Minimum Vertex Guard problem for orthogonal polygons: a genetic approach

ANTONIO L. BAJUELOS [*]
Universidade de Aveiro
Departamento de Matematica & CEOC
3810-193 Aveiro, PORTUGAL
leslie@ua.pt

SANTIAGO CANALES [†]
Universidad Pontificia Comillas de Madrid
Escuela Técnica Superior de Ingeniería, ICAI
28015 Madrid, SPAIN
scanales@dmc.icai.upcomillas.es

GREGORIO HERNÁNDEZ [†]
Universidad Politécnica de Madrid
Facultad de Informática
28660 Madrid, SPAIN
gregorio@fi.upm.es

ANA MAFALDA MARTINS [‡]
Universidade de Aveiro
Departamento de Matematica & CEOC
3810-193 Aveiro, PORTUGAL
mafalda.martins@ua.pt

Abstract: The problem of minimizing the number of guards placed on vertices needed to guard a given simple polygon (MINIMUM VERTEX GUARD problem) is NP-hard. This computational complexity opens two lines of investigation: the development of algorithms that determine approximate solutions and the determination of optimal solutions for special classes of simple polygons. In this paper we follow the first line of investigation proposing an approximation algorithm based on the general metaheuristic Genetic Algorithms to solve the MINIMUM VERTEX GUARD problem.

Key–Words: Art Gallery Problems, Orthogonal Polygons, Metaheuristics, Genetic Algorithms

1 Introduction

The Art Gallery Problems are well-studied visibility problems in Computational Geometry. The *original art gallery* problem was introduced by Victor Klee, in 1973, when he established the following problem: How many stationary guards are needed to cover an art gallery room with n walls? Informally the floor plan of the art gallery room is modeled by a *simple polygon* (simple closed polygon with its interior) P and a guard is considered a fixed point in P with 2π range visibility. We say that a point x *sees* point y (or y is visible to x) if the line segment connecting them does not intersect the exterior of P. A set of guards covers P, if each point of P is visible by at least one guard. Thus, the art gallery problem deals with setting a minimal number of guards in a gallery room whose floor plan has polygonal shape, so that they could see every point in the room. Two years later Chvátal established the well known *Art Gallery Theorem*: $\lfloor \frac{n}{3} \rfloor$ *guards are occasionally necessary and always sufficient to cover a simple polygon of* n *vertices*. Numerous variations of the original problem have been considered and studied over the years, such as: where must the guards be positioned (anywhere or in specific positions, e.g., on vertices) what kind of guards are to be used (e.g., stationary guards versus mobile guards) and what assumptions are on the input polygon (such as being orthogonal) (see [14]). An interesting variant is the *Orthogonal Art Gallery Theorem*. This theorem was first formulated and proved by Kahn et al, in 1983. It states that $\lfloor \frac{n}{4} \rfloor$ *guards are occasionally necessary and always sufficient to cover an orthogonal simple polygon of* n *vertices*. *Orthogonal simple polygons* (simple polygons whose edges meet at right angles) are an important subclass of polygons. Indeed, they are useful as approximations to polygons; and they arise naturally in domains dominated by Cartesian coordinates, such as raster graphics, VLSI design, or architecture. Efficient algorithms, based on the proofs of the above theorems, were developed to cover both arbitrary and orthogonal simple polygons with $\lfloor \frac{n}{3} \rfloor$ and $\lfloor \frac{n}{4} \rfloor$ guards, respectively. While this number of guards is necessary in some cases, often it is far more than it is needed to cover a particular simple polygon. For instance, it is known that convex polygons only require one guard. A variant of this

[*]Supported by CEOC through *Programa* POCTI, FACT, cofinanced by EC fund FEDER

[†]Supported by grant MEC-HP2005-0137

[‡]Supported by CEOC through *Programa* POCTI, FCT, cofinanced by EC fund FEDER and supported by a FCT fellowship, grant SFRH/BD/19138/2004

problem is the MINIMUM VERTEX GUARD (MVG) problem, that is the problem of finding the minimum number of guards placed on vertices (*vertex-guards*) needed to cover a given simple polygon. This is a NP-hard problem both for arbitrary and orthogonal simple polygons [8, 12].

