An algorithm for solving the open gallery problem

Marianne M. Robles
Institute of Mathematics, UPDiliman
C.P. Garcia Ave., Diliman
Quezon City 1101
mmrobles1@up.edu.ph

Joseph M. Pasia
Institute of Mathematics, UPDiliman
C.P. Garcia Ave., Diliman
Quezon City 1101
jmpasia@up.edu.ph

Henry N. Adorna
Department of Computer Science, UPDiliman
Velasquez Ave., Diliman
Quezon City 1101
hnadorna@up.edu.ph

ABSTRACT

Given a polygon and a desired number of guards resulting after performing an reflex vertex straddling (RVS) deployment or a polygonal triangulation, our objective is to find the minimum number of guards that could cover the entire polygon, that is, each point on the polygon should be visible to at least one of the guards. The initial number of guards of the algorithm is obtained by partitioning the polygon. Two methods of partitioning the polygon are considered. This initial number is then reduced using the sweeping mechanism of the Parallel Tree Sweep Strategy developed by Obermeyer et al. [5].

1. INTRODUCTION

Our problem is associated with the one Victor Klee had proposed in 1973 now known as the Original Art Gallery Problem: What is the number of guards sufficient to cover the interior of an n-wall art gallery room (where n refers to the number of vertices of the art gallery). In 1975 however, it was Vasek Chvátal who established what has become known as the Chvátal's Art Gallery Theorem: $\lfloor \frac{n}{3} \rfloor$ guards are occasionally necessary and always sufficient to cover a polygon of n vertices.

Although our problem is considered to be NP-hard, we let the $\lfloor \frac{n}{3} \rfloor$ guards (proved by Chvátal) to be our upper bound and we then determine a "good" lower bound (i.e. a possible minimum number of guards, if there is), such that this number of guards would be able to cover the entire polygon.

In this paper, we propose a method of finding the minimum number of guards given a specific polygon we would like to call as polygon M. This method is based on an algorithm used in the Searchlight Scheduling Problem which is known as the Parallel Tree Sweep Strategy (PTSS). To be able to achieve a schedule, PTSS works roughly in two steps: (1) partitioning the polygon and, (2) performing the sweep. However, we do not wish to find a schedule for our guards. But then again, using the two steps mentioned above helped us reach our goal and will be further discussed in Sections 3 and 4.

More importantly, we assume that the guards may be located at the edges (called edge guards) and/or at particular vertices of our polygon (called vertex guards). Also, each guard may perform a 360° sweep about their fixed positions in a counter-clockwise direction. Moreover, we say that a region is cleared if at least one guard swept over this particular region.

2. NOTATIONS AND PRELIMINARIES

This section presents us with the definitions and concepts that are further used within the study. These definitions will then prove to be useful in the discussion in the succeeding sections.

DEFINITION 1. A ***polygon P*** is defined as a collection of n vertices $v_1, v_2, \ldots, v_n$ and n edges $v_1v_2, v_2v_3, \ldots, v_{n-1}v_n, v_nv_1$ such that no pair of non consecutive edges share a point [6].

DEFINITION 2. The collection of vertices and edges is referred to as the ***boundary of P***, denoted by ∂P.

DEFINITION 3. A ***guard g*** is any point of a polygon P. Specifically, a vertex guard is one that is positioned at a vertex of a polygon while an edge guard is positioned at an edge.

DEFINITION 4. We say that a point $x \in P$ is said to be ***seen*** by (or ***visible*** from) a guard g if $\overline{xg} \subset P$. Here we note that $\overline{xg}$ may touch ∂P at one or more points [6].

DEFINITION 5. A collection of guards S is said to ***cover*** polygon P if every point $x \in P$ can be seen by some guard $g \in S$.

DEFINITION 6. The ***visibility set*** $V(x) \subset P$ from a point $x \in P$ is the set of points in P visible from x [5].

DEFINITION 7. A ***visibility gap*** of a point x with respect to some region $R \subset P$ is defined as any closed segment $[a, b]$ such that open segment $(a, b) \subset int(R)$, $[a, b] \subset \partial V(x)$ [5].

