Exact Solutions and Bounds for General Art Gallery Problems*

Tobias Baumgartner† Sándor P. Fekete† Alexander Kröller† Christiane Schmidt†

Abstract

The classical Art Gallery Problem asks for the minimum number of guards that achieve visibility coverage of a given polygon. This problem is known to be NP-hard, even for very restricted and discrete special cases. For the case of vertex guards and simple orthogonal polygons, Cuoto et al. have recently developed an exact method that is based on a set cover approach. For the general problem (in which both the set of possible guard positions and the point set to be guarded are uncountable), neither constant-factor approximation algorithms nor exact solution methods are known.

We present a primal-dual algorithm based on linear programming that provides lower bounds on the necessary number of guards in every step and—in case of convergence and integrality—ends with an optimal solution. We describe our implementation and give results for an assortment of polygons, including non-orthogonal polygons with holes.

Keywords: Art Gallery Problem, Linear Programming, Duality, Separation, Geometry, Visibility

1 Introduction

The classical art gallery problem asks for the minimum number of guards placed inside of a polygon that suffice to perceive the entire polygon (interior and edges). For several classes of polygons and variants on the placement of guards this problem was shown to be NP-hard (e.g., [16]). Originally, interest from the theoretical side focused on extremal results, like the classical $\lfloor \frac{n}{3} \rfloor$ bound first established by Chvátal [5] and very elegantly proven by Fisk [12].

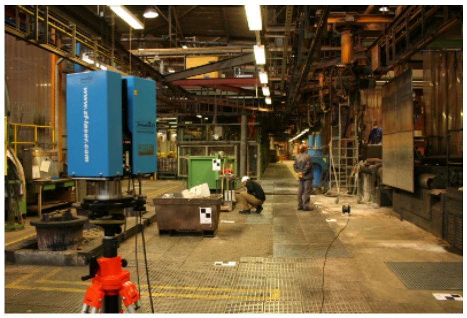
On the practical side, good solutions to art gallery problems have gained in importance, e.g., for measuring (the interior of) buildings using a static laser scanner (`http://www.inmetris3d.de/`). For these measuring tasks positions of the laser scanner must be identified that ensure coverage of the given environment, e.g., a production hall, a tunnel or a bridge construction, see Figure 1. Hence, a solution to the art gallery problem or good upper bounds enable the company to reduce the working hours, both during the actual scan process on site and for the postprocessing (scan matching etc.). This also applies to lower bounds: Not only do they allow quality estimates of feasible solutions, they are also of crucial importance for contract bidding, allowing an estimate of the necessary expenses for personnel and equipment that cannot be avoided, neither by the company, nor its competitors.

Related Algorithmic Work. In recent years, there has been a growing amount of work dealing with algorithmic aspects of the art gallery problem. Even for the restricted case of vertex guards and simple polygons, Eidenbenz et al. [11] established lower bounds on the achievable approximation ratio. On the other hand, approximation algorithms are only known for restricted versions of the problem (e.g., [10]) and allow for a logarithmic approximation ratio; one of the reasons is that both the set of possible guard locations and the set that is to be covered have infinite cardinality, and no easy reduction to discrete sets is known. Thus, previous work mostly consists of heuristics [3] with good practical performance.

The work by Couto et al. [9, 8, 6] considers exact

*Supported by DFG Priority Programme "Algorithm Engineering" (SPP 1307), project "RoboRithmics" (Grant number Fe 407/14-1).

†Department of Computer Science, Technische Universität Braunschweig, D-38106 Braunschweig, Germany, Email: {t.baumgartner,s.fekete,a.kroeller,c.schmidt}@tu-bs.de. http://www.ibr.cs.tu-bs.de/alg/


(a)

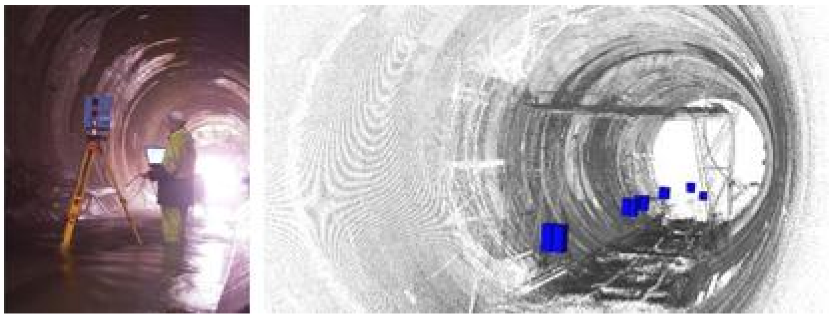

(b)

Figure 1: A 360°laser scanner of inmetris3D placed in a production hall (a) and in a tunnel with the resulting scan data (b). (All images curtesy of inmetris3D.)

solutions for a special case of the art gallery problem. Compared to the general problem as discussed here, their variant has two additional restrictions:

1. polygons are simple, i.e., have no holes, and
2. guards may only be placed on polygon vertices.

Their algorithms therefore benefit from a small and finite set of covering points in combination with various ways of reducing the candidate set of points that need to be covered. Their approach is based on a Set Cover integer program, applied to a grid discretization, which is iteratively refined if necessary. They are able to bound the number of iterations, but have to solve an instance of the set cover problem in each iteration. In experiments with polygons of up to 200 vertices and later [8], [6] with a variety of discretization strategies of up to 1000 and 2500 vertices they are able to solve these instances within ca. 110-1000 seconds (for the different strategies and 1000 vertices). Note that instance sizes and computation times are not comparable to our results, as we consider a more general problem.

Other authors have considered even more restricted versions of the problem; e.g., Keil [15] gave an $O(n^2)$ algorithm for computing an optimal solution in horizontally convex orthogonal polygons.

To the best of our knowledge, there is little work on good lower bounds. Amit et al. [3] are the only ones to consider the general art gallery problem, where the entire interior has to be guarded. They use the lower bound to compare heuristic solutions, with the guards taken from different candidate sets, like vertices and other significant points. Lower bounds for the version in which guards only need to cover the edges of the polygon are given by Bottino and Laurentini [4].

