Partitioning a Deformed Urban Grid

Leonard Hagger Ian Sanders

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

Abstract

Partitioning a deformed urban grid has applications in town planning, circuit design and computer vision. This research arises from the automation of *space syntax*. In *space syntax* patterns of architectural space are described and analyzed. One aspect of space syntax is to place axial lines upon the grid and in order to do this you need to partition it first. An urban grid is a polygon with holes that is arranged into corridors which begin and end with intersections that are convex quadrilaterals. There are currently two algorithms that create a convex map of the deformed urban grid. These algorithms are very similar and would find the same convex map of any given deformed urban grid. In this research I present two different algorithms to solve this problem. Both are shown to find a significantly better convex map of the deformed urban grid. The first of these two is an intersection based algorithm. The second algorithm is based upon the two previous algorithms but it uses a finer partitioning algorithm to partition thoroughfares. The intersection based algorithm runs in $O(n)$ time. The complexity of the thoroughfare based algorithm depends upon the partitioning algorithm used to partition thoroughfares.

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 [1982] showed the problem of partitioning polygons with holes to be NP-hard so special cases are of interest, especially those that are suited to specific applications. This is the case with deformed urban grids (DUGs). This application is town planning and placing axial lines on a town map which results from a need for greater architectural knowledge with a rapidly growing population. The problem of axial line placement is covered in detail in Sanders [2002]. Also the *deformed urban grid* is one of the most general cases of a polygon with holes considered so far in the context of axial line placement, so it is important to find a solution to this problem (if one does exist) and see how far we can relax the definition of a *deformed urban grid* and still get polynomial solutions. As the name suggests a deformed urban grid is a polygon with holes that looks like the layout of a network of roads (see fig1).

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 (or does not exist) these two algorithms take advantage of the arrangement to give a good approximation. These algorithms were implemented and compared to a previous algorithm [Sanders 2002].

2 The Deformed Urban Grid

In this section the deformed urban grid is defined. The original definition of the problem is given by Sanders Sanders [2002] and is based on the idea of a complete grid **?** – the *complete*

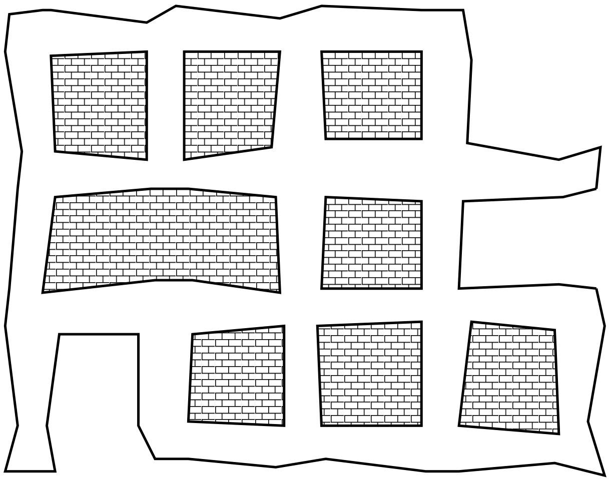

Figure 1: A deformed urban grid

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.

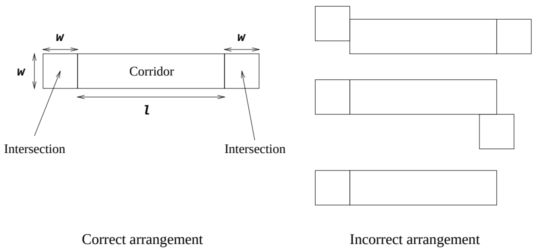

Figure 2: Urban grid arrangement

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 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.

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 **?** prove that

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

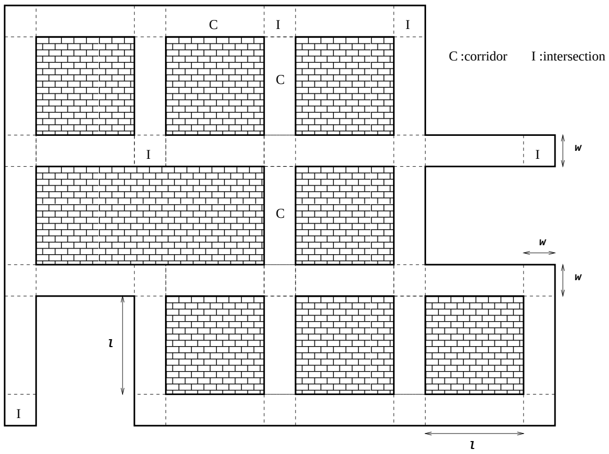

Figure 3: A simple urban grid

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.

