DIMACS Technical Report 92-52
December 1992

A Tight Bound for Edge Guards in Monotone Polygons

by

Iliana Bjorling-Sachs[1]
Department of Computer Science
Rutgers University
New Brunswick, New Jersey 08903

Diane L. Souvaine[2,3]
Department of Computer Science
Rutgers University
New Brunswick, New Jersey 08903

[1]Supported by DIMACS research assistantship
[2]Permanent Member of DIMACS
[3]Supported in part by National Science Foundation Grants #CCR-88-03549 and #CCR-91-04732.

DIMACS is a cooperative project of Rutgers University, Princeton University, AT&T Bell Laboratories and Bellcore.

DIMACS is an NSF Science and Technology Center, funded under contract STC–88–09648; and also receives support from the New Jersey Commission on Science and Technology.

ABSTRACT

In this paper we examine a variation of the art gallery problem with the gallery restricted to the shape of a monotone polygon and with each guard allowed to patrol a designated edge of the polygon. We show that any n-vertex gallery of this type is completely covered by at most $\lceil (n-2)/5 \rceil$ guards and that this bound is tight. In addition we give an algorithm for placing the guards, which runs in $O(n)$ time.

1 Introduction

The original art gallery problem asks "What is the minimum number of guards needed to watch every point inside an art gallery" where the art gallery is modelled as a simple polygon and the guards as points inside the polygon. A point x is visible to a guard y as long as the line segment $\overline{xy}$ does not intersect the exterior of the polygon. The polygon is covered if every point inside it is seen by at least one guard. In 1975 Chvatal showed that no polygon with n vertices requires more than $\lfloor n/3 \rfloor$ guards and that polygons exist for which this number of guards is necessary [3].

Variations of the original art gallery problem have attracted considerable attention since that time, and numerous results have been the focus of a book by O'Rourke [5] and of an extensive survey paper by Shermer [6]. One variation posed in 1981 by Toussaint and still unsolved asks for the minimum number of guards necessary to cover any polygon of n vertices if the guards are allowed the freedom to move along individual edges of the polygon. Guards of this type are called *edge guards*. For edge guards the notion of *weak visibility* is used. A point inside the polygon is considered visible as long as it is seen by some guard at some point during the walk along the edge.

Here we show a tight bound of $\lceil (n-2)/5 \rceil$ for the number of edge guards needed when the gallery is restricted to the shape of a monotone polygon. The lower bound is given by means of an example in section 2 and a proof of the upper bound is given in section 3. The proof is constructive and leads to an algorithm for placing the guards in time linear in the number of vertices of the polygon. Algorithmic aspects are discussed in section 4. It is of interest to note that allowing the guards to move away from the edges, i.e. letting each guard move in a straight line between any two points in the polygon, does not improve the bound. Guards of this type are called line guards. A tight bound of $\lfloor (n+2)/5 \rfloor$ line guards for monotone polygons has been shown by Aggarwal [1].

2 The lower bound is $\lceil (n-2)/5 \rceil$.

The behavior of the $\lceil (n-2)/5 \rceil$ bound can be seen in table 1. Starting with the simplest

Table 1

n	$\lceil (n-2)/5 \rceil$	n	$\lceil (n-2)/5 \rceil$
3	1	12	2
4	1	13	3
5	1	14	3
6	1	15	3
7	1	16	3
8	2	17	3
9	2	18	4
10	2	...	...
11	2		

polygon possible, the triangle, exactly one edge guard is clearly needed. When the number of vertices is increased to eight, the bound jumps to two. A polygon with eight vertices requiring two edge guards is shown in figure 1. The topmost part of this polygon is seen as long as a guard is assigned to edge a, b, g or h. Since no guard on one of these edges also sees the bottom part of the polygon, an additional guard is needed e.g. on edge d. Continuing to increase the number of vertices, the bound becomes three when n reaches thirteen and thereafter it jumps by one for each additional five vertices. A polygon with thirteen vertices requiring three edge guards is shown in figure 2.

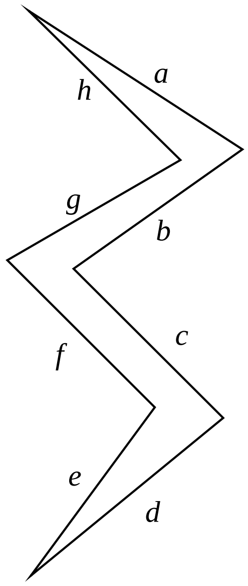

Figure 1: A polygon with 8 vertices requiring 2 edge guards.

Here a group of five edges j, k, l, m and n have been inserted into the polygon in figure 1. Edge m in this group can only be seen by one of the edges in the group. Since none of these edges can see either the top or the bottom part of the polygon, two additional guards are needed e.g. on edges a and d. The total is thus three edge guards.

The polygon in figure 2 can be generalized by adding an arbitrary number of identical center groups to the original polygon (see figure 3). Each addition requires one new edge guard and we have an example of a polygon where in addition to the two edge guards needed for the original eight vertices, an extra guard is required for each additional five vertices. Obviously 1 - 4 vertices can be added to each such polygon without decreasing the number of guards needed. This establishes the lower bound.

3 The upper bound is $\lceil (n-2)/5 \rceil$.

Let P be a polygon with n vertices and assume P is monotone in the y-direction. To establish the upper bound we start by triangulating P. Monotone polygons are easily triangulated

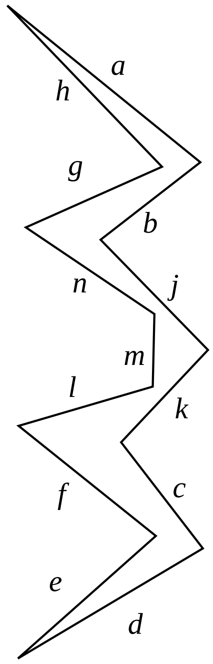

Figure 2: A polygon with 13 vertices requiring 3 edge guards.

using an algorithm by Garey, Johnson, Preparata and Tarjan [4]. We convert this algorithm into an algorithm for assigning edge guards. The original algorithm starts at the topmost vertex and moves downward creating triangles by drawing internal diagonals between vertices. We show that as the triangles are formed, they can be grouped in such a way that the union of triangles in each group is covered by one edge guard and every group except one either contains at least five triangles or contains four triangles with a following group containing at least six triangles (see figure 4). The bottom group will consist of any triangles left over after the last full group was formed and thus may have fewer than five triangles. Since the triangulation of P yields $n-2$ triangles, there are at most $(n-2)/5$ complete groups and $\lceil (n-2)/5 \rceil$ groups total. With one edge guard assigned to each group the number of guards sufficient to cover P is thus $\lceil (n-2)/5 \rceil$ and the upper bound is the same as the lower bound.

The algorithm by Garey *et al.* produces a triangulation with a very predictable structure. It will have certain properties which we take advantage of to assign the edge guards. To see what these properties are and how they come about, we next review the triangulation algorithm. The first step is to sort the vertices in order of descending y-coordinate. The two topmost vertices are then pushed onto a stack and the rest of the vertices are examined in sorted order. An examined vertex may be pushed onto the stack, which at all times is maintained in such a way that the bottom entry is a vertex on one side (left or right) of P and all the other stack entries are vertices lying in consecutive order on the opposite side of P. Let p_i be the next vertex to be examined and let the stack have entries $v_0, \ldots, v_t$. Vertex p_i is adjacent to v_0 or v_t (or both if p_i is the last vertex in sorted order). If p_i is adjacent to

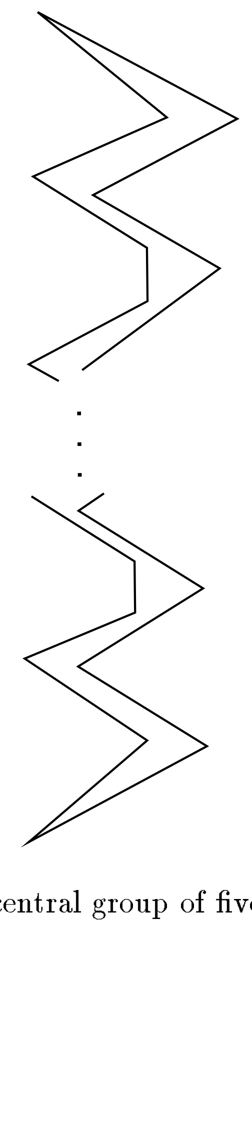

Figure 3: A polygon where each central group of five vertices requires one edge guards.

