INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

An Exact Algorithm for an Art Gallery Problem

M. C. Couto and P. J. de Rezende and C. C. de Souza

Technical Report - IC-09-46 - Relatório Técnico

November - 2009 - Novembro

An Exact Algorithm for an Art Gallery Problem

Marcelo C. Couto Pedro J. de Rezende* Cid C. de Souza†

Abstract

We consider an Art Gallery problem (AGP) which aims to minimize the number of vertex guards required to monitor an art gallery whose boundary is an n-vertex simple polygon. In this paper, we compile and extend our research on exact approaches for solving the AGP. In prior works [1, 2], we proposed and tested an exact algorithm for the case of orthogonal polygons. In that algorithm, a discretization that approximates the polygon is used to formulate an instance of the Set Cover Problem which is subsequently solved to optimality. Either the set of guards that characterizes this solution solves the original instance of the AGP, and the algorithm halts, or the discretization is refined and a new iteration begins. This procedure always converges to an optimal solution of the AGP and, moreover, the number of iterations executed highly depends on the way we discretize the polygon. Notwithstanding that the best known theoretical bound for convergence is $\Theta(n^3)$ iterations, our experiments show that an optimal solution is always found within a small number of them, even for random polygons of many hundreds of vertices. Herein, we broaden the family of polygon classes to which the algorithm is applied by including non orthogonal polygons. Furthermore, we propose new discretization strategies leading to additional trade-off analysis of preprocessing vs. processing times and achieving, in the case of the novel *Convex Vertices* strategy, the most efficient overall performance so far. We report on experiments with both simple and orthogonal polygons of up to 2500 vertices showing that, in all cases, no more than 15 minutes are needed to reach an exact solution, on a standard desktop computer. Ultimately, we more than doubled the size of the largest instances solved to optimality compared to our previous experiments, which were already five times larger than those previously reported in the literature.

Keywords: art gallery problem, exact algorithm, set covering, visibility.

*Partially supported by CNPq – Conselho Nacional de Desenvolvimento Científico e Tecnológico – Grant #201205/2005-0, 472504/2007-0 and FAPESP – Fundação de Amparo à Pesquisa do Estado de São Paulo – Grant #07/52015-0 and a Grant from FAEPEX/UNICAMP.

†Partially supported by CNPq – Grants #307773/2004-3, 472504/2007-0 and FAPESP – Grant #107/97.

1 Introduction

According to Honsberger [3], in 1973, Victor Klee posed to Vasek Chvátal the question of determining the minimum number of watchmen sufficient to guard an art gallery shaped as an n-wall simple polygon. Chvátal's proof [4] that $\lfloor n/3 \rfloor$ guards are occasionally necessary and always sufficient for that purpose was the first result in what has become a whole new field of study. Witnessing to the broad spectrum of literature that has appeared since, we have O'Rourke's classical book [5], Shermer's and Urrutia's surveys [6, 7] and a multitude of journal papers cited within these. More recently, Ghosh's book [8], which presents a vast set of topics on visibility problems spun from related questions, cites over three hundred references, mostly from the last fifteen years.

Among the earliest and most important results, one finds Lee and Lin's NP-completeness proof of a related minimization problem [9] that remained open for more than ten years. Namely, given a planar simple polygon P, determine a placement of a *minimum* number of stationary guards that cover P.

Many variants of this problem have been considered in the literature. The formulation we study here restricts the placement of guards to vertices of the polygon that represents the outer boundary of a given art gallery. Throughout this paper, we will refer to this particular formulation as the *Art Gallery Problem* (AGP). Furthermore, we consider general simple polygons as well as the subclass of simple orthogonal polygons, which are particularly relevant due to most real life buildings and galleries being orthogonally shaped [7].

One of the earliest major result concerning the latter problem, due to Kahn *et al.* [10], states that $\lfloor n/4 \rfloor$ guards are occasionally necessary and always sufficient to cover an orthogonal polygon with n vertices. Later, Schuchardt and Hecker proved that minimizing the number of guards in this variation is also NP-hard [11], settling a question that remained open for almost a decade [12].

Several placement algorithms have been proposed in the past, such as Edelsbrunner *et al.* [13] and Sack and Toussaint [12], which deal with the problem of efficiently placing exactly $\lfloor n/4 \rfloor$ guards covering a given orthogonal gallery.

On the other hand, in a recently revised manuscript, based on [14], Ghosh presents an $O(n^4)$ time approximation algorithm for simple polygons yielding solutions within a $\log n$ factor of the optimal. Further approximation results include Eidenbenz [15] who designed algorithms for several variations of terrain guarding problems and Amit *et al.* [16] who analyze heuristics with experimental evidence of good performance in covered area as well as in the number of guards.

Another approach tackled by Erdem and Sclaroff [17] and Tomás *et al.* in [18] consists of modeling the problem as a discrete combinatorial problem and then solving the corresponding optimization problem. The former discretizes the interior of the polygon with a fixed grid, yielding an approximation algorithm and the latter gives empirical analysis of an exact method of successive approximations based on dominance of visibility regions.

Finally, in [1], we presented an exact algorithm to optimally solve the orthogonal AGP. In this algorithm, we iteratively discretize and model the problem as a classical *Set Cover problem* (SCP) citewolsey-book. Besides demonstrating the feasibility of this approach, we showed that, in practice, the number of iterations required to solve instances of up to 200 vertices was very small and that the resulting algorithm turned out to be quite efficient.

Though the number of iterations executed by the exact algorithm we proposed in [1] and improved in [2] was shown to be polynomially bounded, its practical performance is far better depending on how the polygon is discretized. This becomes evident when we notice that in each iteration an instance of SCP, an NP-hard problem, has to be solved to optimality, in our case, by an Integer Programming solver.

In this paper, we build upon all our previous studies and conduct a thorough experimental investigation concerning the trade-off between the number and nature of discretizing methods and the number of iterations and we analyze the practical viability of each approach. Moreover, while our previous works [1, 2] dealt only with orthogonal polygons, here we show that the same approach works well for general simple polygons. Besides dealing with new classes of polygons, two new and superior discretization strategies are introduced in this paper and are compared to the previously studied ones. All of our test data are available in [19] and include multiple instances for each size of the vertex set, for various classes of polygons with up to 2500 vertices.

The new experimental results significantly surpass those we reported in [1, 2]. This is due to the exploration of alternative discretization strategies, which allow us to address difficult instances as well as to handle a substantial increase in polygon size compared to earlier results, while still attaining low execution times.

In the next section, we describe the process of modeling the AGP as an SCP and the basic ideas necessary for the description of the algorithm which appears in Section 3, along with its proof of correctness and complexity. Section 4 is devoted to the description of the alternative strategies to discretize the input polygon. Next, in Section 5 we give an account of the set up of the testing environment and present the different classes of instances used. Complying with the recommendations of Johnson [20], McGeoch and Moret [21], Sanders [22] and Moret [23], we show in Section 5.2 an extensive experimental analysis of the algorithm considering multiple discretization strategies, and include an evaluation of various comparative measurements. Concluding remarks are drawn in the last section.

2 Modeling

In an instance of the AGP, we are given a simple polygon that bounds an art gallery and we are asked to determine the minimum number and an optimal placement of vertex guards in order to keep the whole gallery under surveillance. Vertex guards are assumed to have a range of vision of 360°.

The approach used by the algorithm described in Section 3 transforms the continuous AGP into a discrete problem which, in turn, can easily be modeled as an instance of the SCP. In fact, for the last two decades, this has been the only known technique for developing efficient approximation algorithms for the art gallery problem. Before we present our algorithm and establish its correctness, let us review some basic definitions.

An *n-wall art gallery* can be viewed as a planar region whose boundary consists of a simple polygon (without holes) P. The set of vertices of P is denoted V and a vertex $v \in V$ is called a *reflex vertex* if the internal angle at v is greater then 180°. Whenever no confusion arises, *a point in* P will mean a point either in the interior or on the boundary of P.

Any point y in P is said to be visible from any other point x in P if and only if the closed segment joining x and y does not intersect the exterior of P. The set $\mathrm{Vis}(v)$ of all points in P visible from a vertex $v \in V$ is called the *visibility region of* v. It is easy to see that $\mathrm{Vis}(v)$ is always a star shaped polygon. A boundary description of $\mathrm{Vis}(v)$ can be computed in linear time by an algorithm proposed by Lee [24] and extended by Joe and Simpson [25, 26].

