Partitioning a Deformed Urban Grid

Leonard Hagger Ian Sanders *

Extended Abstract

1 Introduction

The problem of partitioning a polygon into a minimum number of convex pieces is well studied in computational geometry. It has applications in syntactic pattern recognition, database systems and graphics. Lingas [Lin82] showed the problem of partitioning polygons with holes to be NP-hard so special cases, especially those that are suited to specific applications, are of interest. This paper discusses such a special case – partitioning a *deformed urban grid* (DUG). This problem arises from the automation of a technique – *space syntax* analysis – aimed at providing greater understanding of urban layouts. Sanders [San02] addresses the computational problems associated with automating *space syntax* analysis and focuses on the problem of axial line placement (ALP) which is one of the phases of the technique. In essence, ALP is the problem of placing a minimum number of straight line segments to cross all of the shared edges (adjacencies) between convex polygons in a given collection of adjacent convex polygons. A deformed urban grid (as shown in figure 1) can be considered as a model of an urban layout. Application of space syntax would require the deformed urban grid to be minimally partitioned and then have axial lines placed to cross the adjacencies between the convex polygons in the partition. Thus it is important to find a polynomial time solution to this problem (if one does exist) and then to see how far we can relax the definition of a *deformed urban grid* and still get polynomial time solutions.

In this paper two heuristic algorithms are presented that partition deformed urban grids into their convex parts. One algorithm takes the approach of partitioning an even more restricted case optimally and then applying the algorithm to a DUG. The other uses the fact that polygons without holes can be partitioned optimally. Although a polynomial algorithm that partitions the DUG into the minimum number of convex parts has not been found yet these two algorithms take advantage of the arrangement to give a good approximation. These algorithms were implemented and compared to a previous algorithm presented by Sanders [San02].

*University of the Witwatersrand, Private Bag 3, WITS, 2050, South Africa, lhagger@cs.wits.ac.za, ian@cs.wits.ac.za

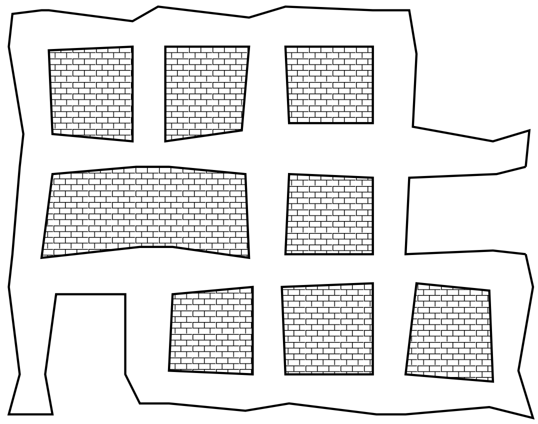

Figure 1: A deformed urban grid

2 The Deformed Urban Grid

In this section the deformed urban grid is defined. The original definition of the problem is given by Sanders [San02] and is based on the idea of a complete grid [GN93] – the *complete two-dimensional grid* of size n is the graph with vertex set $V = \{1, 2, \ldots, n\} \times \{1, 2, \ldots, n\}$ and the edge set $E = \{\{(i,j),(k,m)\} : |i-k| + |j-m| = 1\}$ where all edges are parallel to the major axes. In a geometric setting, the grid edges can be thought of as corridors and the grid vertices as intersections of corridors. A (partial) grid is any subgraph of the complete grid. In a simple urban grid the corridors and intersections are given dimension. A simple urban grid is thus a polygon with holes that is the union of rectangles and squares. The rectangles are the “corridors” and have width w and length l. The squares are the “intersections” and have width w. Each corridor must begin and end with an intersection as shown in figure 2. The figure also shows illegal arrangements.

No intersection can be adjacent to another intersection. A horizontal (or vertical) sequence of corridors and intersections is called a “thoroughfare”. An example of a simple urban grid is given in fig 3. The dotted lines in the figure are not part of the grid (neither are the “bricks”) they are just there for illustration. A deformed urban grid is generated from a simple urban grid and is formed by deforming (moving the vertices of) the

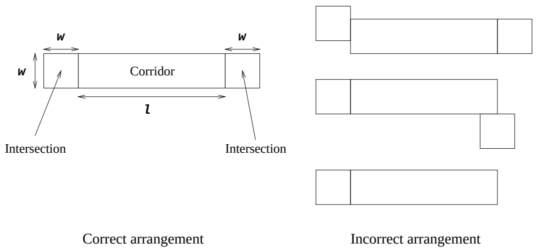

Figure 2: Urban grid arrangement

intersections, hence deforming the corridors since they share the same vertices. Corridors, intersection and thoroughfares in DUGs are similar to those in simple urban grids.

