Can greedy-like heuristics be useful for solving the Weighted Orthogonal Art Gallery Problem under regular grid discretization?

Milan Predojević[a], Marko Djukanović[a], Milana Grbić[a], Dragan Matić[a]

[a] *Faculty of Natural Science and Mathematics, University of Banja Luka, Bosnia and Herzegovina*

Abstract

In this paper we deal with the Weighted Orthogonal Art Gallery Problem under the regular grid discretization. We propose 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. This new approach and an existing greedy algorithm are further hybridized with the Integer Linear programming (ILP), originally formulated for the well known Minimum Set Cover problem. Experimental results show that the proposed greedy methods is able to achieve the optimal solutions in most cases for a class of large area polygons, while for small area polygons, they achieve solutions of reasonable quality within lower runtime than the exact algorithms.

1. Introduction

Given a polygon P, the *Art Gallery Problem* (AGP) asks for a set of points G of minimal cardinality, such that for each point $y \in P$ there is $x \in G$ such that $xy \subset P$. We say that the point y is covered by the point x, or y is visible from x. Set G is called *guard set* of P and the points from G as *guards*. In the *Orthogonal Art Gallery Problem* (OAGP) we suppose that edges of the polygon are only horizontal and vertical w.r.t. the axes, i.e. the angles allowed between adjacent edges are 90° or 270°. The original AGP was initially stated by Victor Klee in 1973. [13]. 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. [8] formulated and proofed that $\lfloor \frac{n}{4} \rfloor$ guards are sufficient to cover an orthogonal polygon with n vertices. In the course of this study, we are interested in the variant of the OAGP which allows only that guards are positioned at the vertices of polygon P. This restricted problem is known to be $\mathcal{NP}$–hard [9, 16]. When it comes to real situations (like installing the cameras in a building), it is justified to assume that the prices of cameras are not equal and may depends on several factors, like the quality of a camera (respecting its range of spectrum of view) or installation price at some specific parts of the building (like corners or tight places). In the *Weighted Orthogonal Art Gallery Problem* (WOAGP) the task is to place guards on some vertices of orthogonal polygon P which cover all points from P, such that the total sum of prices assigned to the chosen vertices is minimal. In this paper we

consider WOAGP problem under the regular grid discretization of P, described in Section 1.2.

It is well known that AGP can be reduced to *the Minimum Set Cover Problem* (MSCP) by a discretization of the set of all points of polygon P. The appropriate discretization should be performed in such a way that if each point from the discretized set $D(P)$ is covered, then the whole polygon P is covered. After the discretization is made, for each vertex of polygon P, a set of visible points from $D(P)$ 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, which is MSCP. Analogously, WOAGP can be reduced to the *Minimum Weighted Set Cover Problem* (MWSCP).

Concerning the exact and heuristic techniques to solve OAGP, Couto et al. [4] presented an exact and efficient algorithm for the OAGP based on preprocessing and refinement phases of the discretized instance. In [5] an approximate solution of the minimum vertex guard problem, which can be computed in $O(n^4)$ time and this solution is at most $O(\log n)$ times the optimal one. After that, on these constructed sets Johnson's approximation algorithm [7] for the MSCP is applied. An anytime algorithm to compute successively better approximations of the optimum for Minimum Vertex Guard is proposed in [18]. A major idea of this approach is exploring dominance of visibility regions to first detect pieces of the polygon that are more difficult to guard. The same problem is solved in [19] by applying successive approximations from [18]. Tozoni et al. [20, 21] 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. Although many variants AGP are present in literature, WOAGP has not been so intensively studied, which motivated us to consider this problem. A comprehensive analysis of various greedy-like heuristics for the MWSCP was presented in [22]. More detailed overview of the extensive literature regarding MSCP and AGP is out of the scope of this paper and for further reading we suggest review papers [2, 14, 6].