A set of points S is a *guard set* for P if for every point $p \in P$ there exists a point $s \in S$ such that p is visible from s. Hence, a *vertex guard set* G is any subset of vertices such that $\bigcup_{g \in G} \mathrm{Vis}(g) = P$. In other words, a vertex guard set for P gives the positions of stationary guards who can oversee an entire art gallery of boundary P. Thus, the AGP amounts to finding the smallest subset $G \subset V$ that is a vertex guard set for P.

The reader who is familiar with the set cover problem (SCP) [27] must already have perceived that the problem of finding the smallest vertex guard set for P can be regarded as a specific SCP. Namely, we wish to find a smallest cardinality set of star-shaped polygons (visibility regions of the vertices of P) whose union cover P. Notice that, strictly speaking, this is a *continuous* SCP since there are infinitely many points in the interior of P to be covered. However, one can discretize the problem by generating a finite number of representative points in P so that the formulation becomes manageable. We shall see below how this approach will lead us to a viable solution of the original problem.

We now describe how the solutions to successively refined discrete instances are guaranteed to converge to an optimal solution to the continuous problem. To this end, consider an arbitrary discretization of P into a finite set of points $D(P)$. We will denote by $I(P, D(P))$ an instance of the discretized Art Gallery problem generated in this fashion. An IP formulation of the corresponding SCP instance is shown below.

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

where the binary variable x_j is set to 1 if and only if vertex j of P is chosen to be in the guard set. Moreover, given a point p_i in $D(P)$ and a vertex j of P, a_{ij} is a binary value which is 1 if and only if $p_i \in \mathrm{Vis}(j)$.

Given a feasible solution x to the IP above, let $Z(x) = \{j \in V \mid x_j = 1\}$. 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 $Z(x)$. Clearly, as the set $D(P)$ is finite, it may happen that $Z(x)$ does not form a vertex guard set for P. In this case, we must pick a new point inside each uncovered region and include these points in $D(P)$. A new SCP instance is then created and the corresponding IP is solved, leading to an iterative procedure. At the end of Section 3.2 we establish the convergence of this process.

The actual number of iterations that are required depends on how many uncovered regions might be successively generated. As the cost of each iteration is related to the number of constraints in (1), an interesting trade-off naturally sprouts and leads one to attempt multiple choices of discretization schemes. On the other hand, any method of cleverly choosing the initial points of the discretization will have a corresponding cost in preprocessing time, opening another intriguing time exchange consideration. These questions are precisely what we address in Section 4 where we consider several possible discretization schemes which lead to the various performance analysis discussed in Section 5.

3 Description of the algorithm and proof of correctness

The algorithm is divided into two phases: a *Preprocessing Phase*, where the initial discretization described in Section 1 is constructed and the Integer Programming problem is set up, and a *Solution Phase* in which the algorithm iterates as described above, solving SCP instances for the current discretization, until no regions remain uncovered.

As mentioned earlier, a solution set $Z(x)$ 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 $Z(x)$.

To formalize our subsequent reasoning, we start with the following definition.

Definition 1 *Let $I(P, D(P))$ be an instance of the discretized Art Gallery problem with polygon P as the gallery boundary and $D(P)$ a discretization of P. A solution $Z(x)$ of this instance is called* viable *if $Z(x)$ is a guard set for P, i.e.,*

$$\bigcup_{g \in Z(x)} Vis(g) = P.$$

Any exact method for the original AGP which solves its discretized version must address the fact that a solution to $I(P, D(P))$ might not necessarily be viable. As we will see, our algorithm overcomes this difficulty and always produces a viable solution by successively refining the given discretization whenever it detects that the present solution is not viable. Furthermore, the following theorem establishes that a solution obtained through this iterative process is also minimal.

Theorem 1 *Let Z be a solution of an instance $I(P, D(P))$ of the discretized Art Gallery problem. If Z is viable then Z is optimal.*

Proof. From the fact that Z is a solution of the minimization problem $I(P, D(P))$, it follows that 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 minimum 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 $z \geq z^*$. $\square$

Theorem 1 establishes that when the algorithm finds a solution for the discretized formulation which is viable, that solution is also a minimal vertex guard cover for P, i.e., it is a guard set for P.

We are now able to describe in detail the algorithm we first proposed in [1]. In the *Preprocessing Phase*, three procedures are executed: the first one computes the visibility polygons for the points in V, the second one computes the initial discretization $D(P)$ and the third one builds the corresponding IP model. In the *Solution Phase*, the discretized problem is successively solved and refined until a viable (and optimal) solution is found.

3.1 Preprocessing Phase

The main steps of the preprocessing phase are summarized in Algorithm 3.1.

In order to assemble the formulation outlined in Section 2, we start by building an initial discretization $D(P)$ of the polygon (step 1). In Section 4 we describe alternative discretization strategies and their impact on the efficiency of this algorithm.

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

Algorithm 3.1 Preprocessing Phase

1: $D(P) \leftarrow$ chosen initial discretization of P;
2: **for** each $j \in V$ **do**
3:    Compute $\text{Vis}(j)$;
4:    **for** each discretization point $p_i \in D(P)$ **do**
5:       $a_{ij} \leftarrow$ **Boolean**$(p_i \in \text{Vis}(j))$;
6:    **end for**
7: **end for**

The total complexity of step 3 is $O(n^2)$ [25] and, assuming that $m = |D(P)|$, the full complexity of step 5 is $O(nm \log n)$ since point location of each of the 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 5 whenever $m \in \Omega(n/\log n)$, and by that of step 3, otherwise.

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

3.2 Solution Phase

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

It remains to be argued that Algorithm 3.2 halts, as it will then follow from Theorem 1 that the algorithm is exact and the solution given is indeed a guard set for P. In order to

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 point c: $\sum_{j \in V} a_{rj} x_j \geq 1$ where $a_{rj} \leftarrow$ **Boolean**$(c \in \mathrm{Vis}(j))$, $\forall j \in V$;
7: **end for**
8: **until** Z is viable

determine the worst case for the number of iterations executed 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 an arrangement of line segments within P whose faces we call *atomic visibility polygons*, or AVPs (see Figure 1). It follows from the definition of the AVPs that if the centroid of (or, for that matter, any point in the interior of) an atomic visibility polygon $\mathcal{V}$ is visible from a vertex guard, the entire area of $\mathcal{V}$ must also be.

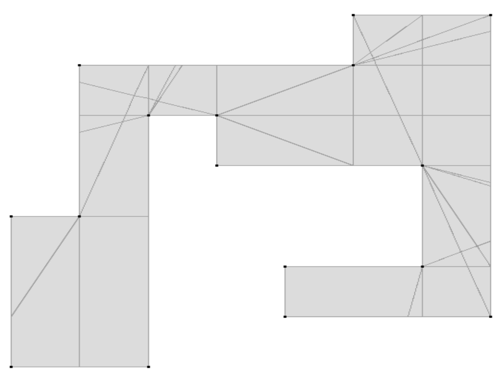

Figure 1: Visibility arrangement and AVPs.

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

Note that in step 3, any uncovered region (witness to the fact that Z is not viable) is necessarily a simple polygon formed by the union of neighboring AVPs. Therefore, an upper bound on the maximum number of iterations effected by the algorithm is $O(n^4)$ and this establishes the convergence.

Moreover, it can be derived from a result by Bose *et al.* [28] that $\Theta(n^3)$ is a tight bound on the number of AVPs, improving the above worst case result. However, in practice, this is still hugely over estimated and should be regarded solely as proof of convergence of the iterative method.

4 Discretization strategies

As presented in the previous section, the convergence of the algorithm follows from an upper bound on the number of uncovered regions. Yet, as each iteration solves an instance of an NP-hard problem (the set cover problem, SCP), the chosen discretization strategy must ideally be light enough to set up instances of SCP that can quickly be solved while minimizing the number of iterations required to attain an optimal solution.

Thus, there is a tradeoff between speed and precision that must be taken into account when designing a good discretization strategy. While the use of sophisticated geometric properties to build more efficient discretizations may reduce the number of iterations done by the algorithm, the corresponding cost in preprocessing might outweigh its benefits.

The following sections go into details on several alternatives for the discretization of the polygon and discusses the theoretical advantages and possible drawbacks of each one.

4.1 Single vertex

The simplest strategy one might consider is to start a discretization with a single vertex of the polygon P. At first glance, the *Single Vertex* strategy may seem weak since a single point in P conveys no geometric information and the solution of the discretized AGP associated to this simple discretization is expected to leave several uncovered regions, leading to a number of iterations of the algorithm.

