On Solving the Weighted Orthogonal Art Gallery Problem under regular grid discretization by greedy methods and their hybridizations

Abstract

In this paper...

1. Introduction

The Orthogonal Art Gallery Problem (OAGP) is one of many variants of Art Gallery Problem (AGP). AGP asks for a set of points G of minimal cardinality on some polygon P such that for each point $y \in P$ there is $x \in G$ such that $xy \subset P$. Set G is called guard set of P and the points from G as guards. The orthogonal AGP consider on arbitrary polygon but whose angles are $90°$ and $270°$. This problem was stated by Victor Klee in 1973. [7]. The problem is motivated from installing the cameras inside a building (or gallery) such that the whole area of the building is covered. Orthogonality constraint naturally comes out from the orthogonality of the walls in buildings.

Kahn et al. in [5] formulated and proofed that $\lfloor \frac{n}{4} \rfloor$ guards are sufficient to cover an orthogonal polygon with n vertices. A variant of the OAGP for which we are interested in this study allows only that guards are positioned at the vertices of polygon P. Actually, we are looking for minimum vertex guards needed to cover an orthogonal polygon. This restricted problem is known to be NP- hard in [8, 6].

By discretization $D(P)$ the set of points of the polygon P, the art gallery problem can be reduced to the well-known Min Set Cover Problem. FiXme: Marko: ovdje fali motivacija zasto radimo diskretizaciju poligona For each vertex of the polygon P, a set of points of visibility is determined. In that way, the problem of determining the minimum number of guards covering the entire polygon is reduced to determining the minimum number of subsets of points, such that each point from $D(P)$ is included in at least one of the chosen subsets.

In [2] an approximate solution of the minimum vertex guard problem, which can be computed in $O(n^4)$ time and the size of the solution is at most $O(\log n)$ times the optimal, is presented for simple polygon with n vertices. Firstly, this algorithm do a partitioning the polygonal region into convex components and construct sets consisting of these convex components. After that, on these constructed sets Johnson's approximation algorithm [4] for the Minimum set-covering problem (MSC) is applied to get solution.

An anytime algorithm to compute successively better approximations of the optimum to Minimum Vertex Guard is proposed in [9]. A major idea of this approach is exploring dominance of visibility regions to first detect pieces that are more difficult to guard. The same problem is solved in [10] by applying successive approximations from [9].

Tozoni et a. [11, 12] presented an exact Integer linear programming (ILP)-based algorithm, which iteratively generates upper and lower bounds through the resolution of discretized space of the AGP.

More detailed overview of the results on art gallery problems is out the scope of this paper and can be found in a survey paper [3].

Couto et al. presented an exact and efficient algorithm for the Orthogonal Art Gallery Problem in [1]. The algorithm is divided into two phases. In the preprocessing phase an initial discretization is constructed and the Integer Programming model for solving MSC is applied. In the solution phase, the discretized instance is iteratively refined and solved, until the solution becomes viable. FiXme: Marko: ja bih ovdje samo rekao: Couto et al. an efficient exact algorithm for OAGP based on preprocessing and refinement phases of the the discretized instance.

1.1. Problem definition

1.2. Main contributions

The main contributions of this paper are:

- We developed a novel greedy approach which is based on balancing the trade off between the total sum of guards' costs and the total number of not yet covered points from the discretization.
- We developed a specially designed shaking procedure, which perturbs the current partial solution in such a way that some vertices are replaced with others based on a distance measure. FiXme: TODO: popraviti
- Both an existing and the novel greedy algorithms are hybridized with the proposed shaking procedure and with the ILP
- We considered different types of weights for our benchmarks, based on a... FiXme: TODO

FiXme: TODO: work on literature approaches

1.3. Discretization

FiXme: TODO: Milan treba da ovo objasni proces transofrmacije, zasto uzimamo ovakav tip diskretizacije, kako se onda problem svodi na SC, da obajsni zasto diskretni grid ne mora dati potpunu pokrivenost poligona – dvije slike. Proces transfomracije objasniti kratko (mozda i pseudokod ako je moguce da se prikaze)...