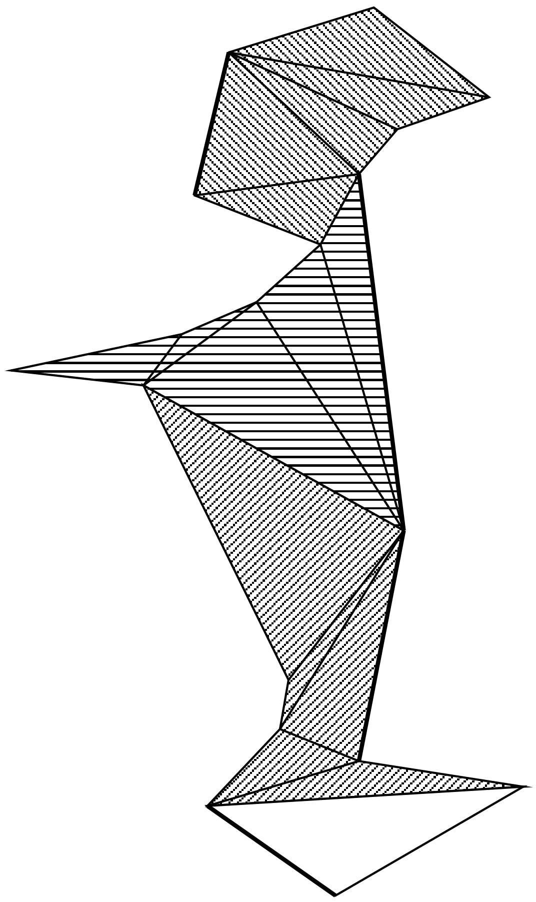

Figure 4: One edge guard indicated by a heavy edge is assigned per group.

v_0, diagonals are drawn between p_i and $v_1, \ldots, v_t$ on the stack (see figure 5). Vertices $v_1, \ldots, v_t$ are popped from the stack and vertex p_i is pushed. If instead p_i is adjacent to v_t, the angle

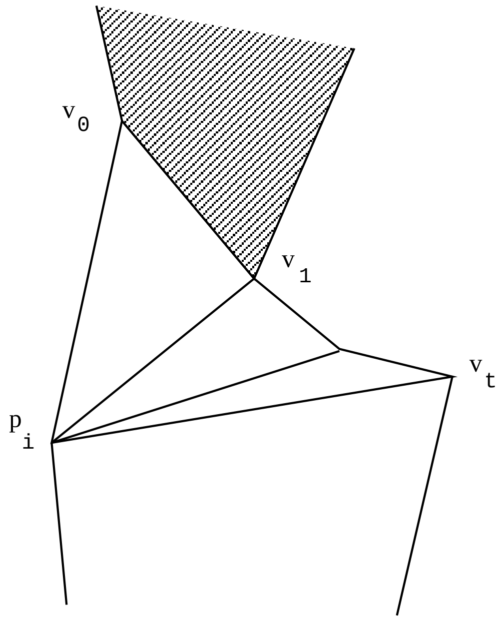

Figure 5: Examined vertex p_i is adjacent to bottom stack entry v_0. One or more cross-triangles are produced.

$v_{t-1}v_tp_i$ is examined. If this angle is reflex then p_i is pushed onto the stack, else the diagonal p_iv_{t-1} can be drawn (see figure 6), v_t is popped from the stack and the angle formed by p_i and the new top two entries on the stack is examined.

We see that the diagonals drawn are of two types, *cross-diagonals* with end points on opposite sides of P and *side-diagonals* with both end points on the same side of P. The triangles created by drawing each type of diagonal are of corresponding types. *Cross-triangles* have two vertices on one chain and the third vertex on the opposite chain and thus stretch across the polygon from one side to the other. *Side-triangles* have all vertices on the same chain and can be said to lie either on the right side or the left side of the polygon. Since side-diagonals are drawn only when the stack holds a number of consecutive vertices all on one side of P and the vertex under examination also lies on the same side of P (see figure 6), we can make the following observation regarding the creation of side-triangles:

Observation 1 *The extent in the y-direction of a side-triangle or group of adjacent side-triangles along one chain is spanned by a single edge on the opposite chain.*

Our new augmented algorithm will use this observation to group triangles and assign the edge guards at chosen points during the triangulation. This is done in the following way: as the diagonals are generated keep a count on the number of currently unguarded triangles. Side-triangles are included in the count as soon as a cross-diagonal below them has been

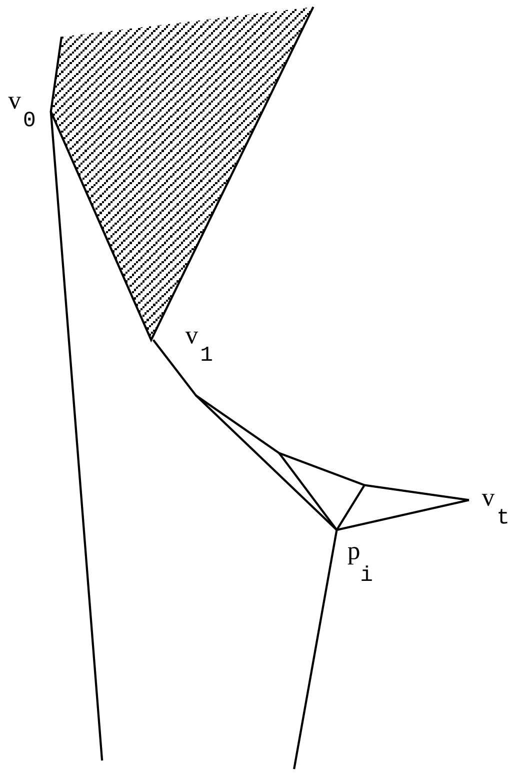

Figure 6: Examined vertex p_i is adjacent to top stack entry v_0 and angle $v_{t-1}v_tp_i$ is not reflex. Side-triangle $v_{t-1}v_tp_i$ is produced.

produced. Each time the newly generated diagonal is a cross-diagonal, check if the count is at least five, if so stop and assign a guard. Which particular edge we pick as guard will depend on the configuration of the triangles in the group. The number of possible configurations is limited and each such configuration and its associated edge guard is described below. Since the count is checked whenever a cross-diagonal is generated, at least one and at most five cross-triangles are among the unguarded triangles. We can thus divide the cases of possible configurations into five major categories, where all cases within a category have the same number of cross-triangles. Even if the number of cross-triangles are the same, the generated cross-diagonals may differ as will the position of any existing side-triangles. It may happen that when the latest cross-diagonal was generated a number of adjacent side-triangles were added to the configuration (see figure 7). Possibly we may decide not to include all of these

