A Heuristic for the Orthogonal Polygon Art Gallery Problem

Helmuth Trefftz
Rutgers University
CoRE Building Room 719
96 Frelinghuysen Road
Piscataway, New Jersey, 08854
USA
e-mail trefftz@caip.rutgers.edu
Christian Trefftz
Universidad EAFIT
A.A. 3300
Medellín
Colombia-SouthAmerica
e-mail ctrefftz@sigma.eafit.edu.co

September 24, 1999

Abstract

A new heuristic to find non-optimal solutions to the Orthogonal Polygons instance of the Art Gallery Problem is described. [1]
Keywords: Cellular Automata, Art Gallery Problem, SIMD Parallel Processing.

1 Introduction

Two-Dimensional Cellular Automata (CA) [1] have been used as tools for modeling a variety of physical phenomena. Systems like CAMEL [2], NEMO [3] and the work on Route Planning by Stiles and Glickstein [4] allow users to use a computer to solve a number of problems interactively using cellular automata to model different real-world situations. Cellular Automata models are easily implemented on SIMD (Simple Instruction Multiple Data) parallel machines.

The Art Gallery Problem [5] was posed in 1976 by Klee: "How many guards are always sufficient to guard any polygon with n vertices?". The problem is also known as an *Illumination Problem* because it can be posed

[1]Funding for this project has been provided by EAFIT University.

in terms of lights illuminating the Art Gallery instead of guards guarding it. The Art Gallery Problem is a classic problem in Computational Geometry.

In this paper we restrict ourselves to the *Orthogonal Polygons* instance of the Art Gallery problem. A simple polygon divides the plane in two regions: The exterior region (unbounded) and the interior. A simple polygon is called *orthogonal* if all its edges are parallel to either the x-axis or the y-axis [5]. This is, the polygon in which we will attempt to solve the problem will be orthogonal. Furthermore, we restrict ourselves to the instance of the problem where guards can be placed only at the vertices of the polygon.

In the NEMO system, another Computational Geometry Problem was solved (approximately) using Cellular Automata: The Voronoi Diagram. In a Voronoi Diagram, a number of "seed" points or generators are given. The plane is divided in tiles such that the points in each tile are closer to a generator than to any other generator. In the NEMO solution, initially, every generator is assigned a different color and all other cells are "blank". Every cell in the CA examines its neighbors and adopts the color of a colored neighbor if there is one [2]. Once a cell has adopted a color, it keeps it. The process repeats until there are no further changes on the CA.

In a sense, the generators are acting as sources of light and we use this behavior to help solving the Art Gallery Problem. We have modified the CA to account for the effect of shadows as the CA used for the Voronoi problem allows light to expand around the corners of the polygon. Based on this CA, a heuristic has been developed to solve the Orthogonal Polygon Illumination Problem on SIMD machines.

The rest of this paper is structured as follows: Section 2 contains the description of the CA used to simulate the propagation of light. Section 3 contains a description of how this CA could be incorporated into a heuristic for a SIMD machine to find solutions to the Orthogonal Polygon Illumination Problem. Conclusions and Future Work are included in section 4.

2 A Cellular Automata to model the propagation of light

The first addition to the CA used for the Voronoi Diagram is to include cells that are part of the "walls" of the Art Gallery. These cells do not acquire any color. They are given a "wall" color from the onset of the CA and they never change their state.

A point can be illuminated by more than one source of light, there is no need to draw boundaries among the areas lit by different light sources in the Illumination Problem. The desired result is to find out if a given number and location of light sources is enough to illuminate all points (cells) inside the polygon.

[2] If a cell has two neighbors with different colors it is on an edge and it is takes the color designated for edges

The simple rules used for the CA that approximates the Voronoi Diagram do not model the propagation of light correctly because they allow light to bend around corners. Figure 1 illustrates the problem: The original rules would allow light from the source s to reach the cell labeled "a".

