Chapter 5

Placing axial lines with arbitrary orientation to cross the adjacencies between orthogonal rectangles

5.1 Introduction

In Chapter 3 the aim and focus of this thesis are discussed. This chapter addresses one of the questions raised in Chapter 3 – the problem of finding the minimum number of longest axial lines with arbitrary orientation which cross all of the adjacencies of a collection of adjacent orthogonal rectangles (***ALP-ALOR*** Axial Line Placement – Arbitrary Lines and Orthogonal Rectangles). In Chapter 4 ***ALP-OLOR*** (Axial Line Placement – Orthogonal Lines and Orthogonal Rectangles) was shown to be NP-Complete by a transformation from biconnected planar vertex cover to stick diagram and then to ***ALP-OLOR***. In this chapter, a similar process is used to show that ***ALP-ALOR*** is NP-Complete. In the next chapter of this thesis (Chapter 6) the NP-Completeness proof is extended to show that ***ALP-ALCP*** (Axial Line Placement – Arbitrary Lines and Convex Polygons) is also NP-Complete.

In the remainder of this chapter (***ALP-ALOR***) is discussed in more detail. The problem is restated in detail in Section 5.2 below. In Section 5.3 ***ALP-ALOR*** is shown to be NP-Complete using a similar transformation to that discussed in Chapter 4 for ***ALP-OLOR***. Then, because ***ALP-ALOR*** is NP-Complete, some ideas for heuristics to find approximate axial maps are discussed in Section 5.5. In Section 5.7 some ideas for future research are briefly discussed.

5.2 Statement of the Problem

ALP-ALOR can be formally stated as follows:
Given a number of adjacent, orthogonally-aligned rectangles find the fewest axial

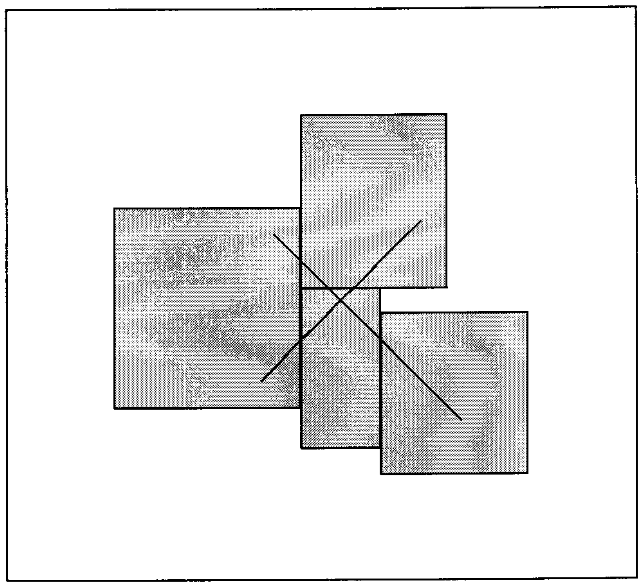


Figure 5.1: An example of the problem

lines (line segments), contained wholly inside the rectangles, required to cross all of the boundaries shared between adjacent rectangles. An additional requirement is that each axial line should cross as many of the shared boundaries as possible – a *maximal* axial line.

As in Chapter 4 for ***ALP-OLOR***, depending on how the problem is considered there are two similar but distinct problems which can be addressed – adjacencies can be crossed more than once but every adjacency must be crossed at least once; and any adjacency must be crossed exactly once. In this thesis only the first variation is addressed. Figure 5.1 shows an example of this.

The decision problem can then be stated as below.

ALP-ALOR
Instance: A collection of orthogonal rectangles $R_1 \ldots R_n$, where each R_i is adjacent to at least one other rectangle, and a positive integer $O \leq 4n$.
Question: Is there a set P of (possibly non-orthogonal) axial lines where each axial line is maximal in length, each axial line is contained wholly within the rectangles, each adjacency is crossed at least once by the axial lines in P and $|P| \leq O$?

In section 5.3 ***ALP-ALOR*** is shown to be NP-Complete.

5.3 Proving the problem is NP-Complete

In Chapter 4 it is shown that the problem of finding the minimum number of maximal *orthogonal* axial lines to cross all of the adjacencies in a configuration of adjacent orthogonal rectangles (***ALP-OLOR***) is NP-Complete. The proof is accomplished through a transformation from ***biconnected planar vertex cover***. This transformation is accomplished by mapping vertices in a biconnected planar graph to choice axial lines. In this mapping an edge between two vertices represents an adjacency which is crossed by two choice axial lines.

The transformation is done in two steps. First, a biconnected planar graph is transformed to a stick diagram where each vertex in the original graph is mapped to a horizontal line and each edge in the original graph is mapped to a vertical line which is crossed by the two horizontal lines which represent the two vertices to which the edge is incident. The problem then becomes that of choosing the minimum number of horizontal lines to cross all of the vertical lines. ***Stick diagram*** is thus NP-Complete – if it is possible to determine in polynomial time which of the set of horizontal lines cross all of the vertical lines in the stick diagram then it is possible to solve ***biconnected planar vertex cover*** in polynomial time. Finding the minimum set of horizontal lines is equivalent to finding the minimum vertex cover of the original graph. Second, the stick diagram is represented as a collection of adjacent orthogonal rectangles and horizontal axial lines crossing all of the adjacencies in the collection of rectangles. These horizontal axial lines are of two types "essential axial lines" which are the only axial lines to cross a particular adjacency and "choice axial lines" where a number of axial lines (none of which are essential) cross some adjacency. Not all of the choice axial lines are necessary to cross all of the adjacencies in the collection of rectangles. If it is possible to determine in polynomial time which of the set of choice axial lines cross all of the adjacencies in the diagram then it is possible to solve ***stick diagram*** in polynomial time – finding the minimum set of choice axial lines is equivalent to finding the minimum set of horizontal lines. Thus ***ALP-OLOR*** has been shown to be NP-Complete.

In this chapter the fact that ***stick diagram*** is NP-Complete is used to show that ***ALP-ALOR*** is also NP-Complete. This is done by transforming an instance of ***stick diagram*** to an instance of ***ALP-ALOR***. Once again this is accomplished by using "choice units" although the units used here are somewhat different to those used in Chapter 4 and different choice axial lines with arbitrary orientation are generated. Note that, in the remainder of this chapter any reference to an axial line should be taken to mean an axial line which is not necessarily orthogonal.

Theorem 5.3.1 *ALP-ALOR is NP-Complete*

Proof

Clearly ***ALP-ALOR*** is in NP. Given a set of axial lines with arbitrary orientation it

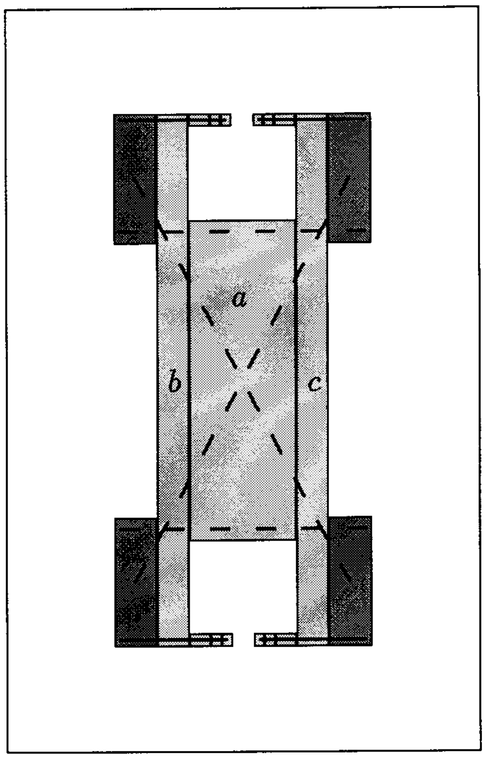


Figure 5.2: The Canonical Choice Unit which produces choice axial lines with arbitrary orientation

is possible to check in polynomial time that each adjacency has been crossed by at least one axial line.

Now transform ***stick diagram*** to ***ALP-ALOR***.

A collection of rectangles which create choice axial lines with arbitrary orientation can be represented by a canonical choice unit, *ccu*, shown in Figure 5.2. In this ccu, the adjacencies between the middle rectangle a and the rectangles b and c can be crossed by four "sets" of axial lines with arbitrary orientation (the upper, lower and two diagonal sets). Figure 5.2 shows as a dashed line a representative axial line from each of the four sets. Only one of these axial lines is actually necessary to cross the adjacencies between rectangles a, b and c. All the other adjacencies are crossed by the axial lines which originate in the "horns" of the ccu. These axial lines *do not* have to be horizontal but the size and position of the rectangles in the horns means that the axial lines are restricted to a small range of different slopes. Scaling of the canonical choice unit does not change the fact that it can/does produce choice axial lines.

The transformation proceeds by replacing each vertical line in the stick diagram by a ccu of an appropriate size. The horizontal lines which cross through the vertical line are represented by a subset of the choice axial lines of the ccu. It is then

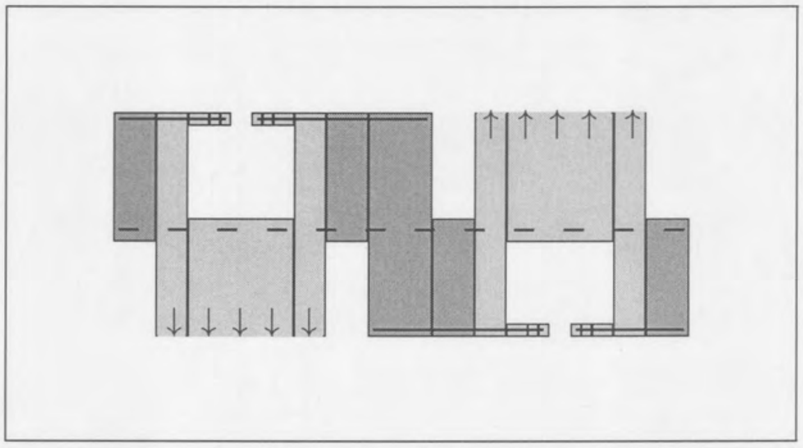


Figure 5.3: Connecting the upper portion of one ccu to the lower portion of the next

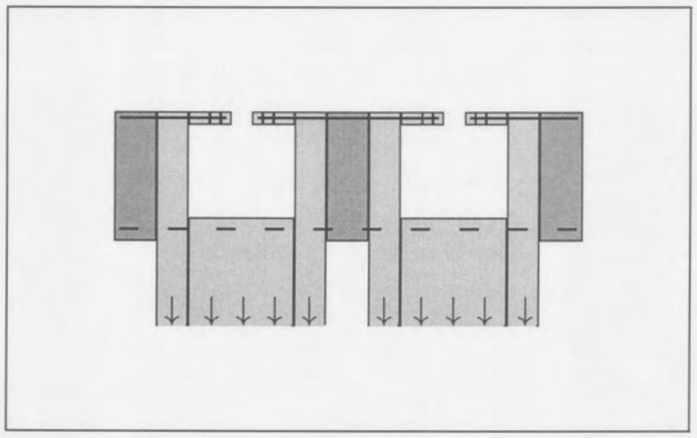


Figure 5.4: Connecting the upper portions of two ccus

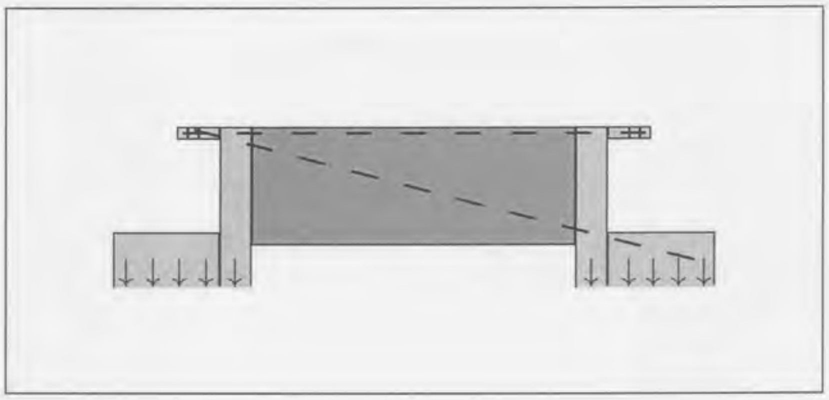


Figure 5.5: Possible rays from a "horn"

necessary to show that these canonical choice units can be joined together in a fashion which maintains the relation between the horizontal lines in the stick diagram and the choice axial lines in the configuration of adjacent rectangles. The situation here is somewhat different from Chapter 4 for ***ALP-OLOR***. There the fact that the choice and essential axial lines had to be horizontal could be used to control the stopping or continuing of axial lines. In this case the size and length to breadth ratio of the ccu's and the connecting rectangles are more critical.

As in Chapter 4, there are four ways in which horizontal lines could cross through successive vertical lines. (Actually there are only two ways, each with a vertical reflection.) These are

- the horizontal line could be the upper (lower) line through one vertical line and the upper (lower) line through the next vertical line,
- the horizontal line could be the upper (lower) line through one vertical line and the lower (upper) line through the next vertical line.

It must be shown that in each of these cases it is possible to connect two ccu's in such a fashion that the choice is preserved.

These cases are now considered in turn.

- *upper to lower*
  Here the two ccu's are connected by placing a rectangle of appropriate size into the position indicated in Figure 5.3. This "connecting rectangle" ensures that the upper choice axial line through the left ccu and the lower choice axial line through the right ccu are the same choice axial line. After adding the connecting rectangles, the adjacencies between rectangles a, b and c in each ccu are still only crossed by choice axial lines. The sets of choice axial lines which cross each ccu from the bottom left to the top right (and top left to bottom right) are still possibilities for crossing the adjacencies between

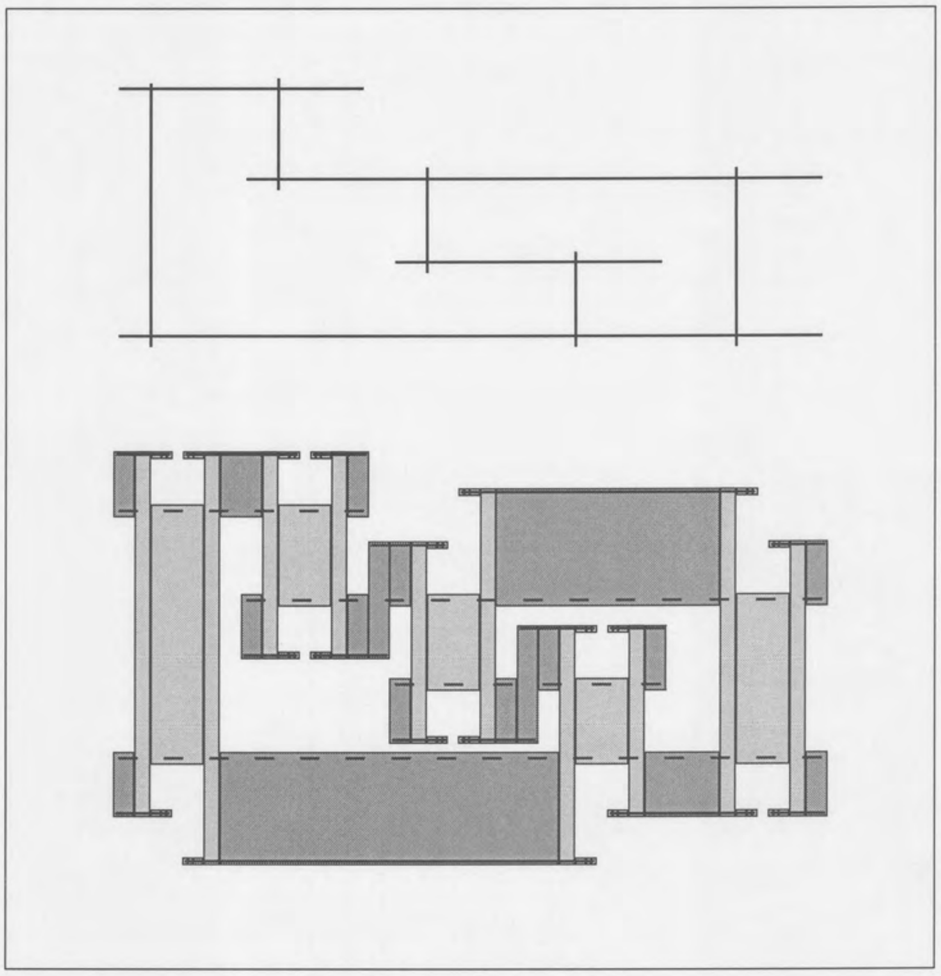


Figure 5.6: An example of converting a stick diagram to a collection of adjacent rectangles

a, b and c in each ccu but there are longer axial lines which cross the same adjacencies so axial lines from these sets will not be in the final set of axial lines for the collection of rectangles. The axial lines from the horns in the left ccu cannot be extended to cross the adjacencies between a, b and c in the right ccu. The only axial line which could be extended into the right ccu in this case is the upper choice axial line from the lefthand ccu. After connecting the two ccu's, the adjacencies in both ccu's which were only crossed by choice lines are still only crossed by choice lines. All other adjacencies are crossed by essential axial lines. In this sense, choice is maintained.

- *lower to upper*
  This is a mirror image of the case above about the x-axis, see Figure 5.3.
- *upper to upper*
  In this case no connecting rectangles are required, it is enough to simply merge the appropriate connector rectangles. This is shown in Figure 5.4. Again the adjacencies previously crossed only by choice lines are still only crossed by choice lines and the essential axial lines cross all other adjacencies.
- *lower to lower*
  Again this case is a mirror image of the case above, see Figure 5.4.

A potential problem could arise when a number of ccu's are joined together to make up a collection of adjacent rectangles which represents a stick diagram. This could happen because it is possible for the axial line through the rectangles in one of the horns to cross one of the adjacencies previously crossed only by choice axial lines. This can be seen in Figure 5.5 where the range of axial lines from the horn in the left ccu includes axial lines which cross an adjacency which was previously only crossed by choice axial lines. This problem can be overcome by some very simple changes to the structure of the ccu. The horn could be made longer – this would be accomplished by keeping the two smaller rectangles in the horn the same size and making the longer rectangle still longer. This would have the effect of reducing the angle at which axial lines could leave the horn and thus no essential axial lines could cross the adjacency previously crossed only by choice axial lines. The horn could also be made narrower by changing the vertical extent of the rectangles in the horn. This would have the same effect of reducing the angles at which axial lines could leave the horn. Other ways of addressing the problems could be to increase the height of the b and c type rectangles and move the horns higher up these rectangles. In this case the range of angles at which axial lines could leave the horn is kept constant but the required range is increased. A similar approach would be to make the connector rectangle shorter.

The transformation from ***stick diagram*** to ***ALP-ALOR*** is thus accomplished by inserting an appropriately sized ccu for each vertical line and then joining these up by using the appropriate connecting rectangles working from the leftmost to the rightmost ccu. ccu's should also be scaled as necessary to ensure that the axial lines from the horns (which are essential) cannot interfere which the issue of choice. Figure 5.6 shows the final result after taking an instance of ***stick diagram*** and converting it to an instance of ***ALP-ALOR***. Note that in this diagram some ccu's have been scaled to ensure that the essential axial lines from the horns do not interfere with the choice.

It is now necessary to show that there is a solution to ***stick diagram*** if and only if there is a solution to ***ALP-ALOR***. The construction of the collection of adjacent rectangles from the stick diagram changes the horizontal lines in the stick diagram to choice axial lines in the collection of rectangles. It also introduces 4 essential axial lines for every ccu added (note, some of these axial lines could be shared across ccu's – see Figure 5.6 for an example of this). These essential axial lines must be in the final solution to ***ALP-ALOR***. Suppose there is a solution for ***stick diagram***, i.e. there exists a set of lines H' such that $|H'| \leq S$. Then there must be a solution P to ***ALP-ALOR*** with $|P| = |H'| + 4|U| - p$, where U is the number of ccu's and p is the number of shared essential axial lines. This is because the essential axial lines must be in P and the choice axial lines which correspond to the selected horizontal lines in S must also be in P. Conversely if there is a solution P to ***ALP-ALOR*** then there must be a solution $S = P - \{e \mid e$ is an essential line in $P\}$.

This transformation can clearly be done in polynomial time – each vertical line is visited twice, once when it is replaced by a ccu and a second time when it is connected to the ccu(s) to its right in the stick diagram. If the stick diagram can be drawn then a configuration of ccu's can be drawn by scaling the ccu's to be the same size as the vertical lines that they represent. The ccu's (and their connecting rectangles) can thus be drawn as a non-overlapping collection of adjacent rectangles – an instance of ***ALP-ALOR***.

ALP-ALOR is thus NP-Complete.