Our Contribution. We develop a primal-dual approach for general art gallery problems in arbitrary polygons with holes, in which guards can be placed anywhere, such that the entire interior of the polygon is guarded. Our method computes a sequence of lower and upper bounds on the optimal number of guards until—in case of convergence and integrality—eventually an optimal solution is reached. Our algorithm is based on a formulation of the problem as a (covering) linear program. It solves the problem using a cutting plane and column generation approach, i.e., by solving the primal and dual separation problems. Computational results show the usefulness of our method.

The rest of the paper is organized as follows. In Section 2 the main part of our algorithm, using linear programming, is presented, together with a discussion of geometric aspects of separation. Section 3 discusses implementation aspects of the algorithm and present heuristic ingredients. We evaluate our implementation using a set of test instances in Section 4. Problems of convergence for degenerate cases are discussed in Section 5. Finally, in Section 6 we discuss possible implications and extensions.

2 An LP-Based Procedure

Notation and Preliminaries. We consider a given polygon P, possibly with holes. For a point $p \in P$ the *visibility polygon* $\mathcal{V}(p)$ is the (star-shaped) set of all points of P visible from p. A *guard set* $G \subset P$ *covers* P if $\cup_{g \in G} \mathcal{V}(g) = P$.

For linear programming the *separation problem* is defined as follows: Given an instance of a linear programming problem, e.g., $\max\{c^T x \mid Ax \leq b\}$ and a point y, determine whether y belongs to the polyhedron $P = \{x \mid Ax \leq b\}$ or not, and in the latter case, find a violated constraint. Depending on whether we consider the primal or dual linear program, we speak of the *primal separation problem* or *dual separation problem.*

For linear programs with a large number of variables, i.e., columns, *column generation* is a finite and exact solution method. For a subset of the columns a restricted problem is solved. Using the dual solution to this problem it is possible to decide which additional column should be integrated in the primal problem.

LP Formulations. Our approach to the art gallery problem (AG) focuses on good lower bounds. It may also find upper bounds, i.e., feasible solutions, although this cannot be guaranteed.

AG can be trivially formulated as the linear problem with infinitely (actually uncountably) many binary variables and inequalities:

$$\min \sum_{g \in P} x_g \tag{2.1}$$

$$\sum_{g \in \mathcal{V}(w)} x_g \geq 1 \quad \forall w \in P \tag{2.2}$$

$$x_g \in \{0,1\} \qquad \forall g \in P \tag{2.3}$$

Here a guard placement is modelled by setting $x_g = 1$ iff $g \in P$ is a guard position. Inequality (2.2) ensures that the polygon is fully covered. We sometimes refer to a point $w \in P$ as a *witness*, with the interpretation that this point witnesses whether it is sufficiently covered or not. Note that the known upper bound of $\lfloor \frac{n}{3} \rfloor$ guarantees that the formulation is well-defined.

Due to its infinite size, the above formulation cannot be solved using linear programming techniques. Instead, we consider the relaxation AGR(G,W), in which we relax the integrality constraint (2.3), restrict the guard positions to be from a finite set $G \subset P$, and only require a finite set $W \subset P$ of witnesses to be covered. Throughout this paper, we assume that every witness $w \in W$ is visible from at least one guard position $g \in G$. This ensures feasibility of the formulations.

AGR(G,W) can be formulated as follows:

$$\min \sum_{g \in G} x_g \tag{2.4}$$

$$\sum_{g \in G \cap \mathcal{V}(w)} x_g \geq 1 \quad \forall w \in W \tag{2.5}$$

$$0 \leq x_g \leq 1 \qquad \forall g \in G \tag{2.6}$$

The associated dual linear program of (2.4)–(2.6) reads as follows:

$$\max \sum_{w \in W} y_w \tag{2.7}$$

$$\sum_{w \in W \cap \mathcal{V}(g)} y_w \leq 1 \quad \forall g \in G \tag{2.8}$$

$$0 \leq y_w \leq 1 \qquad \forall w \in W \tag{2.9}$$

The primal LP is a fractional covering problem using fractional guards, and the dual is a fractional packing problem.

An optimal solution to AGR(G,W) has no straightforward interpretation in terms of the original AG problem. It is neither a lower bound (as there may be points in $P \setminus W$ that are not sufficiently covered), nor an upper bound (as there may be guard positions in $P \setminus G$ that allow for covering with fewer guards). But fortunately some conclusions can be drawn. Let x^* be an optimal solution to (2.4)–(2.6) with associated optimal dual solution y^*. We use the notation $z(Q)$ to denote $\sum_{q \in Q} z_q$ for a set Q and a vector $(z_q)_{q \in Q}$. Consider the following:

1. The primal separation problem for AGR(G,W) is to identify a witness point $w \in W$ whose constraint is violated, i.e., $x^*(G \cap \mathcal{V}(w)) < 1$.
   (a) If such a point exists, then w solves the separation problem for AGR$(G, W \cup \{w'\})$ and therefore also AGR(P,P). This proves that x^* is not feasible for AGR(P,P).
   (b) If no such point exists, it proves that x^* is feasible and optimal for AGR(G,P). Should it also hold that x^* is integral, i.e., $x^*_g \in \{0,1\}$ $\forall g \in G$, it also defines a feasible solution for the original art gallery problem, i.e., an upper bound.
2. The dual separation problem is to identify a guard position $g \in G$ with $y^*(W \cap \mathcal{V}(g)) > 1$.
   (a) If this point exists, it solves the separation problem for AGR$(G \cup \{g'\}, W)$ and proves that x^* is not optimal for AGR(P,W).
   (b) If no such point exists, x^* is optimal for AGR(P,W). As AGR(P,W) is a relaxation of AGR(P,P), the objective value is a lower bound for it and also the original art gallery problem.
3. If both 1(b) and 2(b) hold, i.e., neither guard nor witness points can be generated, x^* is feasible and optimal for AGR(P,P).

Both separation problems can be solved by the following method: Construct the arrangement of the input polygon and every visibility polygon $\mathcal{V}(p)$ for points p with $x_p > 0$ (resp. $y_p > 0$ for the dual case). Now for every element, i.e., face, edge, or vertex of the arrangement all points of this element receive a constant amount of coverage. All that is necessary is to iterate over all elements to find one whose coverage is less than 1 (resp. more than 1).