However, the size of the SCP instances that the *Single Vertex* strategy generates is very small and they come without preprocessing cost. Therefore, it is still worth determining whether this strategy pays off.

4.2 All vertices

A reasonable approach to try to reduce the number of iterations from the *Single Vertex* strategy is to start with a larger discretization whose points are adequately distributed over P. However, to maintain the benefits of the previous strategy, the number of points in such discretization should be kept small and easy to compute. The *All Vertices* strategy is an attempt to reach this goal. We consider the still sparse case where the starting discretization contains all the vertices of the polygon (see Figure 2).

One can see that this strategy explores the fact that the vertices of the polygon should capture enough geometric information to prevent uncovered regions near the convex vertices from emerging.

Furthermore, experiments show that the *All Vertices* strategy generates smaller uncovered regions than the *Single Vertex* strategy and, in this case, more meaningful constraints get added, leading to better solutions in each iteration.

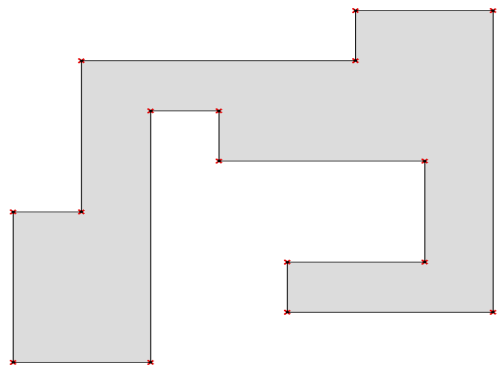

Figure 2: Example of the initial discretization used in the *All Vertices* strategy.

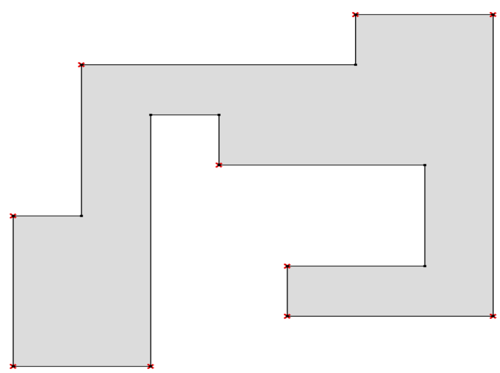

Figure 3: Example of the initial discretization used in the *Convex Vertices* strategy.

4.3 Convex vertices

Convex vertices are clearly more useful discretization points than reflex vertices since these are more easily guarded than those. Therefore, if any vertices might be redundant in gathering visibility information, it is natural to assume that the reflex ones are the most superfluous.

These observations lead us to consider the *Convex Vertices* strategy which starts with a discretization of P composed solely by the convex vertices (see Figure 3). In doing this, we further reduce discretization size, while still capturing much of the combinatorial visibility relationships at the price of a negligible increase in preprocessing cost.

While one might expect that this reduction could increase the number of iterations, we detected no such consequence. Besides, this strategy preserves the same nice features of the two previous alternatives, namely, an inexpensive preprocessing phase and SCP instances with a low number of constraints in each iteration.

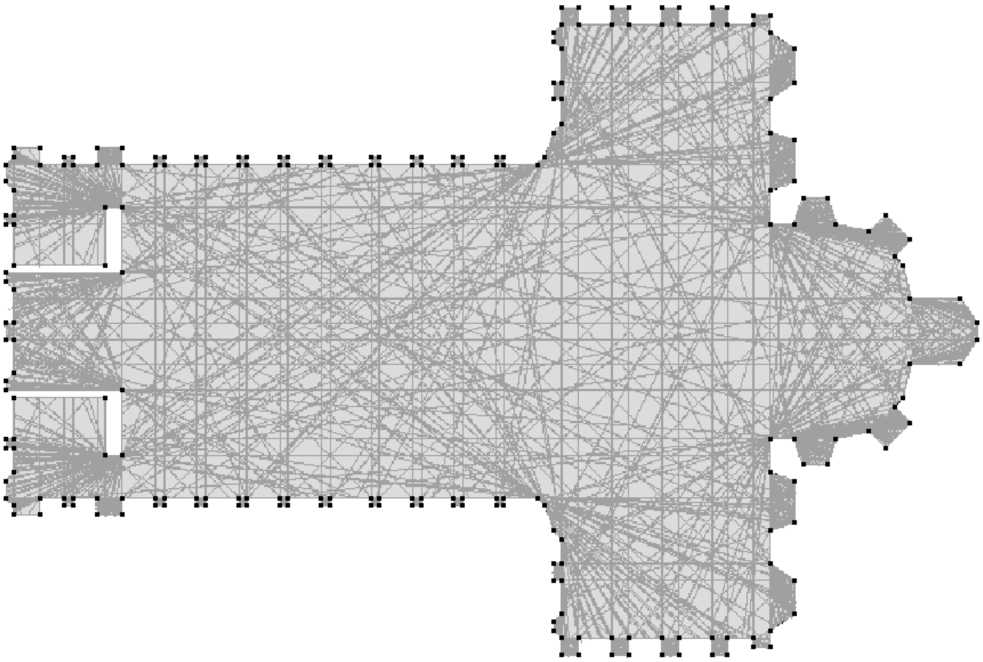

Figure 4: A sizeable gallery and its visibility arrangement showing all AVPs.

4.4 AVPs

The strategies seen so far seek to keep the number of constraints in the initial SCP instance small, while trying to reduce the number of iterations. However, it is possible to devise a strategy that can lower the number of iterations to one.

To this end, one has to identify a set of regions that, when covered, guarantee that the whole gallery is guarded. Furthermore, these regions must have the property that, once one of its points is visible from a vertex-guard, the entire region is watched by that guard. Once these regions are discerned, a discretization can be built by picking one point in each of them.

As shown in the previous section, the atomic visibility polygons (AVPs) of P fulfill both properties stated in the above paragraph. Therefore, the initial discretization containing the centroids of all AVPs leads to an SCP instance that, once solved, produces an optimal vertex-guard set for P. However, as there can be $O(n^3)$ AVPs, the number of constraints in the SCP model might also be $O(n^3)$.

Although this proves that we could solve the problem in a single iteration of the algorithm, building a discretization with the centroids of all AVPs of a complex gallery could result in a huge SCP instance (see Figure 4). Nonetheless, as shown next, not all AVPs need to be represented in the set of constraints in order to guarantee a single iteration, which makes the *all AVPs* discretization pointless.

Shadow AVPs.

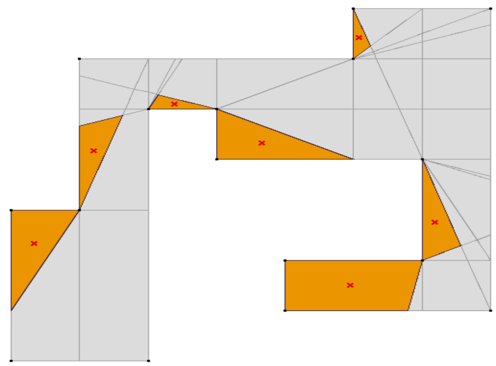

Figure 5: Example of the initial discretization used in the *Shadow AVPs* strategy.

As seen before, solving an SCP instance with the centroids of all AVPs is very costly. However, we can significantly reduced the number of discretized points and still guarantee that the algorithm finds the minimum number of guards necessary to cover P after the first iteration. In order to do so, we introduce the notion of a *shadow* AVP.

Initially we say that a line segment is a *visibility edge* for P if there exists a vertex $v \in P$ such that this segment is an edge of $\text{Vis}(v)$. Moreover, a visibility edge e originated from vertex v is said to be *proper* for v if and only if e is not contained in any edges of P.

Notice that since an AVP is a face in the arrangement generated by the visibility edges, the edges of an AVP are either portions of edges of P or portions of proper visibility edges of vertices of P.

We say that an AVP $\mathcal{S}$ is a *shadow AVP* if there exists a subset U of vertices of V such that $\mathcal{S}$ is not visible from any vertex in U and the only proper visibility edges that spawn $\mathcal{S}$ emanate from vertices in U.

The *Shadow AVPs* discretization strategy consists of taking the centroids of every shadow AVP (see Figure 5).

We now establish the fundamental relation between the optimal solutions of the discretized AGP with the *Shadow AVPs* strategy and those of the original AGP.

