Algorithms and Heuristics for Deployment of Sensors (“Guards”) for Optimal Coverage

Joe Mitchell

Work joint with Y. Amit, A. Efrat, M. Irfan, J. Iwerks, J. Kim, E. Packer

STONY BROOK
STATE UNIVERSITY OF NEW YORK

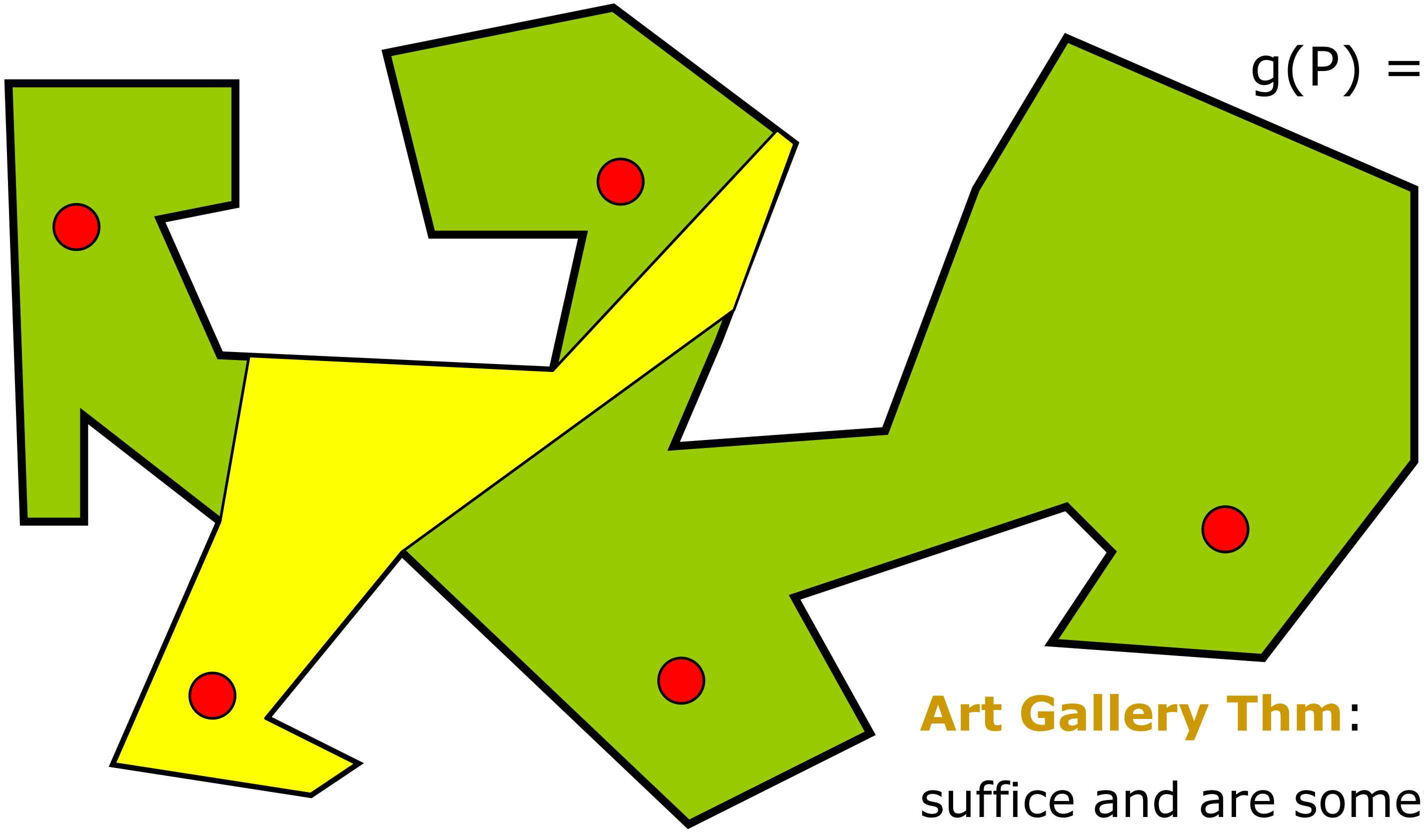
Stationary Guards

The Art Gallery Problem

Determine a small set of "guards" to see all of a given n-vertex polygon P

NP-hard, even in simple polygon

g(P) = min # guards



Art Gallery Thm: $\lfloor \frac{n}{3} \rfloor$ guards suffice and are sometimes necessary

Motivation: Sensor coverage, security

Experimental Investigation [Amit, M, Packer]

- Propose several heuristics for computing guards

- Experimental analysis and comparison

- Compute both upper bounds and lower bounds on OPT, so we can bound how close to OPT we get

- Conclude: heuristics work well in practice:
  - Either find OPT solution or close to optimal
  - Almost always 2-approx
    (always for “random” polygons)

Related Work

- Combinatorics: Lots!

Art Gallery Thm: $\lfloor \frac{n}{3} \rfloor$ guards suffice and are sometimes necessary

- Approximation algorithms for discrete candidate sets (vertex guards, grid-point guards, etc):
  - O(log n)-approx: set cover (greedy) [G87]
  - O(log k)-approx: reweighting ([Cl,BG]) [EH03,GL01]
  - O(1)-approx in special cases:
    - 1.5D terrains (best: 4-approx) [BKM05,K06,EKMMS08]
    - Monotone polygons [Ni05]
- Pseudo-poly O(log k)-approx (poly in spread, n) [DKDS07]
- Exact poly-time solutions:
    - Rectangle visibility in rectilinear polygons [WK06]
    - *Partitioning* P into min # star-shaped pieces [Ke85]
    - Min-length watchman tour (mobile guard) [CN86]
- Other recent experiments
  - Experiments with (exp-time) combinatorial algorithm for guarding the boundary of P [BL06]

Greedy Heuristics

- Two phases:
  - Generate a set of good *candidate* guard positions
  - Greedily select a subset of candidates that fully cover *P*

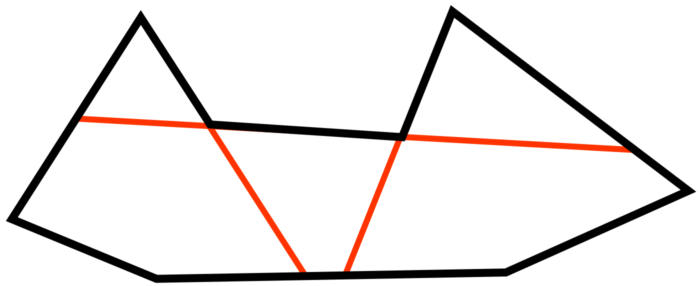
- Algorithm design choices:
  - How to specify the set of candidates?
  - How to score candidates for greedy selection?

Phase 1: Generating Candidates

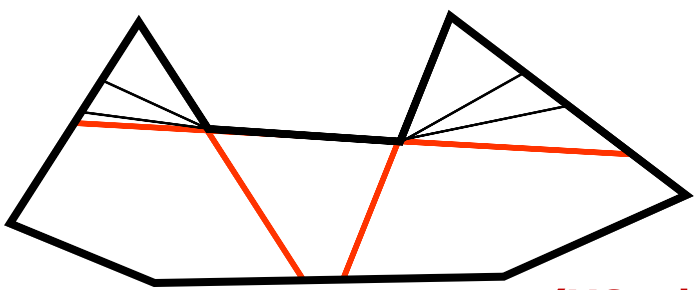
1. Use set V(P) = vertices of polygon P

(actually used points perturbed interior to P)

2. Centers C(P) of convex cells in an arrangement:
   - Edge extensions [size $O(n^2)$]



   - Visibility extensions [size $O(n^4)$]

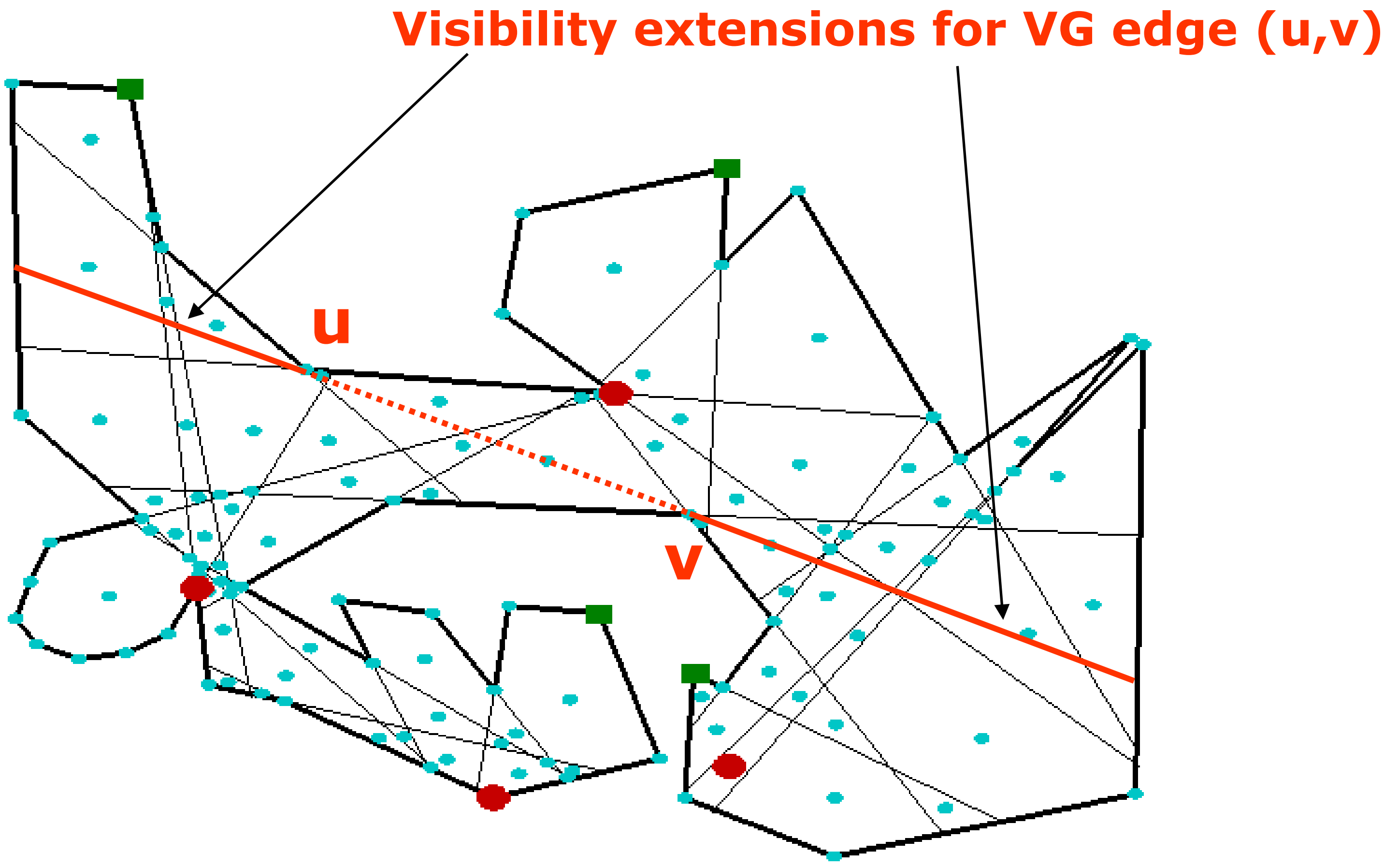


(VG edges incident on at least 1 reflex vertex)

3. $V(P) \cup C(P)$

Example

Centers of cells in arrangement of edge extensions



Phase 2: Greedily Selecting Candidates

- Set of candidates: $W(P)$

- Greedily add "good" candidates $g \in W(P)$ until P is covered: Max $\mu(g)$ $g \in W(P)$
- At end, iteratively remove redundant guards until set is *minimal*

Heuristics Used in Experimentation

- A_1 :

  Candidates W(P) = V(P) ∪ C(P)

  Vertices and center points in arr

  Score $\mu(g)$ = # unseen candidates

  Arrangement: Edge extensions

- A_2 :

  Variant: With each guard g chosen, add to arrangement the visibility edges V(g) induced by g

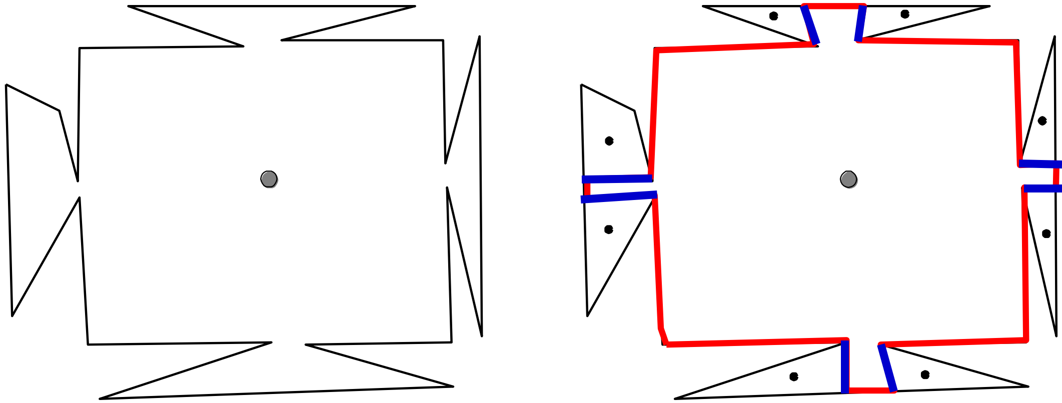


Figure 2: Using algorithm A_2: (a). The polygon and the first guard to be selected (shaded). (b). The visibility polygon of the guard (highlighted, in red) caused the addition of 8 new candidates (small black disks).

