WEA 2008

Experimental Evaluation of an Exact Algorithm for the Orthogonal Art Gallery Problem

Marcelo C. Couto, Cid C. de Souza and Pedro J. de Rezende

Institute of Computing
University of Campinas (UNICAMP)

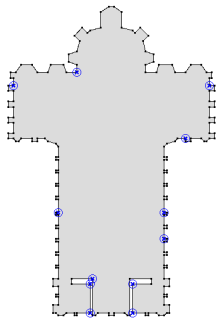
May 30th, 2008

History

- Originally posed by Victor Klee in 1973
  How many guards are sufficient to see every point in the interior of an n-wall Art Gallery room?
- Example: Basilique St. Sernin, Toulouse

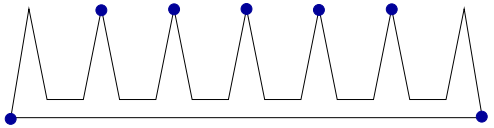
History

- Originally posed by Victor Klee in 1973
  How many guards are sufficient to see every point in the interior of an n-wall Art Gallery room?
- Example: Basilique St. Sernin, Toulouse



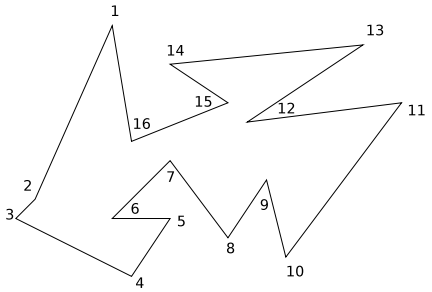
Art Gallery Theorem

- Chvátal proved:
  $\lfloor \frac{n}{3} \rfloor$ *guards are occasionally necessary and always sufficient to cover a polygon of n vertices*



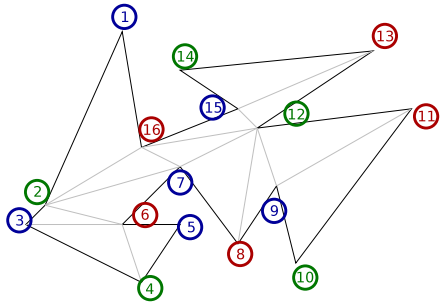
Art Gallery Theorem

- Chvátal proved:
  $\lfloor \frac{n}{3} \rfloor$ *guards are occasionally necessary and always sufficient to cover a polygon of n vertices*
- Fisk had a simpler proof.



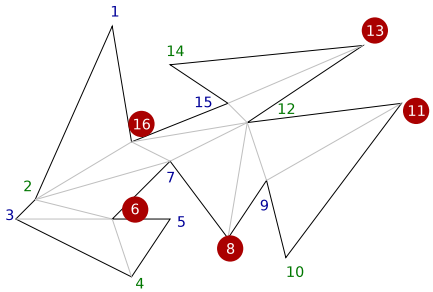
Art Gallery Theorem

- Chvátal proved:
  $\lfloor \frac{n}{3} \rfloor$ *guards are occasionally necessary and always sufficient to cover a polygon of n vertices*
- Fisk had a simpler proof.



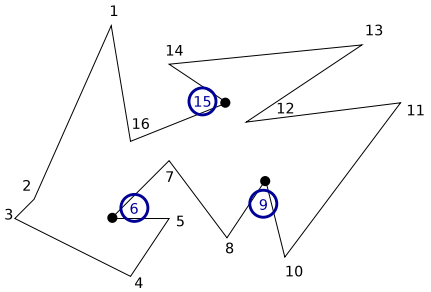
Art Gallery Theorem

- Chvátal proved:
  $\lfloor \frac{n}{3} \rfloor$ *guards are occasionally necessary and always sufficient to cover a polygon of n vertices*
- Fisk had a simpler proof. It leads to placement of $\lfloor \frac{n}{3} \rfloor$ guards



Art Gallery Theorem

- Chvátal proved:
  $\lfloor \frac{n}{3} \rfloor$ *guards are occasionally necessary and always sufficient to cover a polygon of n vertices*
- But it may not the minimum!



Introduction
Modeling
Algorithm
Experimental Setup
Summary
Background

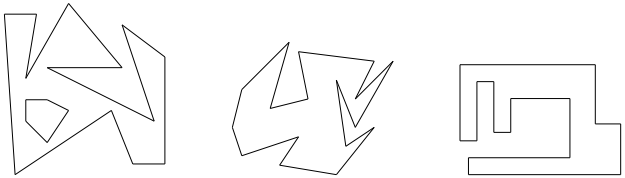
Art Gallery Theorem

- Chvátal proved:
  $\lfloor \frac{n}{3} \rfloor$ *guards are occasionally necessary and always sufficient to cover a polygon of n vertices*
- The Minimization Problem is NP-Hard [Lee & Lin (1985)]

Introduction
Modeling
Algorithm
Experimental Setup
Summary
Background

Variations on the Art Gallery Problem

- Types of polygons

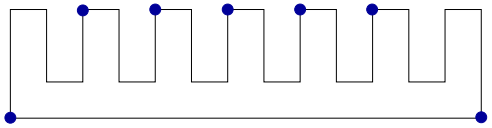


- Types of guards
  - Stationary guards
    - Vertex guards
    - Point guards
  - Mobile guards
    - Edge guards
    - Diagonal guards
- External visibility
- See surveys by O'Rourke, Shermer, Urrutia, Ghosh.

Introduction
Modeling
Algorithm
Experimental Setup
Summary
Background

Orthogonal Art Gallery Problem (OAGP)

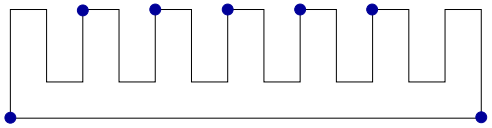
- Kahn *et al.* (1983) proved that $\lfloor \frac{n}{4} \rfloor$ *guards are occasionally necessary and always sufficient to cover an ortho-gon of n vertices*
  - Necessity



  - Sufficiency: four-color a quadrilateralization of P (mimics Fisk's proof).

Orthogonal Art Gallery Problem (OAGP)

- Kahn *et al.* (1983) proved that $\lfloor \frac{n}{4} \rfloor$ *guards are occasionally necessary and always sufficient to cover an ortho-gon of n vertices*
  - Necessity



  - Sufficiency: four-color a quadrilateralization of P (mimics Fisk's proof).
- The minimization problem is NP-Hard, too [Schuchardt *et al.* (1995)].

