An Exact and Efficient Algorithm for the Orthogonal Art Gallery Problem

Marcelo C. Couto
marcelo.couto@students.ic.unicamp.br

Cid C. de Souza*
cid@ic.unicamp.br

Pedro J. de Rezende†
rezende@ic.unicamp.br

Institute of Computing
State University of Campinas
Campinas, Brazil

Abstract

In this paper, we propose an exact algorithm to solve the Orthogonal Art Gallery problem in which guards can only be placed on the vertices of the polygon P representing the gallery. Our approach is based on a discretization of P into a finite set of points in its interior. The algorithm repeatedly solves an instance of the Set Cover problem obtaining a minimum set Z of vertices of P that can view all points in the current discretization. Whenever P is completely visible from Z, the algorithm halts; otherwise, the discretization is refined and another iteration takes place. We establish that the algorithm always converges to an optimal solution by presenting a worst case analysis of the number of iterations that could be effected. Even though these could theoretically reach $O(n^4)$, our computational experiments reveal that, in practice, they are linear in n and, for $n \leq 200$, they actually remain less than three in almost all instances.

Furthermore, the low number of points in the initial discretization, $O(n^2)$, compared to the possible $O(n^4)$ atomic visibility polygons, renders much shorter total execution times. Optimal solutions found for different classes of instances of polygons with up to 200 vertices are also described.

1. Introduction and Related Work

The classical *Art Gallery Problem* originally posed by Victor Klee in 1973 asked for determining the minimum number of guards sufficient to cover the interior of an n-wall art gallery [12]. Soon thereafter, Chvátal established what became known as *Chvátal's Art Gallery Theorem*: $\lfloor \frac{n}{3} \rfloor$ guards are occasionally necessary and always sufficient to cover a simple polygon with n vertices [3]. A simpler proof based on polygon triangulation and on the fact that this triangulation can be 3-colored was shown by Fisk in 1978 [10].

Many variations of the art gallery problem have been studied in the literature. Early on, Lee and Lin [17] proved that the vertex guards version is NP-hard by reduction from 3-SAT. Their result was extended to point guards by Aggarwal [1].

Since these early results, much research on related problems has been carried out by mathematicians and computer scientists. For broad surveys, the reader is referred to [18, 21, 25] where comprehensive analysis of many variations and results on this subject can be found.

In this paper, we study a variation of the classical art gallery problem, called *Orthogonal Art Gallery Problem*, which is an important subclass, due to most real life buildings and galleries being orthogonally shaped [25]. This problem deals specifically with the case where the guards can only be placed on the vertices of the polygon that defines the outer boundary of the gallery and whose edges are parallel to the x or y axis.

The earliest major result concerning this problem, due to Kahn *et al.* [15], states that $\lfloor \frac{n}{4} \rfloor$ guards are occasionally necessary and always sufficient to cover an orthogonal polygon with n vertices. Their proof is based on quadrilateralization and on a 4-coloring of the resulting graph.

More importantly, Schuchardt and Hecker proved that minimizing the number of guards in this variation is also NP-hard [20], settling a question that remained open for almost a decade [19].

Improvements on the efficiency of algorithms that place exactly $\lfloor \frac{n}{4} \rfloor$ guards, such as Edelsbrunner *et al.* [6] and Sack and Toussaint [19] who showed a linear time guard placement algorithm for monotone orthogonal polygons as well as an $O(n \log \log n)$ time algorithm for arbitrary orthogonal polygons, still do not treat the challenging problem of minimization.

In this line, as asserted by Urrutia [25], one approach

* Partially supported by CNPq – Conselho Nacional de Desenvolvimento Científico e Tecnológico – Grants # 307773/2004-3 and 471460/2004-4 and FAPESP – Fundação de Amparo à Pesquisa do Estado de São Paulo – Grant # 107/97

† Partially supported by CNPq – Conselho Nacional de Desenvolvimento Científico e Tecnológico – Grant # 201205/2005-0.

that has not been sufficiently undertaken in the study of art gallery problems, is the one of finding algorithms that seek approximate solutions.

One of the first known results on this topic, due to Ghosh [11] in 1987, is an $O(n^5 \log n)$ time approximation algorithm, that finds a vertex guard set that is at most $O(\log n)$ times the minimum number of vertex guards needed, regardless of whether the polygon contains holes or not. Ghosh's work was later extended by Eidenbenz (see [7]) who designed approximation algorithms and heuristics for several variations of terrain guarding problems.

Recently, Erdem and Sclaroff in [8, 9] and Tomás *et al.* in [23, 24] modeled the problem as a discrete combinatorial problem and then solved the corresponding optimization problem. The results we present here follow this line.