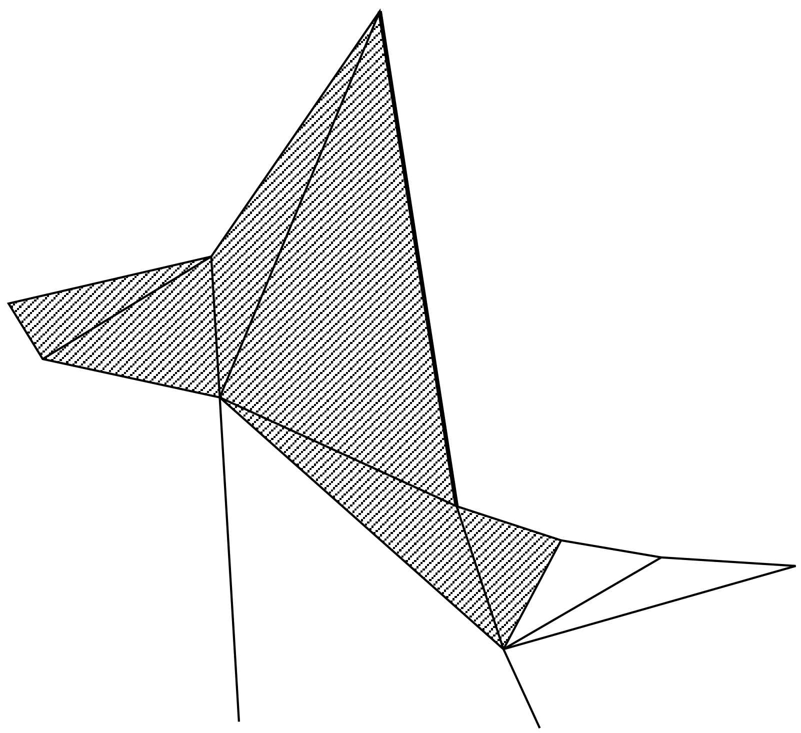

Figure 7: All side-triangles above the cross-diagonal are not necessarily included in the group.

in the group being formed, but instead leave some of them unguarded until the next stop. At that point these side-triangles would be included in the group formed then. When this is the case, we always make sure that the guard is chosen in such a way that we have the following invariant:

Invariant: Following the stop at a cross-diagonal any side-triangles left unguarded by the newly assigned guard lie between the two most recently generated cross-diagonals and on the same side as the bottom-most vertex.

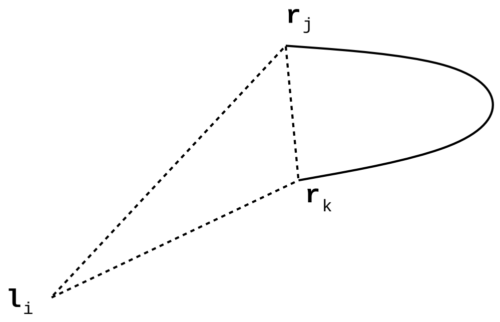

Figure 8: A guard on edge $l_{i-1}l_i$ sees all triangles.

We next describe in detail every possible configuration of triangles and how an edge guard is chosen in each case. The first major category of cases are the one where exactly one cross-triangle is present in each configuration.

3.1 Exactly one cross-triangle is present

Suppose that, at the creation of a cross-diagonal, the tally of unguarded triangles is at least five, but that exactly one of these triangles is a cross-triangle. All the other unguarded triangles are then side-triangles and there must be at least four of them present. Assume that the cross-triangle has one vertex l_i on the left chain and two vertices r_j and r_k on the right chain, i.e. the cross-diagonals are l_ir_j and l_ir_k. Vertex l_i can lie below, between or above the two vertices on the right chain. We study the three cases separately.

Case 3.1.1 *Cross-diagonals are l_ir_j and l_ir_k. Vertex l_i lies below r_j and r_k.*

If a vertex l_h on the left chain were to lie above l_i and below r_k then the cross-diagonals l_hr_j and l_hr_k would have been generated by the triangulation algorithm and cross-diagonal l_ir_j would not exist (see figure 8). Thus no vertex lies on the left side between r_k and l_i and no edge on the right side spans the y-extent of the area between l_{i-1} and l_i. From observation 1 we know that side-triangles are always spanned by a single edge on the opposite side so we see that no side-triangles lie on the left side between l_{i-1} and l_i. All side-triangles in this configuration (and we know there are at least four of them) must then lie on the right chain between r_j and r_k. Using observation 1 again, we know that these side-triangles are spanned by an edge on the left side, i.e. vertex l_{i-1} must lie above vertex r_j. Thus a guard assigned to edge $l_{i-1}l_i$ will see all triangles in the configuration.$\diamond$

Case 3.1.2 *Cross-diagonals are l_ir_j and l_ir_k. Vertex l_i lies above r_k but below r_j.*

Since vertex l_i lies between vertices r_j and r_k (see figure 9), no single edge on the left side spans the y-extent of the area between these two vertices. Thus, using observation 1, we see that no side-triangles lie on the right chain between r_j and r_k. All present side-triangles (at least four) must then lie on the left side immediately above l_i and be spanned by the edge

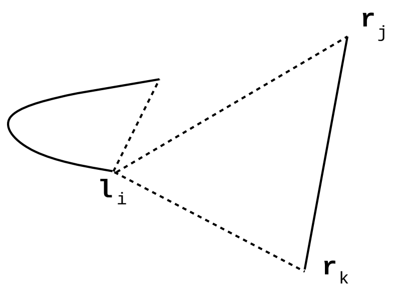

Figure 9: A guard on edge r_jr_k sees all triangles.

r_jr_k. These side-triangles were left unguarded when a stop was made at cross-diagonal l_ir_j and the edge guard for that group was assigned. We see that the conditions regarding the position of these left over side-triangles are as described in the invariant above. They are now included in the group to be created and a guard assigned to r_jr_k sees all the triangles in the configuration.◇

Case 3.1.3 *Cross-diagonals are l_ir_j and l_ir_k. Vertex l_i lies above r_j and r_k.*

Side-triangles are possible on the right chain between r_j and r_k and also above r_j spanned by the edge l_il_{i+1} (see figure 10). A guard assigned to l_il_{i+1} sees all the involved triangles.

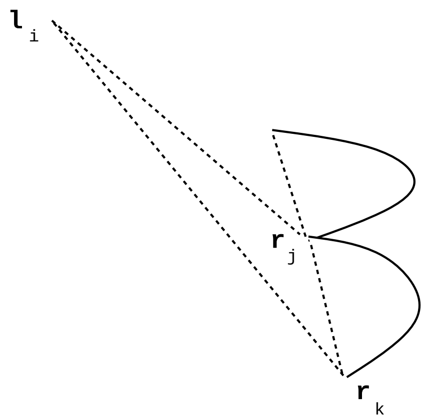

Figure 10: A guard on edge l_il_{i+1} sees all triangles.

No side-triangles above l_ir_k are left unguarded.◇

These cases exhaust the possibilities when the cross-triangle has one vertex on the left chain and two on the right chain. The cases for a cross-triangle with two vertices on the left chain and one vertex on the right chain are symmetric.

3.2 Exactly two cross-triangles are present.

Suppose that after the creation of a cross-diagonal, the tally of unguarded triangles is at least five, but the number of cross-triangles is exactly two. Two cross-triangles are formed by three cross-diagonals running between four endpoints. These endpoints may be distributed with one endpoint on one chain and three endpoints on the opposite chain, or with two endpoints on each chain. We first look at the situation when one endpoint l_i is on the left chain and the three endpoints on the right chain are in order r_j, r_k and r_l. The position of vertex l_i relative to the vertices on the right chain, gives rise to three cases.