The above observations translate directly into an algorithm, see Algorithm 2.1. It is trivial to see that, should the algorithm terminate, it will have found an optimal solution to AGR(P,P) and thus a lower bound for AG. Unfortunately, there are degenerate problem instances for which the algorithm will never terminate—see Section 5 for an example.

Algorithm 2.1. (LP procedure)

```
Generate initial $G \subset P$, $W \subset P$
repeat
    Solve AGR$(G, W)$
    Identify witness points for condition 1(a),
    add them to $W$
    Identify guard points for condition 2(a),
    add them to $G$
until both conditions 1(b) and 2(b) hold
return current objective value as lower bound for AG.
```

There are three steps where the algorithm behavior can be influenced: First, what should the initial G and W be? Guessing a hopefully good (or even optimal) solution to AG and using the guard positions for G can be a great speedup for the algorithm. Similarly, a clever choice of W can improve algorithm runtime significantly. Finally, the separation phases leave several options: Prioritizing separating guards produces more lower bounds, while prioritizing witness separation produces more upper bounds. All of this is of purely heuristic nature, and will be discussed in Section 3.

Correctness of the Algorithm.

Lemma 2.1. (Lower Bound) *In case algorithm 2.1 terminates, the returned objective value is a lower bound for AG.*

Proof. When algorithm 2.1 terminates, neither a witness point for condition 1(a) nor a guard point for condition 2(a) exists. So, inequalities 2.5 and 2.8 hold for $W \equiv P$ and $G \equiv P$. That is, the current solution x^* is feasible and optimal for AGR(P, P). AGR(P, P) is the relaxation for AG, the linear program formulated in (2.1)–(2.3). If we denote the optimal objective values for AGR(P, P) and AG by $opt(\text{AGR}(P, P))$ and $opt(AG)$, resp., we have: $opt(\text{AGR}(P, P)) \leq opt(AG)$. Consequently, the objective value $opt(\text{AGR}(P, P))$ of x^* is a lower bound for $opt(AG)$.

Lemma 2.2. (Upper Bound) *In case the algorithm outputs an upper bound, the returned value is an upper bound for AG.*

Proof. The algorithms claims to give an upper bound whenever condition 1(b) holds and the current solution x^* is integral. With the integrality, x^* is feasible for inequality 2.3. Moreover, condition 1(b) assures feasibility for inequality 2.2. Thus, x^* is a feasible solution for AG, and such it defines an upper bound for $opt(AG)$.

Lemma 2.3. (Optimality) *In case algorithm 2.1 terminates and the current solution x^* is integral, the returned objective value is the optimal value for AG.*

Proof. The current solution x^* is feasible and optimal for AGR(P, P). As AGR(P, P) is the relaxation of AG, an optimal integer solution for AGR(P, P) is a feasible solution for AG. We have $opt(\text{AGR}(P, P)) \leq opt(AG)$, hence, it is also optimal for AG.

Geometric Aspects of the Separation Problem.

Solving the primal and the dual separation problem means identifying a witness point $w \in W$ whose constraint is violated, i.e., $x(G \cap \mathcal{V}(w)) < 1$, or a guard position $g \in G$ whose constraint is violated, i.e., $y(W \cap \mathcal{V}(g)) > 1$. The coverage of a point in P depends on the visibility polygons of the guards (or witnesses) p with $x_p > 0$ ($y_p > 0$). Let $\mathcal{G}_s = \{p \in P : x_p > 0\}$, $g_s = |\mathcal{G}_s|$, $\mathcal{W}_s = \{p \in P : y_p > 0\}$ and $w_s = |\mathcal{W}_s|$.

This means that a crucial subroutine consists in computing visibility regions inside of the polygon P; an important complication arises from the fact that P may be non-simple polygon with h holes. This problem has received a considerable amount of attention: Suri and O'Rourke [18] were the first who gave an algorithm for computing a visibility polygon inside a polygon P that may have holes. Their algorithm performs an angular plane sweep and runs in $O(n \log n)$; at this point, we have chosen a similar approach for ease of implementation. This theoretical runtime can be improved by using a method by Heffernan and Mitchell [13] to $\Theta(n + h \log h)$.

For the primal and dual separation we use the same polygon P. Hence, we can use the query versions with one preprocessing step. For each separation we need to compute the arrangement of g_s (w_s for the dual separation) visibility polygons.

At this point we have not made use of preprocessing approaches, as the most powerful methods require a large amount of memory. For example, using the algorithm of Zarei and Ghodsi [21], we need $O(g_s(1 + h') \log n + \sum_{q \in \mathcal{G}_s} |\mathcal{V}(q)|)$ time to construct the arrangement for the primal separation, with $O(n^3 \log n)$ preprocessing and $O(n^3)$ space. Other options include the method of Pocchiola and Vegter [17], or by Inkulu and Kapoor [14]. These improvements are left for future work.

3 Implementation

In this section, we describe implementation aspects of our algorithm. The software will be released under the GNU General Public License on the project website[1].

Our software implements the LP-based procedure from Section 2. It uses standard linear programming libraries to solve the LPs. Currently SoPlex [20]

[1] `http://www.ibr.cs.tu-bs.de/projects/artgallery`

and ILOG CPLEX can be used for that matter, but generally any solver that provides an implementation of the primal-dual simplex algorithm could be included. Furthermore we employ CGAL [2] for visibility queries and the separation problems. We make heavy use of the `Arrangement_2` package [19] for both.

The arrangements use exact rational arithmetic to avoid numerical issues. To keep the computational overhead low, a point rounding heuristic is used. Every separated point will be perturbed to coordinates with smaller representation.

The implementation does not maintain the full overlay of all visibility polygons for G resp. W, as these tend to become complex very quickly. We have observed that solution times improve when a different approach is used. For every point in G (resp. W), we compute the visibility polygon and store it in a DCEL. In the separation routines, we then compute the overlay only for those points that have a positive value in the current solution. As the cardinality of this point set is usually much smaller than the whole variable set, the overlay is much simpler.

Finally, our implementation has limited support for multicore processors. We use the Boost Threads library to parallelize parts of the computation. Support is currently preliminary, essentially because GMP [1], the library providing rational arithmetic, is not threadsafe. It is obvious that many algorithmic steps can be done in parallel. Examples are the primal and dual separation routines, which operate on completely disjoint data structures, or visibility polygon computations.

