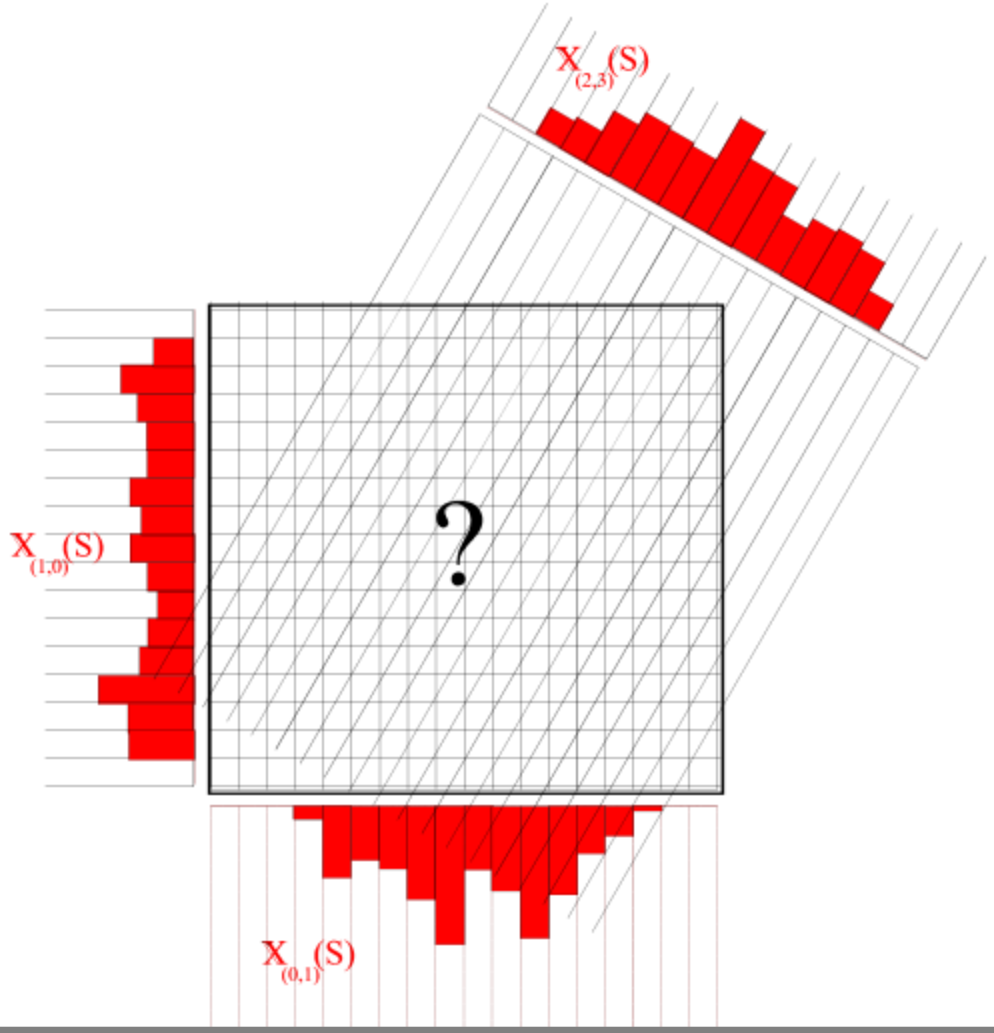
according to
the *length ray model*

a shape *S*

the projection of *S*

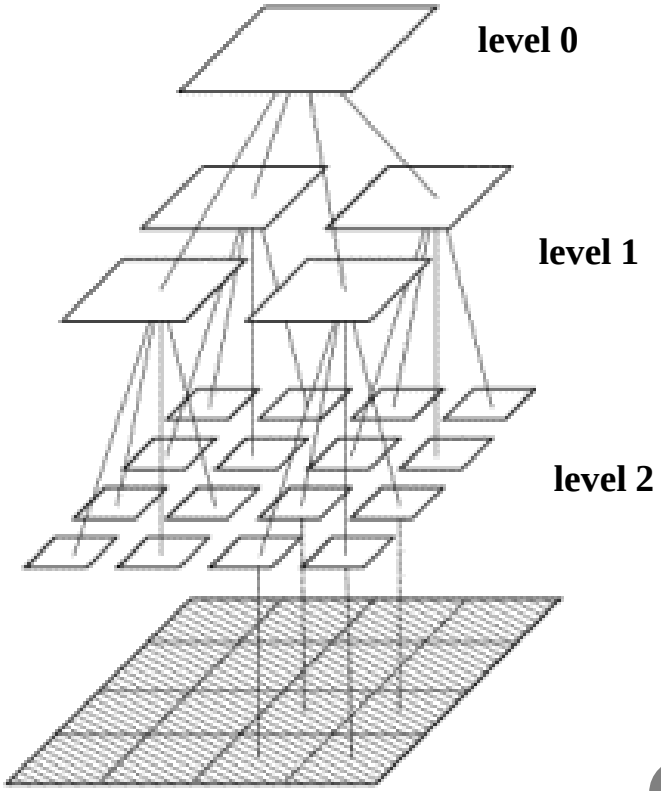


according to
the *strip-based model*

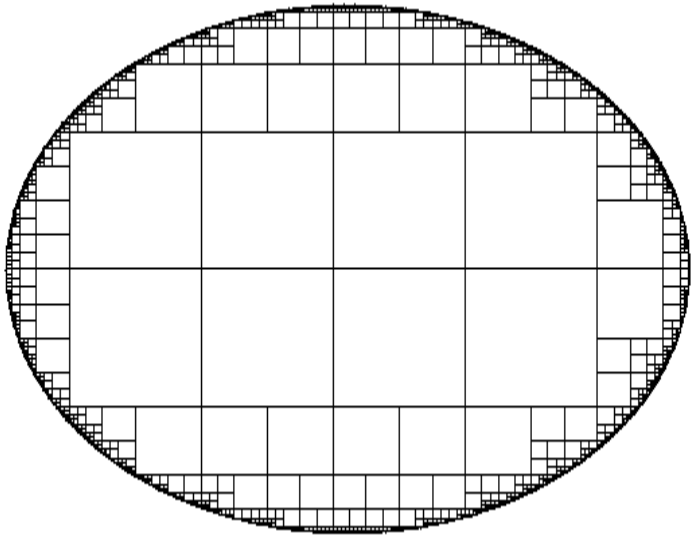


Input: a finite set of directions
for each direction, a projection
Output: A subset of $[0,1]^2$ with the prescribed projections