Theorem 2 *Let $I(P, D(P))$ be an instance of the discretized Art Gallery problem for polygon P where $D(P)$ is the set of centroids of the shadow AVPs of P. Then, G is a vertex-guard set for $D(P)$ if and only if G is a vertex-guard set for P.*

Proof. The necessity part is trivial since $D(P) \subset P$, therefore, we focus on the proof of sufficiency.

Suppose $G \subset V$ is a vertex-guard set for $D(P)$, but not for P. Thus, there exist regions of P that are not covered by any of the vertices of G. Let R be a maximal connected region not covered by G. Note that R is the union of (disjoint) AVPs.

To prove that at least one of those AVPs is of type shadow, notice that the entire region R is not seen by any vertex in G whose proper visibility edges spawn R. If R is an AVP, it is by definition a shadow AVP. Otherwise, there is a vertex $v_i \in V$ which has a proper

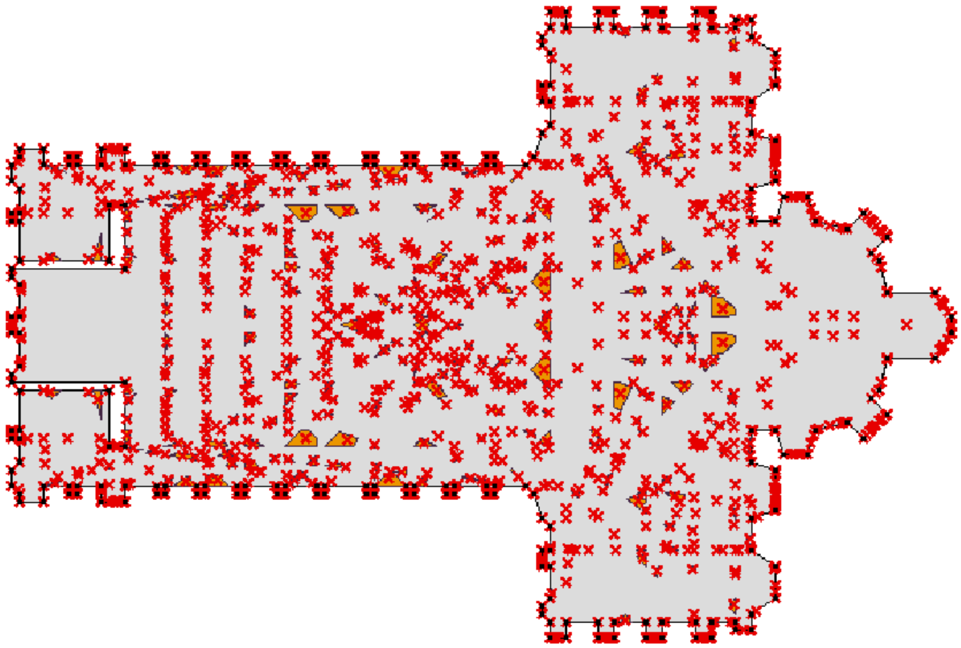

Figure 6: A sizeable gallery and the discretization points used in the *Shadow AVPs* strategy.

visibility edge e_{vi} that intersects and partitions R in two other regions. One of these regions matches the side of e_{vi} visible from v_i while the opposite one does not. Hence, through an inductive argument, by successively partitioning R, at least one shadow AVP is bound to be contained in R and therefore uncovered.

This contradicts the hypothesis since G is a guard set for $D(P)$ which is comprised of the centroids of all shadow AVPs. □

From Theorem 2, it is clear that with the *Shadow AVPs* strategy the algorithm converges in one iteration. Also, when we restrict ourselves to shadow AVPs, the size of the discretization decreases considerably when compared to the AVP strategy. Even for complex galleries (contrast Figures 4 and 6), this reduction may be large enough to render the algorithm practical.

Although the *Shadow AVPs* strategy requires only a single iteration of the algorithm to find an optimal solution, we will see later that the time spent in the preprocessing phase may become the bottleneck of the algorithm. This issue is investigated in Section 5.2.

4.5 Other Strategies

For completeness, it should be mentioned that other strategies have been considered in our prior works, such as those based on the regular [1] and on the induced [2] grid discretizations. Experiments have shown that none of them are competitive with the strategies discussed in

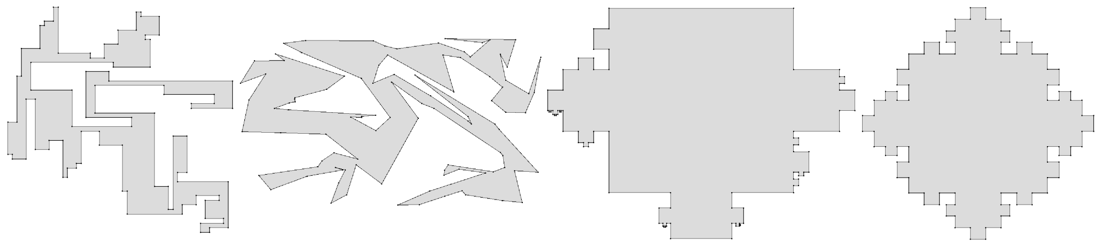

Figure 7: Sample polygons with 100 vertices: Random Orthogonal, Random Simple, Random von Koch and Complete von Koch.

this section, which are far more efficient.

5 Computational Experiments

We now discuss the experimental investigations that we carried out to evaluate the algorithm proposed in Section 3. In particular, we analyze the behavior of the algorithm with respect to the various discretization strategies discussed in the previous section.

All our programs were coded in `C++` and compiled with `GNU g++ 4.2`, on top of `CGAL 3.2.1` [29]. The visibility algorithm from [25] was implemented and `Xpress v18.10.04` [30] was used to compute the IP models corresponding to the SCP instances.

As for hardware, we used a desktop PC featuring a Pentium IV at 3.4 GHz and 3 GB of RAM running `GNU/Linux 2.6.24`. We observe that no other processes were allowed to execute in the machine during our tests. Besides, for each instance, the algorithm stopped either because an optimal solution was found or because the program ran out of memory.

5.1 Instances

To be considered a realistic method to solve the AGP, an algorithm must be able to handle a large variety of instances with distinctive characteristics. Thus, to test the robustness of our algorithm, we devised four classes of instances (see Figure 7). Each of them captures peculiar geometric properties that either appear in actual art galleries or represent extreme situations needed to exercise some of the algorithm's characteristics. The set of all instances used in the experiments reported here, plus several others, are downloadable at [19].

In the first two classes of instances, the art gallery is represented as an orthogonal or as a simple polygon, respectively. The former are thought to be good representatives of many actual art gallery buildings. The last two classes correspond to polygons assembled from a closed version of the von Koch fractal curve (see [31]). Instances generated in this way tend to have small protuberances on the boundary of the polygon which create tiny areas that are visible only by a small number of vertices. These instances are supposedly harder to solve than similarly sized instances of the two first classes. As an example, consider the two instances in Figure 8. Both polygons have 100 vertices, but the visibility arrangement of the Random Orthogonal instance has 2216 edges and 1085 AVPs (with only 99 Shadow

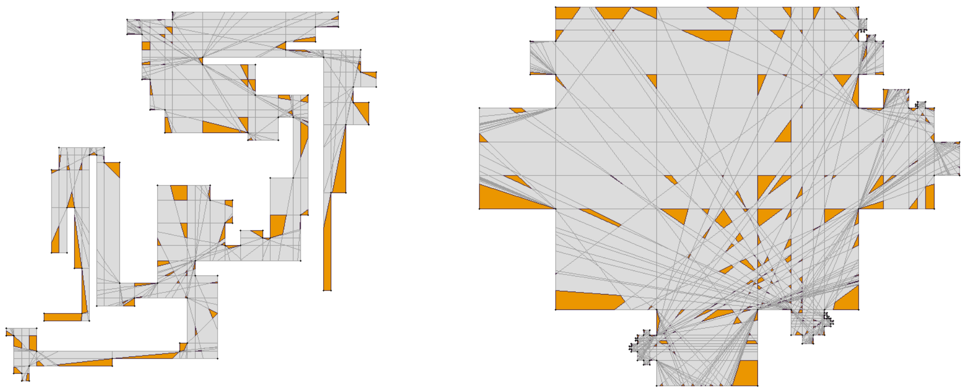

Figure 8: Instances of Random Orthogonal and Random von Koch polygons with the same size but distinct complexities.