Blue: added edges

Heuristics Used in Experimentation

A_3 : Like A_1 but: Score $\mu(g)$ = *area* newly seen

A_4 : Like A_1 but: $\mu(g)$ weighted by *cell area*

A_5 : Like A_4 but: $\mu(g)$ weighted by shared bd(P)

A_6 : Like A_4 but: $\mu(g)$ weighted by % of shared bd(P)

A_7 : Like A_1 but: Candidates $W(P) = V(P)$

A_8 : Like A_1 but: Candidates $W(P) = C(P)$

A_9 : Like A_1 but: $\mu(g)$ = # newly seen vertices

A_{10} : Like A_1 but: $\mu(g)$ = # newly seen cell centers

A_{11} : Like A_1 but: Arrangement of *visibility extensions*

A_{12} : Combination of A_2 and A_{11}

(dynamically added edges, arr of visibility extensions)

Method: A_{13} : Probabilistic Reweighting

We also implemented an algorithm based on the Clarkson/Bronnimann-Goodrich framework: [EH03,GL01]

Each candidate is assigned a weight : probability it is selected

Initially: All weights = 1

Iteration: A candidate is selected at random

If there is an unguarded point, q, then the weights of candidates that see q are *doubled* **(improve chances q is guarded on future iterations)**

Continue until all points of P are guarded

Method: A_{14} : Polygon Partition

We also implemented an algorithm based on partitioning P into star-shaped pieces

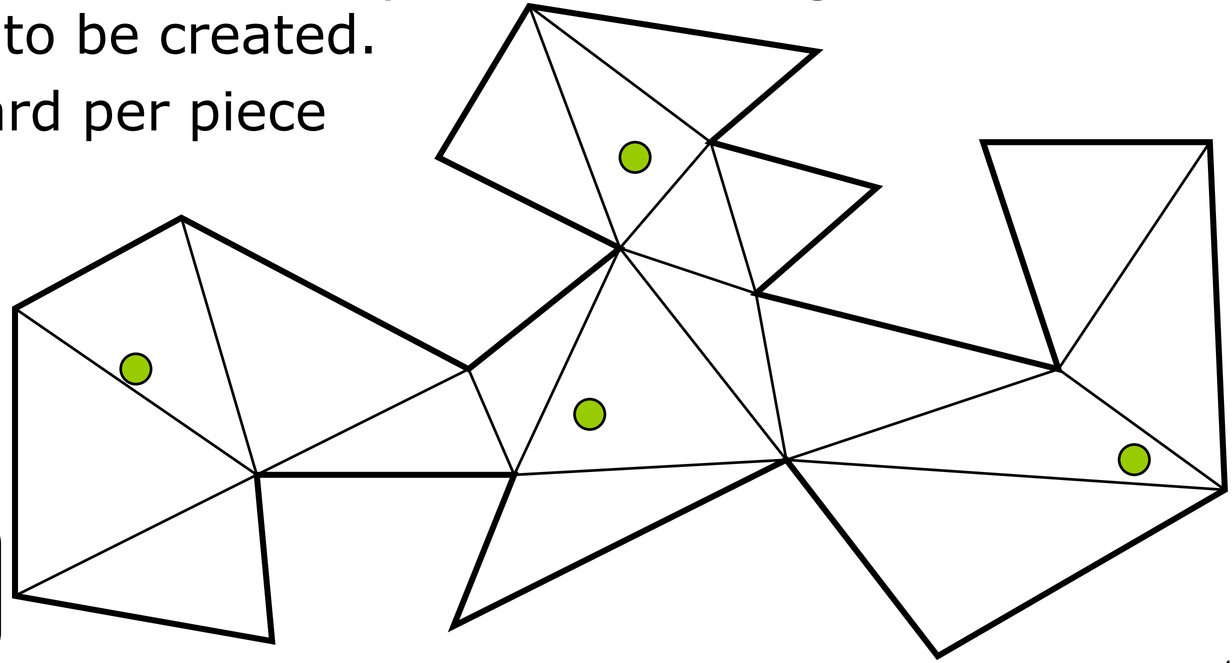
(Note: min-size partition into star-shaped polygons is poly-time, using DP)

We use a simple heuristic similar to Hertel-Mehlhorn 4-approx for min-cardinality convex partition:

- Triangulate P
- Remove diagonals iteratively, never allowing a non-star-shaped piece to be created.
- Place one guard per piece

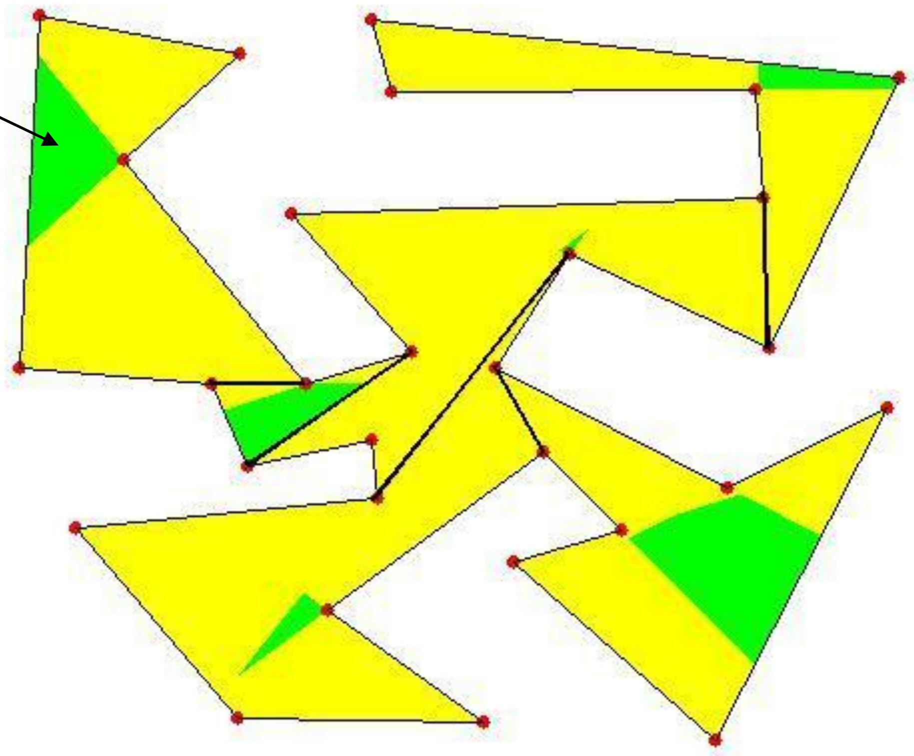
Not competitive with other methods (most cases)

Particularly poor on "spike box" examples



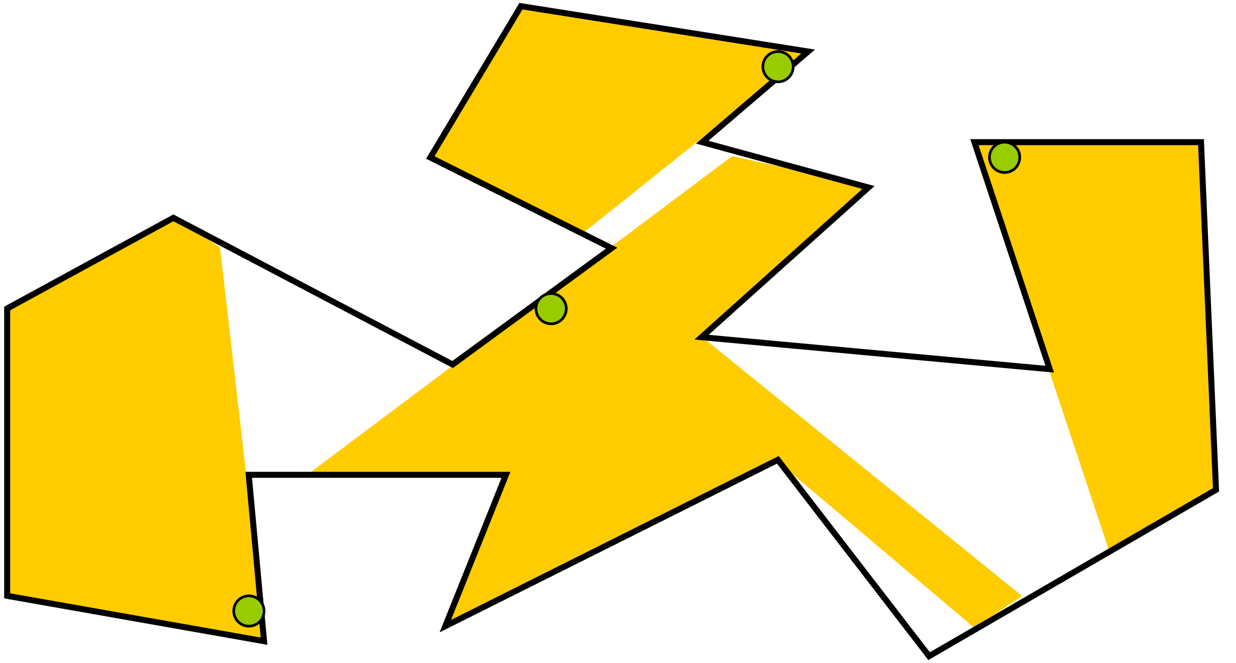
Example: A_{14} : Polygon Partition

kernels in green



Lower Bounds on OPT

Lemma: $g(P) \geq |I|$, for any visibility-independent set I of points in P



$g(P) \geq 4$

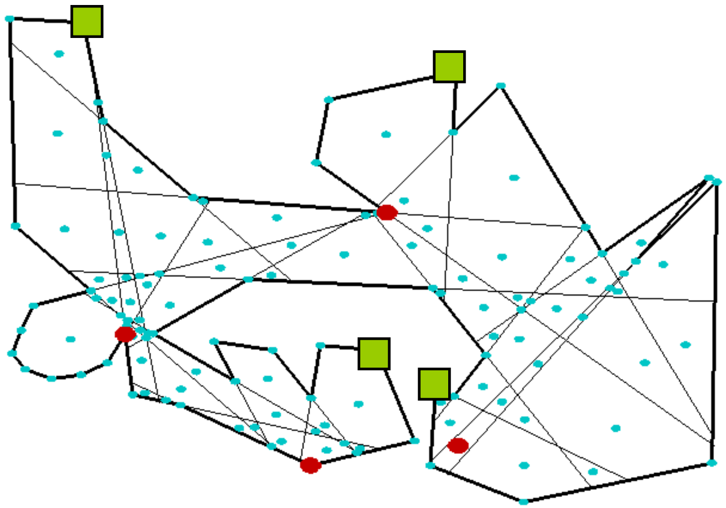
Lower Bounds on OPT

We greedily compute a visibility-independent set I:

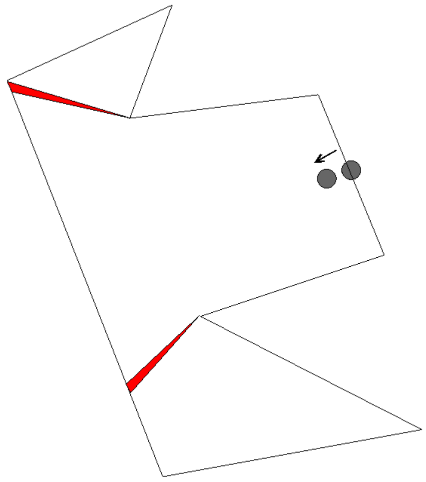
- Generate candidate set S   (not vis-indep)
- Add points $p \in S$ iteratively to I, minimizing # points of S seen by p, making sure that VP(p) is disjoint from VP(q), for $q \in I$

(We use CGAL arrangements to maintain VP's and test vis-independence)

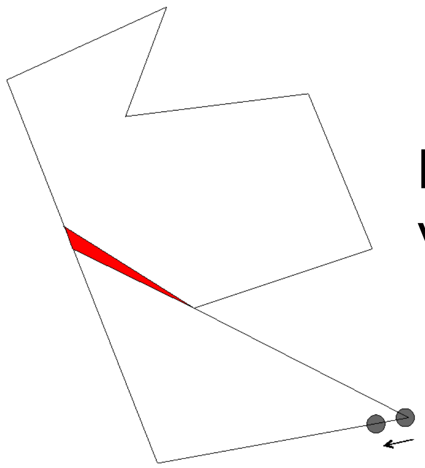
- Remove from S points seen by p
- Stop when S is empty