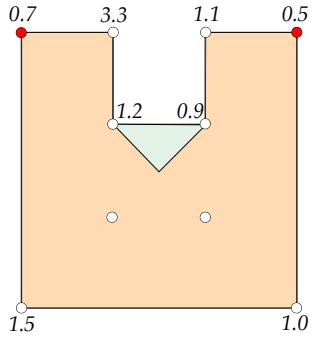
1.1. Main contributions

The main contributions of this paper are:

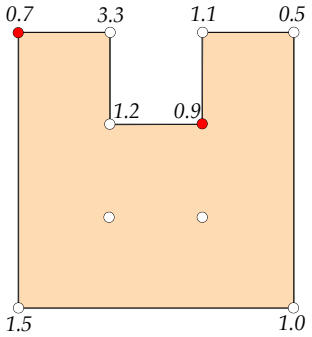
- We developed a novel greedy approach which is based on balancing the trade off between the total sum of guards' costs and the total number of not yet covered points from the discretization.
- The greedy algorithm from [3] and the novel greedy algorithm are hybridized with the ILP.
- We considered different types of weights for our benchmarks, based on an approximation of the costs in real situations.
- In a comprehensive computational experiment, we tested, analysed and checked the efficiency of the developed algorithms. The methods are then compared to the exact approaches ILP and *Constraint Programming* (CP) w.r.t. the quality of obtained heuristic solutions as well as runtimes.

Algorithm 1 Discretization $D(P)$ of polygon P

1: **Input:** The set of vertices V of polygon P
2: **Output:** The discretization $D(P)$ of polygon P
3: $BB \leftarrow bounding\_box(P)$;
4: $\triangle_x, \triangle_y \leftarrow resolution(P)$;
5: $D(BB) \leftarrow regular\_grid(BB, \triangle_x, \triangle_y)$;
6: $D(P) \leftarrow D(BB) \cap P$;
7: $D(P) \leftarrow D(P) \cup V$;



(a) Optimal coverage of discretization $D(P)$.



(b) Optimal coverage of polygon P.

Figure 1: Optimal coverage of discretization $D(P)$ does not imply a coverage of polygon P.

1.2. The Regular Grid Discretization of Polygon

Building of the regular grid discretization $D(P)$ of the polygon P is shown in Algorithm 1. At the beginning, the minimum bounding rectangle (usually called *bounding box*) of P is created. Then, the regular grid, with resolution $\triangle_x \times \triangle_y$ and starting at the lower left corner of the bounding box of polygon P, is defined as follows:

$$\triangle_x = \min\{|u_x - v_x| \mid u_x \neq v_x\} \text{ and } \triangle_y = \min\{|u_y - v_y| \mid u_y \neq v_y\}, \quad (1)$$

where $(u_x, u_y), (v_x, v_y)$ are adjacent vertices of polygon P.

The algorithm further forms the whole regular grid of the bounding box (line 5 of the pseudocode) and finally, all points of the regular grid that belong to P and all vertices of P are added into the discrete set $D(P)$.

It should be noticed that the optimal solution of WOAGP on $D(P)$ (i.e. an optimal covering of $D(P)$) is not necessarily an optimal covering of P. In the simple example shown in Figure 1, one can see that the guards placed on top left and top right vertices (shown in red color) cover all points from $D(P)$ (Figure 1 (a)), but not the whole polygon, since the gray triangle is not visible from any of these two vertices. The optimal solution value in this case is 1.2. Two red vertices in Figure 1 (b) optimally cover the whole polygon, but this covering is not optimal w.r.t. discrete set $D(P)$, since the total sum of weights is larger (1.6).

The chosen resolution $\triangle_x \times \triangle_y$ is a compromise between the algorithm speed and accuracy. More precisely, higher resolution increases the accuracy, but also

increases the execution time of Algorithm 1 and vice versa, lower resolution decreases the accuracy, but discretization is generated faster.

2. Exact methods

In this section we present the exact ILP, initially developed for MWSCP from [23] and *Constraint Programming* (CP) models for solving WOAGP under regular grid discretization, which are used in the rest of the paper.

2.1. Integer linear programming model

Let us suppose we are given a polygon P with weights assigned to vertices and the discretization $D(P)$ of P. The task we consider is covering all points from $D(P)$ by some vertices $V = \{v_1, ..., v_n\}$ of P such that the sum of their weights is minimized. The problem is related to the known MWSCP as follows. Family $\mathcal{F}$ of nonempty sets consists of the sets $S_i \in \mathcal{F}$ which include points $p \in D(P)$ that are visible from guard $v_i \in V$. For each set S_i, the cost $c(S_i) = w_i$ is assigned. In this way, our starting task is equivalent of finding a covering $\mathcal{C} \subseteq \{S_1, ..., S_n\}$ of the set of points $D(P)$ of minimum weight, that is

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

such that

$$f(\mathcal{C}) = \sum_{S_i \in \mathcal{C}} c(S_i)$$

is minimized.