3.1 Separation Strategies

The algorithmic framework does not specify how the separation problem should be solved. This is generally irrelevant for the theory of the algorithm, but not for the practice. We have implemented a number of different procedures for these problems.

Steering towards Bounds. Attempting to solve both the primal and the dual separation problem in each iteration is not neccessary. By adjusting which separator is used in an iteration, one can steer the algorithms towards lower or upper bounds. Though this steering is purely heuristic and cannot be guaranteed. Our implementation offers four steering strategies:

- The "*Both*" strategy always runs both primal and dual separators in each iteration. The idea is to reach an optimal solution quickly, without caring to find bounds before the final step.

- "*Primal*" focusses on finding upper bounds, i.e., solutions. It always runs the primal separation routine. Dual separation only happens if the primal fails. This is motivated by the idea to find solutions often, and gradually turn them into optimal ones.

- "*Dual*" is the counterpart of *Primal*. It favors dual separation over primal, in the hope of finding lower bounds quickly.

- "*Stay*" alternates between the two previous ones. It will start by only running primal separators, until this fails for the first time. A fractional solution is thus found. Then it heads for a lower bound by using dual separators only, until they fail. This is repeated, resulting in an alternating sequence of upper and lower bounds.

Point Separators. To solve the separation problems, we have implemented three different heuristics for each problem. Note that for the primal separation problem it is sufficient to consider only points in faces of the guard arrangement. This is because $x(G \cap \mathcal{V}(w))$ is constant for points w on such a face, and of same or higher value on the face's boundary. A similar argument holds for the dual separation problem. There it suffices to consider only vertices of the witness arrangement.

We provide the following separators. Two of them separate several points on each call, to avoid having to re-optimize the LP for every single new column or row.

- "*Bulk*" simply searches through the faces (vertices) of the guard (witness) arrangement to find points defining violated constraints. It returns an arbitrary selection of these points, with a user-provided limit on the number of points.

- "*Maximally Violated*" follows the principle to always separate using the maximally violated constraint, i.e., $\arg\min_{w \in P} x(G \cap \mathcal{V}(w))$ for the primal problem and $\arg\max_{g \in P} y(W \cap \mathcal{V}(g))$ for the dual.

- "*Greedy*" attempts to find just a few points, which ideally become independent rows (columns) with large support in the LP. We model dual separation as a Set Cover problem, where a minimal number violated guard points are used to cover the witnesses. The standard greedy algorithm is used to solve it. For the primal separation, we use an analogous procedure.

Vertex Guards. The implementation offers solving the problem variant where only vertex guards are allowed. This allows for direct comparison with the algorithm from [6]. It is implemented by another steering strategy, "*Primal-Only*". This strategy runs only the primal separator, thereby limiting the algorithm to the guard positions that are initially available.

4 Computational Results

We tested our implementation on a variety of different classes of polygons, including floor plans and random polygons already tested by Couto et al. [6]. Different strategies for the initial guard placement were used: We started with a single guard or initially located guards at all vertices of the given polygon. As mentioned in Section 2, a wise choice influences the behavior of the algorithm.

For all figures in the following we show solutions to the primal problem (guards) and to the dual problem (witnesses). The coloring of the polygons makes it easier to visually test for feasibility. For the primal problem gray denotes covering with a value of exactly 1, solutions with bluish coloring are feasible (they indicate a value > 1), solutions with a yellow-green coloring (indicating a value < 1) are not. Accordingly, for the dual problem gray denotes covering with a value of exactly 1 again, solutions with a yellow-green coloring are feasible (they indicate a value < 1), solutions with bluish coloring (indicating a value > 1) are not.

4.1 Examples. The first example (Figure 2) shows that switching from vertex guards to general guards may considerably improve the optimal value, at the expense of higher computational difficulty.

The second example (Figure 3) shows the floor plan of our institute, a non-orthogonal polygon with a single hole and a number of columns. We start with a greedily obtained set of witnesses (left), corresponding to an (infeasbile) dual solution; a corresponding set of guards is shown on the right. In iteration #3, the dual problem is feasible, but the primal problem still has a tiny region near the central column that is not sufficiently covered. Finally, in iteration #5 the primal and the dual problem are feasible, so we are in case 3 of Section 2 and have achieved an optimal feasible solution with 10 general guards.

A third example (Figure 4) is a random orthogonal polygon with 100 vertices. As shown on the left, an optimal solution of 15 vertex guards is found within four iterations. In the center is a solution with 13 general guards, found after 123 iterations; a corresponding proof of optimality by means of a matching dual solution is shown on the right of the figure.

4.2 Evaluation. For a thorough evaluation of our implementation, we solved a number of problem instances:

- The institute floor plan from Figure 3 with 94 vertices.
- Three sets of orthogonal simple polygons from [7], each consisting of 30 instances. See Figure 4 for one of them. They have 50, 70, and 100 vertices, respectively.
- Eight spike boxes with holes, such as the one depicted in Figure 2. Each has 60–68 vertices.

	Frac.	Int.
Random 50	100%	100%
Random 70	92%	92%
Random 100	90%	80%
Spike Boxes	100%	100%

Table 1: Overall optimality rates.

Each instance was run with different steering strategies and separators. The tests were conducted on a Linux PC with a 3.0 GHz CPU and 2 GB of RAM. We used SoPlex 1.4.1 and CGAL 3.4.

Each run was abort if it exceeded 1000 iterations or 20 minutes of CPU time. Therefore, some runs did not finish with an optimal fractional solution. In the remainder of this section, we report on the percentages of runs that lead to optimal fractional solution and to optimal integral solution. Furthermore we list how long it took to finish on average, not counting the runs that were aborted. Note that the algorithm can be interrupted earlier if optimal solution are not neccessary, as we produce lower and upper bound throughout runtime.

The default configuration uses an initial placement of one guard and witness per polygon vertex, the *Stay* strategy, and *Greedy* separators. This configuration is marked with an asterisk in table headers. Table 1 lists the percentage of runs in which any configuration could identify an optimal fractional solution ("Frac.") resp. optimal integral solution ("Int."). Clearly, these rates are very high. This is especially surprising for integral solutions, as this problem is not directly addressed by the algorithm. Tables 4, 5, and 7 show average algorithm runtime under different configurations.

