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
Stationary Guards
The Art Gallery Problem

Determine a small set of “guards” to see all of a given $n$-vertex polygon $P$  

**Art Gallery Thm:**  
$\left\lceil \frac{n}{3} \right\rceil$ guards suffice and are sometimes necessary  

Motivation: Sensor coverage, security
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!

• 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]

**Art Gallery Thm:** $\left\lceil \frac{n}{3} \right\rceil$ guards suffice and are sometimes necessary
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 $V(P) = \text{vertices of polygon } P$
   
   (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 $O(n^4)$ ]

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

Centers of cells in arrangement of edge extensions

Visibility extensions for VG edge \((u, v)\)
Phase 2: Greedily Selecting Candidates

- Set of candidates: \( W(P) \)
- Greedily add “good” candidates \( g \in W(P) \) until \( P \) is covered: \( \text{Max } \mu(g) \quad g \in W(P) \)
- At end, iteratively remove redundant guards until set is \textit{minimal}
Heuristics Used in Experimentation

- $A_1$:
  
  Candidates $W(P) = V(P) \cup C(P)$

  Score $\mu(g) = \# \text{ unseen candidates}$

  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).
Heuristics Used in Experimentation

A₃ : Like A₁ but: Score \( \mu(g) = \text{area} \) newly seen
A₄ : Like A₁ but: \( \mu(g) \) weighted by cell area
A₅ : Like A₄ but: \( \mu(g) \) weighted by shared bd(P)
A₆ : Like A₄ but: \( \mu(g) \) weighted by \% of shared bd(P)
A₇ : Like A₁ but: Candidates \( W(P) = V(P) \)
A₈ : Like A₁ but: Candidates \( W(P) = C(P) \)
A₉ : Like A₁ but: \( \mu(g) = \# \) newly seen vertices
A₁₀ : Like A₁ but: \( \mu(g) = \# \) newly seen cell centers
A₁₁ : Like A₁ but: Arrangement of visibility extensions
A₁₂ : Combination of A₂ and A₁₁

(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 \( P \)

\[ g(P) \geq 4 \]
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: \( p \in \text{bd}(P) \) 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
Examples: \( n=100 \)

\[ A_1 \quad A_2 \quad A_{11} \]
More Examples

A_1

(a)

(b)

(c)

(d)

(e)

(f)

Spike box
More Examples

$A_1$
More Examples

$A_1$

(a)  
(b)  
(c)  
(d)  
(e)  
(f)  
(g)  
(h)  
(i)  
Comparison of Heuristics

Results on 40 polygons:

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

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

K - average **excess** = number of guards more than the \textit{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 \textbf{best} among all heuristics