The ILP model for the MWSCP [23] is adapted to WOAGP as follows:

$$\sum_{i=1}^{n} w_i x_i \longrightarrow \min \tag{2}$$

s.t.

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

$$x_j \in \{0, 1\}, \ j \in \{1, 2, \ldots, n\}, \tag{4}$$

where $a_{ij} = \begin{cases} 1, p_i \in S_j, \\ 0, \text{otherwise.} \end{cases}$

Set $Z = \{v_j \in V \mid x_j = 1\}$ represents a solution of the problem w.r.t. discretization $D(P)$ of polygon P. Constraint (3) enforces that any point $p_i \in D(P)$ is visible from at least one guard from Z.

In order to solve this model, we apply a general purpose solver CPLEX [11].

Algorithm 2 Greedy Heuristic

1: **Input:** an instance of a problem
2: **Output:** A (feasible) non-expandable solution s^{ps} (or reporting that no feasible solution exists)
3: $s^{ps} \leftarrow ()$ // partial solution set to empty solution
4: **while** Extend(s^{ps}) $\neq \emptyset$ **do**
5: Select solution component $e \in$ Extend(s^{ps}) w.r.t. some criterion g
6: Extend s^{ps} by e
7: **end while**

2.2. Constraint programming model

An equivalent CP model was implemented and tested by *IBM ILOG CP Optimizer* [10]. In this case, Constraint (3) is transformed into

$$\bigvee_{j=1}^{n} (a_{ij} \wedge x_j) = 1, \forall p_i \in D(P), \tag{5}$$

whereas the other constraints and the objective function are the same as in the above ILP model. Note that CP approach works in a branch-and-bound manner employing a constraint propagation and variable domain filtering [15].

3. Algorithmic Approaches for solving WOAGP

In this section we present the heuristic greedy approaches for solving the WOAGP and their hybridization with the exact ILP method.

3.1. Greedy approaches for solving WOAGP

Greedy algorithms produce a solution of reasonable quality within a short interval of time and are, in essence, easy to implement. Efficiency of such heuristic is related to a greedy criterion utilized to expand current (non-complete, i.e., partial) solution up to its completion. In order to extend current partial solution, among all candidates (solution components for expansion, that is not-yet-considered guards), we choose one with the smallest greedy value and add it to the current solution. This procedure keeps repeating until the current solution becomes complete (i.e., all points from $D(P)$ are covered).

A general pseudocode of Greedy heuristics is given in Algorithm 2. The set Extend(s^{ps}) includes those guards from set $V \setminus s^{ps}$ that cover some points from $D(P)$ that are not yet covered by s^{ps}. Solution component of the problem is a guard. Extension of partial solution s^{ps} by a vertex v corresponds to adding v into set s^{ps}.

3.1.1. An existing greedy method

Concerning the greedy heuristic from the literature for solving MWSCP [3, 12], one of the most efficient greedy heuristics was based on the following criterion:

$$g_1(s^{ps}, v_i) = \frac{w_i}{h(s^{ps} \cup \{v_i\}) - h(s^{ps})}, \tag{6}$$

where

$$h(s^{ps}) = \left| \bigcup_{v_i \in s^{ps}} S_i \right|. \tag{7}$$

This heuristic also ensures an approximation with $O(\log(n))$ approximation factor.

3.1.2. A Novel Greedy Heuristic

In this subsection we present a novel greedy criterion. First, we 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 the current partial solution s^{ps}. Let us denote by $incorrect_{total}(s^{ps})$ the total number of incorrect points from discretization $D(P)$ w.r.t. s^{ps}. Note that

$$incorrect_{total}(s^{ps}) = |D(P)| - h(s^{ps}), \tag{8}$$

where $h(s^{ps})$ is defined in (7) and $|D(P)|$ is the cardinality of discretization set. Let w_{total} be the total sum of all weights among all vertices. The greedy function w.r.t. partial solution s^{ps} and candidate guard v is defined as follows.

$$g_2(s^{ps}, v) = \frac{\sum_{i \in s^{ps} \cup \{v\}} w_i}{w_{total}} + \frac{incorrect_{total}(s^{ps} \cup v)}{|D(P)|}. \tag{9}$$

In the Equation (9) both terms are normalized in order to achieve better balance between the total sum of guards' costs and the total number of not yet covered points from the discretization. The algorithm does not ultimately prefer any of criteria for choosing next vertex. Although it cannot be a rule, it is justified to suppose that the second term has a more influence on the choice of the next vertex in earlier phases of the algorithm execution. At the beginning, more points are uncovered and the algorithm chooses such vertices which cover larger area of the polygon. At the ending stage of the algorithm's execution, it could be expected that the first term plays more significant role, since the total number of uncovered points is small.

