Solving the Art Gallery Problem: A Practical and Robust Method for Optimal Point Guard Positioning

Davi Colli Tozoni
Institute of Computing
University of Campinas (Unicamp)
Campinas, SP, Brazil
Email: davi.tozoni@gmail.com

***Abstract*—This Master's thesis focused on studying and developing techniques for optimally solving the Art Gallery Problem (AGP), one of the most investigated problems in Computational Geometry. The AGP, which is a known $\mathbb{NP}$-hard problem, consists in finding the minimum number of guards sufficient to ensure the visibility coverage of an art gallery represented as a polygon.**

We studied how to apply Computational Geometry concepts and algorithms as well as Integer Programming techniques in order to solve the AGP to optimality. This work culminated in the creation of a new algorithm for the AGP, whose idea is to iteratively generate upper and lower bounds for the problem through the resolution of discretized versions of the AGP.

The algorithm was implemented and tested on more than 2800 instances of different sizes and classes of polygons. The technique was able to solve in minutes more than 90% of all instances considered, including polygons with thousands of vertices, greatly increasing the set of instances for which exact solutions are known. To the best of our knowledge, in spite of the extensive study of the AGP in the last four decades, no other algorithm has shown the ability to solve the AGP as effectively as the one described here. For illustration, in a direct comparison with the algorithm by Kröller et al., considered one of the most promising techniques for the AGP, our method solved almost 32% more instances than its competitor.

In addition, we provide a free version of our code and of our benchmark for download, which is unprecedented in the literature.

***Index Terms*—Art Gallery Problem, Exact Algorithm, Computational Geometry, Combinatorial Optimization, Visibility.**

I. Introduction

During this Master's Degree at the Institute of Computing (Unicamp), a new method was developed for optimally solving the Art Gallery Problem (AGP), one of the most famous and studied problems in Computational Geometry. The original version of the Art Gallery Problem, also known as the AGP with Point Guards, was proposed in 1973 by Klee and consists in answering the following question: how many guards are sufficient to ensure that an art gallery (represented by a polygon) is fully guarded, assuming that a guard's field of view covers 360 degrees as well as an unbounded distance limited only by the walls of the gallery? As an example of an AGP instance, consider the floor plan of the second level of the Brazilian National Museum of natural history and anthropology, in Rio de Janeiro, displayed in Fig. 1[1]. In this example, we see that 25 guards are sufficient to watch over the entire area.

[1]Obtained from `www.museunacional.ufrj.br`.

The main motivation in working with the AGP lies in its interdisciplinarity and its complexity. The AGP can be defined as a Computational Geometry problem, but it is at the same time an Optimization problem. In addition, despite being a theoretical problem, the endless possibilities of relaxing and restricting the initial AGP constraints allow us to create a number of variations, some of which have important practical applications. For example, by limiting the visibility range of the guards and forcing connectivity between them, we have a problem similar to what is found when placing nodes in a sensor network.

Another interesting fact in working with the AGP is the challenge in pursuing exact solutions. Until the first decade of 2000, the main achievements in the study of the AGP were on the theoretical side of the problem. As early as 1975, Chvátal shown that $\lfloor n/3 \rfloor$ guards are always sufficient and sometimes necessary for covering a simple polygon of n vertices [1]. Roughly a decade later, Lee and Lin [2] proved the $\mathbb{NP}$-hardness of the *Art Gallery Problem with Vertex Guards* (AGPV), which is the variant where the guards are restricted to the vertices of the polygon. The case of point guards (the original AGP) is also known to be $\mathbb{NP}$-hard, as proved by Aggarwal et al. in 1988 [3].

On the other hand, before this thesis, despite several decades of extensive investigation on the AGP, including contributions from renowned researchers as O'Rourke, Mitchell, Urrutia, Ghosh and Fekete, all previously published algorithms for the AGP with Point Guards were unable to handle instances with hundreds of vertices and often failed to prove optimality for a significant fraction of the instances tested. As a matter of fact, some experts have claimed at that time that "practical algorithms to find optimal solutions or almost-optimal bounds are not known" for this problem [4].

In this context, our objective was to find a practical and robust method for the AGP that would make it possible to solve complex instances in a reasonable amount of time. To achieve this goal, we decided to approach the problem with Integer Linear Programming (ILP) techniques and to employ the exact method for the AGP with Vertex Guards presented in [5] as a tool. In their work, Couto et al. solved instances of the AGPV with thousands of vertices.

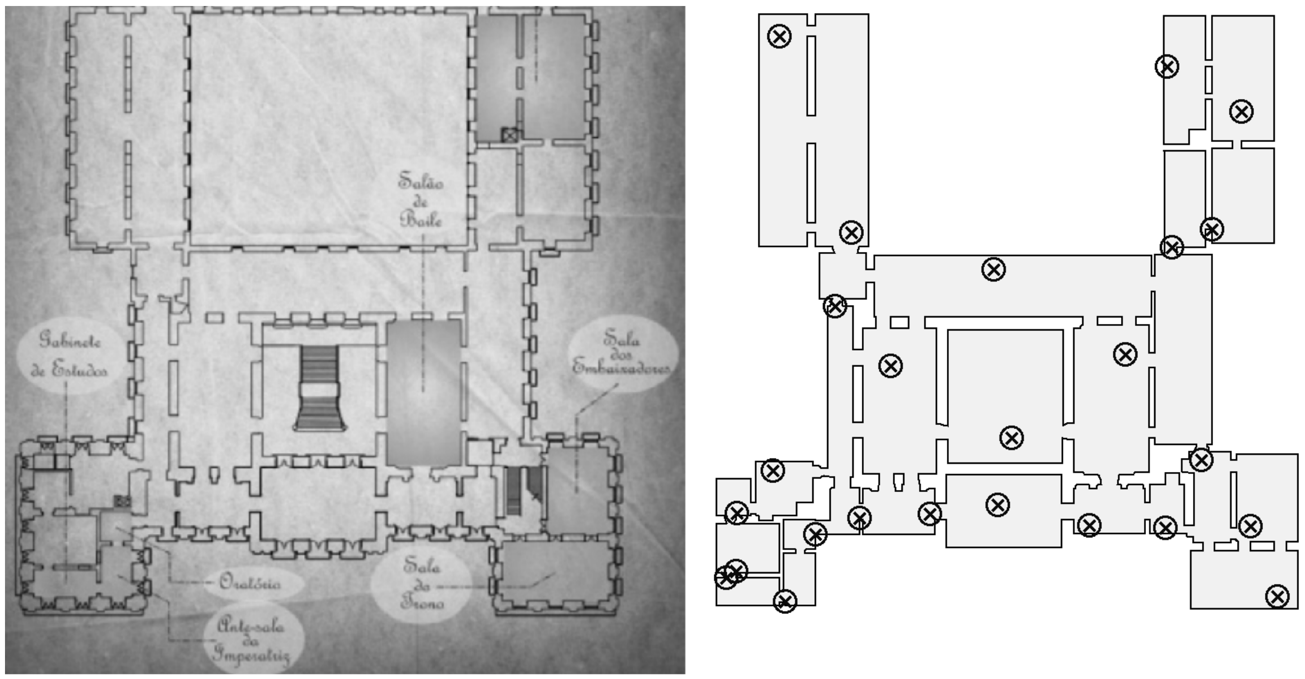


Fig. 1. National Museum floor plan (left); an optimal guard positioning (right).

A. *Our contribution*

In the thesis, we detailed a new robust method for solving the Art Gallery Problem with Point Guards. The new algorithm, which was presented in papers [6], [7], iteratively solves discretized versions of the AGP making use of ILP solvers to quickly obtain new lower and upper bounds for the problem, until an optimal solution is found or a timeout is reached.

The technique was tested in different occasions and, as seen in Section IV, it led to good results, being considered today the state-of-the-art technique for optimally solving the AGP. In total, 2880 publicly available instances with sizes reaching 5000 vertices were tested and the method achieved an optimality rate of more than 90%, which means a significant improvement over all previously published techniques.

Furthermore, due to the success of our implementation, we have also recently released a free version of our source code in the project's website [8]. This action is a milestone in the quest for practical solvers for the AGP, since, to our knowledge, it is the first time that an implementation with verified efficiency is made publicly available. Hopefully, this will be a catalyst in the search for new techniques for the problem and an incentive for other experimentation projects.

B. *Text Organization*

This text is organized as follows. The concepts, definitions and theorems necessary to understand the algorithm to be described are presented in the next section. In Section III, the technique we developed is explained. Section IV focuses on how the algorithm was implemented, on the environment and instances used for testing and also on the most significant results obtained, including a comparison with other state-of-the-art techniques. Finally, some conclusions are presented in Section V.

II. Preliminaries

Before digging into the algorithm and its particulars, it is necessary to fully understand some important concepts, being they in the computational geometry field or part of combinatorial optimization. In the next sections, the background related to the Art Gallery Problem and our solver is explained.

A. *Computational Geometry*

The objective of the AGP is to watch over an art gallery, which may be represented as a simple polygon or as a polygon with holes. A *simple polygon* consists of straight, non-intersecting line segments that are joined pair-wise to form a closed path. The set of vertices V contains all points where consecutive segments are joined. Figures 2 and 3 display examples of simple and non-simple polygons, respectively.

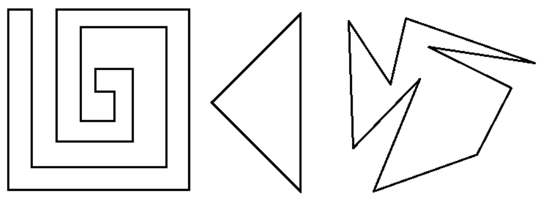


Fig. 2. Simple polygons.