Geometric Tomography
with
Strip-based Model, Quad-trees and Greedy Algorithms

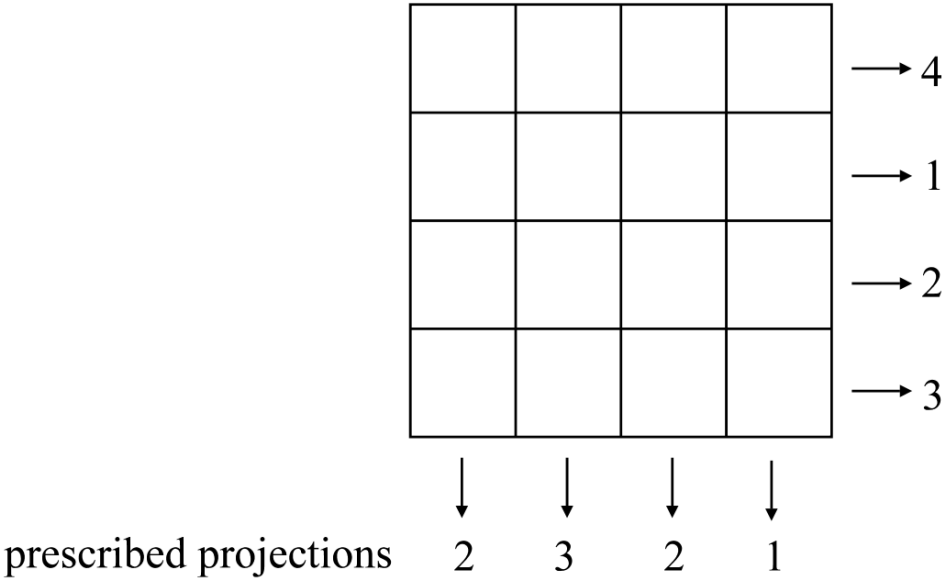


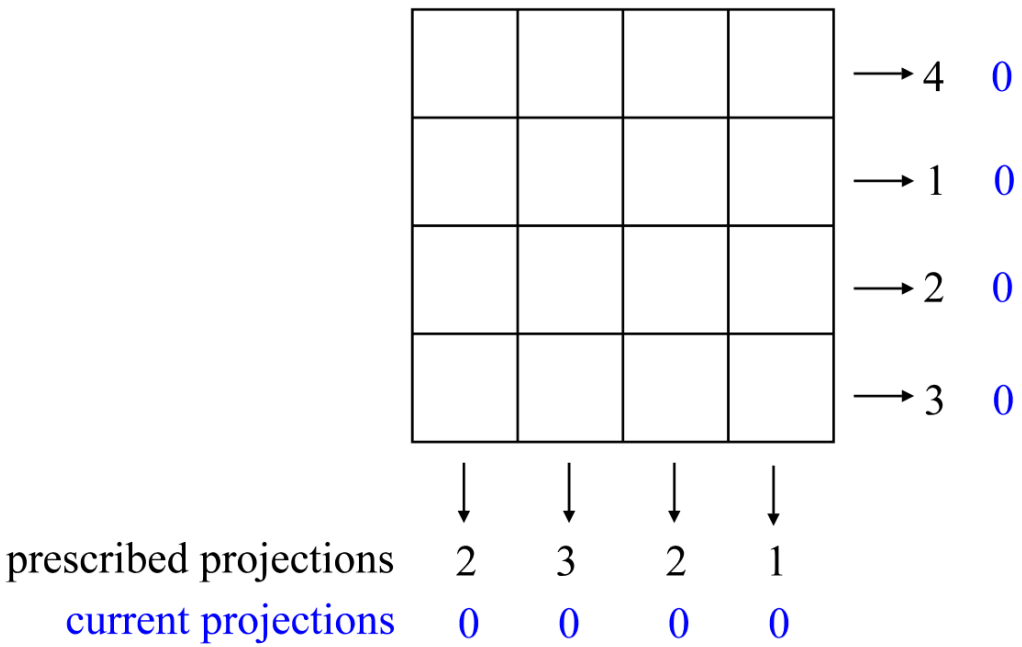
A quad-tree



shapes are represented by
unions of cells (quads) of different sizes

Geometric Tomography
with
Strip-based Model, Quad-trees and Greedy Algorithms

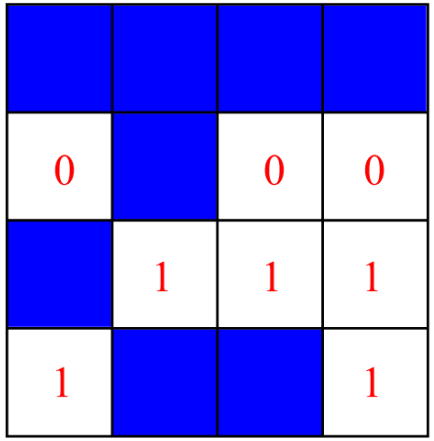

4
1
2
3
prescribed projections
2
3
2
1



Initialization:
Take an empty current shape

6	7	6	5	→ 4	0	4
3	4	3	2	→ 1	0	1
4	5	4	3	→ 2	0	2
5	6	5	4	→ 3	0	3
↓	↓	↓	↓			
prescribed projections 2	3	2	1			
current projections 0	0	0	0			
error on projections 2	3	2	1			

Initialization:
Take an empty current shape
Compute error table



				4	4	0
	0		0	0	1	1	0
		1	1	1	2	1	1
	1			1	3	2	1
prescribed projections	2	3	2	1			
current projections	2	3	2	1			
error on projections	0	0	0	0			