Ties occurred in the search are broken by using price-per-unit heuristic which is stated as follows. For each not yet considered vertex v_i, we denote the region of polygon P that is visible from v_i by $Surf(v_i)$. As the next candidate to extend s^{ps}, we choose such a guard, with the smallest ratio between the price and the visible surface area. More precisely, this criterion is given as:

$$g'(s^{ps}, v_i) = \frac{w_i}{|Surf(v_i)|}, \tag{10}$$

where $|Surf(v_i)|$ represents the area of surface $Surf(v_i)$. In our experimental studies, we found out that this heuristic does not perform well on its own, but it represents a reasonable tie-breaking mechanism which boosts quality of the aforementioned greedy heuristics.

3.1.3. Partial calculation of greedy functions

In order to enable fast calculation of greedy functions described in previous subsections, we noticed that it could be useful to allow fast updating of the number of uncovered vertices. Therefore, we introduced two useful structures in our calculations:

- structure `numberOfGuards` – which is a map structure, where each point from $D(P)$ is mapped to the number of guards in solution which cover that point;
- structure `CoveredPoints` – as a set structure which keeps the points from $D(P)$ that are covered by the partial solution;

When adding vertex v into partial solution, we update current `CoveredPoints` and `numberOfGuards` by considering only new points from $D(P)$ covered by v. When v is removed from partial solution, we update current `CoveredPoints` and `numberOfGuards` by omitting all points which are covered only by v. These two functions allow also a fast calculation of functions (7) and (8): a candidate vertex v is temporarily added to the partial solution, new status is checked and then being removed from the solution.

3.2. A Hybrid of the Greedy *and* Cplex

The performance of Cplex sooner or later degrades w.r.t. instance size due to the complexity of the problem. On the other hand, in the later stage, there is an increased chance for Greedy to worsen the obtained greedy solution due to an increased number of guards which are similar w.r.t. greedy value. So, it makes sense to combine partial solutions generated with a Greedy procedure over a few iterations and only then to make use the Cplex to do the completion of the partial solution. In details, our approach consists of the following steps:

1. Run a Greedy method up to K iterations to obtain a partial solution s^{ps}, where K is a parameter of the algorithm;
2. Take solution s^{ps} and make it complete by solving a corresponding sub-model via Cplex:
   - Cplex solves corresponding sub-model which is formed by adding constraints $x_i = 1$, for all $v_i \in s^{ps}$ into the existing ILP model (2)–(4);
   - a complete solution $\overline{s}$ is obtained;
3. Return $obj(\overline{s}) = |\overline{s}|$.

This method is, therefore, called Greedy+Cplex.

4. Computational Results

We used two different kind of benchmark sets from [1, 19]:

- *MinArea* instances: include polygons with small areas and tiny interior. They present a lower boundary case for the cardinality of set $D(P)$ which means that, beside the polygon vertices, $D(P)$ contains only a few additional points.
- *FAT* instances: include polygons of large areas and wide interior. In this case, a large number of points of regular grid is included into $D(P)$.

For each benchmark set – *MinArea* and *FAT* instances – one n-ogon for each $n \in \{8, 10, ..., 200\}$ has been considered and the appropriate discretization is made, which makes 97 instances per each benchmark set. In overall, we have 194 instances. Discretization is performed by using CGAL library, version 5.0.2 [17].

We included two kinds of weights into each of 194 instances:

- *point-based related weights* ($W0$): For each vertex v_i, the assignment of prices is based on the number of points in $D(P)$ that are visible from v_i, as follows:
$$w_i = n \cdot \frac{|S_i|}{|D(P)|},\ i = 1, ..., n. \tag{11}$$
Introducing these weights can be justified by the assumption that cameras which cover larger range of points should be of a higher spectrum and quality, and thus more expensive.