Our Contribution. In this paper, we propose an algorithm to find a solution to the *Orthogonal Art Gallery Problem* by solving and refining discretizations of the boundary polygon. We show that the algorithm always produces a correct solution and we demonstrate, by a sizeable experimental analysis of its performance, that this algorithm derives its notable efficiency from the fact that the number of iterations required to achieve an optimal solution is very small, together with the fact that we deal with relatively few discretized points. We also present additional results regarding its implementation.

2. Preliminaries

In this paper, we address the *Orthogonal Art Gallery Problem* in which, given an orthogonal simple polygon that bounds an art gallery, the goal is to determine the minimum number and an optimal placement of (vertex) guards so as to keep the whole gallery under surveillance. In order to establish a uniform notation, whenever we speak of an n-*wall orthogonal art gallery* we are referring to a planar region whose boundary consists of an orthogonal simple polygon (without holes), i.e., one whose n edges are parallel to the x or y axis. Given one such polygon P, we denote by V the set of vertices of P. A vertex $v \in V$ is called *reflex* if the internal angle at v is greater than $180\,^\circ$. Whenever no confusion arises, *a point in* P will mean a point either in the interior or on the boundary of P.

Given two points x and y in a simple polygon P, we say that y is visible from x if and only if the closed segment $\overline{xy}$ does not intersect the exterior of P. The set $V(v)$ of all points of P visible from a vertex $v \in V$ is called the *visibility region of* v. To determine the visibility region of a vertex v we employ the linear time algorithm proposed by Lee [16] and extended by Joe and Simpson [13, 14].

A set of points G is a *guard set* for P if for every point $p \in P$ there exists a point $g \in G$ such that p is visible from g. In other words, a guard set for P gives the positions of stationary guards who can oversee an entire art gallery of boundary P. Hence, the Orthogonal Art Gallery problem amounts to finding the smallest subset $G \subset V$ that is a guard set for P.

2.1. The discretized orthogonal art gallery problem

Approximate solutions to the Orthogonal Art Gallery problem can be obtained by modeling it as a discrete combinatorial problem, the *Minimum Set Cover problem*, as shown by Erdem and Sclaroff in [8, 9] and by Tomás *et al.* in [23, 24]. Both approaches discretize the polygon P in some way and then solve the corresponding optimization problem. The latter uses Constraint Programming to solve the problem, while the former applies Integer Programming.

In our formulation, we discretize polygon P into a set of points $D(P)$ and use a simplification model from [8, 9] as follows.

$$
\begin{aligned}
z \;=\; & \min \quad \sum_{j \in V} x_j \\
& \text{s.t.} \quad \sum_{j \in V} a_{ij} x_j \geq 1, \forall p_i \in D(P) \qquad (1) \\
& \qquad\quad x_j \in \{0,1\}, \forall j \in V \qquad (2)
\end{aligned}
$$

where

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

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

The set $Z = \{j \in V \mid x_j = 1\}$ is called the solution set. Constraint (1) states that each point $p_i \in D(P)$ is visible from at least one selected guard position in the solution and the objective function minimizes the cardinality z of the solution set Z.

Thus, instead of solving the problem based on the entire polygon P, a discretization $D(P)$ is employed to generate a small number of constraints in the formulation. Clearly, it may happen that the solution of the discrete problem does not form a guard set for P. However, one can think of Z as an approximation to the original problem whose quality can be measured by the area of the regions not visible from the points in Z. Furthermore, the uncovered regions depend on the choice of the discretization. Among the various possible methods to discretize P, we focus on and extend the approach from [8] in which a regular grid is built prior to sampling the polygon. Refining the grid increases its resolution

but also increases the number of constraints in (1). This constitutes an interesting tradeoff between speed and accuracy for the discrete approximation. The way we address this issue in our algorithm is further discussed below.

2.2. Initial discretization of P

We now describe how we build the first discretization of P. Consider the regular grid with resolution $\Delta_x \times \Delta_y$ starting at the lower left corner of the bounding box of P, where
$\Delta_x = min\{|v_x - u_x|,\ v_x \neq u_x\ :\ u, v \in V\}$ and
$\Delta_y = min\{|v_y - u_y|,\ v_y \neq u_y\ :\ u, v \in V\}$.

The intersection points of this grid that are interior to P form the initial discretization $D(P)$ of P. Below, we justify why this is a particularly good choice.

Consider the method developed by Culberson and Reckhow [4, 5] in which one partitions the interior of P by extending the two edges adjacent to each reflex vertex within the polygon until the first boundary edge of P is reached. These line segments induce a subdivision of P into rectangles, called *basic regions* of P.