On the other hand, a *polygon with holes* P_h can be described using a simple polygon P_b as the outer boundary and a set of m disjoint simple polygons $H_1, H_2, ..., H_m$ totally contained inside P_b as the holes. In this case, P_h consists in the set $P_b -$

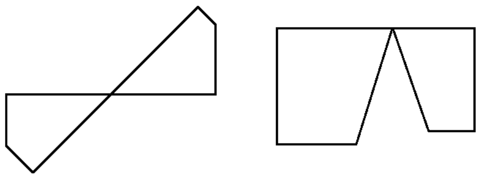


Fig. 3. Non-simple polygons.

$\bigcup_{i=1}^{m} H_i$. Two examples of polygons with holes are displayed in Fig. 4.

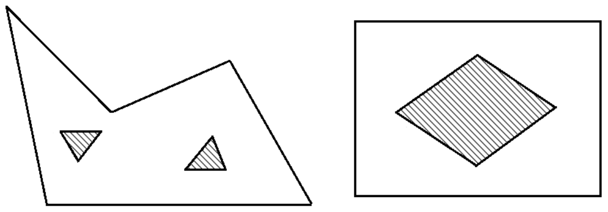


Fig. 4. Polygons with holes.

A vertex in a polygon can also receive a special name depending on the angle between its two incident edges with the interior of the polygon. If this angle is less than 180°, than the vertex is called *convex*. Otherwise, it is a *reflex* vertex. Fig. 5 presents some examples. Note that, in the case of a hole H_i, the convex vertices of H_i are actually reflex in relation to the entire polygon P_h. Obviously, the reverse is also true for reflex vertices.

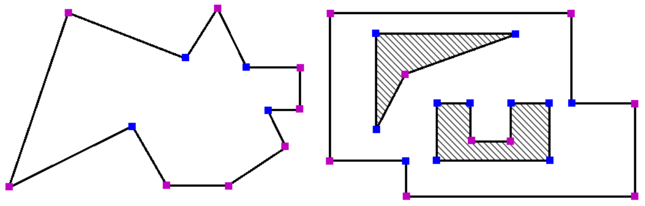


Fig. 5. Convex (purple) and reflex (blue) vertices.

In the AGP, we say that a position is *surveilled* (*watched*, *guarded* or *covered*) by a guard g if this position is visible to g. The concept of visibility can be described as follows: two points in a polygon P are *visible* to each other if the line segment that joins them does not intersect the exterior of P. Thereafter, the *visibility polygon* of a point $p \in P$, denoted by $\text{Vis}(p)$, is the set of all points in P that are visible from p. The edges of $\text{Vis}(p)$ are called *visibility edges* and one visibility edge is said to be *proper* for p if it is not contained in any of the edges of P. Fig. 6 illustrates the concept of visibility.

After defining visibility, the next step is to define the coverage of a given region. Given a finite set S of points in P, a *covered* (respectively, *uncovered*) region induced by S in P is a maximal connected region in $\bigcup_{p\in S} \text{Vis}(p)$ (respectively, $P - \bigcup_{p\in S} \text{Vis}(p)$). Knowing this, we say that S *fully covers* the polygon P if the covered region induced by S in P equals

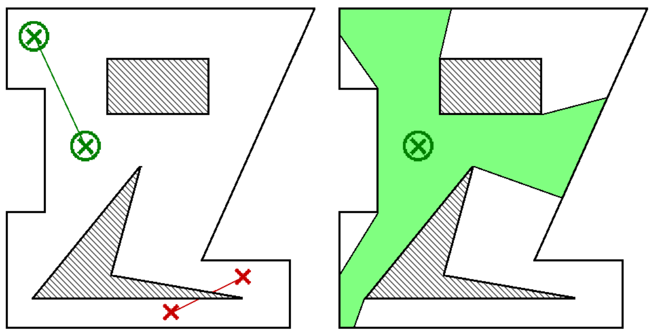


Fig. 6. Two points visible (green) and two other that are not visible (red) to each other (left); the visibility polygon of a point (right).

P. In addition, we can also define $C_{\mathcal{U}}(S)$ as a set containing exactly one interior point of each uncovered region induced by S.

Moreover, the geometric arrangement defined by the visibility edges of the points in S, denoted $\text{Arr}(S)$, partitions P into a collection of closed polygonal faces called *Atomic Visibility Polygons* or simply *AVP*s. Clearly, the edges of an AVP are either portions of edges of P or portions of proper visibility edges of points of S. Denote by $C_f \subset S$ the set of points in S that cover an AVP f. Define a partial order $\prec$ on the faces of $\text{Arr}(S)$ as follows. Given two AVPs f and f', we say that $f \prec f'$ if and only if $C_f \subset C_{f'}$. We call f a *shadow AVP* (*light AVP*) if f is minimal (maximal) with respect to $\prec$. Applying these concepts, it is possible to define $C_{\mathcal{L}}(S)$ as a set containing exactly one point within each light face of $\text{Arr}(S)$ and $V_{\mathcal{L}}(S)$ as the set of all vertices of light AVPs. Figures 7 and 8 illustrate these concepts.

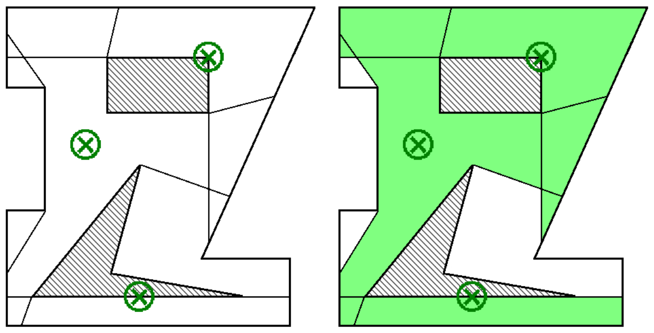


Fig. 7. The arrangement induced by a finite set S of points (left); the set S and its covered (light green) and uncovered (white) regions (right).

In our work, the complexity of the resulting arrangement is of great importance for the solver's performance. In [9], Bose et al. showed that, for the case where the set of vertices V induces the arrangement in a simple polygon, the arrangement complexity is $\Theta(|P|^3)$. This result can be easily adapted to prove that, for a generic set S in a hole-free polygon P, the complexity of constructing $\text{Arr}(S)$, as well as the number of AVPs in the arrangement, is $\Theta(|S|^2 \cdot |P|)$. In the general situation, where P can have holes, we did not find

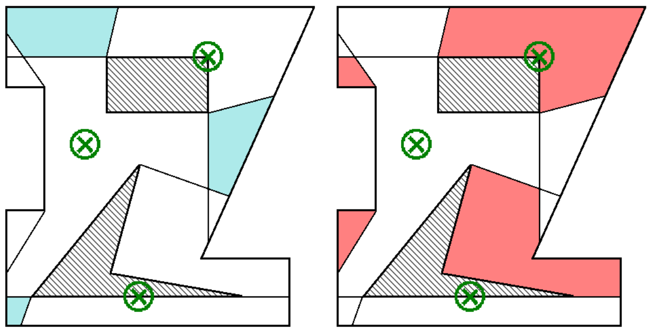


Fig. 8. The arrangement induced by S with the *light* (blue) (left); and *shadow* (red) AVPs (right).

any complexity results in literature. However, as we have $O(|S| \cdot |P|)$ visibility edges in the polygon, in the worst case, considering that all of them intersect, we will have a final complexity of $O(|S|^2 \cdot |P|^2)$.

B. Integer Programming

In 1987, Ghosh presented an approximation technique [10] for the AGP with Vertex Guards in which a reduction of the AGPV to the Set Cover Problem (SCP) is employed. A similar reduction is also used in the method developed in this Master's project.

The SCP is one of the most famous combinatorial problems. Given a set of elements U, called Universe, and a set A containing subsets of U, the SCP asks for the minimum number of sets from A whose union equals U. As proved by Karp in 1972, the Set Cover Problem is $\mathbb{NP}$-complete [11], which means that obtaining an algorithm with polynomial worst-case complexity is not possible, unless $\mathbb{P} = \mathbb{NP}$. Nevertheless, in practice, a good option for solving an SCP instance is to model the problem as an Integer Linear Program (ILP), even though the cost of solving a general ILP is theoretically exponential. Today, several ILP solvers are capable of finding optimal solutions for large SCP instances, with thousands of variables and constraints, in just a few seconds or minutes. Below the classic ILP model for the SCP is given.

$$\begin{aligned} \min \; & \sum_{s \in A} x_s \\ \text{s.t.} \; & \sum_{\substack{s \in A \\ e \in s}} x_s \geq 1, \; \forall e \in U \\ & x_s \in \{0,1\}, \; \forall s \in A \end{aligned}$$

In the model, the variable x_s has value 1 if the set s is chosen to be part of the resulting collection of subsets and 0 otherwise. The objective is to minimize the sum of variables x_s, which actually means to minimize the number of sets selected from A. Finally, the set of constraints presented in the model ensures that, for every element $e \in U$, at least one of the subsets containing e is chosen, which gives rise to a viable solution.

Although ILP solvers are normally a good choice for solving SCP instances, there are cases where their use may not be so efficient. In these situations, a technique that can take advantage of particular characteristics of the problem can behave better and be used to improve the ILP solver's performance. In our AGP solver, we implemented some techniques with this purpose. One well tested method to find good viable solutions for SCP instances and that was implemented in this project is a Lagrangian Heuristic.