2. Exact methods

FiXme: TODO: Marko

2.1. Integer linear programming method

Solvers which solves IP model for OAGP problem are one of the most efficient techniques to approach this problem. It is known that the problem is related to the known Minimal-Set-Cover (MSC) problem.

Let us suppose we are given a discretization $D(P)$ of the polygon P (with a family of rectangles). Then we relate the OAGP with the known MSC problem. Family $\mathcal{F} \subseteq D(P)$ of nonempty sets is given as: $S_i \in \mathcal{F}$ iff it includes any point from $D(P)$ which is visible from guard $i \in V$. Note that set S_i includes a point p_i which can also included by some other guard $j \in V$, $i \neq j$. So, the task of OAPG becomes finding a minimal cardinalty cover $\mathcal{C} \subseteq \{S_1, ..., S_n\}$ of set $D(P)$, that is

$$\bigcup_{c \in \mathcal{C}} c = D(P).$$

The IP model for the Set Cover problem is well-known, and based on it the following model

$$\sum_{i=1}^{n} x_i \longrightarrow \min \tag{1}$$

$$\text{s.t.} \tag{2}$$

$$\sum_{j \in V} a_{ij} x_j \geq 1 \ (\forall p_i \in D(P)) \tag{3}$$

$$x_j \in \{0, 1\}, j \in V, \tag{4}$$

where $a_{ij} = \begin{cases} 1, p_i \in V(j), \\ 0, \text{otherwise} \end{cases}$ and $x_i = \begin{cases} 1, \text{ if the point } i \in \mathcal{C}, \\ 0, \text{otherwise,} \end{cases}$

presents a MIP model to solve OAGP where $V(j)$ is the set of all points from $D(P)$ that are visible from j-th vertex of P. Set $Z = \{j \in V \mid x_i = 1\}$ represents a solution of the problem w.r.t. discrete relaxation D of the polygon. Constraint (3) enforces that any point $p_i \in D(P)$ will be visible from at least one guard from Z.

If we want to add weights into the OAGP problem such that all guards have no equal weight (of 1), we obtained the weighted variant of OAGP, labelled by WOAGP problem. Introducing prices into the basic problem can be augmented by the fact that the prices of cameras do not need to be equal and these prices might be different due to the quality (respecting the range of spectrum of view) of the cameras that are installed at some specific corners. Let us suppose that for any guard p_i, we assign a price $w_i > 0$, $i = 1, ..., n$. Then, the above model is may be extended to its weighted version where the objective function has to be replaced by

$$\sum_{i} x_i w_i,$$

and all other constraints remain the same.

In order to solve this model, we use a general purpose solver Cplex.

2.2. Constraint programming method

FiXme: TODO: Marko

Algorithm 1 Greedy Heuristic

Input: an instance of a problem
Output: A (feasible) non-expandable solution (or reporting that no feasible solution)
$s^P \leftarrow ()$ // partial solution set to empty solution
while Extend$(s^P) \neq \emptyset$ **do**
    Select component $e \in$ Extend(s^P) // w.r.t. some criterion
    Extend s^P by e
end while

3. Algorithmic Approaches for WOAGP

3.1. Greedy approaches for solving WOAGP

Since the WOAGP can be seen as the Weighted Set Cover problem (WSCP) and since the best known heuristics to solve WSCP is an enhanced greedy heuristic, our idea is to transform a WOAGP instance into a WSC problem instance and then apply greedy algorithms to solve WOAGP. These algorithms produce a solution of reasonable quality within a short interval of time. Efficiency of such greedy heuristics is related to a greedy criterion utilized to expand current (non-complete, i.e., partial) solution to complete one. Among all candidates (solution components for expansion, that is not-yet-considered guards, we choose one with the smallest greedy value until the solution is complete (i.e., cover all points from $D(P)$). A general pseudocode of Greedy heuristics is given in Algorithm 1

3.1.1. An existing greedy method

FiXme: TODO: Milana

Concerning the literature for WSC problem **??**, one of the most efficient greedy heuristic was based on the following greedy rule:

$$g(s^p, p_i) = \frac{w_{p_i}}{f(s^p \cup \{p_i\}) - f(s^p)}, \tag{5}$$

where $f(s^p) = |\bigcup_{s \in s^p} s|$. This heuristic is also a approximation algorithm which ensures $O(\log(n))$ approximation factor.