In Figure 1, we illustrate these definitions by showing a polygon, within a 7×6 bounding box, whose basic regions are bounded by segments from the boundary of the polygon and by dashed lines. A regular grid of resolution 2×2 whose segments are represented by solid gray lines is superimposed.

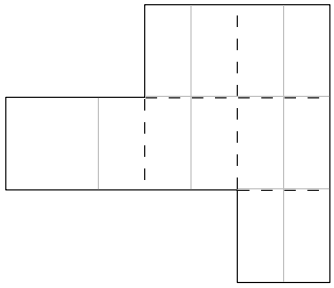

Figure 1. Regular grids and basic regions.

In the next proposition, we show that each basic region contains at least one point from the regular grid.

Proposition 2.1 *Let the regular grid $D(P)$ be the discretization of P with resolution $\Delta_x \times \Delta_y$ as defined above. Then, every basic region of P contains at least one point from $D(P)$.*

Proof. Let B be any basic region of P. From its construction, B is a rectangle with sides $\ell_x \geq \Delta_x$ and $\ell_y \geq \Delta_y$. Let *LL* be the lower-left and *UR* be the upper-right vertices of B. It follows that $UR_x \geq LL_x + \Delta_x$ and $UR_y \geq LL_y + \Delta_y$.

If *LL* is a vertex of P, let $p = LL$. Otherwise, let $p \in D(P)$ be the closest grid point to *LL* by the Manhattan (or L_1) distance dominated by *LL*, i.e., so that $p_x \leq LL_x$ and $p_y \leq LL_y$ (see Figure 2).

Now, from the definitions of Δ_x and Δ_y, the point $q = (p_x + \Delta_x, p_y + \Delta_y)$ is a grid point.

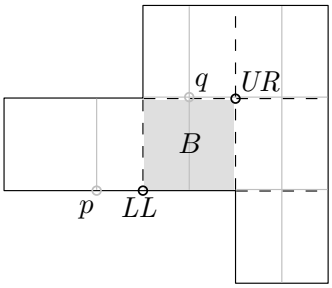

Figure 2. Illustration for the proof of Proposition 2.1.

Since $LL_x < q_x \leq LL_x + \Delta_x \leq UR_x$ and $LL_y < q_y \leq LL_y + \Delta_y \leq UR_y$, we have $q \in B$. □

Therefore, every basic region will have non empty intersection with the visibility polygon of at least one vertex guard from a solution to the discretized formulation, which makes the regular grid a good starting point to obtain an optimal solution.

Finally, we add to $D(P)$ all the vertices in V to form the complete discretization of the polygon P that we sought.

3. Algorithm

As mentioned earlier, a solution set Z to the discretized formulation in Section 2 may not always constitute a guard set for P since there might be regions inside P that are not visible from any guard in Z.

Definition 3.1 *Let $I(P, D(P))$ be an instance of the discretized Orthogonal Art Gallery problem with polygon P as the gallery boundary and $D(P)$ a discretization of P. A solution Z of this instance is called* viable *if Z is a guard set for P, i.e., $\bigcup_{g \in Z} V(g) = P$.*

Our algorithm overcomes the aforementioned drawback of the solution to the original discretized formulation and always produces a viable solution by refining the discretization of P whenever it detects that the present solution is not viable. The following theorem establishes that a solution thus obtained is also optimal.

Theorem 3.1 *Let Z be a solution of an instance $I(P, D(P))$ of the discretized Orthogonal Art Gallery problem. If Z is viable then Z is optimal.*

Proof. Due to the fact that Z is a solution of the minimization problem $I(P, D(P))$, Z is optimal as a vertex guard cover for the set $D(P)$ of points which discretize the polygon P, i.e., $z = |Z|$ is minimal among the cardinalities of all vertex guard covers of $D(P)$.

Now, let Z^* be an optimal vertex guard set for P and let $z^* = |Z^*|$. 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 $z \geq z^*$. □

Theorem 3.1 states that when the algorithm finds a solution for the discretized formulation which is viable, that solution is also an optimal vertex guard cover for P, i.e., it is a guard set for P.

The algorithm is divided into two phases: a Preprocessing Phase, where the initial discretization described in Section 2 is constructed and the Integer Programming problem is set up, and a Solution Phase in which the discretized problem is successively solved and refined until a viable (and optimal) solution is found.

3.1. Preprocessing Phase

In order to assemble the formulation outlined in Section 2, we start by building a regular grid of resolution $\Delta_x \times \Delta_y$ restricted to the interior of the polygon P, to which we add the vertices of P, as described in Section 2.2.