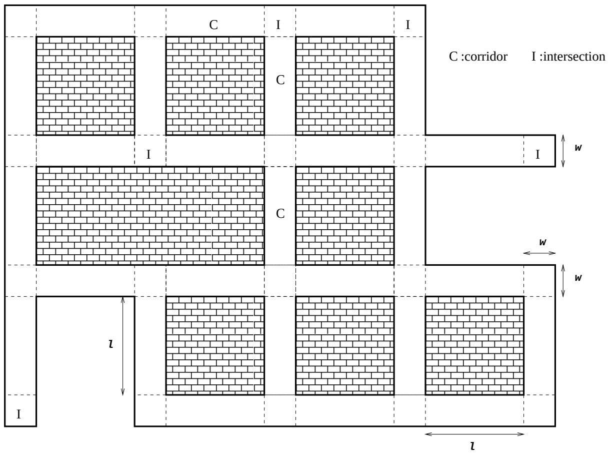

Figure 3: A simple urban grid

The degree to which the intersections should be deformed is also an issue. Consider a simple urban grid of dimensions w and l where $w < l$. To form the deformed urban grid the vertices of any intersection may be moved around but only so that they still lie inside the original simple urban grid's intersection. The intersections and corridors must remain convex.

This restriction has been made to ensure that a convex polygon cannot be formed from the union of two corridors [1] that are in different thoroughfares. Konidaris and Sanders [KS02] prove that this definition has this property. It is this deformation constraint that is thought to lead to a polynomial solution. If a polynomial solution exists then this constraint would be relaxed to push the problem as far as it could go.

[1] Or triangles formed from taking the diagonals of the corridors

3 Background

The problem has been shown to be NP-hard by Lingas [Lin82] and as a result approximation algorithms have been developed to solve the problem [MS98].

Keil [Kei85] developed an $O(N^2 n \log n)$ time, dynamic programming algorithm to partition a polygon *without* holes, were N is the number of reflex vertices. Keil and Snoeyink [KS98] improved this algorithm to $O(n + N^2 \min(N^2, n))$ time. This algorithm is of interest because a DUG can be broken up into polygons without holes quite easily.

The original problem was presented by Sanders [San02]. A simple algorithm was given by Sanders [San02, San00] to partition a DUG. Bukovska [Buk00] improved on it somewhat by improving the running time by a constant factor. They are both $O(n)$ algorithms. The polygons in the final partition only included whole corridors and intersections. The algorithm in [San02] is referred to as "the original algorithm" in this paper.

4 Preliminaries

In this section some definitions are given to classify the types of intersections that are possible. Also an algorithm is given that partitions a simple urban grid optimally which is used in the heuristic to partition the DUG.

4.1 Types of Intersections

To discuss the algorithms we need some additional terminology which is illustrated in figure 4

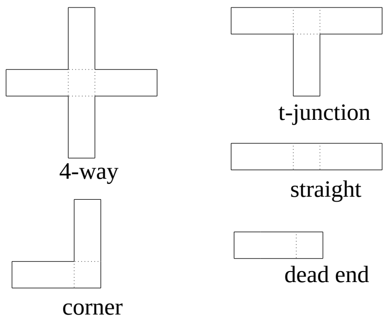

Figure 4: Types of intersections

4.2 The Simple Urban Grid Algorithm

Partitioning the simple urban grid is important even outside of this research on deformed urban grids. This is because many town plans are made up of simple urban grids or at least come very close. If they are very close then they can be turned into the simple case.

The algorithm works by partitioning each intersection as in figure 5. The complete algorithm appears in the full paper which also presents arguments as to why it works. Sanders [San02] gives a formal proof of correctness.

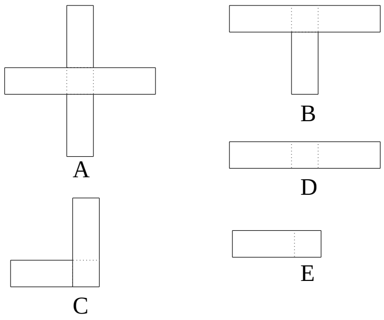

Figure 5: Partitions for the simple urban grid algorithm

5 Intersection Based Algorithm

The intersection based algorithm is presented here. The assumption or heuristic for this algorithm is that since the algorithm finds the optimal solution for the simple urban grid case a good solution can be found by using it on deformed urban grids. The complete algorithm is given in the full paper. Note that in figure 6 the partitions are for simplicity drawn as if it was in a simple urban grid (with squares and rectangles) but we are working with a deformed urban grid. The arguments presented apply for the specific case of a simple urban grid and for the more general deformed urban grids.