Initialization:
Take an empty current shape
Compute error table

Loop:
Find maximum error
Add point and update error table

End:
No more positive error in one direction

Plan

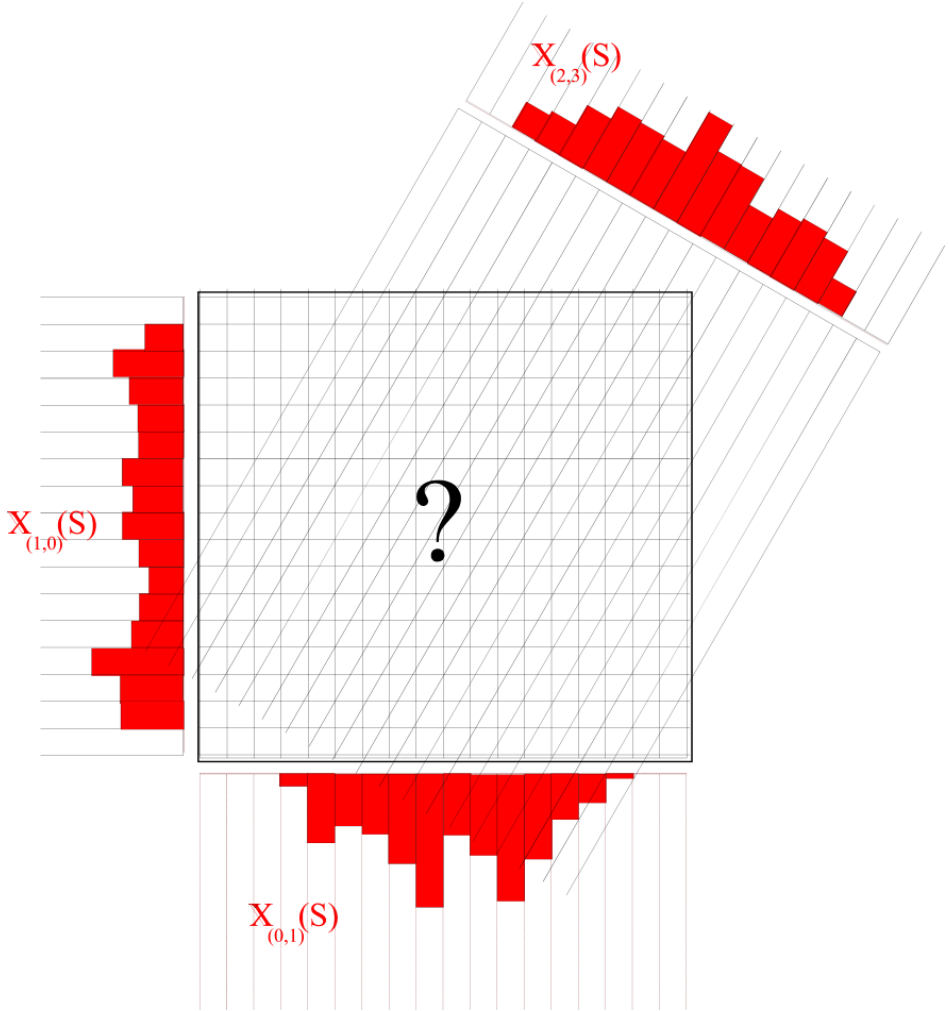
I Introduction

II Linear Programming

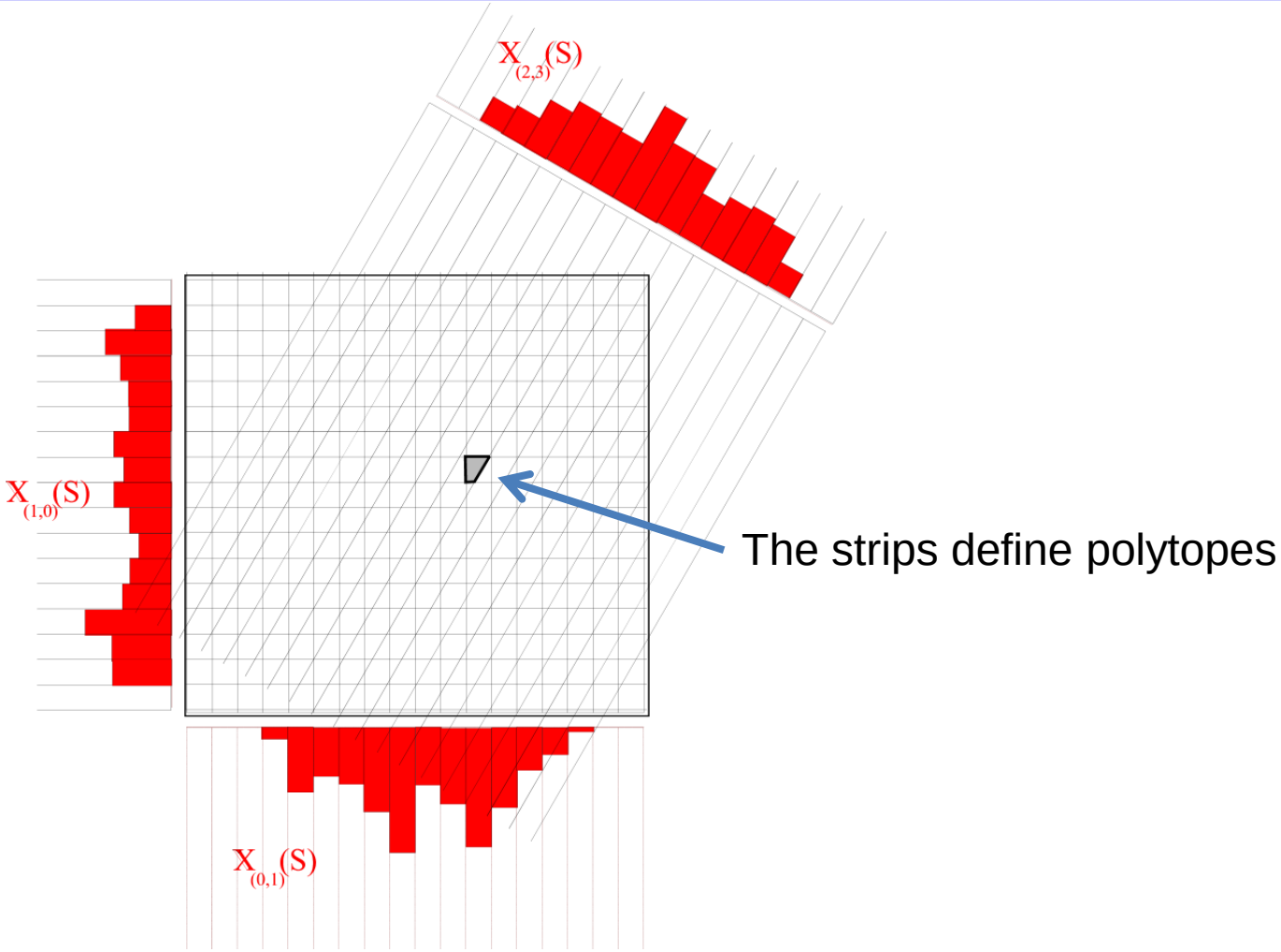
III Error Density & Greedy Algorithms

IV Heuristic

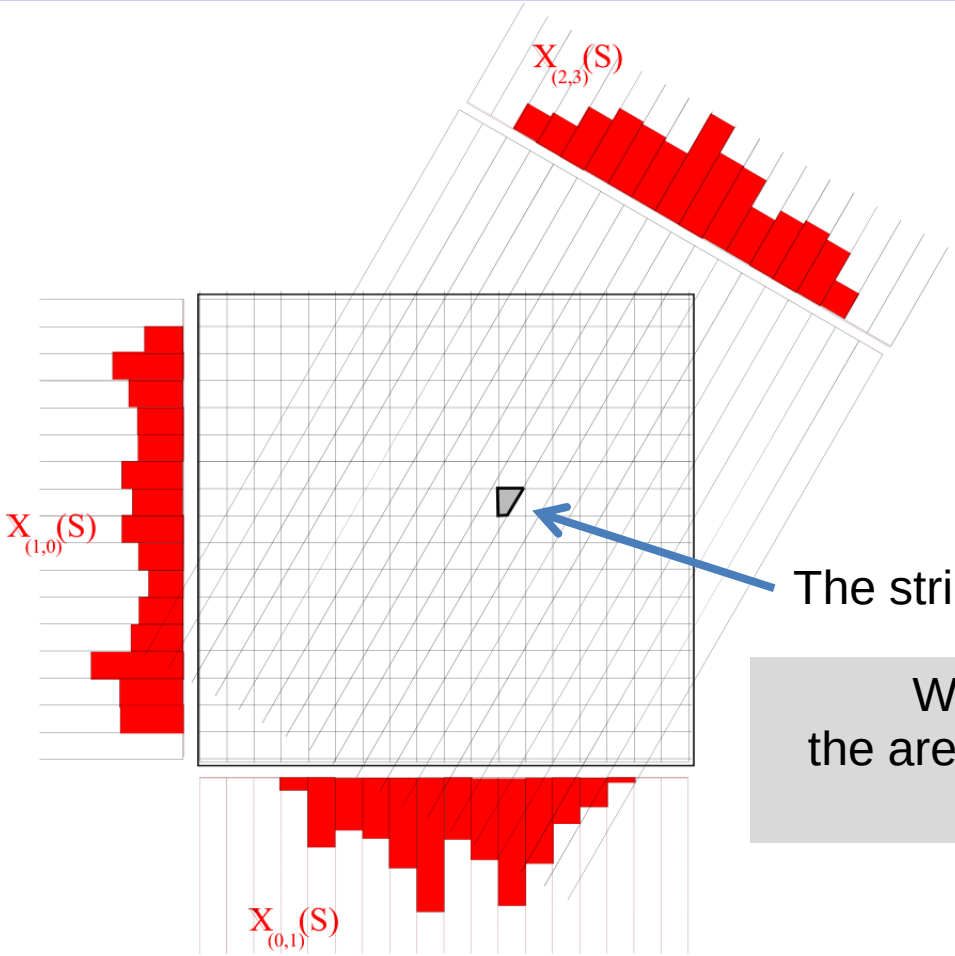
V Experiments



An instance



An instance



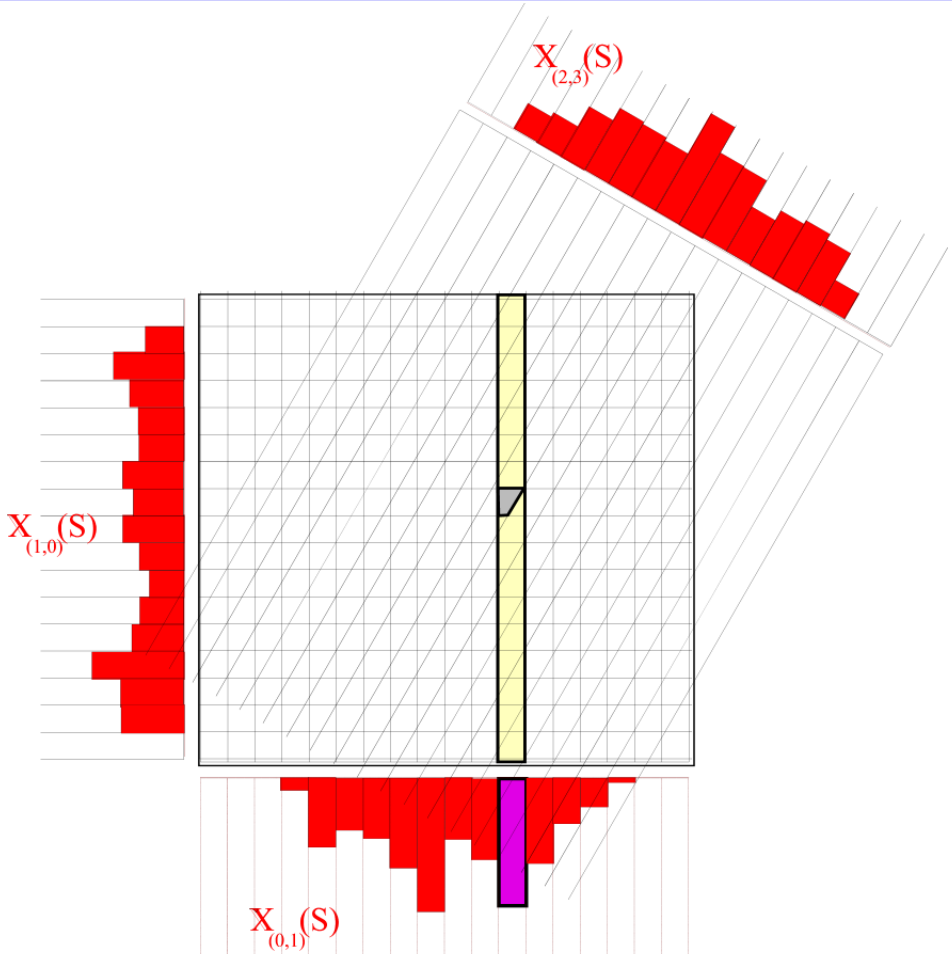
The strips define polytopes

We can compute
the area of each polytope :
Area(P_i)

Let a_i be the area of the intersection of a solution with the polytope P_i

For each polytope P_i , $0 \leq a_i \leq area(P_i)$

Linear Programming



Let a_i be the area of the intersection of a solution with the polytope P_i

For each polytope P_i , $0 \leq a_i \leq area(P_i)$

For each strip S_i , we know $\sum_{i \,/\, P_i \text{ in strip } S_j} a_i = projection(\text{strip } S_j)$

Linear Programming

For each polytope P_i , $0 \le a_i \le area(P_i)$

For each strip S_i , we know $\sum_{i \,/\, P_i \text{ in strip } S_j} a_i = projection(strip\ S_j)$

Linear Programming

For each polytope P_i , $0 \le a_i \le area(P_i)$

For each strip S_j , $proj(strip\ S_j) - h_j \le \sum_{i \,/\, P_i \text{ in strip } S_j} a_i \le proj(strip\ S_j) + h_j$

Minimize $\sum_j h_j$

Linear Programming

For each polytope P_i , $0 \leq a_i \leq area(P_i)$

For each strip S_j , $proj(strip\ S_j) - h_j \leq \sum_{i \text{ / } P_i \text{ in strip } S_j} a_i \leq proj(strip\ S_j) + h_j$

Minimize $\sum_j h_j$

But how many variables ?

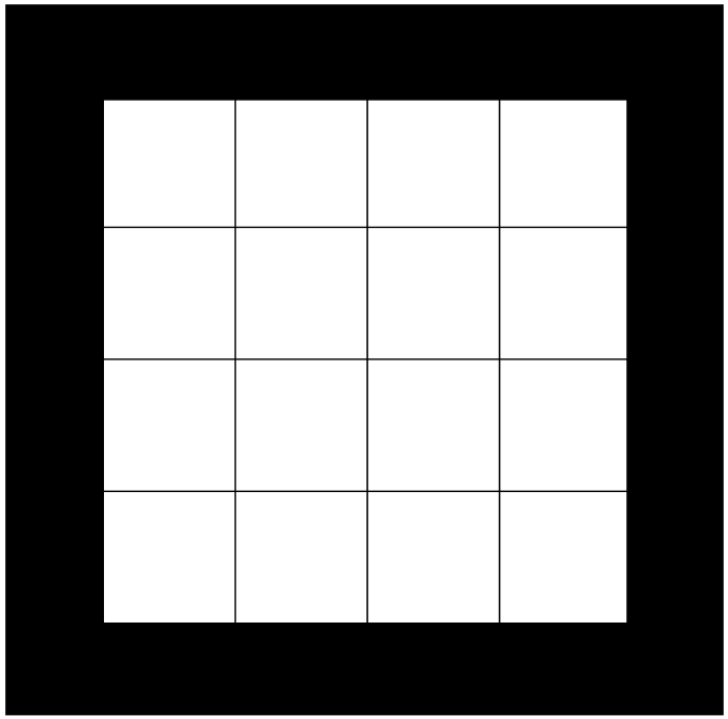
2 x number of directions + number of polytopes

How many polytopes ?

Strip Combinatorics

How many polytopes ?