Case 3.2.1 *Cross-diagonals are l_ir_j, l_ir_k and l_ir_l. Vertex l_i lies below r_k.*

As for case 3.1.1, no vertex l_h can lie on the left side below r_j since if that was the case cross-diagonal l_ir_j would not exist. Thus no edge on the right side spans the y-extent of the area between l_{i-1} and l_i and no side-triangles lie on the left side. All side-triangles must lie on the right side between r_j and r_k and if l_i lies below r_l possibly also between r_k and r_l (see figure 11). A guard assigned to $l_{i-1}l_i$ sees all triangles. No side-triangles are left

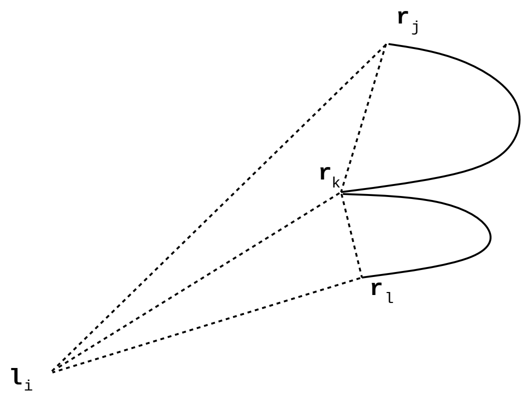

Figure 11: A guard on edge $l_{i-1}l_i$ sees all triangles.

unguarded.◇

Case 3.2.2 *Cross-diagonals are l_ir_j, l_ir_k and l_ir_l. Vertex l_i lies above r_k and r_l but below r_j.*

Side-triangles may be present on the right chain between r_k and r_l and there may also be unguarded side-triangles left from the previous guard assignment (see figure 12). These latter side-triangles lie on the left chain immediately above l_i. If two or more unguarded side-triangles lie above l_i, a guard assigned to r_jr_k sees these triangles, the two cross-triangles and at least one side-triangle on the right chain below r_k. Any additional side-triangles between r_k and r_l are left unguarded. If at most one side-triangle is present above l_i, the guard can instead be assigned to l_il_{i+1}. This guard sees all the triangles and no side-triangles are left unguarded.◇

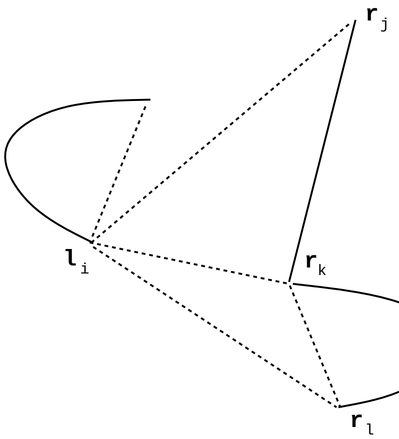

Figure 12: Case 3.2.2. Vertex l_i lies between r_j and r_k.

Case 3.2.3 *Cross-diagonals are l_ir_j, l_ir_k and l_ir_l. Vertex l_i lies above r_j, r_k and r_l.*

As for case 3.1.3, a guard assigned to l_il_{i+1} sees all the involved triangles.$\diamond$

The cases when there are three endpoints on the left chain and one on the right are symmetric. Next we consider the situation where two endpoints l_i and l_j lie on the left chain and two endpoints r_k and r_l lie on the right chain. We have three possible cases depending on whether vertex l_i lies below, between or above the two vertices on the right side. Assume that l_ir_k,l_ir_l and l_jr_l are the cross-diagonals generated by the triangulation. The cases when instead the cross-diagonals are l_ir_k, l_jr_k and l_jr_l are symmetric.

Case 3.2.4 *Cross-diagonals are l_ir_k, l_ir_l and l_jr_l. Vertex l_i lies below r_k and r_l.*

Side-triangles can lie on the right chain between r_k and r_l and on the left chain between l_i and l_j (see figure 13). If there are two or more side-triangles on the right chain, a guard on edge $l_{i-1}l_i$ will see these in addition to the two cross-triangles and at least one side-triangle on the left chain. Any additional side-triangles on the left chain are left unguarded. If at most one side-triangle is present on the right chain, a guard assigned to r_lr_{l+1} sees all the triangles that are present.$\diamond$

Case 3.2.5 *Cross-diagonals are l_ir_k, l_ir_l and l_jr_l. Vertex l_i lies between r_k and r_l.*

Vertex l_j may lie below r_l or above r_l (see figure 14). If l_j lies below r_l all unguarded side-triangles must be left from the previous guard assignment and lie on the left chain above l_i. If l_j lies above r_l, side triangles may also lie on the left chain between l_i and l_j. In either case, the edge r_kr_l spans the side-triangles present and a guard assigned to r_kr_l sees all the triangles involved.$\diamond$

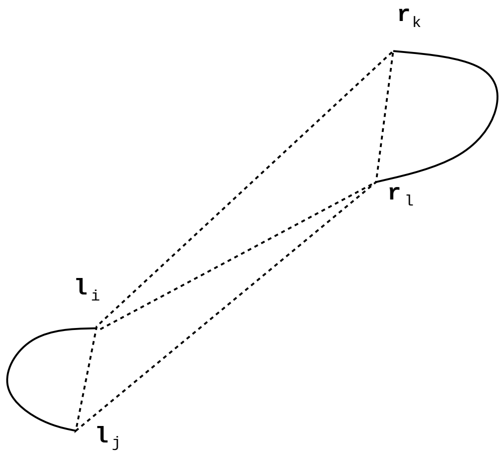

Figure 13: Case 3.2.4

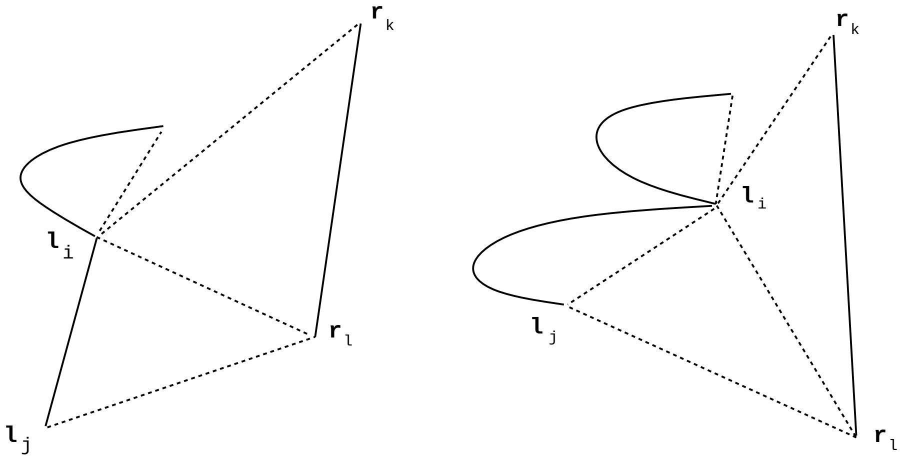

Figure 14: Case 3.2.5.

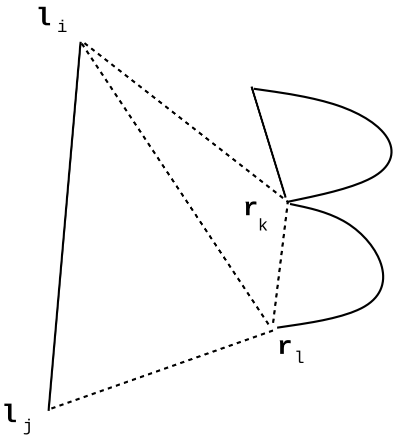

Figure 15: Case 3.2.6.

Case 3.2.6 *Cross-diagonals are l_ir_k, l_ir_l and l_jr_l. Vertex l_i lies above r_k and r_l.*

Vertex l_j must lie below r_l or l_ir_l would not be a cross-diagonal (see figure 15). All side-triangles must lie on the right chain either immediately above r_k or between r_k and r_l. A guard assigned to edge l_il_j sees all triangles.◇

The six cases above and their symmetric counterparts exhaust the possible configurations when exactly two cross-triangles are present.

3.3 Exactly three cross-triangles are present

The endpoints of the four cross-diagonals are distributed with one endpoint on one chain and four endpoints on the opposite chain, or with two endpoints on one chain and three endpoints on the opposite chain. We consider first the case where one endpoint l_i lies on the left chain. The endpoints on the right chain are in order r_j, r_k, r_l and r_m. There are four possible positions for vertex l_i: below r_l, between r_k and r_l, between r_j and r_k or above r_j. We study the four cases separately. Symmetric cases exist when four endpoints lie on the left chain and one endpoint lies on the right chain.