Steering Strategies. We evaluated the influence of different steering strategies on the success rate (Table 3) and runtime (Table 4). Recall that *Primal Only* differs from the others, as it solves a different problem variant. We conclude that *Both* usually outperforms the other strategies. However, we observed that indeed *Dual* and *Primal* produce more bounds of their respective kind. Also *Stay* is almost as fast as *Both*, and it produces bounds of both kinds. Tables illustrating this behavior are omitted due to limited space.

Point Separators. Comparisons for choosing primal/dual separators different from the default *Greedy* can be found in Tables 2 and 5. Notice that they have little influence on whether the algorithm finishes in time; however, choosing the computationally more expensive *Greedy* speeds up algorithm runtime.

Initial Placement. Finally we compared the influence of the initial choice of guard and witness points. This is reported in Tables 6 and 7. In the runs marked "*All*", G and W were initialized with the full set of polygon vertices. In "*One*", just a polygon vertex was selected for G and W. The latter suffers from considerable reduction in both optimality rate and longer runtimes, as it completely shifts the task of exploring the polygon to the separation routines.

Vertex Guards. Using the *Primal Only* steering strategy lets the algorithm solve the vertex guard variant of the problem, in order to allow for a comparison with the algorithm in [6]. Note that there are two important differences: Our algorithm can deal with polygons with holes; on the other hand, [6] always produces integral solutions. Two conclusions can be drawn: Firstly, our algorithm has a speed well comparable to [6]. Secondly, as success rates increase and runtimes drop in this variant, we can see that the apparent loss of performance in the other case stems from the increased complexity of the problem, and not from implementation issues.

5 Problems of Convergence

For some polygons, some guards in an optimal solution for the art gallery problem *must* be located at a certain point or on a line segment, see the examples in Figure 5. This means that the measure of the solution in an appropriate high-dimensional space is zero; as there is no known simple description of these sets (in particular, they are not intersections of diagonals or edge extensions), it is not surprising that dealing with these situations is problematic.

Since our method is based on introducing guards (or witnesses) in insufficiently covered areas, we are able to identify a region around the guard's only possible position and shrink its size during the course of the algorithm. The region will shrink to a point after infinitely many iterations, but not within finite time.

Using a different strategy for the initial guard placement, we may obtain convergence towards optimal fractional solutions, but obviously this is not guaranteed. However, real-world instances do not show these problems; moreover, such solutions are undesirable, as they yield fragile solutions that may cause serious problems in scan registration.

For the polygon in Figure 5, left, and only one initial guard we have to deal with these problems of convergence, see Figure 7 In the solution to the primal problem after 407 iterations (see Figure 7, right) we can identify many guard positions inserted "around" the only feasible locations of the optimal solution, but still the primal and the dual problem are not feasible. On the other hand, for the same polygon we can find the optimal solution value, based on an optimal fractional solution, if we start with guards placed at every vertex.

6 Conclusion and Outlook

We have presented an LP-based procedure for obtaining bounds on the Art Gallery Problem that—in case of convergence and integrality—provides optimal solutions. Our algorithm is based on a formulation as a fractional covering problem and the corresponding dual fractional packing problem. For the separation problem we use properties of visibility polygons, i.e., information that is intrinsic to the art gallery problem but not to the underlying covering formulation.

There is a variety of directions for extending our work. As discussed above, improving the separation routine may lead to considerable speedup; however, this may come at the expense of tremendous memory requirements, so careful balancing will be necessary. Other extensions arise from fine-tuning the choice of guards and witnesses, making more extensive use of the intrinsic geometric structure of the underlying set cover instances.

Yet another improvement should come from combining geometry with polyhedral combinatorics, refining the polyhedral description in order to remove large classes of non-integral vertices. As shown in Figure 6, we have identified various classes of cutting planes that may turn out to be useful for quicker resolution of fractional vertices, at the expense of solving the corresponding separation problems (which are different from what we described in Section 2.)

Finally, discussions with our inmetris3D partners have revealed a variety of practical conditions and relaxations on guards and environment. In particular, there is a strong interest in fractional solutions instead of integral ones, making our LP-based approach even more powerful. We are optimistic that we can report on these and other improvements in the foreseeable future.

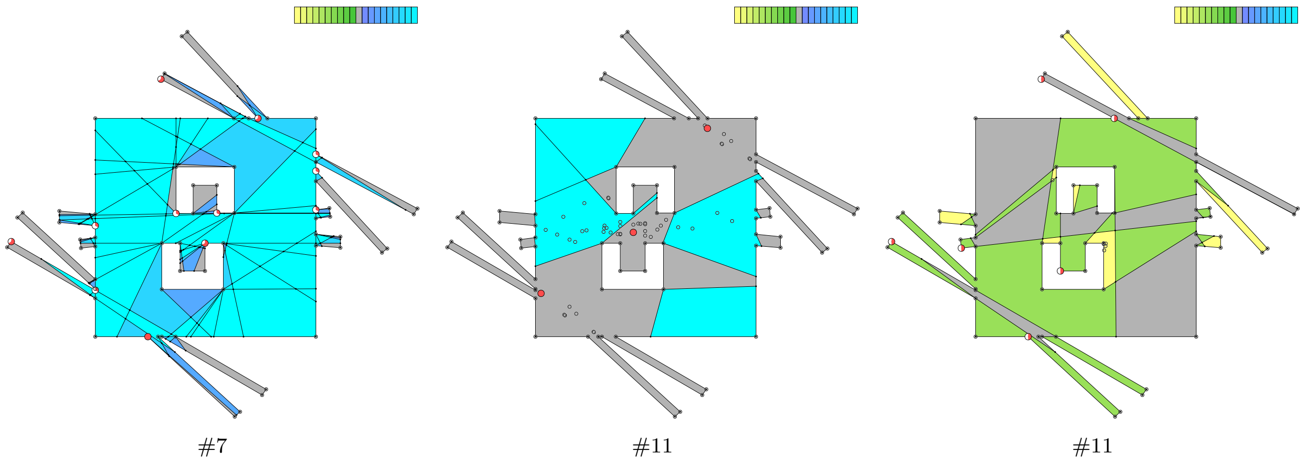


Figure 2: (Left) A lower bound of 7 on the number of vertex guards; (Center) a feasible solution with 3 general guards; (Right) a matching set of witnesses showing optimality of the general solution.