With *2* directions

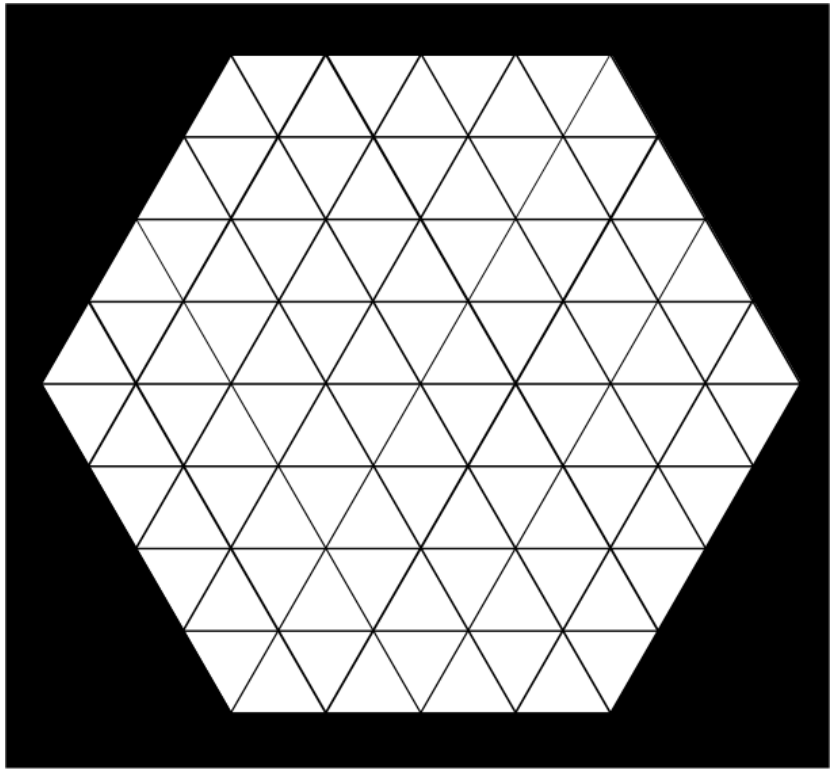


number of captors ^{2}

Strip Combinatorics

How many polytopes ?

With *3* directions



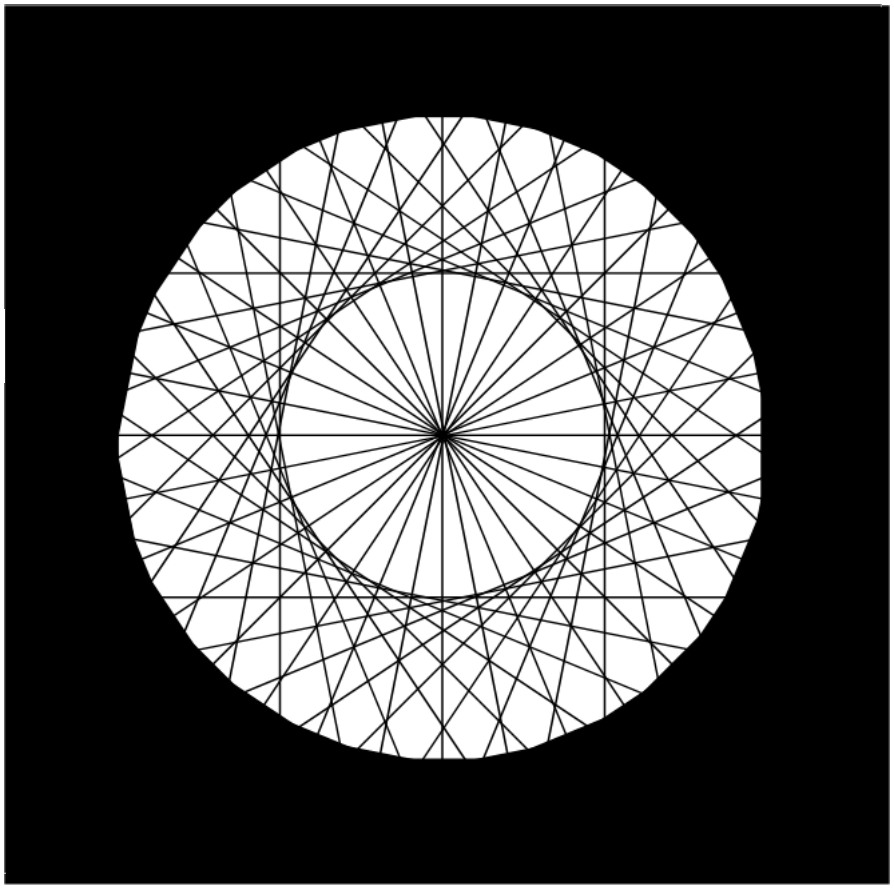
With an even number of captors *(2n)* : $6\,n^2$

With an odd number of captors *(2n+1)* : $6n^2+6n+1$

How many polytopes ?

With *16* directions

and *4* captors

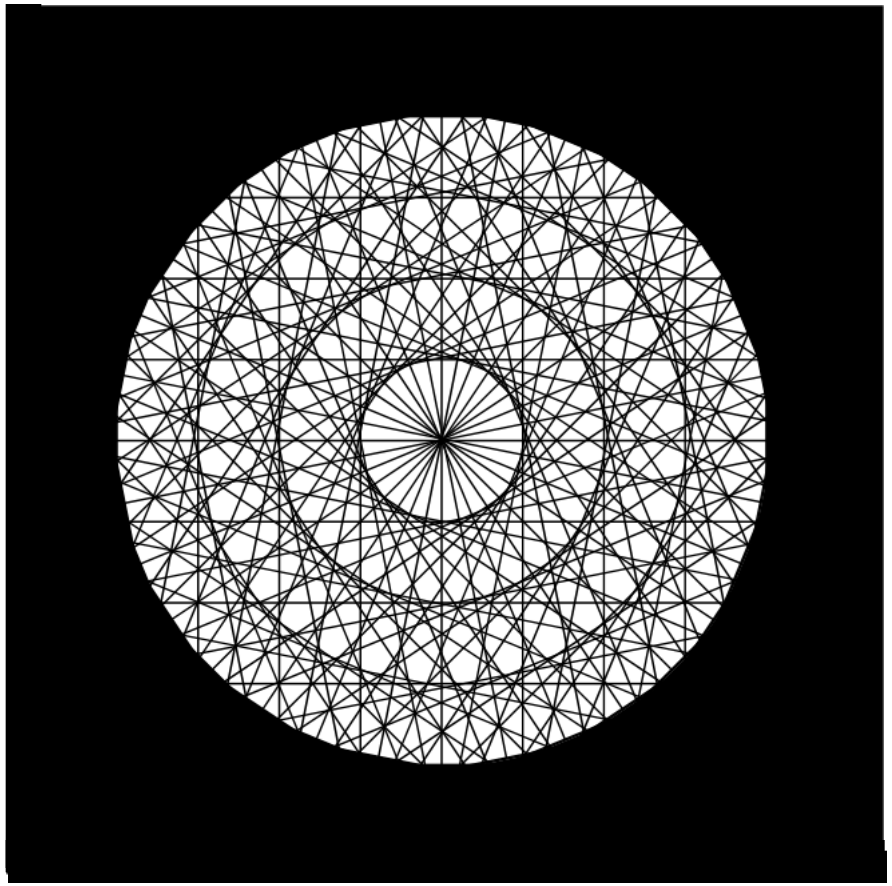


576 polytopes

How many polytopes ?

With *16* directions

and *8* captors

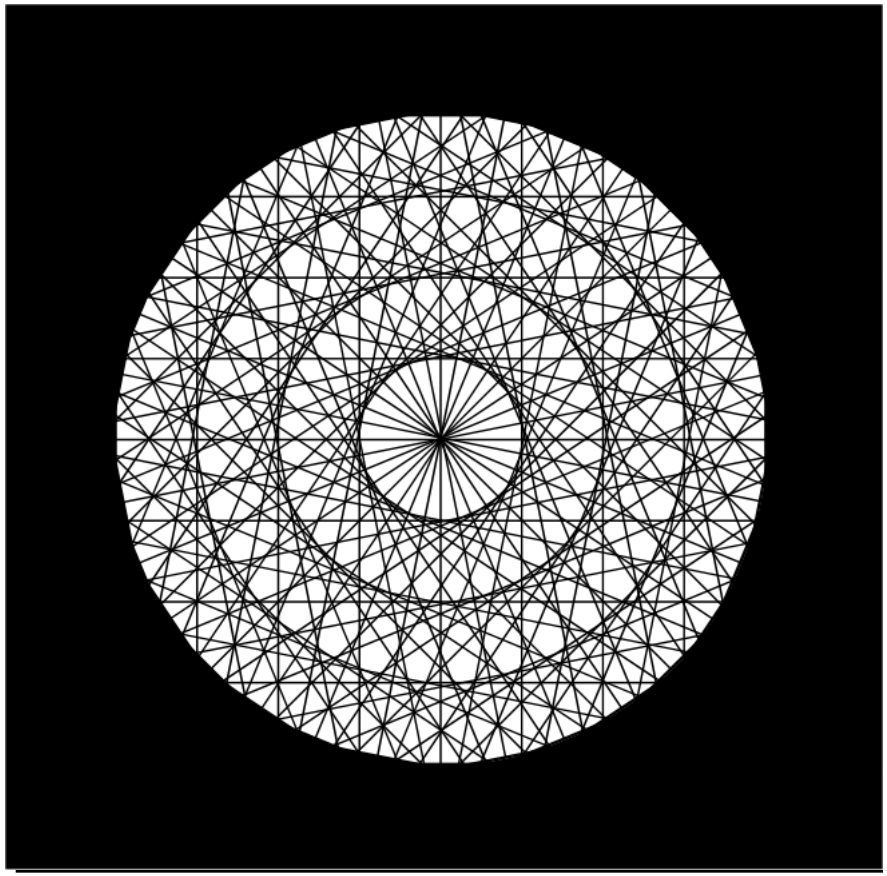


? polytopes

How many polytopes ?