Once this discretization is built, we compute which grid points are located inside the visibility region of each vertex in V, and, then, include these new restrictions in the formulation.

The main steps of the preprocessing phase are summarized in Algorithm 3.1. If the regular $\Delta_x \times \Delta_y$ grid $D(BB)$ has m points, the generation of the initial discretization $D(P)$, in step 3, requires $O(mn)$ time when identifying the grid points that lie inside P. The total complexity of step 6 is $O(n^2)$ [13] and, assuming that $m \in \Omega(n)$, the full complexity of step 8 is $O(nm \log n)$ since point location of each of the $O(m)$ points of $D(P)$ in a star-shaped visibility n-polygon can be accomplished in $O(\log n)$ time. Hence, the overall complexity of the preprocessing phase is dominated by that of step 8.

The result of the preprocessing phase is an Integer Programming (IP) formulation for the Set Cover Problem [26] which, once solved, generates a solution Z that, while not necessarily constituting a guard set for P, will always cover all the grid points in $D(P)$.

Algorithm 3.1 Preprocessing Phase

1: $BB \leftarrow$ bounding box of P;
2: $D(BB) \leftarrow \Delta_x \times \Delta_y$ regular grid generated as described in Section 2.2;
3: $D(P) \leftarrow D(BB) \cap P$;
4: $D(P) \leftarrow D(P) \cup V$;
5: **for** each $j \in V$ **do**
6: Compute $V(j)$;
7: **for** each grid point $p_i \in D(P)$ **do**
8: $a_{ij} \leftarrow$ **Boolean**$(p_i \in V(j))$;
9: **end for**
10: **end for**

3.2. Solution Phase

In the second phase of the algorithm, starting from the IP formulation generated in the preprocessing step, we solve the discretized instance followed by a refinement of the grid iteratively until the solution becomes viable. This refinement is attained by generating one more point in the discretization for each uncovered region (its centroid) and by adding the corresponding constraints to the current Set Cover model. These additional points enhance the regular grid and lead to a solution closer to a viable one. Algorithm 3.2 outlines the steps executed in the solution phase.

Algorithm 3.2 Solution Phase

1: **repeat**
2: $Z \leftarrow$ solution of $I(P, D(P))$;
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 (1) corresponding to the new uncovered point c:
$a_{rj}x_j \geq 1$ where, $\forall j \in V$,
$a_{rj} \leftarrow$ **Boolean**$(c \in V(j))$;
7: **end for**
8: **until** Z is viable

It remains to be argued that Algorithm 3.2 converges, as it will then follow from Theorem 3.1 that the algorithm is exact and the solution given is indeed a guard set for P. In order to determine the worst case for the number of iterations done by the algorithm, we proceed as follows.

Consider the set of all visibility regions of the vertices in V, whose union, obviously, covers P. The edges of these visibility regions induce a subdivision of P which is comprised of what is called *atomic visibility polygons*, or AVPs [11] (see Figure 3). Note that in step 3: any uncovered region (witness to the fact that Z does not cover the entire polygon) is necessarily a simple polygon formed by the union of neighboring AVPs. Furthermore, it follows from the construction of the AVPs that if the centroid of (or, for that matter, any point within) an atomic visibility polygon

$\mathcal{V}$ is visible from a vertex guard, the entire area of $\mathcal{V}$ must also be.

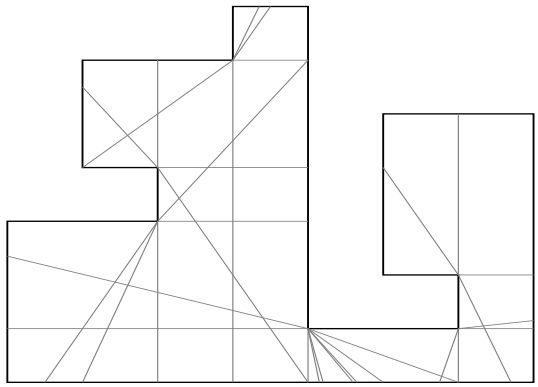

Figure 3. Atomic Visibility Polygons.

As the visibility region of any vertex can have at most $O(n)$ edges, the induced subdivision is generated from an arrangement of $O(n^2)$ lines and has a total complexity of no more than $O(n^4)$ faces (or AVPs).

Therefore, an upper bound on the maximum number of iterations effected by the algorithm is $O(n^4)$ and this establishes the convergence. Of course, in the worst case, each iteration can take exponential time as step 2: amounts to solving an instance of the Set Cover Problem.