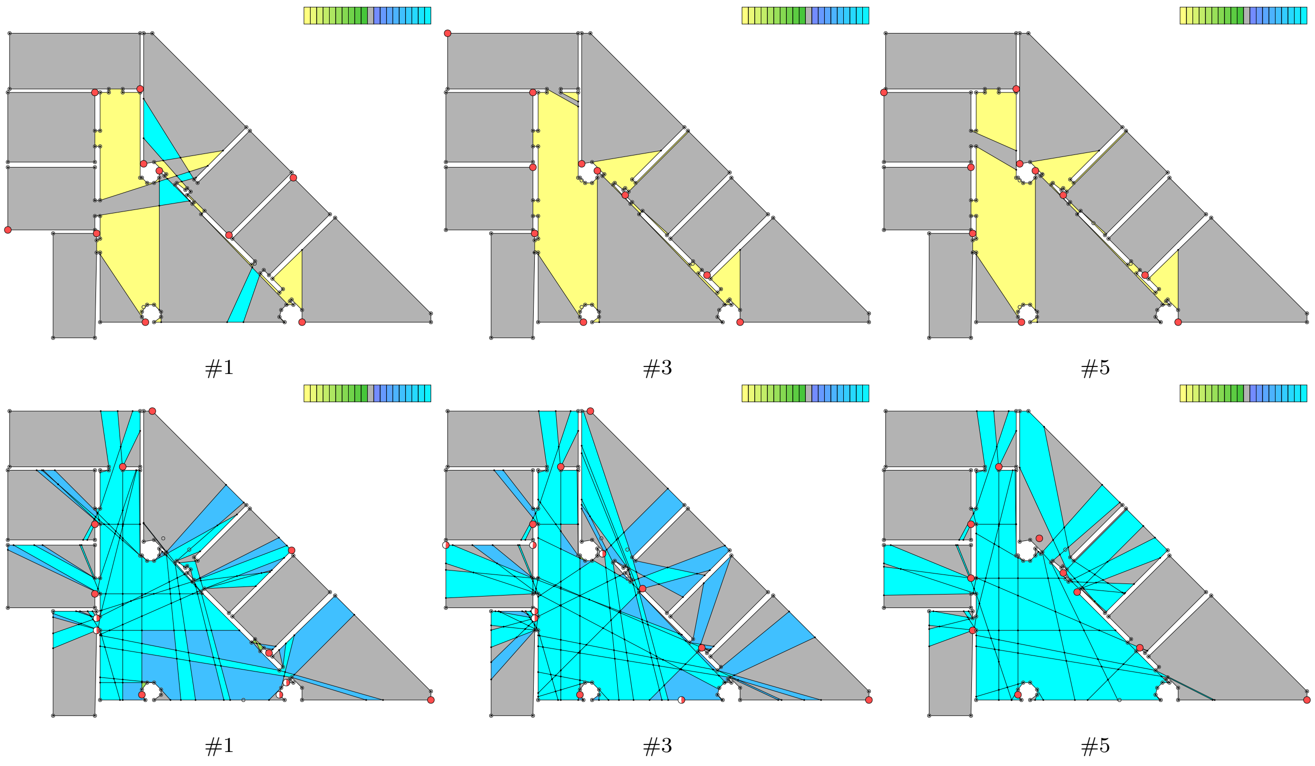


Figure 3: Floor plan of our institute; the top row shows witness sets, while the bottom row shows guard sets for iterations #1, #3, #5. Fractional weights are indicated by partially filled red dots. An optimal solution and a matching lower bound are found in the fifth iteration.

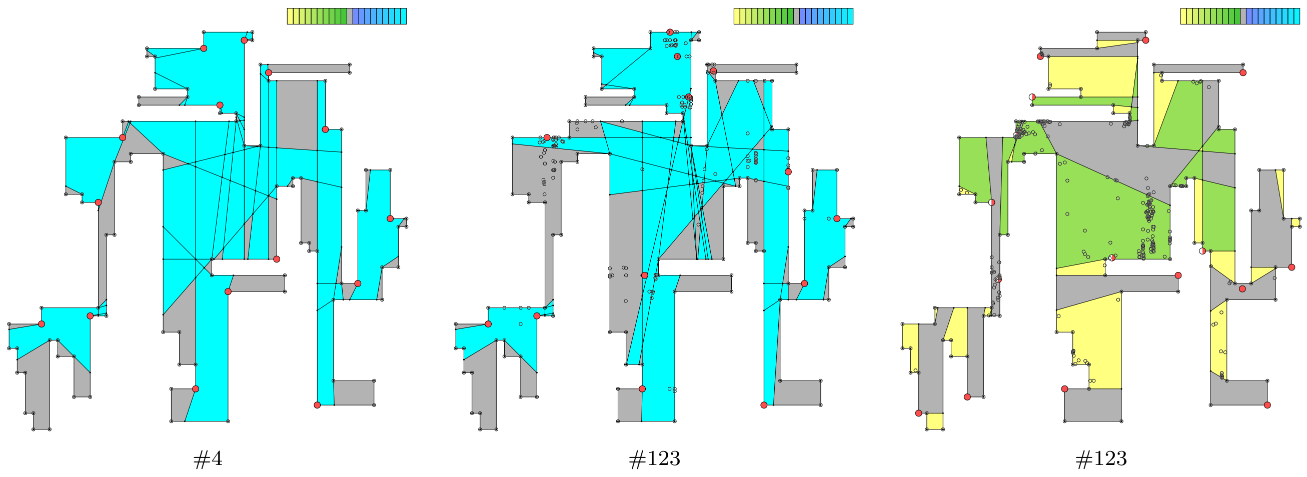


Figure 4: Solutions for a random polygon with 100 vertices: (Left) An optimal vertex solution consisting of 15 guards; (center) a general solution consisting of 13 guards; (right) a matching set of witnesses showing optimality of the general solution.