DEFINITION 8. A ***diagonal*** is an open line segment that connects two vertices of P and lies in the interior of P [1].

DEFINITION 9. A decomposition of a polygon into triangles of non-intersecting diagonals is called a ***triangulation of the polygon*** [1].

For the purpose of our study, it is best to give emphasis on the fact that we are dealing with polygon M with no holes, having 46 vertices (21 of those are reflex vertices), and having 46 edges (23 vertical, 21 horizontal and 2 slanting edges). This is given in Figure 1.

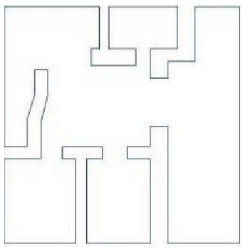

Figure 1: Polygon M

3. PARALLEL TREE SWEEP STRATEGY

Closely related to the Art Gallery Problem is the Searchlight Scheduling Problem wherein guards want to know how they should coordinate their sweeping (rotation about a point) such that they would be able to cover a particular polygon. [5] then will help us determine this using algortihms for the Searchlight Scheduling Problem which in turn are useful in our study.

3.1 Searchlight Scheduling Problem

The *Searchlight Scheduling Problem* is that of determining a way (or finding a schedule) to move a set of searchlights in a polygon P such that an intruder in P can be detected. Hence, this problem consists of an environment and a set of stationary guard positions [7].

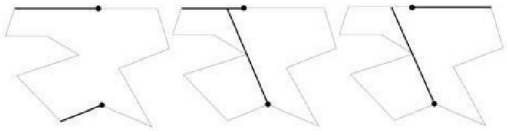

Figure 2: A schedule for two searchlights.

In 1990, Sugihara, Suzuki and Yamashita were the first to introduce and consider the problem of searching for a mobile robber in a simple polygon using stationary searchlights. Their primary task was to devise a schedule for aiming the searchlights so that the mobile robber will be detected in finite time. But more importantly, in 2007, two simple asynchronous distributed searchlight scheduling algorithms for multiple robotic agents in non-convex polygonal environments were developed in a paper by Obermeyer, Ganguli and Bullo, [5]. These are the Distributed One Way Sweep Strategy (DOWSS) and the Parallel Tree Sweep Strategy (PTSS).

To illustrate DOWSS, let us consider Figure 3. The configuration in (a) results from $s^{[0]}$ clearing the very top of the region with the help of $s^{[2]}$, $s^{[3]}$, and $s^{[4]}$ followed by $s^{[1]}$ attempting to clear the semiconvex subregion below where $s^{[0]}$ is aimed. When $s^{[1]}$ gets stuck, it requests help by broadcasting the thick black polyline in (a), in this case just a line segment. $s^{[2]}$ then helps $s^{[1]}$ but gets stuck right off, so it broadcasts the thick black polyline shown in (b). Next $s^{[3]}$ helps $s^{[2]}$ but gets stuck and broadcast the polyline in (c). Similarly $s^{[4]}$ broadcasts the polyline in (d), in this case a convex polygon, which only $s^{[5]}$ can clear. In general, information passed between guards during any execution of DOWSS will be in the form of either an oriented line segment (a), a general oriented polyline (b) and (c), or a convex polygon (d)[5].

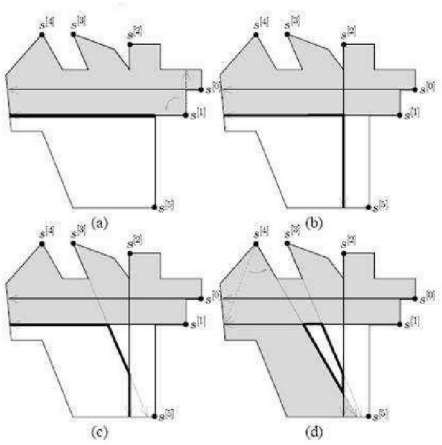

Figure 3: An example execution of DOWSS.