Note that if $D(P)$ were built by taking one interior point from each AVP, the algorithm would halt right after the first iteration. In this case, the total complexity would be EXP(n^4). On the other hand, by starting with $D(P)$ as described in Section 2.2, the complexity of the algorithm is bound by EXP(n^2), provided that the required number of iterations remains within $O(1)$. In Section 5, we show that this assumption is true in practice.

4. Experimental Setup

This section presents the experimental setup for performance evaluation of our method. For the experiments conducted, we implemented the algorithms described in Section 3 along with a visibility algorithm from [13]. The implementation was done in `C++` on top of the CGAL 3.2.1, and used the Integer Linear Programming solver Xpress-Optimizer `v17.01.02`. The algorithm was tested on a PC featuring a Pentium IV at 2.66 GHz and 1 GB of memory.

Three sessions of tests were carried out, one for each of the three classes of polygons discussed in Section 4.1. Each session consisted of solving the Orthogonal Art Gallery problem for multiple instances. In each test, we computed several measures: preprocessing time spent by Algorithm 3.1, execution time and the number of iterations performed by Algorithm 3.2, and the number of guards in the optimal solution. Lastly, we compiled this information to show evidence of the efficiency of our algorithm, as is presented in Section 5.

4.1. Instances

We conducted the experiments on a large number of instances which consisted of n-vertex orthogonal polygons placed on an $\frac{n}{2} \times \frac{n}{2}$ unit square grid. These polygons were generated devoid of collinear edges, in accordance to the method described in [22].

We generated a total of 10282 sample polygons, each one having between 8 and 200 vertices, classifiable in three classes introduced in [24]: *random, small area* and *large area* polygons. See samples in Figure 4. Large area and small area polygons are the extreme scenarios for the IP approach in terms of the number of constraints in the model (see [24]). On one hand, large area polygons give rise to a very dense initial grid, while small area polygons lead to very few grid points. Recalling that orthogonal polygons have an even number of vertices, the data set used in our experiments was obtained thus: (a) 10088 Random polygons: for each k between 8 and 200, we generated k distinct polygons, with k vertices in each, by applying the algorithm for random orthogonal polygon generation described

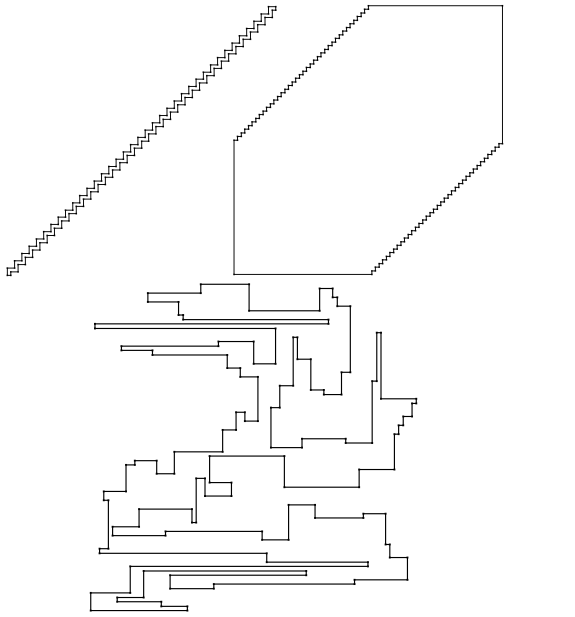

Figure 4. Sample polygons with 150 vertices: Small Area, Large Area and Random polygons.

in [22]; (b) 97 Small and 97 Large Area polygons: one for each number of vertices between 8 and 200.

5. Results

We now describe the experimental results that attest to the efficiency of the algorithm presented in Section 3. The testing was performed using the instances described in the previous section. Charts are employed to present plots of the data that summarize the statistical analysis.

Although we established that in the worst case the number of iterations of Algorithm 3.2 is bound by $O(n^4)$, in Figure 5 we show a histogram demonstrating that the practical results are quite impressive for polygons with 200 vertices or less. Indeed, the Boxplot [2] below reveals that in our experiments the median was only *one* iteration and that the third quartile is *two*. In fact, the algorithm finds an optimal solution for more than 99% of the instances within three iterations. This means that the instances of up to 200 vertices that need four or more iterations are outliers.

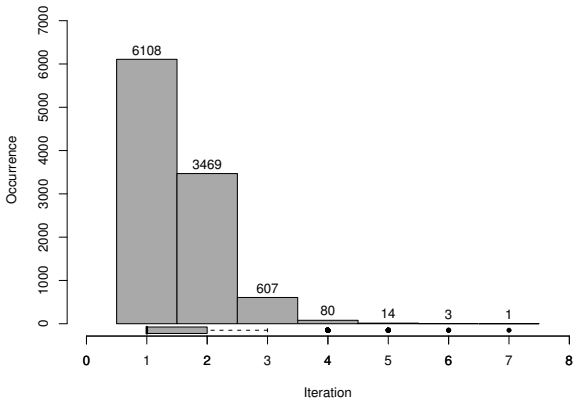

Figure 5. Histogram with a Boxplot showing how often each number of iterations occur.