$\square$

The result which has been proved above can be extended to axial lines with arbitrary orientation crossing the adjacent edges between convex polygons ***ALP-NLCP*** – rectangles are a special case of convex polygons. This is discussed in more detail in Chapter 6.

The fact that ***ALP-ALOR*** is NP-Complete means that in general only exact solutions for small problem instances can be found and that investigating heuristics is a worthwhile area of research. In Section 5.5 of this thesis some ideas for possible heuristics to be used in finding approximate solutions for ***ALP-ALOR*** are discussed. Fully investigating these heuristics is outside of the scope of this thesis but they are presented to give the reader some idea of approaches which could be taken.

A problem which arises, whether an exact solution is being calculated or a heuristic is being applied to find an approximate solution, is to be able to efficiently decide whether an axial line can be placed to cross a given set of adjacencies. The next section of this thesis (Section 5.4) addresses this problem before heuristics which are assumed to use the idea proposed here are presented.

5.4 Determining whether axial lines can be placed in chains of adjacent rectangles

In Chapter 4 which deals with placing orthogonal lines to cross the adjacencies between orthogonal rectangles, it is very easy to determine whether an axial line crosses a number of adjacencies. If the adjacent rectangles which give rise to the adjacencies are treated as a chain of rectangles (i.e. the rectangles are adjacent and each rectangle has at most one left neighbour and one right neighbour) and if there is an overlap in the y-values of the adjacencies then an axial line can be placed to cross those adjacencies. Note that, as discussed before, an axial line is actually a representative of a set of lines all of which fall within the same range of y-values and cross the same adjacencies between rectangles. In the algorithms discussed in Chapter 4 (Sections 4.5, 4.7.2.3 and 4.7.2.4) each time a line is extended forwards or backwards in effect an axial line is being extended through a chain of rectangles.

In the case of arbitrary lines and orthogonal rectangles, a single axial line could cross both vertical and horizontal adjacencies and these thus have to be considered at the same time. This makes determining whether an axial line can be placed to cross all of the adjacencies in a "chain" of adjacent rectangles more complicated. Here a chain of rectangles is defined as a sequence of adjacent rectangles where each rectangle has at most one "incoming" neighbour (left, top or bottom) and one "outgoing" neighbour (right, top or bottom). In Figure 5.7 the rectangles 1, 2, 4, 5 and 7 can be said to make up a chain of rectangles. As can the rectangles 1, 2, 4, 6 and 8. At some stage of an algorithm which finds the axial lines for the configuration of rectangles in Figure 5.7 it might be necessary to determine whether an axial line can be placed to cross the adjacencies 1–2, 2–4, 4–5 and 5–7 in the chain of rectangles 1, 2, 4, 5 and 7. As can be seen in this example such a line can, in fact, be found – the difficulty is in deciding how to place the line or how to determine the range of possible lines. Clearly this could be accomplished by solving sets of simultaneous equations to find the lines with the minimum and maximum slopes which intersect the adjacencies. Another approach is to use the idea of visibility in polygons (as discussed in Chapter 2, Section 2.4.4.3.1).

The idea behind this approach is to convert the problem of determining whether an axial line can be placed to cross the adjacencies in a given chain of rectangles to the problem of determining partial visibility of two edges.

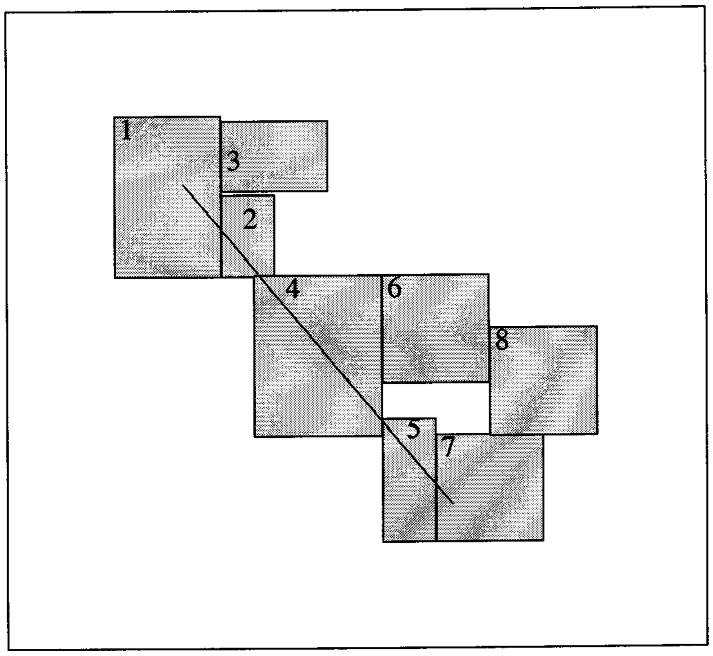


Figure 5.7: Placing a arbitrary axial line in a chain of rectangles

The process is as follows. Given a chain of rectangles, convert the chain into an adjacency polygon – an adjacency polygon is a polygon which uses the first and last adjacency in the chain of rectangles as two of the edges of the resultant polygon and remaining edges of the adjacency polygon are the edges or the parts of the edges in the remaining rectangles in the chain. Figure 5.8 shows the chain of rectangles discussed previously and its associated adjacency polygon. If it were possible for an axial line to be placed to cross all of the adjacencies in the chain of rectangles then there will be partial edge visibility between the two edges of the adjacency polygon which were the first and last adjacencies in the chain (edges a_f and a_l in Figure 5.8). Bilbrough and Sanders [1998] discusses an expected time linear algorithm for determining this partial visibility.

The algorithm can be converted to a linear algorithm by making use of an approach suggested by Avis *et al.* [1986] of generating the "Inner Convex Hull" of the adjacency polygon. This is done by finding the inner convex hulls of the top chain and the bottom chain respectively and using these to create the "reduced" top and bottom chains. See Figure 5.9 for an example of how this is accomplished. du Plessis and Sanders [2000] discuss in detail how the reduced adjacency polygon can be calculated and from that how partial edge visibility can be determined. Again, if it were possible for an axial line to be placed to cross all of the adjacencies in the chain of rectangles then there will be partial edge visibility between the two edges of the reduced adjacency polygon which were the first and last adjacencies in the chain (edges a_f and a_l in Figure 5.9). This approach is used in the heuristics

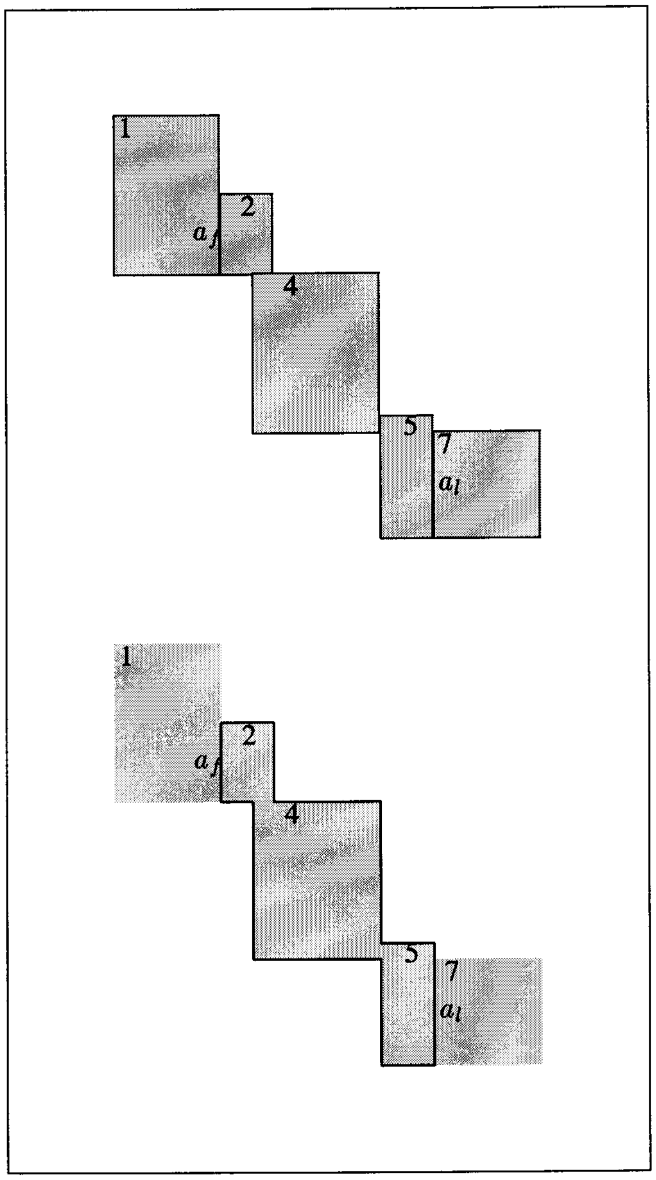


Figure 5.8: Converting a chain of rectangles into an adjacency polygon

discussed below.

5.5 Heuristics to find approximate solutions for *ALP-ALOR*

5.5.1 Overview

ALP-ALOR has been shown to be NP-Complete (Section 5.3) and thus in the general case finding a minimal solution could take an unreasonable amount of time. It is thus necessary to consider heuristics to find approximate solutions in reasonable time. In this section some ideas for heuristics are presented but the actual testing of the heuristics is beyond the scope of this thesis and thus will be future work which arises from this thesis.

5.5.2 Extending lines into all neighbours

In Section 4.5 an O(n^2) algorithm to return a non-redundant set of maximal orthogonal axial lines for the problem of placing orthogonal axial lines to cross the adjacencies between orthogonal rectangles (***ALP-OLOR***) was presented. This algorithm has four phases (after determining which rectangles are adjacent). It generates all the possible orthogonal axial lines which cross the adjacencies between rectangles; it determines which axial lines are essential (i.e. are the only lines which cross a particular adjacency); it removes any lines which only cross adjacencies crossed by the essential lines (redundant lines); and then it resolves the choice conflict. The resolving of the choice is done by repeatedly choosing the choice line which crosses the highest number of previously uncrossed adjacencies. One possible approaching to finding an approximation to ***ALP-ALOR*** is to adopt a similar approach to that used before. The main difference would be that the restriction on the axial lines being orthogonal would be removed – lines with arbitrary orientation would be allowed. This would mean that the algorithm would need to consider *all* neighbours rather than just the right neighbours of a particular rectangle. All of these adjacencies would have to be determined and thus the algorithm for calculating the adjacencies between the rectangles would be more complicated – the algorithm developed in Chapter 4 will no longer suffice. In addition, there seem to be two potential problems with adapting this algorithm to work in this case.

First, the algorithm for ***ALP-OLOR*** works by considering each rectangle in turn from left to right at each step attempting to extend any lines which cross into a rectangle from the left into its right neighbours. In the arbitrary case it would be necessary to determine if the incoming lines could be extended into all the current rectangle's neighbours. The effect of this change could mean that many more lines

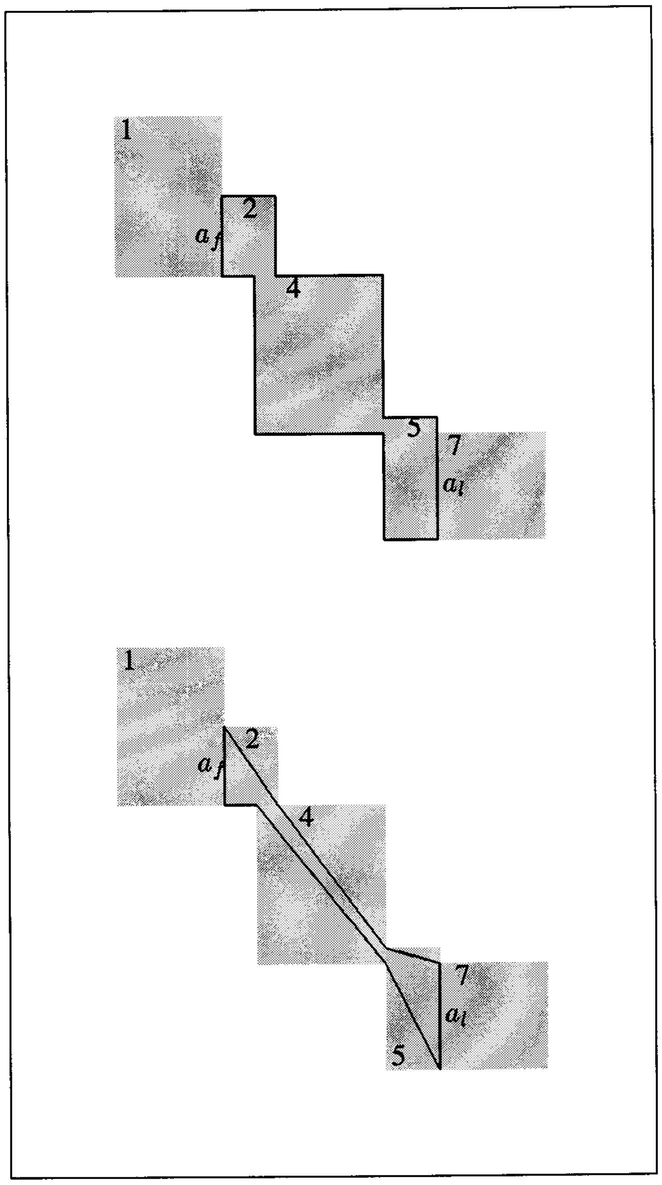


Figure 5.9: Converting an adjacency polygon into a reduced adjacency polygon

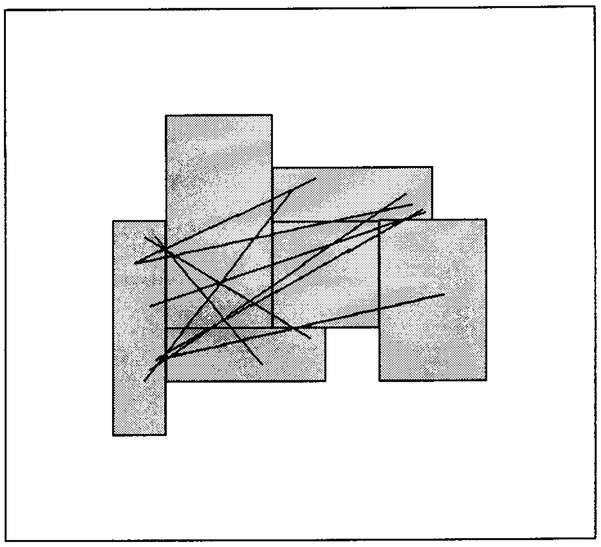


Figure 5.10: An example of using the algorithm extend lines into all neighbours

than are actually necessary are generated in the first phase of the algorithm and many of these lines would have to be removed later – either as redundant lines or in the phase of resolving choice. Figure 5.10 shows an example of the axial lines which would be generated by extending any axial lines coming into a rectangle into all of its neighbours.

Second, it seems that the lines which are generated are very dependent on the initial configuration of the rectangles and some lines which could be part of a minimal set of axial lines would not be generated by this slightly modified algorithm. Figure 5.11 shows some lines which would not be generated by this changed algorithm. This happens because, in this particular example, as each rectangle is considered it is possible to extend one (or more) of the incoming lines into each of its neighbours. There is never the necessity to start a new line.

Some way of addressing these problems would be required if this approach was to be used. Alternatively a different approach should be considered. Some other approaches are suggested in the following sections of this thesis.

5.5.3 Separating top-bottom and left-right lines

Another heuristic for finding an approximate solution for ***ALP-ALOR*** would be to calculate a solution in two passes – a horizontal sweep followed by a vertical sweep. In the horizontal sweep the algorithm works from the leftmost rectangle to the rightmost rectangle in the configuration. For each rectangle, any line which enters the rectangle from the left is checked to see if it can be extended into one or more of the rectangle's right neighbours. If any line can be extended, this new line is added to the set of lines and the line which was extended is removed. This is analogous

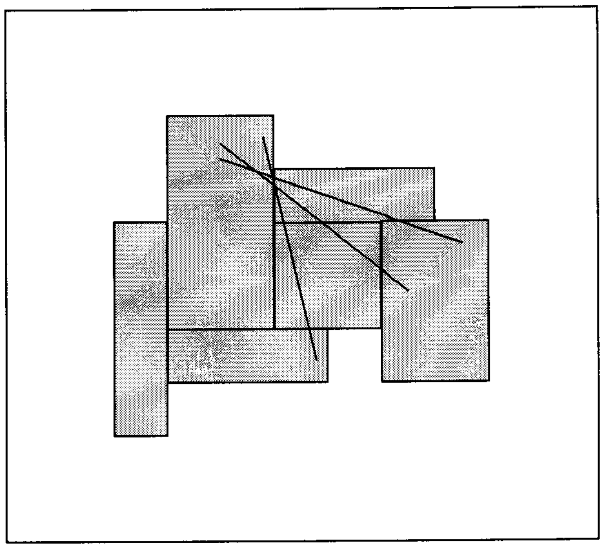


Figure 5.11: An example showing some lines which would not be generated

to the solution for ***ALP-OLOR*** except here the lines need not be orthogonal. The vertical sweep works from the highest rectangle (based on the y-value of the top edges of the rectangles) to the lowest rectangle (the rectangle which has the lowest y-value for its top edge). This second pass extends lines which enter the rectangle being considered from the top into its lower neighbours (if this can be done). Each of these passes would (as before) generate all possible lines (while obeying the left-right or top-bottom constraint); determine essential lines; remove redundant lines; and resolve any choice. These two passes then generate two sets of lines – a set which crosses the vertical adjacencies and a set which crosses the horizontal adjacencies. Combining the final two sets of lines would give an approximation to ***ALP-ALOR***. Figure 5.12 shows an example of the lines which would be generated when extending lines from left to right and top to bottom.

This approach seems unlikely to give an approximation which is close to the exact solution but at least it is more efficient in terms of the number of unnecessary lines than the approach of extending all possible lines (see Section 5.5.2).

An advantage of this approach is that the left-right and top-bottom neighbours can be determined separately and thus the algorithm from Chapter 4 can be used directly.

5.5.4 Longest Chains

A slightly different approach for finding an approximate solution for ***ALP-ALOR*** could be to attempt to find the longest possible axial line at any stage of the algorithm. Clearly to do this would require first generating all possible lines which is inefficient (see the argument in Section 5.5.2). An approach which goes some

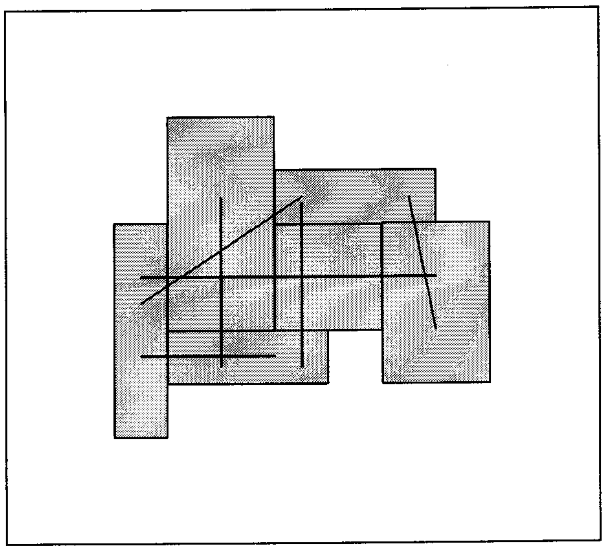


Figure 5.12: An example showing the lines which would be generated in a top-bottom and left-right manner

way towards using this idea is to identify "extreme" rectangles, generate the chains which include these rectangles and then successively choose the longest of such chains until all of the adjacencies have been crossed. The motivation for doing this is that lines which cross the adjacencies of these extreme rectangles could potentially cross a number of other adjacencies as well and so might be a good "greedy" choice. An algorithm to find an approximation based on this idea is sketched below.

1. Determine all of the adjacencies between the orthogonal rectangles. Store this information in an adjacency matrix.

2. Determine all of the extreme rectangles – those which do not have neighbours on both of the top and bottom and the left and right sides. In Figure 5.13 rectangles 1, 2, 5 and 6 are extreme rectangles (the extreme rectangles are more lightly shaded than the rectangles which are not characterised as extreme). Rectangle 3 has left and right neighbours and is thus not extreme. Rectangle 4 has left and right and top and bottom neighbours and is thus not extreme.