Also, according to [5], DOWSS can be a very slow algorithm in clearing the entire environment since only one searchlight may sweep at a time. Hence, this gave them the reason to develop another algorithm, the PTSS, in which searchlights sweep in parallel if guards are placed according to an environment partition which they called PTSS partitions. In addition, another comparison they have observed between the two algorithms is that, the DOWSS allows flexibility in guard positions; while PTSS requires that guards are positioned according to a PTSS partition, specifically a PTSS tree which will be discussed in the following section. Here we note that this last observation may then be helpful in achieving our objective in this paper.

3.2 Reflex Vertex Straddling (RVS) deployment

Using now polygon M and a PTSS partition called a Reflex Vertex Straddling (RVS) deployment, we were able to determine not only the positions of our guards, but also the number of guards who can cover the entire polygon. This RVS deployment begins with all guards located at the "root". This "root" may be chosen arbitrarily but in our case, we consider only choosing among the four (4) main corners of polygon M. From this root, one guard moves to the furthest end of each of the root's visibility gaps, thus becoming the children of the root. Likewise, further guards are deployed from each child to take positions on the furthest end of the child's visibility gaps.

In Figure 4, we chose the lower left corner of polygon M to be the "root" i.e. the circled guard, and the other guards as its children. Respective children will now become the parents and will then create their children, shown in (b) and (c). The final positions of the guards after the RVS deployment is shown in (d) where the PTSS partitions are represented by coloring the cells alternately (shaded area does not depict clarity) and the lines show edges of the PTSS tree.

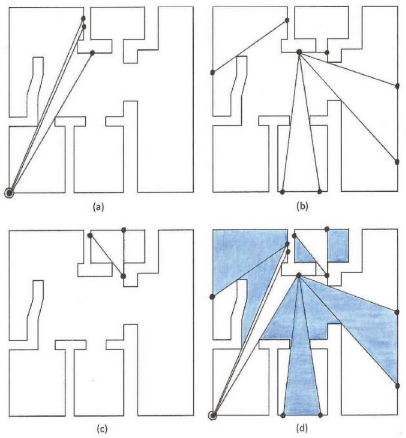

Figure 4: Guard positions resulting from an RVS deployment. 12 guards.

Hence, we were able to determine the number of guards who can cover polygon M after performing the RVS deployment. However, results also show that this number depends on where we assign out "root". Even though we tried to minimize the number of vertices which may be considered as the root by choosing just among four vertices, this gave us different numbers of guards resulting from the RVS deployment. Moreover, the RVS deployment cannot be applied to all four vertices. Only two out of these four vertices - either the lower left or the lower right corner as root - gave us a number of guards that can cover polygon M (see Figures 4 and 5).

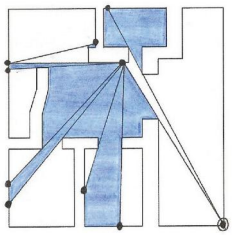

Figure 5: Lower right corner of polygon M as root. 10 guards.

Now we consider the upper left or upper right corner as our chosen root. Based from our results, saying that the RVS deployment cannot be applied for these two vertices meant that we were not able to determine the number of guards that can cover polygon M. There are dead spots (or a region) in polygon M not seen by any guard. Having a closer look at Figure 6, the shaded areas represent the dead spots. Hence, we decided to construct an algorithm such that we would still be able to get a number of guards using any of these two particular vertices as our root.

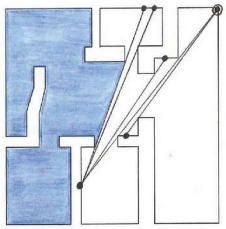

Figure 6: Upper right corner as root.

3.3 Algorithm for finding the number of guards using the upper right and upper left corners of polygon M as roots

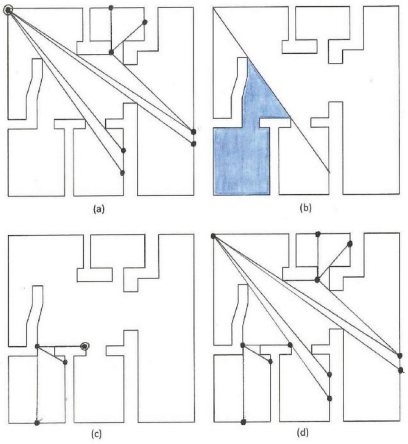

Figure 7: Upper left corner as root. 12 guards.