Some known Results

- log n-approximation algorithm [Ghosh (1987)]
- log c_{opt}-approximation algorithm [Efrat and Har-Peled (1998)]
- Eidenbenz (2002) showed AGP is APX-hard (for minimum vertex, point and edge-guards)
- Erdem and Sclaroff (2006) approximation algorithm through fixed grid using an Integer Programming heuristic
- Tomás *et al.* (2006): an exact constraint programming approach through successive approximations

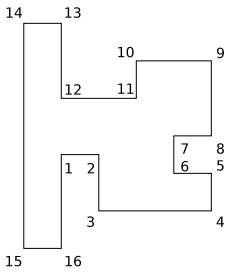
What we set out to do

To study the minimization version of OAGP for polygons without holes and stationary vertex-guards:

- To design an exact algorithm to optimally solve the problem through discretization
- To show the practical viability of the algorithm
- To study various discretization strategies
- To conduct and analyze experimental results

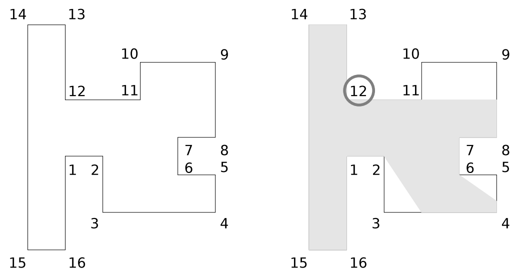
Vertex Guard Set G

- A Vertex Guard Set G is any subset of vertices such that $\bigcup_{g \in G} V(g) = P$



Vertex Guard Set G

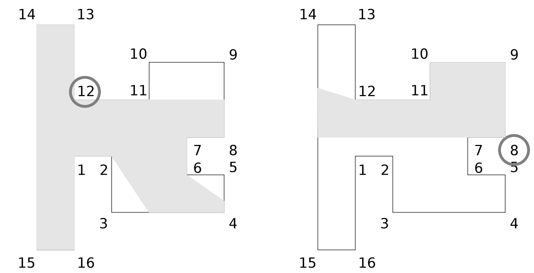
- A Vertex Guard Set G is any subset of vertices such that $\bigcup_{g \in G} V(g) = P$



Introduction
Modeling
Algorithm
Experimental Setup
Summary

Vertex Guard Set G

- A Vertex Guard Set G is any subset of vertices such that $\bigcup_{g \in G} V(g) = P$

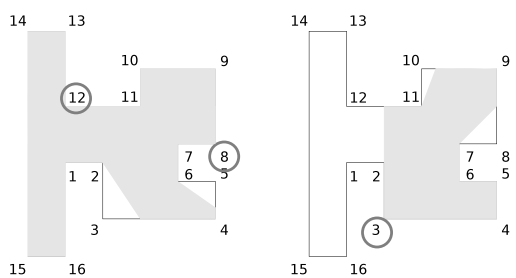


M.C. Couto, C.C. de Souza & P.J. de Rezende
An Exact Algorithm for the Art Gallery Problem

Introduction
Modeling
Algorithm
Experimental Setup
Summary

Vertex Guard Set G

- A Vertex Guard Set G is any subset of vertices such that $\bigcup_{g \in G} V(g) = P$

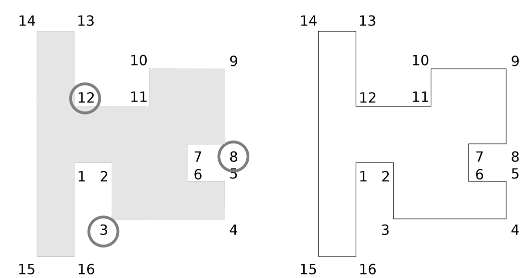


M.C. Couto, C.C. de Souza & P.J. de Rezende
An Exact Algorithm for the Art Gallery Problem

Introduction
Modeling
Algorithm
Experimental Setup
Summary

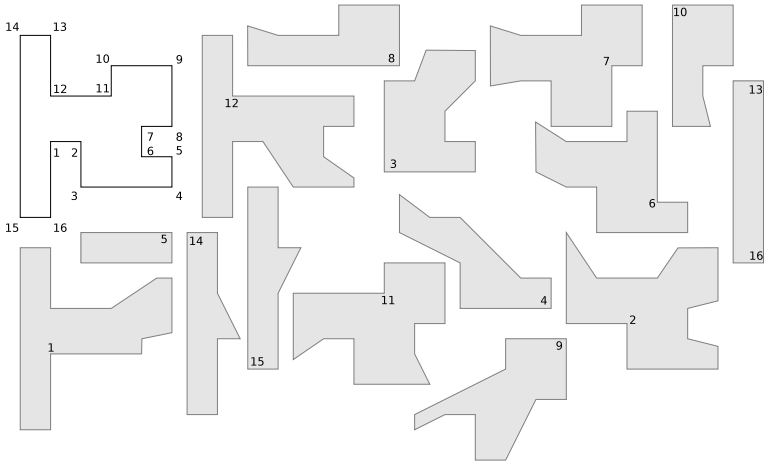
Vertex Guard Set G

- A Vertex Guard Set G is any subset of vertices such that $\bigcup_{g \in G} V(g) = P$



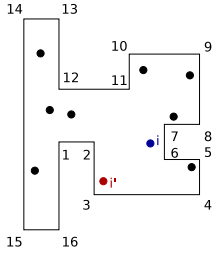
M.C. Couto, C.C. de Souza & P.J. de Rezende
An Exact Algorithm for the Art Gallery Problem

Set Cover Problem



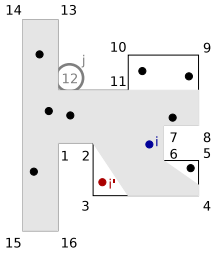
The Discretized Problem

- In order to solve OAGP we:
  - Model the problem as a discrete combinatorial problem
  - Discretize P into a set of points $D(P)$
  - Encode the visibility of $D(P)$ by the vertices, on a binary matrix:
    For each point $i, i' \in D(P)$,
    $a_{ij} = 1$ if i is visible from $j \in V_P$ and
    $a_{i'j} = 0$ if i' is not visible from $j \in V_P$



The Discretized Problem

- In order to solve OAGP we:
  - Model the problem as a discrete combinatorial problem
  - Discretize P into a set of points $D(P)$
  - Encode the visibility of $D(P)$ by the vertices, on a binary matrix:
    For each point $i, i' \in D(P)$,
    $a_{ij} = 1$ if i is visible from $j \in V_P$ and
    $a_{i'j} = 0$ if i' is not visible from $j \in V_P$