3. As long as there are still some unvisited extreme rectangles.

   (a) Pick the next extreme rectangle

   (b) For each adjacency of the extreme rectangle which has not yet been crossed

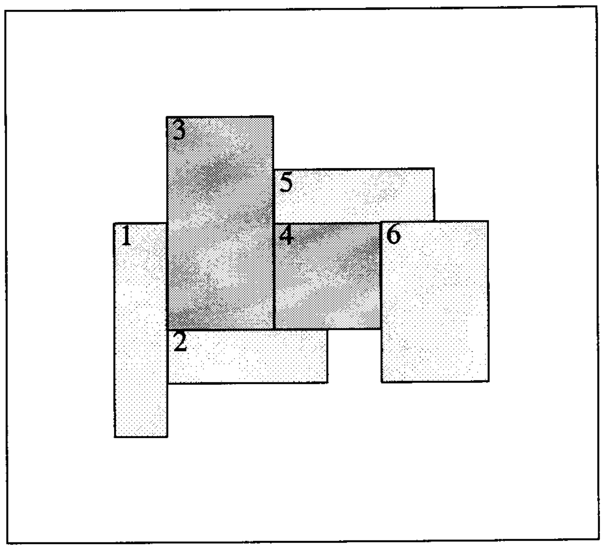


Figure 5.13: Longest chains – identifying "extreme" rectangles

i. using the information stored in the adjacency matrix determine all of the chains of rectangles starting from the extreme rectangle selected in 3a, crossing the current adjacency and ending in another extreme rectangle.

ii. select the longest chain – the chain with the most rectangles in it – that includes that adjacency

iii. determine whether all of the adjacencies in this longest chain can be crossed by one line. To answer this question the approach of determining partial visibility between two edges in a polygon (see Section 5.4) can be used.

   If all of the adjacencies in this chain can be crossed by one axial line

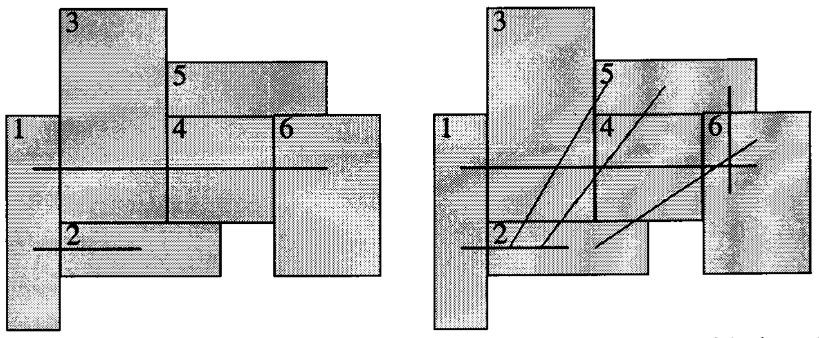
   - include that line as one of the final set of lines
   - mark all of the adjacencies in that chain as crossed

   Or else if all of the adjacencies in this chain *cannot* be crossed by one axial line then

   - select the next longest chain of rectangles and repeat the process until an axial line can be placed or there are no more chains

(c) Repeat this process until all the extreme rectangles have been considered.

4. If there are still uncrossed adjacencies then place an axial line to cross each of those adjacencies.



a) Lines from the first extreme rectangle

b) A solution after considering all extreme rectangles

Figure 5.14: The longest chain heuristic

5. The resulting set of lines is an approximation to the solution for ***ALP-ALOR***

If this heuristic was applied to the configuration of rectangles in Figure 5.13 then the first extreme rectangle to be considered would be rectangle 1. This gives rise to the chains 1–2, 1–3–2, 1–3–5, 1–3–4–2, 1–3–4–5 and 1–3–4–6. These chains all start and end with one of the extreme rectangles. A chain like 1–2–3–4–5 would not be generated as it has an extreme rectangle 2 which is not at the beginning or end of the chain. The algorithm would stop generating a chain as soon as rectangle 2 was encountered. In the list of valid chains originating from rectangle 1, the chain 1–2 is the only chain that includes adjacency 1|2 so this chain is selected and the line 1–2 can be placed. There are three longest chains which include the adjacency 1|3 and any could be chosen. If the chain 1–3–4–6 was chosen then the line 1–3–4–6 could be placed. Figure 5.14.a shows the lines placed for the first extreme rectangle. Figure 5.14.b shows a possible solution once all of the extreme rectangles have been visited. This figure also demonstrates a problem with the longest line heuristic – the line crossing adjacency 1|2 is not "as long as possible" – the line should have been extended to cross adjacencies 2|3, 3|4 and 4|6.

Clearly other solutions could occur if different chains are selected when there is a choice of "longest chains". This local choice can, in fact, lead to another problem with this heuristic. This is shown in Figure 5.15. In this case the line chosen to cross the adjacency 1|3 was the line 1–3–4–5. This would leave the adjacency 4|6 uncrossed until the extreme rectangle 6 was considered. Then the line 1–3–4–6 would be placed. This would result in a solution with one more line than if 1–3–4–6

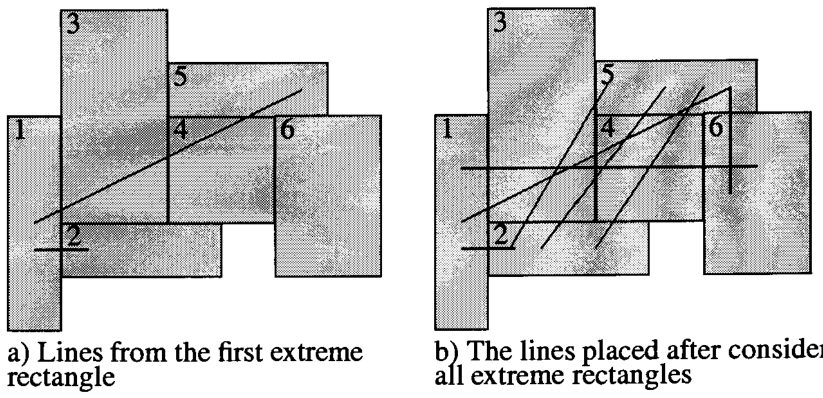

3
5
1
4
6
2
a) Lines from the first extreme rectangle
3
5
1
4
6
2
b) The lines placed after considering all extreme rectangles

Figure 5.15: The longest chain heuristic: A problem with the heuristic – there are redundant lines in the final set of lines

had been chosen when the adjacency 1|3 was being considered.

Another problem with this "longest chains" heuristic (related to a problem discussed earlier) could occur if the configuration of rectangles is as shown in Figure 5.16.a. Here the configuration has only two extreme rectangles 1 and 7. There are two chains from 1 to 7 (and from 7 to 1) but in neither chain can one line cross all of the adjacencies between the rectangles in the chain. Thus no line will be placed while considering each extreme rectangle in turn. Thereafter all the adjacencies are still uncrossed and lines will be placed to cross each adjacency – see the example in Figure 5.16.b. Clearly this results in far more lines than are actually necessary and each line is not "as long as possible".

This problem could possibly be solved by pre-processing to determine that the configuration of adjacent rectangles is of this form but more complicated configurations of rectangles could display similar behaviour. A heuristic should not be as susceptible to the actual configurations of input rectangles as this one is. The heuristic in Section 5.5.5 goes some way to addressing the problems which could arise with the heuristic discussed here.

5.5.5 Crossing one adjacency at a time

Another idea for a heuristic to find an approximation to ***ALP-ALOR*** is to consider crossing each as yet uncrossed adjacency in turn with a line which is "as long as possible". The detail of this idea is expanded upon below.

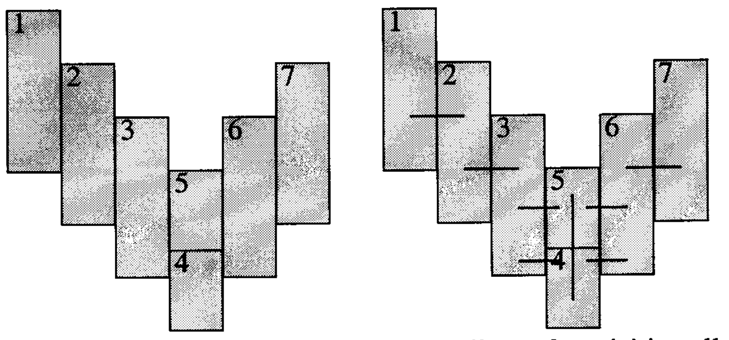

1
2
3
4
5
6
7
a) A configuration with two extreme rectangles
b) The lines after visiting all extreme rectangles

Figure 5.16: The longest chain heuristic: A second problem with the heuristic – all the adjacencies are uncrossed after all the extreme rectangles have been considered

1. Determine all of the adjacencies between the orthogonal rectangles. Store this information in an adjacency matrix.
2. Sort the adjacencies based on their minimum x-coordinate, break ties based on minimum y-coordinate.
3. As long as there are still some uncrossed adjacencies.
   (a) Pick the next adjacency.
   (b) Using the information stored in the adjacency matrix and starting with the two rectangles which define the adjacency selected in point (3a) successively add rectangles to this chain until adding another rectangle would cause the chain to "kink'.

   The sweep works from left to right – so any rectangle to be added to the chain would have to be added to the right side or the top or bottom of the last rectangle in the chain. Kinking then occurs if

   Case 1 if the rectangle which is being considered for adding to the chain is adjacent to the left side of the last rectangle in the chain

   Case 2 if the rectangle which is being considered for adding to the chain is adjacent to the bottom of the last rectangle in the chain and the last rectangle in the chain was added to the top of the penultimate rectangle in the chain

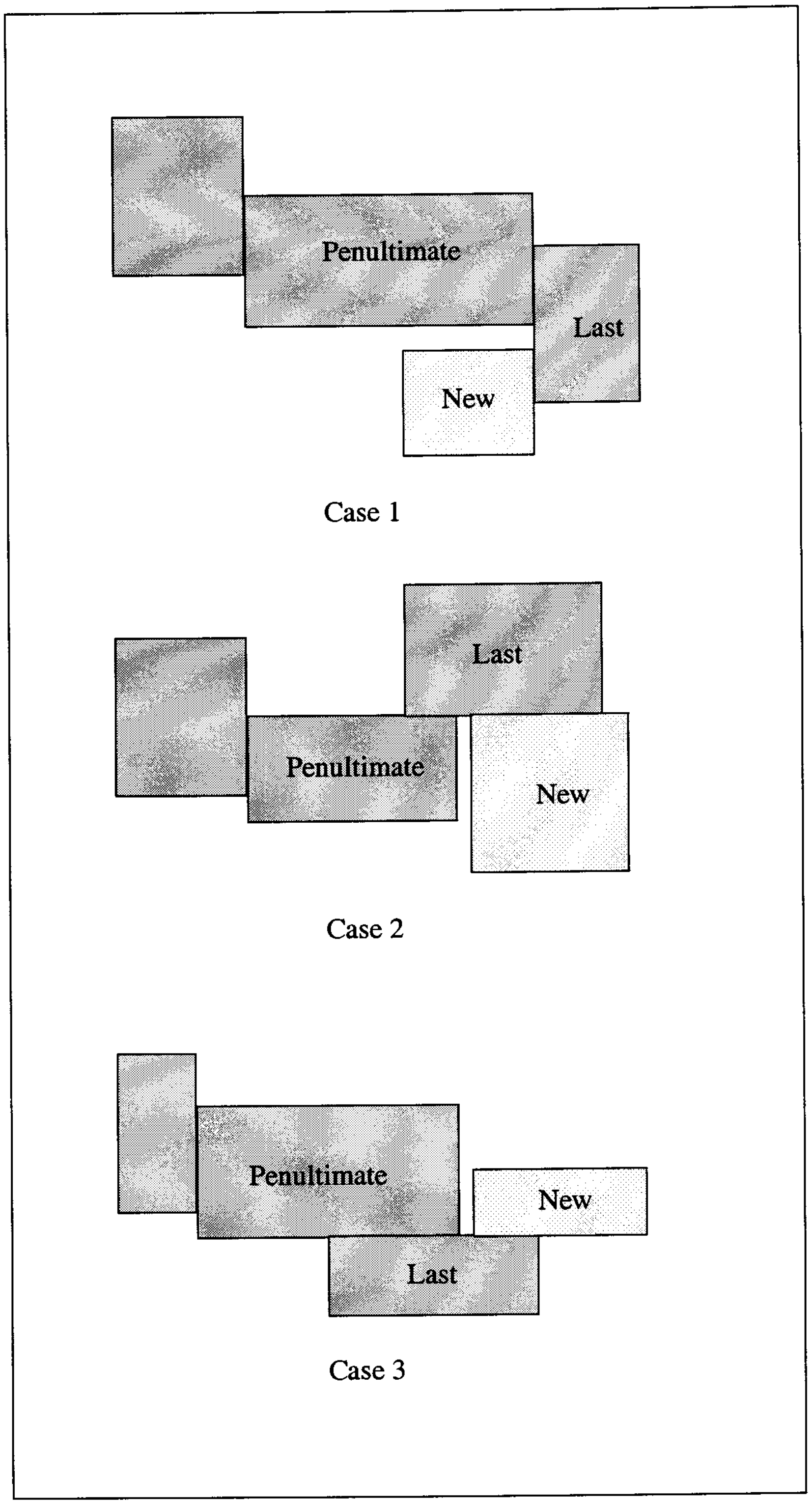


Figure 5.17: Crossing one adjacency at a time: Cases which cause “kinking ” in a chain of rectangles

Case 3 if the rectangle which is being considered for adding to the chain is adjacent to the top of the last rectangle in the chain and the last rectangle in the chain was added to the bottom of the penultimate rectangle in the chain

Figure 5.17 shows examples of these three cases.

(c) Determine if a line can be placed to cross the adjacencies in the chain. (This can be done using the idea of partial edge visibility discussed in section 5.4.)

If a line can be placed then add this line to the final set of lines.

If a line cannot be placed then repeatedly

i. generate a different chain starting with the same adjacency and following the same constraints

ii. determine whether a line can be placed to cross all of the adjacencies in that chain

until either a line is found or all the possible chains have been considered.

If a line cannot be placed through any chain starting at that adjacency and generated as above then make a line which crosses just that adjacency.

(d) Mark off all the adjacencies crossed by the line generated above.

4. The resulting set of lines is an approximation to the solution for ***ALP-ALOR***

The application of this idea is illustrated by considering the configuration of rectangles in Figure 5.18. The first adjacency considered is 1|2. The chain thus starts as being 1–2. Then suppose that rectangle 3 is considered, this rectangle can be added to the chain which now becomes 1–2–3. This chain can be extended to 1–2–3–4–5. If an attempt is made to add rectangle 6 to the chain then kinking occurs. Rectangles 4 and 6 are on the same side of rectangle 5 and it would thus be impossible to put a straight line through the chain of rectangles 1–2–3–4–5–6. It might however be possible to place a line through the chain 1–2–3–4–5. This is, in fact, the case so the line 1–2–3–4–5 is created. This line crosses the adjacencies 1|2, 2|3, 3|4 and 4|5 – see Figure 5.19.a.

The next adjacency to be considered would be 1|3 – this is the leftmost as yet uncrossed adjacency. A number of possible chains (without kinking) start with this adjacency: 1–3–2, 1–3–5, 1–3–4–5 and 1–3–4–6. Suppose that 1–3–2 was the first one considered. A line can be placed through the adjacencies in this chain and so this line is added to the final set of lines – see Figure 5.19.b.

The next adjacency to be considered would then be 3|5 (1|2, 1|3, 2|3 and 3|4 have already been crossed) and the process would continue. A possible final set of

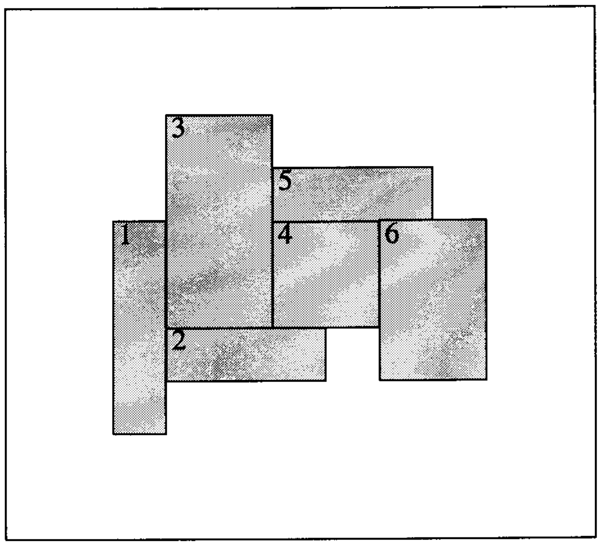


Figure 5.18: Crossing one adjacency at a time – example input

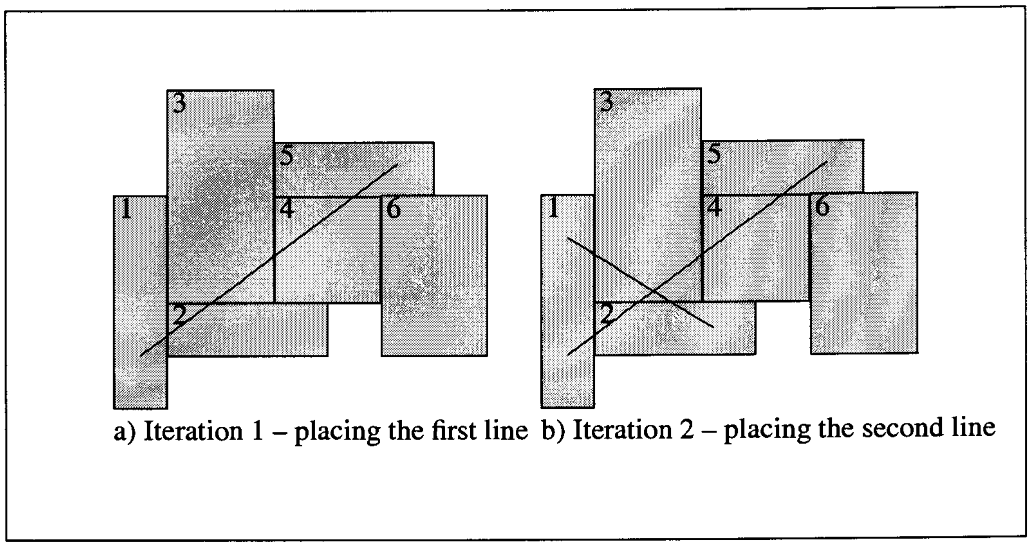

3
5
1
4
6
2
a) Iteration 1 – placing the first line
b) Iteration 2 – placing the second line

Figure 5.19: Crossing one adjacency at a time – the first two passes of the algorithm

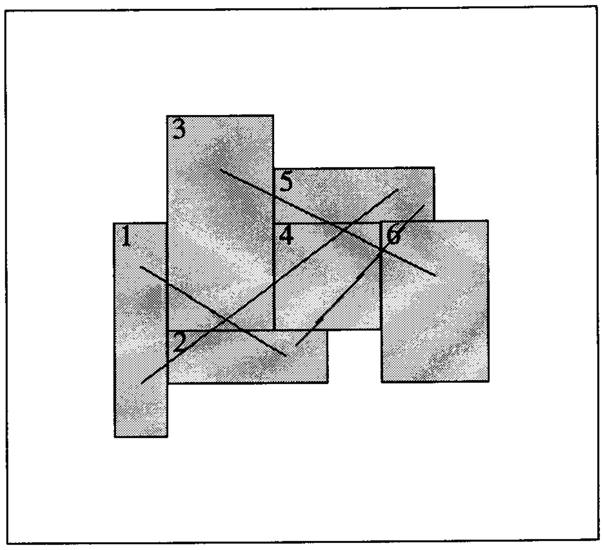


Figure 5.20: Crossing one adjacency at a time – a possible solution

lines is shown in Figure 5.20. This is in fact a minimal solution for this instance of the problem.

There are, however, some problems with this algorithm. Suppose that a different chain was selected when attempting to cross adjacency 1|2 and that after two iterations the lines 1–2–4–6 and 1–3–2 had been placed – see Figure 5.21.a. The next adjacency to be crossed would be 3|4. Three chains of rectangles are possible 3–4–2, 3–4–5 and 3–4–6. A line can be placed in each of these chains. Assume (for the sake of the discussion) that the second chain was chosen (the argument applies to the other two chains as well). Then the situation after placing this line (3–4–5) would be as shown in Figure 5.21.b. This last line is clearly not "as long as possible" – it should cross the adjacency 1|3 as well. A second pass through the final set of lines would be required to extend such lines backwards as far as possible.

Another problem which could occur with this heuristic is that it does not take into account "curves" in chains – see Figure 5.22.a. In a case like this where the adjacencies being considered is 1|2, the rectangles 1, 2, 3, 4, 5 and 6 form a chain (1–2–3–4–5–6) and although there is no point at which the chain wraps around (none of the cases identified above occur), no line can be placed through the whole chain. The heuristic would thus generate the line 1–2 and move on to consider the adjacency 2|3 where a similar situation would occur. The final set of lines (shown in Figure 5.22.b) would thus not be minimal.