The algorithm works in a similar way to the simple urban grid algorithm. Each intersection is visited and is partitioned according to the guide in figure 6. In the figure there are multiple partitions for each grid. Each case must be tested for viability in alphabetical order (eg.4-A must be tested before 4-B and so on). The test for viability will be done by placing the lines as in the case to be considered and then seeing if the resulting polygons are still convex. The reason for the preferences is that each one partitions the intersection into the same number of or more polygons successively. If 4-A or 4-B is viable then the intersection will be partitioned into three parts. If neither is viable then 4-C might be and hence the intersection would be partitioned into four parts. In a simple urban grid the first case of each type of intersection would always be chosen.

The cases in figure 6 are all the possible partitions for an intersection. This is provided that no lines of the

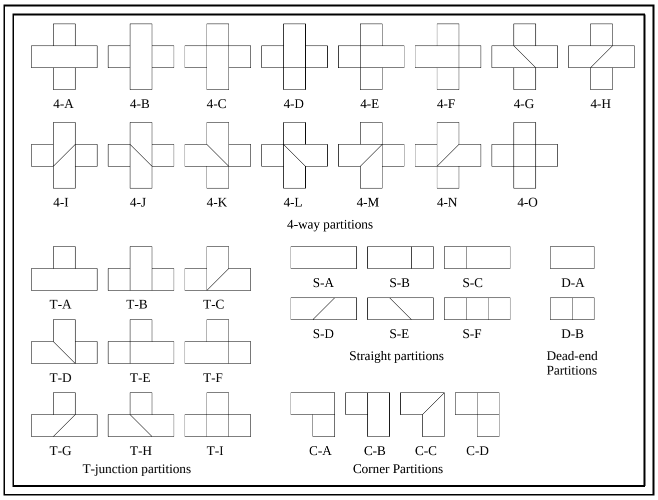

Figure 6: Partitions for algorithm 2

partition include the diagonals of the corridors. Note that all other intersection types except the 4-way have symmetric cases. There are thirty-one unique cases you have to test for if you simply rotate the symmetric cases and then apply the cases to the rotated intersection.

For many of the different cases in figure 6 the order does not matter. For example when partitioning a 4-way you can test A or B first because they both partition the intersection into two polygons. Testing for partition types C to N when doing the 4-way can be done in any order because all of these only partition the intersection into four polygons. Note that you don't actually have to test for the last case in each intersection type because the corridors and intersections are convex. The full paper shows that this algorithm always finds a convex map with the same or smaller number of convex polygons as the original algorithm.

6 Thoroughfare Based Algorithm

The thoroughfare based algorithm is presented in this section. This algorithm works by partitioning an entire thoroughfare at a time. This approach was taken because a thoroughfare is a polygon without holes and hence can be partitioned in polynomial time.

The algorithm works by partitioning all the horizontal thoroughfares first. The convex polygons in the partition that are intersections on their own are excluded from the partition and left to be partitioned in the vertical section. The vertical section works in much the same way but the intersections that were included in the partition in the horizontal section are excluded and the thoroughfares are split up accordingly. It does not turn out to be as good as the intersection based algo-

```
Algorithm 3
01   For each horizontal thoroughfare //Horizontal section
02       If first intersection is a T-junction
03           Then If the horizontal part of the thoroughfare localized at the first intersection
                     can be partitioned into one or two parts
04                 Then do not include first intersection in the thoroughfare

05       If last intersection is a T-junction
06           Then If the horizontal part of the thoroughfare localized at the last intersection
                     can be partitioned into one or two parts
07                 Then do not include last intersection in the thoroughfare

08       partition thoroughfare by some partitioning algorithm //preferably an optimal one
09       scan partition for intersections that are convex polygons on their own and
             exclude them //leave for vertical section
10       add partition to convex map

11   For each vertical thoroughfare //Vertical section
12       scan thoroughfare for intersections that are part of an intersection and break up
             thoroughfare accordingly
13       partition thoroughfare by some partitioning algorithm //preferably an optimal one
14       add partition to convex map
```

Figure 7: Partitioning a deformed urban grid - Thoroughfare based algorithm - Algorithm 3

rithm when implemented. Reasons for this are outlined in the full paper. The algorithm is given in figure 7.

Lines 3 and 6 may need some clarification. If you refer back to figure 6 we are treating the intersection as a vertical straight and testing for cases S-A — S-D. Lines 2 to 7 are small optimizations that use the idea that partitioning along the **straight** part of the T-junction will yield a better partition. This comes from the simple urban grid algorithm where the T-junction is always partitioned in this way. If this was not done then the vertical thoroughfare that crosses the intersection would always be cut in two. Unless the intersection could only be included in the partition of the horizontal thoroughfare on its own.