Formulation

- The set $Z = \{j \in V_P \mid x_j = 1\}$ is called a solution set
- (1) states that each point $i \in D(P)$ is visible from at least one selected guard
- The objective function minimizes the cardinality z of the solution set Z
- Solution might not form a guard set for P

$$z = \min \sum_{j \in V_P} x_j$$

$$\text{s.t.} \sum_{j \in V_P} a_{ij} x_j \geq 1, \forall i \in D(P) \quad (1)$$

$$x_j = \begin{cases} 1, & \text{if } j \text{ belongs to the solution} \\ 0, & \text{otherwise.} \end{cases}$$

$$a_{ij} = \begin{cases} 1, & \text{if } i \in V(j) \\ 0, & \text{otherwise.} \end{cases}$$

Initial Discretization of P

- The number of uncovered regions depends on the choice of $D(P)$
  - Tradeoff between speed and accuracy
  - We should generate a small number of constraints

Strategy (betting on)

Discretization strategies

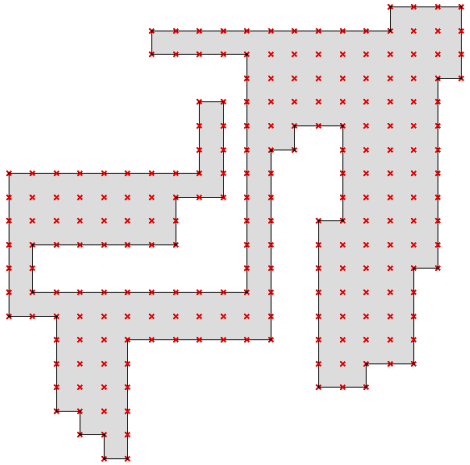
- The number of uncovered regions depends on the choice of $D(P)$
  - Tradeoff between speed and accuracy
  - We should generate a small number of constraints

Strategy (betting on)

- **Regular grid** (density)
  [1]: up to 200 vertices with few iterations in < 100 secs
- **Induced grid** (reflex angles cause hidden areas)
- **Just vertices** (convex vertices identify hard to guard areas)
- (Based on) **AVP**s (ingenious but has costly preprocessing)

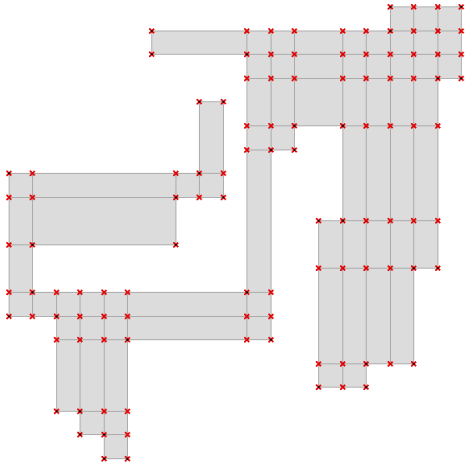
Discretization strategies

- **Regular grid** (density)



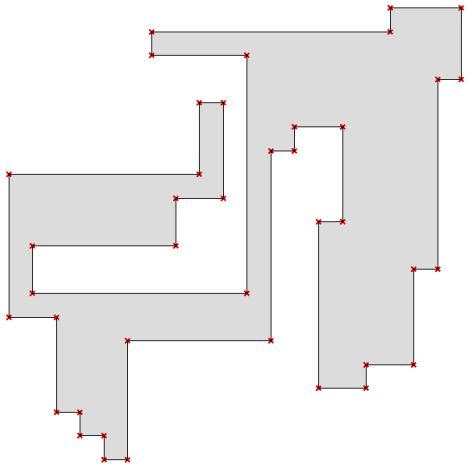
Discretization strategies

- **Induced grid** (reflex angles cause hidden areas)



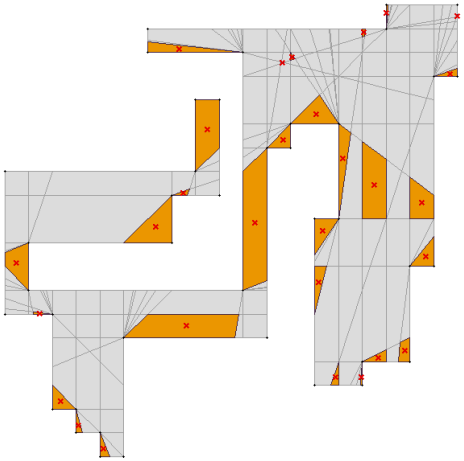
Discretization strategies

- **Just vertices** (convex vertices identify hard to guard areas)



Discretization strategies

- (Based on) **AVP**s (ingenious but has costly preprocessing)



Viable Solution

- A solution Z of the discretized problem is *viable* if Z is a vertex guard set for P

Theorem

Let Z be an optimal solution of an instance $I(P, D(P))$ of the discretized Orthogonal Art Gallery problem. If Z is viable then Z is optimal for the original problem.

Proof. Let Z^* be an optimal vertex guard set for P. Since Z^* is also a vertex guard cover for $D(P)$, we must have $|Z^*| \geq |Z|$. On the other hand, since Z^* is viable, it follows that $|Z| \geq |Z^*|$. □

Preprocessing Phase

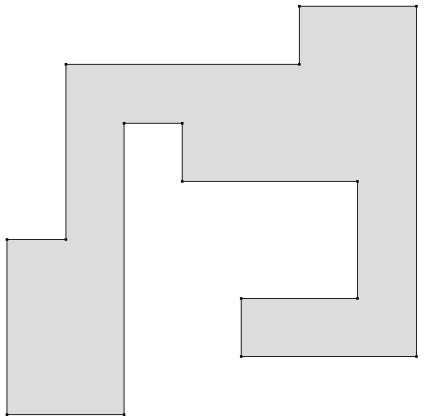
- Build $D(P)$: an initial discretization of P
- Compute the visibility region $V(u)$ for each vertex u
- Locate each point $p \in D(P)$ w.r.t. each visibility region
- Return the initial formulation for the Set Cover Problem

Solution Phase

1: **repeat**
2: $Z \leftarrow$ Solution(SetCover($P, D(P)$));
3: **for** each uncovered region R **do**
4: $c \leftarrow$ centroid of R;
5: $D(P) \leftarrow D(P) \cup \{c\}$;
6: Add a new row, r, to the set of constraints corresponding to the new uncovered point c:
$\sum a_{rj} x_j \geq 1$ where, $\forall j \in V_P$,
$a_{rj} \leftarrow$ **Boolean**$(c \in V(j))$;
7: **end for**
8: **until** Z is viable