	Greedy*		Primal Bulk		Pri. Max Viol.		Dual Bulk		Dual Max Viol.	
	Frac.	Int.	Frac.	Int.	Frac.	Int.	Frac.	Int.	Frac.	Int.
Random 50	87%	80%	87%	82%	87%	83%	88%	78%	80%	73%
Random 70	87%	85%	88%	82%	88%	75%	90%	76%	75%	65%
Random 100	73%	48%	73%	55%	75%	50%	77%	43%	50%	40%
Spike Boxes	100%	75%	100%	69%	94%	75%	88%	63%	63%	38%

Table 2: Optimality rates for different separators: Percentage of runs leading to optimal fractional (“ Frac.”) or integral (“ Int.”) solution.

	Stay*		Both		Primal		Dual		Primal Only	
	Frac.	Int.	Frac.	Int.	Frac.	Int.	Frac.	Int.	Frac.	Int.
Random 50	87%	80%	90%	73%	63%	60%	87%	67%	100%	77%
Random 70	87%	85%	88%	62%	77%	77%	85%	62%	100%	69%
Random 100	73%	48%	80%	43%	37%	27%	67%	43%	100%	53%
Spike Boxes	100%	75%	100%	88%	63%	63%	100%	63%	100%	38%

Table 3: Optimality results for different steering strategies.

	Stay*	Both	Dual	Primal	Primal Only
Random 50	1.790	1.779	2.505	1.750	0.908
Random 70	2.570	2.686	4.021	2.891	1.930
Random 100	10.373	8.709	5.842	21.482	5.159
Spike Boxes	3.909	4.316	9.870	3.535	1.917
Floor Plan	4.385	4.330	4.860	6.920	4.280

Table 4: Runtimes for different steering strategies, in seconds.

	Greedy*	Primal Bulk	Pri. Max Viol.	Dual Bulk	Dual Max Viol.
Random 50	1.790	1.580	3.292	1.305	1.305
Random 70	2.570	2.355	2.649	2.628	3.192
Random 100	10.373	17.406	12.294	23.473	12.458
Spikes	3.909	5.711	4.393	3.958	12.023
Floor Plan	4.385	4.455	4.350	5.005	4.310

Table 5: Runtimes for different choices of separators, in seconds.

	All		One	
	Frac.	Int.	Frac.	Int.
Random 50	87%	80%	68%	63%
Random 70	87%	85%	63%	47%
Random 100	73%	48%	33%	23%
Spike Boxes	100%	75%	31%	25%

Table 6: Optimality for different G and W.

	All	One
Random 50	1.790	6.646
Random 70	2.570	8.981
Random 100	10.373	41.444
Spikes	3.909	115.183
Floor Plan	4.385	9.420

Table 7: Runtimes.

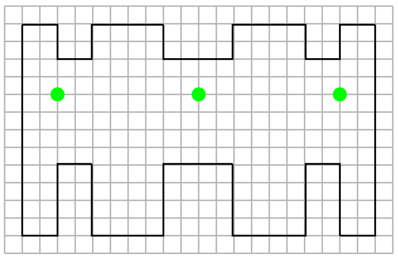
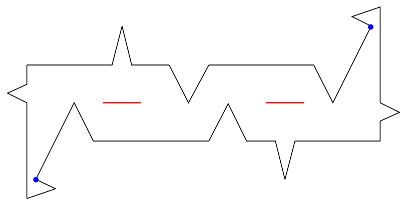



Figure 5: Left: For this polygon guards cannot be moved in any direction. Right: A polygon in which two guards in the set of an optimal guard cover need to be located on the red line segments.

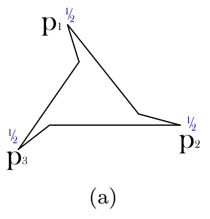
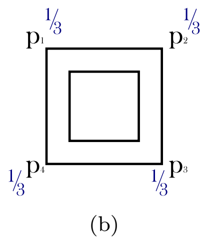
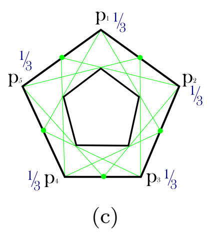
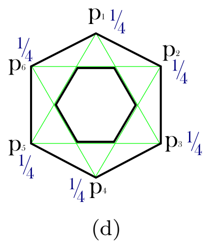
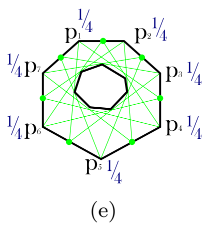






Figure 6: Examples of “facet”-defining inequalities for the art gallery “polytope”.