- *topologically-based related weights* ($W1$): For each vertex v_i of polygon P we denote by l_i and l_{i+1} the lengths of two edges that come out of the vertex v_i (note that indices are taken modulo n). Then, we set
$$w_i = \frac{l_i + l_{i+1}}{2},\ i = 1, ..., n. \tag{12}$$
Similarly as in the case of $W0$, if the arithmetic length of both edges that comes out of vertex v_i is longer, it is expected that a guard can see a larger pieces of polygon P. This implies that the spectrum of camera installed at v_i has to be larger, which again implies that it should be of a higher quality, i.e,. a higher price.

4.1. Settings and the choice of the Parameters

All variants of our algorithms were implemented in C++ with g++ 7.4 compiler and the experiments were conducted in single-threaded mode on a machine with an Intel Xeon E5-2640 processor with 2.40 GHz and a memory limit of 8GB. The maximum computation time of each of our algorithms was set to 5 *min*. For solving ILP and CP models, CPLEX version 12.7 was used.

After conducting preliminary results, for GREEDY + CPLEX we decided to set up $K = \lceil 0.05 \cdot n \rceil$.

The benchmark sets and the executable file of our software for this project are provided at a public git repository, available on https://github.com/milanagrbic/WOAGP

4.2. Results and Discussion

The following algorithms are included in our computation:

- exact approaches: CP and ILP approach. The latter is, henceforth, called CPLEX;
- four heuristic approaches:
  - two pure greedy methods, guided by g_1 and g_2, labeled as GREEDY-1, GREEDY-2, respectively;
  - two GREEDY+CPLEX variants, henceforth labeled by GREEDY-1+CPLEX and GREEDY-2+CPLEX.

Algorithm	$\overline{obj}$	$\overline{t}[s]$	$\overline{\lvert g\rvert}$	#optHit	$\sigma(Opt)$	#hitsMinH	$\sigma(MinH)$
CPLEX	*15.93	0.152	18.93	97/97	0.00	N/A	N/A
CP	16.04	98.215	18.96	67/97	0.02	N/A	N/A
GREEDY-1	22.63	**0.002**	26.28	0/97	0.79	3/97	7.61
GREEDY-2	19.97	**0.002**	20.14	0/97	0.44	0/97	0.20
GREEDY-1+CPLEX	18.49	0.073	21.19	**2/97**	0.30	**57/97**	**0.06**
GREEDY-2+CPLEX	**18.39**	0.004	**19.59**	0/97	**0.26**	53/97	0.04

Table 1: The results on the benchmark set $MinArea$ for the type of weight $W0$.

Algorithm	$\overline{obj}$	$\overline{t}[s]$	$\overline{\lvert g\rvert}$	#optHit	$\sigma(Opt)$	#hitsMinH	$\sigma(MinH)$
CPLEX	*36.03	0.003	18.93	97/97	0.00	N/A	N/A
CP	*36.03	18.953	18.93	97/97	0.00	N/A	N/A
GREEDY-1	39.28	**0.001**	20.15	2/97	0.35	4/97	0.24
GREEDY-2	40.92	**0.001**	20.96	0/97	0.51	0/97	0.40
GREEDY-1+CPLEX	37.78	0.006	19.41	**4/97**	0.21	19/97	**0.08**
GREEDY-2+CPLEX	**37.31**	0.003	**19.58**	2/97	**0.15**	**81/97**	0.19

Table 2: The results on the benchmark set $MinArea$ for the type of weight $W1$.

Summarized numerical results for each of 6 variants of our algorithms are displayed in Tables 1 – 4. For each table, the average results are reported for each subset of instances grouped w.r.t. a specific kind of instances and a specific weight (97 instances per each group). Each of the tables list the tested algorithms in the first column. Starting with column two, each of the lines provide detailed statistics for respective algorithm. The statistics report: the average solution quality ($\overline{obj}$), the average time ($\overline{t}[s]$) in seconds, the average number of guards which are included in each of the solutions ($\overline{|g|}$), the number of solutions which match to the optimum solutions (#optHit), average standard deviation of the obtained solutions w.r.t. optimal solutions ($\sigma(Opt)$), then the number of minimal solutions the algorithm obtained w.r.t. the four heuristics achieved by the respective algorithm (#hitsMinH) and average standard deviation w.r.t. best (minimal) solutions achieved by the four heuristic algorithms ($\sigma(MinH)$). An asterisk in front of a number means that the exact approach could prove optimality in all cases. Best values among heuristic approaches in each column are bolded.