Apart from the techniques employed to find viable solutions for the SCP, others can be used to simplify the problem, before an actual solver is used. For example, after a problem is reduced to an ILP instance, it is possible to search for redundant variables or constraints and remove them from the original matrix. This type of operation can normally be done quickly and may greatly minimize the size of the initial instance, probably improving the performance of the ILP solver.

Details about the Lagrangian Heuristic and the matrix reduction methods that were implemented in our solver can be seen in the full text of the thesis [12].

C. The Art Gallery Problem

After discussing important geometric and combinatorial concepts, it is now possible to discuss the AGP in a more formal way.

In a geometric setting, the AGP can be restated as the problem of determining a smallest set of points $G \subset P$ such that $\bigcup_{g \in G} \text{Vis}(g)$ equals P. This leads to a reduction from the AGP to the SCP, in which the points in P are the elements to be covered (set U) and the visibility polygons of the points in P are the sets used for covering (which compose the collection A). Accordingly, we can use this reduction to construct an ILP formulation for the AGP:

$$\begin{aligned} \min \; & \sum_{c \in P} x_c \\ \text{s.t.} \; & \sum_{\substack{c \in P \\ w \in \text{Vis}(c)}} x_c \geq 1, \; \forall w \in P \\ & x_c \in \{0,1\}, \; \forall c \in P \end{aligned}$$

However, for non-trivial cases, this formulation has an infinite number of constraints and an infinite number of variables, rendering it of no practical benefit. A natural idea that arises is to make at least one of these quantities finite. By fixing only the guard candidates to be a finite set, we obtain the so-called *Art Gallery Problem With Fixed Guard Candidates* (AGPFC). On the other hand, by restricting solely the points that need to be covered (here called witness set) to a finite set, we end up with the special AGP variant known as the *Art Gallery Problem With Witnesses* (AGPW). In principle, in the first case, we are still left with an infinite number of constraints while, in the second case, we still have an infinite number of variables. However, in order to have a tractable SCP instance, one should have both the witness and the guard candidate sets of finite size. The AGP variant that fulfills this property is named

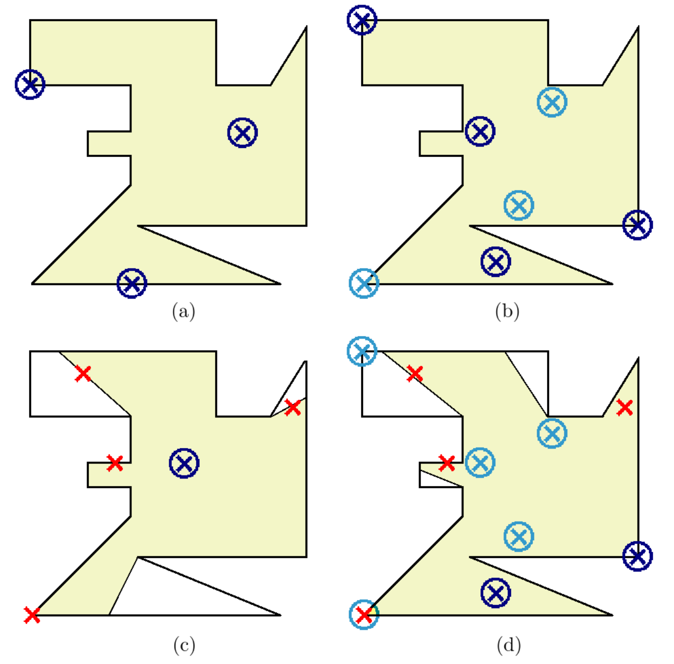


Fig. 9. An illustration of the four different variants of the Art Gallery Problem: (a) AGP; (b) AGPFC; (c) AGPW; (d) AGPWFC.

the *Art Gallery Problem with Witnesses and Fixed Guard Candidates* (AGPWFC). Examples of these three versions of the AGP are shown in Fig. 9. Therein, the witnesses and the guard candidates are identified by the symbols "$\times$" and "$\otimes$", respectively. Darker guard candidates refer to guards present in an optimal solution of the corresponding problem and, when appropriate, have their visibility polygons also depicted.

To assist in the description of the algorithm in the next section, we introduce here some other useful notations. Let D and C be finite witness and guard candidate sets, respectively. We denote the AGPW, AGPFC and AGPWFC problems defined for the sets C and D by AGPW(D), AGPFC(C) and AGPWFC(D, C), respectively. The SCP model associated to AGPWFC(D, C) is then

$$\begin{aligned}
\min \quad & \sum_{c \in C} x_c, \\
\text{s.t.} \quad & \sum_{\substack{c \in C \\ w \in \text{Vis}(c)}} x_c \geq 1, \ \forall w \in D, \\
& x_c \in \{0, 1\}, \ \forall c \in C.
\end{aligned}$$

D. Basic Theorems

To close this section, we briefly introduce the theorems that form the basis of our algorithmic solution, which will be fully explained in Section III. The following theoretical results allow us to apply reductions of the AGPFC and the AGPW to the AGPWFC (SCP) and also guarantee its usage to find correct bounds for the original AGP. It is noteworthy that Theorems 2, 3 and 4 are actually adaptations of results presented in the work of Couto et al. [5], where the specific problem called AGPV (AGPFC(V)) was treated.

Theorem 1. *Let D be a finite subset of points in P. Then, there exists an optimal solution for AGPW(D) in which every guard belongs to a light AVP of* Arr(D).

Proof: Let G be an optimal (cardinality-wise) set of guards that covers all points in D. Suppose there is a guard g in G whose containing face f in Arr(D) is not a light AVP. This means that f is not maximal with respect to the order relation $\prec$ (see Section II-A). In other words, there exists a face f' of Arr(D) that shares an edge with f such that $f \prec f'$, i.e., a point in f' sees more points of D than one in f does. An inductive argument suffices to show that this process eventually reaches a light AVP (maximal w.r.t. $\prec$) wherein lies a point that sees at least as much of D as g does, i.e., g may be replaced by a guard that lies on a light AVP. The Theorem then follows, by induction, on the number of guards of G. ■

Theorem 2. *Let C be a finite subset of points in P. Consider the set D composed of a point of each AVP of* Arr(C). *Then, $G \subseteq C$ guards D if and only if G is a guard set for P.*

Proof: The necessity part is trivial since $D \subset P$, therefore, we focus on the proof of sufficiency. By the construction process of Arr(C), all interior points of a given AVP A_i are visible to the same set $S_i \in C$. Otherwise, there would be an edge of Arr(C) separating two different points in A_i, which is obviously not possible. Consequently, if a set G guards one interior point of A_i, G directly covers the entire AVP. Thus, since the union of all faces of Arr(C) equals P, G can watch over the whole polygon by simply covering one interior point within each AVP. ■

Theorem 3. *Let C be a finite subset of points in P. Consider the set D composed of a point of each shadow AVP of* Arr(C). *Then, $G \subseteq C$ guards D if and only if G is a guard set for P.*

Proof: The necessity part is trivial since $D \subset P$, therefore, we focus on the proof of sufficiency. Suppose $G \subset C$ guards D, but not P. Thus, there exist regions of P that are not covered by any of the points 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 point in G whose proper visibility edges spawn R. If R is an AVP, it is by definition a shadow AVP. Otherwise, there is a candidate $c_i \in C$ which has a proper visibility edge e_{ci} that intersects and partitions R in two other regions. One of these regions matches the side of e_{ci} visible from c_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 guards D, which is comprised of interior points of all shadow AVPs. ■

Theorem 4. *Let D and C be two finite subsets of P, so that C fully covers P. Assume that $G(D, C)$ is an optimal solution for AGPWFC(D, C). If $G(D, C)$ also covers P, then $G(D, C)$ is an optimal solution for AGPFC(C).*

Proof: Assume that $G(D, C)$ covers P, but it is not an optimal solution for AGPFC(C). Then, there exists $G' \subseteq C$ with $|G'| < |G(D, C)|$ such that G' is a feasible solution for AGPFC(C), i.e., G' covers P. This implies that G' is also a feasible solution for AGPWFC(D, C), contradicting the fact that $G(D, C)$ is optimal for this problem. ■

III. The Algorithm

The core idea of our algorithm consists in computing increasing lower bounds and decreasing upper bounds for the AGP until a proven optimal solution is found or a pre-established maximum time limit is exceeded. The procedure for obtaining these bounds involves the resolution of discretized versions of the AGP. To find a new lower bound, an AGPW instance is solved, while in the upper bound case, an AGPFC instance in which the guard candidate set covers the polygon is worked out. Observe that an optimal solution for such an instance of the AGPFC is also viable for the AGP, since the AGPFC asks for a solution that guards the entire polygon. Consequently, reducing the gap between bounds to zero means reaching an optimal solution for the original AGP.

In Algorithm 1, we summarize how our technique works. After initializing the witness and guard candidate sets, the algorithm enters its main loop (lines 4 to 10). At each iteration, new lower and upper bounds are computed and, if optimality has not been proved, the witness and guard candidate sets are updated in line 8.

In the following two subsections, the procedures for solving AGPW (line 5) and AGPFC (6) instances are described. After this, Section III-C briefly describes the resolution method for AGPWFC instances through ILP techniques. Both the AGPW and the AGPFC can be reduced to an AGPWFC instance, justifying why we focus on the latter. In Section III-D, we present how the management of the witness set is done and, in the following section, we discuss the selection of guard candidates. Section III-F gathers all the algorithmic information presented previously and describes the complete algorithm for the AGP.