Figure 6 shows another interesting result about the number of iterations required to achieve optimal solution to the test instances. Notwithstanding the claim by Tomás *et al.* in [24] that large area and small area polygons constitute extremal cases, our algorithm vanquishes these, at least in regard to the number of iterations to achieve optimal solutions — which turns out to be *one*. A further analysis of Figure 6 also indicates that the growth of the average number of iterations to solve instances of random polygons is linear and slow growing on n as its linear regression $0.003 \cdot n + 1.1$ asserts.

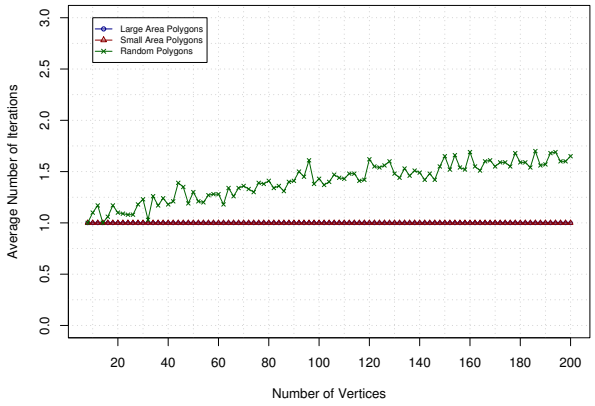

Figure 6. Average number of iterations to completely solve instances of different sizes.

In Figure 7 we plot the average size of the guard sets as a function of the number of vertices, n. Here, we have evidence that, as one would expect, the result by Kahn *et al.* [15], which states that $\lfloor \frac{n}{4} \rfloor$ guards are occasionally necessary and always sufficient to cover an orthogonal polygon of n vertices, gives in practice a fairly distant upper bound on the average for these polygons. Figure 7 also shows that large area polygons need no more than *two* guards in their optimal solutions.

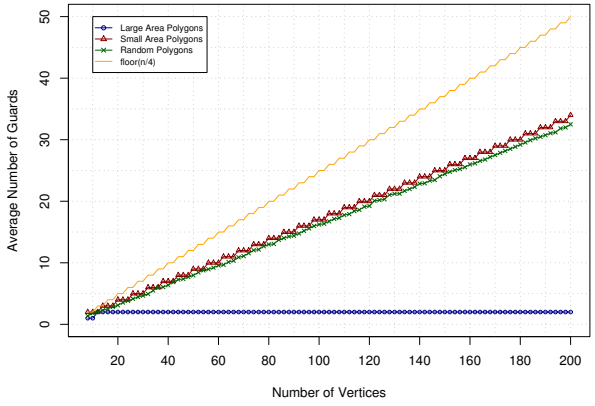

Figure 7. Average size of guard sets.

Timewise, the preprocessing phase, Algorithm 3.1, presents the behavior indicated in Figure 8, where it becomes evident that the large area polygons produce a much

denser regular grid than the other two types of polygons. The higher growth in this case comes from the fact that, as the number of vertices of P increases, the combinatorial complexity of the regular grid grows even faster since the area of the polygons available for intersections in step 3 of Algorithm 3.1 is quadratic in n. Thus, the size of the initial discretization $D(P)$ is $\Theta(n^2)$ and, according to the complexity analysis done in Section 3, the expected running time of Algorithm 3.1 for these polygons is $O(n^3 \log n)$. The plot for large area polygons in Figure 8 is consistent with this result.

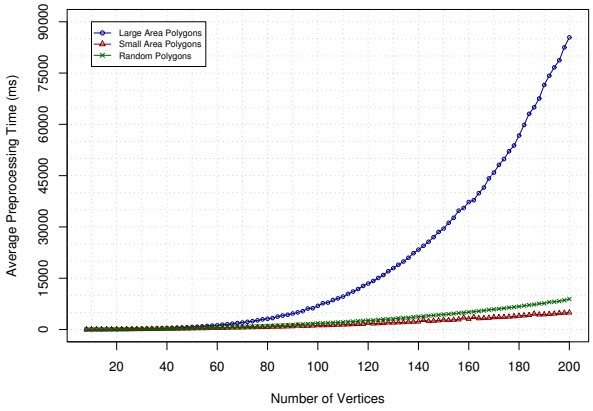

Figure 8. Average preprocessing time.