Case 3.3.1 *Cross-diagonals are l_ir_j, l_ir_k, l_ir_l and l_ir_m. Vertex l_i lies below r_l.*

This case is similar to case 3.2.1. A guard assigned to edge $l_{i-1}l_i$ sees all the triangles.◇

Case 3.3.2 *Cross-diagonals are l_ir_j, l_ir_k, l_ir_l and l_ir_m. Vertex l_i lies between r_k and r_l.*

Side-triangles may lie between r_j and r_k and/or between r_l and r_m (see figure 16). Between

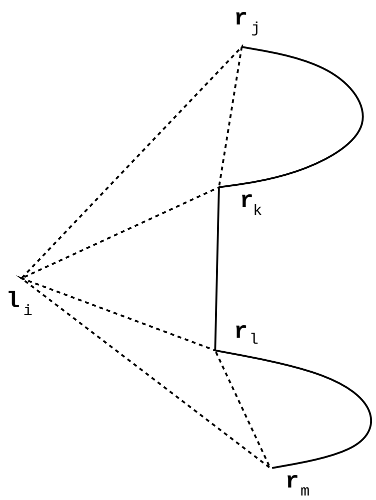

Figure 16: Case 3.3.2

r_j and r_k there can be at most two side-triangles or the count would have reached five when cross-diagonal $l_i r_l$ was generated. If two side-triangles in fact lie between r_j and r_k, a guard assigned to edge $l_{i-1}l_i$ sees these two triangles in addition to the three cross-triangles. Only side-triangles between r_l and r_m are left unguarded. If one side-triangle lies between r_j and r_k, a guard assigned to $r_k r_l$ sees this triangle, the three cross-triangles and at least one side-triangle between r_l and r_m. Any additional side-triangles between r_l and r_m are left unguarded. If no side-triangle lies between r_j and r_k, a guard assigned to $l_i l_{i+1}$ sees all triangles.◇

Case 3.3.3 *Cross-diagonals are $l_i r_j$, $l_i r_k$, $l_i r_l$ and $l_i r_m$. Vertex l_i lies between r_j and r_k.*

Side-triangles may be present on the right chain between r_k and r_m and also on the left chain immediately above l_i (see figure 17). If there are two side-triangles on the left chain,

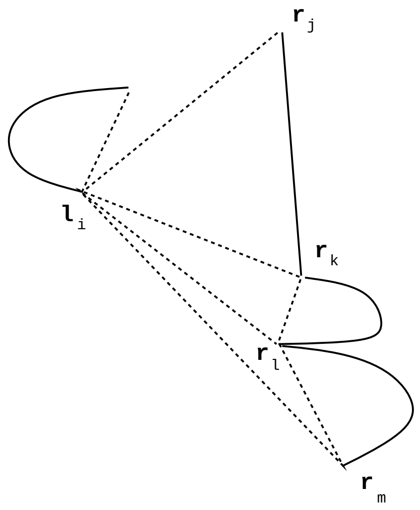

Figure 17: Case 3.3.3

no side-triangle can be present on the right chain between r_k and r_l, or the count would have reached five unguarded triangles when cross-diagonal $l_i r_l$ was generated. Therefore in this case a guard assigned to $l_{i-1}l_i$ sees the two side-triangles on the left chain and the three cross-triangles. Any side-triangles left unguarded for the next guard assignment lie on the right chain between r_l and r_m. If there are less than two side-triangles on the left chain, a guard assigned to $l_i l_{i+1}$ sees all triangles.◇

Case 3.3.4 *Cross-diagonals are $l_i r_j$, $l_i r_k$, $l_i r_l$ and $l_i r_m$. Vertex l_i lies above r_j.*

As for case 3.2.3, a guard assigned to $l_i l_{i+1}$ sees all the triangles in the group.◇

Next we consider the cases when the four cross-diagonals are formed with two endpoints l_i and l_j on the left chain and three endpoints r_k, r_l and r_m on the right chain. Symmetric cases exist when there are three endpoints on the left chain and two endpoints on the right chain. Assume first that l_ir_k, l_ir_l, l_ir_m and l_jr_m are the cross-diagonals generated by the algorithm. The possible positions of vertex l_i give rise to four separate cases.

Case 3.3.5 *Cross-diagonals are l_ir_k, l_ir_l, l_ir_m and l_jr_m. Vertex l_i lies below r_m.*

As for case 3.1.1, no unguarded side-triangles are left from the previous group. At most two side-triangles lie between r_k and r_m, since otherwise the count would have reached five when cross-diagonal l_ir_m was generated (see figure 18). Any number of side-triangles can be

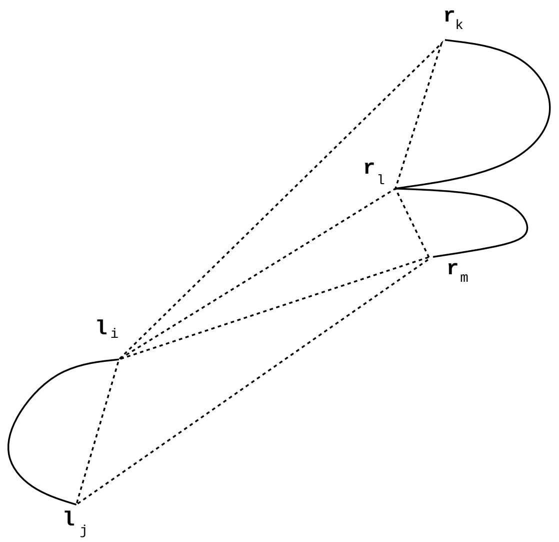

Figure 18: Case 3.3.5

present between l_i and l_j. A guard on edge $l_{i-1}l_i$ sees any side-triangle present between r_k and r_m, the three cross-triangles and one side-triangle between l_i and l_j if such a triangle exists. If no side-triangle is present above r_m then a guard on edge l_il_{i+1} sees the cross-triangles and at least two side-triangles that must be present between l_i and l_j. Any additional side-triangles between l_i and l_j are left unguarded for the next guard assignment.◇

Case 3.3.6 *Cross-diagonals are l_ir_k, l_ir_l, l_ir_m and l_jr_m. Vertex l_i lies between r_l and r_m.*

No unguarded side-triangle is left from the previous guard assignment and no side-triangles can lie on the right chain between r_l and r_m. At most two side-triangles can lie between r_k and r_l (see figure 19). If l_j lies above r_m then side-triangles are also possible between l_i and l_j. If there are two side-triangles on the right chain and at most one side-triangle between l_i and l_j, a guard on edge $l_{i-1}l_i$ sees all triangles. If instead at least two side-triangles are present between l_i and l_j, a guard on edge $l_{i-1}l_i$ sees the two side-triangles on the right chain, the two topmost cross-triangles and one side-triangle between l_i and l_j. The guard assignment continues below l_ir_m with one less side-triangle counted between l_i and l_j. If no more than one side-triangle is present above r_l, a guard on edge r_lr_m sees all triangles.◇

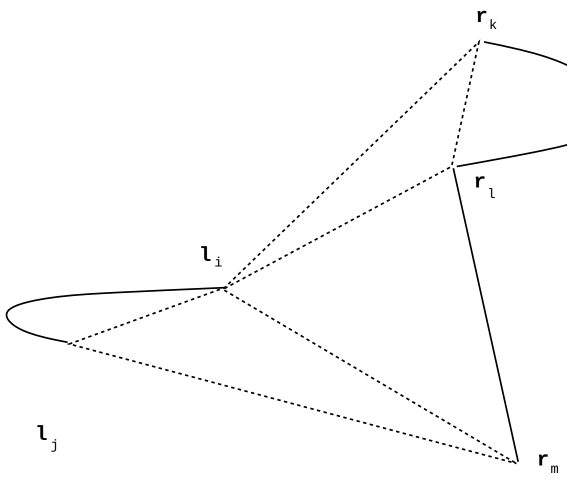

Figure 19: Case 3.3.6

Case 3.3.7 *Cross-diagonals are l_ir_k, l_ir_l, l_ir_m and l_jr_m. Vertex l_i lies between r_k and r_l.*