Our contribution. The computational complexity of the MVG problem for simple polygons opens two lines of investigation: the development of algorithms that determine approximate solutions and the determination of optimal solutions for special classes of simple polygons. In this paper we follow the first line of investigation proposing an approximation algorithm based on the general metaheuristic Genetic Algorithms to solve the MVG problem on simple polygons. It is important to note that in spite of the proposed algorithm can be applied to any kind of polygon, in this work we only present experimental results on orthogonal simple polygons. Our implementation has been developed using the CGAL library [5]. The paper is structured as follows: in the next section we introduce some preliminary definitions and useful results. In section 3 we present a strategy, based on the general metaheuristic Genetic Algorithms, to solve the MVG problem on simple polygons. Section 4 is devoted to present our experiments and results on orthogonal polygons. Finally, in section 5 we draw conclusions and future work.

2 Preliminaries

Let P be a simple polygon with n vertices, $v_0, v_1, \ldots, v_{n-1}$. As in this paper we only deal with simple polygons, we use the term polygon to refer to a simple polygon. A vertex of P is called reflex if the interior angle between its two incident edges is at least π, otherwise it is called convex. We use r to represent the number of reflex vertices of P. It has been shown by O'Rourke that $n = 2r + 4$, for every orthogonal simple polygon of n vertices. We assume, without loss of generality, that the vertices of P are ordered in a counterclockwise direction around the interior of P. For $p \in P$, we call *visibility polygon* of p, $Vis(p)$, the set of all points $q \in P$ that are visible to p , i.e., $Vis(p) = \{q \in P : p \text{ sees } q\}$. We say that G, a given a set of vertices of P, is a *vertex-guard set* of P if they cover P, i.e., if $\bigcup_{v \in G} Vis(v) = P$. We denote by $|G|$ the cardinality of a vertex-guard set.

As stated before, the MVG problem is NP-hard for polygons and a way deal with this computational complexity is to develop approximation algorithms to tackle the problem. In a general way, these algorithms can be designed specifically to solve the problem (e.g., greedy constructive strategies) or can be based on general metaheuristics. A metaheuristic is a general algorithmic framework which can be applied to different optimization problems with relatively few modifications to make them adapted to a specific problem. For a comprehensive survey on metaheuristics see, e.g., [4, 7].

There are various works where approximation algorithms (heuristics) were developed to solve the MINIMUM SET GUARD (MSG) problem (e.g., [6], [2], [13] and [9]). In recent works, metaheuristics techniques have proven to behave very well in solving the MSG problem [1] and the MAXIMUM HIDDEN VERTEX SET problem [3] , which are, also, NP-hard visibility problems. For that reason, in this paper we propose an algorithm based on the metaheuristic Genetic Algorithms (GA) for computing a small vertex-guard set for a given polygon P.

3 Approximation Algorithm Based on GA

Genetic Algorithms are technics that simulate the processes of the natural evolution (biological). To solve an optimization problem with the GA metaheuristic it is necessary to specify the following components: a genetic representation of the possible solutions, called *individuals* or *chromosomes*, to the problem (*Encoding*); a way of creating an initial population of possible solutions (*Initial Population*); a function to evaluate the individuals and make the effect of natural selection, sorting solutions according to their "strength" (*Objective or Fitness function*); genetic operators to alter the composition of the solutions (*Selection*, *Crossover* and *Mutation*) and the values of various parameters used by the genetic algorithm (e.g., population's size, probability of the genetic operators, population's evaluation, population's generation, termination condition).

In the next subsection we describe these components designed for our problem.

3.1 GA's Components Definition

Encoding. In our algorithm an individual (or chromosome) I is represented by a chain of 0's and 1's, with length n, i.e., $I = g_0 g_1 \ldots g_{n-1}$, where each element, g_i, is called a *gene*. Each gene represents a vertex of the polygon, i.e., the i^{th} gene represents the vertex $v_i \in P$. The value of each gene is 0 or 1. If the $g_i = 1$ then the vertex v_i is a vertex-guard; otherwise ($g_i = 0$) the vertex v_i is not a vertex-guard (see Figure 1).

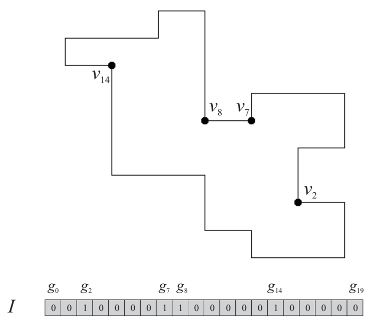

Figure 1: An individual I (for an orthogonal polygon with $n = 20$) and its representation. Black dots represent vertex-guards.