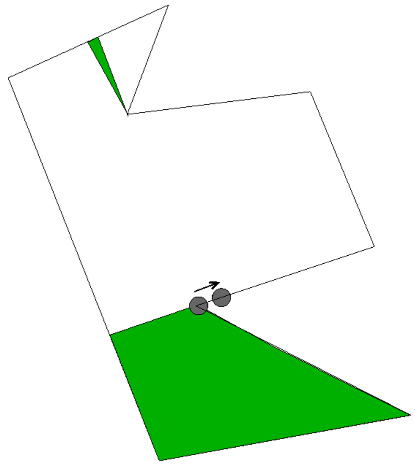
Lower Bounds on OPT



Most cases: $p \in bd(P)$ sees less



Moving away from a convex vertex tends to see more



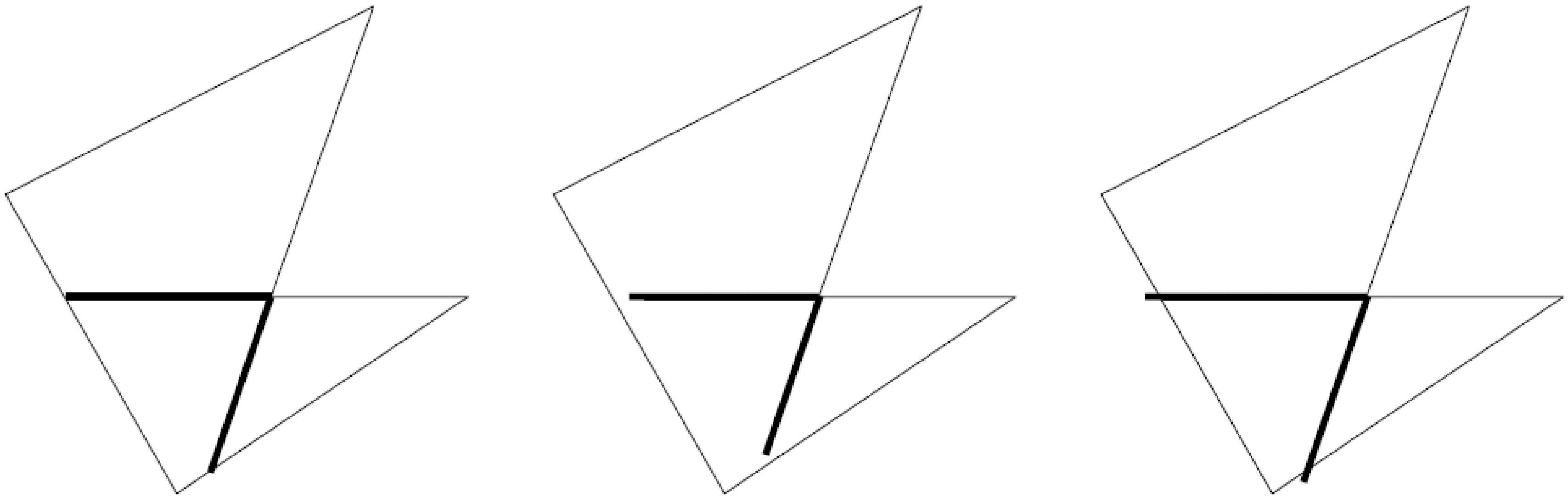
Moving away from a reflex vertex tends to see less

Heuristic: Candidates S are convex vertices and midpoints of edges of P joining two reflex vertices

Experimental Setup

- Windows XP, Pentium 4 (3.2GHz, 2.0GB)
- Visual .Net compiler; openGL; CGAL
- Randomly generated polygons:
  - RPG of Auer and Held, 50-200 vertices
- Manually generated special polygons

Robust computation of cells

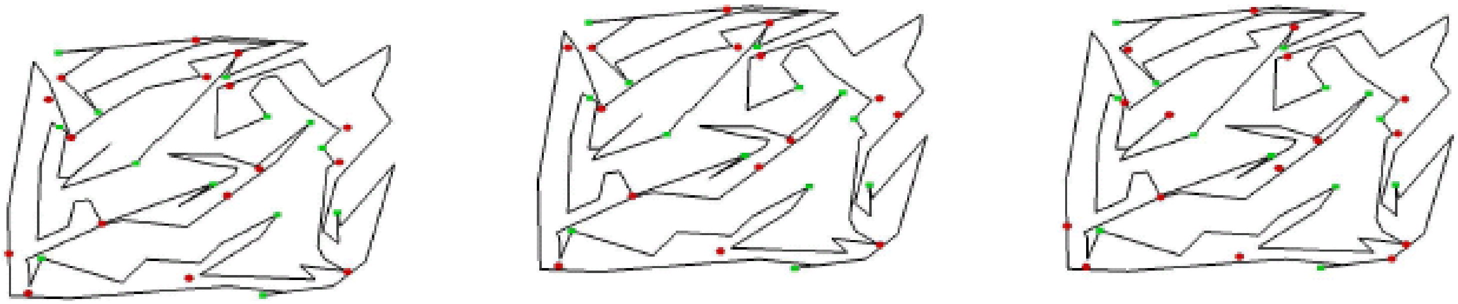


With exact arithmetic

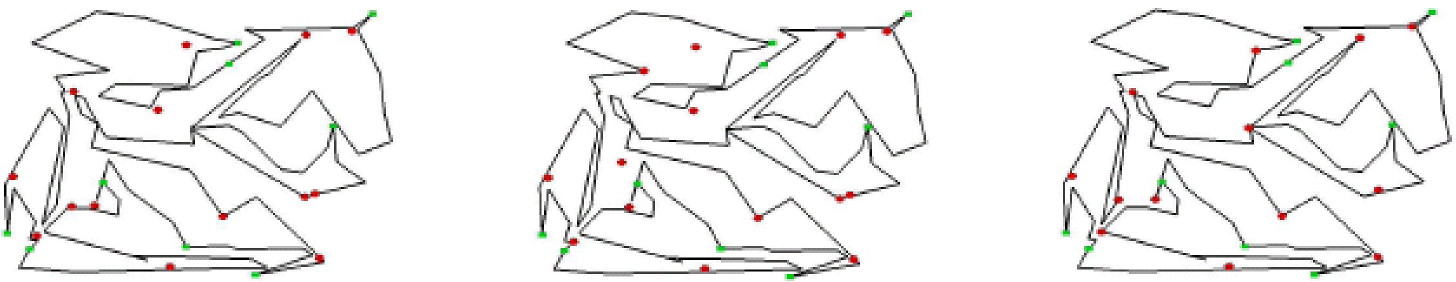
Possible error with floating-point

Solution: push extensions

Examples: n=100



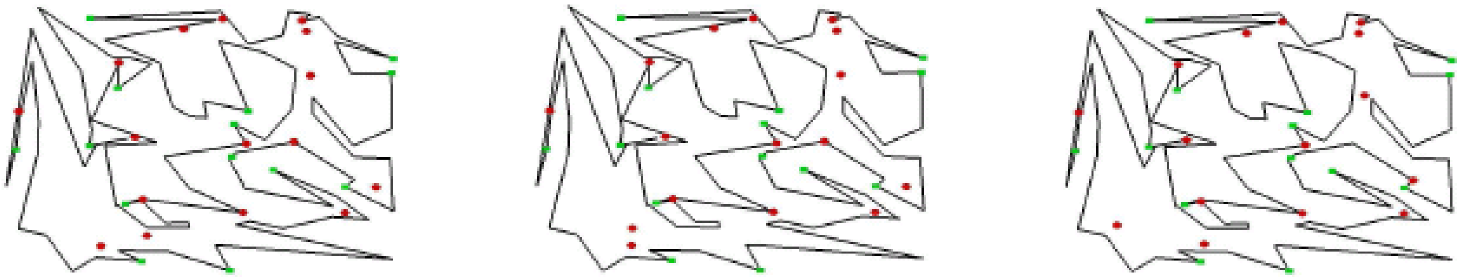
(a) 16 guards (b) 15 guards (c) 16 guards



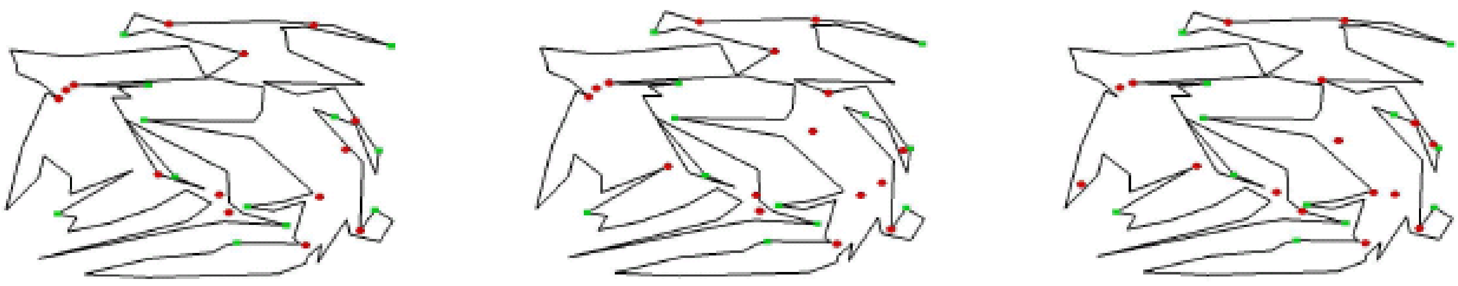
(d) 14 guards (e) 14 guards (f) 13 guards

A_1 A_2 A_{11}

Examples: n=100



(g) 16 guards (h) 16 guards (i) 15 guards

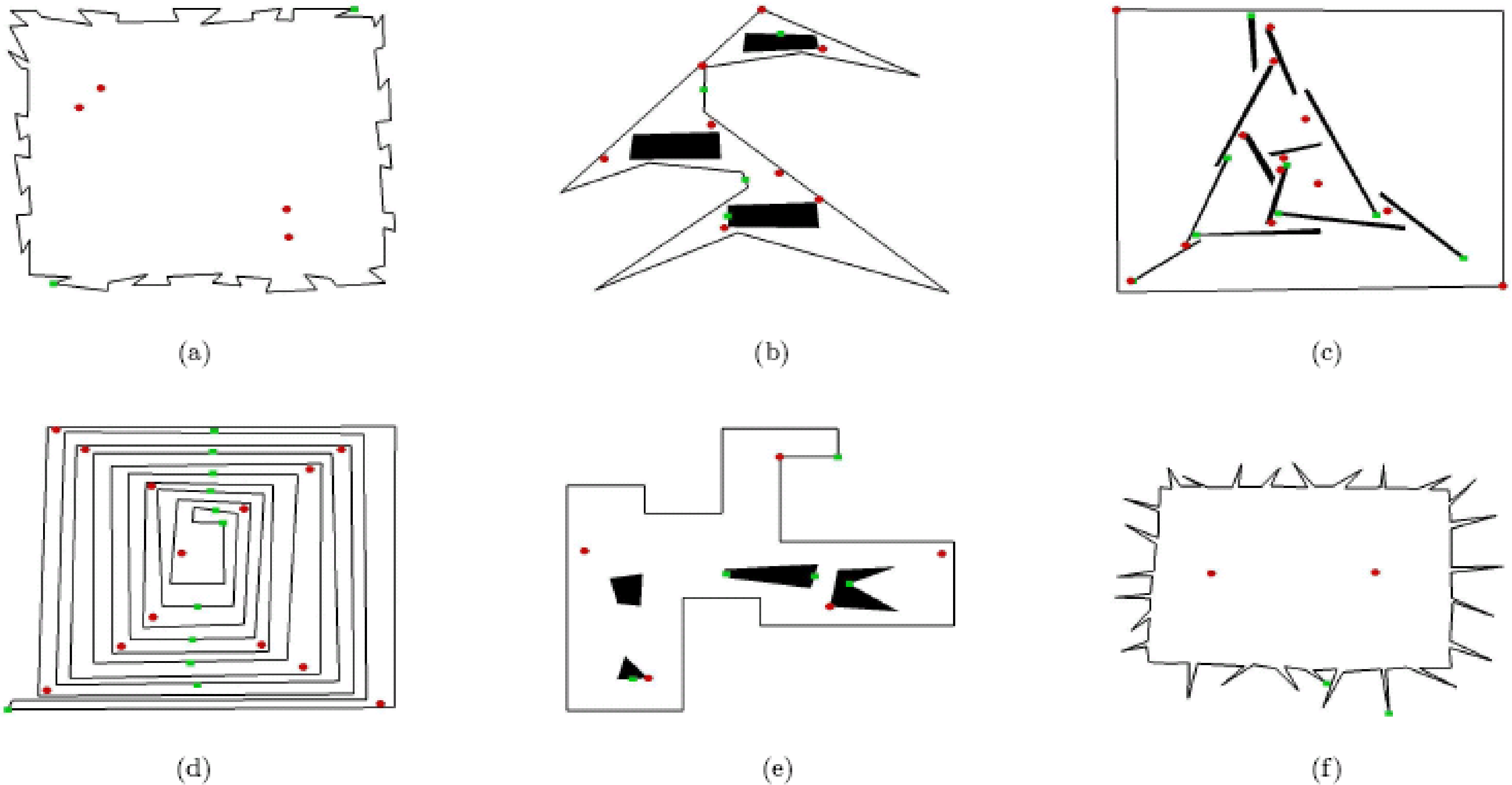


(j) 14 guards (k) 16 guards (l) 16 guards

A_1 A_2 A_{11}

More Examples

A_1



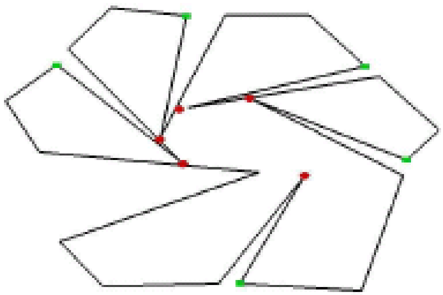
(a) (b) (c)

(d) (e) (f)