From the numerical results, we observe the following conclusions concerning exact solving:

- CPLEX could solve all solution to prove optimality within a fraction of a second;
- CP was able to solve all those instances with weight $W1$ but with signifi-

Algorithm	$\overline{obj}$	$\overline{t}[s]$	$\overline{\lvert g\rvert}$	#optHit	$\sigma(Opt)$	#hitsMinH	$\sigma(MinH)$
CPLEX	*2.62	0.02	7.32	97/97	0.00	N/A	N/A
CP	*2.62	8.17	7.32	97/97	0.00	N/A	N/A
GREEDY-1	3.39	0.35	7.81	0/97	0.09	23/97	0.05
GREEDY-2	3.52	**0.32**	6.61	2/97	0.10	**28/97**	0.08
GREEDY-1+CPLEX	3.34	0.35	8.91	4/97	**0.08**	25/97	**0.04**
GREEDY-2+CPLEX	**3.32**	0.33	**6.81**	**6/97**	**0.08**	25/97	**0.04**

Table 3: The results on the benchmark set FAT for the type of weight $W0$.

Algorithm	$\overline{obj}$	$\overline{t}[s]$	$\overline{\lvert g \rvert}$	#optHit	$\sigma(Opt)$	#hitsMinH	$\sigma(MinH)$
CPLEX	*6.72	0.02	6.44	97/97	0.00	N/A	N/A
CP	*6.72	0.36	6.44	97/97	0.00	N/A	N/A
GREEDY-1	6.77	**0.31**	6.49	92/97	0.02	92/97	0.04
GREEDY-2	6.77	**0.31**	6.49	92/97	0.02	92/97	0.04
GREEDY-1+CPLEX	**6.73**	0.32	**6.45**	**96/97**	**0.01**	**96/97**	**0.03**
GREEDY-2+CPLEX	7.42	**0.31**	6.46	90/97	0.33	92/97	0.33

Table 4: The results on the benchmark set FAT for the type of weight $W1$.

cantly more time than the time required for CPLEX;

- For the subset of instances $MinArea$ which includes weight $W0$, the performance of CP degrades significantly, which can be seen by the respective average runtime and the number of proven optimal solutions found (67/97).

From the numerical results, we observe the following conclusions concerning heuristic solving:

- In case of the instances that include small–area polygons ($MinArea$) and weight $W0$ (Table 1): A novel GREEDY-2 was able to outperform the GREEDY-1 from literature. In this case, the best performing heuristic algorithms is hybrid GREEDY-2+CPLEX which needs the two order of magnitude lower average runtime to finish than the pure greedy methods. The average runtimes of all four heuristic methods are an order of magnitude lower than the average runtime of CPLEX. Pure greedy methods deliver solutions of reasonable quality within 20% of the quality of optimal solutions but needs the time which is 7-8 times lower then the average runtime of CPLEX. The overall average execution times of all tested algorithms are displayed in Figure 4. Note that vertical axis is scaled logarithmically. Although all heuristic methods could not achieve optimal solutions (with the exception of two cases for GREEDY-1+CPLEX heuristic), the average standard deviation w.r.t. optimal solution is rather small. From last two columns, it is evident that both hybrid variants are more successful, hitting more than a half best solution each, with very small average standard deviation w.r.t. optimal solutions.

- In case of the instances that include small-area polygons and weight $W1$ (Table 2): the best heuristic approach is GREEDY-2+CPLEX which is able to deliver solutions which are within 3% of the optimum ones. Note that in 81 cases (out of 97) GREEDY-2+CPLEX is able to deliver equally good or better results that the other heuristic approaches. As it was the case of the weight $W0$, heuristic methods could not achieve many optimal solutions, but the obtained average standard deviation w.r.t. optimal solutions is rather small. The results of pure greedy method GREEDY-1 outperform the results of GREEDY-2. These two methods are effective since they deliver solutions which are within 10% of the CPLEX results but need 3 times lower runtime. The results of GREEDY-1 match just in two cases to the optimum.