It was quite difficult to decide how many optimizations could be made for this algorithm. The only optimization made on this algorithm is the one mentioned above. This is detecting the T-junctions at either end of the thoroughfare to be partitioned. This was chosen too keep in line with the heuristic assumption of the intersection based algorithm - that if it partitions the simple urban grid optimally then it partitions the deformed urban grid well. It is easy to verify that making this small optimization enables this algorithm to do this.

It seems that the algorithm used in partitioning each thoroughfare should preferably be an optimal one for simple polygons (such as the algorithm in [Kei85]). This could, however, cause problems as optimally partitioning a horizontal thoroughfare might mean that a vertical thoroughfare which intersects that horizontal thoroughfare cannot be optimally partitioned – a decision about how some part of that thoroughfare must be partitioned has already been made. Some way of going back on previous decisions is required. This could mean that the overall partition is not optimal. Thus it is possible that when we are partitioning the horizontal thoroughfare we would like the lines of the partition to be as *vertical* possible. This could result in more intersections, that are convex polygons on their own, being available for inclusion in the partitions of the vertical thoroughfares.

The optimal algorithms are quite complicated and it was thought that they would be pretty hard to implement and time was limited so a reasonable approximation algorithm was derived. The algorithm is described in the full paper.

7 Evaluation

In the full paper the algorithms are fully evaluated. Details of test data, a complete set of results and more complete discussion are given.

The intersection based algorithm performed as expected and is the best algorithm developed so far to partition a DUG. Compared with the original algorithm an average improvement of 12% was achieved. This is pretty good considering that the original algorithm could possibly find the optimal solution.

The thoroughfare algorithm performed pretty well against the original algorithm though not as well as expected. The average improvement was 10% over the original algorithm in the general case. This could potentially be improved if an optimal partitioning algorithm was used for the thoroughfares.

The full paper gives a theoretical comparison between these algorithms and the optimal solution. This shows that the algorithms find a convex map with at most twice as many polygons as in the optimal convex map (give or take a few constant factors).

8 Conclusion

The deformed urban grid is the most general arrangement of polygons considered so far in the work done by Sanders in axial line placement. Recall that axial line placement is the placement of lines across all adjacencies in an arrangement of polygons. This research has lessened the number of polygons in the convex map of a deformed urban grid and hence decreased the number of adjacencies. This means that fewer axial lines will have to be placed and hence could increase the efficiency of any algorithm to do this. So this work makes a significant contribution to the automation of *space syntax*.

Any future work done on deformed urban grids should not try to find a better heuristic algorithm but rather to find an optimal algorithm if one does indeed exist. The author believes that the algorithms presented in this paper are good approximations. Any attempt at finding an optimal solution should try to combine these two approaches. This is discussed in more detail in the full paper.

This research has gone a long way to answering the urban grid problem The ideas presented in this work will hopefully aid greatly in any further research on deformed urban grids.

References

[Buk00] D. Bukovska. *Partitioning a Deformed Urban Grid*. Honours research report, School of Computer Science, University of Witwatersrand, October 2000.

[GN93] L. Gewali and S. Ntafos. Covering grids and orthogonal polygons with periscope guards. *Computational Geometry: Theory and Applications*, 2:309–334, 1993.

[Kei85] J. M. Keil. Decomposing a polygon into simpler components. *SIAM Journal of Computing*, 14:799–817, November 1985.

[KS98] J. M. Keil and J. Snoeyink. On the time bound for convex decomposition of simple polygons. In Mike Soss, editor, *Proceedings of the 10th Canadian Conference on Computational Geometry*, pages 54–55, Montréal, Québec, Canada, 1998. School of Computer Science, McGill University.

[KS02] G.D. Konidaris and I.D. Sanders. *Axial Line Placement in Deformed Urban Grids*. Technical Report TR-Wits-CS-2002-4, School of Computer Science, University of the Witwatersrand, April 2002.

[Lin82] A. Lingas. The power of non-rectilinear holes. In *The 9th International Colloquim on Automata, Languages and Programming*, number 140 in Lecture Notes in Computer Science 140, pages 369–383, New York, 1982. Springer-Verlag.

[MS98] H. Martini and V. Soltan. Minimum convex partition of polygonal domains by guillotine cuts. *Discrete Computational Geometry*, 19:291–305, 1998.

[San00] I. D. Sanders. Placing axial lines in urban grids. *South African Computer Journal*, (26):145–153, 2000.

[San02] I.D. Sanders. *The Axial Line Placement Problem*. PhD Thesis, Department of Computer Science, University of Pretoria, 2002.