3 Background

Many of the problems in the area of partitioning polygons are NP-hard. This is such an important area of research so a large amount of effort has been put into it. Much research has been done on specific cases and quite exhaustively.

The problem is shown to be NP-hard in Lingas [1982] as a result there have been approximation algorithms developed [Martini and Soltan 1998].

Keil [1985] 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 [1998] improved this algorithm to $O(n + N^2 \min(N^2, n))$ time. The authors show that the previous algorithm can be implemented using stacks in place of a search structure, which removes a $\log r$ factor from the running time. The algorithm does not use Steiner points. This is of interest because a DUG can be broken up into polygons without holes quite easily.

The original problem was presented by Sanders [2002]. This was as a result of an area the author was researching which is axial line placement which has applications in town planning. This is the process by which the smallest number of lines must be placed upon the shared segments between convex polygons. A simple algorithm was given in Sanders [2002] to partition a DUG. Bukovska [2000] 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 Sanders [2002] is referred to as “the original algorithm”.

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 defined in figure 4

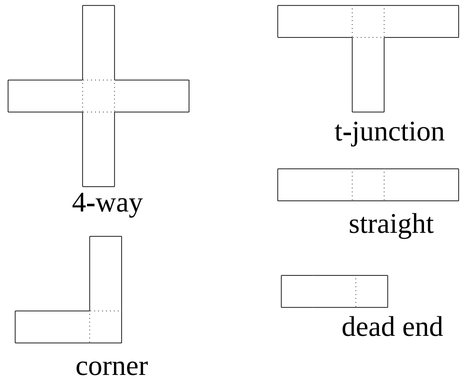

Figure 4: Types of intersections

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 is given in figure 5 and references figure 6.

Algorithm 1

```
For all intersections do the following
   If intersection is a 4-way
      Then partition as A
   If intersection is a T-junction
      Then partition as B
   If intersection is a corner
      Then partition as C
   If intersection is a straight
      Then partition as D
If intersection is a dead end
      Then partition as E
Create convex map by visiting each intersection in horizontal order.  Then in vertical order.
```

Figure 5: Optimally partitioning a simple urban grid - Algorithm 2

As you can see it is a very simple algorithm but it manages to partition the simple urban grid optimally. You may be thinking that partitioning the 4-way in the manner shown in figure 6 would affect the result but you get the same result by partitioning a 4-way by turning A on its side. You can see this if we simplify what we are looking at by reducing the grid to lines as in figure 7. In this figure the lines cannot cross. Observe what happens when we join the two vertical lines and split the horizontal line. The same number of lines is present, so the number of polygons in the convex map remains the same. It is immaterial how long the horizontal line is.

The reasons for the partitions of the other intersections should be obvious except B. It is important that the intersection is partitioned this way. See what happens in figure 7 when we split the T-junction in half. We land up with an extra line, ie. another polygon in the convex map. See Sanders [2002] for the formal proof of correctness.