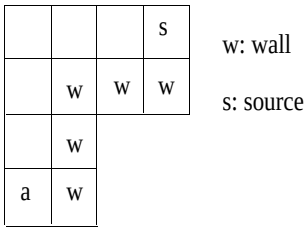

Figure 1: The problem with the original Voronoi rules.

The missing element in this model is the direction of the ray of light hitting a particular cell. All non-wall neighbors of a source of light will be illuminated by it. But not all neighbors of a lit cell should change their state to lit as well: Only those neighbors which are in the same direction as the ray of light hitting the cell.

A possible solution is to assign to the cell a state that represents the angle formed by it with respect to the source of light. Figure 2 shows the states assigned to the neighboring cells of a light source.

135	90	45
180	s	0
225	270	315

Figure 2: The states assigned to the neighbors of a light source.

If a cell has a neighbor that is lit, it will change its state also to lit only if value of the lit neighbor is in a certain range that indicates that the ray of light hitting the neighbor will also hit this cell. For instance, if the cell immediately below a cell "A" is lit, cell "A" will change its state to lit only if the state of its neighbor below is between 68 and 112 (90 +/- 22). This solution is not good enough though, as can be seen in Figure 3.

Examining a single neighbor may lead to situations like the one depicted in Figure 3, where a number of cells that should be lit are left "dark" because there is no single neighbor that is hit by a ray of light in an angle that propagates the ray of light to those cells.

This can be solved by examining pairs of adjacent neighbors. This second inspection of the neighbors is necessary only if the first inspection of the neighbors did not result in the cell becoming lit. If both adjacent

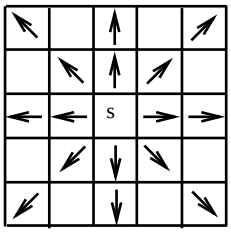

Figure 3: Problems with examining one single neighbor.

neighbors are lit, then the average of both neighbors is taken. Again, if the result is in a certain range that depends on the particular pair of neighbors being examined, then the change of the cell is changed to lit.

3 Finding solutions to the Orthogonal Polygon Illumination Problem

The minimum vertex guard problem for polygons was proven by Lee and Lin to be NP-hard [6]. Furthermore, the minimum vertex guard problem for orthogonal polygons was also proven to be NP-hard by Schuchardt et al. [7]. Kahn et al. proved that any orthogonal polygon with n vertices can always be illuminated with $\lfloor \frac{n}{4} \rfloor$ vertex guards. $\lfloor \frac{n}{4} \rfloor$ guards are sometimes necessary [8].

The goodness of a given set of guards can be measured by the number of dark cells after no more changes occur in the CA. A good solution (not necessarily optimal) will leave no dark cells.

A prototype was implemented in the X11 graphical environment. Figures 4 and 5 show a polygon with one and three guards respectively. Unfortunately our choice of colors was not reflected on the black-and-white image, but the program behaved as expected: With one single guard there are areas of the polygon that are left unlit while with the three guards, the entire polygon is illuminated.

Implementing a Cellular Automata on a SIMD machine can be straightforward if the SIMD machine offers the appropriate primitives. For instance, on MasPar MP machines there are primitives available to display on the screen images in which each pixel is associated with a (virtual) processor. The color of the pixel can reflect the value of a local variable on that processor.

Furthermore, the programming environments on SIMD machines usually offer a collective operations like reduction.

Thus, the following greedy heuristic has been defined to cover all points in the CA:

Let n be the number of vertices on the polygon. Every cell in the cellular automata will contain an array called *illuminatedBy* of size n. *illuminatedBy*[i] will be 1 if point i of the polygon illuminates this cell or 0 otherwise.

Observe that every cell will contain at least one entry in *illuminatedBy* with a non-zero value.

Another array will be used: $totalNumberOfCells$, also of size n. $totalNumberOfCells[i]$ will contain the number of cells that are illuminated by the i source of light.