3.1.2. A Novel Greedy Heuristic

FiXme: TODO: Dragan In this subsection we present a novel greedy function, which is used in the greedy algorithm. Let us introduce a term "incorrect point". For a point from $D(P)$ we say that it is incorrect if it is not covered by any guard from a current partial solution s^{ps}. Let denote by $incorrect_{total}$ the total number of incorrect points from discretization $D(P)$ for the current partial solution. Let w_{total} be the total sum of all weights among all vertices and $|D(P)|$ be total number of discretization set. The greedy function for the given partial solution s^{ps} and the guard v takes into account both the value of the objective function of the WOAGP applied to the partial solution if v is included in it and the total number of incorrect points, given by the formula:

$$g(s^{ps}; v) = \alpha \cdot \sum_{i \in s^{ps} \cup \{v\}} w_i + \beta \cdot incorrect_{total} \tag{6}$$

where α and β are positive real parameters.

Although many combination of parameters α and β may be used, in our investigation we used parameter α only to normalize the sum of prices, i.e. $\alpha = \frac{1}{w_{total}}$ and two values for β: $\beta = 1$ and $\beta = \frac{1}{|D(P)|}$. Thus, we consider two variants of the greedy function:

$$g_2(s^{ps}; v) = \frac{\sum_{i \in s^{ps} \cup \{v\}} w_i}{w_{total}} + incorrect_{total} \tag{7}$$

and

$$g_3(s^{ps}; v) = \frac{\sum_{i \in s^{ps} \cup \{v\}} w_i}{w_{total}} + \frac{incorrect_{total}}{|D(P)|} \tag{8}$$

From Equation (7) one can see that the greedy function g_2 mainly depends on the second term, since the value of the first one is not greater than 1. On other words, this function prefers such guards which inclusion in the partial solution will cover more points. On the other side, if the inclusion of some two guards covers the same number of points, then the function suggest the guard with lower cost. In the early stage of the greedy algorithm this approach enables that the guards which covers more points are preferable, while in a later stage, when less points remain uncovered, the function prefers the guards with lower cost.

In the Equation (8) the second term is also normalized and the trade off between two terms is more balanced. In this case, the algorithm does not ultimately prefer any of criteria FiXme: TODO...

- start Greedy with empty solution
- in each iteration add such a guard to s^{ps} for which the obj value is minimal
- end when $incorrect_{total}$ becomes 0

Ties occurred in the search are broken by using price-per-unit heuristic which i stated as follows. For each not yet considered guards p_i, we assign the region which is visible from p_i by $S(p_j)$. As the next candidate to extend s^P, we choose a guard p^* which covers set $V(j)$ by smallest price per unit, among the other candidate to extend s^P. More precisely, greedy criterion g is given as:

$$g(s^p, p_i) = \frac{w_{p_i}}{S_{p_i}}, \tag{9}$$

and p^* that minimizes $g(s^p, p^*)$ value is chosen as an extension of the current partial solution. In our experimental studies, we found out that this heuristic does not perform well on its own, but presents a reasonable tie-breaking mechanism and is able to boost quality of our greedy heuristics.

3.1.3. Partial calculation of objective function

3.1.4. A hybrid of the Greedy and shaking

FiXme: TODO: Ubaciti pricu o shaking $-->$ Dragan?