Initial Population. The population of a given generation/iteration consists of a set of individuals. The total number of individuals in each population has to be large enough to ensure diversity, but not too much that damages the efficiency of the algorithm. In our case it has been taken as the population size the number of reflex vertices of the polygon, $r = \frac{n-4}{2}$, linking in this way the entrance of the problem with the elements of the metaheuristic. Thus, the population for the generation t in our algorithm is represented by: $P(t) = \{I_0^t, I_1^t, \ldots, I_{r-1}^t\}$, where each I_i^t represents an individual belonging to the population $P(t)$ and r is the number of reflex vertices of the polygon P. Remember that, an individual represents a possible solution for our problem, i.e, each individual must be a vertex-guard set. It has been proven that being P a polygon with r reflex vertices then r guards, placed on the reflex vertices of P, are always sufficient and occasionally necessary to guard P. Thus, in our algorithm, being $R = \{u_0, u_1, \ldots, u_{r-1}\}$ the set of reflex vertices of P, to create the initial population, $P(0)$, we generate each of the r individuals in the following way: $\forall i \in \{0, \ldots, r-1\}$, if we $R \backslash \{u_i\}$ form a vertex-guard set we mark all the vertices of $R \backslash \{u_i\}$ has guards; otherwise we we mark all the vertices of R has guards. For example, in Table 1 it is illustrated the initial population, $P(0)$, of the polygon exemplified in Figure 2.

Table 1: Individuals of $P(0)$

$I_0^0 = 00010011001001010100$	$I_4^0 = 01010011000001010100$
$I_1^0 = 01000011001001010100$	$I_5^0 = 01010011001000010100$
$I_2^0 = 01010001001001010100$	$I_6^0 = 01010011001001000100$
$I_3^0 = 01010010001001010100$	$I_7^0 = 01010011001001010100$

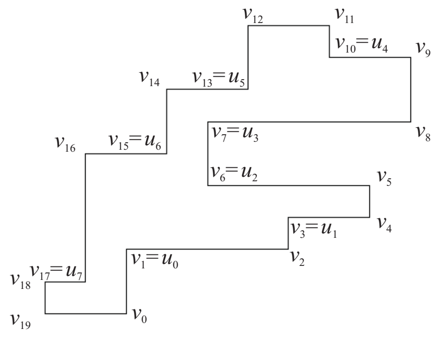

Figure 2: Polygon with $n = 20$.

Objective or Fitness Function. This function should help us to make the best selection of individuals to be reproduced, so that it will assign lower values to the solutions closer to the optimal one. In our case, given an individual I, the fitness function, $f(I)$, returns the number of 1's that exists in the chain that represents it.

Selection. The selection method should choose the best individuals to be reproduced. While there are many different types of selection, we have realized a comparative study taking into account two common methods: the *roulette wheel selection* and the *tournament selection* (see, e.g. [11]). In any method choose the two best individuals to be parents in crossover.

Crossover. Crossover operates on selected genes from parent individuals and creates new individuals (children). While there are many different kinds of crossover, we have realized a comparative study with four different types of crossover: *single point crossover*, *two-point crossover*, *uniform crossover* and a variant of the single point crossover where the generated children cannot be clones of the parents (see, e.g. [11]). In any crossover method we generate only one child from the two parents. Crossover does not always occur, it occurs with a given probability, p_c. The value of p_c is decided on the basis of trial and error, however, p_c is generally between 70% and 95%. We use $p_c = 80\%$. Note that, the child resulting from any of these crossover methods may not be valid (i.e., it may not correspond to a guard-vertex set), see Figure 3, in this case the child is not accepted.

In these example, the obtained child is not valid, because the polygon is not covered by the vertices v_4, v_9, v_{10}, v_{13} and v_{17}, as we can see in Figure 4. In this Figure the covered part by those vertices is the shaded zone.