Lastly, Figure 9 shows the amount of time spent in the solution phase (Algorithm 3.2). For random polygons, the time taken grows almost linearly on the number of vertices because the average number of iterations needed to optimally solve those instances is also a linear function of n (refer back to Figure 6). One also notices that instances of small area polygons take only a negligible amount of time. Regarding large area polygons, the performance is hindered as a consequence of the higher density of the regular grid. A closer look at the results from the experiment with large area polygons revealed some interesting facts. Firstly, the solution phase described in Algorithm 3.2 always halted after *one* iteration. Secondly, Xpress, the IP solver used in step 2 of the algorithm, solves only one linear programming relaxation. This is because, using its internal heuristics, Xpress finds a feasible solution for the Set Cover instance which is proved to be optimal by the dual bound produced by the relaxation. As a result, when applied to large area polygons in our dataset, the solution phase of our algorithm only involved the solution of a linear programming problem with $O(n^2)$ constraints and $O(n)$ variables as well as the computation of the uncovered regions (step 3 of Algorithm 3.2). This explains why the graph of the running time of our algorithm for large area polygons has a polynomial shape. In fact, after a thorough investigation using regression analysis, we found that the polynomial that best fits the running time plot of our algorithm is given by $1.994\times10^{-7}n^5 - 7.920\times10^{-5}n^4 + 0.01238n^3 - 0.6864n^2 + 16.26n - 110.9$. The graph of this polynomial is displayed in Figure 9 and it should be remarked that even though its coefficients were obtained by considering only the observations for n ranging from 8 to 140, the resulting curve extrapolates well for the remaining of the observed range.

Despite the high order of the polynomial, considering that the Art Gallery problem for Orthogonal polygons is NP-hard, it is remarkable that, on the average, the most difficult instances with nearly 200 vertices that we tested could still be solved to optimality within 97 seconds, with only 10% of that time taken by the solution phase.

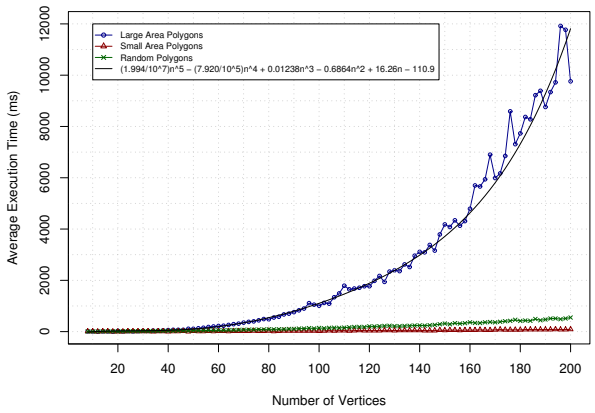

Figure 9. Average execution time.

6. Conclusion

We presented an exact and efficient algorithm for the *Orthogonal Art Gallery problem* when guards are restricted to the vertices of the gallery-bounding polygon. We also proved the theoretical results necessary to guarantee the convergence and the exactness of the method. Even though we presented the algorithm to address polygonal galleries, the same method can be directly extended to terrains as discussed in [7].

Our computational tests were performed on a large number of polygons from three different classes and the results showed that our algorithm is very efficient in practice: more than 99% of the instances with up to 200 vertices were

solved in no more than *three* iterations. In fact, we also obtained computational evidence to support the claim that the average number of iterations required by the algorithm to reach the optimal solution in any of those classes of orthogonal polygons is a slow linear function of the number of vertices. Furthermore, we were able to solve with a single iteration both classes of orthogonal polygons considered to possess extremal behavior in [24].

The tests also showed that the time spent by our algorithm in order to optimally solve the instances of random polygons is an almost linear function of the number of vertices; on the other hand, in the case of small area polygons, the algorithm uses no more than a negligible amount of time. Lastly, for large area polygons, the growth of the execution time as a function of the number of vertices turned out to be no more than a fifth degree polynomial.

Extensions to the present work will include the study of geometric properties that could lead to a reduction on the number of grid points and, hence, to a cutback of the preprocessing time and, as a consequence, to an improvement on the overall efficiency. Through further investigations we also intend to extend the method presented here to the case where guards can be placed on the interior of the edges of the polygons. One last obvious generalization is to obtain similar algorithms for non orthogonal polygons.

References

[1] A. Aggarwal, S. K. Ghosh, and R. K. Shyamasundar. Computational complexity of restricted polygon decompositions. In G. T. Toussaint, editor, *Computational Morphology*, pages 1–11. North-Holland, Amsterdam, Netherlands, 1988.