Consider Figure 7. Without loss of generality, we consider the upper left corner as our root and then perform the RVS deployment. We would then observe from (a) that there is a region in our polygon not covered by any guard, i.e. dead spots. Focusing now our attention to this said region, we would consider the visibility gap shown in (b), separating the "covered" (unshaded) region from the "uncovered" (shaded) region. We then choose the vertex right next to the furthest point on the visibility gap, reached by a guard and then apply again the RVS deployment, see (c). We continue this process of finding "new roots" until our guards cover the entire polygon. (d) shows the final positions of the guards. We may do the same for the upper right corner as root. This resulted to 12 guards as well.

Observe that we need to verify if 10 and 12 are less than $\lfloor \frac{n}{3} \rfloor$ by Chvátal's Art Gallery Theorem. But since our polygon has $n = 46$ vertices, clearly, we did not violate the said theorem. It is also very important to point out that according to [5], the final positions resulting from an RVS deployment

show that the guards are able to cover the whole of the polygon.

4. THE ALGORITHM

For us to be able to solve the minimum number of guards needed to cover polygon M, we constructed an algorithm which comprises of two steps, mainly (1) preprocessing/ initialization and (2) reduction.

4.1 Preprocessing/Initialization

In this step, we partition polygon M. Two different approaches of partitioning are considered namely, the RVS deployment discussed in Section 3 and the triangular partitioning and 3-coloring.

Simply stated, a partition of a polygon into triangles is known as triangulation. And since this is one way of partitioning a polygon, again, as mentioned earlier, it is the first step of many advanced algorithms. Therefore, it reduces complex shapes to collection of simpler shapes. We know that every polygon can be triangulated, however, triangulation is not unique (see Figure 8).

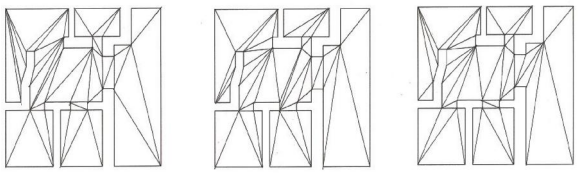

Figure 8: Three different triangulations.

On the other hand, a k-coloring of a graph is an assignment of colors to the nodes, one color per node, using no more than k colors, such that no two adjacent nodes are assigned the same color. The nodes of the triangulation graph correspond to the vertices of our polygon, and the arcs correspond to the polygon's edges plus the diagonals added during triangulation. In particular, we let $k = 3$ [6].

In [6], Fisk's proof on the Art Gallery Problem using concepts on triangulation and 3-coloring showed that there exists a number of guards ($\leq \lfloor \frac{n}{3} \rfloor$) that can cover a polygon of n vertices.

4.2 Reduction Process

After determining the number of guards after partitioning our polygon, we then use *sweeping* to check if we can still reduce that number such that the guards can still cover the whole area of our polygon (see Steps 1-4 in the following section for the description of this process). Note that the algorithm stops after we finish checking the sweep made by all x guards.

4.3 Step-by-Step Algorithm

The Algorithm for Finding the Minimum Number of Guards

Preprocessing: Apply RVS deployment or, triangular partitioning and 3-coloring into our polygon. For 3-coloring, we count how many times each color appears on our polygon and then select the color which has the least number. By choosing the smallest color class, these guards can then cover the whole polygon. The number and the positions corresponding to the smallest color class serve as our basis for our algorithm.

Step 1. Suppose we're done partitioning our polygon. As a result, we get x number of guards to cover the whole of our polygon.

Step 2. Let $i = 0$ and $ctr = x$. We orient polygon M in a counterclockwise or clockwise direction and then denote the guards as $g_1, g_2, \ldots, g_x$.