A. Solving the AGPW

The resolution of an AGPW on D allows for the discovery of a new lower bound for the AGP, since fully covering P requires at least as many guards as the minimum sufficient to cover the points in $D \subset P$. However, despite being a simplification of the original AGP problem, we are still dealing with an infinite number of potential guard positions, which does not allow for a direct reduction to the set cover problem. Thus, our approach is based on discretizing the guard candidate set, creating an AGPWFC instance from our original AGPW. To do this, we apply Theorem 1, presented in Section II-D.

From Theorem 1, one concludes that there exists an optimal solution for AGPW(D) in which all the guards are in Light AVPs of the arrangement induced by D. Besides, as every vertex of an AVP can see at least the same set of witnesses observed by the points inside it, we can state that there is

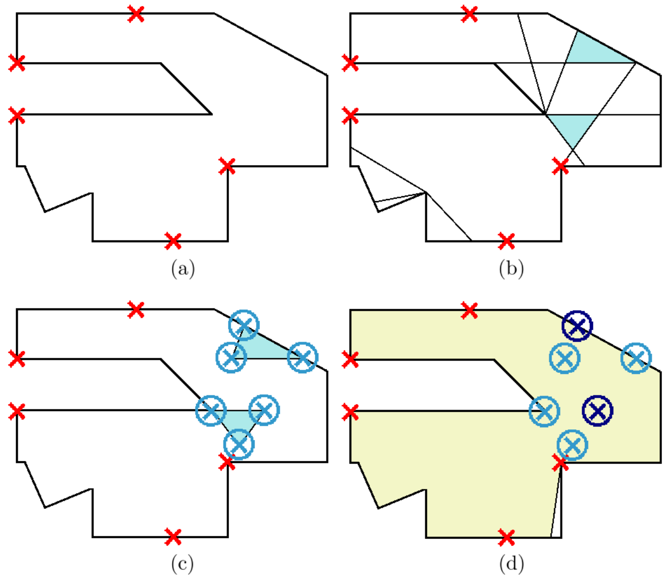


Fig. 10. **Algorithm 2:** (a) AGPW(D); (b) The light AVPs of Arr(D); (c) The guard candidate set; (d) An optimal solution $G \subseteq V_{\mathcal{L}}(D)$ for AGPW(D).

an optimal solution G where each point in G belongs to the set $V_{\mathcal{L}}(D)$ of all vertices from the light AVPs of Arr(D). Therefore, an optimal solution for AGPW(D) can be obtained simply by solving AGPWFC($D, V_{\mathcal{L}}(D)$), as illustrated in the example of Fig. 10. As seen before, the latter problem can be modeled as an ILP, where the numbers of constraints and of variables are polynomial in $|D|$. This follows, as mentioned in Section II-A, from the fact that the number of AVPs (and vertices) in Arr(D) is bounded by a polynomial in $|D|$ and $|P|$. Algorithm 2 shows a pseudo-code of the AGPW resolution method.

As will be shown in Section III-E, the guard candidate set used in the implemented algorithm for the AGP is not actually equal to $V_{\mathcal{L}}(D)$. The final set C includes additional points and, depending on the discretization strategy used, may employ points from $C_{\mathcal{L}}(D)$, which are located in the interior of light faces, instead of the ones in $V_{\mathcal{L}}(D)$.

B. Solving the AGPFC

As we now know how to generate lower bounds for the AGP, the next task is to compute good upper bounds for the problem. A possible way to achieve this is through the resolution of an AGPFC instance in which the guard candidate set is known to cover the polygon. Under this condition, an AGPFC solution is always viable for the original problem.

In contrast to what was discussed regarding the AGPW, we now have a finite number of guard candidates and an infinite number of points to be covered. Therefore, a direct resolution method using a reduction to an SCP is not possible. To circumvent this, our algorithm discretizes the original AGPFC instance, relying on what Theorem 4 establishes. Theorem 4 shows that an optimal solution for the AGPFC may be obtained through the resolution of an AGPWFC instance, provided it fully covers P. Whenever an optimal solution for

Algorithm 1 AGP (Sketch)

```
1: Set UB $\leftarrow |V|$ and LB $\leftarrow 0$
2: Select the initial witness set $D$
3: Select the initial guard candidate set $C \supseteq V$
4: while (UB $\neq$ LB) and (MAXTIME not reached) do
5:     Solve AGPW($D$), set $G_w \leftarrow$ optimal solution of AGPW($D$) and LB $\leftarrow \max\{$LB$, |G_w|\}$
6:     Solve AGPFC($C$), set $G_f \leftarrow$ optimal solution of AGPFC($C$) and UB $\leftarrow \min\{$UB$, |G_f|\}$
7:     if (UB $\neq$ LB) then
8:         Update $D$ and $C$
9:     end if
10: end while
11: return $G_f$
```

Algorithm 2 AGPW(D)

```
1: Arr($D$) $\leftarrow$ construct the arrangement from the visibility polygons of the points in $D$
2: $V_{\mathcal{L}}(D) \leftarrow$ identify the vertices of the light AVPs of Arr($D$)
3: $C \leftarrow V_{\mathcal{L}}(D)$
4: Solve AGPWFC($D$, $C$): set $G_w \leftarrow$ optimal solution of AGPWFC($D$, $C$)
5: return $G_w$
```

the simplified version (AGPWFC) leaves uncovered regions in P, additional work is required. To guarantee that we attain an optimal solution for the AGPFC, we will employ here a technique designed by Couto et al. [5] to solve the AGPV, a special case of the AGPFC where $C = V$, but which may be used to handle the general case without significant changes.

Initially, consider that we have a finite witness set D. Using the guard candidate set C, we can create and solve the AGPWFC(D,C) instance. If the solution fully covers the polygon, we have satisfied the hypothesis of Theorem 4 and, consequently, we have an optimal solution for the AGPFC. Otherwise, there are regions of the polygon that remain uncovered. We now update the witness set, adding new points within the uncovered regions, denoted $C_{\mathcal{U}}(G)$, and repeat the process.

As demonstrated in [5] by Couto et al., the iterative method for solving the AGPFC converges in polynomial time. To clarify this point, consider Theorem 2 and its proof. This theorem states that constructing D by choosing only one point within each AVP of Arr(C) is enough to ensure the whole coverage of P. As the number of AVPs is polynomial (see Section II-A) and we iteratively construct D by choosing witnesses in the interior of uncovered AVPs of Arr(C), it is straightforward that the number of iterations is bounded by the polynomial complexity of Arr(C). Although the convergence time for AGPFC is theoretically guided by this complexity, Couto et al. showed throw extensive experiments that, in practice, the number of necessary iterations is much lower. Moreover, it can even be argued that it suffices to select one point from each shadow AVP of the arrangement induced by C (see Theorem 3). Fig. 11 shows how the algorithm for the AGPFC iteratively adds new witnesses in different AVPs until the polygon is fully covered.

A pseudo-code for the algorithm employed to solve the AGPFC is shown in Algorithm 3.

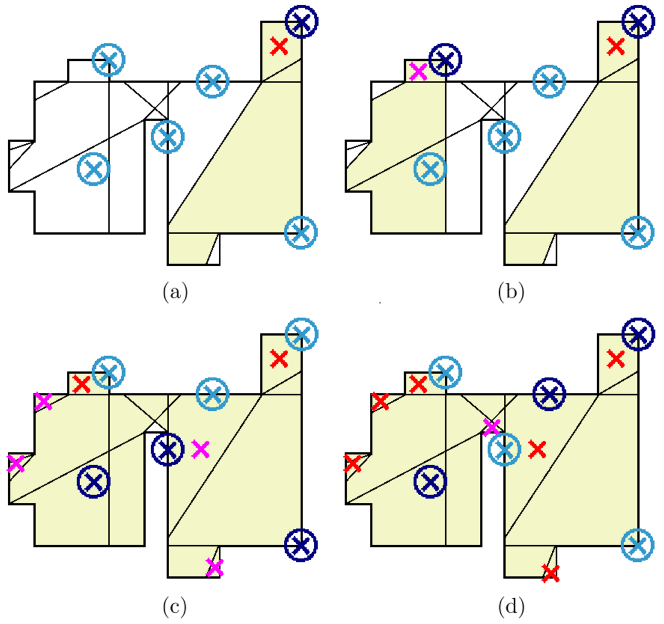


Fig. 11. **Algorithm 3:** A sequence of AGPWFC(D,C) instances is generated until a viable solution for the AGP is obtained.

C. *Solving the AGPWFC (SCP)*

Having reduced the AGPW and the AGPFC into AGPWFC instances in order to obtain the desired bounds, the objective becomes solving the latter efficiently. Since AGPWFC(D, C) can be easily reduced to an SCP, where the witnesses in D are the elements to be covered and the visibility polygons of the guard candidates in C are the subsets considered, we will make use of the ILP formulation for SCP presented in Section II-B.

Algorithm 3 AGPFC(C)

```
1: $D_f \leftarrow$ initial witness set
2: repeat
3:     Solve AGPWFC($D_f, C$): set $G_f \leftarrow$ optimal solution
4:     if $G_f$ does not fully cover $P$ then
5:         $D_f \leftarrow D_f \cup C_{\mathcal{U}}(G_f)$
6:     end if
7: until $G_f$ fully covers $P$
8: return $G_f$
```

A simple and straightforward approach would be to directly use an ILP solver, such as XPRESS [13], CPLEX [14] or GLPK [15], since even large instances of the ($\mathbb{NP}$-hard) SCP can be solved quite efficiently by many modern integer programming solvers. However, as observed in our experiments, some AGP instances can generate significantly complex and very large SCP instances, rendering the solvers less efficient. For this reason, some known techniques were implemented to improve the solvers' running time. Among these, the most effective consisted in the reduction on the number of constraints and variables in the initial model. Moreover, we also implemented a Lagrangian Heuristic to help the solver obtaining initial viable solutions. Details of these implementations can be seen in the full text of the dissertation [12].