Again it is possible that this problem could be resolved by a "post-processing" pass over the "final" set of lines to attempt to extend any lines which only cross a single adjacency as far as possible.

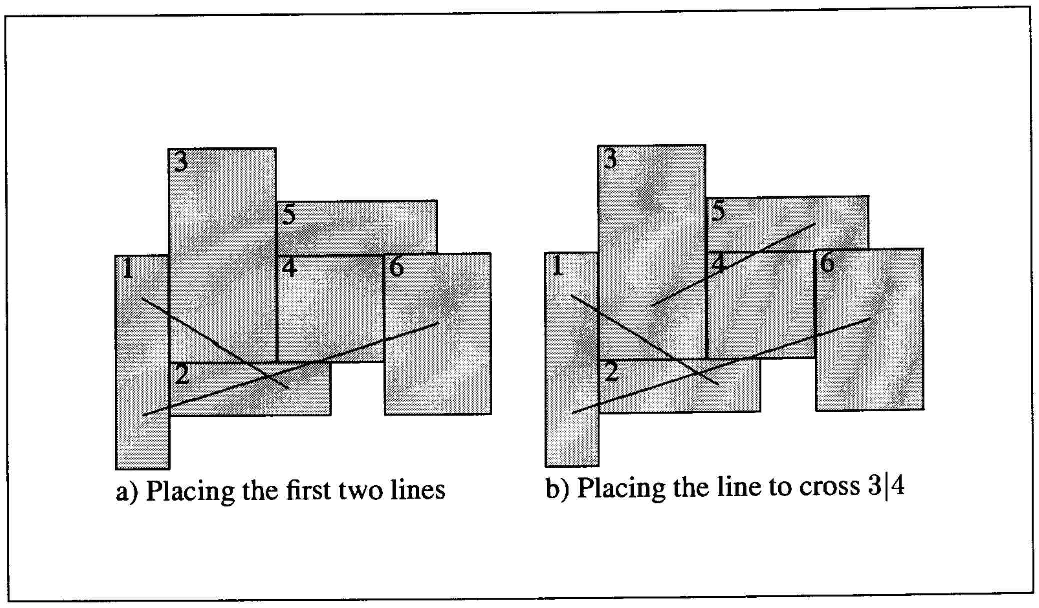

3
5
1
4
6
2
a) Placing the first two lines
3
5
1
4
6
2
b) Placing the line to cross 3|4

Figure 5.21: Crossing one adjacency at a time: A problem with the heuristic – lines that don't extend far enough to the left

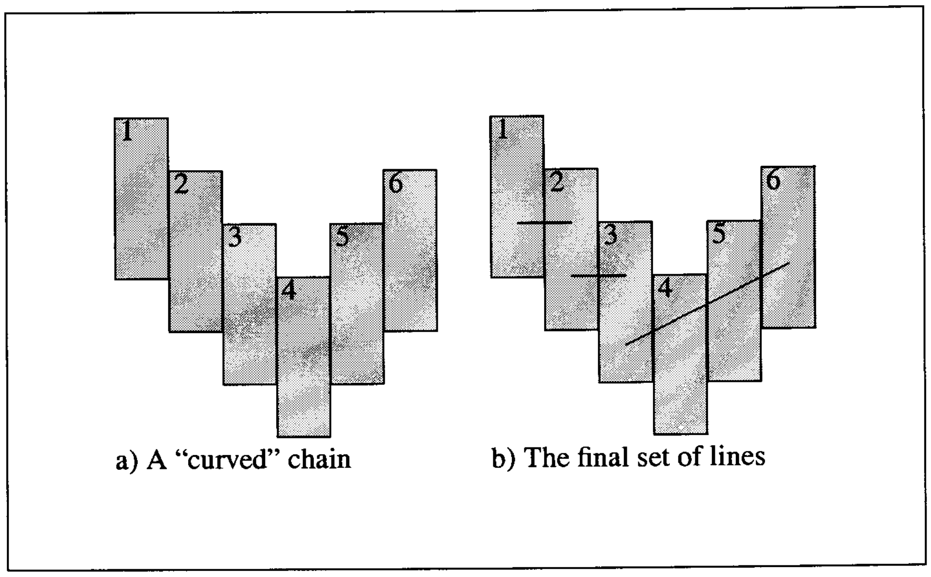


Figure 5.22: Crossing one adjacency at a time: A second problem with the heuristic – lines which only cross a single adjacency

5.5.6 Extending forwards and then backwards

This approach is similar to the heuristic suggested above and was developed by Kenny [2000] working under the supervision of the author of the thesis. Again for each adjacency that is still uncrossed a new chain working from left to right is constructed. When the chain kinks the forward phase is complete and it is necessary to determine whether a straight line can be placed in the chain. If a line can be placed then the next stage of the process – extending the chain to the left – follows. However, if a line cannot be placed then one rectangle at a time will be removed from the right hand side of the chain until a line can be placed or until there are only two rectangles in the chain. These two remaining rectangles are adjacent to each other and clearly a line can be placed to cross the adjacency between them.

This first stage ensures that the lines extend as far as possible to the right. However, it is possible that they don't extend as far to the left as possible. The second stage of the heuristic addresses this.

The chain is now extended backwards (to the left) as far as possible by successively adding rectangles to the chain until kinking would occur. Then a similar process of testing for line placement and removing rectangles from the chain is followed.

The resulting line is now also extends as far to the left as possible and so is "as long as possible". It is thus included in the final set of lines.

Parts a) to i) of Figure 5.23 show the application of this heuristic to a simple configuration of adjacent rectangles. The first uncrossed adjacency to be considered is adjacency A. The two rectangles comprising this adjacency, namely rectangles 1 and 2 form the start of the chain as shown in a) . Rectangle 3 is adjacent to 2 on the right and is added to the chain. Similarly, rectangle 4 is adjacent to rectangle 3 and is also added to the chain. Rectangle 5 is on the left of rectangle 4 and therefore is not considered for inclusion in the chain since it would result in kinking of the chain. The chain comprising rectangles 1, 2, 3, and 4 is now checked for visibility. The chain clearly does not allow visibility so rectangle 4 is removed from the chain. The new chain shown in d) consists of rectangles 1, 2 and 3. This chain is does allow visibility so the forward part of the algorithm is complete and an attempt is made to extend this possible line backwards. Rectangle 1 does not have any left or bottom adjacencies so the chain cannot be extended backwards. Thus a line is placed crossing adjacencies A and B. The next uncrossed adjacency is C. The first two rectangles of the chain are 3 and 4 as shown in e). The chain cannot be extended in the forward direction since this leads to kinking. However, we can extend the chain backwards into rectangles 2 and 1, as shown in f) and g). The chain in g) consisting of rectangles 1, 2, 3, and 4 does not allow visibility. Since the chain is now being extended backwards, rectangle 1 is removed from the left hand side of the chain. The resulting chain shown in h) does allow visibility and thus a

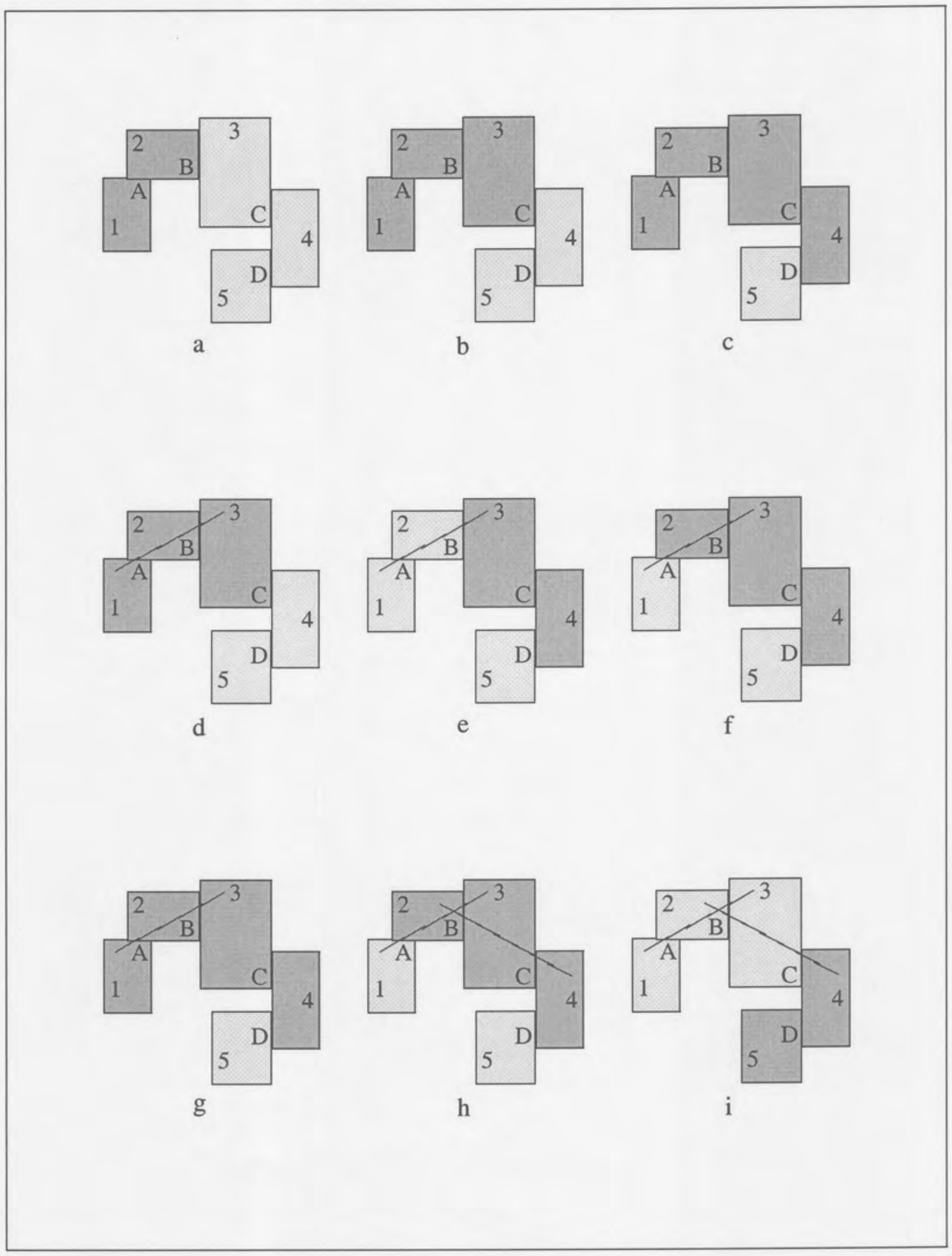


Figure 5.23: Extending forwards and then backwards – the different stages of determining chains, modifying the chains and placing axial lines.

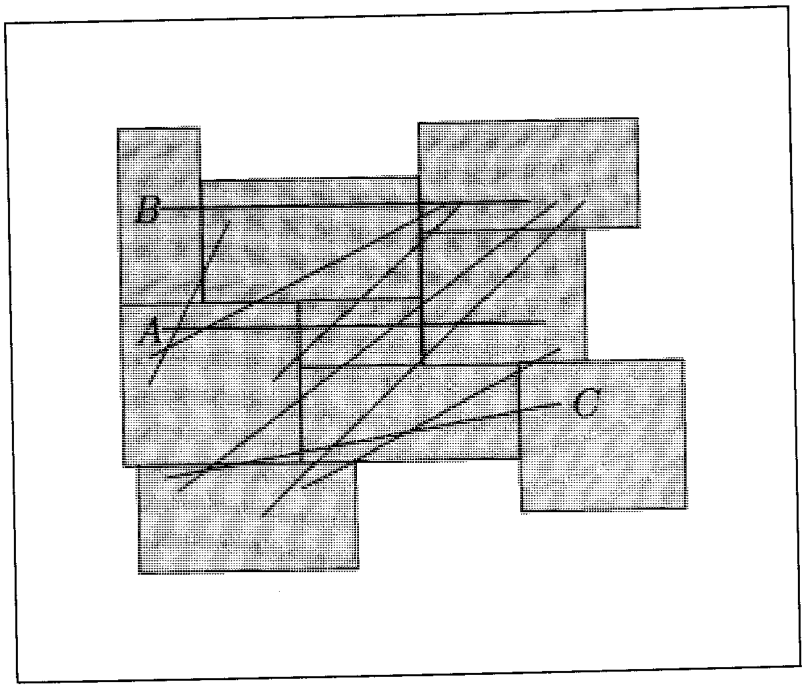


Figure 5.24: Extending forwards and then backwards – redundant lines can be generated.

line can be placed that crosses adjacencies B and C. The only remaining uncrossed adjacency is D. The start of the new chain comprises rectangles 4 and 5 as shown in i). The chain cannot be extended forwards (there are no more rectangles) and cannot be extended backwards into rectangle 3 since that would lead to kinking. Therefore a line can be placed to cross adjacency D. All of the adjacencies have now been crossed.

This heuristic has been implemented and tested on some configurations of adjacent rectangles. The heuristic seems to offer a reasonable approximation to the exact result in the cases tested. There are, however, some cases where the heuristic produces redundant lines – lines which only cross adjacencies which are crossed by other lines generated later in the process (see Figure 5.24 where lines A, B and C are redundant). The removal of redundant lines could be achieved by a post processing phase – identifying "essential lines" (axial lines that cross some adjacencies that are not crossed by any other axial lines) and then removing the redundant lines.

Another problem with the implementation of this heuristic is that it is biased in favour of crossing vertical adjacencies (the first rectangle considered to extend a chain of rectangles is a right neighbour) and this can affect the accuracy of the approximation.

5.5.7 Summing Up

This section of the thesis has considered some heuristics which could be used to produce approximate solutions to ***ALP-ALOR***. Clearly more work in this area is required to clarify the ideas and find a good heuristic. This additional work is outside the scope of this thesis but some work is currently being undertaken (under the author's supervision) by an Honours student in the School of Computer Science at the University of the Witwatersrand.

5.6 Special cases which can be solved in polynomial time

ALP-OLOR is in general NP-Complete but it seems likely that there are some special cases of the general problem for which polynomial time algorithms can be obtained. In particular it seems likely that an exact solution for configurations of rectangles which form chains, as defined in Section 4.7.2.3, could be found in polynomial time. In this variation of the problem a more generalised form of chain where any rectangle in the chain can have at most two neighbours (adjacent on different sides) as shown in Figure 5.25 could also be considered. It seems likely that exact solutions for this type of configuration could also be found in polynomial time.

Whether exact polynomial-time solutions for more complicated configurations of rectangles can be found is still an open question.

5.7 Future Research

Section 5.3 of this thesis has shown that ***ALP-ALOR*** is NP-Complete. This points to two fruitful areas for future research – heuristic algorithms for approximate solutions and special cases where the exact solution can be found in polynomial time.

A previous section of this thesis (Section 5.5) addressed the first of these areas to some extent and suggested some heuristics which could be used to produce approximate solutions to ***ALP-ALOR*** but future work could be focussed on

- developing more, and hopefully better, heuristics to solve the problem,
- testing of the heuristics presented here and any new heuristics which are developed.
- looking at other approaches, for example genetic algorithms, for finding approximate solutions.

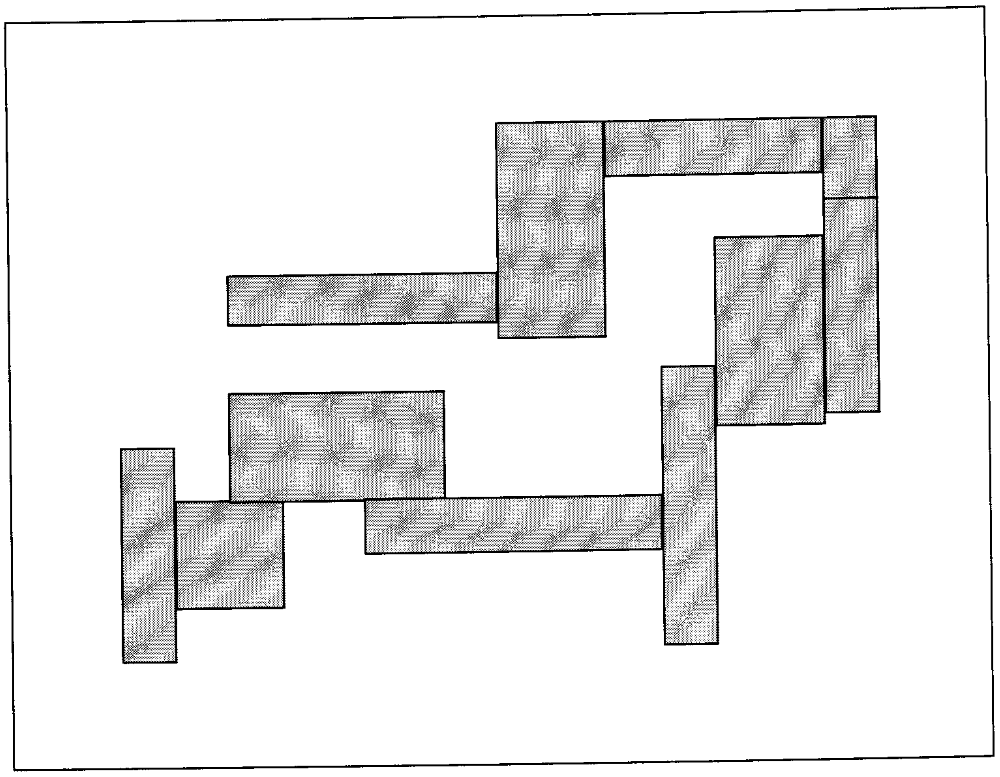


Figure 5.25: A more general chain of rectangles

The area of special cases which can be solved in polynomial time is very briefly addressed in this thesis. It is conjectured that simple chains of adjacent rectangles and even more complicated chains could be solved exactly in polynomial time. Attention should be focussed on proving these claims and should also be focussed on attempting to determine other, more complicated configurations of adjacent rectangles for which the problem can be solved exactly in polynomial time.

5.8 Conclusion

The axial line placement problem for axial lines with arbitrary orientation and orthogonal rectangles (***ALP-ALOR***) has been shown to be an NP-Complete problem as is ***ALP-OLOR***. Also in this chapter some heuristics to find approximations for ***ALP-ALOR*** have been suggested but these heuristics still have to be tested to determine how good the approximations are likely to be. The idea of special cases which can be solved in polynomial time has been briefly touched upon but this is essentially left as future work.

The NP-Completeness result presented in this chapter can be extended to a slightly more general problem – axial lines with arbitrary orientation crossing the adjacent edges between convex polygons (***ALP-ALCP***). This extension of the result is discussed in the next chapter (Chapter 6) of this thesis.

Chapter 6

Placing axial lines with arbitrary orientation to cross the adjacencies between convex polygons

6.1 Introduction

The problem which is being considered in this thesis is that of placing the axial lines through the convex spaces in an urban layout. Chapters 4 and 5 discuss simplifications of this problem. In both cases the convex spaces (convex polygons) are restricted to being rectangles and the problem is that of finding the minimum number of axial lines that cross all of the adjacencies between rectangles. In Chapter 4 the axial lines are restricted to being parallel to the Euclidean axes while in Chapter 5 this restriction no longer applies. This chapter discuss the generalisation of the problem where the aim is to find the minimum number of axial lines to cross all of the shared boundaries between adjacent convex polygons. Both of the simplifications, ***ALP-OLOR*** and ***ALP-ALOR***, are NP-Complete. In this chapter ***ALP-ALCP*** (Axial Line Placement – Arbitrary Lines and Convex Polygons) is also shown to be NP-Complete.

In the remainder of this chapter (***ALP-ALCP***) is discussed in more detail. The problem is restated in detail in Section 6.2 below. In Section 6.3 ***ALP-ALCP*** is shown to be NP-Complete by demonstrating that ***ALP-ALOR*** is a restriction of ***ALP-ALCP***. Then, because ***ALP-ALCP*** is NP-Complete, some ideas for heuristics to find approximate axial maps are discussed in Section 6.4 and some comments are made about special cases which can be solved in polynomial time are made in Section 6.5. In Section 6.6 some ideas for future research are briefly discussed.

6.2 Statement of the Problem

Given a number of adjacent convex polygons find the fewest axial lines contained wholly inside the polygons which will cross all of the shared boundaries (adjacencies) between adjacent polygons. An additional requirement is that each axial line should cross as many of the shared boundaries as possible – a *maximal* axial line.

As in Chapters 4 and 5, depending on how the problem is considered there are 2 similar but distinct problems which can be addressed.