Step 3. Sweeping: Let $i = i + 1$. We consider guard g_i.

> **Step 3.a.** We aim g_i as far clockwise as possible so that it is aligned along the clockwise-most edge. **Step 3.b.** We will then sweep counterclockwise through the polygon, stopping everytime it encounters a visibility gap, until it is pointing along the edge immediately to its left. Note that our guards can sweep up to 360°. **Step 3.c.** If $i \geq 2$, go to **Step 4**. Otherwise, repeat **Step 3.**

Step 4. Let R_{g_i} be the region swept by guard g_i. If $(R_{g_{i-1}} \cap R_{g_i} \neq \emptyset)$ and $[R_{g_i} \subset (R_{g_1} \cup R_{g_2} \cup \ldots \cup R_{g_{i-1}})]$ or $(R_{g_{i-1}} \subseteq R_{g_i})$ then we let $ctr = ctr - 1$ and then repeat **Step 3**. Otherwise, just repeat **Step 3**.

Stopping Criterion: The algorithm stops when $i > x$. The minimum number of guards will be given by the value equal to ctr.

5. RESULTS

Here are some of the results after performing the algorithm discussed in Section 4. We first assume that polygon M is oriented in a counterclockwise direction.

The algorithm using "RVS - sweeping" (RVSS hereinafter) is illustrated in Figures 9 and 10. Figures 9(a) and 10(a) show how we perform the deployment using as our root the lower right and upper left corner of polygon M, respectively. To check, we can refer to Figures 5 and 7. Figures 9(b) and 10(b) is a representation of **Step 4** where the vertices marked with X are not chosen as guards. Lastly, the final positions of the guards and the angle of their sweep is given in Figures 9(c) and 10(c).

On the other hand, the algorithm using "triangulation - 3-coloring - sweeping" (T3S hereinafter) is illustrated in Figure 11. (a) shows the positions of $x = 11$ guards after triangulation and 3-coloring. Similarly, (b) is a representation of **Step 4** where the vertices marked with X are not chosen as guards. And the final positions of the guards and the angle of their sweep is given in (c).

Here we recall that triangulation is not unique. Hence, what is shown in Figure 11(a) is just one of the few ways of triangulating polygon M. In addition, if we use RVS deployment and try to consider the lower left or upper right corner of

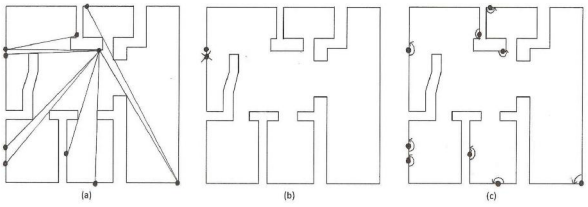

Figure 9: A sample execution of the algorithm using RVSS (lower right corner as root). 9 guards.

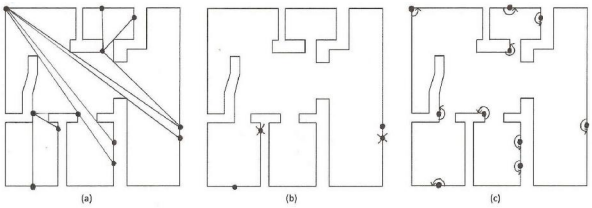

Figure 10: A sample execution of the algorithm using RVSS (upper left corner as root). 10 guards.

polygon M as our root, and then perform the same algorithm, we should be able to get 11 guards for each.

Comparing now that results we got from RVSS (on four specific corner points) and T3S, we have observed that we were able to find a triangulation for polygon M that gave us fewer number of guards.

It is known that according to [1], T3S can be computed in $O(nlogn)$ time while RVSS can be computed in $O(n)$ time according to [5]. Nonetheless, we would like to emphasize the fact that our proposed algorithm is a heuristic approach. Although T3S is optimal in the worst case, see Figure 12, other polygons (e.g. polygon M) may give us a number less than $\lfloor \frac{n}{3} \rfloor$.