Iterations

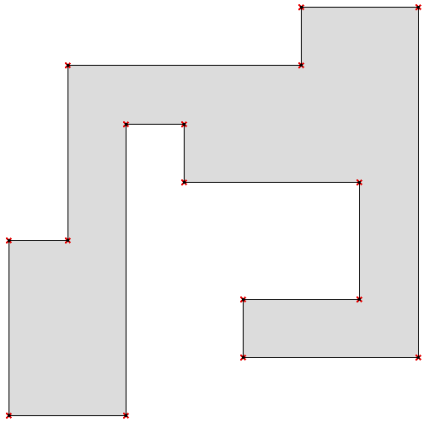
1: **repeat**
2: $Z \leftarrow$ Solution(SetCover($P, D(P)$));
3: **for** each uncovered region R **do**
4: $c \leftarrow$ centroid of R;
5: $D(P) \leftarrow D(P) \cup \{c\}$;
6: Add a new row, r, to the set of constraints corresponding to the new uncovered point c:
$\sum a_{rj} x_j \geq 1$ where, $\forall j \in V_P$,
$a_{rj} \leftarrow$ **Boolean**$(c \in V(j))$;
7: **end for**
8: **until** Z is viable



Polygon

Iterations

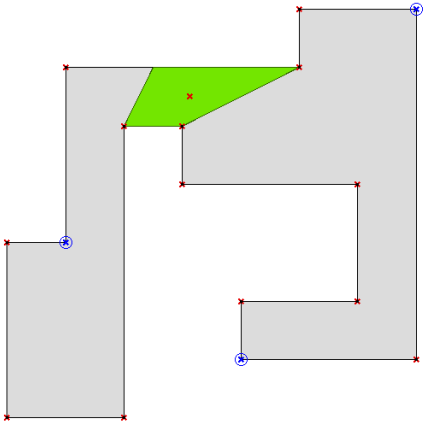
1: **repeat**
2: $Z \leftarrow$ Solution(SetCover($P, D(P)$));
3: **for** each uncovered region R **do**
4: $c \leftarrow$ centroid of R;
5: $D(P) \leftarrow D(P) \cup \{c\}$;
6: Add a new row, r, to the set of constraints corresponding to the new uncovered point c:
$\sum a_{rj} x_j \geq 1$ where, $\forall j \in V_P$,
$a_{rj} \leftarrow$ **Boolean**$(c \in V(j))$;
7: **end for**
8: **until** Z is viable



Initial Discretization

Iterations

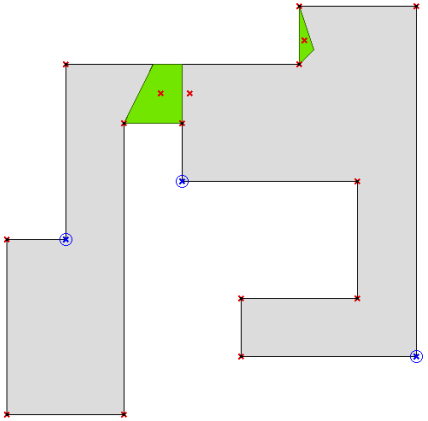
1: **repeat**
2: $Z \leftarrow$ Solution(SetCover($P, D(P)$));
3: **for** each uncovered region R **do**
4: $c \leftarrow$ centroid of R;
5: $D(P) \leftarrow D(P) \cup \{c\}$;
6: Add a new row, r, to the set of constraints corresponding to the new uncovered point c:
$\sum a_{rj} x_j \geq 1$ where, $\forall j \in V_P$,
$a_{rj} \leftarrow$ **Boolean**$(c \in V(j))$;
7: **end for**
8: **until** Z is viable



First Iteration

Iterations

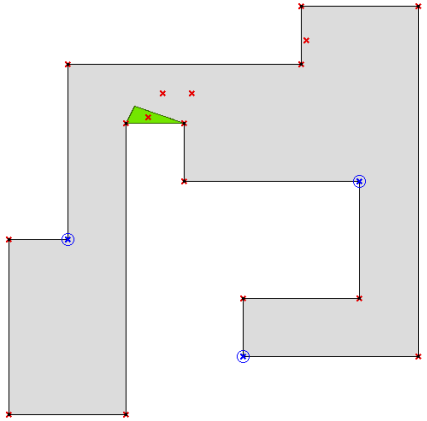
1: **repeat**
2: $Z \leftarrow$ Solution(SetCover($P, D(P)$));
3: **for** each uncovered region R **do**
4: $c \leftarrow$ centroid of R;
5: $D(P) \leftarrow D(P) \cup \{c\}$;
6: Add a new row, r, to the set of constraints corresponding to the new uncovered point c: $\sum a_{rj} x_j \geq 1$ where, $\forall j \in V_P$, $a_{rj} \leftarrow$ **Boolean**($c \in V(j)$);
7: **end for**
8: **until** Z is viable



Second Iteration

Iterations

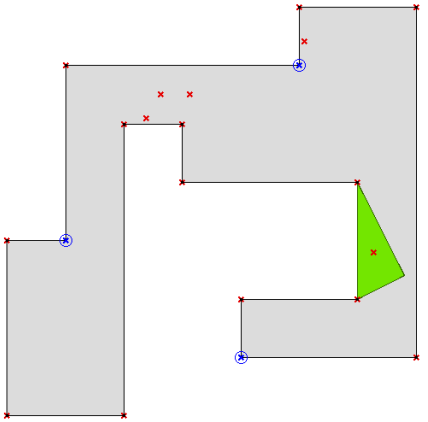
1: **repeat**
2: $Z \leftarrow$ Solution(SetCover($P, D(P)$));
3: **for** each uncovered region R **do**
4: $c \leftarrow$ centroid of R;
5: $D(P) \leftarrow D(P) \cup \{c\}$;
6: Add a new row, r, to the set of constraints corresponding to the new uncovered point c: $\sum a_{rj} x_j \geq 1$ where, $\forall j \in V_P$, $a_{rj} \leftarrow$ **Boolean**($c \in V(j)$);
7: **end for**
8: **until** Z is viable