- In case of the instances that include large-area polygons (FAT) and weight $W0$ (Table 3): the exact CPLEX approach is a clear winner concerning av-

erage solutions' quality as well as the average runtimes; the best heuristic algorithm w.r.t. solution quality is GREEDY-2 +CPLEX. It also matches in more cases (on 6 instances) to the optimum result then the other heuristic approaches. The obtained average results are within 22% of the average of optimal solutions. The pure GREEDY-1 slightly outperforms the pure greedy GREEDY-2. It is interesting that GREEDY-1 assigns in average a higher number of guards in solutions when compared to the average number of guards of GREEDY-2. The runtimes of our heuristic approaches are an order of magnitude lower than the time of CPLEX approach. Reason for that can be seen in the used regular discretization of the polygon. For the polygons with a larger area, more points are involved in the discrete set $D(P)$ and, therefore, iterations of greedy methods take a longer time than in case when $D(P)$ is smaller (which was the case of *MinArea* polygons).

- In case of the instances that include large-area polygons (FAT) and weight $W1$ (Table (4): the best heuristic algorithm w.r.t. solution quality is GREEDY-1+CPLEX which is able to match in 96 instances the quality of optimal solutions. Slightly worse results are delivered by GREEDY-1 and GREEDY-2. The obtained (heuristic) solutions of these two approaches are within 1% of optimal solutions and they are able to reach the quality of the optimal solution for 92 instances. Unfortunately, the runtimes in comparison to the runtimes of CPLEX are significantly higher for all of our heuristic approaches. However, the average runtime for CP is a bit higher that the avg. runtimes of the heuristic approaches. Again, average standard deviation for each heuristic method is rather small w.r.t. optimal solutions, indicating good quality of the proposed algorithms.

- Concerning the percentage of covering of polygons for the best solutions (found by CPLEX) we noticed that FAT instances are covered in almost all cases (see Fig. 2 and 3 and the blue curve). Concerning the *MinArea* instances, it gets harder to cover all polygon and in allmost all cases the whole area of polygon P cannot be covered. For these instances, we see that the solutions cover more regions of small–area polygons when weight $W1$ is considered then when considering $W0$ ($\approx 93\%$ vs. $\approx 85\%$).

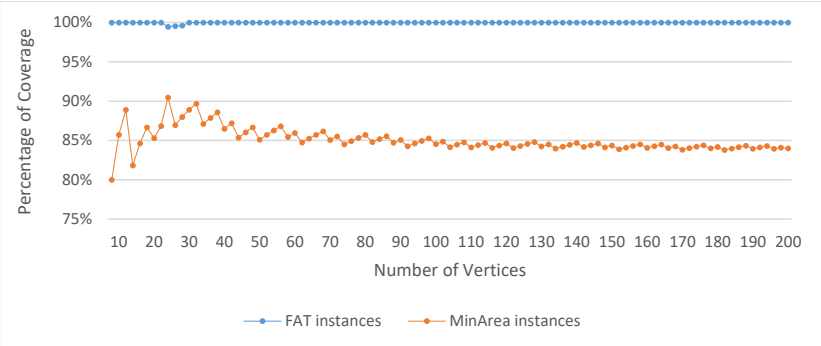


Figure 2: Polygon coverage for the type of weight W0.

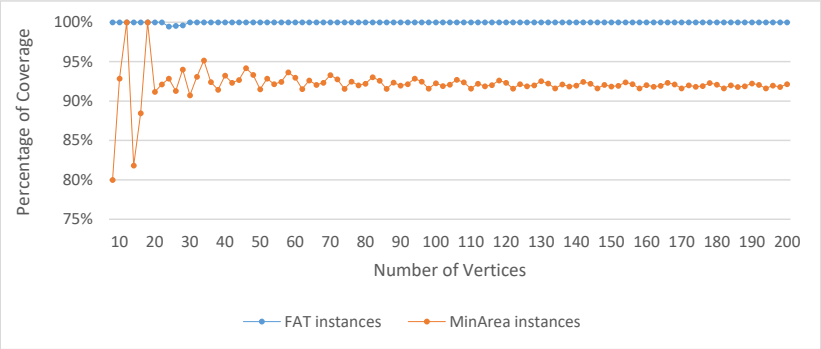


Figure 3: Polygon coverage for the type of weight W1.