With *16* directions

and *8* captors



Strip Combinatorics

How many polytopes ?

	4 captors	8 captors	n captors
2 directions	***16***	***64***	n^2
3 directions	***24***	***96***	$3n^2/2$ *or* $(3n^2-1)/2$
4 directions	***48***	***232***	***?***
8 directions	***160***	***800***	***?***
16 directions	***576***	***?***	***?***
k directions	***?***	***?***	***?***

It can probably make a lot of polytopes

A lot of variables for LP-solvers

Linear Programming

At the end, the user has to choose where he puts the points in each polytope

Other ideas ?

Other ideas ?

make something completely different with
quad-trees,
greedy algorithms,
and simulated annealing

Next Section

I Back to Radon transform

II Linear Programming

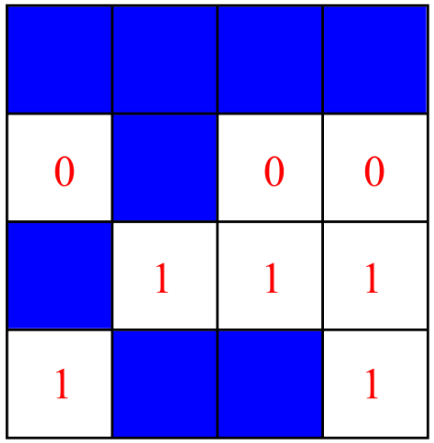
III Error Density & Greedy Algorithms

IV Heuristic

V Experiments

Error Density

Principle of greedy algorithm



				4	4	0
	0		0	0	1	1	0
		1	1	1	2	1	1
	1			1	3	2	1
prescribed projections	2	3	2	1			
current projections	2	3	2	1			
error on projections	0	0	0	0			

Initialization:
Take an empty current shape
Compute error table

Loop:
Find maximum error
Add point and update error table

It uses a local criterion of error

Error Density

It uses a local criterion of error

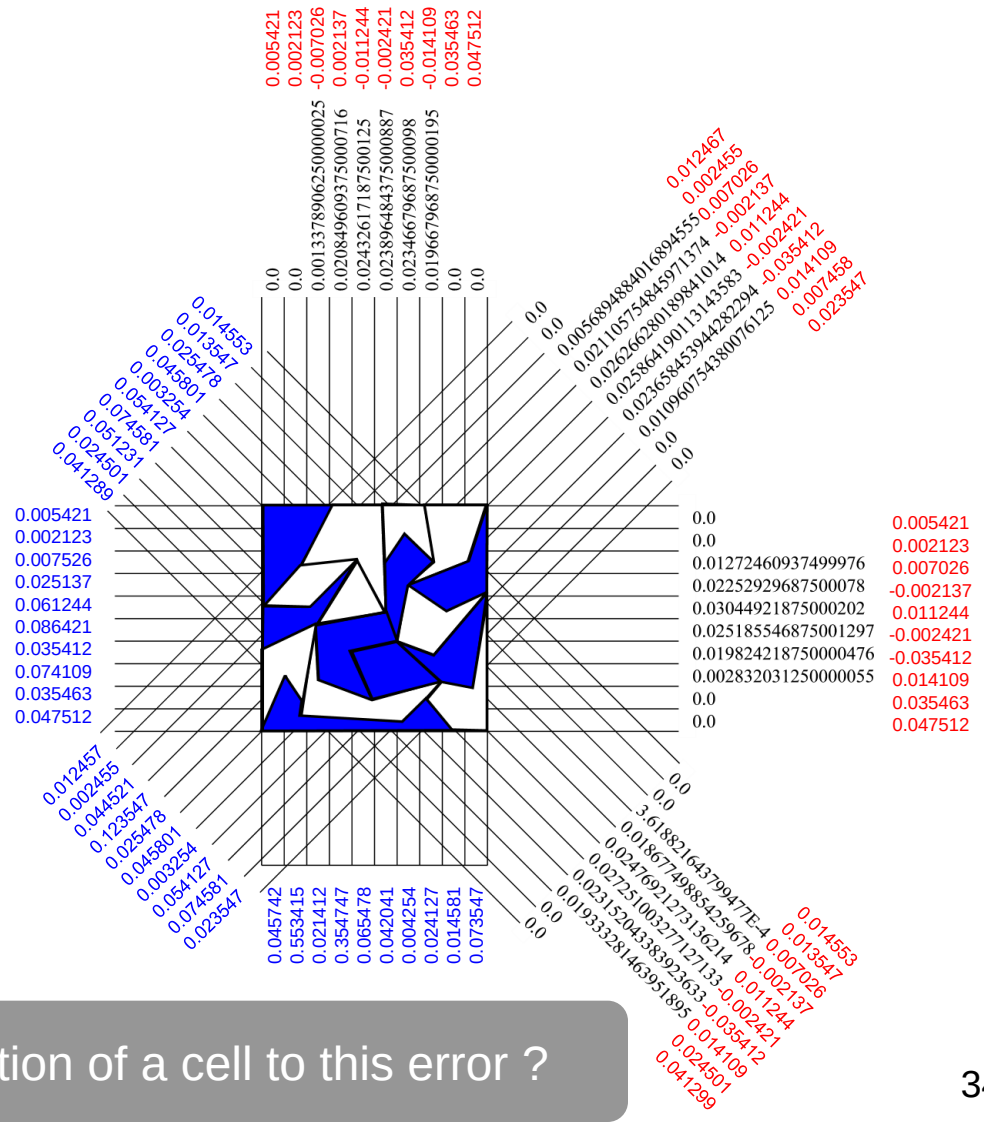
An instance

A partition of the domain

A set of cells

Its projections

The error on each projection

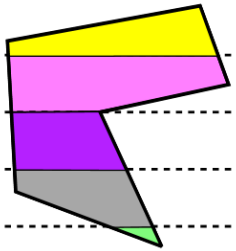


What is the contribution of a cell to this error ?

Error Density

What is the contribution of a cell to this error ?

Projections of the cell	Errors on projections		
0.0	+ 0.019820	x	0.0
0.0	- 0.052203	x	0.0
0.0	+ 0.045781	x	0.0
0.001231	- 0.010175	x	0.001231
0.002307	- 0.001265	x	0.002307
0.000843	+ 0.021479	x	0.000843
0.000912	- 0.012728	x	0.000912
0.000005	+ 0.022521	x	0.000005
0.0	- 0.030404	x	0.0
0.0	- 0.025168	x	0.0
			Sum for all directions



We compute for each cell error sum

Error Density

We compute for each cell error sum

But small cells are penalized : their error sum is small

Then we introduce for each cell its error density $\frac{\text{error sum}}{\text{cell area}}$

We use the error density to drive greedy algorithms

If its negative, there is a deficit of points at the cell

If its positive, there is an excess of points at the cell

III Greedy Algorithm

We use the error density to drive greedy algorithms

If its negative, there is a deficit of points at the cell

If its positive, there is an excess of points at the cell

Greedy Algorithm :

Initialization:
Take a decomposition of the domain in cells
Take an empty current set of cells
Compute error density table

Loop:
Find minimal (negative) error density
Add cell and update error table

End:
No more negative error density

III Greedy Algorithm

Initialization:
Take a decomposition of the domain in cells
Take an empty current set of cells
Compute error density table

Loop:
Find minimal (negative) error density
Add cell and update error table

End:
No more negative error density

Other ideas ?

The cell structure is not modified !

May be, we can improve it

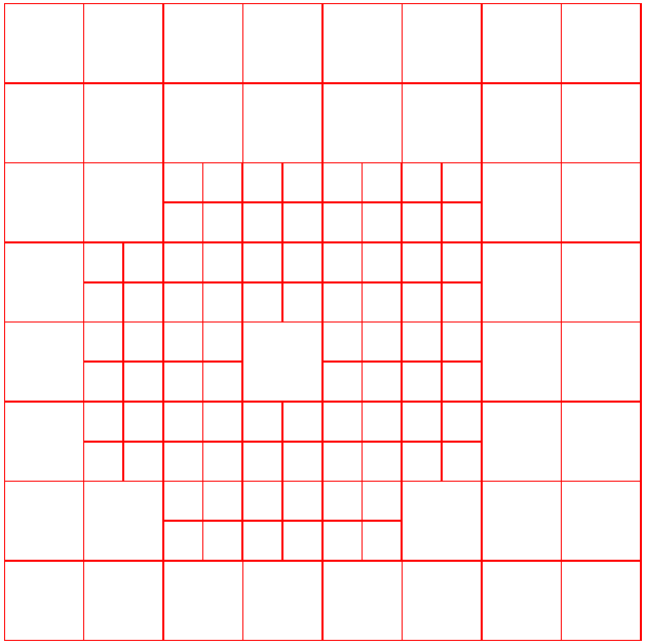
Greedy Algorithm

The cell structure is not modified!