Third Iteration

Iterations

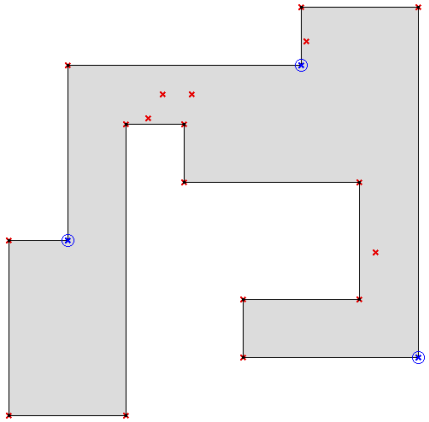
1: **repeat**
2: $Z \leftarrow$ Solution(SetCover($P, D(P)$));
3: **for** each uncovered region R **do**
4: $c \leftarrow$ centroid of R;
5: $D(P) \leftarrow D(P) \cup \{c\}$;
6: Add a new row, r, to the set of constraints corresponding to the new uncovered point c: $\sum a_{rj} x_j \geq 1$ where, $\forall j \in V_P$, $a_{rj} \leftarrow$ **Boolean**($c \in V(j)$);
7: **end for**
8: **until** Z is viable



Fourth Iteration

Iterations

1: **repeat**
2: $Z \leftarrow$ Solution(SetCover($P, D(P)$));
3: **for** each uncovered region R **do**
4: $c \leftarrow$ centroid of R;
5: $D(P) \leftarrow D(P) \cup \{c\}$;
6: Add a new row, r, to the set of constraints corresponding to the new uncovered point c: $\sum a_{rj} x_j \geq 1$ where, $\forall j \in V_P$, $a_{rj} \leftarrow$ **Boolean**($c \in V(j)$);
7: **end for**
8: **until** Z is viable



Fifth Iteration: viable solution

What Remains To Be Seen

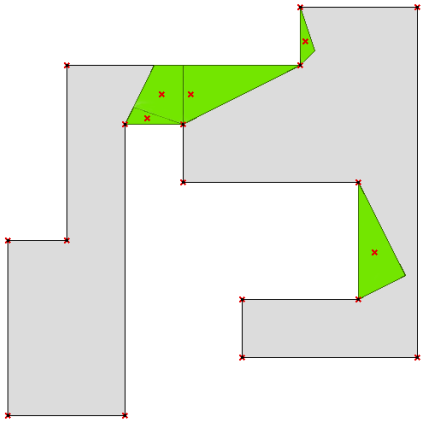
- That the algorithm converges
- How to improve the initial discretization strategy to reduce the number of iterations
- Is it possible to reduce the iterations to one? At what cost?
- What types of polygons we considered
- An analysis of the experimental results

How many uncovered regions can there be?

- Convergence will follow from an upper bound on the number of uncovered regions
- As a byproduct, we might see how to improve the initial discretization strategy

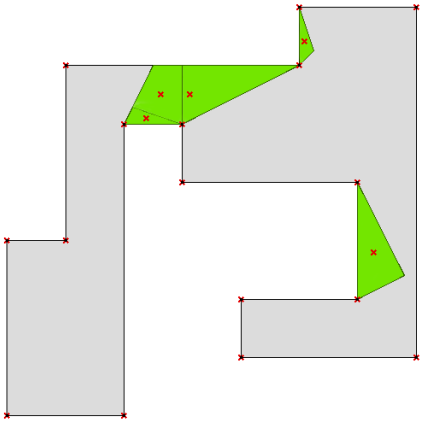
How many uncovered regions can there be?

- Could we predict these regions *before* running the algorithm?



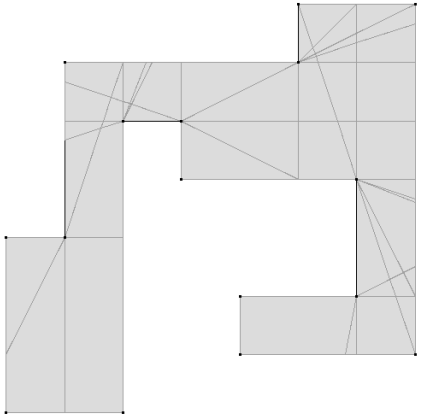
How many uncovered regions can there be?

- Better yet: could we pre-identify *all* relevant regions regardless of discretization?



How many uncovered regions can there be?

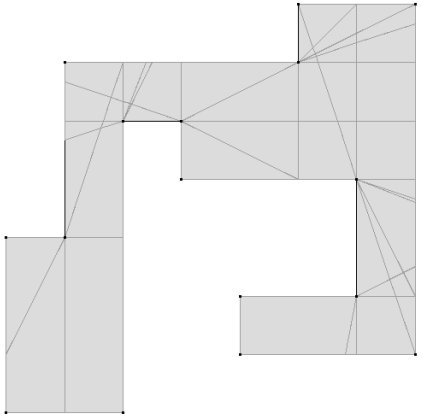
- Better yet: could we pre-identify *all* relevant regions regardless of discretization?
- They are (unions of) faces of the visibility arrangement $\propto \{P, \text{guard-candidates}\}$



Introduction
Modeling
Algorithm
Experimental Setup
Summary
Optimality
Preprocessing Phase
Solution Phase
Convergence of the Algorithm
Improving the Initial Discretization

How many uncovered regions can there be?

- Better yet: could we pre-identify *all* relevant regions regardless of discretization?
- They are (unions of) faces of the visibility arrangement $\propto \{P, \text{guard-candidates}\}$
- We call these faces *Atomic Visibility Regions* (AVPs). There are $O(n^3)$ of them [Bose *et al.* (2002)].

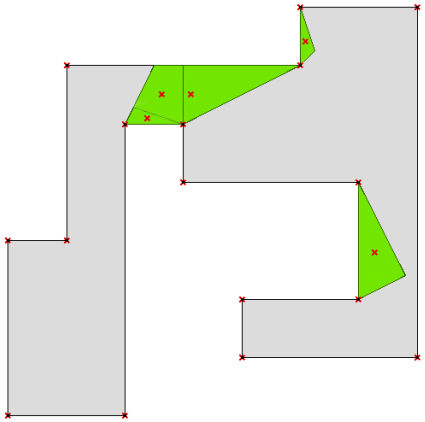


M.C. Couto, C.C. de Souza & P.J. de Rezende — An Exact Algorithm for the Art Gallery Problem

Introduction
Modeling
Algorithm
Experimental Setup
Summary
Optimality
Preprocessing Phase
Solution Phase
Convergence of the Algorithm
Improving the Initial Discretization

How many uncovered regions can there be?

- Better yet: could we pre-identify *all* relevant regions regardless of discretization?
- They are (unions of) faces of the visibility arrangement $\propto \{P, \text{guard-candidates}\}$
- We call these faces *Atomic Visibility Regions* (AVPs). There are $O(n^3)$ of them [Bose *et al.* (2002)].
- Any uncovered region contains an AVP.
- Coverage of a centroid $\times$ implies its AVP is covered.