Vertex l_j must lie below r_m or l_ir_m could not be a cross-diagonal (see figure 20). Exactly

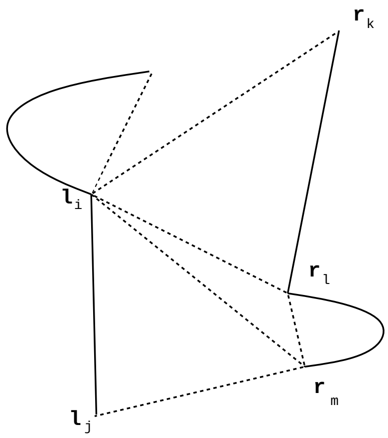

Figure 20: Case 3.3.7

two unguarded side-triangles are present. If both of these are left from the previous guard assignment, a guard on edge $l_{i-1}l_i$ sees all the triangles. If no more than one side-triangle is left above l_i, another side-triangle must be lie between r_l and r_m. A guard on edge l_il_j sees all triangles.◇

Case 3.3.8 *Cross-diagonals are l_ir_k, l_ir_l, l_ir_m and l_jr_m. Vertex l_i lies above r_k.*

As for case 3.2.6 a guard on edge l_il_j sees all triangles.◇

Assume next that the cross-diagonals are instead l_ir_k, l_ir_l, l_jr_l and l_jr_m. Since l_jr_l is a cross-diagonal, l_i cannot lie below r_m, but instead must lie between r_l and r_m, between r_k and r_l or above r_k. We study the three cases separately.

Case 3.3.9 *Cross-diagonals are l_ir_k, l_ir_l, l_jr_l and l_jr_m. Vertex l_i lies between r_l and r_m.*

As for case 3.1.1. there are no side-triangles left unguarded from the previous guard assignment. Any side-triangles must lie between r_k and r_l and/or between l_i and l_j (see figure 21). If there are two side-triangles above r_l, there can be no side-triangle present

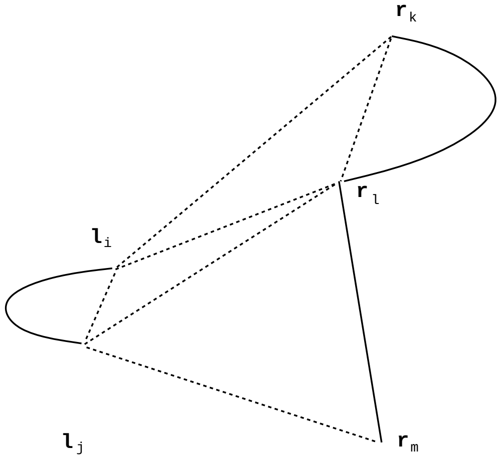

Figure 21: Case 3.3.9

between l_i and l_j. A guard on edge $r_{l-1}r_l$ sees all triangles. If no more than one side-triangle lies above r_l, a guard on edge r_lr_m sees all triangles.◇

Case 3.3.10 *Cross-diagonals are l_ir_k, l_ir_l, l_jr_l and l_jr_m. Vertex l_i lies between r_k and r_l.*

There may be unguarded side-triangles on the left chain immediately above l_i. If vertex l_j lies above r_l, side-triangles are also possible between l_i and l_j and between r_l and r_m (see figure 22). If such is the case, a guard on edge r_kr_l sees all side-triangles present above l_jr_l, the cross-triangles and one side-triangle below r_l. If no side-triangles lie above l_jr_l, a guard on edge r_lr_{l+1} sees all cross-triangles and at least two side-triangles below r_l. Any additional side-triangles below r_l are left unguarded for the next guard assignment. If instead the vertex l_j lies between r_l and r_m, all present side-triangles must lie above l_i on the left chain. A guard on edge r_kr_l sees all triangles. Finally if l_j lies below r_m, side-triangles are also possible between r_l and r_m on the right chain (see figure 23). If at most one side-triangle lies on the left chain, a guard on edge l_il_j sees all triangles. If two side-triangles lie on the left chain and no more than one side-triangle lies on the right chain, a guard on edge r_kr_l sees all triangles. If both the left and the right chains have two side-triangles, a guard on edge r_kr_l sees the two side-triangles on the left chain, the two topmost cross-triangles and one side-triangle on the right chain. The guard assignment then continues below l_jr_l with one less side-triangle counted between r_l and r_m. ◇

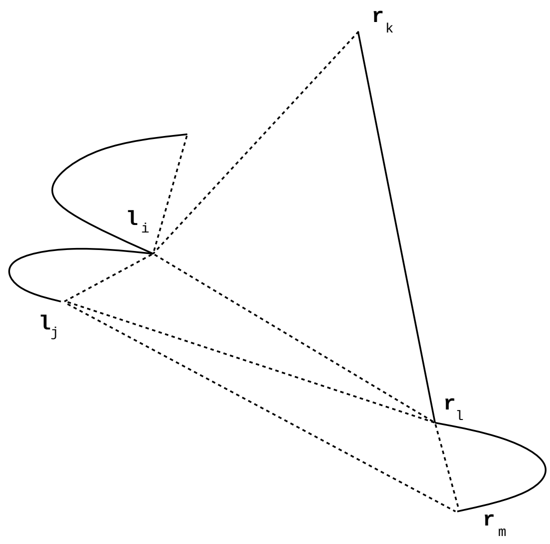

Figure 22: Case 3.3.10. Vertex l_j lies above r_l.

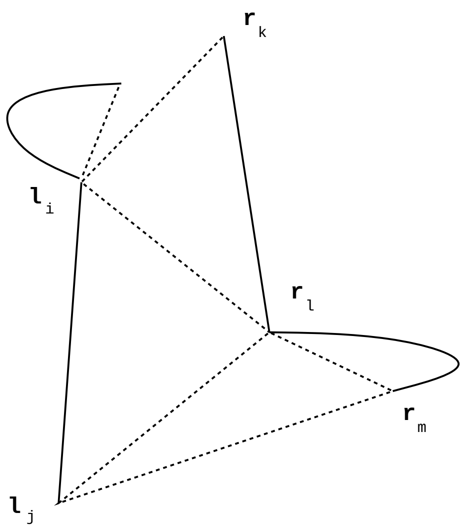

Figure 23: Case 3.3.10. vertex l_j lies below r_m

Case 3.3.11 *Cross-diagonals are $l_i r_k$, $l_i r_l$, $l_j r_l$ and $l_j r_m$. Vertex l_i lies above r_k.*

Side-triangles are present on the right side. They may lie above r_k, between r_k and r_l and if l_j is below r_m they may also lie between r_l and r_m (see figure 24). A guard on edge

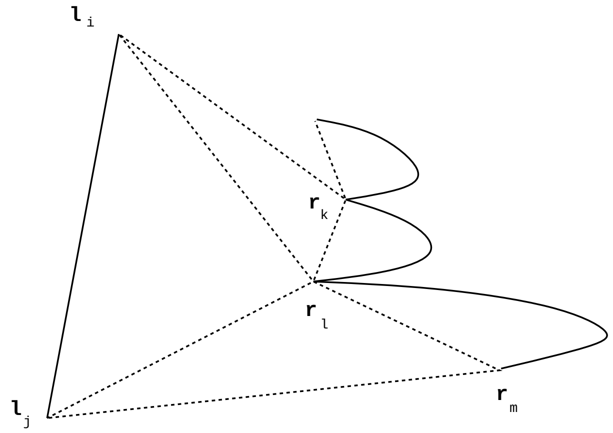

Figure 24: Case 3.3.11

$l_i l_j$ sees all triangles. ◇

Assume finally that the cross-diagonals are $l_i r_k$, $l_j r_k$ $l_j r_l$ and $l_j r_m$. Vertex l_i must lie above r_l, since $l_j r_k$ is a cross-diagonal. Thus there are two possible cases, either l_i lies between r_k and r_l or l_i lies above r_k.