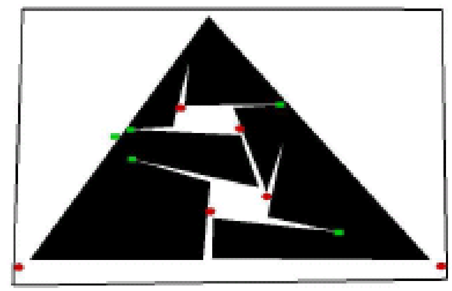
Spike box

More Examples

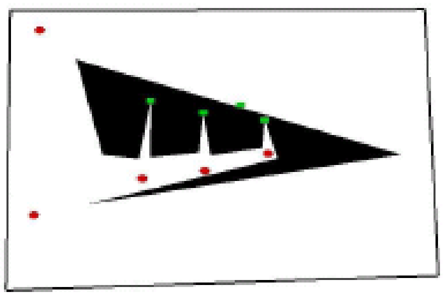
A_1



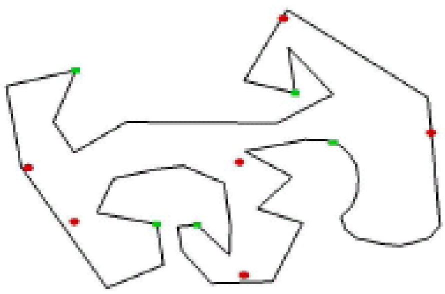
(g)



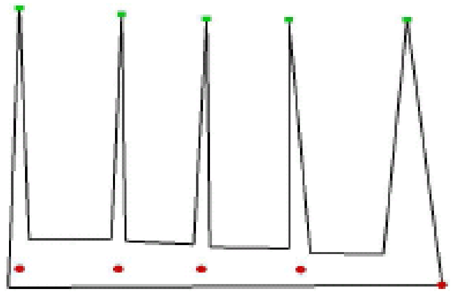
(h)



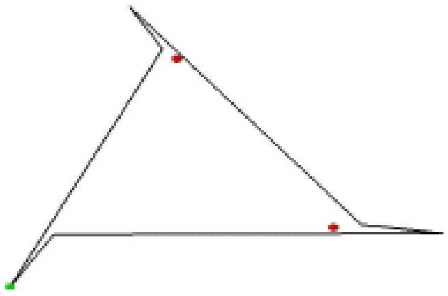
(i)



(j)



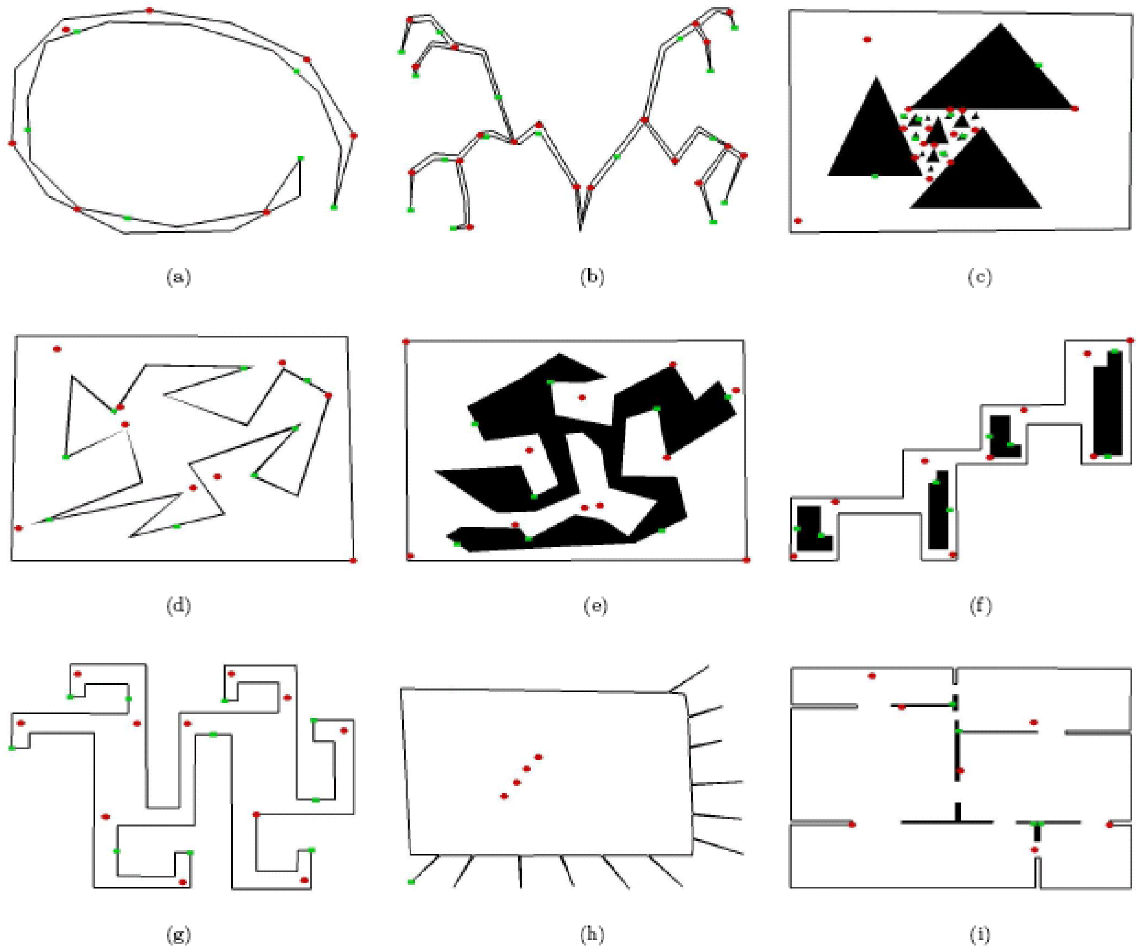
(k)



(l)

More Examples

A_1



Comparison of Heuristics

Results on 40 polygons:

	A_1	A_2	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}	A_{12}	A_{13}	A_{14}
K	0.7	0.47	1.47	1.6	1.3	1.22	0.9	0.83	1.75	0.48	0.5	1.64	3.33
M	0.10	0.06	0.22	0.22	0.16	0.29	0.13	0.14	0.41	0.08	0.09	0.27	0.69
Q	16	17	11	11	10	10	11	12	8	15	15	9	8
B	40	40	40	40	40	40	40	30	29	39	38	39	30

Table 1: Results obtained with our heuristics on 40 input sets.

K - average **excess** = *number* of guards *more* than the *min* guard number over all heuristics

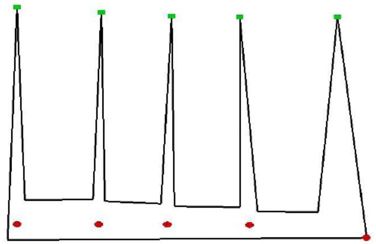
M – average **relative excess** (relative to min)

Q - number of times (out of 40) the guarding obtained with the heuristic was the **best** among all heuristics

B - number of completed tests

Comparison of Heuristics

PN	V	RV	H	VH	RVH	A_1	A_2	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}	A_{12}	A_{13}	A_{14}	LB
1	24	10	0	0	0	5	4	6	5	4	5	5	4	5	4	4	3	5	2
2	9	3	0	0	0	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	12	7	3	12	12	8	7	8	8	7	8	8	7	6	7	7	8	X	4
4	44	20	0	0	0	13	13	12	12	13	14	13	13	11	12	12	11	20	11
5	83	52	0	0	0	2	2	2	2	2	10	2	2	X	X	X	3	22	2
6	33	19	0	0	0	2	2	3	2	3	4	2	2	4	2	2	4	4	2
7	24	6	0	0	0	5	5	5	5	5	5	5	5	6	5	5	5	6	5
8	4	0	12	60	48	13	13	14	14	16	12	16	18	15	9	9	14	X	8
9	17	7	0	0	0	3	3	5	4	4	3	4	3	4	3	3	3	4	3
10	5	2	1	3	3	3	3	3	3	3	3	3	3	3	3	3	3	X	2
11	6	2	1	5	5	4	4	4	4	4	4	3	3	4	4	4	5	X	2
12	4	0	13	48	44	15	13	17	17	16	15	18	16	14	13	X	15	X	7
13	16	6	4	16	5	5	5	7	9	8	6	6	7	6	6	6	7	X	5
14	6	3	0	0	0	2	2	2	2	2	2	2	2	2	2	2	3	X	1
15	4	0	1	20	14	6	6	6	6	6	6	6	6	6	6	6	7	X	5
16	4	0	1	13	9	5	5	5	5	5	5	5	5	5	5	5	5	X	4
17	15	8	0	0	0	5	5	5	5	5	5	5	5	5	5	5	5	5	5
18	49	24	0	0	0	6	7	6	7	7	7	6	6	28	5	5	X	12	4
19	100	46	0	0	0	13	12	16	16	15	14	13	X	X	13	13	15	X	11
20	100	47	0	0	0	18	15	19	19	18	15	15	X	X	16	16	18	20	12
21	100	44	0	0	0	16	15	18	17	16	16	16	X	X	16	16	19	19	13
22	100	50	0	0	0	14	14	15	15	15	15	14	X	X	14	14	16	16	9
23	100	45	0	0	0	18	18	18	19	18	17	16	X	X	17	17	17	19	13
24	100	55	0	0	0	16	16	18	18	18	16	16	X	X	15	15	16	21	14
25	100	49	0	0	0	18	17	18	18	18	18	18	X	X	17	17	22	22	14
26	100	46	0	0	0	17	16	19	18	18	17	18	X	X	16	16	17	19	13
27	100	49	0	0	0	12	12	15	16	13	15	12	X	X	14	14	18	16	11
28	100	52	0	0	0	14	16	15	15	17	17	15	X	X	16	16	16	19	12
29	10	3	0	0	0	2	1	2	2	1	2	2	2	2	2	2	2	2	1
30	10	3	0	0	0	1	1	1	1	1	2	1	1	2	1	1	2	1	1
31	10	4	0	0	0	2	2	2	2	2	2	2	2	3	2	2	3	2	2



Comparison of Heuristics

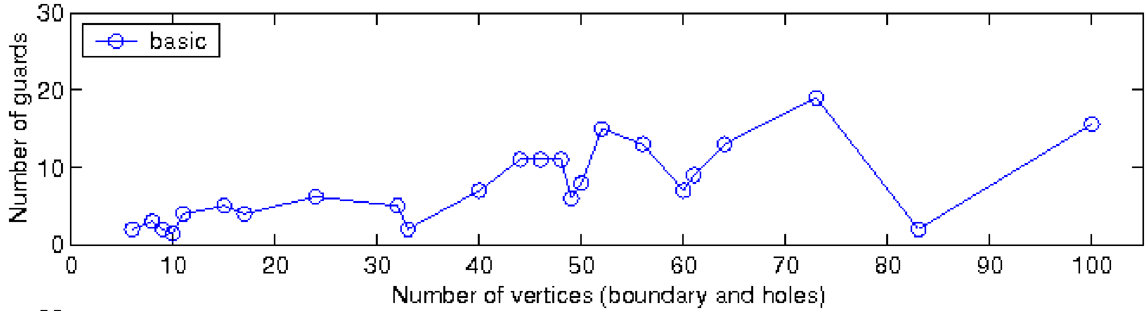
PN	A_1	A_2	A_{11}	I_1	I_2	I_3	min ratio
1	5	4	4	2	1	1	2
2	2	2	2	2	2	X	1
3	8	7	7	4	2	4	1.75
4	13	13	12	11	8	10	1.09
5	2	2	X	2	2	1	1
6	2	2	2	2	2	1	1
7	5	5	5	5	5	X	1
8	13	13	9	8	1	8	1.125
9	3	3	3	3	3	2	1
10	3	3	3	2	1	1	1.5
11	4	4	4	2	2	2	2
12	15	14	13	7	2	6	1.85
13	5	5	6	5	3	3	1
14	2	2	2	1	1	X	2
15	6	6	6	5	5	3	1.2
16	5	5	5	4	4	2	1.25
17	5	5	5	5	5	1	1
18	6	6	5	4	4	X	1.25
19	13	12	13	11	11	6	1.09
20	18	15	16	11	12	8	1.25
21	16	15	16	13	12	7	1.15
22	14	15	14	9	9	5	1.55
23	18	18	17	13	13	5	1.3
24	16	16	15	14	14	7	1.07
25	18	17	17	14	14	8	1.21
26	17	16	16	13	11	7	1.23
27	12	12	14	11	11	5	1.09
28	14	16	16	12	11	5	1.16
29	1	1	1	1	1	X	1
30	1	1	1	1	1	X	1
31	2	2	2	2	2	1	1
32	2	2	2	2	2	1	1
33	1	1	1	1	1	1	1
34	2	2	2	2	2	1	1
35	1	1	1	1	1	X	1
36	1	1	1	0	1	1	1
37	1	1	1	1	1	1	1
38	2	2	2	2	2	X	1
39	10	9	9	8	8	3	1.125
40	8	8	8	7	7	3	1.14
41	9	9	9	8	8	3	1.125
42	6	6	8	6	6	5	1
43	6	6	7	6	6	1	1
44	10	10	10	8	8	3	1.25
45	8	8	8	8	7	4	1
46	8	9	8	8	8	3	1
47	6	6	6	6	6	3	1