1. Adjacencies can be crossed more than once but every adjacency must be crossed *at least* once.

2. Any adjacency has *exactly* one line crossing it.

In this chapter and in this thesis only problem 1 is addressed. An example of the problem is shown in Figure 6.2.

6.3 Proving the Problem is NP-Complete

The problem can be formally stated as

ALP-ALCP
Instance: A collection of convex polygons $P_1 \ldots P_n$, where each polygon is adjacent to at least one other polygon, and a positive integer M.
Question: Is there a set L of axial lines where each axial line is maximal in length, each line is wholly contained in the collection of polygons, each shared boundary between adjacent polygons is crossed at least once and $\mid L \mid \leq M$?

Theorem 6.3.1 *ALP-ALCP is NP-Complete.*

Proof
Clearly ***ALP-ALCP*** is in NP. Given a set of axial lines with arbitrary orientation it is possible to check in polynomial time that each adjacency has been crossed by at least one axial line.

The problem ***ALP-ALOR*** which was proved to be NP-Complete in Chapter 5 is a restricted instance of ***ALP-ALCP***. It is a special case of ***ALP-ALCP*** where the convex polygons are restricted to being orthogonal rectangles. Thus using the approach of proof by restriction (see Section 2.3) Theorem 6.3.1 has been proved.
□

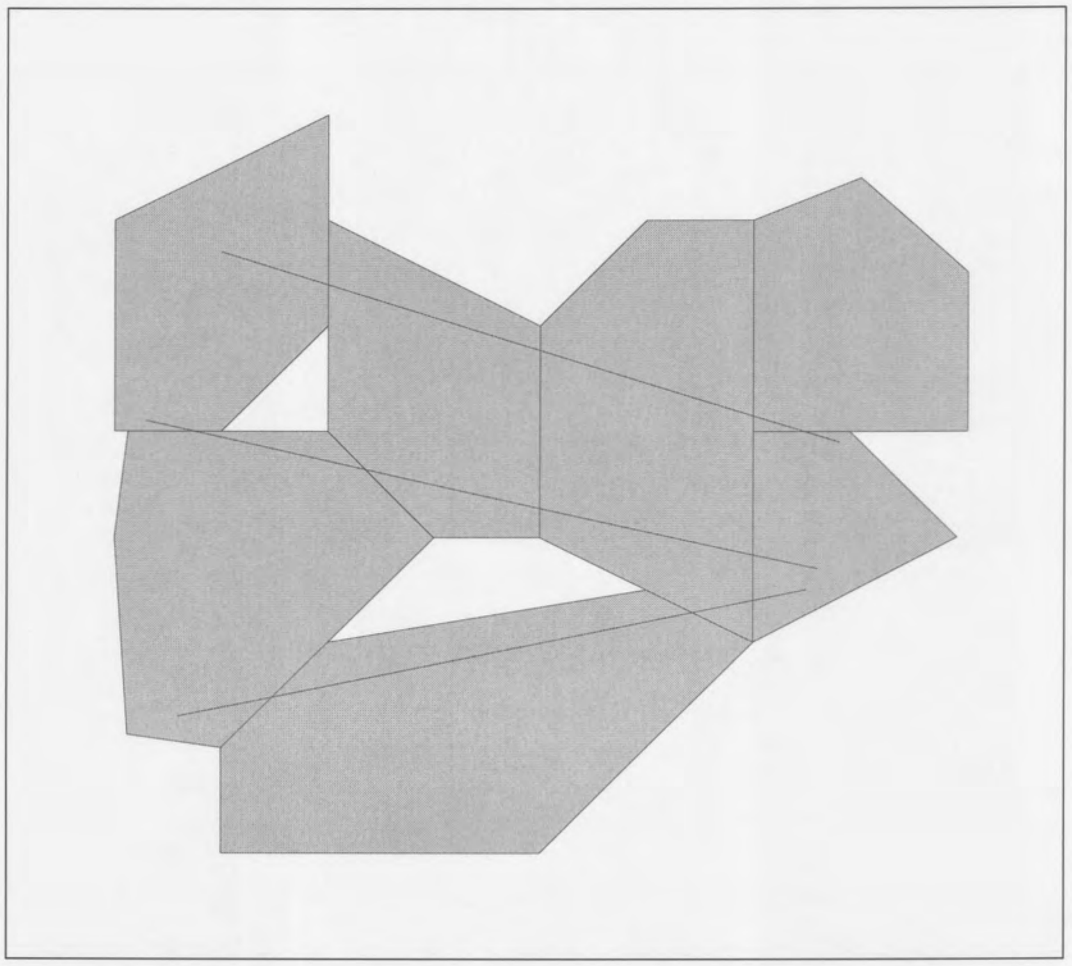


Figure 6.1: An example of placing axial lines to cross all of the adjacencies in a configuration of adjacent convex polygons

6.4 Heuristics to find approximate solutions for *ALP-ALCP*

ALP-ALCP has been shown to be NP-Complete (Section 6.3) and thus in the general case finding a minimal solution could take an unreasonable amount of time. It is thus necessary to consider heuristics to find approximate solutions in reasonable time. Developing the heuristics is beyond the scope of this thesis and thus will be future work which arises from this thesis. However some of the heuristics which are suggested in Section 5.5 could possibly be modified to produce approximate solutions to ***ALP-ALOR***.

6.5 Special cases of *ALP-ALCP* which can be solved in polynomial time

ALP-OLOR is in general NP-Complete but it seems likely that there are some special cases of the general problem for which polynomial time algorithms can be obtained. In particular it seems possible that an exact solution for configurations of convex polygons which form chains (analogous to the more general chains presented in Section 5.6) could be found in polynomial time. See Figure 6.2 for an example of such a chain.

Whether exact polynomial-time solutions for more complicated configurations of convex polygons can be found is still an open question and is outside the scope of this thesis.

6.6 Future Work

Section 6.3 of this thesis shows that ***ALP-ALCP*** is NP-Complete. This points to two fruitful areas for future research – heuristic algorithms for approximate solutions and special cases where the exact solution can be found in polynomial time. Neither of these areas has been addressed in this thesis but both certainly warrant attention.

6.7 Finding the adjacencies in a configuration of adjacent convex polygons

In order to develop heuristics to find reasonable approximations to the exact solution for ***ALP-ALCP*** or to consider special cases for ***ALP-ALCP***, it is necessary to be able to efficiently find the adjacencies in a configuration of adjacent convex polygons.

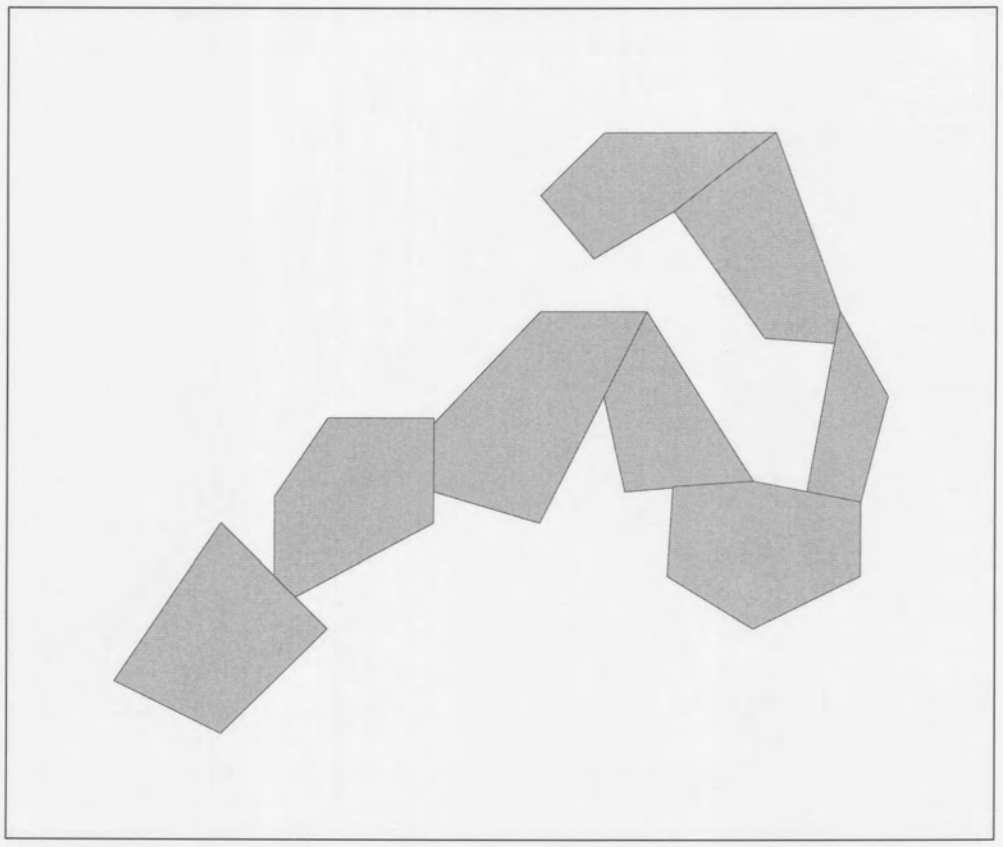


Figure 6.2: A chain of convex polygons

Recently Adler *et al.* [2001] produced a solution to this problem[1].

6.8 Conclusion

In this chapter of the thesis the axial line placement problem for axial lines with arbitrary orientation and convex polygons (***ALP-ALCP***) has been shown to be an NP-Complete problem as are ***ALP-OLOR*** and ***ALP-ALOR***. The result in this chapter (***ALP-ALCP***) follows because ***ALP-ALOR*** is a restriction of ***ALP-ALCP*** – orthogonal rectangles are a subclass of convex polygons. This result raises that question of whether or not the original axial line placement problem – placing the minimum number of axial lines to cross the adjacencies in a convex map of a town or urban layout – is also NP-Complete. In the next chapter of the thesis this question is addressed.

[1]The author of this thesis set the problem as an assignment for the 2001 University of the Witwatersrand Computer Science Honours class in their Analysis of Algorithms course. The resulting assignment solutions were then combined to produce a paper which was submitted to the SAICSIT 2001 Research Symposium and accepted as an "electronic publication"

Chapter 7

Placing Axial Lines in Town Plans

7.1 Introduction

The original problem from the town planning domain which is considered in this thesis is to find the axial map for an urban area given the convex map of area. A convex map is composed of the minimum number of convex spaces (convex polygons) which partition the urban area (represented as a polygon with holes). In this thesis only partitions which do not make use of Steiner points (see Section 2.2) are allowed. An axial map consists of the minimum number of "axial lines" (straight line segments) necessary to cross the adjacencies between the convex polygons which make up the convex map. In this thesis only the problem where adjacencies can be crossed more than once but *must* be crossed at least once is considered. An additional requirement is that the axial lines should cross as many adjacencies as possible. These constraints are chosen to reflect the constraints in the real world problem as closely as possible.

As was seen in Chapter 2 the first of these problems – covering or partitioning a general polygon by the smallest number of convex polygons (or other types of polygons) – has been quite well studied in the general case. Many of these covering and partitioning problems have been shown to be NP-Complete or NP-Hard. In particular O'Rourke and Supowit [1983] have shown that the problem of partitioning a general polygon with holes into convex polygons is NP-Hard. This thesis concentrates on the second of these problems – crossing the adjacencies between convex spaces by the minimum number of axial lines. In Chapter 4 the problem of crossing the adjacencies between orthogonal rectangles by orthogonal lines is shown to be NP-Complete. Likewise in Chapter 5 the problem of crossing the adjacencies between orthogonal rectangles by lines which are not necessarily orthogonal is shown to be NP-Complete. The proof in Chapter 5 can be extended to show that the problem of crossing the adjacencies between general convex polygons by lines with arbitrary orientation is NP-Complete (Chapter 6).

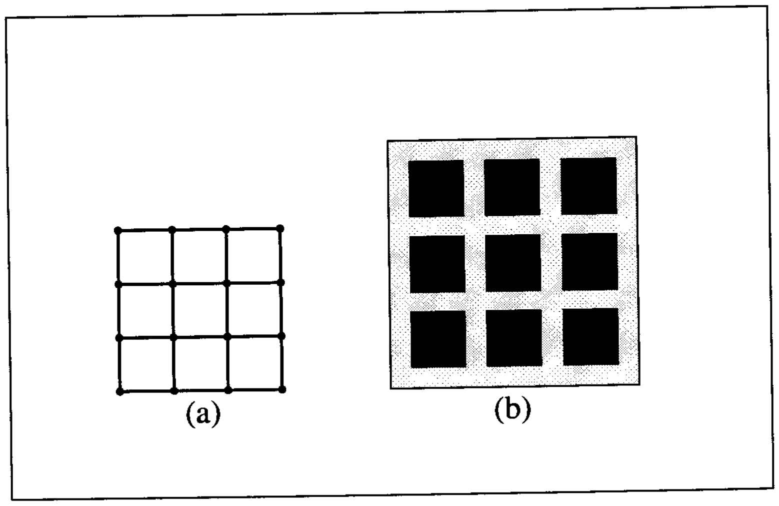


Figure 7.1: (a) A complete grid of size 4, (b) The corresponding complete urban grid of size 4

In this chapter the emphasis is on finding sets of lines to cross the adjacencies between convex polygons in a situation which is representative of the real world problem, i.e. covering the convex map of some town with an axial map. The chapter begins by considering some special cases where it is possible to prove that the minimal axial line cover can be found in polynomial time (Section 7.2). The problem in somewhat less restricted cases is then considered (Section 7.3) and finally the more general cases are briefly considered.

7.2 Urban Grids

Gewali and Ntafos [1993] define the *complete two-dimensional grid* of size n as 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 this thesis the grid definitions of Gewali and Ntafos [1993] are extended to define the concept of *urban grids*.

Definition 7.2.1 *A* complete urban grid *of size n is a complete two-dimensional grid of size n where each grid edge or corridor $E_p, 1 \leq p \leq 2n(n-1)$ is an orthogonal rectangle with length l and width w, $l > w > 0$, and each grid vertex or intersection, $I_{i,j}, i, j = 1, 2, \ldots n$, is an orthogonal square of size $w \times w$. Each end of any corridor rectangle must be a side of an intersection.*

Figure 7.1 shows in (a) an example of a complete grid of size 4 and in (b) a complete urban grid of size 4.

Definition 7.2.2 *An urban grid is any subgraph of the complete urban grid.*

Gewali and Ntafos [1993] also define a *grid segment* as a succession of grid edges along a straight line bounded at either end by a missing edge. A *simple grid* is a grid where all of the endpoints of the grid segments lie on the outer face of the planar subdivision formed by the grid. A *general grid* is a grid which can have holes – some of the endpoints of the segments may lie on the inner face of the planar subdivision.

Definition 7.2.3 *A thoroughfare is a grid segment in an urban grid.*

Definition 7.2.4 *A simple urban grid is an urban grid where all of the endpoints of the thoroughfares lie on the outer face of the planar subdivision formed by the grid.*

Definition 7.2.5 *A general urban grid is an urban grid which can have holes – some of the endpoints of the thoroughfares may lie on the inner face of the planar subdivision.*

In order to place the axial lines in a complete urban grid it is necessary first to be able to find the convex map of the urban grid, i.e. to be able to partition the complete urban grid into the smallest number of convex polygons. The complete urban grid of size n has n^2 intersections and $2n(n-1)$ corridors. The number of corridors is easily seen by observing that each of the n horizontal thoroughfares have $n-1$ corridors, giving a total of $n(n-1)$ corridors lying horizontally. Similarly, there are $n(n-1)$ corridors lying vertically. The convex map must have fewer convex polygons than the sum of the corridors and intersections as the convex polygons representing these can be merged into a smaller number of larger polygons.

The minimum partition for the corridors and intersections of corridors which make up the four outer thoroughfares will be comprised of 4 convex polygons. Any attempt to partition the urban grid by combining the partitioning of the inner and outer regions would lead to more than 4 convex polygons in the outer region without resolving the partitioning of the inner region. There are a number of partitions of size 4 – see Figure 7.2 and Figure 7.3 for examples of the partitioning of the outer region of a complete urban grid of size 4. In Figure 7.2 the partition is accomplished by drawing a diagonal across each corner of the grid and closing off each corridor leading into the interior of the grid. In Figure 7.3 the corner points of the corner intersection are used to define the shared edges of the partition.

Partitioning the inner region will involve the placement of appropriately sized rectangles. This is essentially accomplished by determining where the adjacencies

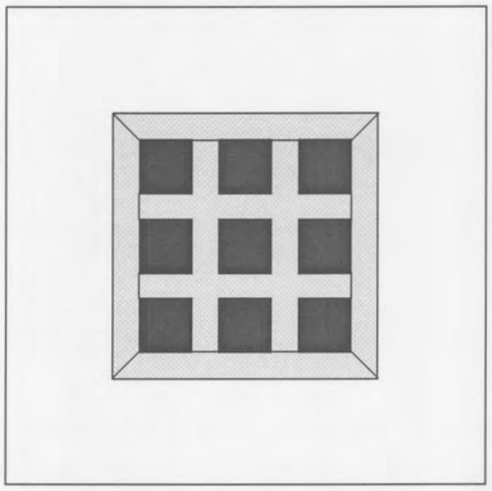


Figure 7.2: An example of partitioning the outer thoroughfares of a complete urban grid of size 4

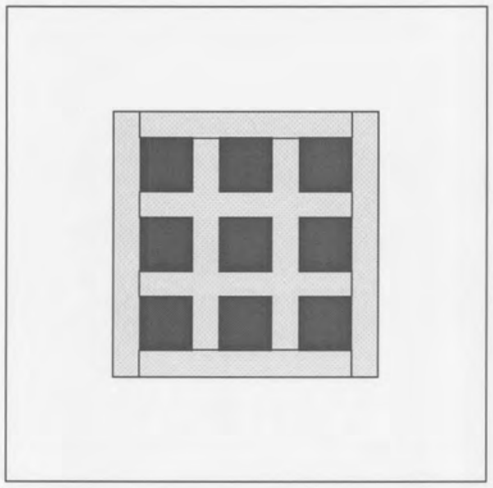


Figure 7.3: Another partitioning of the outer thoroughfares of a complete urban grid of size 4 – other similar partitionings exist

between rectangles can occur. These adjacencies are essentially edges connecting vertices on the boundary of the inner region. The geometry of the problem means that the best place to put these edges will be at the intersections of corridors. There are 4 cases which can occur – see Figure 7.4. Case 1 involves the fewest convex polygons so it would make sense to use as many of these types of partitions as possible. Clearly it is possible to place Case 1 adjacencies at each intersection. If this is done so that these adjacencies are all parallel to the x-axis then the interior region is partitioned into $n-2$ long rectangles (1 for each horizontal thoroughfare in the interior region) and $(n-2)(n-1)$ short rectangles ($n-1$ of them for each vertical thoroughfare in the interior region). The total number of convex polygons required is thus $4+(n-2)+(n-2)(n-1)$. The same number of convex polygons would be required if the long rectangles were aligned with the y-axis. Figure 7.5 shows an example of a partition using Case 1 adjacencies for a complete urban grid of size 4. In the case of an urban grid of size 4 the minimum number of convex polygons required is $4+2+(2)(3)=12$.

The question remains as to whether or not this is the minimum number of convex polygons which can partition a complete urban grid. Theorem 7.2.1 below addresses this question.

Theorem 7.2.1 *A minimum of* $4+(n-2)+(n-2)(n-1)$ *convex polygons are required to partition any complete urban grid of size* $n \geq 2$

Proof

As has been shown above the outer region cannot be partitioned by fewer than 4 convex polygons. Thus it is necessary to show that no partition of the inner region can have fewer than $(n-2)+(n-2)(n-1)$ convex polygons.

Any solution can only be achieved by merging two or more of the short rectangles representing the corridors, plus the squares representing the intersections, in the same thoroughfare (merging across thoroughfares will not result in convex polygons). There are $2(n-2)(n-1)$ of these short rectangles and $(n-2)^2$ intersections in the inner region. The intersections must be merged with the short rectangles to form longer rectangles or convex polygons. As argued above, the best way to do this is to use Case 1 (from Figure 7.4) – using any of the other forms of merging will result in more convex polygons in the final partition. Case 1 merging involves 3 rectangles, Case 2 involves 4 rectangles and Cases 3 and 4 involve 4 convex polygons.