B - number of completed tests
### Comparison of Heuristics

<table>
<thead>
<tr>
<th>PN</th>
<th>V</th>
<th>RV</th>
<th>H</th>
<th>VH</th>
<th>RVH</th>
<th>A1</th>
<th>A2</th>
<th>A4</th>
<th>A5</th>
<th>A6</th>
<th>A7</th>
<th>A8</th>
<th>A9</th>
<th>A10</th>
<th>A11</th>
<th>A12</th>
<th>A13</th>
<th>A14</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>24</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>4</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>7</td>
<td>3</td>
<td>12</td>
<td>12</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>7</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>7</td>
<td>7</td>
<td>8</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>44</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>13</td>
<td>13</td>
<td>12</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>13</td>
<td>13</td>
<td>11</td>
<td>12</td>
<td>12</td>
<td>11</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>83</td>
<td>52</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>10</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>22</td>
</tr>
<tr>
<td>6</td>
<td>33</td>
<td>19</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>24</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>0</td>
<td>12</td>
<td>60</td>
<td>48</td>
<td>13</td>
<td>13</td>
<td>14</td>
<td>14</td>
<td>16</td>
<td>12</td>
<td>16</td>
<td>18</td>
<td>15</td>
<td>9</td>
<td>9</td>
<td>14</td>
<td>X</td>
</tr>
<tr>
<td>9</td>
<td>17</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>6</td>
<td>2</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>0</td>
<td>13</td>
<td>48</td>
<td>44</td>
<td>15</td>
<td>13</td>
<td>17</td>
<td>17</td>
<td>16</td>
<td>15</td>
<td>16</td>
<td>18</td>
<td>14</td>
<td>13</td>
<td>X</td>
<td>15</td>
<td>X</td>
</tr>
<tr>
<td>13</td>
<td>16</td>
<td>6</td>
<td>4</td>
<td>16</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>7</td>
<td>9</td>
<td>8</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>X</td>
</tr>
<tr>
<td>14</td>
<td>6</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>X</td>
</tr>
<tr>
<td>15</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>20</td>
<td>14</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>13</td>
<td>9</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>15</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>49</td>
<td>24</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>7</td>
<td>6</td>
<td>7</td>
<td>7</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>28</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>X</td>
</tr>
<tr>
<td>19</td>
<td>100</td>
<td>46</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>13</td>
<td>12</td>
<td>16</td>
<td>16</td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>X</td>
<td>X</td>
<td>13</td>
<td>13</td>
<td>15</td>
<td>X</td>
</tr>
<tr>
<td>20</td>
<td>100</td>
<td>47</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>18</td>
<td>15</td>
<td>19</td>
<td>19</td>
<td>18</td>
<td>15</td>
<td>15</td>
<td>X</td>
<td>X</td>
<td>16</td>
<td>16</td>
<td>18</td>
<td>20</td>
</tr>
<tr>
<td>21</td>
<td>100</td>
<td>44</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td>15</td>
<td>18</td>
<td>17</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>X</td>
<td>X</td>
<td>16</td>
<td>16</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>22</td>
<td>100</td>
<td>50</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>14</td>
<td>14</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>14</td>
<td>X</td>
<td>X</td>
<td>14</td>
<td>14</td>
<td>16</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>100</td>
<td>45</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>18</td>
<td>18</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
<td>X</td>
<td>X</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>100</td>
<td>55</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td>16</td>
<td>18</td>
<td>18</td>
<td>16</td>
<td>16</td>
<td>X</td>
<td>X</td>
<td>15</td>
<td>15</td>
<td>16</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>100</td>
<td>49</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>18</td>
<td>17</td>
<td>18</td>
<td>18</td>
<td>18</td>
<td>18</td>
<td>X</td>
<td>X</td>
<td>17</td>
<td>17</td>
<td>22</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>100</td>
<td>46</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>17</td>
<td>16</td>
<td>19</td>
<td>18</td>
<td>18</td>
<td>17</td>
<td>X</td>
<td>X</td>
<td>16</td>
<td>16</td>
<td>17</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>100</td>
<td>49</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>12</td>
<td>12</td>
<td>15</td>
<td>16</td>
<td>13</td>
<td>15</td>
<td>12</td>
<td>X</td>
<td>X</td>
<td>14</td>
<td>14</td>
<td>18</td>
<td>16</td>
</tr>
<tr>
<td>28</td>
<td>100</td>
<td>52</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>14</td>
<td>16</td>
<td>15</td>
<td>15</td>
<td>17</td>
<td>17</td>
<td>15</td>
<td>X</td>
<td>X</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>19</td>
</tr>
<tr>
<td>29</td>
<td>10</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>10</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>10</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>
### Comparison of Heuristics

<table>
<thead>
<tr>
<th>PN</th>
<th>$A_1$</th>
<th>$A_2$</th>
<th>$A_{11}$</th>
<th>$I_1$</th>
<th>$I_2$</th>
<th>$I_3$</th>
<th>min ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>X</td>
<td>1.75</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>7</td>
<td>7</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>1.09</td>
</tr>
<tr>
<td>4</td>
<td>13</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>8</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>2</td>
<td>X</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>13</td>
<td>13</td>
<td>9</td>
<td>8</td>
<td>1</td>
<td>8</td>
<td>1.125</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1.5</td>
</tr>
<tr>
<td>11</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>7</td>
<td>2</td>
<td>6</td>
<td>1.85</td>
</tr>
<tr>
<td>13</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>1.2</td>
</tr>
<tr>
<td>16</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>1.25</td>
</tr>
<tr>
<td>17</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>18</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>X</td>
<td>1.25</td>
</tr>
<tr>
<td>19</td>
<td>13</td>
<td>12</td>
<td>13</td>
<td>11</td>
<td>11</td>
<td>6</td>
<td>1.09</td>
</tr>
<tr>
<td>20</td>
<td>18</td>
<td>15</td>
<td>16</td>
<td>11</td>
<td>12</td>
<td>8</td>
<td>1.25</td>
</tr>
<tr>
<td>21</td>
<td>16</td>
<td>15</td>
<td>16</td>
<td>13</td>
<td>12</td>
<td>7</td>
<td>1.15</td>
</tr>
<tr>
<td>22</td>
<td>14</td>
<td>15</td>
<td>14</td>
<td>9</td>
<td>9</td>
<td>5</td>
<td>1.55</td>
</tr>
<tr>
<td>23</td>
<td>18</td>
<td>18</td>
<td>17</td>
<td>13</td>
<td>13</td>
<td>5</td>
<td>1.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PN</th>
<th>$A_1$</th>
<th>$A_2$</th>
<th>$A_{11}$</th>
<th>$I_1$</th>
<th>$I_2$</th>
<th>$I_3$</th>
<th>min ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>25</td>
<td>18</td>
<td>17</td>
<td>17</td>
<td>16</td>
<td>16</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>26</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4.125</td>
</tr>
<tr>
<td>27</td>
<td>12</td>
<td>12</td>
<td>14</td>
<td>13</td>
<td>13</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>28</td>
<td>14</td>
<td>16</td>
<td>16</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>29</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>31</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>32</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>33</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>34</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>35</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>36</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>37</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>38</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>39</td>
<td>10</td>
<td>9</td>
<td>9</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>40</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>41</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>42</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>43</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>44</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>45</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>46</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>47</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