D. Witness Management

The witnesses selected during the execution of our algorithm play an important role in the search for optimal solutions for the AGP. The witness set D is not only decisive for producing good lower bounds through the resolution of AGPW(D), but it is crucial to find tight upper bounds, since the candidate set C of AGPFC(C) is constructed from Arr(D). In this context, choosing D wisely may lead to a lower gap between the bounds and, consequently, to a lower number of iterations of Algorithm 1.

Recall that, before the first iteration of Algorithm 1, an initial witness set is chosen and, in the following ones, it gets suitably updated (line 5). In this section, we present all the initialization alternatives that were considered since our first work [6] and, after this, we discuss how the set D is updated.

The first initialization choice, called *All-Vertices* (AV), consists in using all vertices of the polygon as witnesses, i.e., $D = V$. Besides the easy construction of this set, it was confirmed in tests that the coverage of such points is usually a good start for covering the whole polygon.

In an attempt to begin with a smaller number of witnesses, we also considered initializing D with only the convex vertices of P. This strategy is referred to as the *Convex-Vertices* (CV) initialization. The reason for reducing the initial witness set lies on the fact that a smaller set D is likely to lead to a lower number of visibility polygon calculations, to a less complex visibility arrangement and, consequently, to a simpler SCP model. In addition, we decided to choose only convex vertices because the reflex ones are usually more exposed due to its wider visibility angle.

The third alternative is based on a work by Chwa et al. [16]. In this paper, a polygon P is defined to be *witnessable* when there exists a finite witness set $W \subset P$ with the property that any set of guards that covers W must also cover the entire polygon. The authors also present an algorithm that computes a minimum witness set for a polygon whenever it is witnessable. The method for constructing this minimum witness set consists in placing a witness in the interior of every *reflex-reflex* edge of P and on the convex vertices of every *convex-reflex* edge. The terms *convex* and *reflex* here refer to the interior angles at a vertex or at the endpoints of an edge. Based on these selection criteria, we devised our third discretization method, called *Chwa-Points* (CP), which assembles the initial witness set for our algorithm from the midpoints of all reflex-reflex edges and all convex vertices from convex-reflex edges.

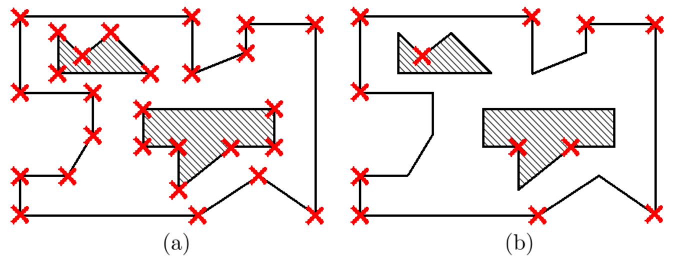
It follows from the results in [16] that, when the Chwa-Points discretization is used for a witnessable input polygon, our AGP algorithm will find an optimal solution in a single iteration. However, as observed in our experiments, non-witnessable polygons are the norm. In fact, among our random benchmark instances, they constitute the vast majority.

Finally, an additional discretization was created based on CP, in an attempt to improve the results previously obtained. In this strategy, called *Chwa-Extended* (CE), besides all points from CP, we also include all reflex vertices of P in the initial discretization.

An example of each one of the four discretization strategies implemented is presented in Fig. 12. Notice that, when characterizing vertices of a hole, the terms *convex* and *reflex* have their meaning inverted.

Let us now focus on the updating process of the witness set throughout the algorithm. This procedure takes place in two different occasions: while solving an AGPFC instance (in line 5 of Algorithm 3) and when jumping to the next iteration of the AGP algorithm (line 8 of Algorithm 1). In the first case, as presented in Section III-B, new witnesses are positioned considering the current solution of AGPWFC(D_f,C). Here, we add to D_f one point placed in the interior of each uncovered region, which (as explained in Section III-B) is enough to guarantee the convergence of the AGPFC resolution method. See Fig. 13 for an example.

On the other hand, a deeper analysis is necessary when dealing with the update procedure of the main algorithm, because, as discussed at the beginning of this section, the



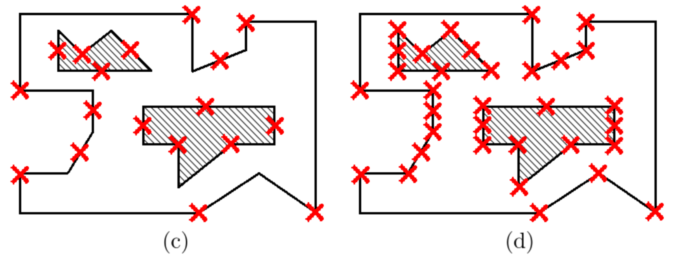


Fig. 12. Examples of the initial set D when using each one of the following discretization strategies: (a) All-Vertices (AV); (b) Convex-Vertices (CV); (c) Chwa-Points (CP); (d) Chwa-Extended (CE).

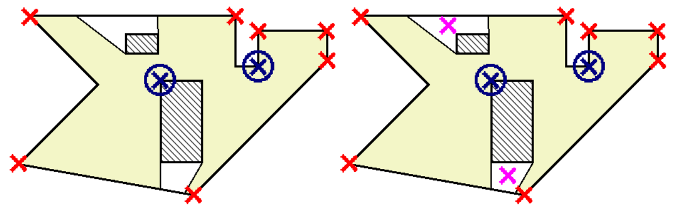


Fig. 13. Solution of an AGPWFC instance (left); New witnesses chosen (violet) for the next iteration of the AGPFC algorithm (right).

selection of D affects all significant parts of the technique. In a summarized form, the better the choice of a new set of points for inclusion into the witness set, the better the lower and the upper bounds obtained would be and, consequently, the faster the convergence.

In essence, the process consists of adding points from the uncovered regions arising from the solution of the previous AGPW instance. Our first attempt was to imitate the strategy adopted by the AGPFC algorithm and to position one new witness inside each uncovered region. However, our experiments later showed that the inclusion of only these points were not sufficient to lead the algorithm to good performance and convergence. The shortfall was traced to the absence of new points on the boundary of the polygon, which proved to be useful to the evolution of the bounds obtained. Therefore, whenever an edge of an uncovered region is found to be contained on the boundary of the polygon, its vertices and its midpoint are also selected to increment the witness set throughout the iterations. These points along with an interior point of each uncovered region form the whole set M of new witnesses. Note that the selection of midpoints in this procedure is arbitrary and, in general, can have a better or worse effect than choosing any other non-extreme point of the segment. Fig. 14 displays an example of how this updated procedure works.

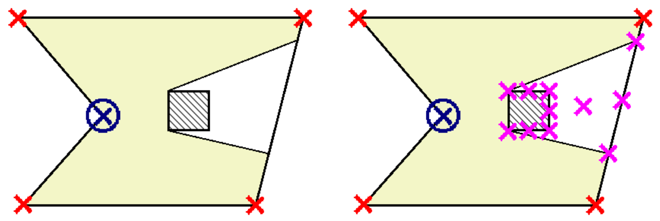


Fig. 14. Solution of an AGPW instance (left); New witnesses chosen (violet) for the next iteration of the AGP algorithm (right).

E. Guard Candidate Management

After constructing the arrangement induced by the set of witnesses D and identifying the corresponding light AVPs, our algorithm is able to build the guard candidate set C. This set must be built in such a way that guarantees that an optimal solution G for AGPW(D) is contained in C. After solving AGPW(D), C is maintained and also used in AGPFC(C), contributing to the discovery of a new upper bound. In this section, we present, in details, how these candidates are selected.

Since our first published work [6], two different discretization strategies for C have been experimented. Both of them follow the idea presented in Theorem 1 and construct C using at least one point from each existing light AVP, thereby ensuring that AGPW(D) is optimally solved. In addition, as using only points from Light AVPs may not guarantee the existence of a solution that covers the entire polygon (a necessary requirement for the AGPFC solvability), both strategies also include all vertices of P ($V \subset C$).

Our first strategy, named *Boundary-Guards* (BG), contains, besides the vertices of P, all points from $V_{\mathcal{L}}(D)$ ($C = V \cup V_{\mathcal{L}}(D)$). This discretization was originally the only method used for choosing C in the two papers that describe our algorithm [6], [7].

However, recall that the arrangement does not grow linearly with the number of witnesses in D. This way, in our experiments with large polygons, the tasks which depend on C, such as the computation of visibility between pairs of points, become increasingly time consuming. For example, in a simple polygon with 5000 vertices, we may have to compute more than 100 million visibility tests between candidates and witnesses in a single iteration.

In this context, a new discretization with a lower number of guard candidates was experimented in [17]. This time, the points from $V_{\mathcal{L}}(D)$ were not included in C. The idea, named *Center-Guards* (CG), was to replace the vertices of a given light AVP by an internal point of it, reducing the number of these candidates by a factor of at least 3. For an example of BG and CG strategies, see Fig. 15.