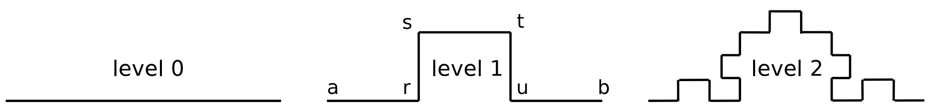

Figure 9: Levels of von Koch polygons.

AVPs) while the one corresponding to the Random von Koch polygon has 8794 edges and 4420 AVPs (with 264 Shadow AVPs).

More details on how to generate the test polygons of each class are given below.

(1) **Random Orthogonal:** A Random Orthogonal instance consists of a random n-vertex orthogonal polygon placed on an $\frac{n}{2} \times \frac{n}{2}$ unit square grid. The polygon is generated devoid of collinear edges in accordance to the method described in [32].

(2) **Random Simple:** Each Random Simple instance amounts to a random simple polygon generated by a special purpose procedure available in `CGAL` [29]. Essentially, this procedure starts by distributing the vertices of the polygon uniformly in a given rectangle and applies the method of elimination of self-intersections using 2-opt moves.

(3) **Complete von Koch (CvK):** Here, polygons are generated based on a modified version of the von Koch curve, which is a fractal with Hausdorff dimension of 1.34. An instance is created by starting with a square and recursively replacing each edge by five other edges as shown in Figure 9, where $\overline{ar} = \overline{st} = \overline{ub}$ and $\overline{sr} = \overline{tu} = \frac{3}{4}\overline{ar}$.

Let us make use of Figure 9 to introduce some notation needed to describe the last class of instances. We say that an edge of the current polygon which remains over the boundary of the initial square is at level 0. When the replacement operation, illustrated in the figure, is applied to an edge e at level k, the new edges that are not collinear with e are said to be at level $k + 1$.

(4) **Random von Koch (RvK):** An instance of this last class is constructed as follows. We start with a square and iteratively apply the replacement operation (from Figure 9) to some edges until the number of vertices of the polygon reaches an *a priori* fixed limit. At each iteration, we select an edge at random whose level is smaller than a given parameter λ

and randomly decide whether to replace it or not.

It is important to remark that for Random von Koch class, the instances of up to 1000 vertices were generated with λ set to 4. Beyond this size, since the number of vertices nears that of the Complete von Koch polygon of level 4 (2500), λ was set to 5. This is a likely explanation for the discontinuity observed around 1000 vertices in certain plots shown in Section 5.2 (see Figure 13) where results obtained by our algorithm for this class of polygons are displayed.

The random instances were generated for the number of vertices, n, in the ranges: $[20, 100]$ with step size 20; $(100, 1000]$ with step size 100 and $(1000, 2500]$ with step size 250. Similar sizes were chosen for the RvK class. Lastly, the CvK class contains, by construction, only 4 instances with 20, 100, 500 and 2500 vertices.

To endow our conclusions with statistical significance, we had to define the sample size, i.e, the number of instances generated for each value of n for the classes Random Orthogonal, Random Simple and RvK. To this end, we analyzed the variance of the results produced by our algorithm while we changed the sample size s. We observed that the variance remained practically unchanged for $s \geq 30$ and, therefore, we decided to generate 30 instances for each value of n.

Therefore, in total, our data set is composed of 1804 instances, having between 20 and 2500 vertices each. It is worth noting that our largest instances more than double the size of the largest ones whose optimal solutions are reported in the literature.

For completeness, we mention that other classes of instances were also considered in our prior works [1, 2], including the FAT polygons, introduced in [18]. These classes were not included in the research presented here because they are by far less challenging than the ones considered in the experiments reported in this paper. As an example, FAT polygons of any size always admit an analytical optimal solution consisting of only two guards.

Next section presents an extensive experimental analysis of the algorithm considering multiple discretization strategies.

5.2 Results

We now discuss the experimental evaluation of the various discretization strategies described in Section 4. All values reported in this section are average results for 30 instances of each size, or 30 runs of the same instance in the case of the CvK class, since for this class there is a single instance of each size.

Recall that the discretization size determines the number of constraints in the SCP instance solved in the second step of Algorithm 3.2. We start by analyzing the relationship between the discretization types and the time spent by the proposed algorithm. CvK polygons are particularly suited to this analysis because they illustrate two extreme situations. In strategies *Single Vertex*, *Convex Vertices* and *All Vertices* the initial discretizations correspond to very sparse grids whose sizes increase only by a small factor throughout the iterations. On the other hand, for *Shadow AVPs* a single iteration of the algorithm suffices, at the expense of building an extremely dense grid. Table 1 summarizes these results.

Notice that *Single Vertex*, *Convex Vertices* and *All Vertices* strategies indeed produce small discretizations whose sizes increase linearly in the number of vertices of P. As for

Table 1: Results for complete von Koch polygons.

	Final discretization size				Total Time (in secs)			
# of vertices	20	100	500	2500	20	100	500	2500
Single Vertex	9	82	326	1185	0.04	1.23	26.26	808.95
Convex Vertices	12	60	279	1403	0.04	0.84	21.09	720.99
All Vertices	20	115	552	2687	0.03	1.22	27.50	828.54
Shadow AVPs	20	244	5029	-	0.06	2.43	143.75	-

the *Shadow AVPs* strategy, the size of the discretization grows dramatically fast, to the point that we were not able to solve the largest CvK instance on account of running into memory limitations. Large grids inflate the number of constraints in the IP formulation, considerably increasing the time necessary to optimally solve the SCP instance. The *Convex Vertices* strategy is the one that spends less time, followed by *Single Vertex* and *All Vertices*. As it can be seen, the relative order among these strategies remains unchanged with respect to the sizes of the final discretizations.

On the other hand, Figure 10 shows the number of discretized points required by each strategy to achieve an optimal solution of the AGP for *Random Orthogonal*, *Random Simple* and *Random von Koch* polygons. One can see that the *Shadow AVPs* strategy follows the same pattern of the CvK case for RvK instances, with the discretization size growing quickly as the number of vertices of the polygons increases. Nevertheless, as we will see, the *Shadow AVPs* approach is well-suited for random polygons. The curves corresponding to the *All Vertices* strategy suggest that the set of vertices of the polygon is a good bid for the initial discretization since few new points are added to it to achieve an optimal solution of an AGP instance in all three classes under consideration. Notice that the same observation applies to the strategies *Single Vertex* and *Convex Vertices*. Hence, one can infer that small well-chosen proper subsets of V might suffice to capture a relevant part of the hardness of the problem. However, as we will soon see, strategies starting from minuscule discretizations, the extreme case being represented by *Single Vertex*, may cause the algorithm to iterate too much, increasing the computation time.

Figure 11 displays the number of iterations each strategy requires to reach an optimal solution for the three classes of random polygons. The lines corresponding to the *Shadow AVPs* strategy equate the constant function of value one and are included in the graphs solely as a reference. We recall that we successfully ran our program for the RvK class for instances of only up to 2250 vertices, due to memory limitations.

Consider now the *Single Vertex* strategy. For *Random Orthogonal* and *Random Simple* polygons, any single point of P only captures strictly local information about the shape of the polygon. Thus, by starting with a unitary discretization, several iterations should be expected before $D(P)$ is dense enough to capture the shape of P. This situation is very clearly depicted in the *Single Vertex* curves for *Random Orthogonal* and *Random Simple* classes. As for the RvK instances, the visibility polygon of most vertices corresponds to large portions of P, leading to many multiply covered areas within P. So, even when we