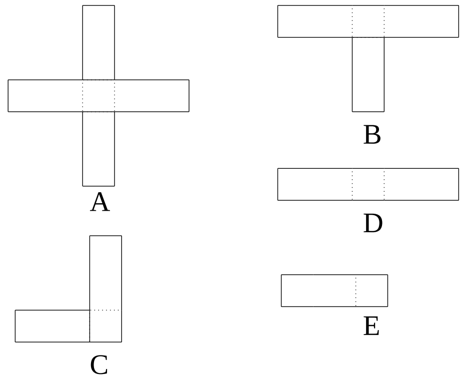

Figure 6: Partitions for the simple urban grid algorithm

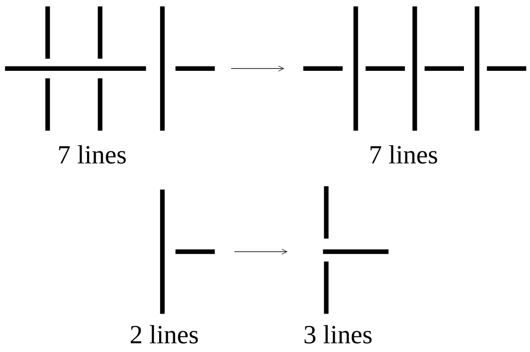

Figure 7: The line representation of a simple urban grid

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. This turns out to be a rather good assumption. The algorithm is given in figure 8 and references figure 9. Note that in figure 9 the partitions are drawn as if it was in a simple urban grid (with squares and rectangles) but we are working with a deformed urban grid. It was done this way for simplicity. In a simple urban grid the first case of each type of intersection would always be chosen.

The reason for the preferences is that each one partitions the intersection into the same or more polygons successively. See section 4.2 for more on reasons for these preferences. In line 22 the algorithm says to traverse the grid in horizontal and then in vertical order. This isn't completely necessary but is probably the simplest. If the grid is traversed in only one direction then many polygons need to be kept open and this makes things complicated. Traversing the other way only requires that one polygon be kept open.

The cases in figure 9 are all the possible partitions for an intersection. This is provided that no lines of the partition lie across 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 9 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

Algorithm 3

```
For each intersections the following
   If intersection is a 4-way
      Then test which 4-way applies in order of 4-A to 4-N
            partition intersection in the first way that applies
         if none apply then partition as 4-O
   If intersection is a T-junction
      Then test which T-Junction applies in order of T-A to T-H
            partition intersection in the first way that applies
         if none apply then partition as T-I
   If intersection is a corner
      Then test which corner applies in order of C-A to C-C
            partition intersection in the first way that applies
         if none apply then partition as C-D
   If intersection is a straight
      Then test which straight applies in order of S-A to S-E
            partition intersection in the first way that applies
         if none apply then partition as S-E
   If intersection is a dead end
      Then test if D-A applies
         if so then leave
         if not then partition as D-B
Create convex map by visiting each intersection
in horizontal order.  Then in vertical order.
```

Figure 8: Partitioning a deformed urban grid - Intersection based algorithm - Algorithm 2

intersections are convex.

Analysis

In this section the intersection based algorithm (algorithm 2) is shown to always find the same or a better solution than does the algorithm in Sanders [2002].

There are two reasons why algorithm 2 finds a better solution than the original algorithm:

1. algorithm 2 can partition diagonally across an intersection where the original algorithm cannot. There are many cases were partitioning diagonally across an intersection or corridor can yield a better solution.
2. algorithm 2 is conscious of the t-junction situation (see section 4.2 and figure 10)

The second point can be seen by considering the first algorithm . Algorithm 1 can only partition 4-ways of types A, B, C, D, E, F and O (see figure 9) and may choose a suboptimal one. Half of the cases are not considered. Further more it will consider 4-E and 4-F before considering 4-B which leads to a sub-optimal partitioning of the intersection. Note that if algorithm 1 finds that 4-A, 4-E and 4-F don't apply then it leaves the intersection for the vertical thoroughfare part.

6 Thoroughfare Based Algorithm

The thoroughfare based algorithm is presented in this section. It does not turn out to be as good as the intersection based algorithm when it was implemented but reasons for this will be outlined later. The algorithm is as follows:

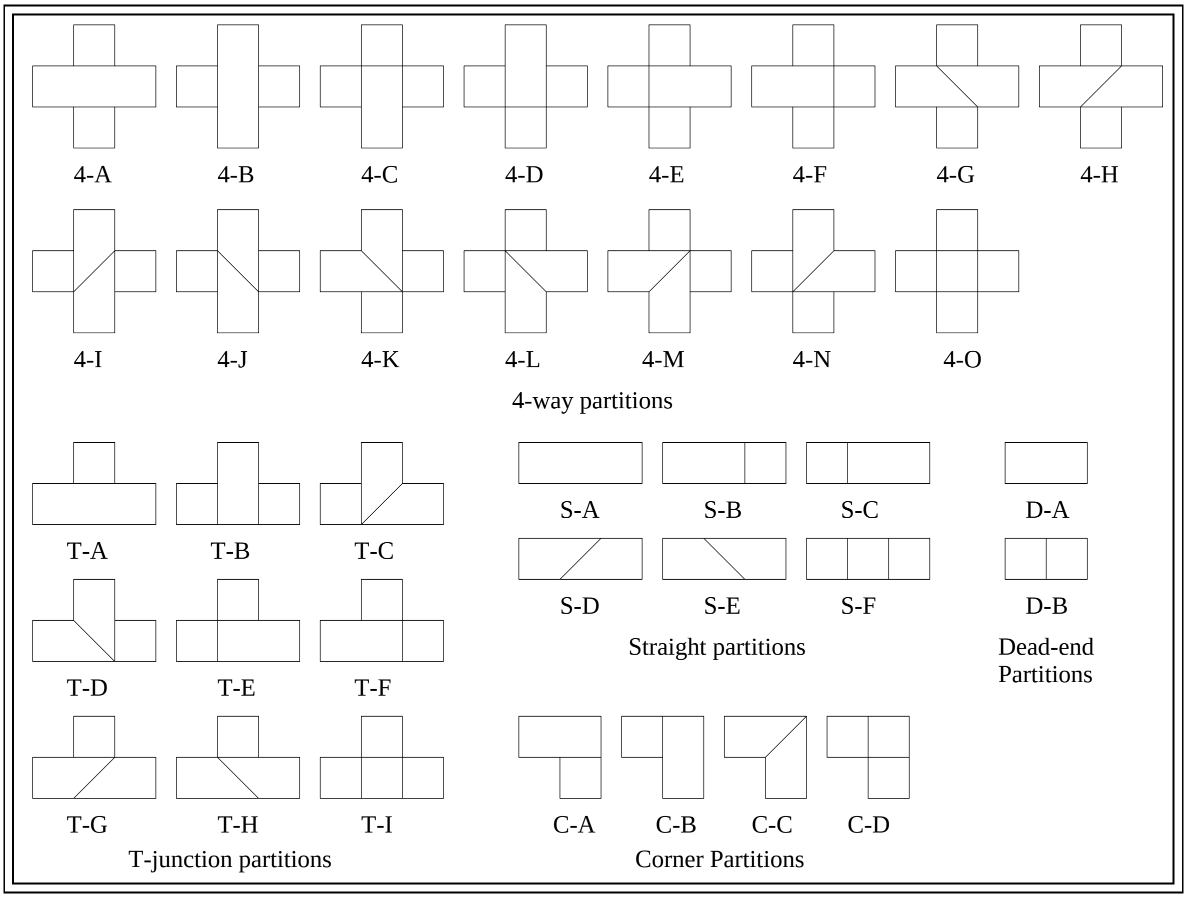

Figure 9: Partitions for algorithm 2

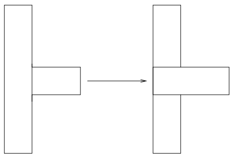

Figure 10: Comparison of the original algorithm to algorithm 2

Lines 3 and 6 may need some clarification. If you refer back to figure 9 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 uses 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.

The partitioning algorithm used should preferably be an optimal one for simple polygons such as the algorithm in Keil [1985]. The problem with using an optimal partitioning algorithm is that there can be many different minimum partitions. Now the reason why this could be a problem is that one partition may "block" a vertical thoroughfare which could pass right through with another minimum partition with the result being one or more unnecessary polygons in the convex map. When we are partitioning the horizontal thoroughfare we would like the lines of the partition to be as *vertical* possible. This would result in more intersections that are convex polygons on their own were possible. This would give the vertical thoroughfares more to work with when being partitioned. Fortunately there may be a way to get the best of both worlds. In Keil and Snoeyink [1998] they discuss a method that gives a "bias" to certain types of diagonals.

Algorithm 3

```
For each horizontal thoroughfare //Horizontal section

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

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

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

For each vertical thoroughfare //vertical section

    scan thoroughfare for intersections that are part of an intersection and break up
        thoroughfare accordingly
    partition thoroughfare by some partitioning algorithm //preferably an optimal one
    add partition to convex map
```

Figure 11: Partitioning a deformed urban grid - Thoroughfare based algorithm - Algorithm 3

We can set then set the bias to the vertical lines.

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 appendix.

7 Empirical Results

In this section the results are presented for the implementation. Algorithms 2 and 3, and the original algorithm were implemented. For algorithm 3 (thoroughfare based) the thoroughfare partitioning algorithm in appendix **??** was used. The simulations ran for about six hours.