Case 3.3.12 *Cross-diagonals are $l_i r_k$, $l_j r_k$, $l_j r_l$ and $l_j r_m$. Vertex l_i lies between r_k and r_l.*

Since $l_j r_k$ is a cross-diagonal, vertex r_l must lie below l_j (see figure 25). Side-triangles may have been left unguarded above l_i by the previous guard assignment. Side-triangles may also lie between l_i and l_j and between r_l and r_m. No side-triangles can lie between r_k and r_l. A guard on edge $r_k r_l$ sees any unguarded side-triangles above l_i, any side-triangles between l_i and l_j, the three cross-triangles and one side-triangle between r_l and r_m, if such a triangle is present. Any additional side-triangles between r_l and r_m are left unguarded for the next group. If no side-triangle lies above $l_j r_l$, a guard on edge $l_j l_{j+1}$ sees all triangles. ◇

Case 3.3.13 *Cross-diagonals are $l_i r_k$, $l_j r_k$, $l_j r_l$ and $l_j r_m$. Vertex l_i lies above r_k.*

If l_j lies below r_l, a guard on edge $l_i l_j$ sees all triangles (see figure 26).

If l_j lies between r_k and r_l no side-triangles can lie on the right chain between r_k and r_l (see figure 27). Unguarded side-triangles may be present immediately above r_k. If there are two side-triangles above r_k, a guard on edge $l_i l_j$ sees these two side-triangles and the three cross-triangles. Any side-triangles between r_l and r_m are left for the next guard assignment.

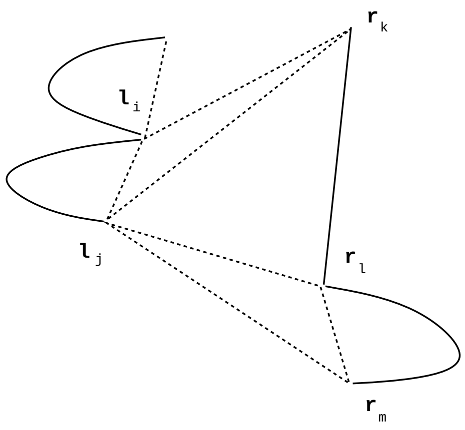

Figure 25: Case 3.3.12

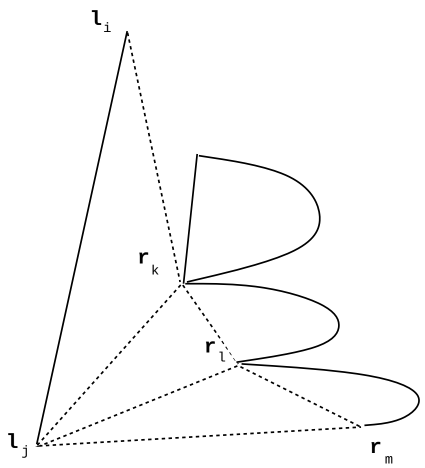

Figure 26: Case 3.3.13, Vertex l_j lies below r_l.

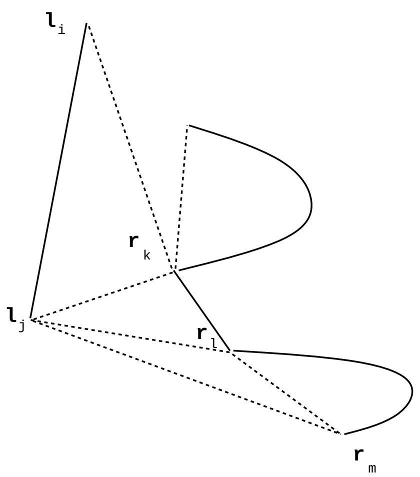

Figure 27: Case 3.3.13, Vertex l_j lies between r_k and r_l.

If there is one side-triangle above r_k, a guard on edge r_kr_l sees this triangle the three cross-triangles and one side-triangle, which must be present between r_l and r_m. Any additional side-triangles are left for the next guard assignment. If there are no side-triangles above r_k, a guard on edge l_jl_{j+1} sees all triangles.

If l_j lies above r_k, side-triangles can lie both between l_i and l_j and between r_k and r_m (see figure 28). However, at most two side-triangles lie above l_jr_l. If both of these are between

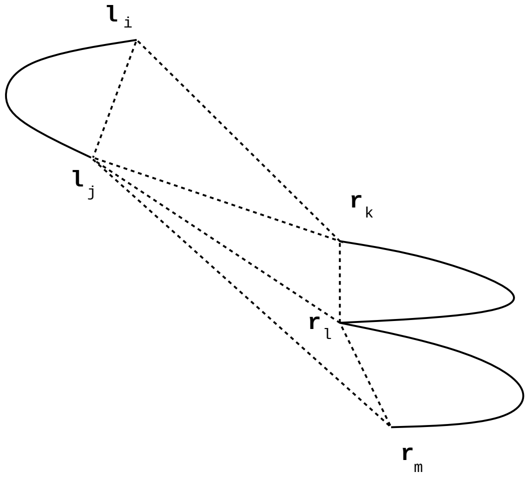

Figure 28: Case 3.3.13, Vertex l_j lies above r_l.

l_i and l_j, a guard on edge $l_{j-1}l_j$ sees these side-triangles and the cross-triangles. Any side-triangle below r_l is left for the next guard assignment. If no more than one side-triangle lies between l_i and l_j, a guard on edge l_jl_{j+1} sees all triangles. ◇

The above thirteen cases and their symmetric counterparts describe all possible configurations with exactly three cross-triangles.

3.4 Exactly four cross-triangles are present.

The endpoints of the five cross-diagonals may be distributed with one, two or three endpoints on one chain and the remaining five, four or three endpoints on the opposite chain.

Again we consider first the case where endpoint l_i is the only vertex on the left chain and the endpoints on the right chain are in order r_j, r_k, r_l, r_m and r_n. We study two cases. In the first case vertex l_i lies below r_m, in the second case l_i lies above r_m. Symmetric cases exist when five endpoints lie on the left chain and one endpoint lies on the right chain. At most one side-triangle can be present above the cross-diagonal l_ir_m or the last cross-triangle would not have been added.

Case 3.4.1 *Cross-diagonals are l_ir_j, l_ir_k, l_ir_l, l_ir_m and l_ir_n. Vertex l_i lies below r_m.*

As for case 3.2.1 a guard on edge $l_{i-1}l_i$ sees all triangles. ◇

Case 3.4.2 *Cross-diagonals are l_ir_j, l_ir_k, l_ir_l, l_ir_m and l_ir_n. Vertex l_i lies above r_m.*

If a side-triangle lies on the left side immediately above l_i, possible if l_i is above r_k, a guard on edge l_il_{i+1} sees all triangles (see figure 29). If a side-triangle lies on the right side

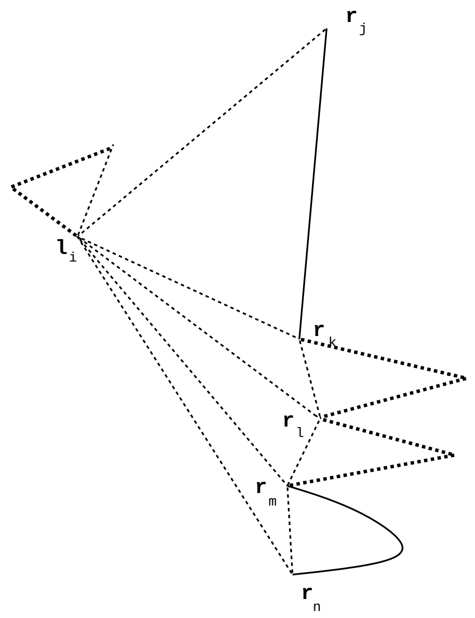

Figure 29: Case 3.4.2, A side-triangle lies on the left side above l_i or on the right side below l_i.

below l_i, a guard on edge l_il_{i+1} sees all triangles. If a side-triangle is present on the right side above l_i, a guard on edge $l_{i-1}l_i$ sees this triangle and all cross-triangles (see figure 30). Side-triangles below r_m are left for the next group. $\diamond$

Next consider the case when two endpoints l_i and l_j lie on the left chain and the four endpoints on the right chain are in order r_k, r_l, r_m and r_n. Symmetric cases exist when four endpoints lie on the left chain and two endpoints lie on the right chain. Let all cross-diagonals except the last emanate from l_i. There are four cases depending on the position of vertex l_i. At most one side-triangle can be present above l_ir_n.