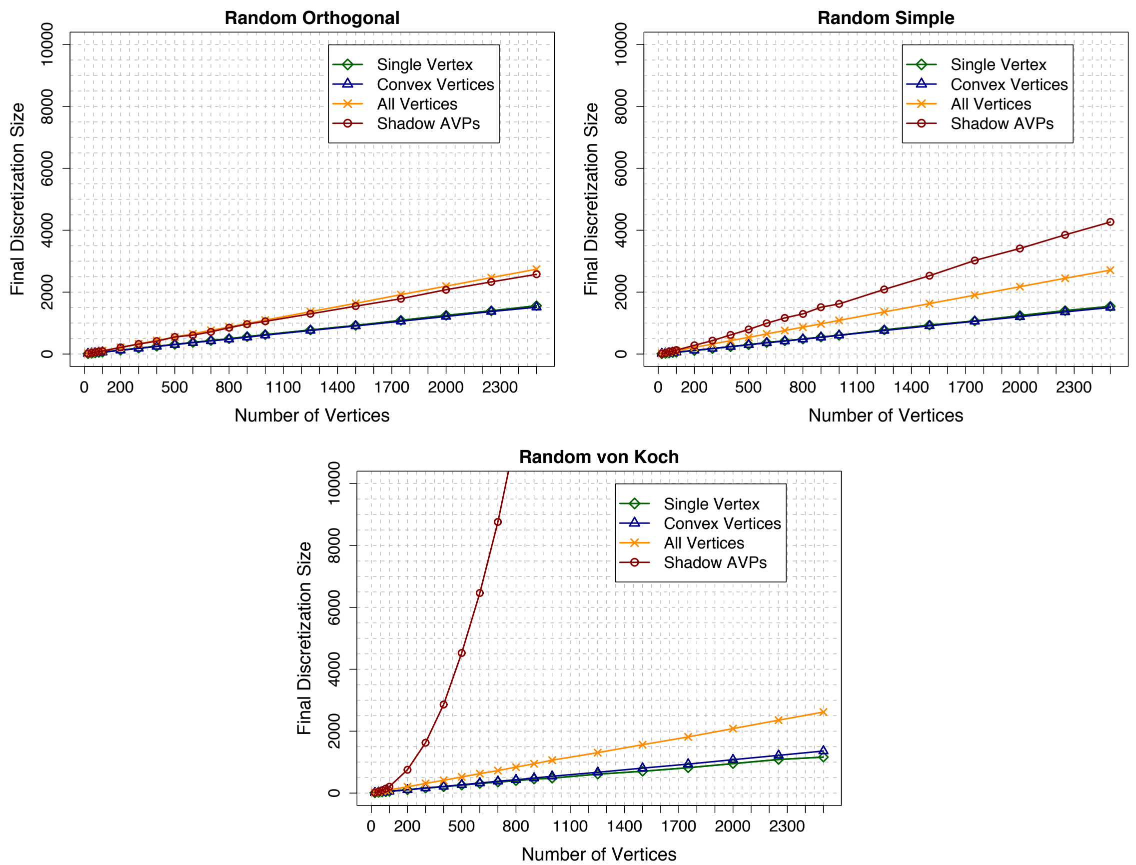

Figure 10: Final discretization size by polygon type.

start with a singleton, convergence happens much faster than with the other two classes.

Now, looking at the three non constant curves within each graph, we see that the number of iterations increases as the size of the *initial* discretizations decreases. In reference to the size of the input polygon, the number of iterations remains negligible when compared to the theoretical bound of $\Theta(n^3)$ (see Section 3.2). In regard to RvK polygons, the number of iterations grows a bit faster with the instance size but it still stays quite small. On the other hand, the proximity of the curves for *Convex Vertices* and *All Vertices* shows that the convex vertices alone seems to capture the shape P well enough to dispense with the reflex vertices. Therefore, the *Convex Vertices* strategy iterates just slightly more than the *All Vertices* strategy.

Figure 12 exhibits a box-plot chart with the number of iterations performed by the algorithm using the *Convex Vertices* strategy for the RvK class. For most instance sizes, one can see that the average and median values are very close. The difference between the upper and lower quartiles never exceeds 5 and, in all but the largest instances, no more than one outlier occurs. Even for the 2500-sized polygons, whose behavior seems unusual, the

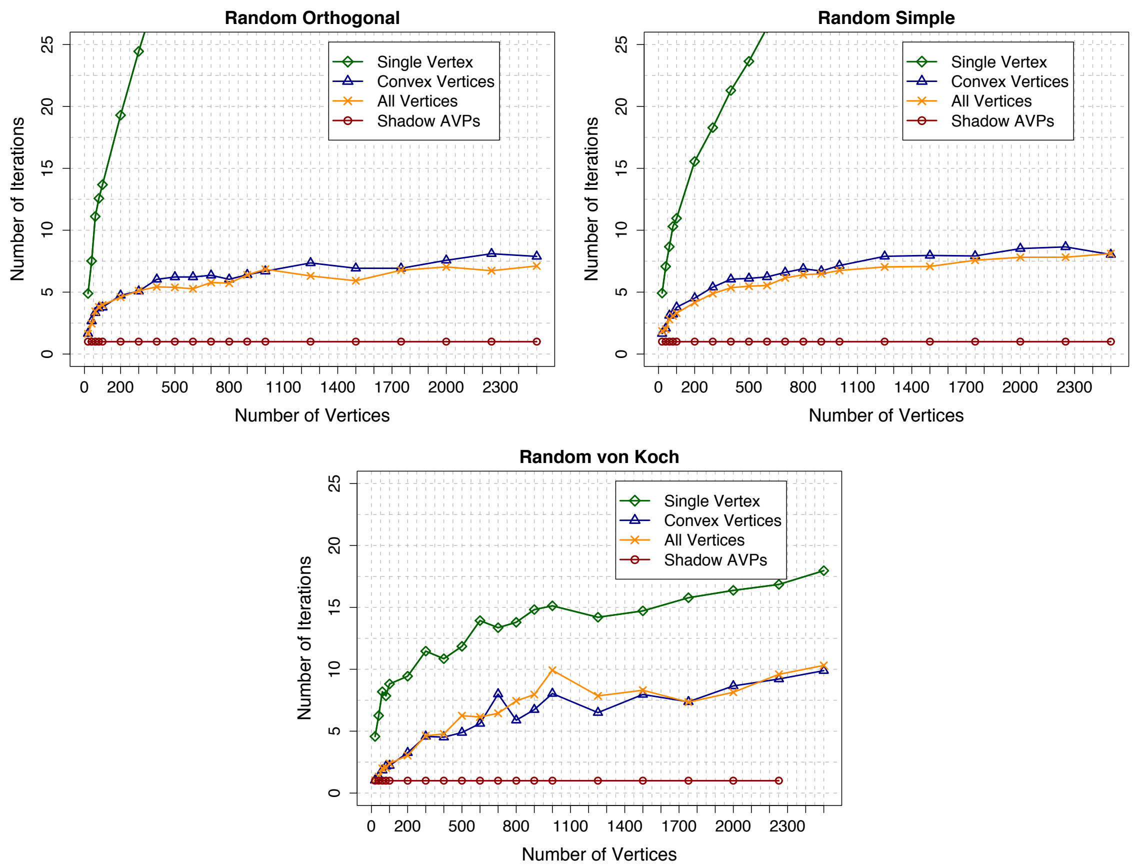

Figure 11: Number of iterations by polygon type.

total of 8 outliers out of the 30 instances is quite acceptable. This confirms the robustness of the proposed algorithm with this strategy.

Figure 13 shows the total amount of time, including preprocessing and processing phases, to solve instances from the three random classes of polygons. Notice that all charts are plotted on the same scale. For a proper analysis of this chart, one has to bear in mind our previous discussion on the number of iterations and the size of the discretizations produced by each of the alternative strategies.

Firstly, notice that the *Convex Vertices* and *All Vertices* strategies lead to very similar computation times. Earlier, we had seen that their iteration counts are small and very close together. We also observed that the size of the final discretizations grows slowly (almost linearly) with the instance size and that the one corresponding to the *All Vertices* strategy is just a bit larger than the one for the *Convex Vertices* strategy. These similarities are due to the fact that, in both cases, the IP solver has to compute a lighter SCP instance at each iteration of the algorithm. The algorithms emerging from these two strategies are not only fast but also very robust. To see that, notice that the curves for *All Vertices* and *Convex*

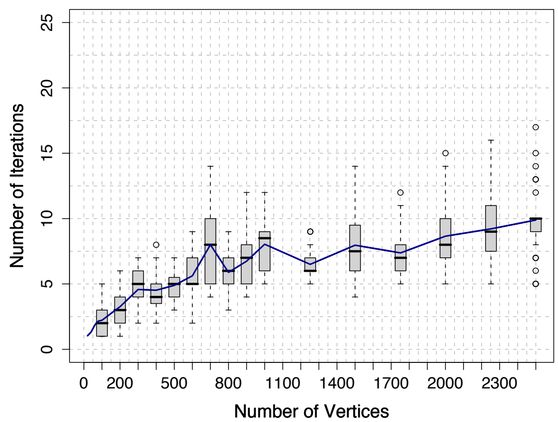

Figure 12: A box-plot graph showing the number of iterations for Random von Koch polygons using the *Convex Vertices* strategy.