7.1 Test Data

The test data consisted of three classes of grids. Each class consisted of 100,000 different grids which was based upon the same simple urban grid (ie. one simple grid per class). The positions of the vertices where randomly determined for each grid in a class. The types of grids consisted of a long straight, a complete grid of dimensions 25×19 and a general arrangement of intersections and corridors. Each grid in each class was partitioned by each of the partitioning algorithms implemented. The results are given in tables 1, 2 and 3. Recall that the original algorithm is the one in Sanders [2002].

The data in tables 1, 2 and 3 show the maximum, minimum and average *differences* in the number of polygons in the convex maps produced by the indicated algorithms. At each step of the experimental testing a grid is generated and partitioned with all three algorithms. Then the differences in the number of polygons was calculated between all three of the partitions. For example look at the first row in table 1. The maximum value was the largest difference between the original and the intersection based algorithm for all the grids.

8 Evaluation

In this section each of the algorithms are evaluated.

8.1 Intersection Based Algorithm

The results for this algorithm were as expected. This algorithm out performs any other algorithm developed to solve the problem of partitioning a deformed urban grid. The results in the tables in section 7.1 show this quite clearly.

Comparing algorithm 2 to the original algorithm shows a large improvement. Look at the long straight class in table 1 it is seen that there is quite a large difference between the number of polygons in the intersection based algorithm compared to the original algorithm. The *minimum* difference between this and the original algorithm is *16.91%*! In the complete grid class there is a minimum difference of 8.02% and an average of 15.17%. The general case shows similar numbers. These differences are quite large and represent quite a significant improvement. Note that it is still possible for the original algorithm to find the same solution as any of the algorithms presented but clearly they do not come close even for 100000 different cases.

There are algorithms to partition polygons with holes and it would be interesting to see how they perform against this algorithm when applied to an urban grid [Martini and Soltan 1998]. But this would be a topic for further research. Clearly algorithm 2 outperformed algorithm 3. A small part of the heuristic used in the intersection algorithm was incorporated into the thoroughfare algorithm. This was so it could handle intersections better but it only handles one type of intersection, namely the T-junction. What is interesting is when partitioning the long straight. The average difference between the thoroughfare based partition and the intersection based partition is 2%. Since the long straight is just a long thoroughfare it would be expected that the thoroughfare based approach would be much better than the intersection based approach even if an approximation algorithm was used. The algorithm used divides up twice the amount of quadrilaterals than does the intersection based approach. This result is quite confusing. It would be interesting to see if an optimal convex polygon partitioning algorithm would increase this number by any significant amount.

8.2 Thoroughfare Based Algorithm

There were relatively few cases were the original algorithm outperformed this algorithm. In the general class of grids the average improvement was 9.96%. Since the general deformed urban grid is representative of most grids it is valid to assume that this will hold in general. One of the reasons for this improvement is the T-junction optimization.

It was quite difficult to decide how many optimizations could be made for this algorithm. The only optimization made on this algorithm was 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.

Difference between..	**Average**		**Maximum**		**Minimum**	
	number	**%**	**number**	**%**	**number**	**%**
Original - Intersection based	161	22.61	208	27.85	116	16.91
Original - Thoroughfare based	178	25.00	221	29.31	137	20.24
Intersection based - Thoroughfare based algorithm	-16	-2.00	11	2.04	-51	-9.85

Table 1: Differences in number of polygons in convex map for the long straight

Difference between..	**Average**		**Maximum**		**Minimum**	
	number	**%**	**number**	**%**	**number**	**%**
Original - Intersection based	118	11.42	162	15.17	80	8.02
Original - Thoroughfare based	31	3.00	93	8.78	-29	-2.86
Intersection based - Thoroughfare based algorithm	87	8.68	134	12.67	41	4.27

Table 2: Differences in number of polygons in convex map for the complete grid of dimensions 25×19

Difference between..	**Average**		**Maximum**		**Minimum**	
	number	**%**	**number**	**%**	**number**	**%**
Original - Intersection based	91	11.48	130	15.78	59	7.70
Original - Thoroughfare based	79	9.96	130	15.82	33	4.26
Intersection based - Thoroughfare based algorithm	12	1.68	59	11.90	-25	-3.62