6. RECOMMENDATIONS

In this paper, we emphasize that we only deal with polygon M to test our algorithm. Hence, people may be interested in working on different and more complex polygons to check if the algorithm still applies in such cases. You may specifically consider orthogonal polygons, polygons with (or without) holes, monotone polygons, star polygons, and the like. Here are some examples shown in Figures 13 and 14.

At the same time, since one of the main properties of our algorithm is partitioning the given polygon, you may find ways other than the RVS deployment and triangulation to partition your polygon. Note that some polygonal partitioning may be dependent on the polygon of your choice.

Based from our algorithm, we were able to observe that the number of sweeps made is equal to the number of guards after partitioning the polygon. This is because we wish to check the area swept by each guard before determining which can be removed. Hence, considering also that *sweeping* is a crucial stage in our algorithm, one may decide to

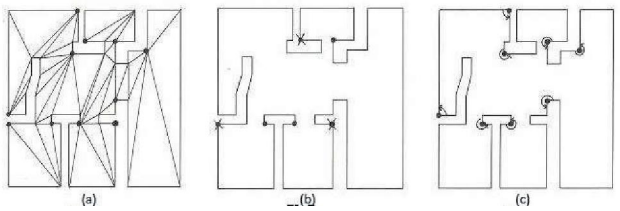

Figure 11: A sample execution of the algorithm using T3S. 8 guards.

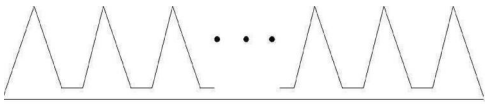

Figure 12: Comb-shaped polygon

improve it by creating their own process of *sweeping* such that the number of sweeps made will be less than the number of guards after partitioning the polygon.

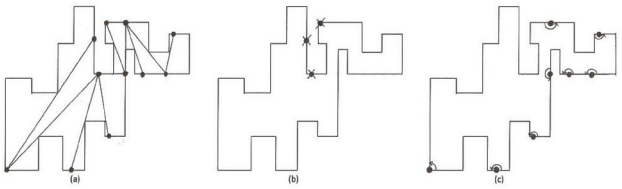

Figure 13: Polygon A ($n = 32$) oriented counterclockwise; using RVSS. 8 guards.

7. REFERENCES

[1] M. Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, Springer-Verlag Berlin Heidelberg, Germany, 1997.

[2] A. Efrat, L. J. Guibas, S. Har-Peled, D. C. Lin, J. S. B. Mitchell, and T. M. Murali, *Sweeping Simple Polygons with a Chain of Guards*, PhD thesis, Stanford University, Tel Aviv University, University at Stony Brook, 1999.

[3] S. Eidenbenz, C. Stamm, and P. Widmayer, *Inapproximability results for guarding polygons and terrains*, Algorithmica, 31 (2001), pp. 79–113.

[4] S. M. Lavalle, B. H. Simov, and G. Slutzki, *An algotrithm for searching a polygonal region with a flashlight*, International Journal of Computational Geometry and Applications, (2000).

[5] K. Obermeyer, A. Ganguli, and F. Bullo, *Asynchronous Distributed Searchlight Scheduling*, PhD thesis, University of California, University of Illinois, 2007.

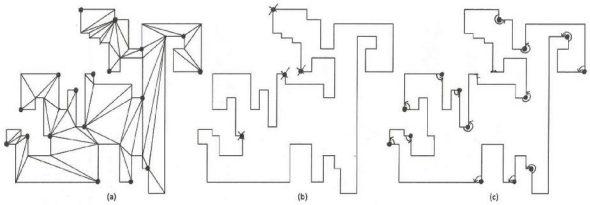

Figure 14: Polygon B ($n = 64$) oriented counterclockwise; using T3S. 14 guards.

[6] J. O'Rourke, *Art Gallery Theorems and Algorithms*, Oxford University Press, Inc., New York, New York, 1987.

[7] J. Urrutia, *Art Gallery and Illumination Problems*, PhD thesis, University of Ottawa, 2000.

[8] P. Żyliński, *Watched guards in art galleries*, Journal of Geometry, 84 (2005), pp. 164–185.