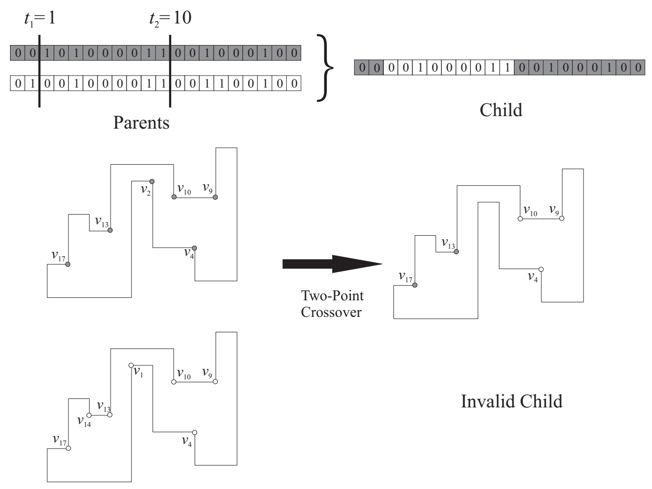

Figure 3: Two-Point crossover and an invalid child.

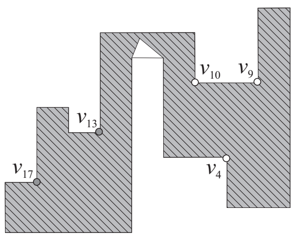

Figure 4: Invalid child.

Mutation. Since we use a binary representation, the action of our mutation operation is relatively simple. For each binary digit it merely flips it from zero to one or vice versa, with a mutation probability p_m. These probability of mutation, is decided on the basis of trial and error, however it is usually less than 1%. In our case we apply the mutation to the child obtained in the crossover operation with $p_m = 0.05$. As in the crossover, if the resultant individual is not valid we do not accept it.

Population's Generation. To generate a new population we replace the worst individual of the population by the child obtained at the crossover.

Population's Evaluation. We consider the evaluation of a population, i.e., the fitness of a population, $F(P(t))$, as the minimum value of the objective function when applied to all individuals of the population, i.e., $F(P(t)) = \min\{f(I_0^t), \ldots, f(I_{r-1}^t)\}$.

Termination Condition. If in a sufficiently large number of generations the fitness has not changed, we can assume that we are close to optimal. Thus, we consider as the termination condition that the fitness of the population $F(P(t))$ remains unchanged for a number of generations h. In our case, has been considered $h = 500$ (this value was chosen empirically).

3.2 Removing Redundant Vertex-Guards

After defining the GA's components, we obtain an approximation algorithm, that allow us to get a vertex-guard set, G. However, it can happen that some elements of G are redundant, i.e., it may be possible that exists a set $U \subset G$ such that $\bigcup_{v \in G \setminus U} Vis(v)) = P$. Thus, we apply a post-processing step where we iteratively remove those guards. This post-processing is done in the following way: for each $v_i \in G$, if P still covered without v_i we remove v_i from G; otherwise it remains in G.

4 Experiments and Results

According to section 3.1 we have various choices for two of the GA's parameters (the genetic operators: selection and crossover). The different combinations produce eight methods (approximation algorithms) (see Table 2). We have performed extensive experiments to analyze to see which of the methods best fits into our problem. In this section we relate our results and conclusions from our experiments.

Methods	
M_1	Roulette Wheel and Single Point Crossover
M_2	Roulette Wheel and Two-Point Crossover
M_3	Roulette Wheel and Uniform Crossover
M_4	Roulette Wheel and Variant of S.P. Crossover
M_5	Tournament and Single Point Crossover
M_6	Tournament and Two-Point Crossover
M_7	Tournament and Uniform Crossover
M_8	Tournament and Variant of S.P. Crossover

Table 2: The 8 different methods

We have implemented these methods on a PC using the CGAL library (version 3.2.1). Our software works with Microsoft Windows XP with Microsoft C++ compiler in Visual Studio 2005. The tests were performed on a Windows XP PC with an Intel(R) Core(TM)2 CPU 6400 2.13GHz, 1.00 GB of RAM. Our experiments were realized on a large set of randomly generated orthogonal polygons, generated by the polygon generator developed by O'Rourke (for evaluation of [10]). We analyze the eight methods by comparing the number of vertex-guards, the time spent and the number of iterations performed by each of the eight methods. The following experiments were realized with four sets of orthogonal polygons, each one with 40 polygons of 30, 50, 70 and 100 vertex polygons, respectively. In Tables 3 and 4 are exposed the results obtained by the first four methods. These tables present, the average number of vertex-guards, the average runtime in seconds and the average number of iterations of the algorithm.