Table 3: Differences in number of polygons in convex map for the general grid

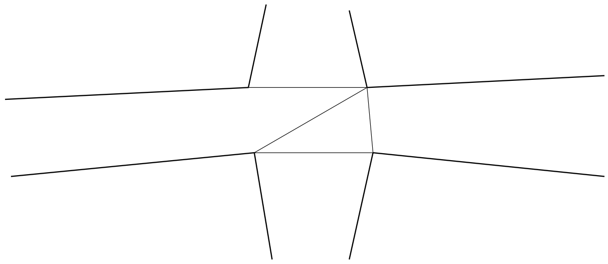

Figure 12: Suboptimal partition

There are other optimizations that can be made such as testing if any of the intersections have been partitioned as in figure 12[1] . Removing these triangles would enable the vertical part of the algorithm to partition the vertical thoroughfare that passes through this intersection. This was excluded because only the nature of a thoroughfare based approach was being studied. Making more and more optimizations to this algorithm would result in a combination of the thoroughfare and intersection based approaches and so the optimizations were kept to a minimum.

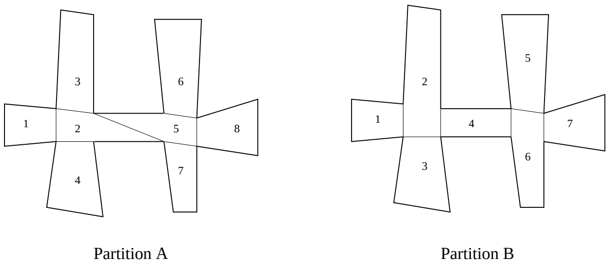

Figure 13: Suboptimal partition

Another reason for not making these optimizations is that there are cases were including ideas from an intersection based approach in a thoroughfare based approach is that it can result in a *worse* partition. In figure 13 there is one such case. Partition b is an optimized approach were as a is not. This raises a more important point that to gain an overall improvement a more **global** approach is needed.

The situation in figure 13 can happen without any optimizations. In fact this is another reason why Sanders' algorithm can find a better solution. Looking at the empirical results it is seen that there are very few cases were this happens. The only case were the difference is negative is in the complete deformed urban grid class where the minimum is -29 polygons. The reason why it is said that this doesn't happen very often is that the average is positive (3% compared to -2.86%) and this is the only class where this happens. This algorithm is hard to judge based on the results because there are cases were the algorithm finds a partition that is worse than original algorithm. But as it turns out this only happens in the complete case. The reason for this is that the thoroughfare based approach is particularly bad at partitioning 4-ways. The slight optimization for the T-junctions makes this approach good at partitioning this type of intersection.

9 New Algorithms vs. Optimal Solution

In this section I will be giving a theoretical comparison of the optimal solution to the two new algorithms. This is by no means a substitute for an empirical comparison but serves to give an

[1]Note that Sanders' algorithm does not have this problem and is one reason why it finds a better partition in some cases

idea of what the worst cases and best cases for the new algorithms could be. It must be noted that any lines in an optimal convex decomposition are between two reflex points or between a reflex point and a convex point. If two convex points are joined then we are dividing up a convex polygon.

9.1 Intersection Based Algorithm

The intersection based algorithm only partitions a thoroughfare from one side to the other. This is not always the optimal way of partitioning a thoroughfare. The partition may involve a line between two reflex points on the same side of the thoroughfare as in figure 14. The intersection based approach creates a partition with three polygons in it where the optimal solution has two.

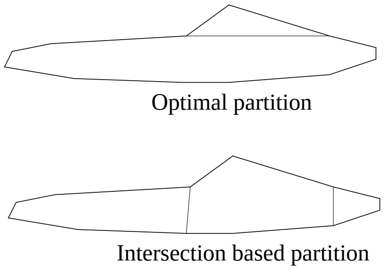

Figure 14: Partition on top of thoroughfare

If the more general case is considered (see figure 15) the intersection based algorithm adds up to one polygon per line drawn in the optimal solution. So if the number of polygons in the optimal convex map is $\mathcal{A}$ and the number of polygons in the intersection based convex map is $\mathcal{B}$ then $\mathcal{B} \leq 2 \times \mathcal{A} - 1$ (easily proved by induction). This relation is only considering the partition of a single thoroughfare within the bounds of this specific case. Other problems happen when thoroughfares cross. This case is a problem for the thoroughfare based approach without an optimal convex polygon partitioning (OCPP) but for different reasons. Though the thoroughfare partitioning algorithm used in the experiments does suffer from this problem for the same reason.