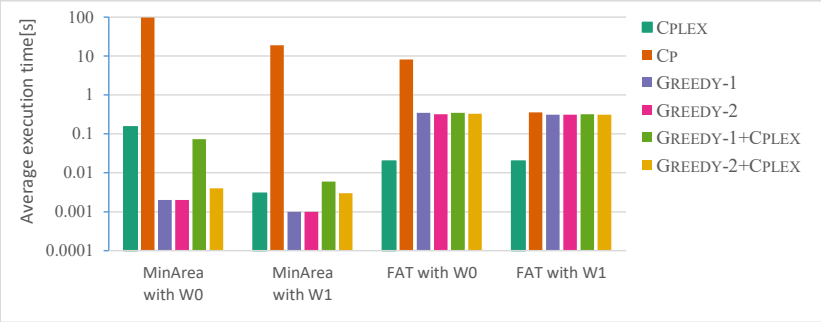


Figure 4: Avg. execution time of the compared algorithms.

5. Conclusions and Future Work

In this work we have considered the Weighted Orthogonal Art Gallery Problem under regular grid discretization. We have developed a novel greedy criterion which provide a trade-off between the number of guards and the cost of guards. Moreover, a hybrid of a greedy method and CPLEX was proposed to make a maximal completion of the greedy solution obtained by the greedy method supplemented by solving respective subproblem via CPLEX. The performances of the heuristic approaches are compared to the exact ILP and CP approaches. From the computational experiments, the heuristic approaches were highly efficient in terms of obtaining solutions of reasonable quality in an order of magnitude lower runtime than the exact approaches for the small-area polygons. For the large–area polygons, heuristic approaches were able to reach the quality of optimal solutions in almost all cases but in cost of larger times than the times of the ILP approach.

For the future work, we tend to improve the results of our greedy method as well as runtimes on the FAT benchmark sets by considering other types of polygon discretisations.

Acknowledgements

This research is partially supported by Ministry for Scientific and Technological Development, Higher Education and Information Society, Government

of Republic of Srpska, B&H under the Project "Development of artificial intelligence methods for solving computer biology problems".

References

[1] A. L. Bajuelos, A. P. Tomás, and F. Marques. Partitioning orthogonal polygons by extension of all edges incident to reflex vertices: Lower and upper bounds on the number of pieces. In *Procceding of ICCSA 2004 – Computational Science and Its Applications*, pages 127–136, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[2] A. Caprara, P. Toth, and M. Fischetti. Algorithms for the set covering problem. *Annals of Operations Research*, 98(1-4):353–371, 2000.

[3] V. Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of operations research*, 4(3):233–235, 1979.

[4] M. C. Couto, C. C. De Souza, and P. J. De Rezende. An exact and efficient algorithm for the orthogonal art gallery problem. In *Proceeding of SIBGRAPI 2007 – The 20th Brazilian Symposium on Computer Graphics and Image Processing*, pages 87–94. IEEE, 2007.

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

[6] S. K. Ghosh. Approximation algorithms for art gallery problems in polygons and terrains. In *Proceedings of WALCOM 2010 – The 4th International Workshop on Algorithms and Computation*, pages 21–34. Springer, 2010.

[7] D. S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9(3):256–278, 1974.

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

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

[10] P. Laborie, J. Rogerie, P. Shaw, and P. Vilím. IBM ILOG CP optimizer for scheduling. *Constraints*, 23(2):210–250, 2018.

[11] R. Lima and E. Seminar. IBM ILOG CPLEX – What is inside of the box? In *Proc. of 2010 EWO Seminar*, pages 1–72, 2010.

[12] L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete mathematics*, 13(4):383–390, 1975.

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

[14] Z.-G. Ren, Z.-R. Feng, L.-J. Ke, and Z.-J. Zhang. New ideas for applying ant colony optimization to the set covering problem. *Computers & Industrial Engineering*, 58(4):774–784, 2010.

[15] F. Rossi, P. Van Beek, and T. Walsh. *Handbook of constraint programming*. Elsevier, 2006.

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

[17] The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 5.1 edition, 2020.

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

[19] A. P. Tomás, A. L. Bajuelos, and F. Marques. On visibility problems in the plane–solving minimum vertex guard problems by successive approximations. In *Proceedings of ISIAM 2006 – The 9th International Symposium on Artificial Intelligence and Mathematics*, 2006.

[20] 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.

[21] 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.

[22] F. J. Vasko, Y. Lu, and K. Zyma. What is the best greedy-like heuristic for the weighted set covering problem? *Operations Research Letters*, 44(3):366–369, 2016.

[23] V. V. Vazirani. *Approximation algorithms*. Springer Science & Business Media, 2013.