Comparison of Heuristics

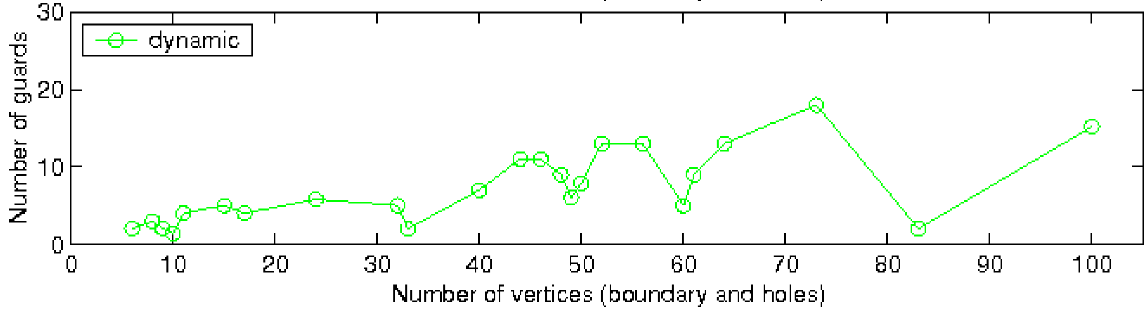
PN	V	RV	H	VH	RVH	A_1	A_2	A_{11}
41	24	11	0	0	0	7	7	6
42	73	40	0	0	0	19	18	18
43	4	0	17	48	48	15	12	14
44	4	0	1	57	26	9	9	9
45	4	0	1	44	26	11	9	8
46	20	8	4	24	20	9	9	10
47	24	10	8	32	32	13	13	13
48	46	21	0	0	0	11	11	11
49	40	19	0	0	0	10	10	10
50	28	12	4	32	24	7	5	6
51	40	120	0	0	0	4	4	4
52	200	101	0	0	0	35	X	33
53	200	95	0	0	0	31	X	31
54	200	100	0	0	0	31	X	27
55	200	100	0	0	0	28	X	31
56	200	104	0	0	0	31	X	31
57	200	97	0	0	0	26	X	26
58	200	98	0	0	0	26	X	27
59	200	98	0	0	0	30	X	31
60	200	98	0	0	0	26	X	26
61	200	99	0	0	0	31	X	30

Number of Guards vs. Number of Vertices

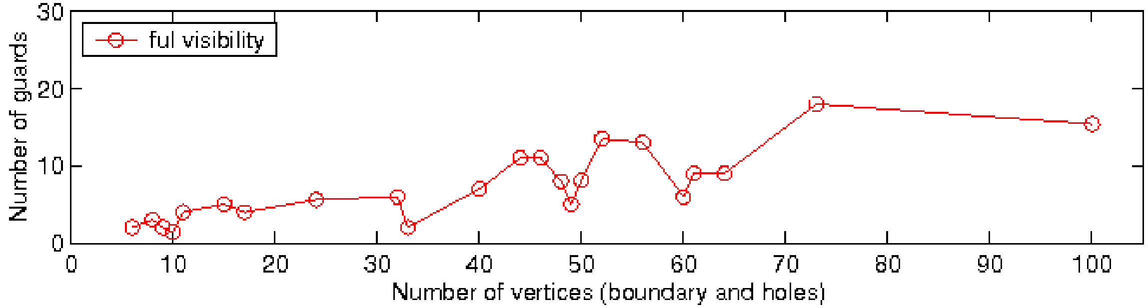
A_1



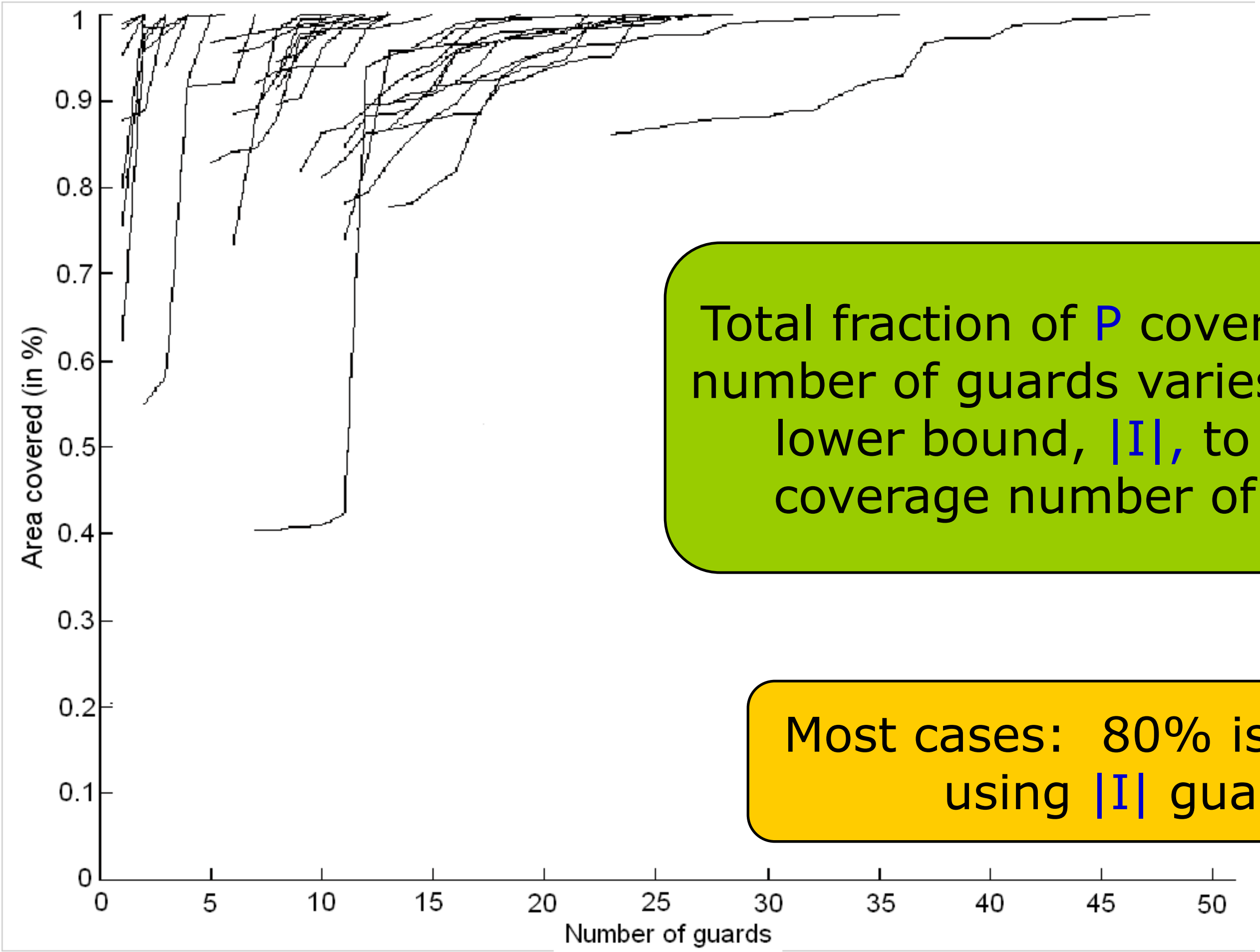
A_2



A_{11}



Early Termination: Partial Covering



Total fraction of P covered as the number of guards varies from the lower bound, |I|, to the full coverage number of guards

Most cases: 80% is covered using |I| guards

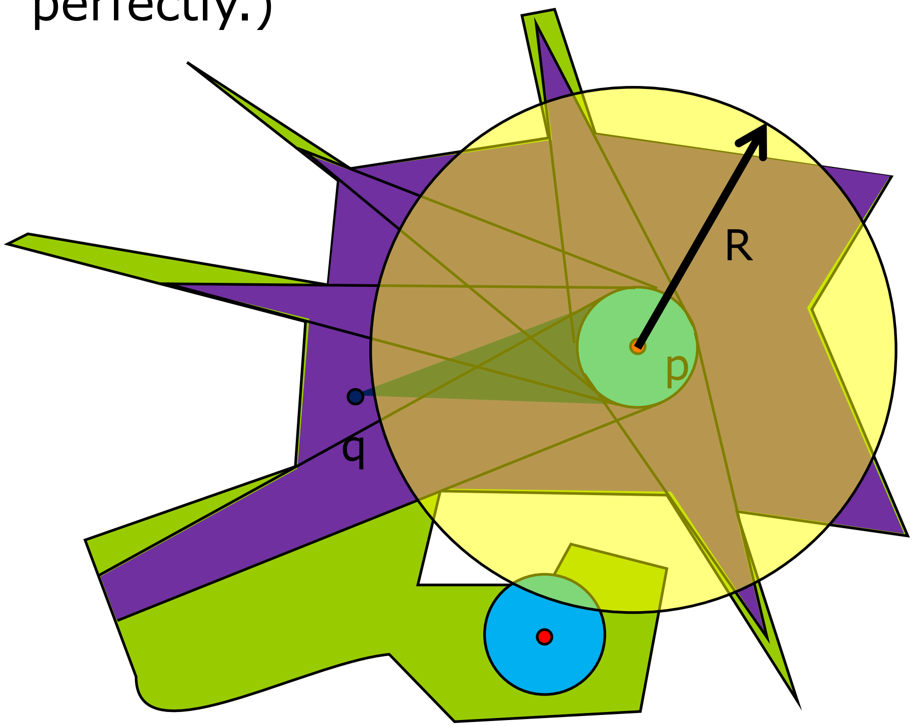
Conclusion

- Extensions:
  - Visibility constraints (view distance, good view angles, robust coverage)
  - Terrain coverage (2.5D)
  - 3D
- Open:
  - Any approx algorithm (better than n/3-approx) for unrestricted guards
  - O(1)-approx for vertex/grid guarding simple polygons
  - Characterization of polygons for which our heuristics perform well (provably well)?

Robust Guards

Issue: Even if we computed exactly a minimum cardinality set of guards, could we know with confidence the domain is really guarded?

Guards may not be placed exactly. (Human guards don't usually stand exactly still, and cameras/sensors cannot be placed perfectly.)



Model: When a guard is placed at p, it will actually reside at some point within a disk, $B_\varepsilon(p)$, of radius ε

In order for q to be "seen" by guard p, it must be able to see the guard no matter where it is within the disk $B_\varepsilon(p)$

Bounded radius, R, of vision

Robust Guards: New Approx Bound

Theorem: There is a PTAS for computing a min # of robust, radius-bounded guards in a polygonal domain (with holes), assuming R/ε is bounded, and a poly-size set G of candidate guard locations is given.

One option for G: use a set L of $O(\lambda \log^2 \lambda)$ *landmarks*, as in [AEG08], and then guarantee at least $(1-\varepsilon_1)$-fraction of the area is seen.

$\lambda = (g_{opt} / \varepsilon_1) \log h$ (h = # holes)

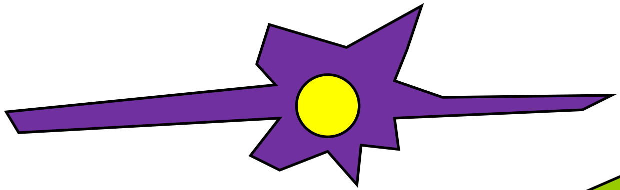
[AEG08] also give randomized greedy algorithm that, whp, computes $O(g_L \log \lambda)$ guards to cover L, where $g_L \leq g_{opt}$ is opt # of guards to cover L

Method: m-guillotine optimization: Convert any OPT to an m-guillotine version; apply DP to optimize

What is Needed for PTAS to Apply

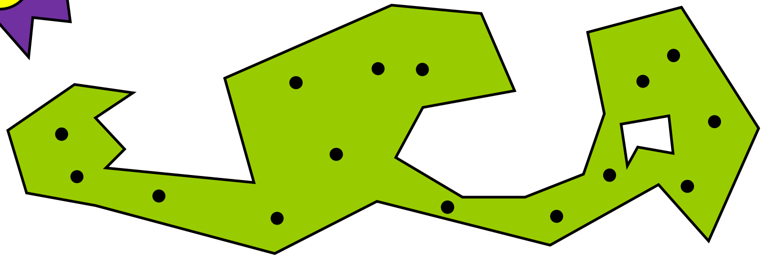
Suffices: Visible regions, VP(g), from candidate guard locations $g \in G$ have area(VP(g)) ≥ c diam^2(VP(g)), for some c. (e.g., each VP(g) contains a disk of radius Ω(diam(VP(g)))

Special Case: Bounded radius visibility in polyominoes



Another Sufficient Model:

Sample points S in P.
Guards placed at subset of S.
Guards must see all of S: Problem is **Dominating Set** in VG(S)



If samples S are δ-*well dispersed* (e.g., no disk of radius δ has more than O(1) samples of S), and guards have visibility radius R, with R/δ bounded, then PTAS also applies

Minimum Dominating Set:
best approx in general is log-approx
PTAS for planar graphs, UDG
APX-complete for degree-B, B≥3

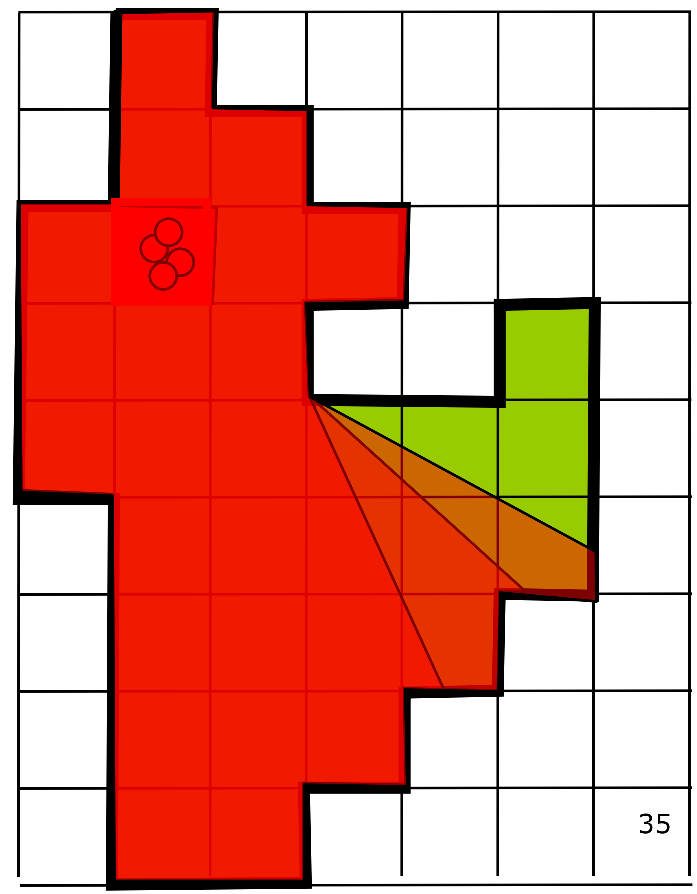
Here, the graph VG(S) is not planar, not UDG, but has bounded degree, depending on R/δ