The gist of the heuristic is to choose the source of light that cover the largest number of cells and to repeat the process until all points inside the polygon are illuminated at least by one source of light.

Another variable , *active*, will be present in every cell. *active* will be initially set to 1 and will be set to 0 when this cell is illuminated by a source that has been chosen as part of the solution.

Solution will denote the set of vertices that has been chosen to illuminate the entire polygon.

```
1 Solution = φ
2 In parallel for each cell in the CA
3    Calculate which sources of light illuminate this cell
4    Update illuminatedBy accordingly
5 endfor
6 repeat
7    for i = 1 to n
8    totalNumberOfCells[i] = 0
9    endfor
10   for i = 1 to n
11      totalNumberOfCells[i] = Number of all cells
        for which (active==1) and (illuminatedBy[i] == 1)
12   endfor
13   j = i such that
     totalNumberOfCells[i] == Max over totalNumberOfCells
14   Solution = Solution ∪ j
15   In parallel for each cell in the CA
16      if (illuminatedBy[j] == 1)
17         active = 0
18      endif
19   endfor
20until (active == 0) for all cells
21print Solution
```

We have not implemented this heuristic yet. As we do not have access to a SIMD machine we plan to implement the heuristic using a simulator. We are currently considering Parallaxis [9].

4 Conclusions and Future Work

CA can be used to model the propagation of "light" on a plane and can be used as an auxiliary tool to find good (not optimal) solutions.

As stated in the previous section, we plan to implement the heuristic described here and test the goodness of the solutions found with it. It

would be interesting to integrate this CA in an environment like CAMEL or NEMO.

References

[1] S. Wolfram and N. Packard, "Two-dimensional cellular automata," *Journal of Statistical Physics*, vol. 38, pp. 901–946, March 1985.

[2] G. Spezzano, D. Talia, S. D. Gregorio, R. Rongo, and W. Spataro, "A parallel cellular tool for interactive modeling and simulation," *IEEE Computational Science and Engineering*, pp. 33–43, Fall 1996.

[3] D. Hutchinson, L. Kuettner, M. Lanthier, A. Maheshwari, Nussbaum, D. Roytenberg, and J. Sack, "Parallel neighbourhood modeling," in *Proceedings of SPAA 96*, pp. 204–207, June 1996.

[4] P. N. Stiles and I. S. Glickstein, "Highly parallelizable route planner based on cellular automata algorithms," *IBM Journal Research and Development*, vol. 38, pp. 167–181, March 1994.

[5] J. Urrutia, *Handbook on Computational Geometry*, ch. Art Gallery and Illumination Problems. Elsevier, 1997.

[6] D. T. Lee and A. K. Lin, "Computational complexity of art gallery problems," *IEEE Transactions on Information Theory*, vol. IT-32, pp. 415–421, 1979.

[7] D. Schuchardt and H. Hecker, "Two np-hard problems for ortho-polygons," *Math. Logiv Quart.*, vol. 41, pp. 261–267, 1995.

[8] J. Kahn, M. Klaw, and D. Kleitman, "Traditional galleries require fewer watchmen," *SIAM Journal on Algebraic and Discrete Methods*, vol. 4, pp. 194–206, 1983.

[9] T. Braeunl, *Parallel Programming*. Prentice Hall, 1993.

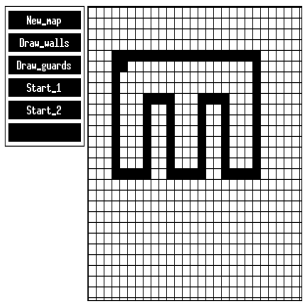

Figure 4: A polygon with 1 guard.

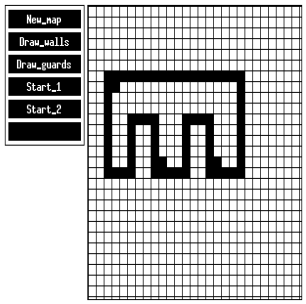

Figure 5: A polygon with 3 guards.