May be, we can improve it

Initialization:
Take a decomposition in cells

Loop:
Perform Greedy Algorithm
Refine decomposition



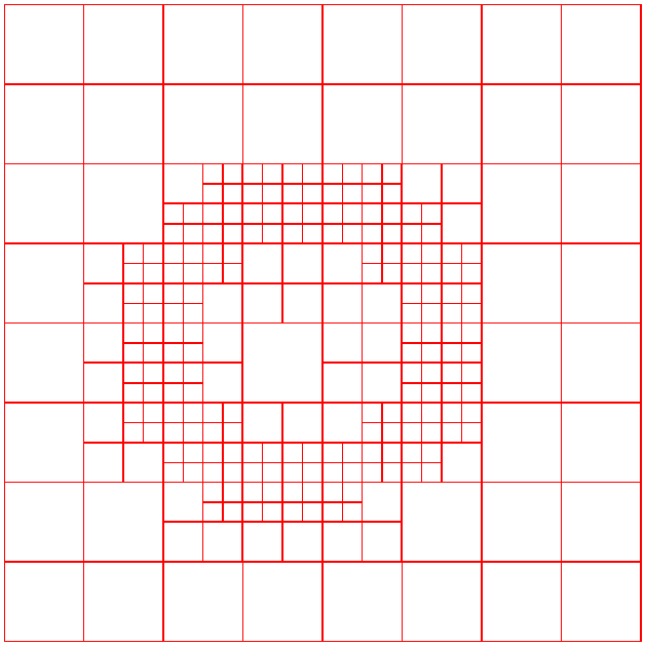
Greedy Algorithm

The cell structure is not modified!

May be, we can improve it

Initialization:
Take a decomposition in cells

Loop:
Perform Greedy Algorithm
Refine decomposition



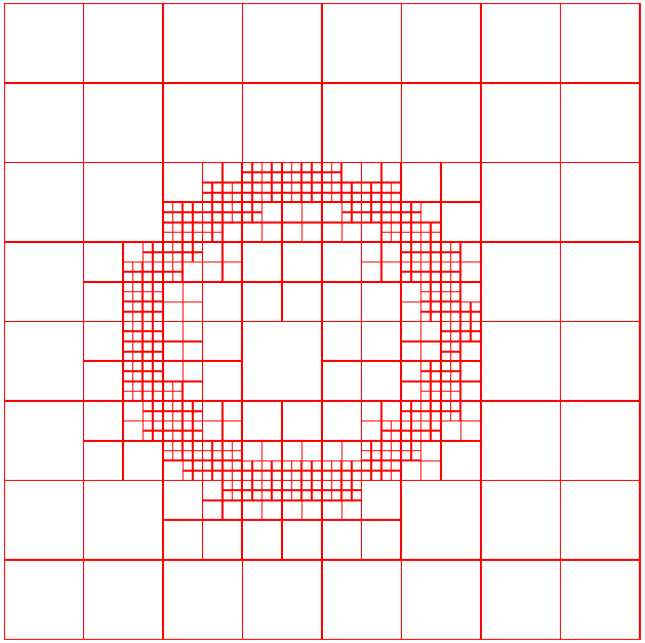
Greedy Algorithm

The cell structure is not modified!

May be, we can improve it

Initialization:
Take a decomposition in cells

Loop:
Perform Greedy Algorithm
Refine decomposition



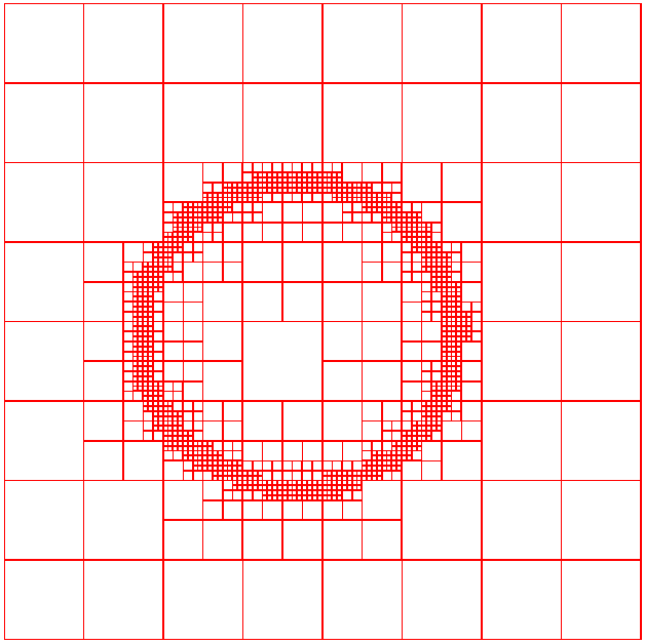
Greedy Algorithm

The cell structure is not modified!

May be, we can improve it

Initialization:
Take a decomposition in cells

Loop:
Perform Greedy Algorithm
Refine decomposition



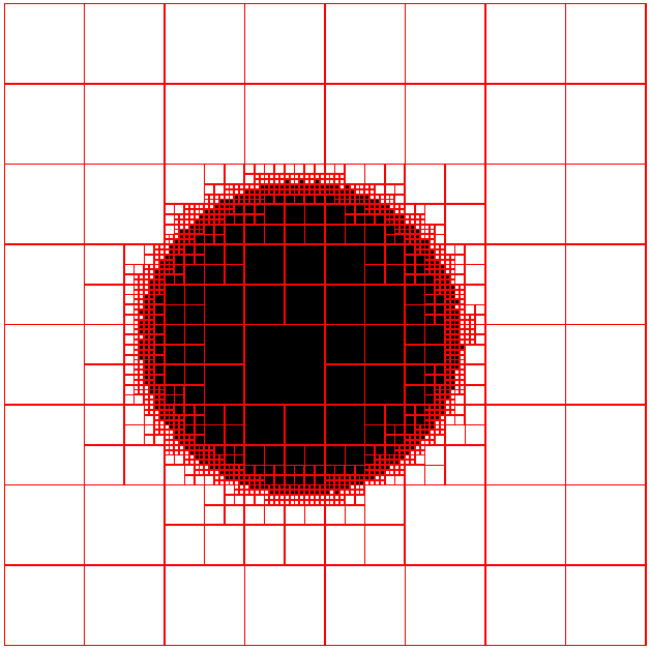
Greedy Algorithm

The cell structure is not modified!

May be, we can improve it

Initialization:
Take a decomposition in cells

Loop:
Perform Greedy Algorithm
Refine decomposition



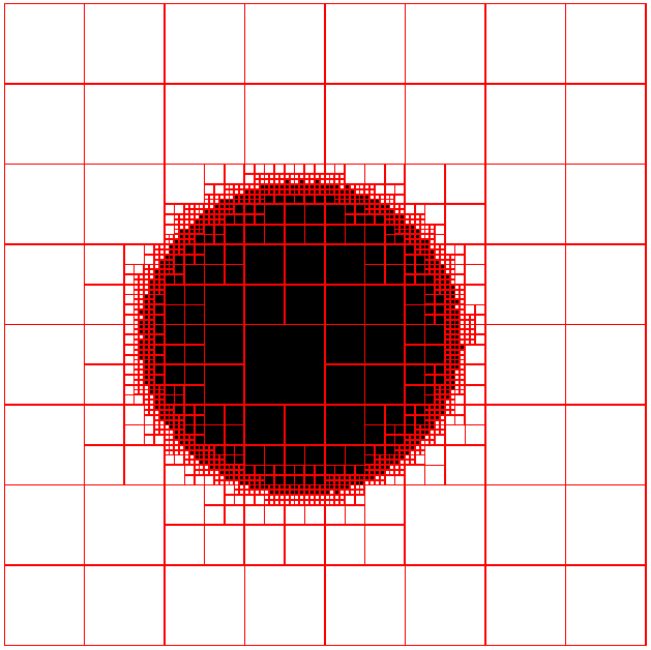
Greedy Algorithm

The cell structure is not modified!