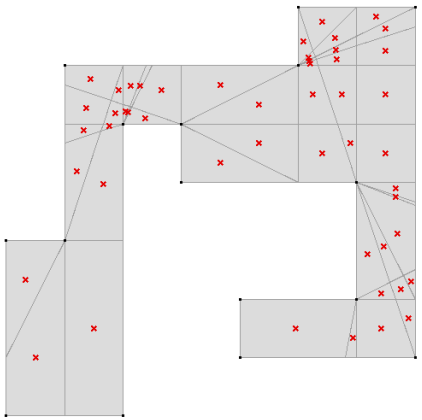


M.C. Couto, C.C. de Souza & P.J. de Rezende — An Exact Algorithm for the Art Gallery Problem

Introduction
Modeling
Algorithm
Experimental Setup
Summary
Optimality
Preprocessing Phase
Solution Phase
Convergence of the Algorithm
Improving the Initial Discretization

How many uncovered regions can there be?

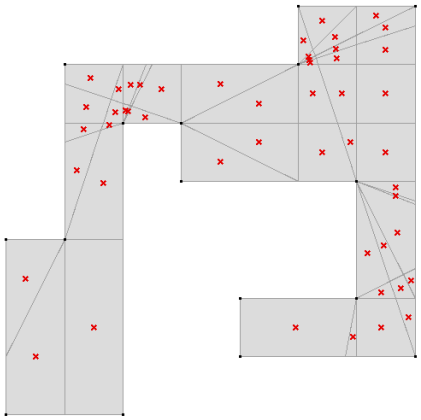
- Better yet: could we pre-identify *all* relevant regions regardless of discretization?
- They are (unions of) faces of the visibility arrangement $\propto \{P, \text{guard-candidates}\}$
- We call these faces *Atomic Visibility Regions* (AVPs). There are $O(n^3)$ of them [Bose *et al.* (2002)].
- Any uncovered region contains an AVP.
- Coverage of a centroid $\times$ implies its AVP is covered.
- This proves that the algorithm converges!



M.C. Couto, C.C. de Souza & P.J. de Rezende — An Exact Algorithm for the Art Gallery Problem

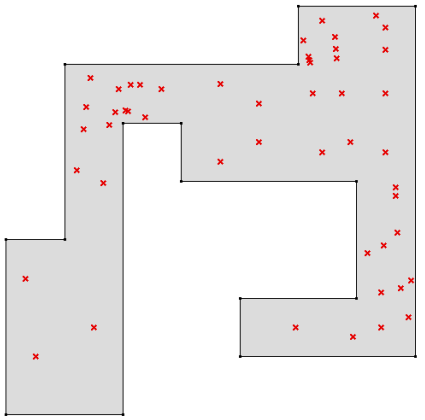
Improving the Initial Discretization

- Coverage of a centroid × implies its AVP is covered.



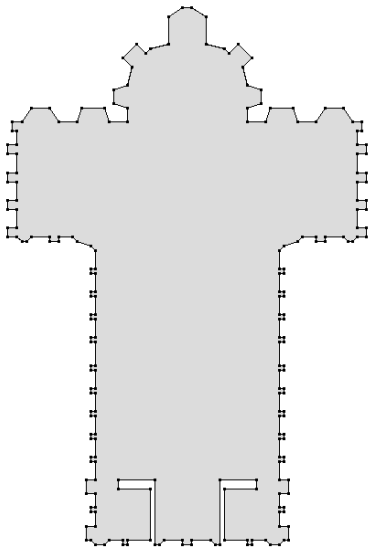
Improving the Initial Discretization

- Coverage of a centroid × implies its AVP is covered.
- This means that we can solve the Art Gallery problem through a single instance of SCP.
- But this may lead to a very large instance of SCP.



Improving the Initial Discretization

- Coverage of a centroid × implies its AVP is covered.
- This means that we can solve the Art Gallery problem through a single instance of SCP.
- But this may lead to a very large instance of SCP.

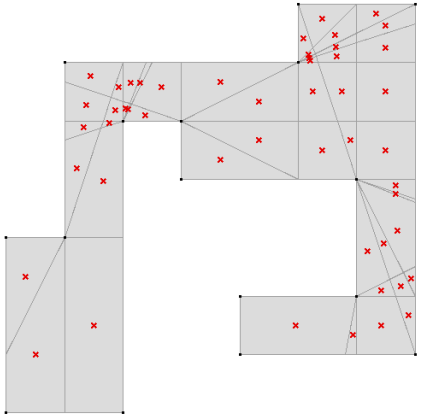


Improving the Initial Discretization

- Coverage of a centroid × implies its AVP is covered.
- This means that we can solve the Art Gallery problem through a single instance of SCP.
- But this may lead to a **very large** instance of SCP.

Improving the Initial Discretization

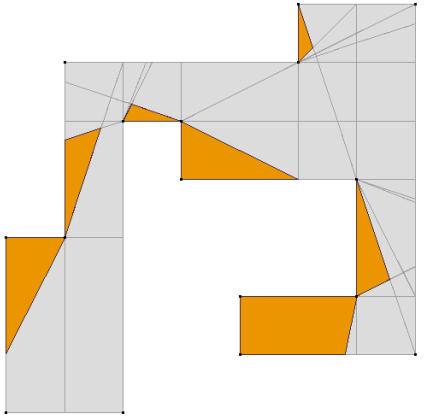
- Can we do better?



Improving the Initial Discretization

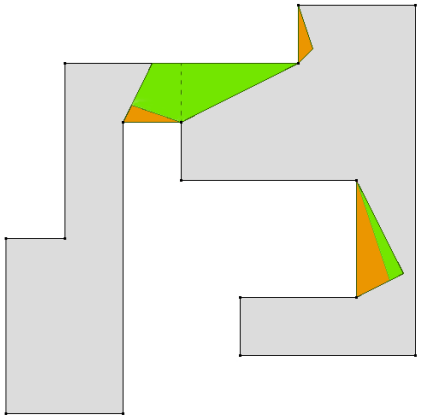
- Can we do better?
- Consider a reduced set of AVPs: "Shadow AVPs"

  An AVP $\mathcal{V}$ is a *shadow AVP* if it is not visible from any of the vertices whose (proper) visibility edges generated $\mathcal{V}$.