In Case 1, merging each intersection with the corridors around it forces the closing of the ends of two rectangles – these two rectangles cannot now be further merged. This result applies whether horizontal or vertical adjacencies are placed to close off the rectangles. Each intersection must be addressed and so $2(n-2)^2$ rectangle ends are closed when merging the intersections with the corridors around them. The partitioning of the outer region forces the closing of the ends of $4(n-2)$

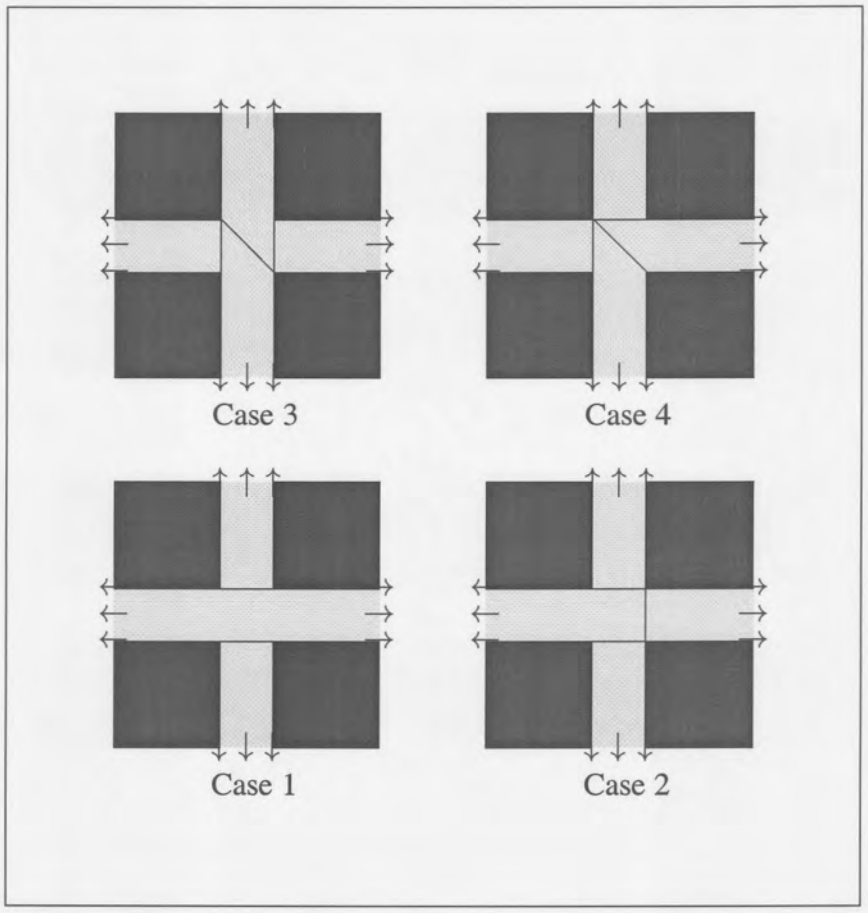


Figure 7.4: Possible adjacencies which could occur in a corridor intersection. Case 1 – Through the intersection (3 convex polygons involved) Case 2 – Rectangular ending at the intersection (4 convex polygons involved) Case 3 – Diagonal ending at the intersection (4 convex polygons involved) Case 4 – L-shaped diagonal ending at the intersection (4 convex polygons involved)

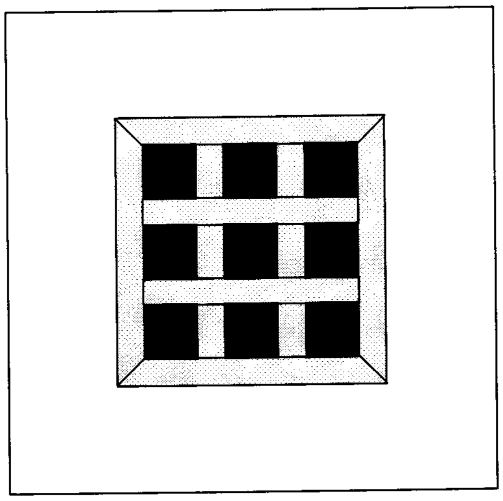


Figure 7.5: A complete partitioning of a complete urban grid of size 4

rectangles – each of the short rectangles which are adjacent to the outer region. Thus in total $4(n-2)+2(n-2)^2$ rectangle ends are closed. This means there must be $(4(n-2)+2(n-2)^2)/2$ or $(n-2)+(n-2)(n-1)$ rectangles or convex polygons in any partition of the inner region $((4(n-2)+2(n-2)^2)/2 = 2(n-2)+(n-2)^2 = 2n-4+n^2-4n+4 = (n-2)+(n^2-3n+2) = (n-2)+(n-2)(n-1))$. The result follows.

□

The next step in the process would be to determine the axial map for the complete urban grid. This is considered in Theorem 7.2.2 below.

Theorem 7.2.2 *The minimal set of axial lines to cross the adjacencies in a minimally partitioned complete urban grid can be found in polynomial time.*

Proof

A minimum partition (convex map) of a complete urban grid consists of $4+(n-2)+(n-2)(n-1)$ convex polygons. Irrespective of the exact form of the convex polygons which make up the partition the form of adjacencies which can occur in any partition are limited. Adjacencies can be

1. diagonals across the intersections in the outer region
2. the edges of the corner intersections in the outer region
3. the shared edge between two rectangles in any thoroughfare (see case 1 in Figure 7.4)
4. at the ends of rectangles between the inner and outer regions

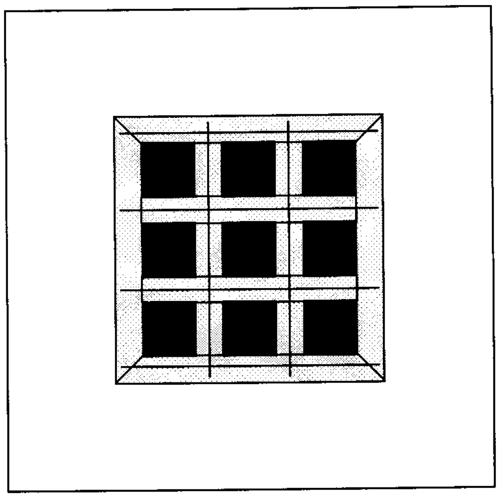


Figure 7.6: The axial map for a complete urban grid of size 4

All of these adjacencies can be crossed as below.

- By placing exactly 2 axial lines to cross the adjacencies in the outer region. The axial lines do not have to be orthogonal and there are only 4 adjacencies to be crossed in any minimally partitioned outer region. The axial map shown in the size 4 version of the problem in Figure 7.6 is one example which shows that only two axial lines are required to cross all of the adjacencies in the outer region. The axial lines in this case could be orthogonal. Figure 7.7 shows an example where axial lines which are non-orthogonal are required. The adjacencies in all other partitions of the outer region can be crossed with similar arrangements of axial lines.

- By placing an axial line in each thoroughfare. These axial lines must cross the adjacencies between the inner and outer regions and all the other adjacencies (if there are any) in the thoroughfare. Note that these axial lines do not necessarily have to be orthogonal.

Clearly this set of axial lines is the minimum to cross the adjacencies in the convex map. Each line through an interior thoroughfare is required. Any such thoroughfare will have at least two adjacencies which must be crossed by axial lines – one at each end of the thoroughfare between the inner and outer regions. One axial line will cross both of these adjacencies and any other adjacencies which appear in that thoroughfare. However, that axial line cannot cross the adjacencies in any other thoroughfare or in the outer region.

Clearly determining these axial lines can be done in polynomial time. If the complete urban grid is of size n then the number of thoroughfares is $2n$ (this includes the thoroughfares in the outer region). There are $2(n-2)$ interior thoroughfares. Placing the axial lines in all of the interior thoroughfares can thus be done

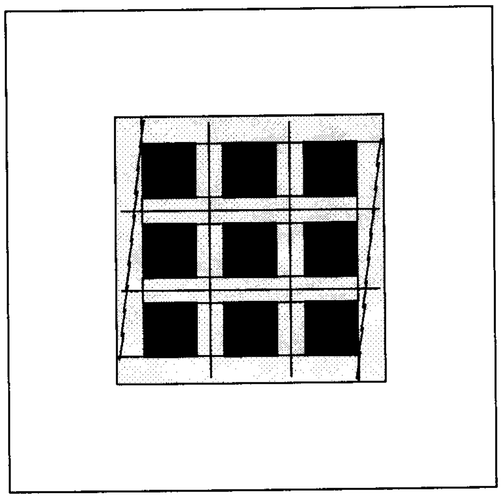


Figure 7.7: The axial map for a complete urban grid of size 4

in $\Theta(n)$ time – placing each line can be done in constant time. Determining which axial lines are necessary in the outer region can clearly be accomplished in $O(1)$ time – it is only necessary to determine what adjacencies are used at each corner intersection of the outer region.

□

The importance of Theorem 7.2.2 is that, irrespective of the convex polygons which make up the convex map of the complete urban grid, determining the axial map of the grid can be done in polynomial time. This holds even though determining the convex map is an NP-Hard problem in general. This theorem means that if the space in town plan can be modelled as a complete urban grid then the axial map for the town can be found in polynomial time. Clearly a central city area with a grid arrangement of streets could be modelled in this fashion. Typically in such a situation the ratio between the length and width of a corridor $l : w$ would be in the range $[2, 10]$. The results above would, however, still hold as all that was required by the definition of an urban grid was that $l > w$.

A further important result follows from Theorem 7.2.2.

Corollary 7.2.1 *The size of the axial map of a minimally partitioned complete urban grid of size n is equal to $2(n-2)+2$.*

In addition to this result a number of other results follow as well.

Corollary 7.2.2 *The minimal set of axial lines to cross all of the adjacencies in a minimally partitioned simple urban grid can be found in polynomial time.*

Proof

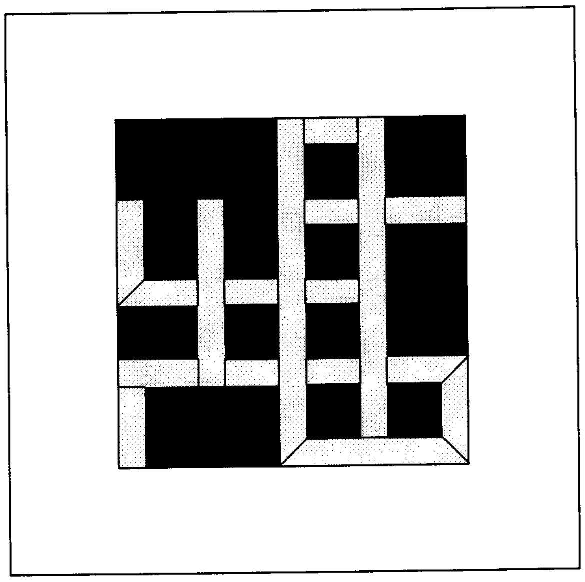


Figure 7.8: An example of minimally partitioning a simple urban grid

A minimally partitioned simple urban grid can be covered with convex polygons which are of similar form to those that cover the complete urban grid. This means that the adjacencies which have to be crossed will be orthogonal at all intersections except where only two corridors meet at right angles at an intersection – a *corner*. In this case the adjacency could be one of the interior edges of the intersection or a diagonal through the intersection. See Figure 7.8 for an example of a possible minimal partition of a simple urban grid.

The partitions which result from placing the different types of adjacencies at corners are equivalent in terms of the size of the partition. Using the diagonals rather than the orthogonal edges to construct the partitions is, however, preferable for the next phase of the process – placing the axial lines. There are two reasons for this. First, if orthogonal adjacencies are used then there are situations where more axial lines are required to cover all of the adjacencies in the partition than if diagonals are used. See Figure 7.9 where the partition using orthogonal adjacencies (a) requires two axial lines and the partition using diagonal adjacencies (b) only requires one axial line. A second, and subsidiary, reason is that the process of placing the axial lines is simplified if only diagonal adjacencies are used in the corners. In the remainder of this section diagonal adjacencies will be used at corners. Figure 7.10 shows a partition of Figure 7.8 under this constraint.

If any thoroughfare has orthogonal adjacencies in it then one axial line must be placed to cross all of the adjacencies in this thoroughfare. Such a thoroughfare may also have diagonal adjacencies in it. These will be crossed by the axial line placed to cross the orthogonal adjacencies.

If any thoroughfare only has diagonal adjacencies in it then, because of the

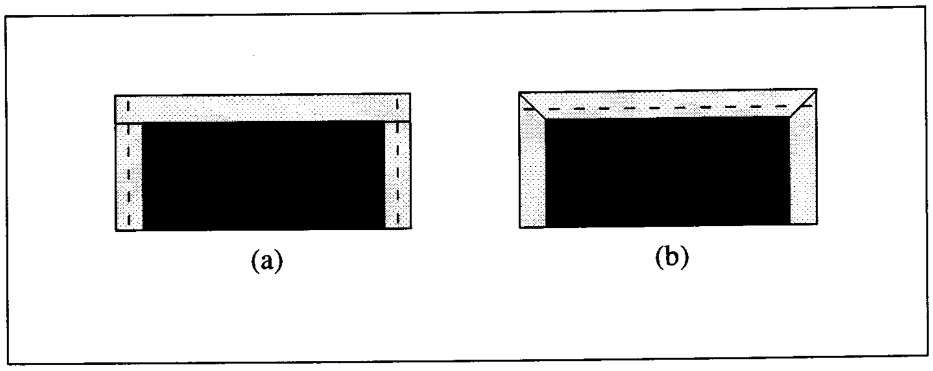


Figure 7.9:

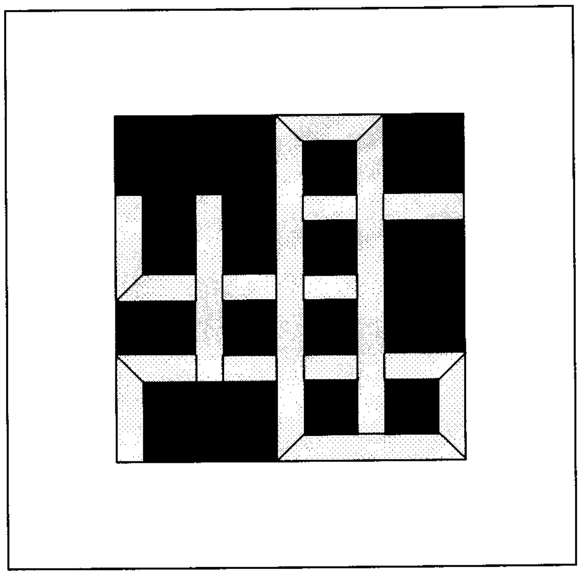


Figure 7.10: An example of minimally partitioning a simple urban grid using only diagonal adjacencies at corners

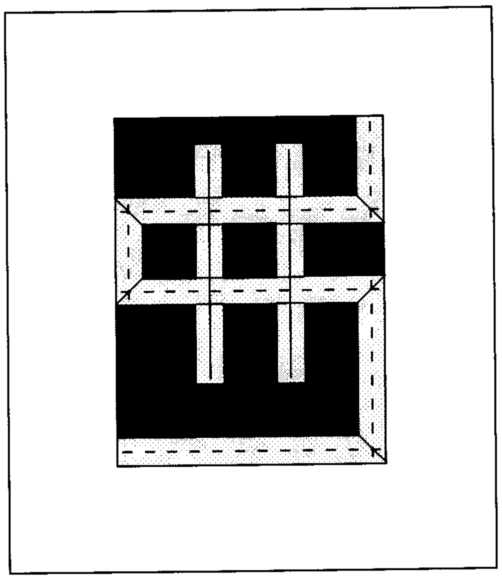


Figure 7.11: An example of limited choice in placing axial lines in a simple urban grid – only 3 of the dashed lines are necessary

geometry of the problem, it can have either one or two such adjacencies – more diagonal adjacencies would result in a non-minimal partition. If a number of thoroughfares with only diagonal adjacencies in them are connected then limited cases of choice can occur. See Figure 7.11 for an example of this. In these situations an axial line through each thoroughfare is sufficient to cross all of the adjacencies but fewer axial lines are actually necessary. In the example in Figure 7.11, 6 axial lines cross all of the diagonal adjacencies but only 3 lines are necessary. In this situation the lines through thoroughfares with only one diagonal adjacency will never be selected and so the problem becomes choosing 3 lines out of 4. Other similar cases could occur but they all have the common property that they would form a chain, or in some cases disjoint chains, of convex polygons. In these chains an axial line crosses the adjacency between a convex polygon and its predecessor in the chain (if it is not the first polygon in the chain) and the adjacency between the convex polygon and its successor (if it is not the last polygon).

It is also possible to get cycles and not just chains of thoroughfares with only diagonal adjacencies. See Figure 7.12 for an example of this situation. Here 4 choice axial lines cross the four diagonal adjacencies but only two are necessary.

Any algorithm to place the minimum number of axial lines in a simple grid would have to be able to determine which of the possible lines to include in the final set of lines.

If choice such as that described above can be resolved in polynomial time, then the problem of finding the axial map can be solved in polynomial time by the following process.

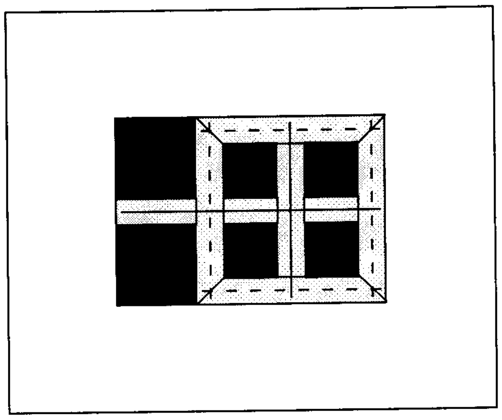


Figure 7.12: An example of the limited choice in placing axial lines in a simple urban grid resulting in a cycle of choice axial lines – only 2 of the dashed lines are necessary

1. Placing an axial line through each thoroughfare which has orthogonal adjacencies.
2. Placing an axial line through each thoroughfare which has two diagonal adjacencies in it.
3. Placing an axial line to cross any diagonal adjacency which has not been crossed in earlier phases of the process. In order to make such a line maximal it should be placed to cross as many adjacencies as possible. In this situation the only adjacencies it could cross would be adjacencies on the boundaries of the two thoroughfares which make up the corner. An axial line which crosses the diagonal adjacency cannot cross more than one such adjacency because of the geometry of the problem. See Figure 7.13 for an example of this situation.
4. Resolving the choice.

Clearly steps 1, 2 and 3 can be done in polynomial time – there are a polynomial number of thoroughfares to consider. To see that step 4 can also be done in polynomial time, note that each axial line in one of these chains or cycles crosses exactly two adjacencies. This means that it is simple to transform each chain or cycle of convex polygons which are adjacent at the diagonals of corners to an instance of ***edge cover*** by mapping adjacencies to vertices and axial lines to edges. ***Edge cover*** gives the minimum number of edges to cover all the vertices in a graph. This maps to the minimum number of axial lines to cross all of the adjacencies in any chain or cycle. ***Edge cover*** can be solved in polynomial time [Garey and Johnson, 1979]. The result follows.

□

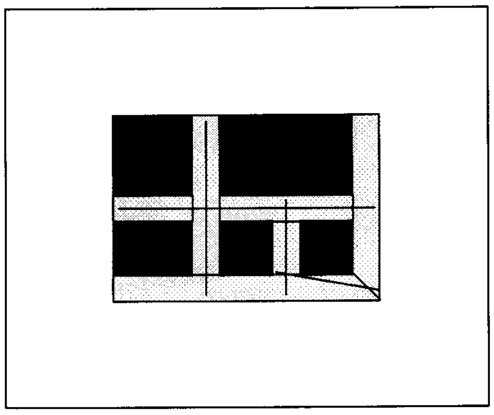


Figure 7.13: An example of placing an axial line to cross a single diagonal adjacency in two thoroughfares in a simple urban grid

Corollary 7.2.3 *The minimal set of axial lines to cross all of the adjacencies in a minimally partitioned general urban grid can be found in polynomial time.*

Proof

The proof in this instance follows from the proof of Corollary 7.2.2. Figure 7.14 shows an example of this version of the problem including an instance of choice. The method of resolving the choice, as discussed above, applies in this case as well. □

From the proofs of corollaries 7.2.2 and 7.2.3 it follows that in both simple and general urban grids the number of axial lines in the axial map is less than or equal to the number of thoroughfares in the grid. The results of these corollaries also mean that if the space in town plan can be modelled as a simple or general urban grid then the axial map for the town can be found in polynomial time. Clearly some towns with grid-like arrangements of streets could be modelled in this fashion.