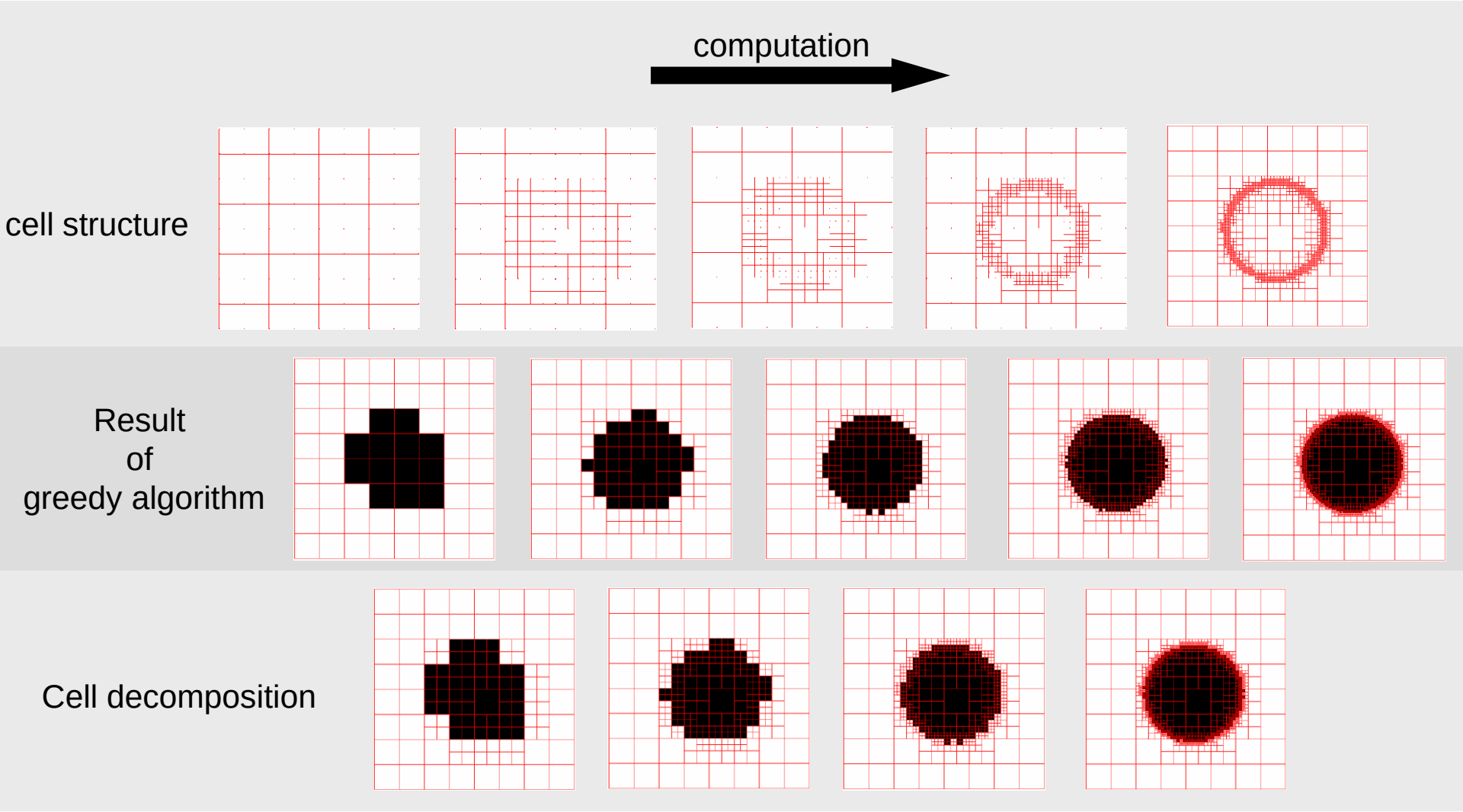
May be, we can improve it

Initialization:
Take a decomposition in cells

Loop:
Perform Greedy Algorithm
Refine decomposition



Multi-resolution Greedy algorithm : we alternate routine greedy algorithm with cell decomposition


computation
cell structure
Result
of
greedy algorithm
Cell decomposition

Next Section

I Back to Radon transform

II Linear Programming

III Energy Minimization

IV Heuristic

V Experiments

Heuristic

In this section, we investigate a heuristic (close to Simulated Annealing)

The algorithm is controlled by a temperature which decreases during the process

At each place, several things can happen :

- a cell can split

- cells with the same color can merge

- the color of a cell may change

Heuristic

- a cell can split

- cells with the same color can merge

- the color of a cell may change

Heuristic

- a cell can split

- cells with the same color can merge

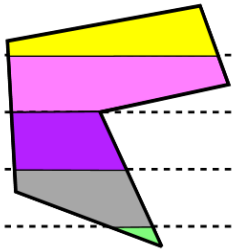
- the color of a cell may change

The probability to change the color of a cell depends on

- the temperature
- ~~the error density~~

Heuristic

Projections of the cell	Errors on projections		
0.0	+ 0.019820	x	0.0
0.0	- 0.052203	x	0.0
0.0	+ 0.045781	x	0.0
0.001231	- 0.010175	x	0.001231
0.002307	- 0.001265	x	0.002307
0.000843	+ 0.021479	x	0.000843
0.000912	- 0.012728	x	0.000912
0.000005	+ 0.022521	x	0.000005
0.0	- 0.030404	x	0.0
0.0	- 0.025168	x	0.0



Sum for all directions

error sum

We compute for each cell

Sum with absolute values

absolute error sum

Heuristic

We compute for each cell

$$\frac{\text{error sum}}{\text{absolute error sum}}$$

We compute their ratio

Its value is between *-1* and *1*

Ratio -1 means deficit of points
Ratio +1 means excess of points
Ratio ~0 means we don't know

The probability to change the color of a cell depends on
- the temperature
- ~~the error density~~ the error ratio

Heuristic

- a cell can split

- cells with the same color can merge

- the color of a cell may change

The probability to split depends also on

- *the temperature*
- the *error ratio* $\dfrac{\textit{error sum}}{\textit{absolute error sum}}$