F. Resulting Algorithm

Once each of the main steps of the algorithm is understood, we are able to describe how these parts fit together to comprise

Algorithm 4 AGP(P)

```
1:  $D \leftarrow$ initial witness set   {see Section III-D}
2:  Set LB $\leftarrow 0$, UB $\leftarrow n$ and $G^* \leftarrow V$
3:  loop
4:      Solve AGPW($D$): set $G_w \leftarrow$ optimal solution and $z_w \leftarrow |G_w|$   {see Section III-A}
5:      if $G_w$ is a coverage of $P$ then
6:          return $G_w$
7:      end if
8:      LB $\leftarrow \max\{\text{LB}, z_w\}$
9:      if LB = UB then
10:         return $G^*$
11:     end if
12:     $C \leftarrow V_{\mathcal{L}}(D) \cup V$ (or $C_{\mathcal{L}}(D) \cup V$)   {see Section III-E}
13:     $U \leftarrow C_{\mathcal{U}}(G_w)$   {one additional point per uncovered region}
14:     $D_f \leftarrow D \cup U$
15:     Solve AGPFC($C$), using $D_f$: set $G_f \leftarrow$ optimal solution and
        $z_f \leftarrow |G_f|$   {see Section III-B}
16:     UB $\leftarrow \min\{\text{UB}, z_f\}$ and, if UB $= z_f$ then set $G^* \leftarrow G_f$
17:     if LB = UB then
18:         return $G_f$
19:     end if
20:     $D \leftarrow D \cup U \cup M$   {see Section III-D}
21: end loop
```

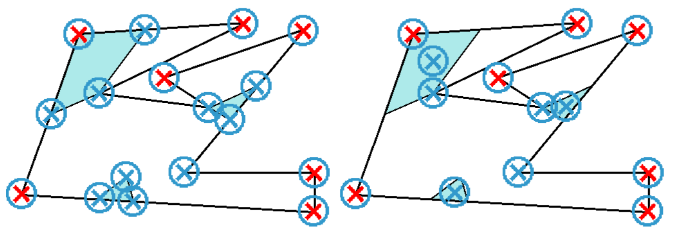


Fig. 15. Examples of the guard candidate set C when using each of the following discretizations: Boundary-Guards (BG) (left); Center-Guards (CG) (right);

the complete algorithm. Algorithm 4 sums up how the process as a whole works.

It is important to notice that the set of guard candidates used in the AGPW resolution is actually the set C from the AGPFC(C) instance solved on Line 15. This means that the AGPW resolution is actually the first iteration of the AGPFC algorithm (Algorithm 3). Thus, all information obtained during the solution of AGPW(D) can be reused for AGPFC(C), which improves the performance of the implementation.

Another relevant aspect of this algorithm is that information on bounds may be used throughout the iterations in order to skip unnecessary steps. For instance, if an upper bound UB was found in a previous iteration and a new AGPFC instance is being solved, whose current solution is not lower than UB, then we may stop the AGPFC resolution before obtaining an optimal solution since the upper bound can not be improved.

IV. Implementation and Computational Results

To verify the quality of the new algorithm described, we coded it in the C++ programming language and used the Computational Geometry Algorithms Library (CGAL) [18] to benefit from visibility operations, arrangement constructions and other geometric tasks. In addition, to exactly solve the AGP discretizations, we employed ILP solvers like XPRESS [13], CPLEX [14] and the free package GLPK [15].

A. Instances

We experimented with AGP instances from 4 sources: [5], [19], [20] and [7]. The instances, whose sizes vary from 20 to 5000 vertices, can be divided into 6 classes of polygons: simple, orthogonal, von Kock, simple-simple (simple with holes), ortho-ortho (orthogonal with holes) and spike. Testing with different polygon classes was crucial to verify the robustness of the technique. In total, we tested our algorithm on 2880 instances and obtained an overall optimality rate of more than 90%. Fig. 16 presents examples of each class of polygons used in our experiments.

B. Evolution of the Implementation

Our implementation went through several versions since its first release. Such changes, which included the employment of new routines, data structures and decisions, substantially improved the performance of the software. Our first version (I1) was completed in late 2012 and was reported in a conference paper [6]. Shortly after that, in the first half of 2013, a second version (I2) was produced and described in a full paper [7]. The latest version (I3) was developed during a two-month internship at the Technische Universität

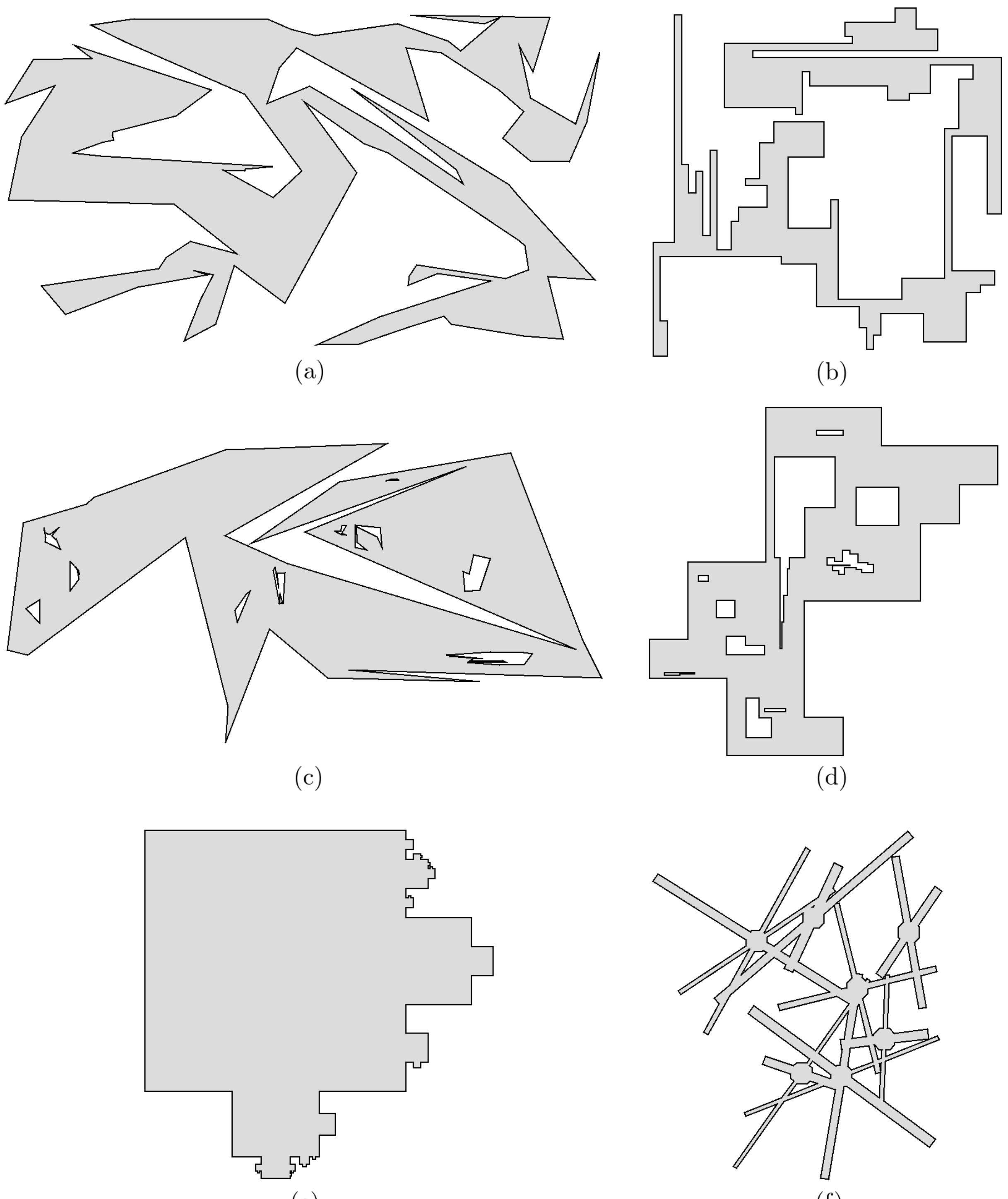


Fig. 16. Examples of instances from different classes. (a) Simple; (b) Orthogonal; (c) Simple-simple; (d) Ortho-ortho; (e) von Koch; (f) Spike.

Braunschweig (TUBS), in Germany, under the supervision of Prof. Dr. A. Kröller. During this visit, the student worked in straight collaboration with researchers from the Algorithms Group headed by Prof. Dr. S. Fekete. Their group has also developed methods for the AGP, including a solution for the original problem [20]. Our new implementation is described in a survey on algorithms for Art Gallery Problems [17], in a joint work by researchers from UNICAMP and TUBS.

During the writing of the AGP Survey [17], we had the chance of experimenting I1, I2 and the new version I3 on 900 instances in a single environment in the laboratory of the Algorithms Group at TUBS. The joint experimentation provided a direct comparison and verification of the improvement occurred during the Master's project. In this occasion, I3 showed to be very successful, being able to optimally solve 768 polygons, including instances with 5000 vertices, in runs of less than 20 minutes. Table I displays the optimality rates achieved by I1, I2 and I3.

TABLE I
OPTIMALITY RATES OF I1, I2 AND I3.