Vertices strategies remain absolutely similar as the instance classes vary.

On the other hand, the *Single Vertex* strategy behaves poorly for the *Random Orthogonal* and *Random Simple* classes. Though it always yields the smallest discretizations on average, the number of iterations required by this strategy grows very rapidly. Even though one might also expect light SCP instances to be optimized at each iteration, the overhead of multiple calls to the IP solver surpasses the benefit of small instances. Only for the RvK polygons the *Single Vertex* strategy becomes competitive with the *Convex Vertices* and *All Vertices* ones which is predictable since, for these polygons, we see that the *Single Vertex* strategy usually executes just 10 iterations over the average of the two other strategies.

Now, we analyze the behavior of the algorithm under the *Shadow AVPs* strategy. This variant of the algorithm outperforms all the others for the *Random Orthogonal* and *Random Simple* classes, but it becomes excessively slow for RvK instances with just a few hundred vertices. To explain these results, we refer again to Figure 10.

In the RvK class, recall that the number of shadow AVPs grows rapidly with the instance size. Thus, although a single SCP instance is solved, the time spent in the computation of the constraint matrix associated to that instance is enormous. Let us briefly defer the discussion of this last issue.

In Figure 10 we have seen that, for the *Random Orthogonal* and *Random Simple* classes, the final discretization or, similarly, the number of shadow AVPs grows almost linearly with the instance size. Actually, it is not much larger than the size of the final discretizations yielded by the *Convex Vertices* and *All Vertices* strategies. However, we know that a single iteration of the algorithm is enough in this case. Therefore, the size of the unique SCP instance to be solved is not much larger than that of the one solved in the last iteration of the *All Vertices* and *Convex Vertices* strategies.

We now turn our attention to the time spent by the algorithm in each phase relative to the discretization strategies. Recall that the preprocessing phase is composed of three

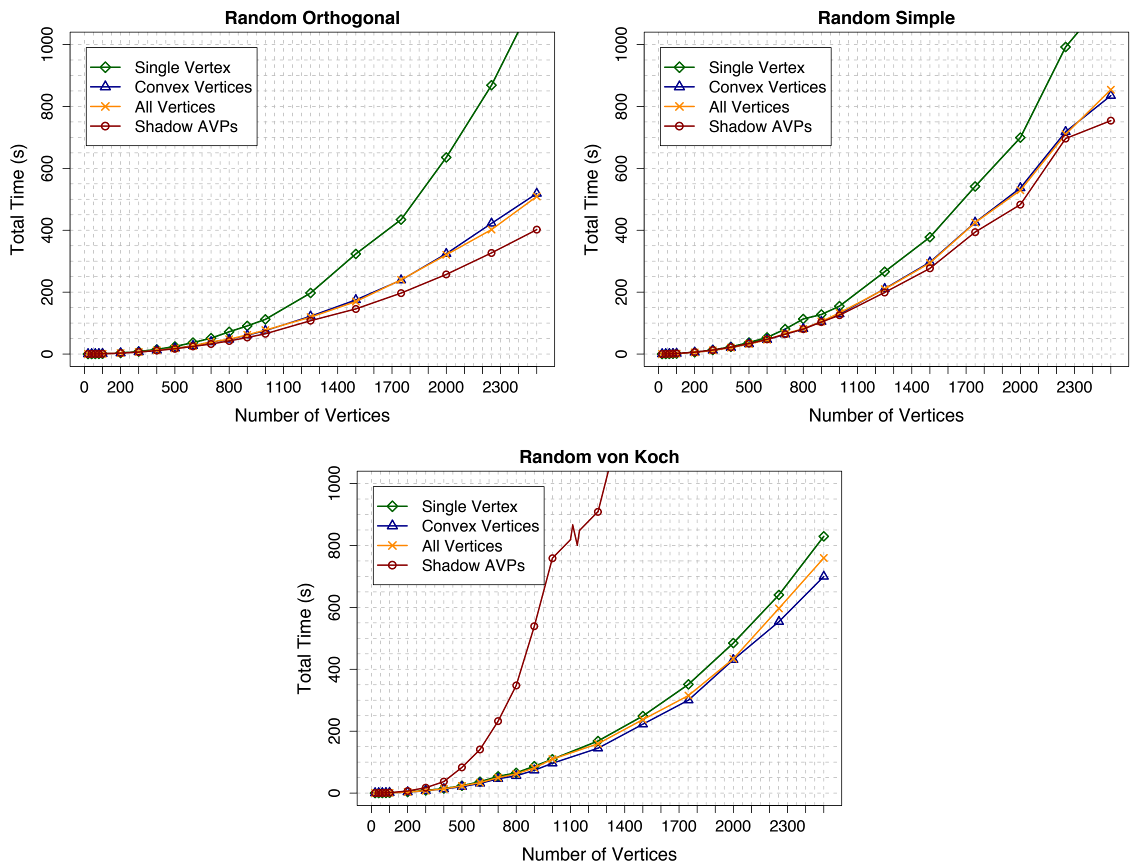

Figure 13: Total time by polygon type.

procedures. The first one is common to all strategies and computes the visibility polygons of each vertex. The second one computes the initial discretization and its cost is highly affected by the choice of the strategy to be implemented. The worst case corresponds to the *Shadow AVPs* strategy since it requires the computation of all AVPs and the determination of the shadow ones along with their centroids. On the other extreme, we have the *Single Vertex* and *All Vertices* strategies where no computation is needed for the second procedure while, for the *Convex Vertices* strategy, some inexpensive calculations are required to determine which vertices are convex. Finally, in the third procedure of the preprocessing phase one has to build the starting IP model and the time spent in doing so depends on the size of the discretization. This clearly benefits the *Single Vertex* strategy and also, though to a minor extent, the *Convex Vertices* and *All Vertices* strategies.

Figure 14 details the computation times of *Random Orthogonal* and *Random Simple* polygons on 2000 vertices and RvK polygons on 1000 vertices. Notice that the same scale is used on the three charts to facilitate comparisons, this being the reason why smaller instances in the RvK class were considered. The bars in these charts highlight the fraction of the total time spent on the processing and preprocessing phases and, for the latter, the fraction consumed by the procedure that computes visibility polygons.

One can see that the time spent in the preprocessing phase is in accordance with the discussion above, the *Shadow AVPs* strategy being the most time consuming for RvK polygons. For the *Random Orthogonal* and *Random Simple* classes, the preprocessing time for the *Shadow AVPs* strategy is about the same as those for the other two strategies and its advantage only shows in the processing phase. The first two charts are also illustrative of the fact that random simple polygons have more complex visibility structure than those in the orthogonal class.

What is somehow surprising is that, although we are solving NP-hard problems in the solution phase, in all cases the majority of the time expenditure takes place in the preprocessing phase, which is entirely polynomial. The extraordinary developments of IP solvers together with the fact the SCP instances arising from the AGP are among the easier ones explains this seeming counterintuitive behavior of the algorithm. Thus, a breakthrough in the performance of our algorithm would be attained if one could devise a discretization obtainable through a very fast procedure and, at the same time, satisfying the property that a single iterations is enough to reach the optimum of an AGP instance. Comparing the sizes of the final discretizations of the different strategies shown earlier, there seems to be room for such improvements.

6 Conclusions and Remarks

In this paper, we compiled and extended our research on exact approaches for solving the Art Gallery problem (AGP). In prior works [1, 2], we focused on galleries represented by orthogonal polygons and proposed an algorithm, also discussed in Section 3, based on successive discretizations of the input polygon. Our earlier computational experiments were constrained to instances of no more than a thousand vertices, while here, we extended the algorithm to handle non orthogonal polygons and tested it with instances of up to 2500 vertices. Moreover, we proposed new discretization strategies since the algorithm is very sensitive to the choice of discretizations. As a result, we introduced the *Convex Vertices* strategy which presents the best performance seen so far.

We recall that the exact algorithm relies not only on the discretization of the interior of the input polygon, but also on the modeling of this simplified discrete problem as a Set Cover problem (SCP). The resulting SCP instance is solved to optimality by an IP solver and, if uncovered regions remain, additional constraints are included and the process is repeated. Clearly, the performance of the algorithm depends also on the number of such iterations.

While focusing on novel strategies to implement the discretization step, a thorough experimentation was carried out to assess the trade-off between the number of iterations and the time spent by the many variants of the algorithm that arise from the alternative discretization methods.