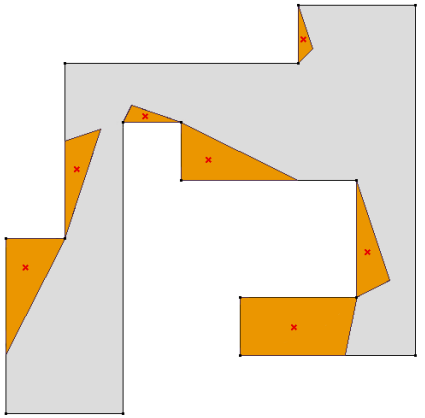
Improving the Initial Discretization

- Can we do better?
- Consider a reduced set of AVPs: "Shadow AVPs"
- Any uncovered region contains a *shadow* AVP! [see paper]



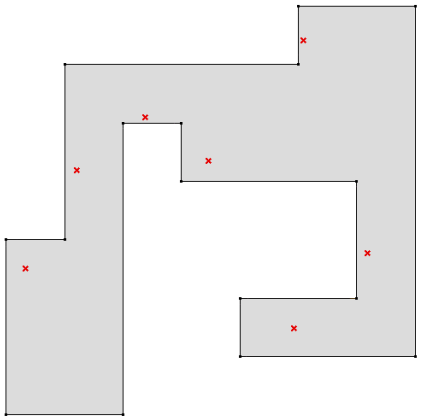
Improving the Initial Discretization

- Can we do better?
- Consider a reduced set of AVPs: "Shadow AVPs"
- Any uncovered region contains a *shadow* AVP! [see paper]
- So, we *can* do better: solve one SCP just for centroids of *shadow AVPs*.



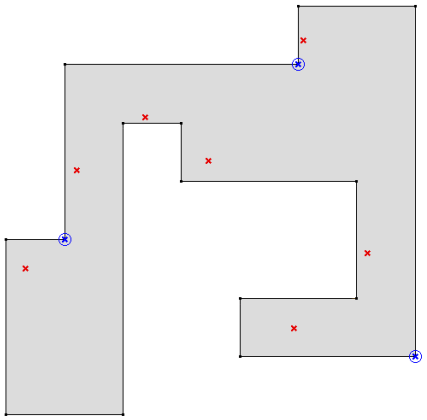
Improving the Initial Discretization

- Can we do better?
- Consider a reduced set of AVPs: "Shadow AVPs"
- Any uncovered region contains a *shadow* AVP! [see paper]
- So, we *can* do better: solve one SCP just for centroids of *shadow AVPs*.
- *Reduced AVPs* strategy leads to a single iteration



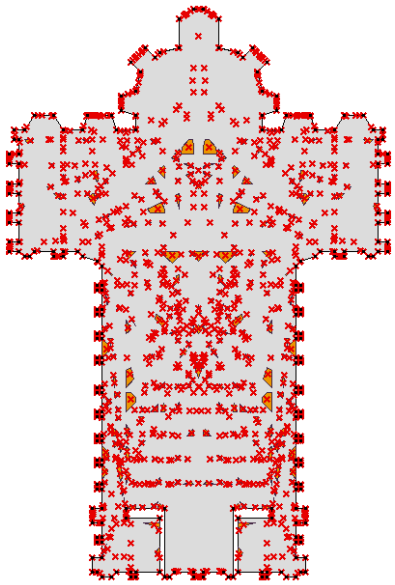
Improving the Initial Discretization

- Can we do better?
- Consider a reduced set of AVPs: "Shadow AVPs"
- Any uncovered region contains a *shadow* AVP! [see paper]
- So, we *can* do better: solve one SCP just for centroids of *shadow AVPs*.
- *Reduced AVPs* strategy leads to a single iteration: guaranteed!



Improving the Initial Discretiza

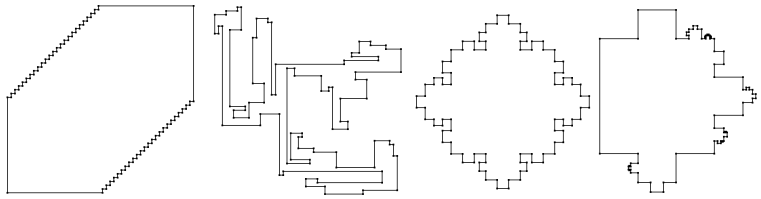
- Can we do better?
- Consider a reduced set of AVPs: "Shadow AVPs"
- Any uncovered region contains a *shadow* AVP! [see paper]
- So, we *can* do better: solve one SCP just for centroids of *shadow AVPs*.
- *Reduced AVPs* strategy leads to a single iteration: guaranteed!



Instances

- 1833 instances of orthogonal polygons without holes (20 to 1000 vertices)
- Polygon Classes:

Large Area || Random || Complete von Koch || Random von Koch



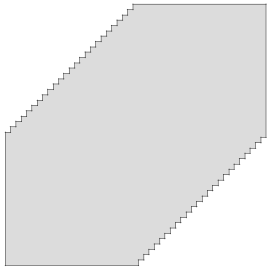
- Available at:
  `www.ic.unicamp.br/~cid/Problem-instances/Art-Gallery/`
  (with over 10000 other instances)

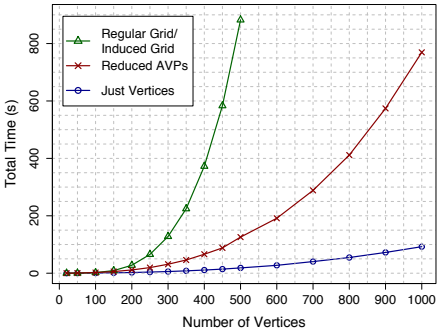
Computational Environment

- The implementation was done in C++ on top of CGAL 3.2.1
- The Integer Linear Programming solver was Xpress-Optimizer v17.01.02 – to solve instances of the Set Cover Problem
- The algorithm was tested on a PC featuring a Pentium IV at 3.4 GHz and 1 GB of memory.