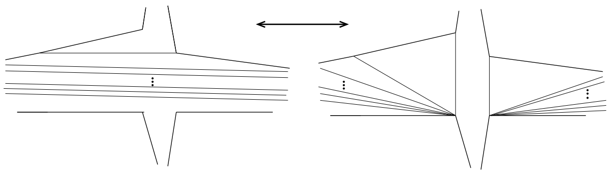

Figure 15: Many lines on top of a thoroughfare

Another problem with the intersection based approach is that the optimal partition may contain a line that spans many corridors or intersections. An example is given in figure 16. In this example we see that the optimal solution has three polygons in it where the approximation has five.

9.2 Thoroughfare Based Algorithm

A thoroughfare can be partitioned optimally such that many lines of the partition cross a single intersection. If an OCPP algorithm is used in the thoroughfare approach then this situation can occur. In figure 17 we see such a situation. If thoroughfare A is partitioned first then it is partitioned into one polygon and added to the convex map. This divides thoroughfare B in two resulting in two thoroughfares to be partitioned instead of one. This means that there will be more polygons in the final partition than in the optimal partition. There are two sub cases of this problem given below. These cases may not happen independently of each other but are discussed separately.

1. When the lines that cross the intersection start at the bottom of the thoroughfare and end at the top (or from left to right, depending on orientation)

2. When the lines that cross the intersection begin and end on the same side of the thoroughfare. This was the case discussed above for the intersection based approach.

In both cases similar things happen when the thoroughfare is split in two. The result of these splits is shown in figures 15 and 18. So if the number of polygons in the optimal convex map is $\mathcal{A}$ and the number of polygons in the thoroughfare based convex map is $\mathcal{B}$ then we can see from the figure that $\mathcal{B} = 2 \times \mathcal{A} - 3$ (easily proved by induction). Note that this relation only considers the case where two thoroughfares cross and only the mentioned conditions hold. If a situation like this occurs where there are many thoroughfares that cross it then the situation is actually *improved*. If the crossing thoroughfares are partitioned first then the main thoroughfare could get divided so that not so many polygons are created.

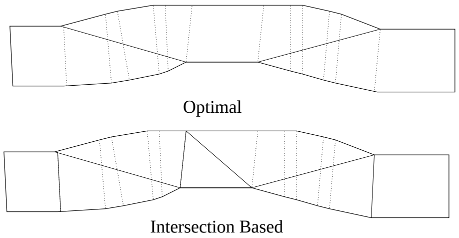

Figure 16: Many lines on top of thoroughfare

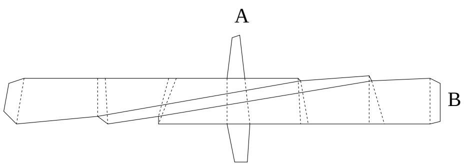

Figure 17: A partition line crossing many intersections

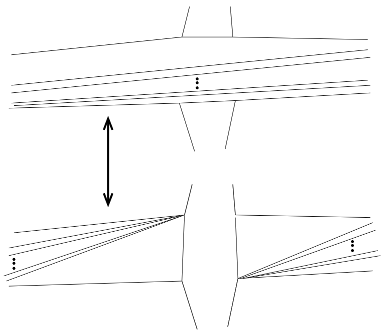

Figure 18: Many lines from bottom to top

10 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 an hence decreased the number of adjacencies. This means that less axial lines will have to be placed and hence increasing 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. I believe that the algorithms presented in this paper are good approximations. In section 9 evidence is presented that shows that the number of polygons in any partition found by the two algorithms is within twice the number in the optimal solution (though not verified). If an optimal algorithm does exist then it should combine the intersection based approach with the thoroughfare based approach. This approach can be in the form of treating the horizontal thoroughfare and the vertical thoroughfare as one polygon and then partitioning that. The problem with this approach is you may block off other thoroughfares that could yield a better partition. To solve this you could add more thoroughfares that cross this "cross" and so on. The problem with this is knowing when to stop and what thoroughfares should be included in the polygon.