Table 3: Results obtained by M_1 and M_2

n	M_1			M_2		
	|G|	Time	Iterations	|G|	Time	Iterations
30	4.725	14.95	746.025	4.625	15.25	749.225
50	7.925	59.75	1150.9	7.875	59.1	1151.2
70	10.675	162.325	1789.5	10.85	157.575	1712.5
100	15.475	421.8	2738.5	15.5	401.875	2601.8

Table 4: Results obtained by M_3 and M_4

n	M_3			M_4		
	|G|	Time	Iterations	|G|	Time	Iterations
30	4.625	14.125	724.425	4.575	9.525	735.2
50	7.775	54.8	1102.7	7.775	50.25	1148.1
70	10.325	149.875	1729.8	10.6	145.675	1733.8
100	14.925	398.625	2745.4	15.35	387.425	2673.1

As we can see, in these first four methods there are almost no differences between the average number of vertex-guards obtained. But, M_3 seems to be the one that obtains slightly best solutions. Concerning the average runtime, M_4 seems to be the best one.

In the following four cases, we analyze how the different types of crossover behave, considering the tournament selection. The obtained results are illustrated in Tables 5 and 6.

Table 5: Results obtained by M_5 and M_6

n	M_5			M_6		
	|G|	Time	Iterations	|G|	Time	Iterations
30	4.7	14.05	712.85	4.75	14.4	740.175
50	7.9	50.05	1035	7.9	48.725	985.7
70	10.725	124.45	1399.8	10.725	122.075	1394.2
100	15.7	312.4	2058.2	15.55	296.85	1964

Table 6: Results obtained by M_7 and M_8

n	M_7			M_8		
	|G|	Time	Iterations	|G|	Time	Iterations
30	4.7	12.8	681.425	4.75	7.1	684.475
50	7.9	44.975	932.65	7.825	37.525	995.625
70	10.85	110.7	1269.5	10.675	99.475	1339.8
100	15.025	285.8	1994.4	15.375	265.625	1973.4

Again, in these four methods there are almost no differences between the average number of vertex-guards obtained. However, M_7 and M_8 seem to be the methods that obtains slightly best solutions. Concerning the average runtime, M_8 seems to be the best one.

Comparing the eight methods, we can notice that the obtained results, concerning the average of $|G|$, are approximately the same for all methods; and M_3 is the method that seems to get a vaguely best solution. Concerning the average runtime, M_8 seems to be the best method.

However, the comparison between the obtained data by our eight methods only makes sense if a statistical study is made to ensure the statistically significance of results. So, first of all we have studied the results related to the number of vertex-guards. To check the data normality we applied the Kolmogorov-Smirnof test (using the Statistics Toolbox (Version 7.3) of the MATLAB software), and we have obtained the following p-values for the method M_1: $1.3471e^{-036}$, for $n = 30$ and $1.0754e^{-036}$, for $n = 50, 70$ and 100. These values mean that there is always a case that is non-normally distributed. So, we used the Kruskal-Wallis test to compare our data. In this test we declare that a result is significantly different if the p-value is less than 0.05. The p-values returned by the Kruskal-Wallis test are 0.9407, 0.9837, 0.6175 and 0.2108 for the data obtained with the polygons with $n = 30, 50, 70$ and 100, respectively. So we can say that there are no significantly differences between the eight methods, regarding $|G|$.

According to the previous conclusion we proceed our statistical study concerning the runtime. To check the data normality we applied the Kolmogorov-Smirnof test, and we have obtained the following p-values for the method M_1: $1.0754e^{-036}$, for $n = 30, 50, 70$ and 100. Consequently, we used again the Kruskal-Wallis test to compare our data. The p-values returned by the Kruskal-Wallis test are 0, for the data obtained with the polygons with $n = 30, 50, 70$ and 100. So we can conclude that at least one methods is significantly different, concerning the runtime. Then we performed a multiple comparison test to determine which pairs of methods are significantly different, and which are not (using the MATLAB's multcompare function). The answers provided by the realized tests allow us to conclude that M_8 is always the best method. As, we want have to find a compromise between the goodness of the solution obtained and the algorithm's runtime we proceed our study considering that the M_8 is the algorithm that obtains the best solutions.

It is important to note that all alternatives with respect to parameters of the GA metaheuristic that could be explored is almost infinite. We have attempted in this work to find general references to these parameters, noting that a more exhaustive study in future investigations might improve the obtained results.

Now, to conclude about the average of the minimum number of vertex-guards needed to cover an orthogonal polygon with n vertices, we applied M_8 to eight sets of orthogonal polygons, each one with 40 polygons of 30, 50, 70, 100, 110, 130, 150 and 200 vertex polygons, respectively. The average of the obtained results, concerning $|G|$, are exposed in Table 7.