Total Time

- Large area polygons

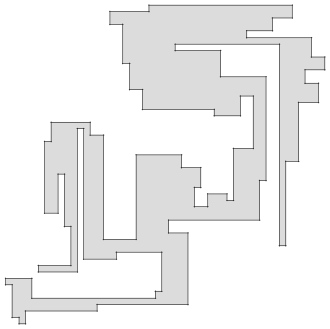




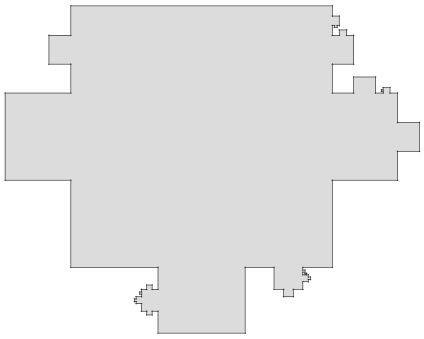
- Five-fold increase in polygon size over Tomás *et al.* [13] within 96 secs

Final Discretization Sizes

- Random polygons

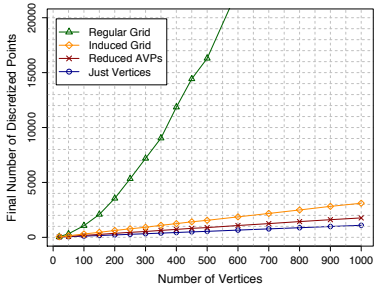


- Random von Koch polygons

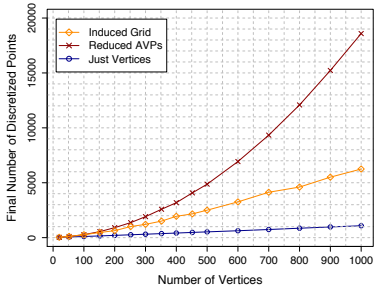


Final Discretization Sizes

- Random polygons



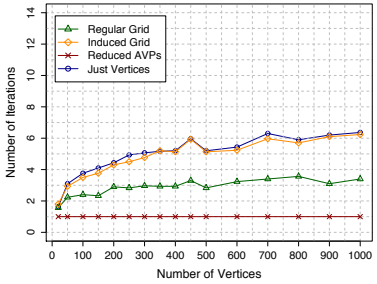
- Random von Koch polygons



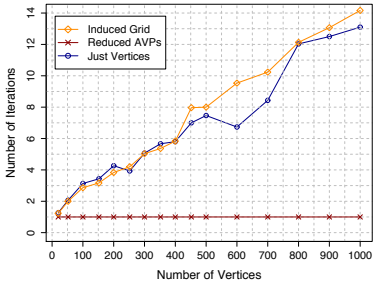
- *Regular grid* is impractical
- *Reduced AVPs* is well-suited for random polygons
- *Just Vertices* is the most economical

Number of iterations

- Random polygons



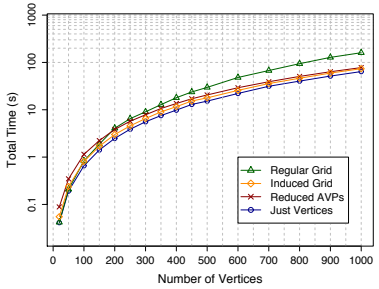
- Random von Koch polygons



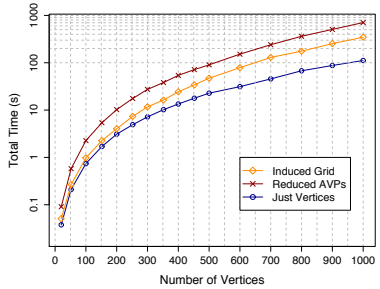
- Obviously one single iteration for Reduced AVPs
- Number of iterations negligible compared to $O(n^3)$ theoretical bound
- For Random von Koch polygons, there is a linear increase, but it is still small

Total Time

- Random polygons



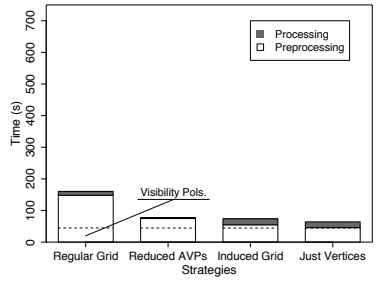
- Random von Koch polygons



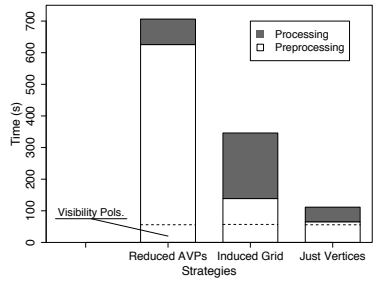
- Note: log×linear graphs
- Random von Koch polygons: three strategies have similarly shaped time curves
- RvKs are harder problems: ∴ *Just Vertices* is a robust strategy

Preprocessing × Processing Time

- Random polygons



- Random von Koch polygons

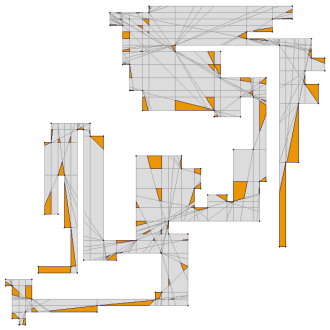


- *Preprocessing* $\in \mathbb{P}$ while *Processing* $\in \mathbb{NP}$-hard and yet, for up to 1000 vertices, *Processing* remains faster

Reduced AVPs is "ingenious but has costly preprocessing": lots of AVPs to sort out

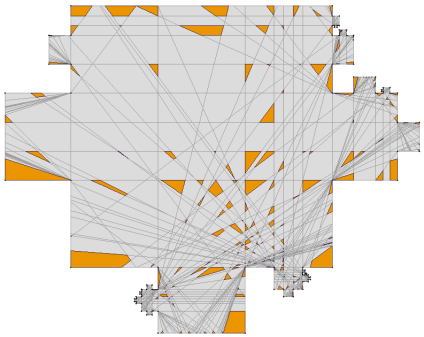
Preprocessing × Processing Time

- Random polygons



- 100 vertices, 2216 edges, 1085 AVPs, 99 shadow AVPs

- Random von Koch polygons



- 100 vertices, 8794 edges, 4420 AVPs, 264 shadow AVPs

Summary

- We obtained an exact algorithm for the *Orthogonal Art Gallery problem*.
- We proved the convergence and correctness of the algorithm.
- Computational tests showed that the algorithm is very efficient in practice.
- Instance sizes of up to 1000 vertices were solved in:

	Polygons Classes			
n	Random	Large Area	RvK	CvK
100	0.65	0.58	0.73	0.97
500	15.21	18.47	22.73	29.35
1000	64.13	92.41	111.55	♯

Total Time (seconds): *Just Vertices* strategy.

What next?

- Simple non-orthogonal polygons without holes (done)
- Multiple coverage (done)
- Solve for point-guards (in progress)
- Solve for edge-guards (next challenge)
- Simple orthogonal polygons with holes
- Extend to general polygons with holes