Class	**Source**	n	**Optimality Rate (%)**		
			I1	**I2**	**I3**
Simple	From [5]	200	100.00	100.00	100.00
		500	100.00	100.00	100.00
		1000	96.67	100.00	100.00
		2000	6.67	50.00	100.00
		5000	0.00	0.00	100.00
Orthogonal	From [5]	200	100.00	100.00	96.67
		500	100.00	96.67	93.33
		1000	100.00	100.00	100.00
		2000	70.00	90.00	100.00
		5000	0.00	0.00	93.33
Simple-simple	From [17]	200	-	100.00	100.00
		500	-	83.33	100.00
		1000	-	0.00	100.00
		2000	-	0.00	46.67
		5000	-	0.00	0.00
Ortho-ortho	From [7]	200	-	96.67	100.00
		500	-	83.33	100.00
		1000	-	3.33	96.67
		2000	-	0.00	33.33
		5000	-	0.00	0.00
von Koch	From [5]	200	100.00	100.00	100.00
		500	96.67	100.00	100.00
		1000	46.67	100.00	100.00
		2000	0.00	0.00	100.00
		5000	0.00	0.00	0.00
Spike	From [20]	200	-	100.00	100.00
		500	-	100.00	100.00
		1000	-	96.67	100.00
		2000	-	96.67	100.00
		5000	-	0.00	100.00

The results in Table I evince two big steps in our headway. From I1 to I2, besides a considerable improvement in the optimality rate, we became able to solve polygons with holes, greatly increasing the range of treatable classes. Subsequently, from I2 to I3, a remarkable performance improvement was conquered, as evidenced by the resolution of polygons of 5000 vertices. These polygons have twice the size of the previous largest instances already treated by AGP solvers, fact achieved by I2 in [7].

TABLE II
AVERAGE TIME OF I1, I2 AND I3.

Class	**Source**	n	**Average Time (sec)**		
			I1	**I2**	**I3**
Simple	From [5]	200	7.31	3.63	0.75
		500	67.81	32.82	2.96
		1000	358.97	158.73	9.18
Orthogonal	From [5]	200	4.10	2.72	0.37
		500	30.06	19.61	1.49
		1000	189.41	111.40	5.22
von Koch	From [5]	200	11.12	3.54	1.20
		500	158.53	31.88	7.80
		1000	767.01	186.49	52.81

In order to confirm this analysis, we collected information about the time necessary to find optimal solutions. Table II shows the average time needed to solve simple, orthogonal and von Koch polygons, considering only instances where optimal solutions were found by all three implementations. From this table, one can see that the average time of I2 can be about 5 times smaller than I1, as verified in results of von Koch polygons with 500 vertices. The difference is even greater when analyzing I2 against I3, which is capable of solving, on average, orthogonal polygons of size 1000 almost 22 times faster than I2.

C. Comparison With Other Techniques

In recent years, other algorithms were proposed for the AGP. In this Master's thesis, we compared our achievements with the two of them that excelled the most. First, we analyzed the differences between our method and the work by Bottino and Laurentini in 2011 [19]. In [19], Bottino and Laurentini proposed a heuristic for the original AGP, aiming to produce good viable solutions with an efficient method. The technique was experimented and obtained promising results, including some optimal solutions. In the paper, the authors compared their technique with the one by Amit et al. [21] and claimed that their method was able to achieve better results.

Upon learning about this work, we decided to try our I2 version with exactly the same instances used by Bottino and Laurentini and compare our findings. The experiments were done using all simple and orthogonal instances from Bottino et al., which vary between 30 and 60 vertices. Table III summarizes the results, showing two types of information: average number of guards and average run time.

In our tests, I2 was able to find proven optimal solutions for all instances, meaning that the column with average number of guards found by our method actually contains optimal values. Knowing this, we can conclude that the heuristic by Bottino and Laurentini was able to find good solutions, but not always

TABLE III
COMPARISON BETWEEN THE METHOD OF BOTTINO ET AL. [19] AND I2.

Class	n	Number of Guards		Time (sec)	
		Method [19]	I2	Method [19]	I2
Simple	30	4.20	4.20	1.57	0.14
	40	5.60	5.55	2.97	0.10
	50	6.70	6.60	221.92	0.24
	60	8.60	8.35	271.50	0.27
Orthogonal	30	4.60	4.52	1.08	0.04
	40	6.10	6.00	9.30	0.07
	50	7.80	7.70	6.41	0.12
	60	9.30	9.10	81.95	0.16

optimal. Except for simple polygons with 30 vertices, the heuristic did not manage to find the best possible solutions for all polygons of a subgroup. Looking more carefully, we can notice a growing gap between the average number of guards from both techniques as the size of the instances increases.

Besides comparing the quality of the solutions, it is also important to evaluate the time needed to find them. To this end, Table III exhibits the computing times for the two methods. It is important to notice though that the experiments were done in different environments, which invalidates a direct comparison of performance between the techniques. While our tests were conducted in machines featuring an Intel® Core™ i7-2600 at 3.40 GHz and 8 GB of RAM, the researchers of [19] performed their experiments on an Intel® Core2™ processor at 2.66 GHz and with 2 GB of RAM. Despite this, Table III shows that the average run time of our technique to compute proven optimal solutions for the AGP is orders of magnitude smaller than the time used by the heuristic. At least it seems safe to say that this large disparity in computing times can not be entirely attributed to hardware and software differences.

Finally, we also performed a complete comparison with the most recent implementation of the algorithm presented in 2012 by Kröller et al. [20], based on results obtained from the survey on algorithms for the AGP [17]. All experiments were performed during the internship at TUBS on the same computing environment, enabling a fair comparison between the two techniques. Considering all experiments, where only polygons of size 200, 500, 1000, 2000 and 5000 were considered, our current implementation (I3) was very successful. I3 was able to optimally solve 768 of 900 polygons, including instances with 5000 vertices, in runs of less than 20 minutes. Meanwhile, Kröller et al. technique (here called BS3) could solve 583 instances. In addition, while our version was able to obtain 100% optimality in 21 out of 30 subgroups of instances considered, the version from TUBS only achieved this for 4 subgroups. Table IV shows the optimality results.

To get a deeper insight into the differences in behavior of the techniques, we also developed a running time comparison between them, using results of all polygon classes. This comparison is shown in Fig. 17. For a fairer analysis, the average times in the charts only considered values of instances resolved by both I3 and TUBS implementation.

TABLE IV
OPTIMALITY RATE OF OUR IMPLEMENTATIONS AND BS3.

Class	Source	n	Optimality Rate (%)	
			BS3	I3
Simple	From [5]	200	96.67	**100.00**
		500	96.67	**100.00**
		1000	90.00	**100.00**
		2000	60.00	**100.00**
		5000	26.67	**100.00**
Orthogonal	From [5]	200	96.67	96.67
		500	93.33	93.33
		1000	86.67	**100.00**
		2000	70.00	**100.00**
		5000	40.00	93.33
Simple-simple	From [7]	200	86.67	**100.00**
		500	60.00	**100.00**
		1000	13.33	**100.00**
		2000	0.00	46.67
		5000	0.00	0.00
Ortho-ortho	From [7]	200	86.67	**100.00**
		500	53.33	**100.00**
		1000	16.67	96.67
		2000	0.00	33.33
		5000	0.00	0.00
von Koch	From [5]	200	**100.00**	**100.00**
		500	93.33	**100.00**
		1000	96.67	**100.00**
		2000	86.67	**100.00**
		5000	0.00	0.00
Spike	From [20]	200	96.67	**100.00**
		500	**100.00**	**100.00**
		1000	**100.00**	**100.00**
		2000	**100.00**	**100.00**
		5000	96.67	**100.00**

In Fig. 17, it is easy to see that BS3 was faster in solving simple-simple, ortho-ortho and von Koch polygons. On the other hand, I3 was more efficient with simple polygons and meaningly better when dealing with orthogonal and spike instances. In the specific case of the spike class, I3 was about 20 times faster than Kröller et al. implementation to solve the instances with 5000 vertices.

Through all results presented, one can conclude that the method from TUBS have a natural difficulty in converging to a proven optimal solution. While some positive results where observed in run time, the optimality rate of BS3 was not able to follow it. For illustration, BS3 needed an average time of 164.65 seconds to solve simple-simple polygons with 1000 vertices (26% percent less than I3), but the optimality rate for this subgroup was only 13.33%, far below the results using I3, when **all** 30 instances were solved within the imposed time limit. In the case of our method, it seems that our technique, since its first release, tends to find the optimal solution in almost all cases and the low optimality observed in larger instances is directly related to the maximum run time imposed in the testing environment.