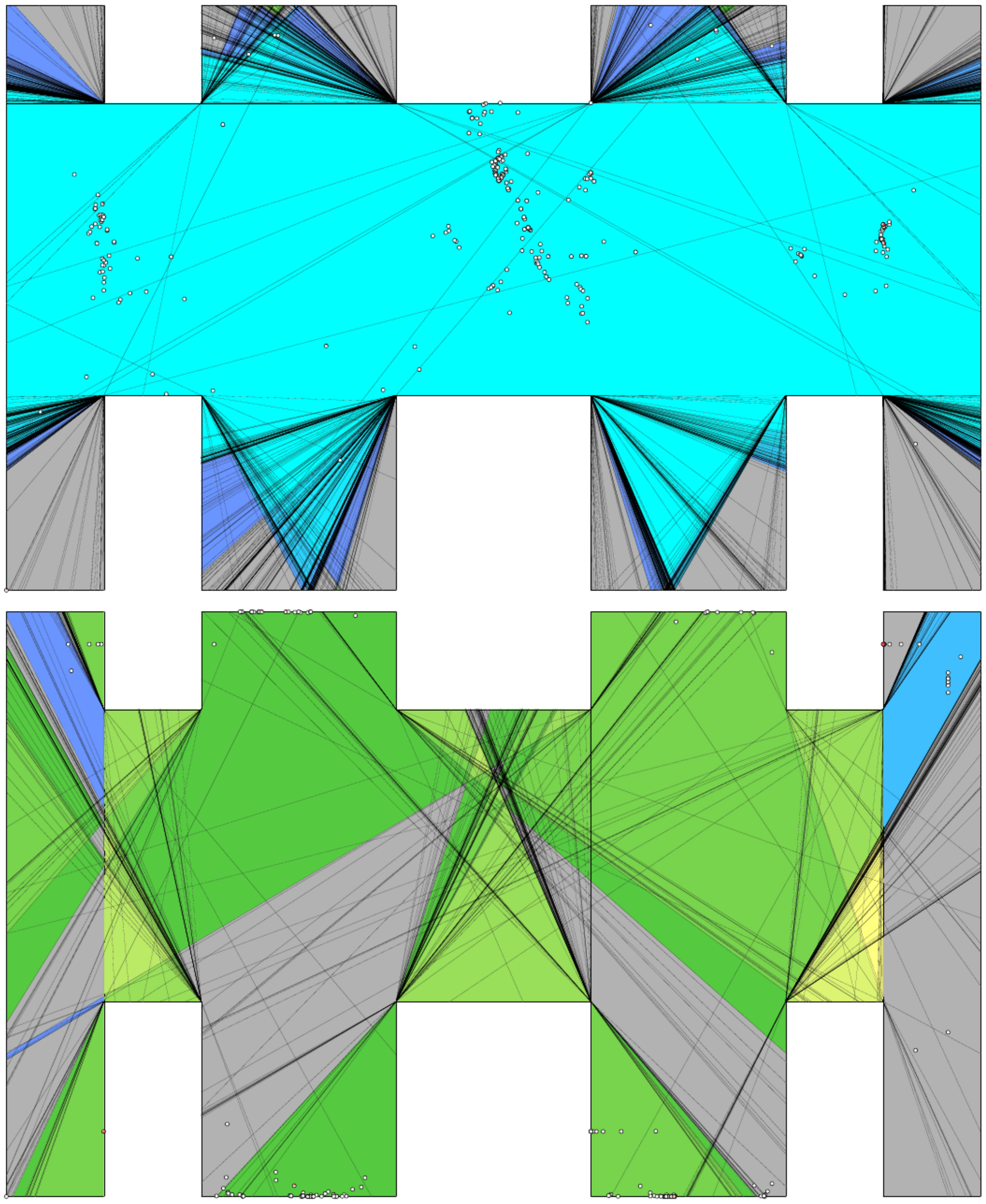


Figure 7: Worst-case problem instance with zero-dimensional optimal solution space. Top: The primal solution after over 400 iterations. (Gray denotes covering with a value of exactly 1, solutions with bluish coloring are feasible (value > 1), solutions with yellow-green coloring (< 1) are not.) Bottom: The corresponding dual solution.

References

[1] GNU Multiple Precision Arithmetic Library. http://gmplib.org.

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

[3] Yoav Amit, Joseph S. B. Mitchell, and Eli Packer. Locating guards for visibility coverage of polygons. In *ALENEX*, pages 120–134, 2007.

[4] Andrea Bottino and Aldo Laurentini. A nearly optimal sensor placement algorithm for boundary coverage. *Pattern Recognition*, 41(11):3343–3355, 2008.

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

[6] Marcelo C. Couto, Pedro J. de Rezende, and Cid C. de Souza. An exact algorithm for an art gallery problem. Technical Report IC-09-46, Institute of Computing, University of Campinas, November 2009.

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

[8] Marcelo C. Couto, Cid C. de Souza, and Pedro Jussieu de Rezende. Experimental evaluation of an exact algorithm for the orthogonal art gallery problem. In *WEA*, pages 101–113, 2008.

[9] Marcelo C. Couto, Cid C. de Souza, and Pedro J. de Rezende. An exact and efficient algorithm for the orthogonal art gallery problem. In *SIBGRAPI '07: Proceedings of the XX Brazilian Symposium on Computer Graphics and Image Processing*, pages 87–94, Washington, DC, USA, 2007. IEEE Computer Society.

[10] Alon Efrat and Sariel Har-Peled. Guarding galleries and terrains. *Inf. Process. Lett.*, 100(6):238–245, 2006.

[11] Stephan Eidenbenz, Christoph Stamm, and Peter Widmayer. Inapproximability results for guarding polygons and terrains. *Algorithmica*, 31(1):79–113, 2001.

[12] Steve Fisk. A short proof of Chvátal's watchman theorem. *Journal of Combinatorial Theory*, (B 24):374–375, 1978.

[13] Paul J. Heffernan and Joseph S. B. Mitchell. An optimal algorithm for computing visibility in the plane. *SIAM J. Comput.*, 24(1):184–201, 1995.

[14] Rajasekhar Inkulu and Sanjiv Kapoor. Visibility queries in a polygonal region. *Comput. Geom. Theory Appl.*, 42(9):852–864, 2009.

[15] J. Mark Keil. Minimally covering a horizontally convex orthogonal polygon. In *SoCG '86: Proceedings of the Second Annual Symposium on Computational Geometry*, pages 43–51. ACM, 1986.

[16] Der-Tsai Lee and Arthur K. Lin. Computational complexity of art gallery problems. *IEEE Trans. Inf. Theor.*, 32(2):276–282, 1986.

[17] Michel Pocchiola and Gert Vegter. The visibility complex. In *SoCG '93: Proceedings of the Ninth Annual Symposium on Computational Geometry*, pages 328–337, New York, NY, USA, 1993. ACM.

[18] Subhash Suri and Joseph O'Rourke. Worst-case optimal algorithms for constructing visibility polygons with holes. In *SoCG '86: Proceedings of the Second Annual Symposium on Computational geometry*, pages 14–23, New York, NY, USA, 1986. ACM.

[19] Ron Wein, Efi Fogel, Baruch Zukerman, and Dan Halperin. 2d arrangements. In CGAL Editorial Board, editor, *CGAL User and Reference Manual*. 3.4 edition, 2008.

[20] Roland Wunderling. *Paralleler und objektorientierter Simplex-Algorithmus*. PhD thesis, Technische Universität Berlin, 1996. `http://www.zib.de/Publications/abstracts/TR-96-09/`.

[21] Alireza Zarei and Mohammad Ghodsi. Efficient computation of query point visibility in polygons with holes. In *SoCG '05: Proceedings of the Twenty-First Annual Symposium on Computational Geometry*, pages 314–320, New York, NY, USA, 2005. ACM.