Guarding Polyominoes

[Irfan, Iwerks, Kim, M]

- Polyomino: simply connected union of m integral unit squares (pixels) – “pixel polygon”

- Models of pixel guards:
  (1) Point guards
  (2) Pixel guards
  (3) Robust (pixel) guards:
  Strong visibility: only those points that are seen from *any* point within the pixel are seen

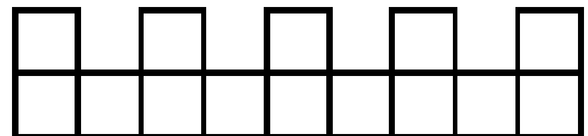


Guarding Polyominoes

Art Gallery Thm:

(1) ceil((m-1)/3) point guards suffice and are sometimes necessary

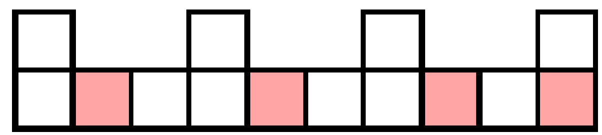
Point Guards



(2) ceil((m-1)/3) pixel guards suffice and ceil((m-1)/4) are sometimes necessary

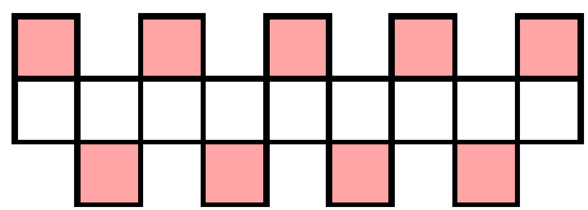
OPEN: Close the gap!

Pixel Guards



(3) floor(m/2) robust guards suffice and are sometimes necessary: Simple coloring argument: 2-color the grid of pixels.

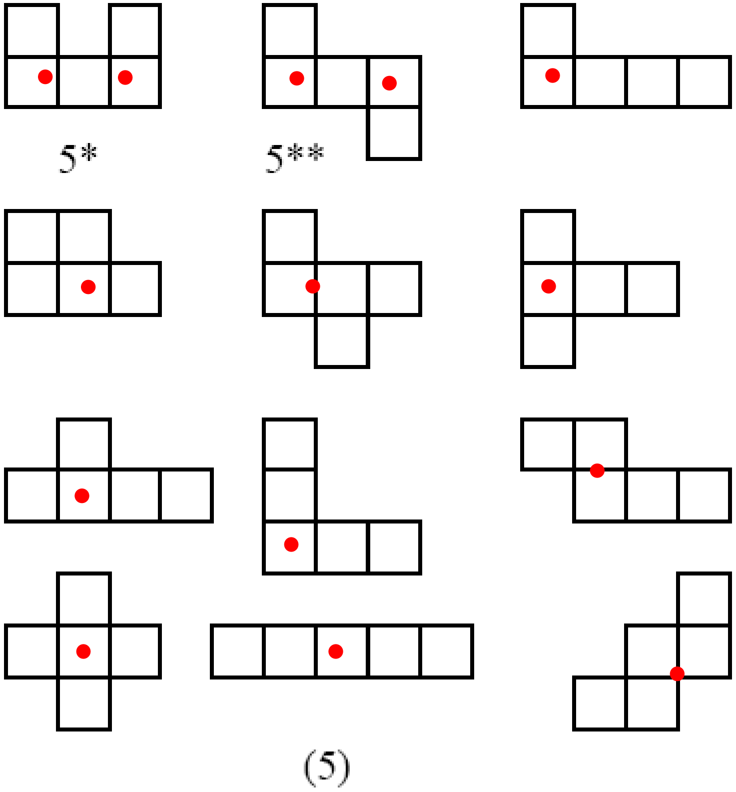
Robust Pixel Guards

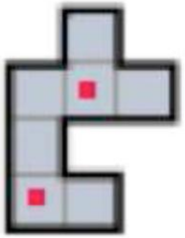


NP-hardness: Computing the guard number in polyominoes is NP-hard

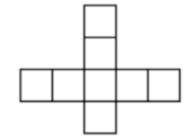
Examples of pentominoes

Each requires just one point guard, *except* 5* and 5**

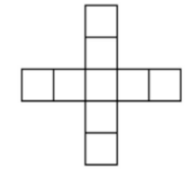




Point Guards in Polyominoes



8*



9*

Claim 1 *Let P be an m-pixel polygon where $m \geq 2$. Then there exists a pixel p that can be removed from P yielding P' such that P' is simply connected.*

Claim 2 *Let P be any (8) besides 8^*. Then we can decompose P into two connected pixel subpolygons P_1 and P_2 such that either $|P_1| = |P_2| = 4$ or $|P_1| = 3$ and $|P_2| = 5$.*

Corollary 1 *If P is any (9) besides 9^*, then we may decompose P into two subpolygons P_1 and P_2 such that either $|P_1| = 3$ and $|P_2| = 6$ or $|P_1| = 4$ and $|P_2| = 5$. Also, any (10), P, is decomposable into two pixel subpolygons P_1 and P_2 such that either $|P_1| = 3$ and $|P_2| = 7$, $|P_1| = 4$ and $|P_2| = 6$, or $|P_1| = |P_2| = 5$.*

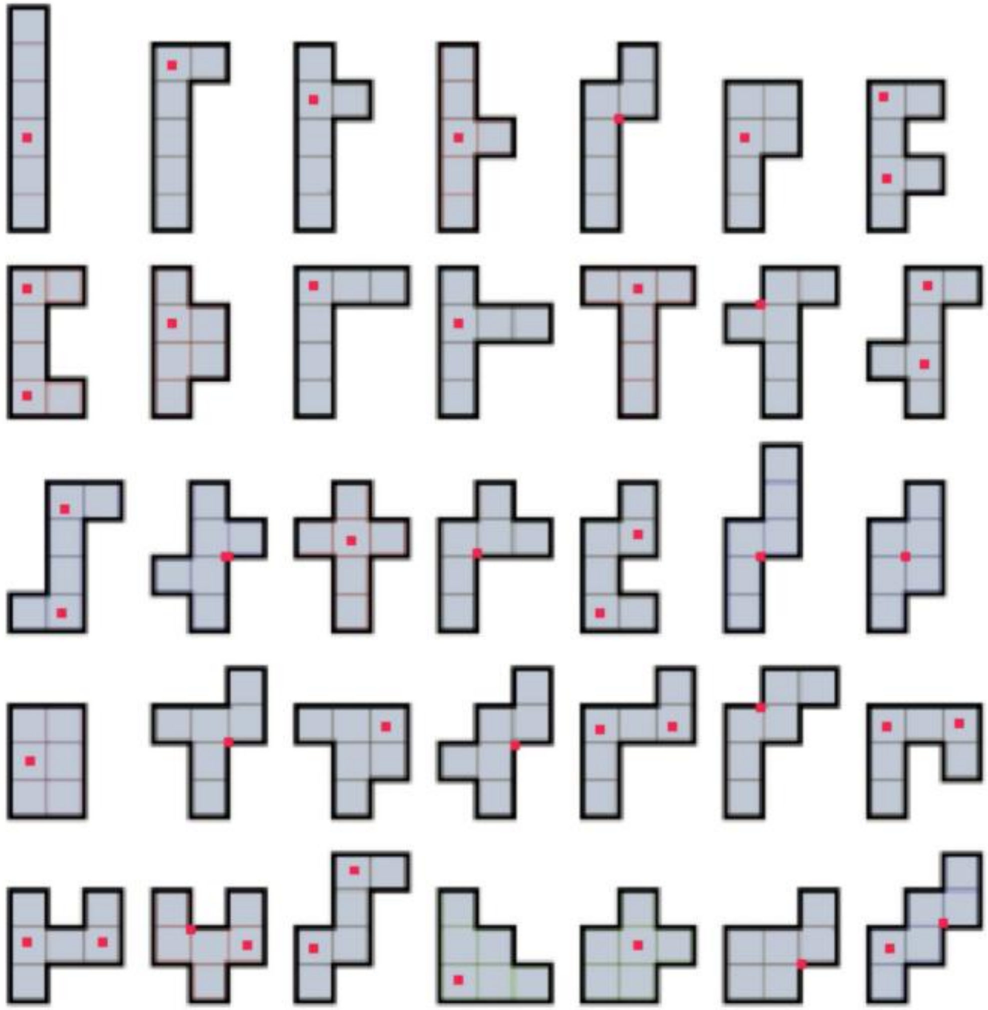
Claim 3 *For any m-pixel polygon P where $m = 1, 2, 3$, or 4, one point guard is sufficient to guard P. For any m-pixel polygon P where $m = 5, 6, 7$, two point guards are sufficient to guard P.*

Claim 4 *For any m-pixel polygon P with $m \geq 3$, pixel subpolygons $S_i \in \{(3), (4), ((5)/5^*, 5^{**}), (6), (7), 8^*, 9^*\}$ $(i = 1, 2, 3, \ldots, f)$ can be removed from P yielding connected pixel polygons $P_1, P_2, \ldots, P_f$ where P_i is the connected pixel polygon remaining after removing the i^{th} pixel subpolygon from P. Also, P_f may contain 0, 1, or 2 pixels.*

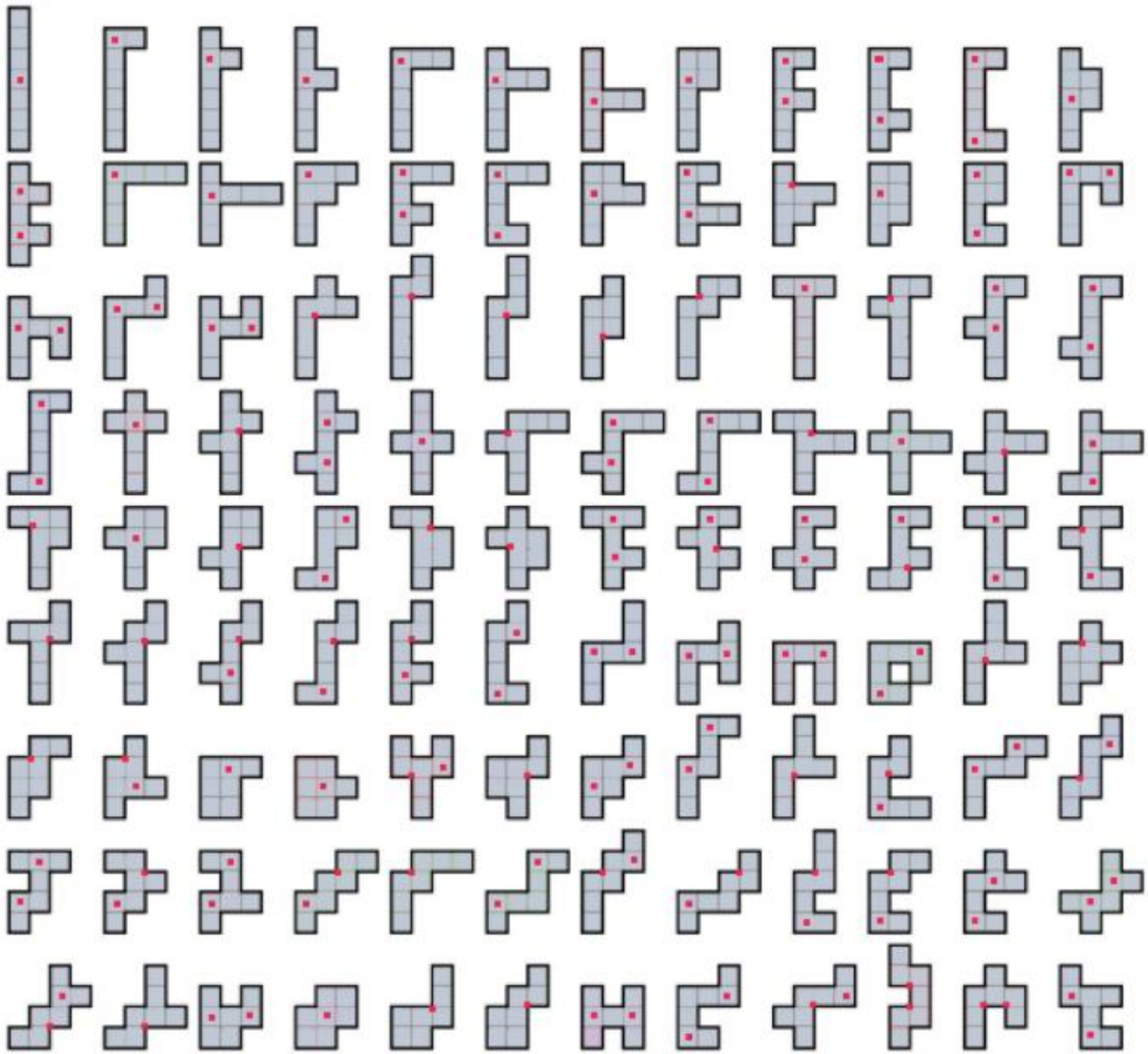
Corollary 2 $\lceil \frac{m}{3} \rceil$ *point guards is sufficient to guard an m-pixel connected polygon P.*