Vertices	30	50	70	100	110	130	150	200
$\lvert G\rvert$	4.75	7.825	10.675	15.375	17.2	20.25	23.375	31.2

Table 7:

Then, using the least squares method, we obtained the following linear adjustment (see Figure 5):

$$f(x) = 0.1561x - 0.0556 \approx \frac{x}{6.40} - 0.0556 \approx \frac{x}{6}$$

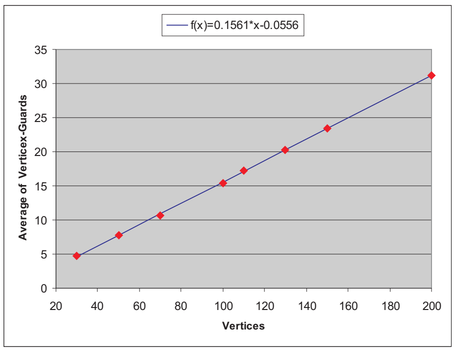

Figure 5: Least Squares Method.

Thus, we can conclude that on average the minimum number of vertex-guards needed to cover an orthogonal polygon P with n vertices is $\frac{n}{6}$.

5 Conclusions and Future Work

We designed and implemented eight approximation algorithms for solving the MINIMUM VERTEX GUARD problem on polygons based on the general metaheuristic Genetic Algorithms. We conducted an experimental study of their performance on orthogonal polygons, and we made a statistical study to elect the best method. Then, using the elected algorithm we concluded that on average the minimum number of vertex-guards on an orthogonal polygon with n vertices is $\frac{n}{6}$. Since the Genetic Algorithms metaheuristic have proven to behave well in solving the MVG problem on orthogonal polygons, there are several directions for further research. First of all, we intend to develop a method that allows us to determine the approximation ratio of the developed algorithms. We plan, also, to apply the described techniques to arbitrary polygons.

References:

[1] M. Abellanas, E. Alba, S. Canales and G. Hernández. Resolución de un problema de iluminación con Simulated Annealing (in spanish), Actas de MAEB'07, Tenerife, España, 2007.

[2] Y. Amit and J. S. B. Mitchell and E. Packer. Locating Guards for Visibility Coverage of Polygons, Proceedings of the Workshop on Algorithm Engineering and Experiments, 1-15, 2007.

[3] A.L. Bajuelos, S. Canales, G. Hernández, A.M. Martins. Estimating the Maximum Hidden Vertex Set in Polygons, Proceedings of ICCSA'08, 421-432, IEEE-CS Press, Perugia, Italy, 2008.

[4] C. Blum and R. Andreia. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Comput. Surv., 35(3):268–308, September, 2007.

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

[6] S.K. Ghosh. Approximation Algorithms for Art Gallery Problems, Proceeddings of the Canadian Information Processing Society Congress, 1987.

[7] F. Glover and G.A. Kochenberger. Handbook of Metaheuristics, Kluwer Academic Publishers, Boston, 2003.

[8] D. Lee and A. Lin. Computational Complexity of Art Gallery Problems, IEEE Transactions on Information Theory IT-32, 276-282, 1996.

[9] C. Marcelo, C. Souza, C. Cid, P.J. Rezende. An Exact and Efficient Algorithm for the Orthogonal Art Gallery Problem, Proceedings of XX Brazilian Symposium on Computer Graphics and Image Processing, v. 1, 87-94, 2007.

[10] J. O'Rourke, I. Pashchenko and G. Tewari. Partitioning orthogonal polygons into fat rectangles. In Proc. of 13th Canadian Conf. on Comp. Geom., 133 -136, 2001.

[11] C.R. Reeves. Genetic Algorithms, In: Handbook of Metaheuristics, F. Glover e G.A. Kochenberger (eds). Kluwer, Boston, 55-82, 2003.

[12] D. Schuchardt and H. Hecker. Two NP-Hard Art-Gallery Problems for Ortho-Polygons. Math. Logiv Quart 41, 261-267, 1995.

[13] A.P. Tomás, A. L. Bajuelos, F. Marques. Approximation Algorithms to Minimum Vertex Cover Problems on Polygons and Terrains, LNCS 2657, Springer-Verlag, 869-878, 2003.

[14] J. Urrutia. Art Gallery and Illumination Problems. In J.-R. Sack and J. Urrutia (eds) Handbook of Computational Geometry, Elsevier, 2000.