7.3 Deformed Urban Grids

The results in the previous section show that axial maps can be found in polynomial time for town layouts which can be modelled by urban grids. Unfortunately all town plans cannot be modelled by urban grids – many town plans are not that regular. More general polygons would be needed to model town plans which are not regular grids. These more general polygons could make the problems of finding the convex map and the axial map more difficult. In this section slightly less restricted layouts than in Section 7.2 are considered. The aim here is attempt to determine whether relaxing the restrictions in this manner means that the problems of finding

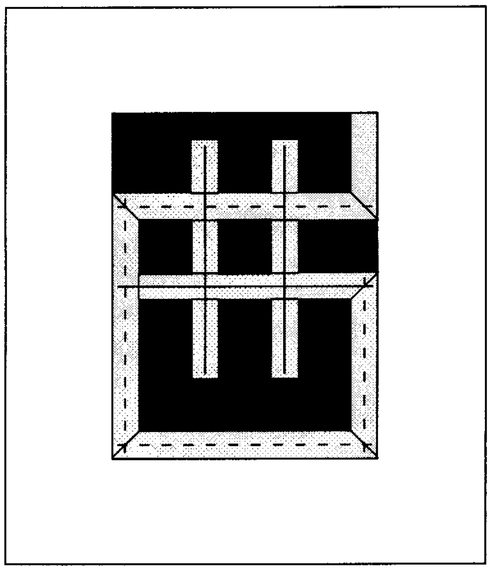


Figure 7.14: An example of choice in placing axial lines in a general urban grid – only 3 of the dashed lines are necessary

the convex and axial maps are still solvable in polynomial time or whether they are more like the general cases – NP-Hard or NP-Complete.

Definition 7.3.1 *A* complete deformed urban grid *of size n is a generalisation of a complete urban grid of size n with the following properties.*

- *The intersections $I_{i,j}, i, j = 1, 2, \dots n$ and corridors $E_p, 1 \leq p \leq 2n(n-1)$ in the grid may be convex quadrilaterals rather than orthogonal squares and rectangles respectively.*
- *Each corridor E_p must be adjacent to exactly two intersections on opposing sides.*
- *Where a corridor E_t is adjacent to an intersection $I_{q,r}$, they must share an edge.*

A deformed urban grid is any subset of a complete deformed urban grid.

In this definition the deformation could very small and the resulting deformed urban grid would be quite regular – similar to an urban grid. The deformation could also be quite large in which case the deformed urban grid could be something more like a general configuration of adjacent rectangles. In the first instance finding the convex map and from that finding the axial map would be similar to solving the corresponding urban grid problems in Section 7.2. In the other extreme the problems could become that of partitioning a general polygon with holes which has been shown to be NP-Hard [Lingas, 1982; O'Rourke and Supowit, 1983] and then solving ***ALP-ALCP*** which is shown to be NP-Complete in Chapter 6. In the remainder of this section deformed urban grids where the deformation is not very great are considered. The interest here is in considering deformed urban grids which could model real world urban layouts which are often grid-like but have to take into account the geographical environment. Figure 1.1 in Chapter 1 gives an idea of the type of layout which it would be useful to model using a deformed urban grid. Another example of a deformed urban grid (which could be an urban layout) is shown in Figure 7.15.

In the deformed urban grids which will be used to model these "grid-like" urban layouts some additional restrictions will be added to the definition of deformed urban grids given above. First, the length of each side, $d_{i,j_a}, a = 1, 2, 3, 4$, of an intersection quadrilateral, $I_{i,j}$, is such that $0 < d_{i,j_a} < 2w$. Second, each corridor E_p must have two opposing sides which are long compared to the other two sides. The ratio of the length of the shortest side to the longest side in any corridor should be approximately $1 : 2$. As stated previously, each of the two short sides of any corridor will also be a side of some intersection.

In the remainder of the thesis any mention of a deformed urban grid should be taken to mean a "grid-like" deformed urban grid. That is, a deformed urban grid with the additional length constraints.

Conjecture 7.3.1 *The problem of finding the minimal set of convex polygons to partition a deformed urban grid is NP-Hard.*

This conjecture is based on the results of Lingas [1982] and O'Rourke and Supowit [1983] that partitioning a polygon with holes is NP-Hard. A deformed urban grid seems to offer enough choice to make the problem difficult to solve exactly. Figure 7.16 shows a possible partition of the deformed urban grid of Figure 7.15. This partition was created by merging corridors and intersections to form larger convex polygons. If the merging allowed parts of intersections to be merged with parts of corridors then there would be more scope for choice and the problem would be harder to solve.

Clearly future work should be focussed on proving or disproving Conjecture 7.3.1.

The algorithm in Figures 7.17 and 7.18 calculates a partition of a deformed urban grid in polynomial time. The algorithm could return a minimal partition but is not guaranteed to do so – this depends on the shapes of the intersections and corridors. If Conjecture 7.3.1 is true then future research can be focussed on determining configurations of intersections and corridors which will allow this algorithm to return minimal solutions. Future research in improving this algorithm or developing a better algorithm is also worthwhile. If Conjecture 7.3.1 is not true then an algorithm which is guaranteed to find a minimal solution in all cases should be sought.

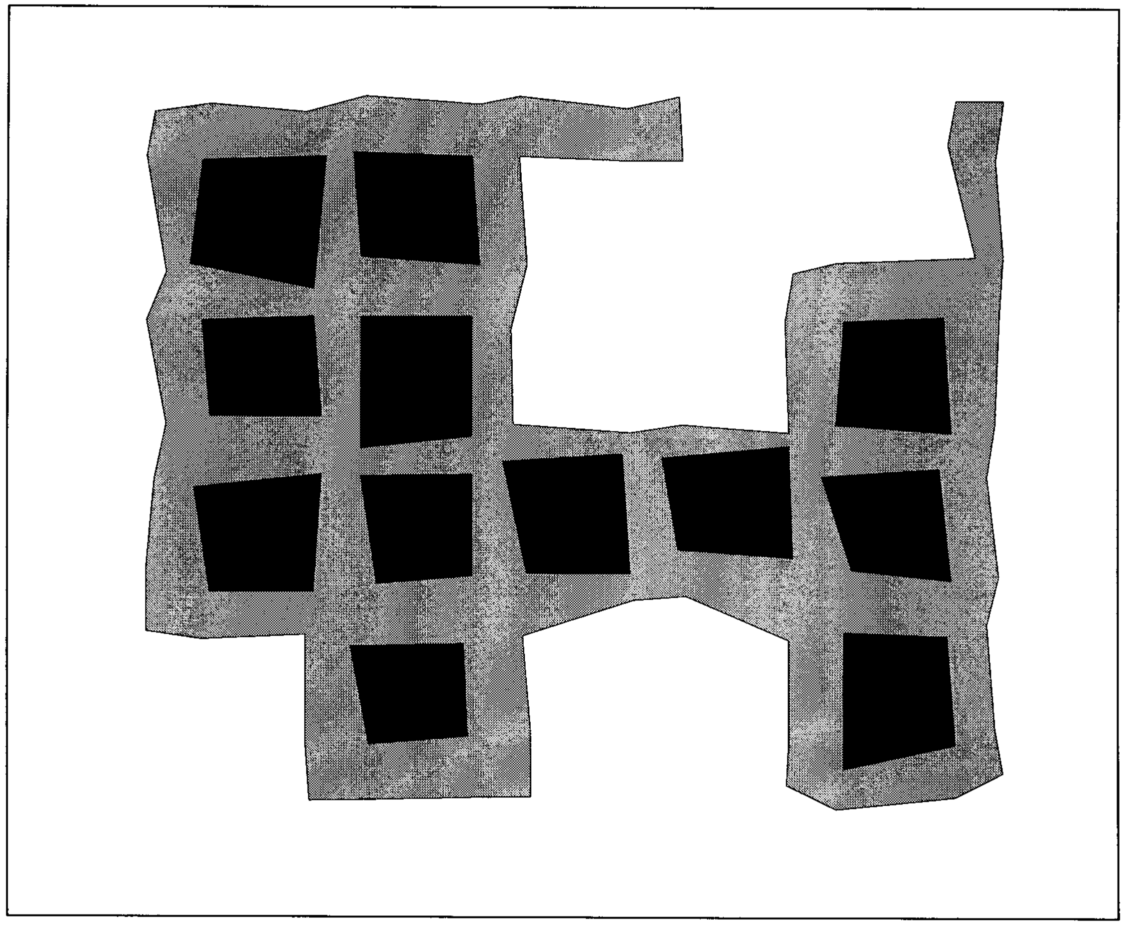


Figure 7.15: An example of a deformed urban grid

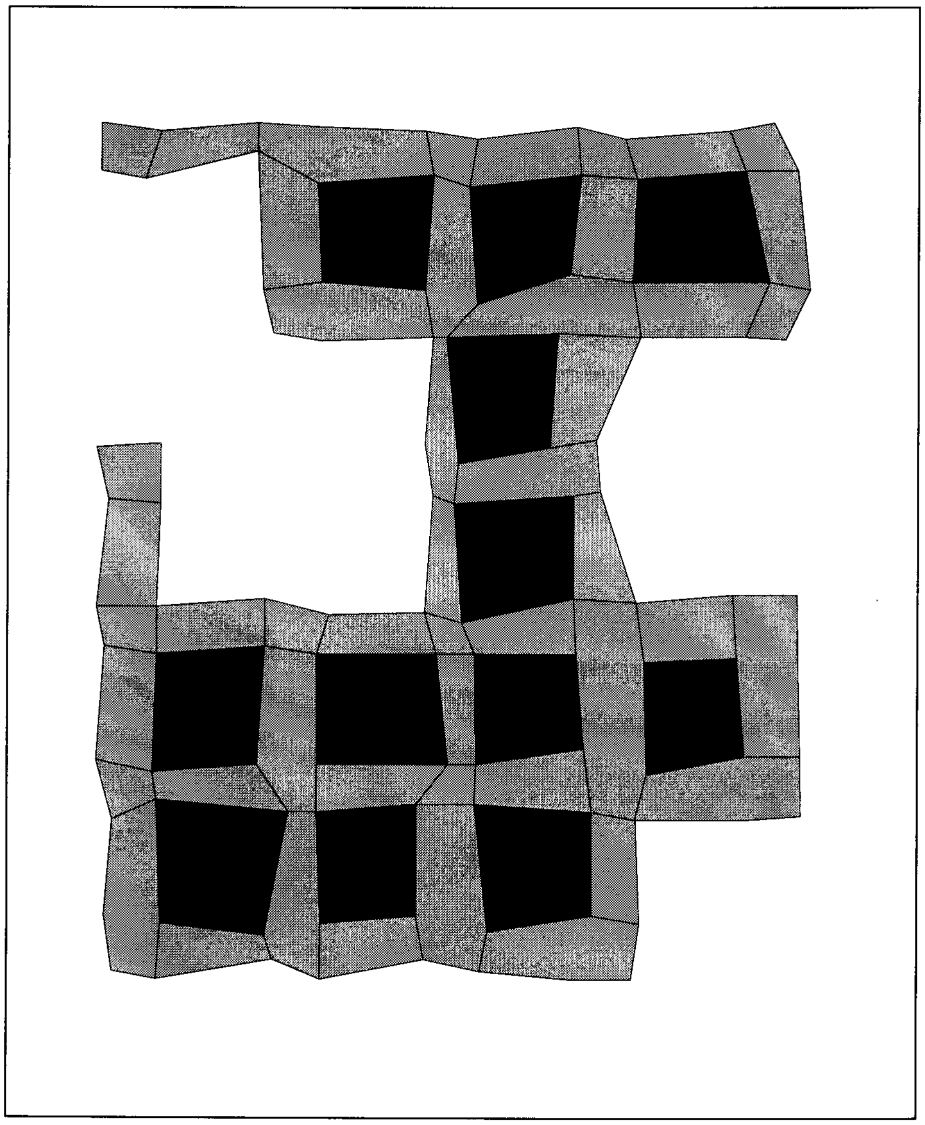


Figure 7.16: A partition of a deformed urban grid

```
{This algorithm assumes that the deformed urban grid is input in
 the form of the thoroughfares defined by the intersections and
 corridors of which they are comprised.}

{Each thoroughfare is an array, where each element in the array
 contains
  information as to whether the polygon is an intersection or a
  corridor and
  the number of the intersection or corridor in the appropriate
  list
 Each element in the corridor or intersection lists contains
  the coordinates of the vertices in that corridor or
  intersection and
  a flag to indicate its status as regards inclusion in the
  final partition}

{$Included(polygon)$ is a function which takes a polygon
 determines if the polygon is an intersection or a corridor and
 checks a flag to see if this polygon has been included in a
 polygon which is already in the partition.
 It returns True or False.}
{$Combine(polygon1, polygon2)$ forms a new polygon which is a
 merging of two convex polygons}
{$Convex(polygon)$ is a function which determines whether a
 given polygon is convex.
 It returns True or False.}
```

Figure 7.17: Partitioning a deformed urban grid – The description of the input into the algorithm and the functions used in the algorithm

```
FOR $i$ from 1 to number of thoroughfares
   IF $included(thoroughfare[i].polygon[1])$
      THEN
         $WorkingPolygon \leftarrow thoroughfare[i].polygon[2]$
         $j \leftarrow 3$
      ELSE
         $WorkingPolygon \leftarrow thoroughfare[i].polygon[1]$
         $j \leftarrow 2$
   WHILE $j \leq$ number of polygons in thoroughfare
      IF $NOT(included(thoroughfare[i].polygon[j]))$
         THEN
         {It might be possible to merge some polygons}
            $TestPolygon \leftarrow Combine(WorkingPolygon,$
            $thoroughfare[i].polygon[j]))$
            IF $Convex(Testpolygon)$
               THEN
                  $WorkingPolygon \leftarrow TestPolygon$
                  $j \leftarrow j + 1$
               ELSE
                  IF $WorkingPolygon$ is not a single intersection
                     THEN
                        Add $WorkingPolygon$ to $Partition$
                        Mark all corridors and intersections in
                        $WorkingPolygon$ as included
                        $WorkingPolygon \leftarrow thoroughfare[i].polygon[j]$
                        $j \leftarrow j + 1$
                     ELSE
                        $WorkingPolygon \leftarrow thoroughfare[i].polygon[j]$
                        $j \leftarrow j + 1$
         ELSE
         {The next polygon is an intersection which was already
          included in a cross thoroughfare.
          This intersection cannot be used again.}
            Add $WorkingPolygon$ to $Partition$
            Mark all corridors and intersections in $WorkingPolygon$
            as included
            $WorkingPolygon \leftarrow thoroughfare[i].polygon[j]$
            $j \leftarrow j + 1$
FOR $k$ from 1 to Number of Intersections
   IF $NOT(Included(intersection[k]))$
      Add $Intersection[k]$ to $Partition$
```

Figure 7.18: Partitioning a deformed urban grid

To see how the algorithm works consider the portion of a deformed urban grid shown in Figure 7.19. In this example only two thoroughfares are shown but this is enough to illustrate how the algorithm computes a partition.

Assume that the horizontal thoroughfare is considered first. This thoroughfare is made up of intersection a, corridor A, intersection b, corridor B and intersection c. The first polygon in this thoroughfare is a. This polygon has not yet been included in a polygon in the final partition (line 01) so the $WorkingPolygon$ becomes that polygon and j is initialised appropriately (lines 06 and 07). This is the beginning of the phase of attempting to merge the polygons in that horizontal thoroughfare into a smaller number of convex polygons.

The `WHILE` loop (lines 08 to 30) is then entered. In the `WHILE` loop the first step is to test whether the next polygon in the thoroughfare has already been included in a polygon in the final partition (line 09). In this case the next polygon is A. A is a corridor and could not have been included in a polygon in the final partition – this is the first and only time it is considered. The $WorkingPolygon$, a, and A are then combined to form the $TestPolygon$ (line 11). $TestPolygon$ is then tested to see if it is convex (line 12). It is convex, a and A make a convex polygon, and so $WorkingPolygon$ becomes this new polygon (line 14) and the pointer to the next polygon to be considered is incremented (line 15). The `WHILE` loop will then be re-entered to consider the next polygon in the thoroughfare.

This next polygon is the intersection b. b has not been included – this is the first time it is being considered – and so is combined with $WorkingPolygon$ to make a new $TestPolygon$. As before $TestPolygon$ is convex and so this polygon becomes the new $WorkingPolygon$.

When the algorithm considers the next polygon in the thoroughfare, now B, $TestPolygon$ is not convex. This means that $WorkingPolygon$ will be saved as a component of the final partition (provided it is not made up of only a single intersection (line 17)). Lines 19 to 22 save the current $WorkingPolygon$, mark the appropriate intersection and corridor polygons as included and make the new $WorkingPolygon$ the next polygon in the thoroughfare. In this case $WorkingPolygon$ becomes B.

In the case where $WorkingPolygon$ was a single intersection then lines 24 and 25 would be executed. Any $WorkingPolygon$ which is only a single intersection would not be immediately saved as part of the final partition as it could later be considered for combining with other polygons when the vertical thoroughfares are considered.

In this example the algorithm would continue and c would be merged with B. This first thoroughfare would thus be partitioned into two convex polygons.

The algorithm would proceed in a similar fashion for all other horizontal thoroughfares before considering the thoroughfare consisting of intersection a, corridor C, intersection d, corridor D and intersection e. In this case the first intersection

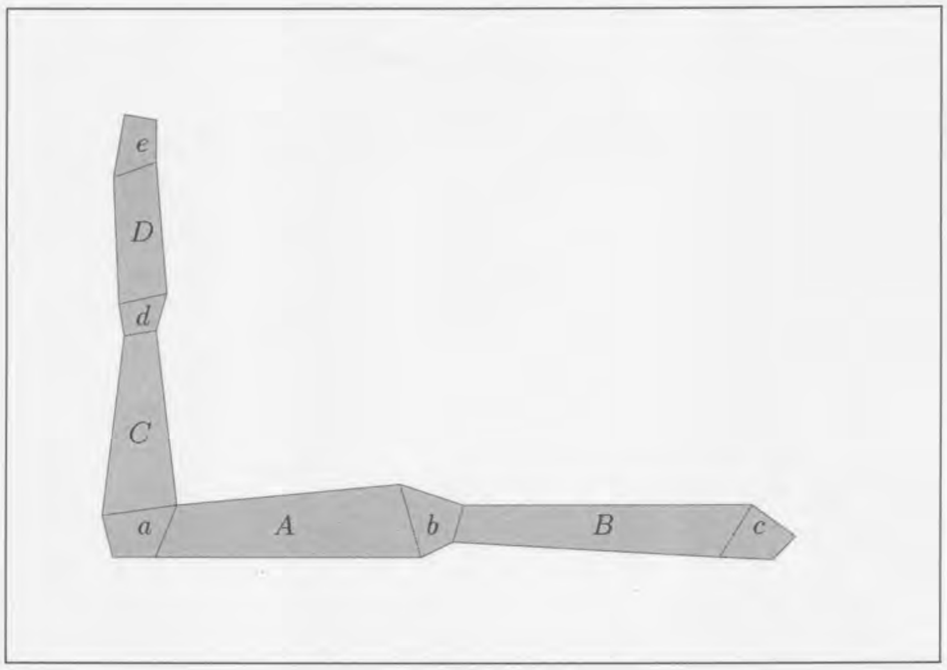


Figure 7.19: A portion of a deformed urban grid – to illustrate the algorithm

in the thoroughfare has already been included in some other larger convex polygon and cannot be reused. Lines 03 and 04 are executed – $WorkingPolygon$ becomes C.

The algorithm would then continue as before to process this thoroughfare and all of the other vertical thoroughfares.

The final stage of the algorithm (lines 31 to 33) would be to ensure that any intersections which were not included in larger convex polygons when they were considered as part of a horizontal thoroughfare and later as part of vertical thoroughfare are added to the final partition.

Figure 7.20 shows the convex polygons (in gray) which would have been generated by the algorithm in Figures 7.17 and 7.18 working on the deformed urban grid in Figure 7.15 once the lower thoroughfares have been considered.

Using the algorithm in Figures 7.17 and 7.18 a partition for a deformed urban grid can be found. The next step in the process would be to place the minimum number of axial lines to cross all of the adjacencies between the convex polygons in the partition.