Actually, ceil((m-1)/3)

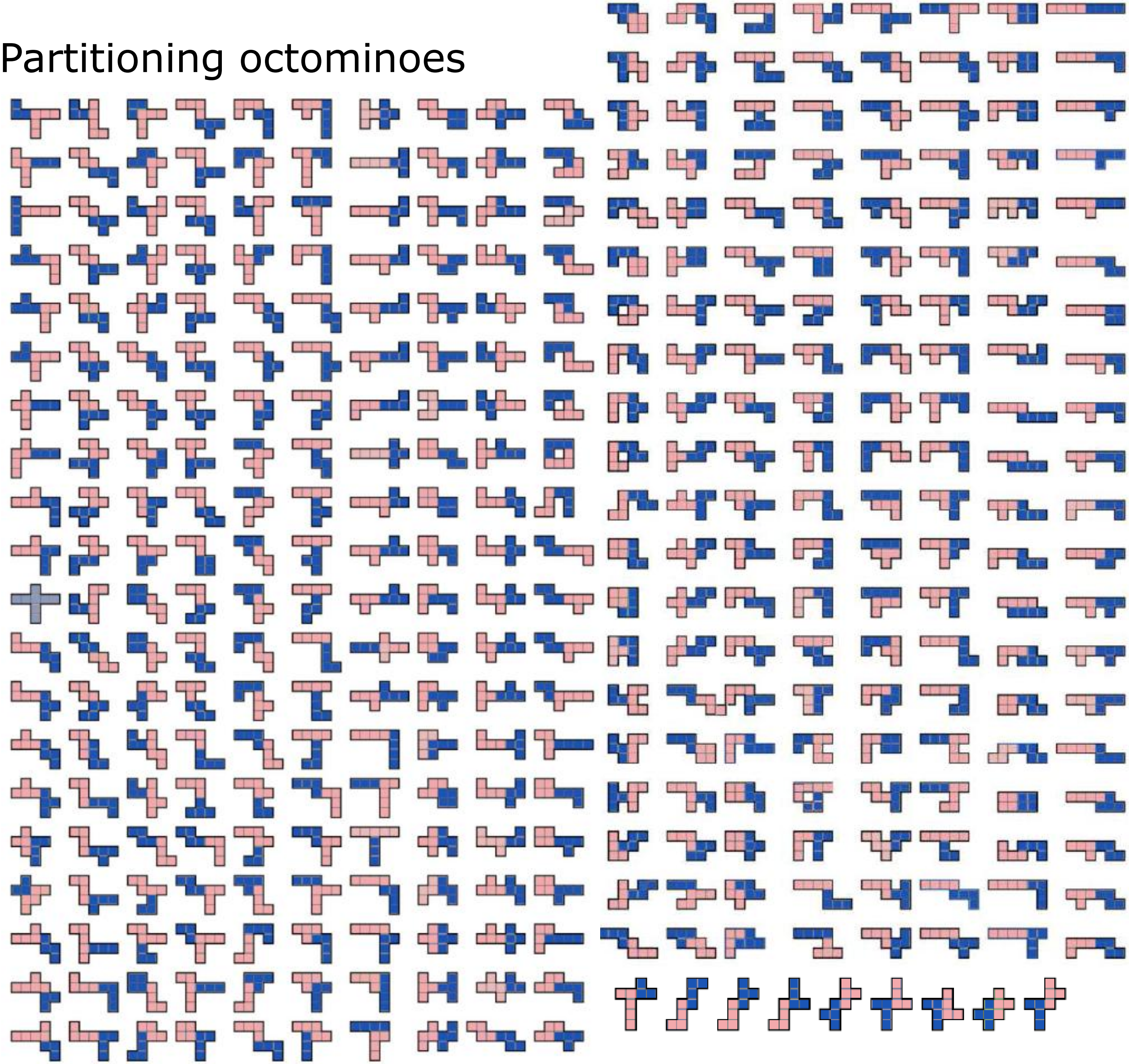
Claim: Any hexomino (m=6) can be guarded with 1 or 2 points.



Claim: Any heptomino (m=7) can be guarded with 1 or 2 points.

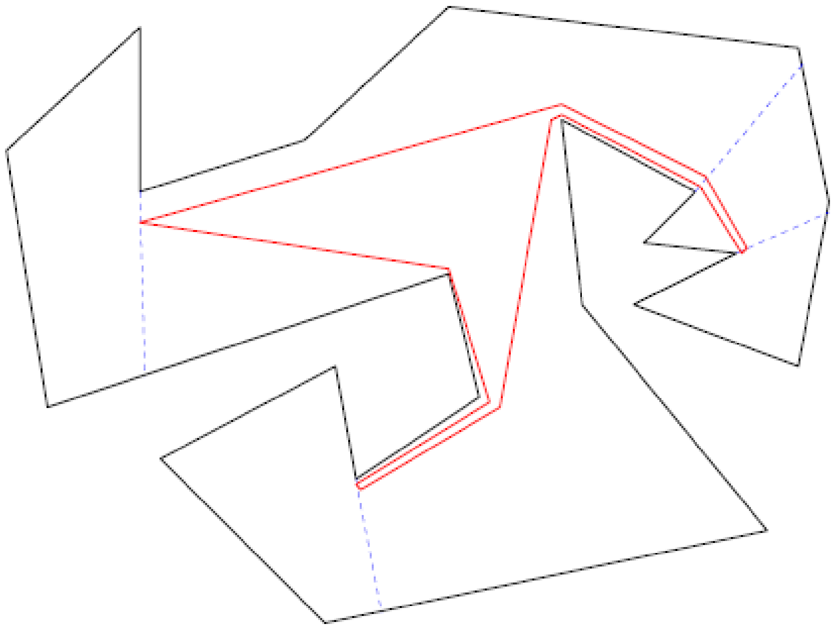


Partitioning octominoes



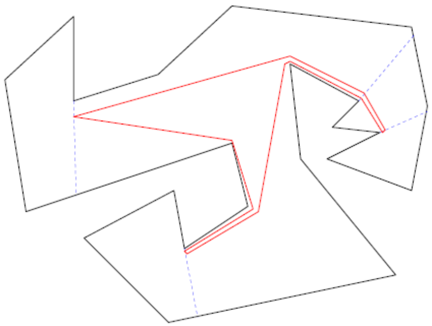
Mobile Guards

Watchman Route Problem



Find a shortest tour for a guard to be able to see all of the domain

Watchman Route Problems



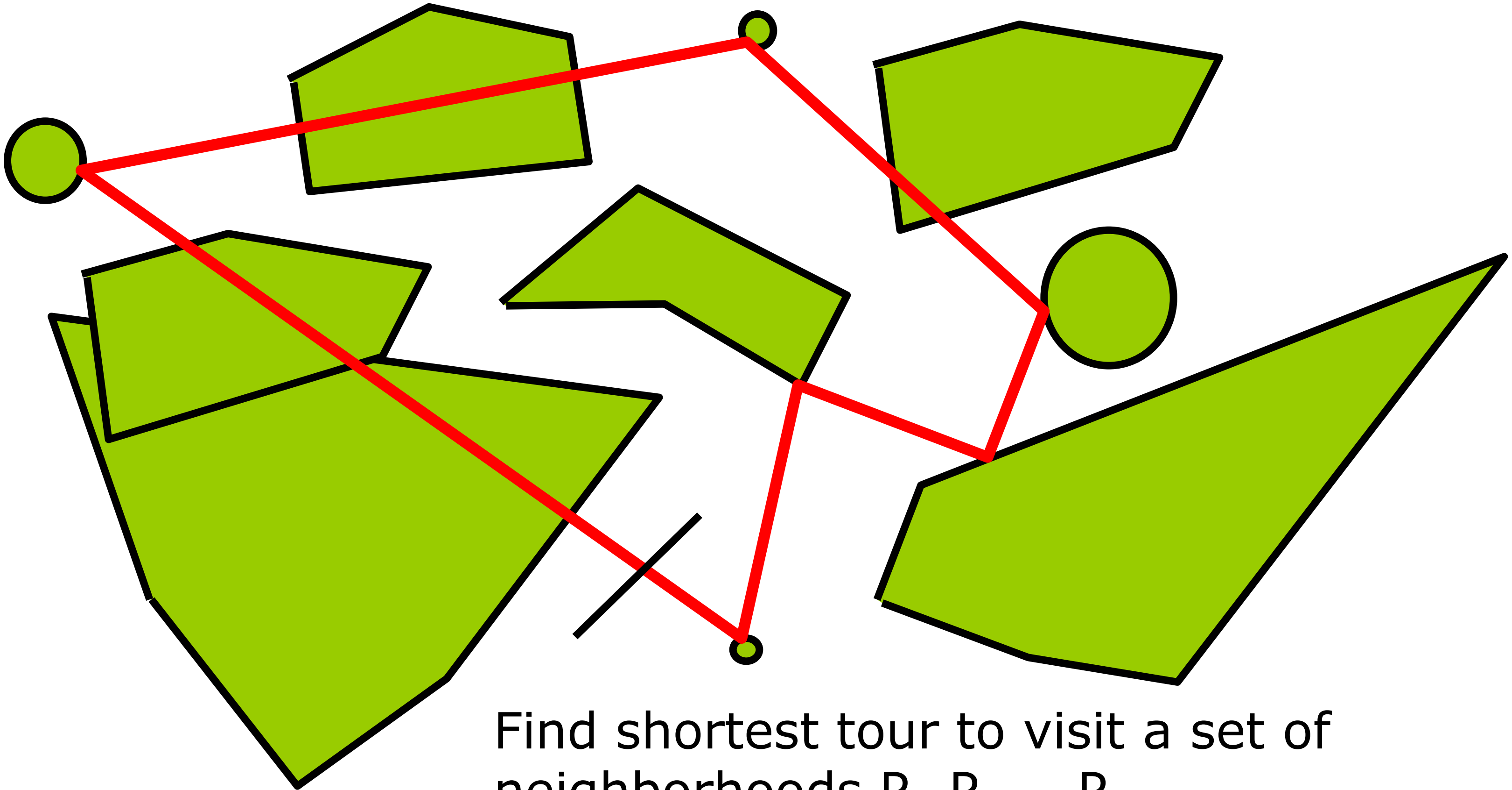
- Closely related to TSPN: visit VP(p), for all p in P
- Poly-time in simple polygons [CN,DELM]
  - Best time bound: $O(n^3 \log n)$ [DELM]

- NP-hard in polygons with holes
  - No approx algorithm known in general!
  - Rectilinear visibility: O(log n)-approx [MM'95]
  - NEW: For fat obstacles, PTAS to see at least one point on the boundary of each obstacle
- 3D: Depends on 3D TSPN [ADDFM]

Q: Approx for planar domain, standard visibility?

Q: Approx for guard on a terrain surface?