27
Comparison of Heuristics

<table>
<thead>
<tr>
<th>PN</th>
<th>V</th>
<th>RV</th>
<th>H</th>
<th>VH</th>
<th>RVH</th>
<th>A₁</th>
<th>A₂</th>
<th>A₁₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>41</td>
<td>24</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>42</td>
<td>73</td>
<td>40</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>19</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>43</td>
<td>4</td>
<td>0</td>
<td>17</td>
<td>48</td>
<td>48</td>
<td>15</td>
<td>12</td>
<td>14</td>
</tr>
<tr>
<td>44</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>57</td>
<td>26</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>45</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>44</td>
<td>26</td>
<td>11</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>46</td>
<td>20</td>
<td>8</td>
<td>4</td>
<td>24</td>
<td>20</td>
<td>9</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>47</td>
<td>24</td>
<td>10</td>
<td>8</td>
<td>32</td>
<td>32</td>
<td>13</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>48</td>
<td>46</td>
<td>21</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>11</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>49</td>
<td>40</td>
<td>19</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>50</td>
<td>28</td>
<td>12</td>
<td>4</td>
<td>32</td>
<td>24</td>
<td>7</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>51</td>
<td>40</td>
<td>120</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>52</td>
<td>200</td>
<td>101</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>35</td>
<td>X</td>
<td>33</td>
</tr>
<tr>
<td>53</td>
<td>200</td>
<td>95</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>31</td>
<td>X</td>
<td>31</td>
</tr>
<tr>
<td>54</td>
<td>200</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>31</td>
<td>X</td>
<td>27</td>
</tr>
<tr>
<td>55</td>
<td>200</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>28</td>
<td>X</td>
<td>31</td>
</tr>
<tr>
<td>56</td>
<td>200</td>
<td>104</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>31</td>
<td>X</td>
<td>31</td>
</tr>
<tr>
<td>57</td>
<td>200</td>
<td>97</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>26</td>
<td>X</td>
<td>26</td>
</tr>
<tr>
<td>58</td>
<td>200</td>
<td>98</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>26</td>
<td>X</td>
<td>27</td>
</tr>
<tr>
<td>59</td>
<td>200</td>
<td>98</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>X</td>
<td>31</td>
</tr>
<tr>
<td>60</td>
<td>200</td>
<td>98</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>26</td>
<td>X</td>
<td>26</td>
</tr>
<tr>
<td>61</td>
<td>200</td>
<td>99</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>31</td>
<td>X</td>
<td>30</td>
</tr>
</tbody>
</table>
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 \( \varepsilon \)

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/\(\varepsilon\) 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.

\[
\lambda = \left(\frac{g_{\text{opt}}}{\varepsilon_1}\right) \log h \quad (h = \# \text{ holes})
\]

[AEG08] also give randomized greedy algorithm that, whp, computes \(O(g_L \log \lambda)\) guards to cover L, where \(g_L \leq g_{\text{opt}}\) 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($VP(g)$) $\geq c \text{ diam}^2(VP(g))$, for some $c$. (e.g., each $VP(g)$ contains a disk of radius $\Omega(\text{diam}(VP(g)))$)

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

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

Here, the graph $VG(S)$ is not planar, not UDG, but has bounded degree, depending on $R/\delta$
Guarding Polyominoes

- 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

[Irfan, Iwerks, Kim, M]
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

   **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**
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 $m = 1, 2, 3, \text{ or } 4$, 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 $S_i \in \{(3), (4), ((5)/5*, 5**), (6), (7), 8*, 9*) \ (i = 1, 2, 3, ..., f)$ can be removed from $P$ yielding connected pixel polygons $P_1, P_2, ..., P_f$ 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 $\left\lceil \frac{m}{3} \right\rceil$ point guards is sufficient to guard an $m$-pixel connected polygon $P$. Actually, \( \text{ceil}((m-1)/3) \).
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 \([\text{CN,DELM}]\)
  - Best time bound: \(O(n^3 \log n)\) \([\text{DELM}]\)
- NP-hard in polygons with holes
  - No approx algorithm known in general!
  - Rectilinear visibility: \(O(\log n)\)-approx \([\text{MM’95}]\)
  - NEW: For fat obstacles, PTAS to see at least one point on the boundary of each obstacle
- 3D: Depends on 3D TSPN \([\text{ADDFM}]\)

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 $P_1, P_2, \ldots, P_n$
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

Bridges

$m = 4$

Region-Bridges

$M = 3$
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
Either: (1) limited view distance

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

Forest Assumptions

Related to Polya’s Orchard Problem

Olber’s paradox [1826]

Dark if tree radius > 1/r

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

Dark Forest Conjecture:
For $R < \text{const}$, there exists a dark point $p$

Recently shown!: $R < \text{const}$
[Dumitrescu and Jiang, 2009]

$R < 2 \times 10^{108}$