D. More Results

In the thesis [12], a complete analysis is presented as well as results of the entire set of experiments, which include other interesting data, as, for example, the effect of employing different techniques for choosing the witness and the guard

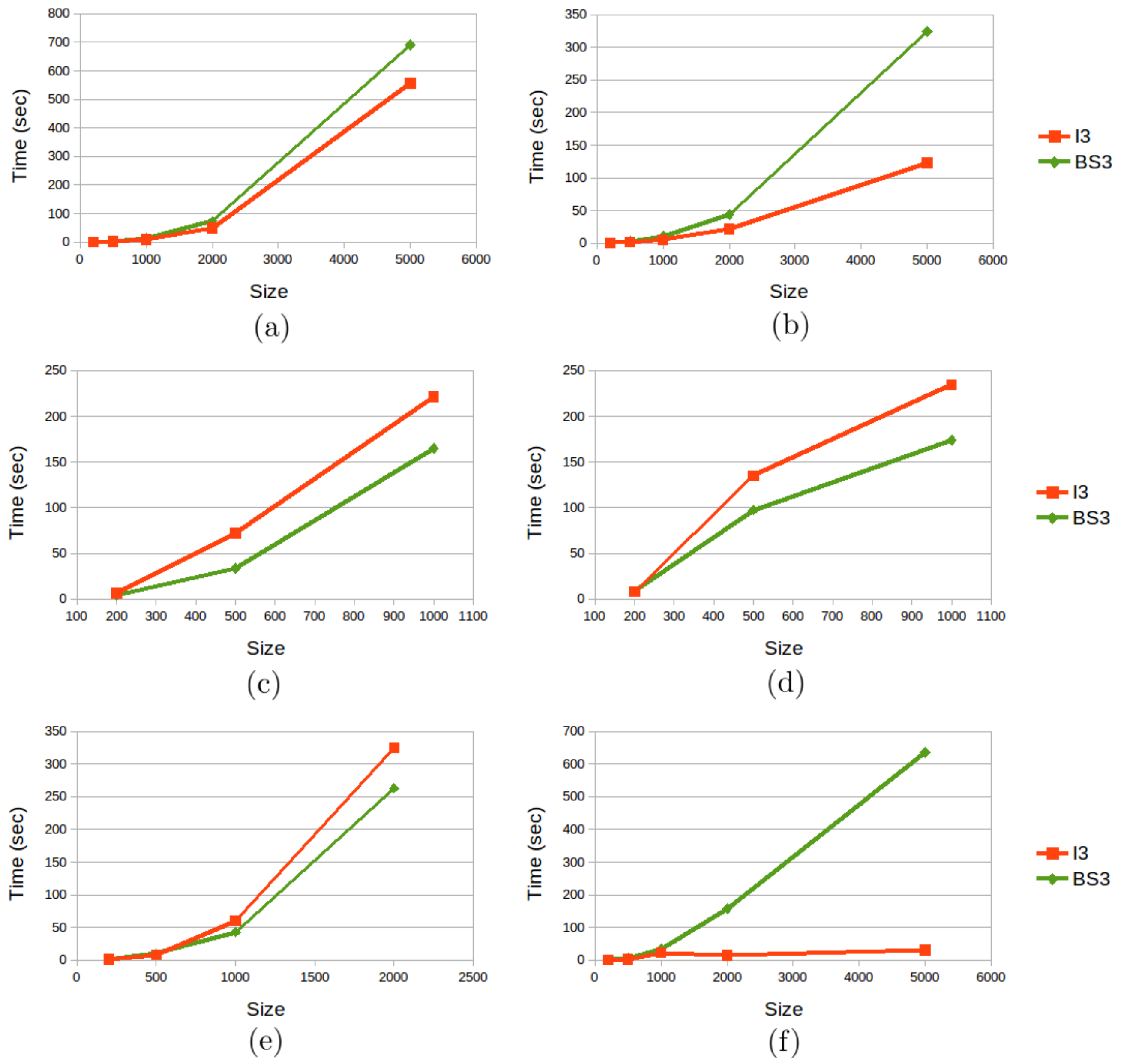


Fig. 17. Performance comparison between I3 and TUBS' technique (BS3) when solving the following classes: (a) Simple; (b) Orthogonal; (c) Simple-simple; (d) Ortho-ortho; (e) von Koch; (f) Spike. Here, only fully solved instances are considered.

candidate sets. In addition, we also reported the results of using different ILP solvers in our implementation.

V. Conclusion

In this work, we designed an algorithm to optimally solve the AGP. The method iteratively discretizes the original problem to find lower and upper bounds while seeking an optimal solution for this $\mathbb{NP}$-hard problem. To allow its correct evaluation, our algorithm was coded and had its implementation modified and optimized over time. In total, we tested our technique on more than 2800 instances from different sources and classes of polygons. Our methodology proved capable of optimally solving polygons with up to 5000 vertices in less than 20 minutes each, something not possible a few years ago.

Moreover, we also compared our results with those produced by other state-of-the-art techniques. These comparisons revealed a significant advantage when using our technique, which proved to be far more effective, faster and more robust than all the others. These results encouraged us to release a free source implementation of our algorithm on the web page of the project [8]. By doing so, we expect to contribute to future research on the topic, since it is now possible for new techniques to be directly tested and compared to our software package.

Lastly, this research generated four papers on the subject, two of which have already been published [22], [6] and two recently submitted [7], [17]. These studies provided a strong interaction with other researchers on the topic, as was the case of the survey for the AGP [17], produced in partnership with a group from TUBS.

REFERENCES

[1] V. Chvátal, "A combinatorial theorem in plane geometry," *Journal of Combinatorial Theory Series B*, vol. 18, pp. 39–41, 1975.

[2] D. T. Lee and A. Lin, "Computational complexity of art gallery problems," *Information Theory, IEEE Transactions on*, vol. 32, no. 2, pp. 276–282, March 1986.

[3] A. Aggarwal, S. K. Ghosh, and R. K. Shyamasundar, "Computational complexity of restricted polygon decompositions," in *Computational Morphology*, G. T. Toussaint, Ed. North-Holland, 1988, pp. 1–11.

[4] A. Kröller, T. Baumgartner, S. P. Fekete, M. Moeini, and C. Schmidt, "Practical solutions and bounds for art gallery problems," August 2012. [Online]. Available: http://ismp2012.mathopt.org/show-abs?abs=1046

[5] M. C. Couto, P. J. de Rezende, and C. C. de Souza, "An exact algorithm for minimizing vertex guards on art galleries," *International Transactions in Operational Research*, vol. 18, no. 4, pp. 425–448, 2011. [Online]. Available: http://dx.doi.org/10.1111/j.1475-3995.2011.00804.x

[6] D. C. Tozoni, P. J. de Rezende, and C. C. de Souza, "The quest for optimal solutions for the art gallery problem: A practical iterative algorithm," in *Proceedings of the 12th International Symposium on Experimental Algorithms, SEA 2013*, ser. Lecture Notes in Computer Science, V. Bonifaci, C. Demetrescu, and A. Marchetti-Spaccamela, Eds., vol. 7933. Rome, Italy: Springer, 2013, pp. 320–336.

[7] ——, "A practical iterative algorithm for the art gallery problem using integer linear programming," *Optimization Online*, Oct. 2013, www.optimization-online.org/DB_HTML/2013/11/4106.html.

[8] P. J. de Rezende, C. C. de Souza, M. C. Couto, and D. C. Tozoni, "The Art Gallery Problem Project (AGPPROJ)," 2013, *www.ic.unicamp.br/∼cid/Problem-instances/Art-Gallery*.

[9] P. Bose, A. Lubiw, and J. I. Munro, "Efficient visibility queries in simple polygons," *Computational Geometry*, vol. 23, no. 3, pp. 313–335, 2002.

[10] S. K. Ghosh, "Approximation algorithms for art gallery problems," in *Proc. Canadian Inform. Process. Soc. Congress*. Mississauga, Ontario, Canada: Canadian Information Processing Society, 1987, pp. 429–434.

[11] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*, ser. The IBM Research Symposia Series, R. Miller, J. Thatcher, and J. Bohlinger, Eds. Springer US, 1972, pp. 85–103. [Online]. Available: http://dx.doi.org/10.1007/978-1-4684-2001-2_9

[12] D. C. Tozoni, "Solving the Art Gallery Problem: A Practical and Robust Method for Optimal Point Guard Positioning," Master's thesis, University of Campinas, Campinas, Sao Paulo, Brazil, 2014.

[13] XPRESS, *Xpress Optimization Suite*, FICO Solutions, 2013, http://www.fico.com/en/products/fico-xpress-optimization-suite/ (access November 2013).

[14] CPLEX, *IBM CPLEX Optimizer*, IBM, 2015, http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/ (access June 2015).

[15] GLPK, *GNU Linear Programming Kit*, GNU, 2013, http://www.gnu.org/software/glpk/ (access December 2013).

[16] K.-Y. Chwa, B.-C. Jo, C. Knauer, E. Moet, R. van Oostrum, and C.-S. Shin, "Guarding art galleries by guarding witnesses," *Intern. Journal of Computational Geometry And Applications*, vol. 16, no. 02n03, pp. 205–226, 2006. [Online]. Available: http://www.worldscientific.com/doi/abs/10.1142/S0218195906002002

[17] P. J. de Rezende, C. C. de Souza, S. Friedrichs, M. Hemmer, A. Kröller, and D. C. Tozoni, "Engineering art galleries," 2014, submitted.

[18] CGAL, "Computational Geometry Algorithms Library," 2012, www.cgal.org (access January 2012).

[19] A. Bottino and A. Laurentini, "A nearly optimal algorithm for covering the interior of an art gallery," *Pattern Recognition*, vol. 44, no. 5, pp. 1048–1056, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/B6V14-51JF80D-1/2/4dd0f7df085fe27ab888c7c017f60512

[20] A. Kröller, T. Baumgartner, S. P. Fekete, and C. Schmidt, "Exact solutions and bounds for general art gallery problems," *J. Exp. Algorithmics*, vol. 17, no. 1, pp. 2.3:2.1–2.3:2.23, May 2012. [Online]. Available: http://doi.acm.org/10.1145/2133803.2184449

[21] Y. Amit, J. S. B. Mitchell, and E. Packer, "Locating guards for visibility coverage of polygons," in *ALENEX*. New Orleans, Lousiana: SIAM, January 2007, pp. 1–15.

[22] D. Borrmann, P. J. de Rezende, C. C. de Souza, S. P. Fekete, S. Friedrichs, A. Kröller, A. Nüchter, C. Schmidt, and D. C. Tozoni, "Point guards and point clouds: solving general art gallery problems," in *Proceedings of the twenty-ninth annual symposium on Computational geometry*, ser. SoCG '13. New York, NY, USA: ACM, 2013, pp. 347–348. [Online]. Available: http://doi.acm.org/10.1145/2462356.2462361