TSPN: TSP with Neighborhoods

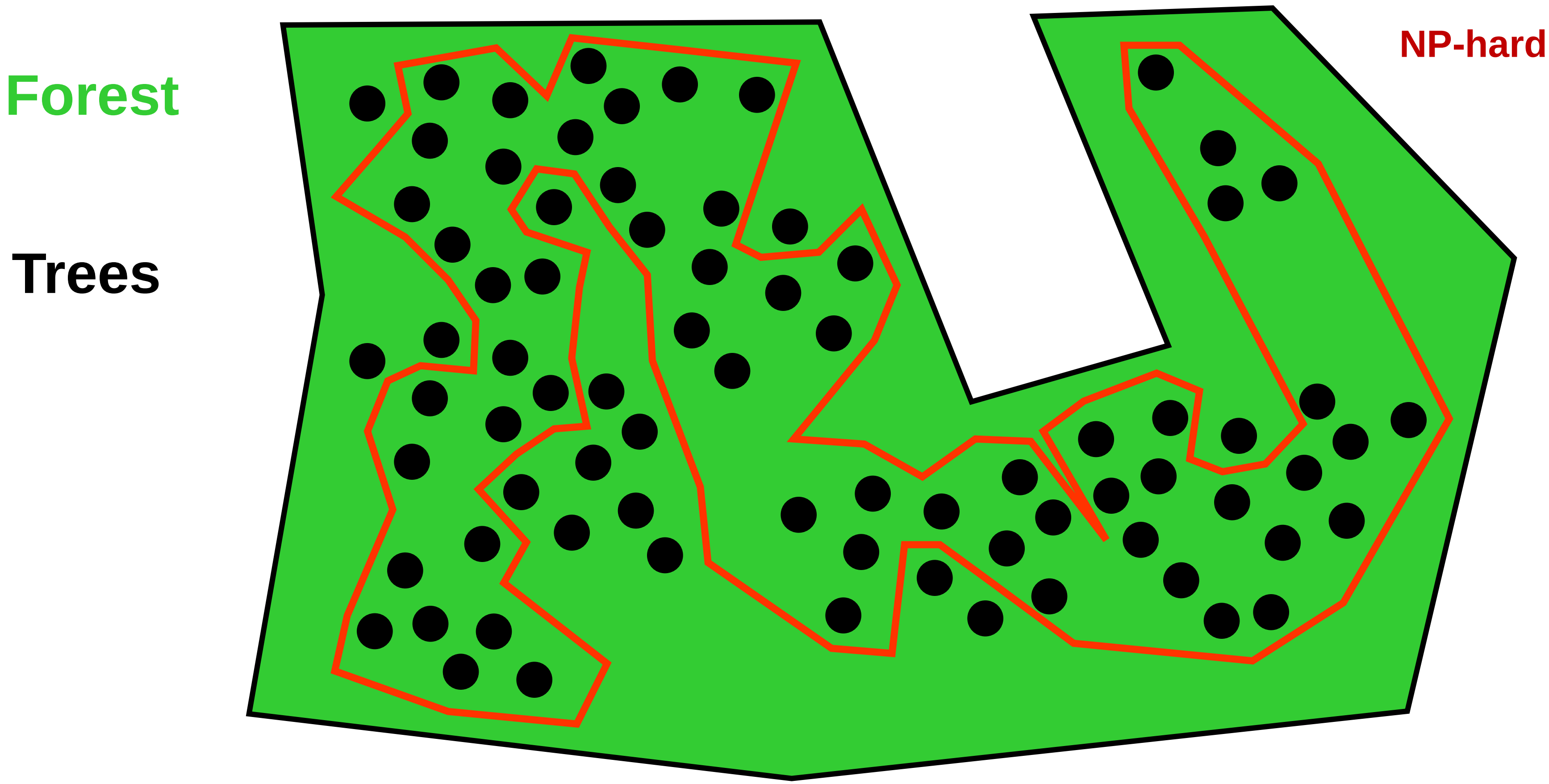


Find shortest tour to visit a set of neighborhoods $P_1, P_2, \ldots, P_n$

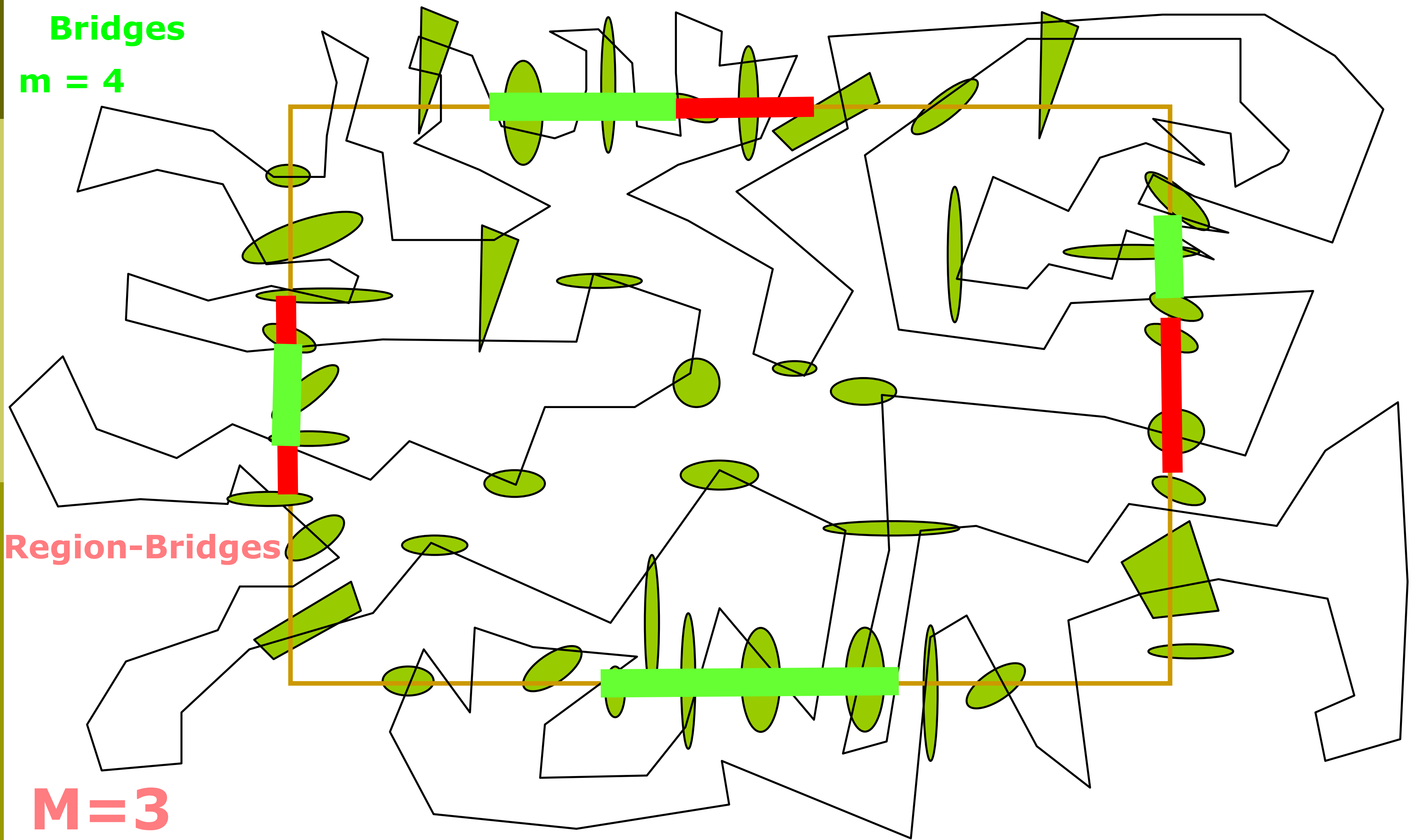
Watchman: How to
"See the Forest for the Trees"

Recent result: Can apply also to yield PTAS for watchman route among fat obstacles

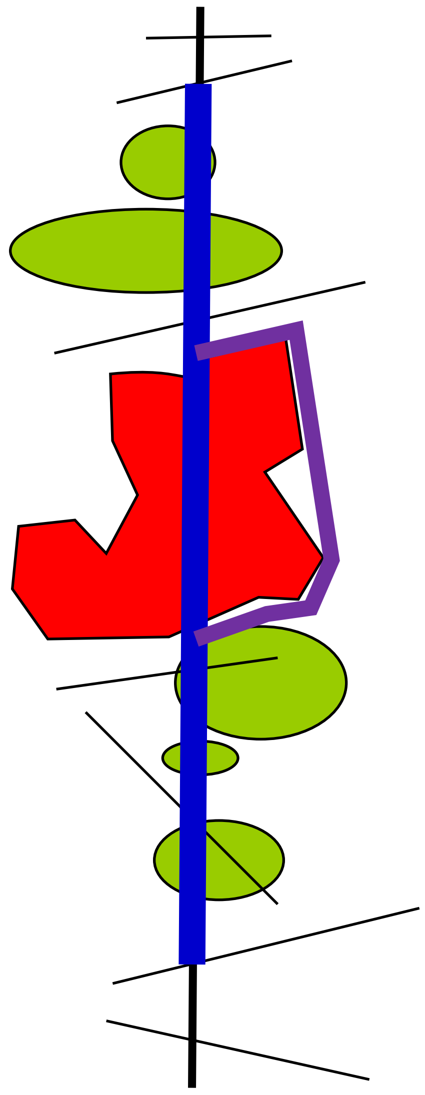
Fat obstacles: Prove m-guillotine PTAS applies to geodesic metric



TSPN Subproblem: A Window into OPT



TSPN with Obstacles: Key Issue



Bridge (as in m-guillotine method)

Obstacle

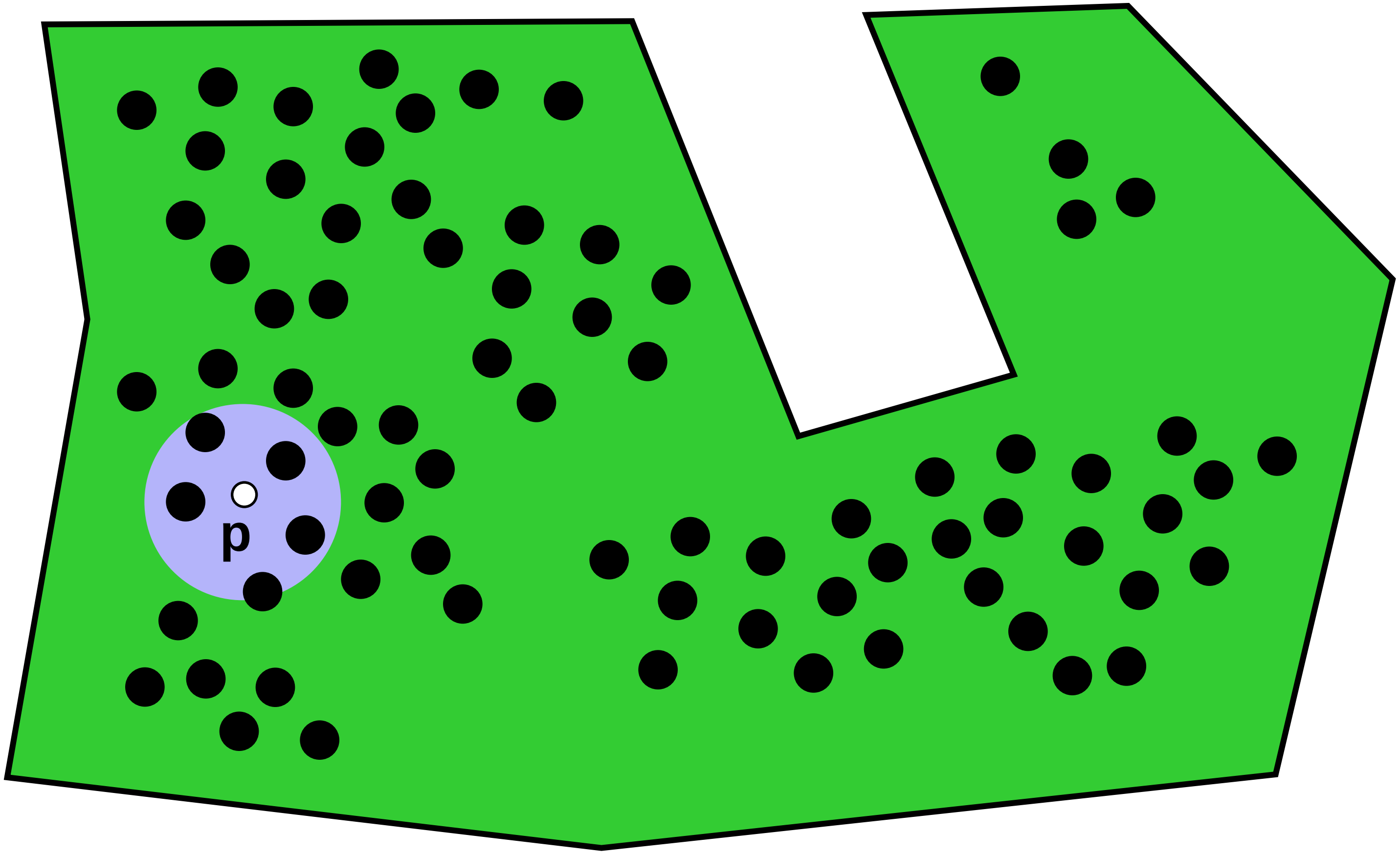
Detour (needed to keep the Bridge connected)

Sufficient: Obstacles are ***fat*** : then the detours to keep bridge connected cause only a constant-factor dilation to bridge length, which is charged off

Forest Assumptions

Either: (1) limited view distance

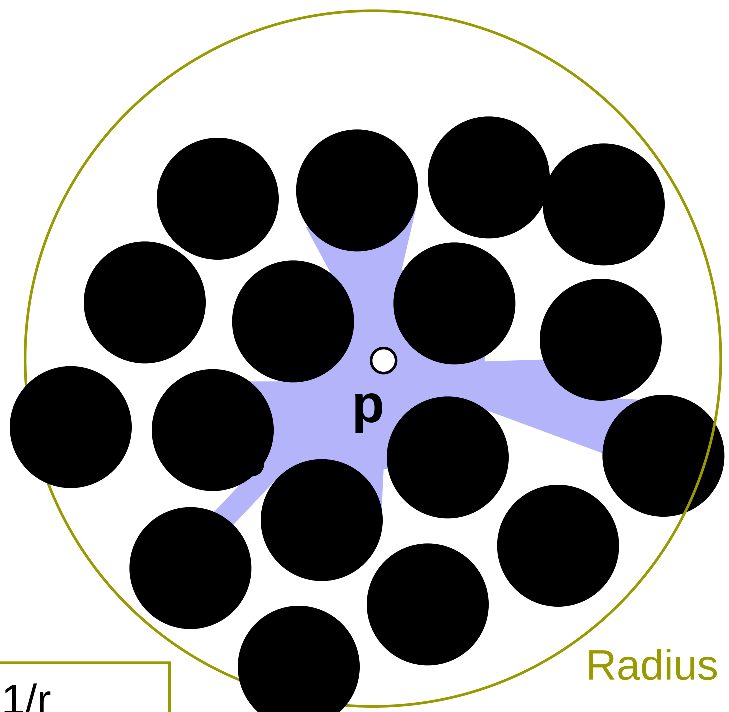
Require robot to get within distance ***R*** of a point ***p*** in order to see it



Forest Assumptions

Or: (2) forest is dense enough (e.g., maximal packing) so that the visibility region from a point deep inside the forest is a **fat** (star-shaped) region.

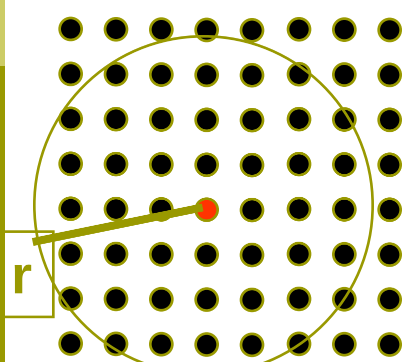
Time: $O(n^{O(R)})$



Dark Forest Conjecture: For R < const, there exists a dark point p

Recently shown!: R < const [Dumitrescu and Jiang, 2009]

$R < 2*10^{108}$



Dark if tree radius > 1/r

Related to Polya's Orchard Problem

Olber's paradox [1826]