3.1.5. A Hybrid of the Greedy + CPLEX

CPLEX will soon or later degrades its performance w.r.t. instance size. On the other hand, the later stage of Greedy increase the chance to worsen the final greedy solution. So it makes sense to combine partial solutions generated by Greedy over a few interactions and then to use CPLEX up to completion of the partial solution. Our approach consists of the following steps:

1. Run a Greedy method up to K iterations (parameter) to obtain a partial solution C (therefore, $|C| = K$);
2. Take solution C and make it complete via. CPLEX model.
   - CPLEX solves corresponding sub-model which is formed by adding constraints $x_{p_i} = 1$, for all $p_i \in C$ into the existing WOAGP model;
   - we obtain a complete solution C';
3. return $f(C')$.

4. Computational Results

We used the instance of a specific OAGP and assigned the weights to each vertex of polygons. We have generated three kind of benchmarks:

- *random benchmarks.* For each vertex i of polygon P we take a random value $w_i \in \{X_1, ..., X_q\}$ as its weight (X_i are some random values for prices), $q \in \mathbb{N}$.
- *topologically-based benchmarks.* For each vertex i of polygon P let us denote by l_i and l_{i+1} the lengths of edges that comes out of vertex i. Then, $w_i := \frac{l_i + l_{i+1}}{2}$. This can be augmented by the fact that if the the arithmetic length of both edges that comes out of vertex i is longer, it is expected that vertex is a guard can see a larger pieces of polygon P. This implies that the range of camera i has to be larger, which again means that it has to be of a higher price.
- *distance-based benchmarks:* For each vertex i, a weight is assigned to i by making use of the information about the distance from that point to the point in the polygon which is at the longest distance but visible from i (among all other points in polygon).
- *point-based benchmarks:* For each vertex j, we are looking for the number of points in D(P) that are visible from the given vertex ($|V(j)|$). Based on this measure, we assign prices to the vertices. If —V(j)— is larger, the large is the price of the vertex j.

5. Conclusions and Future Work

[1] M. C. Couto, C. C. De Souza, and P. J. De Rezende. An exact and efficient algorithm for the orthogonal art gallery problem. In *XX Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 2007)*, pages 87–94. IEEE, 2007.

[2] S. K. Ghosh. Approximation algorithms for art gallery problems in polygons. *Discrete Applied Mathematics*, 158(6):718–722, 2010.

[3] S. K. Ghosh. Approximation algorithms for art gallery problems in polygons and terrains. In *International Workshop on Algorithms and Computation*, pages 21–34. Springer, 2010.

[4] D. S. Johnson. Approximation algorithms for combinatorial problems. *Journal of computer and system sciences*, 9(3):256–278, 1974.

[5] J. Kahn, M. Klawe, and D. Kleitman. Traditional galleries require fewer watchmen. *SIAM Journal on Algebraic Discrete Methods*, 4(2):194–206, 1983.

[6] M. J. Katz and G. S. Roisman. On guarding the vertices of rectilinear domains. *Computational Geometry*, 39(3):219–228, 2008.

[7] J. O'rourke. *Art gallery theorems and algorithms*, volume 57. Oxford University Press Oxford, 1987.

[8] D. Schuchardt and H.-D. Hecker. Two np-hard art-gallery problems for ortho-polygons. *Mathematical Logic Quarterly*, 41(2):261–267, 1995.

[9] A. P. Tomás, A. L. Bajuelos, and F. Marques. Approximation algorithms to minimum vertex cover problems on polygons and terrains. In *International Conference on Computational Science*, pages 869–878. Springer, 2003.

[10] A. P. Tomás, A. L. Bajuelos, and F. Marques. On visibility problems in the plane–solving minimum vertex guard problems by successive approximations. In *ISAIM*, 2006.

[11] D. C. Tozoni, P. J. de Rezende, and C. C. de Souza. A practical iterative algorithm for the art gallery problem using integer linear programming. *Optimization Online*, pages 1–21, 2013.

[12] D. C. Tozoni, P. J. D. Rezende, and C. C. D. Souza. Algorithm 966: a practical iterative algorithm for the art gallery problem using integer linear programming. *ACM Transactions on Mathematical Software (TOMS)*, 43(2):1–27, 2016.