[2] J. M. Chambers, W. S. Cleveland, B. Kleiner, and P. A. Tukey, editors. *Graphical methods for data analysis*. The Wadsworth statistics/probability series. Wadsworth International, Duxbury Press, Boston, 1983.

[3] V. Chvátal. A combinatorial theorem in plane geometry. In *Journal of Combinatorial Theory Series B*, volume 18, pages 39–41, 1975.

[4] J. Culberson and R. Reckhow. Dent diagrams: A unified approach to polygon covering problems. Technical Report TR 87-14, Dept. Comput. Sci., Univ. Alberta, Edmonton, Alberta, Canada, 1987.

[5] J. Culberson and R. A. Reckhow. Covering a simple orthogonal polygon with a minimum number of orthogonally convex polygons. In *Proc. 3rd Annu. ACM Sympos. Comput. Geom.*, pages 268–277, 1987.

[6] H. Edelsbrunner, J. O'Rourke, and E. Welzl. Stationing guards in rectilinear art galleries. *Comput. Vision Graph. Image Process.*, 27:167–176, 1984.

[7] S. Eidenbenz. Approximation algorithms for terrain guarding. *Inf. Process. Lett.*, 82(2):99–105, 2002.

[8] U. M. Erdem and S. Sclaroff. Optimal placement of cameras in floorplans to satisfy task requirements and cost constraints. In *Proc. International Workshop on Omnidirectional Vision, Camera Networks and Non-classical Cameras*, pages 30–41, 2004.

[9] U. M. Erdem and S. Sclaroff. Automated camera layout to satisfy task-specific and floor plan-specific coverage requirements. *Comput. Vis. Image Underst.*, 103(3):156–169, 2006.

[10] S. Fisk. A short proof of Chvátal's watchman theorem. In *Journal of Combinatorial Theory Series B*, volume 24, page 374, 1978.

[11] S. K. Ghosh. Approximation algorithms for art gallery problems. In *Proc. Canadian Inform. Process. Soc. Congress*, 1987.

[12] R. Honsberger. *Mathematical Gems II*. Number 2 in The Dolciani Mathematical Expositions. Mathematical Association of America, 1976.

[13] B. Joe and R. B. Simpson. Visibility of a simple polygon from a point. Report CS-85-38, Dept. Math. Comput. Sci., Drexel Univ., Philadelphia, PA, 1985.

[14] B. Joe and R. B. Simpson. Correction to Lee's visibility polygon algorithm. *BIT*, 27:458–473, 1987.

[15] J. Kahn, M. M. Klawe, and D. Kleitman. Traditional galleries require fewer watchmen. *SIAM J. Algebraic Discrete Methods*, 4:194–206, 1983.

[16] D. T. Lee. Visibility of a simple polygon. *Comput. Vision, Graphics, and Image Process*, 22:207–221, 1983.

[17] D. T. Lee and A. K. Lin. Computational complexity of art gallery problems. *IEEE Trans. Inf. Theor.*, 32(2):276–282, 1986.

[18] J. O'Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, 1987.

[19] J.-R. Sack and G. T. Toussaint. Guard placement in rectilinear polygons. In G. T. Toussaint, editor, *Computational Morphology*, pages 153–175. North-Holland, Amsterdam, Netherlands, 1988.

[20] D. Schuchardt and H.-D. Hecker. Two NP-hard art-gallery problems for ortho-polygons. *Mathematical Logic Quarterly*, 41:261–267, 1995.

[21] T. C. Shermer. Recent results in art galleries. *Proceedings of the IEEE*, 80(9):1384–1399, 1992.

[22] A. P. Tomás and A. L. Bajuelos. Generating random orthogonal polygons. In *Current Topics in Artificial Intelligence*, volume 3040 of *Lecture Notes in Computer Science*, pages 364–373. Springer Berlin / Heidelberg, 2004.

[23] A. P. Tomás, A. L. Bajuelos, and F. Marques. Approximation algorithms to minimum vertex cover problems on polygons and terrains. In *Proceedings of the International Conference on Computational Science (ICCS 2003)*, volume 2657 of *Lecture Notes in Computer Science*, pages 869–878. Springer Berlin / Heidelberg, 2003.

[24] A. P. Tomás, A. L. Bajuelos, and F. Marques. On visibility problems in the plane - solving minimum vertex guard problems by successive approximations. In *Proceedings of the 9th International Symposium on Artificial Intelligence and Mathematics (AI & MATH 2006)*, 2006. to appear.

[25] J. Urrutia. Art gallery and illumination problems. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 973–1027. North-Holland, 2000.

[26] L. A. Wolsey. *Integer Programming*. Wiley-Interscience, 1998.