Confirming the results from our earlier works, the proposed algorithm had excellent overall performance. It also proved to be robust, in the sense that it was able to tackle instances from a broad range of polygon classes. Moreover, the fastest variants of the

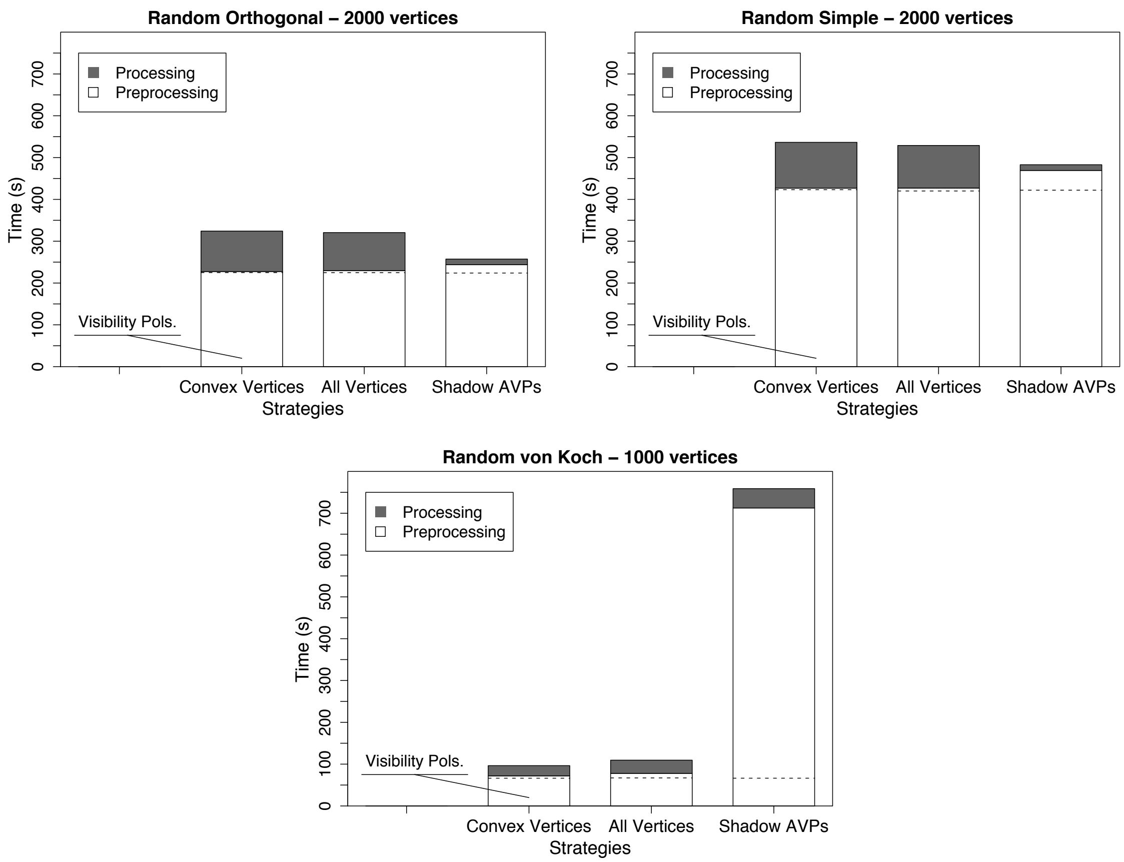

Figure 14: Break up of the execution time into processing and preprocessing, for polygons of the random classes.

algorithm very quickly found solutions to instances of more than 2000 vertices. This more than doubled the size of the largest instances we had previously solved which, in turn, were five times larger than those reported earlier in the literature.

The *Convex Vertices* strategy proposed in Section 4 yields sparse discretizations and, as a consequence, small SCP instances. As it can be seen in Table 2, this leads to a very fast implementation for instances of up to 2500 vertices.

On the other hand, as we observed in [2], the apparent advantage of a discretization which ensures an exact solution after a single iteration of the algorithm has not been verified. In particular, this occurred with the *Shadow AVPs* strategy whose inefficiency was due to the expensive preprocessing phase in which the shadow AVPs are computed. For these computations, we employed a polynomial time algorithm implemented with powerful data structures and efficient library packages for performing the necessary geometric operations. And yet, we could not significantly lower the preprocessing time of the exact algorithm under the *Shadow AVPs* strategy.

Table 2: Total Time (in seconds) for the *Convex Vertices* strategy.

	Polygons Classes			
n	Random Ortho	Random Simple	RvK	CvK
100	0.74	1.38	0.76	0.84
500	18.64	32.48	20.82	21.09
2500	518.60	834.78	699.74	720.99

Contrary to what was expected, in the case of *Shadow AVPs*, preprocessing remained more costly in time than the solution of the SCP instance, a well-known NP-hard problem. One could credit the extraordinary developments of IP solvers in recent years with the success of this algorithm. The advances in this field made possible the solution of large instances of SCP in very small amounts of time.

It remains an open question whether we can find yet another discretization leading to a single iteration of the algorithm, which is computable in time bounded by a very small degree polynomial on the number of vertices. This is a promising topic for future research which might be beneficial to our algorithm.

References

[1] Marcelo C. Couto, Cid C. de Souza, and Pedro J. de Rezende. An exact and efficient algorithm for the orthogonal art gallery problem. In *Proc. of the XX Brazilian Symp. on Comp. Graphics and Image Processing*, pages 87–94. IEEE Computer Society, 2007.

[2] Marcelo C. Couto, Cid C. de Souza, and Pedro J. de Rezende. Experimental evaluation of an exact algorithm for the orthogonal art gallery problem. In *WEA*, volume 5038 of *Lecture Notes in Computer Science*, pages 101–113. Springer, 2008.

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

[4] Vasek Chvátal. A combinatorial theorem in plane geometry. *Journal of Combinatorial Theory Series B*, 18:39–41, 1975.

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

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

[7] Jorge Urrutia. Art gallery and illumination problems. In Jörg-Rüdiger Sack and Jorge Urrutia, editors, *Handbook of Computational Geometry*, pages 973–1027. North-Holland, 2000.

[8] S. K. Ghosh. *Visibility Algorithms in the Plane.* Cambridge University Press, New York, 2007.

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

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

[11] Dietmar Schuchardt and Hans-Dietrich Hecker. Two NP-hard art-gallery problems for ortho-polygons. *Mathematical Logic Quarterly*, 41:261–267, 1995.

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

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

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

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

[16] Y. Amit, J. S. B. Mitchell, and E. Packer. Locating guards for visibility coverage of polygons. In *Proc. Workshop on Algorithm Eng. and Experiments*, pages 1–15, 2007.

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

[18] Ana Paula Tomás, António Leslie Bajuelos, and Fábio Marques. On visibility problems in the plane - solving minimum vertex guard problems by successive approximations. In *Proc. of the 9th Int. Symp. on Artificial Intelligence and Mathematics*, 2006.

[19] Marcelo C. Couto, Cid C. de Souza, and Pedro J. de Rezende. Instances for the Art Gallery Problem, 2009. *www.ic.unicamp.br/∼cid/Problem-instances/Art-Gallery*.

[20] D. S. Johnson. A theoretician's guide to the experimental analysis of algorithms. In M. H. Goldwasser et al., editor, *Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implem. Challenges*, pages 215–250. AMS, Providence, 2002.

[21] C. C. McGeoch and B. M. E. Moret. How to present a paper on experimental work with algorithms. *SIGACT News*, 30, 1999.

[22] Peter Sanders. *Presenting data from experiments in algorithmics*, pages 181–196. Springer-Verlag New York, Inc., New York, NY, USA, 2002.

[23] B. Moret. Towards a discipline of experimental algorithmics. In *Proc. 5th DIMACS Challenge*.

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

[25] 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.

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

[27] Laurence A Wolsey. *Integer Programming*. Wiley-Interscience, 1998.

[28] Prosenjit Bose, Anna Lubiw, and J. Ian Munro. Efficient visibility queries in simple polygons. *Computational Geometry*, 23(3):313–335, 2002.

[29] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.

[30] Xpress–optimizer. www.dashoptimization.com.

[31] Kenneth Falconer. *Fractal Geometry, Mathematical Foundations and Applications*. John Wiley & Sons, 1990. pp. 120–121.

[32] Ana Paula Tomás and António Leslie Bajuelos. Generating random orthogonal polygons. In *Current Topics in Artificial Intelligence*, volume 3040 of *LNCS*, pages 364–373. Springer, 2004.