Heuristic

- a cell can split

- cells with the same color can merge

- the color of a cell may change

The probability to merge depends on

- the temperature

Next Section

I Back to Radon transform

II Linear Programming

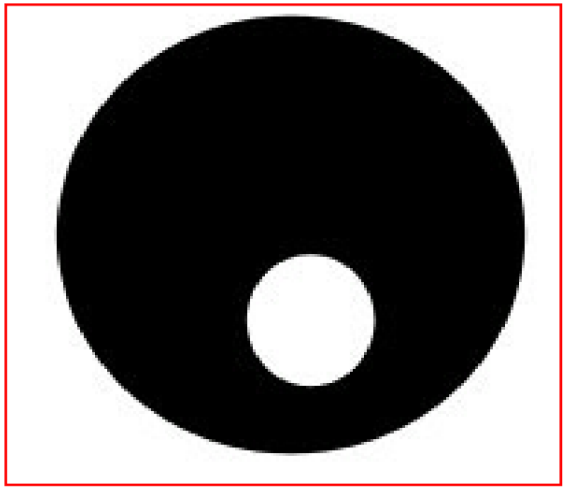
III Energy Minimization

IV Heuristic

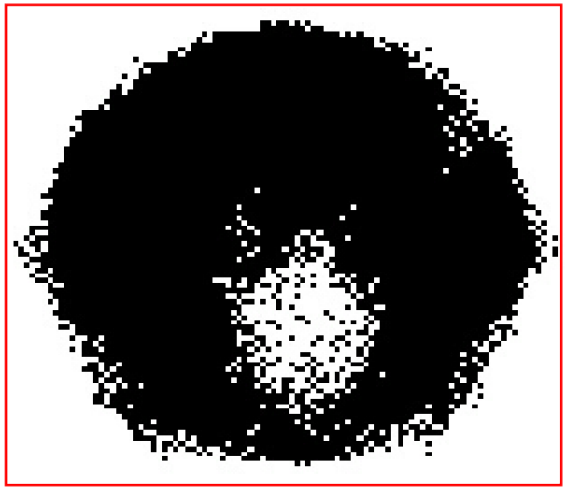
V Experiments

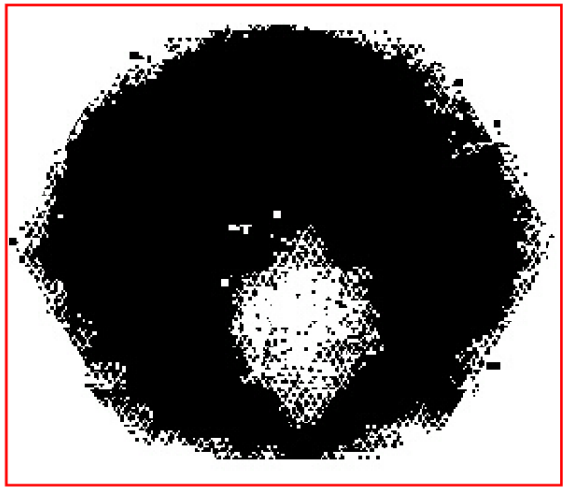
Experiments

Phantom

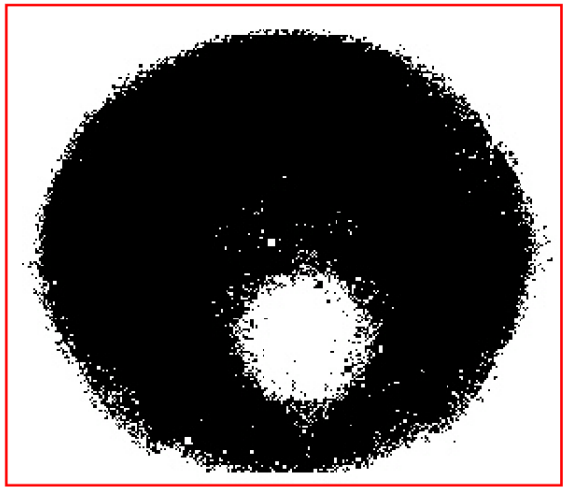


Greedy Algorithm





Multiresolution GA



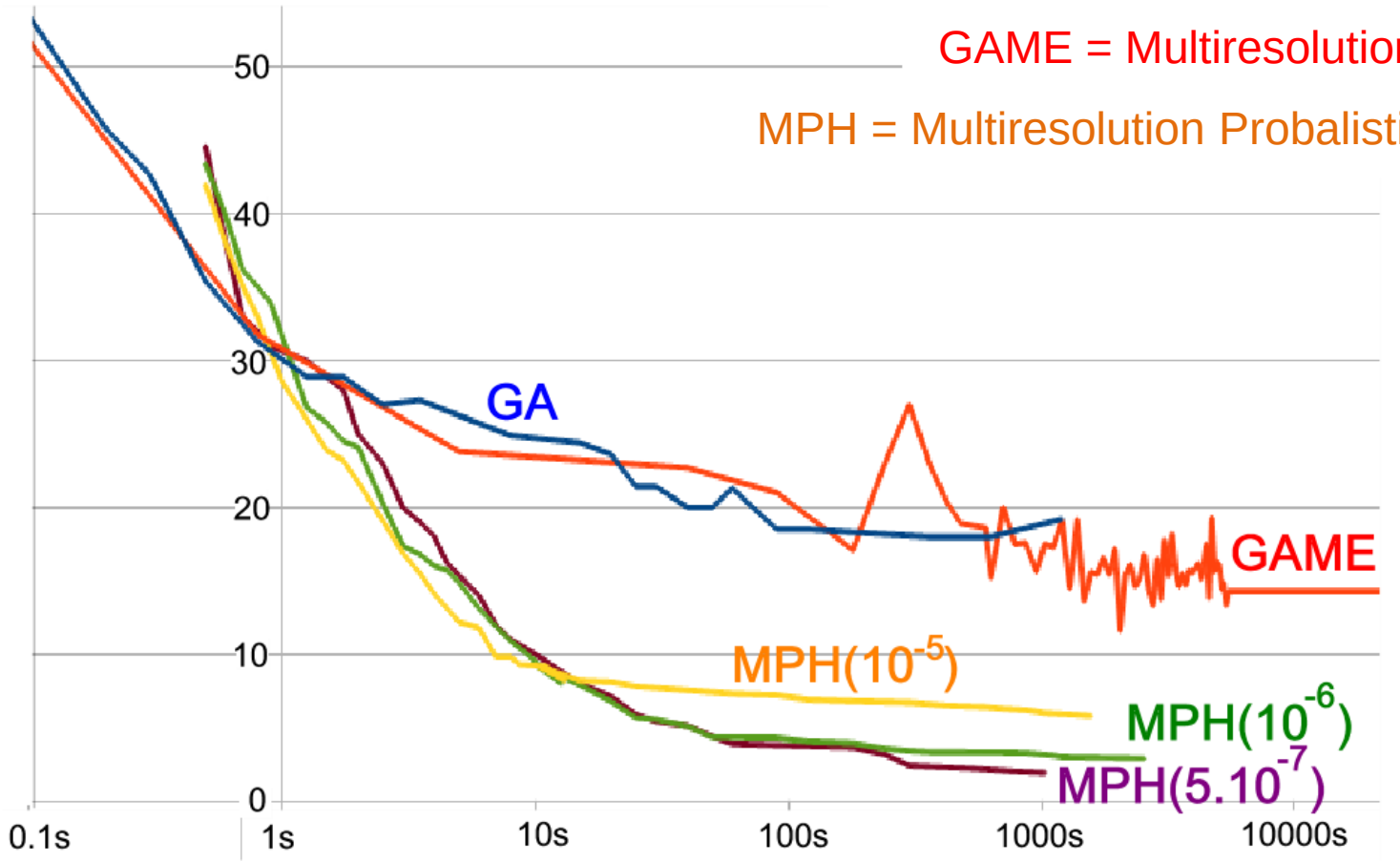
Simulated Annealing

Experiments

GA = Greedy Algorithm

GAME = Multiresolution GA

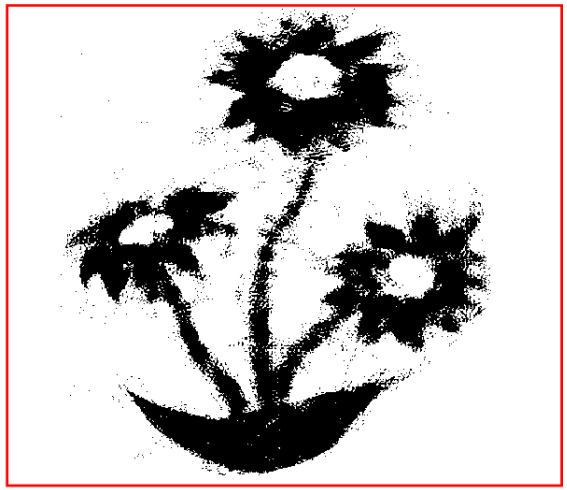
MPH = Multiresolution Probalistic Heuristic

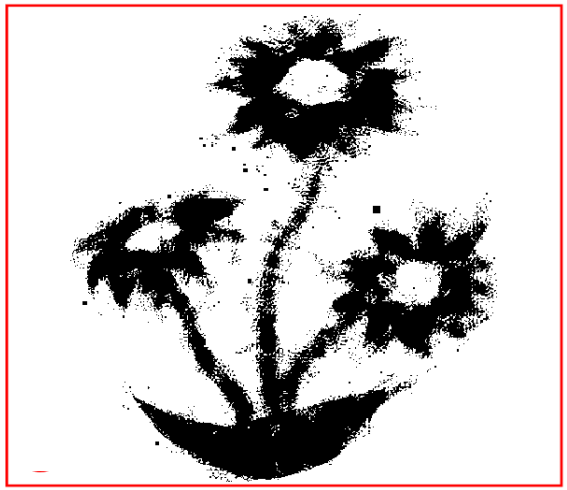


Experiments

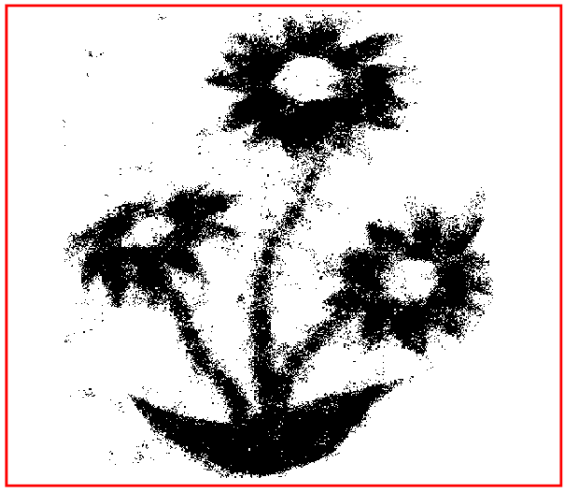
Phantom

Greedy Algorithm



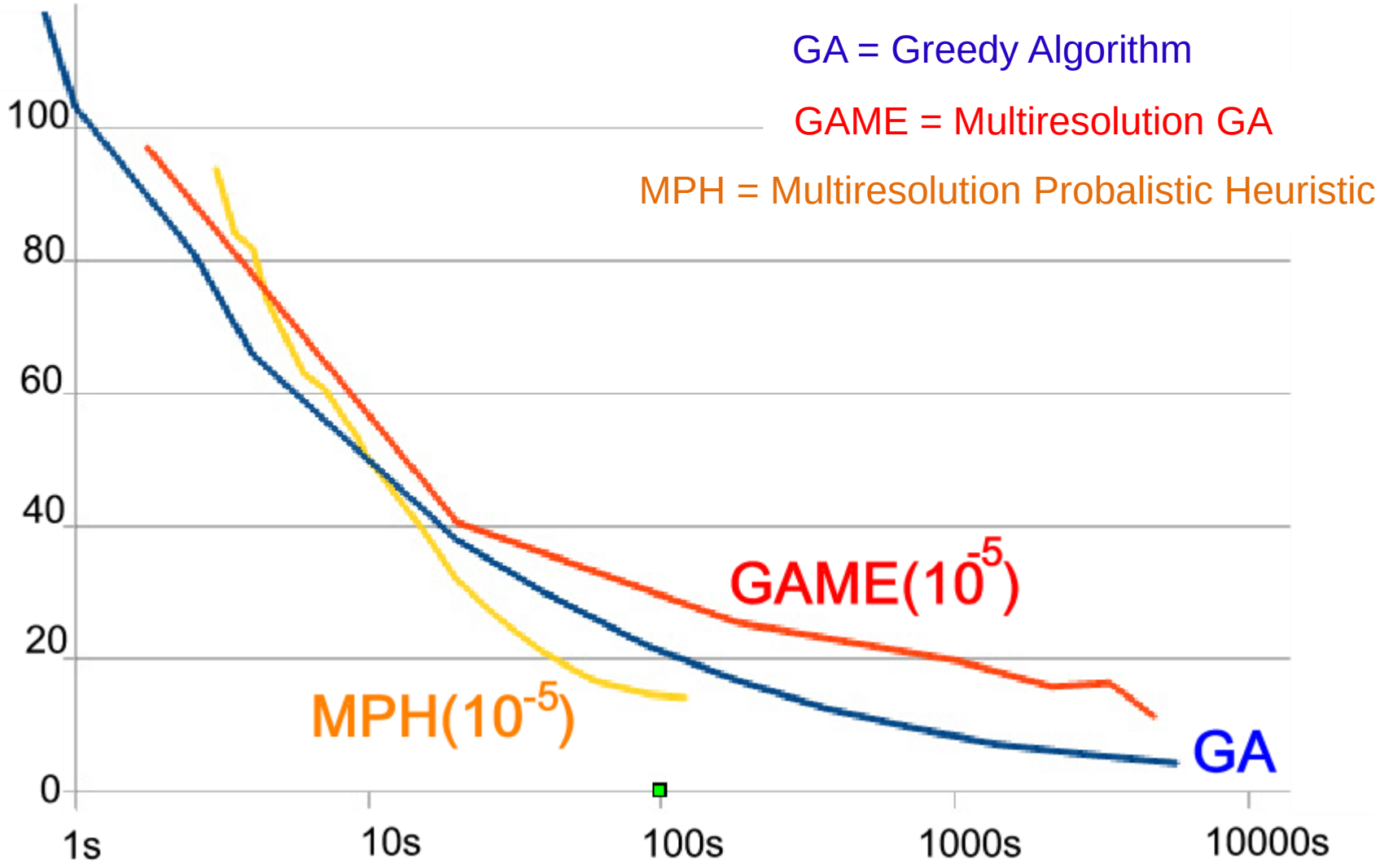


Multiresolution GA



Simulated Annealing

Experiments



Conclusion

Thank you for your attention