Conjecture 7.3.2 *The minimal set of axial lines to cross the adjacencies in the convex map of a deformed urban grid can be found in polynomial time.*

This conjecture seems to be true because the convex polygons which make up the convex map of the area are likely to be aligned along deformed thorough-

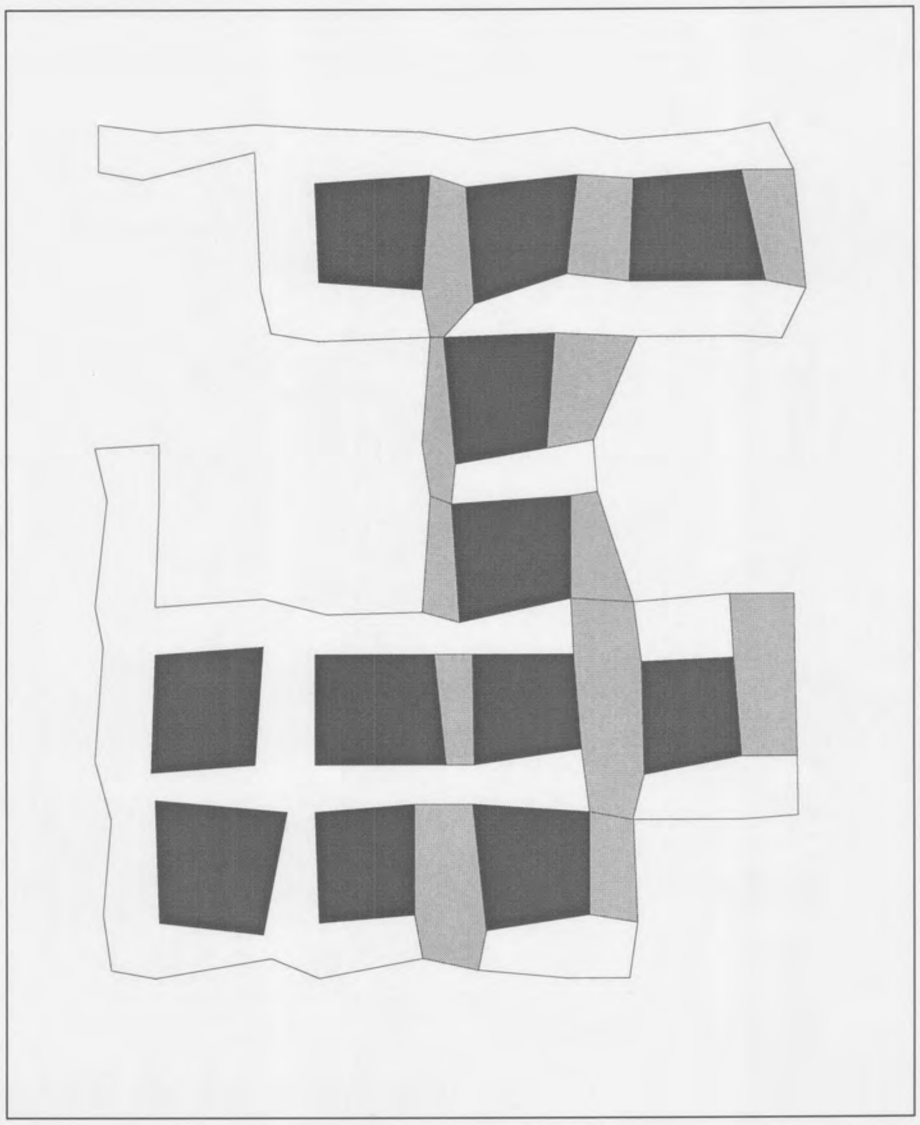


Figure 7.20: A partial solution from the algorithm to partition a deformed urban grid

fares. The shapes of the corridors (deformed rectangles) and intersections (deformed squares) mean that each intersection can potentially be merged with the corridors in one thoroughfare but it is unlikely that an intersection could be merged with corridors from more than one thoroughfare to form a convex polygon. Thus the number of axial lines needed to cover each thoroughfare can be found and the final solution can be determined by removing any unnecessary lines from this set. Figure 7.21 shows the axial lines which would be placed to cross the adjacencies in the (not necessarily) minimal partition of Figure 7.16. This figure supports the idea that there is little choice in placing the axial lines and that a solution in polynomial time is likely.

Future research can be focussed on proving Conjecture 7.3.2 by finding a polynomial time algorithm to place a minimal set of lines or on disproving the conjecture by extending the NP-Completeness proofs presented above to cover this situation as well.

The results presented in this section of the thesis are important because they show that if the space in the original town plan can be modelled as a deformed urban grid then it is likely (if Conjecture 7.3.2 is true) that an axial map can be found in polynomial time.

7.4 More general urban polygons

In reality it is not likely to be possible to model all urban layouts (polygons with holes) as deformed urban grids – the layouts are more irregular than this. Here the results of Lingas [1982] and O'Rourke and Supowit [1983] show that the partitioning problem is NP-Hard. In Chapter 5 of this thesis it is shown that the axial line placement problem restricted to rectangles and lines of arbitrary orientation is NP-Complete and in Chapter 6 the axial line placement problem for convex polygons and axial lines of arbitrary orientation is shown to be NP-Complete. These results indicate strongly that finding the convex map of an urban layout and the placing of the minimum number of axial lines to cross the adjacencies in the convex map of some urban layout are likely to be NP-Complete. The first step in future research in this regard would be to prove these claims. Subsequent research could then be concentrated on approximation algorithms to give "reasonable" approximations to the convex map and axial map for urban layouts of this form.

7.5 Conclusion

This chapter shows that the axial line placement problem can be solved exactly in polynomial time for some restricted cases – a complete urban grid, a simple urban grid and a general urban grid. It also seems (but is not proven here) that an

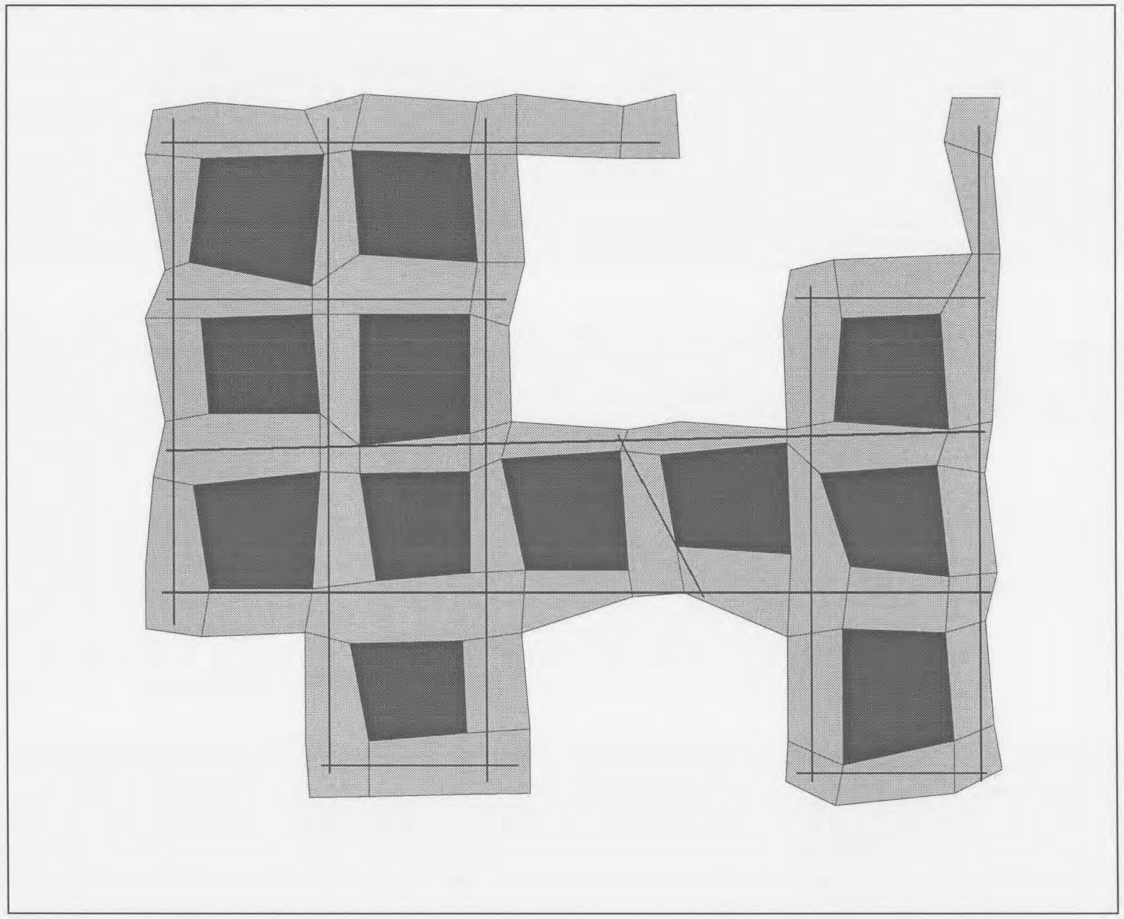


Figure 7.21: Placing the axial lines to cross the adjacencies in a partitioned deformed urban grid

exact solution is possible in polynomial time for a deformed urban grid. The more general problem – placing a minimal set of axial lines to cross all of the adjacencies between arbitrary convex polygons – was shown to be NP-Complete in Chapter 6 of this thesis and this suggests that finding polynomial time solutions in general town plans could also be NP-Complete and heuristics will have to be used to find acceptable approximations.

The next chapter of the thesis presents some ideas for further research to build on the results of this work.

Chapter 8

Future Research

8.1 Introduction

This thesis addresses a number of computational problems which arise from the potential automation of the space syntax method [Hillier *et al.*, 1983]. The research focussed specifically on the problems associated with the placing of axial lines in a convex map. It was, however, impossible to address all of the possible research questions which were originally identified in this area. In addition, as the work for this thesis progressed other problems were raised. The future research which stems from the work discussed in this thesis is thus of two types

1. problems which were presented in Chapter 3 but were not tackled at all in this research,
2. problems which arose in the course of pursuing this research but were not addressed (or at least not fully addressed) at the time.

Sections 8.2 and 8.3 of this chapter discuss the problems which fall into these two groups in more detail.

8.2 Open problems

This thesis considers a number of axial line placement problems which have derived from the original idea of automating the Space Syntax Analysis method. In all cases the problems which have been considered are where multiple crossings of adjacencies are permitted. If the restriction is made that each adjacency is crossed *exactly* once then the list of problems below are still open questions.

1. Restricting the problem to dealing with orthogonally aligned rectangles and axial lines which are parallel to the edges of the rectangles and where each adjacency must be crossed by *exactly* one axial line.

2. Restricting the problem to dealing with orthogonally aligned rectangles but where the axial lines are not necessarily orthogonal and where each adjacency must be crossed by *exactly* one axial line.

3. Considering the problem of general convex polygons (not restricted to the form which would occur in the town planning domain) where the adjacencies between polygons are crossed by non-orthogonal axial lines and where each adjacency must be crossed by *exactly* one axial line.

4. Considering the problem of convex polygons, which represent the deformed grid in the town planning domain, where the adjacencies between polygons are crossed by non-orthogonal lines and where each adjacency must be crossed by *exactly* one axial line.

Based on the results given in Chapters 4, 5 and 6 there is evidence to suggest that each of these problems is NP-Complete. Future research would have to substantiate this claim. Once this has been done heuristics will have to be derived and tested to produce approximations to the solutions to these problems.

Heuristics which are likely to work well in the case of (1) (and possibly (2), (3) and (4)) above are

- choosing the longest line first.
- choosing the line with the highest number of single crossings first.
- choosing the line with the highest number of grouped single crossings first.
- choosing essential lines first (as was done for the heuristic discussed in Chapter 4) then using one or all of the other approaches suggested here. This makes sense as the adjacency which is crossed only by the essential line will generate a line which must be in the final set of lines.

Again a profitable area of research could be to study special cases of these problems where exact solutions can be found in polynomial time.

8.3 Future research arising from this thesis

The problems here are related to the various problems which have been solved as part of this work for this thesis. They are given below grouped according to the variation of the axial line placement problem in which they occurred.

- ***ALP-OLOR***

- In Chapter 4 an algorithm which finds a non-redundant set of axial lines for ***ALP-OLOR*** is discussed. In some instances this algorithm does extra work. Future work could focus on reducing the amount of work done by this "most uncrossed adjacencies" heuristic algorithm. In particular, attempting to address the issue of redundant calculations which are made for the collection of rectangles shown in Figure 4.22 or similar configurations.
- Developing other heuristics to produce approximate solutions to the exact solution could also be a fruitful area of research.
- Section 4.7.1 presents some variations of ***ALP-OLOR*** which can be solved in polynomial time. Also polynomial time algorithms for the orthogonal axial line placement problem for chains of orthogonal rectangles and trees of orthogonal rectangles are presented. Future research can be focussed on finding other arrangements of rectangles which are more general than a tree of rectangles which can be solved in polynomial time.
- An interesting, but not directly related, area of further research is in the generation of test data. Generating non-trivial trees of adjacent but non-overlapping rectangles proved to be relatively complex in this research.

• ***ALP-ALOR***

- In Chapter 5 some ideas for heuristics to produce approximate solutions to the exact solution for ***ALP-ALOR*** are presented. These ideas need to be more fully developed or better heuristics derived. Once this has been done, any resulting heuristics should be implemented and tested for some carefully selected test cases.
- A related area of further research would be looking at other approaches, for example genetic algorithms, for finding approximate solutions.
- The idea of special cases of the problem which can be solved in polynomial time was also very briefly raised in Chapter 5. This is an area in which a lot of additional work could be done.

• ***ALP-ALCP***

- In Chapter 6 ***ALP-ALCP*** is shown to be NP-Complete. This means that finding reasonable approximations to the exact solution in polynomial time becomes important. Thus heuristics for this problem should be developed and tested.
- As part of the work in developing heuristics for this problem it would be interesting to consider other approaches, for example genetic algorithms, to generate approximate solutions.

  - The idea of special cases of the problem which can be solved in polynomial time was also very briefly raised in Chapter 6. This is an area in which a lot of additional work could be done.

- Placing Axial Lines in Town Plans
  - Conjecture 7.3.1 of Chapter 7 states that "The problem of finding the minimal set of convex polygons to partition a deformed urban grid is NP-Hard." This conjecture is based on the results of Lingas [1982] and O'Rourke and Supowit [1983] that partitioning a polygon with holes is NP-Hard. A deformed urban grid seems to offer enough choice to make the problem difficult to solve exactly. Clearly future work should be focussed on proving or disproving this conjecture.
  - Conjecture 7.3.2 states that "The minimal set of axial lines to cross the adjacencies in the convex map of a deformed urban grid can be found in polynomial time." Future research can be focussed on proving this conjecture by finding a polynomial time algorithm or on disproving the conjecture by extending the NP-Completeness proofs to cover this situation as well.
  - Not all urban layouts can be modelled by deformed urban grids as defined in Chapter 7. A fruitful area of research would be considering more general configurations of polygons with holes which could represent urban layouts. The focus here should be on determining whether the problem remains NP-Complete or whether the restriction to modelling urban layouts is enough to remove the choice which complicates the problem.

8.4 Conclusion

This thesis has considered a number of Axial Line Placement problems but the area is still new and there are lots of other problems which should be of interest to computer scientists. Some of these problems are discussed above but the list is not complete and it is likely that the list will grow rather than shrink as more work is done in this area.

Chapter 9

Conclusion

9.1 Introduction

Computers can be used to automate many mundane and boring tasks and can thus free the professional to concentrate on the more intellectually demanding aspects of many jobs. The original intent of this thesis was to consider the possibility of automating some or all of the task of applying the Space Syntax method to an urban layout. An architect or town planner would apply the space syntax method ([Mills, 1992]) to a town or city in 4 main steps.

1. Finding the "deformed grid" of the urban layout.
2. Creating the convex map of the area.
3. Creating the axial map of the area from the convex map.
4. Combining the information from the convex map and the axial map to produce an integration factor for the town/city.

Currently the town planner or architect does all but the final analysis phase by hand. The initial aim of this thesis was thus to determine which phases of this work could potentially be automated in order to free the architect or town planner to concentrate on the more intellectually stimulating design phases.

In the introduction to this thesis it is shown that all of these phases could benefit from automation and that the potential automation of space syntax gives rise to many areas of research. These areas include image processing, computational geometry and algorithms. However, tackling all of these areas would be an immense tasks. For this reason, the problems of separating space from non-space, determining the convex map and the final analysis stage with its associated algorithms were not considered as part of this research. The decision was made to focus the research for this PhD on the problem of finding the axial lines which cross all of the shared boundaries between the convex polygons in the convex map, that is finding the axial map for a given layout – the *Axial Line Placement* problem or ***ALP***. This problem on its own is still very big and could not be solved entirely. This thesis only tackled a small subset of the research questions proposed in Chapter 3 but the contributions of this research (discussed in Section 9.2) can go some way towards answering the original research question and are interesting problems in computational geometry in their own right.

9.2 Contributions of this thesis

As discussed in Section 9.1 above, this thesis only considered a small subset of the possible problems which arise in Axial Line Placement. The process which was applied in the research was to start with simplifications of ***ALP*** and then to build towards solving the general problem. The specific contributions that were made by this research are presented below.

- This research proved that the axial line placement problem for orthogonal axial lines and collections of adjacent orthogonal rectangles (***ALP-OLOR***) is NP-Complete. This proof was accomplished by a transformation from ***biconnected planar vertex cover*** to ***stick diagram*** and hence to ***ALP-OLOR***. As ***ALP-OLOR*** is NP-Complete the research then focussed on approximations and the thesis presents a heuristic algorithm which produces a non-redundant set of maximal axial lines to cross all of the adjacencies in a collection of adjacent orthogonal rectangles. This heuristic algorithm was implemented and tested on a number of configurations of adjacent rectangles and was shown to produce reasonable approximations in polynomial time. The research then focussed on special cases of ***ALP-OLOR*** which can be solved in polynomial time. Specifically it was shown that exact polynomial-time solutions can be found for configurations of adjacent orthogonal rectangles which can be converted to interval graphs; chains of adjacent orthogonal rectangles and trees of adjacent orthogonal rectangles.

- The next phase of the research involved relaxing the restriction that the axial lines had to be orthogonal. The research proved that the axial line placement problem for axial lines with arbitrary orientation and collections of adjacent orthogonal rectangles (***ALP-ALOR***) is NP-Complete. This proof was also accomplished by a transformation from ***biconnected planar vertex cover*** to ***stick diagram*** and hence to ***ALP-ALOR***. Some heuristics for ***ALP-ALOR*** were derived and are presented in this thesis.

- The next step in the research was to allow general convex polygons instead of orthogonal rectangles. The research proved that the axial line placement problem for axial lines with arbitrary orientation and collections of adjacent convex polygons (***ALP-ALCP***) is NP-Complete.

- After considering the simplifications discussed above, the research emphasis was shifted to focus on finding axial lines to cross the adjacencies between convex polygons in a situation which is representative of the real world problem, i.e. covering the convex map of some town with an axial map. In this phase of the research it was proved that the minimum partition of complete, simple and general urban grids can be found in polynomial time. It was also proved that finding a minimal set of maximal axial lines for complete, simple and general urban grids can be done in polynomial time. It was conjectured that finding the convex map of a deformed urban grid is NP-Complete or NP-Hard (based on the result of Culberson and Reckhow [1994]). Assuming that this conjecture is true a polynomial-time heuristic algorithm to produce a partition of a deformed urban grid was developed. It was further conjectured that the axial map of a deformed urban grid can be found in polynomial time.

In summary this research has shown that ***ALP-OLOR***, ***ALP-ALOR*** and ***ALP-ALCP*** are NP-Complete. It has also shown that placing axial lines in some simplified configurations of polygons can be done in polynomial-time. There are, however, still many open problems in this area. These are discussed in some detail in Chapter 8 and discussed briefly below.

9.3 Future work

The results of the research discussed in this thesis do not answer all the questions raised in ***ALP***. In fact, as the research progressed more problems were raised than were solved. Chapter 8 discusses in some detail the open problems which were not tackled in this research and the subsidiary problems which arose out of the problems which were tackled.

The biggest area of future research is in considering the groups of problems (similar to those discussed in Section 9.2) where any adjacency must be crossed by exactly one axial line – in contrast to this research where multiple crossings of adjacencies were permitted. It has been conjectured [O'Rourke, 1999] that these problems can be solved in polynomial time but as yet this has not been proved.

Another important area of research is in extending the results of this research to address the original problem of finding the convex map and axial map of some urban layout. The partitioning problem for polygons with holes is NP-Hard so heuristic algorithms will have to be found to return acceptable approximations to the exact solution. It seems that when the architect or urban designer applies the Space Syntax Analysis technique that she/he uses heuristics in her/his solution for the convex map. This means that an approximate solution is likely to be acceptable to the urban designer especially if the solution is similar to her/his own. This suggests an artificial intelligence approach to designing the heuristic could be a fruitful area of research. A similar comment applies for determining the axial map which covers the convex map of some urban layout.

9.4 Overall Conclusions

The original aim of this thesis – automating the Space Syntax method – was clearly too big a task to be undertaken in a single PhD. With this in mind the research was focussed only on the computational problems which arise with the placing of axial lines. Even with this scaling down, the research area was still very big and only some of the problems which were identified were tackled. The work which was completed is, however, important in that a number of new NP-Complete problems (***ALP-OLOR***, ***ALP-ALOR*** and ***ALP-ALCP***) were identified and also because some new and interesting problems in the area of computational geometry have been introduced.