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. Particularly with respect to finding an optimal partition. The problem of partitioning a polygon with holes has been shown to be NP-hard so finding the limits of partitioning polygons with holes is important. Just as important is finding good approximation algorithms to solve this and other special cases that arise. The author believes that this research has contributed greatly to the problem of partitioning polygons and hopefully will benefit many researchers in their en-devour to go were no deformed urban grid has gone before.

A Thoroughfare partitioning algorithm

The algorithm described here was the algorithm used in the thoroughfare partitioning stage of algorithm 3. It is given in fig 19. It starts on the "left hand" side of the thoroughfare and adds the beginning intersection to the working polygon. The algorithm then continues by adding a triangle formed by the two right most vertices of the working polygon one of the vertices left most that has not been considered. If the addition of the triangle causes the working polygon to be non convex then the triangle is removed and the working polygon is added to the partition and the triangle is made the new working polygon. If the working polygon remains convex then the next triangle is considered. This is done in two passes, they differ in the manner that the left most unconsidered vertex is chosen. The algorithm is $O(n)$.

```
00  set Count1 to 0
01  set Count2 to 0
//First Pass
02  set WorkingPolygon to quad(polygon[1], polygon[2], polygon[n − 1], polygon[n])

03  For i = 2 to n/2 − 2 //n is always even
04     add triangle(polygon[i], polygon[n − i], polygon[n + 1 − i]) to WorkingPolygon
05     If WorkingPolygon is not convex Then
06        delete triangle(polygon[i], polygon[n − i], polygon[n + 1]) from WorkingPolygon
07        add WorkingPolygon to Partition1
08        increment Count1
09        set WorkingPolygon to triangle(polygon[i], polygon[n − i], polygon[n + 1 − i])
10     add triangle(polygon[i], polygon[i+1], polygon[n-i]) to WorkingPolygon
11     If WorkingPolygon is not convex Then
12        delete triangle(polygon[i], polygon[i + 1], polygon[n − i]) from WorkingPolygon
13        add WorkingPolygon to Partition1
14        increment Count1
15        set WorkingPolygon to triangle(polygon[i], polygon[i + 1], polygon[n − i])
    //End For

16  add quad(polygon[n/2 − 1], polygon[n/2], polygon[n/2 + 1], polygon[n/2 + 1]) to WorkingPolygon
17  If WorkingPolygon is not convex Then
18     delete quad(polygon[n/2 − 1], polygon[n/2],polygon[n/2 + 1], polygon[n/2 + 1]) from WorkingPolygon
19     add WorkingPolygon to Partition1
20     increment Count1
21     set WorkingPolygon to quad(polygon[n/2 − 1], polygon[n/2], polygon[n/2 + 1], polygon[n/2 + 1]])
22  add WorkingPolygon to Partition1
23  increment Count1
//Second Pass
24  set WorkingPolygon to quad(polygon[1], polygon[2], polygon[n − 1], polygon[n])

25  For i = 2 to n/2 − 2 //n is always even
26     add triangle(polygon[i], polygon[i + 1], polygon[n + 1 − i]) to WorkingPolygon
27     If WorkingPolygon is not convex Then
28        delete triangle(polygon[i], polygon[i + 1], polygon[n + 1 − i) from WorkingPolygon
29        add WorkingPolygon to Partition2
30        increment Count2
31        set WorkingPolygon to triangle(polygon[i], polygon[i + 1], polygon[n + 1 − i])
32     add triangle(polygon[i + 1], polygon[n − i], polygon[n + 1 − i]) to WorkingPolygon
33     If WorkingPolygon is not convex Then
34        delete triangle(polygon[i + 1], polygon[n − i], polygon[n + 1 − i]) from WorkingPolygon
35        add WorkingPolygon to Partition2
36        increment Count2
37        set WorkingPolygon to triangle(polygon[i + 1], polygon[n − i], polygon[n + 1 − i])
    //End For

    Repeat lines 16 to 23 but replace Partition1 with Partition2 and Count1 with Count2

38  If Count1 < Count2
39     Then return Partition1
40     Else return Partition2
```

Figure 19: Partitioning thoroughfare into convex polygons - Approximation algorithm

References

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

[Keil and Snoeyink 1998] 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.

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

[Lingas 1982] 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.

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

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