Case 3.4.3 *Cross-diagonals are l_ir_k, l_ir_l, l_ir_m, l_ir_n and l_jr_n. Vertex l_i lies below r_m.*

No unguarded side-triangles can lie above l_i on the left side (see figure 31). If no more than one side-triangle lies between l_i and l_j, a guard on edge $l_{i-1}l_i$ sees all triangles. If more than one side-triangle lies between l_i and l_j and at least one side-triangle is present on the right chain, a guard on edge $l_{i-1}l_i$ sees the side-triangle on the right chain, the three topmost cross-triangles and one side-triangle on the left chain below l_i. The guard assignment continues below l_ir_n with one less side-triangle counted between l_i and l_j. Finally if no side-triangles

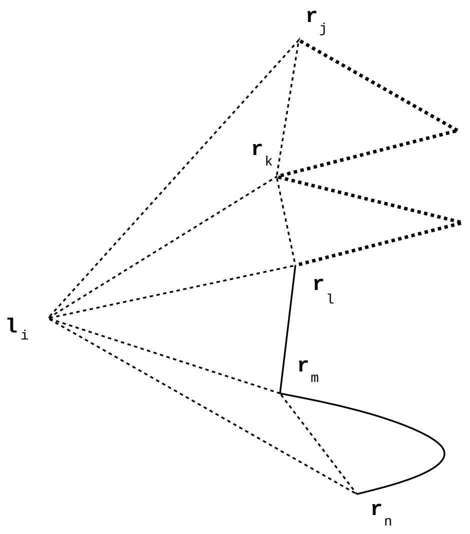

Figure 30: Case 3.4.2, A side-triangle lies on the right side above l_i.

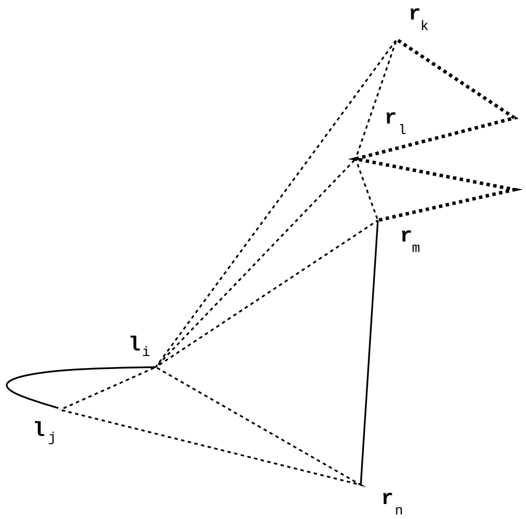

Figure 31: Case 3.4.3

lie on the right chain, a guard on edge $l_i l_{i+1}$ sees the three topmost cross-triangles and two side-triangles between l_i and l_j. The guard assignment then continues below $l_i r_n$ with two less side-triangles counted between l_i and l_j. ◇

Case 3.4.4 *Cross-diagonals are $l_i r_k$, $l_i r_l$, $l_i r_m$, $l_i r_n$ and $l_j r_n$. Vertex l_i lies between r_l and r_m.*

No unguarded side-triangles can lie on the left chain above l_i. Vertex l_j must lie below r_n, since $l_i r_n$ is a cross-diagonal (see figure 32). Exactly one side-triangle is present. If this

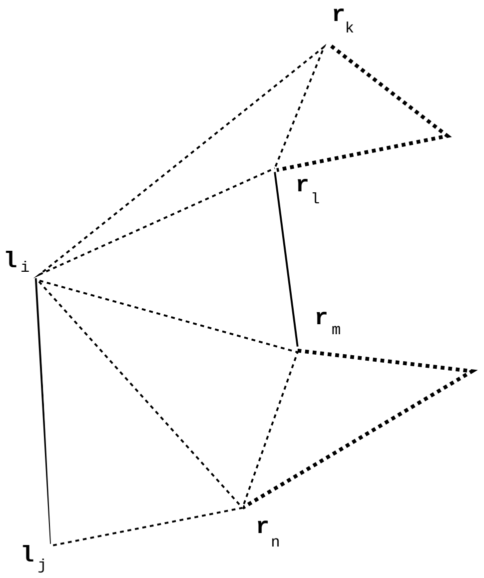

Figure 32: Case 3.4.4

lies between r_k and r_l then a guard on edge $l_{i-1} l_i$ sees all triangles. If the side-triangle lies between r_m and r_n then a guard on edge $l_i l_j$ sees all triangles. ◇

Case 3.4.5 *Cross-diagonals are $l_i r_k$, $l_i r_l$, $l_i r_m$, $l_i r_n$ and $l_j r_n$. Vertex l_i lies between r_k and r_l.*

One unguarded side-triangle may be present on the left chain above l_i or on the right chain between r_l and r_n (see figure 33). Vertex l_j must lie below r_n, since $l_i r_n$ is a cross-diagonal. A guard on edge $l_i l_j$ sees all triangles. ◇

Case 3.4.6 *Cross-diagonals are $l_i r_k$, $l_i r_l$, $l_i r_m$, $l_i r_n$ and $l_j r_n$. Vertex l_i lies above r_k.*

As for case 3.2.6, a guard on edge $l_i l_j$ sees all triangles. ◇

Next let three cross-diagonals emanate from l_i and two from l_j, i.e. the cross-diagonals are $l_i r_k$, $l_i r_l$, $l_i r_m$, $l_j r_m$ and $l_j r_n$. Again the different positions of vertex l_i give rise to four possible cases.

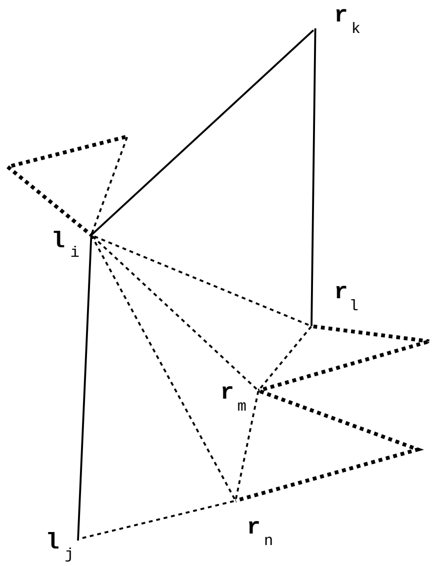

Figure 33: Case 3.4.5

Case 3.4.7 *Cross-diagonals are l_ir_k, l_ir_l, l_ir_m, l_jr_m and l_jr_n. Vertex l_i lies below r_m.*

No unguarded triangles can lie above l_i. Vertex l_j must lie above r_n since l_jr_m is a cross-diagonal. The single side-triangle present may lie between r_k and r_l, between r_l and r_m or between l_i and l_j.

If the side-triangle lies between r_k and r_l (see figure 34), a guard on edge r_lr_m sees all triangles.

If the side-triangle lies between r_l and r_m (see figure 35), no single edge is guaranteed to see all triangles in every situation. Edge $l_{i-1}l_i$ certainly sees the side-triangle and the three topmost cross-triangles. However part of the fourth cross-triangle $l_jr_mr_n$ may be hidden from view. Similarly edge l_il_j sees all cross-triangles, but not necessarily the side-triangle. If l_i, l_j, r_n forms a left turn, edge $l_{i-1}l_i$ sees all triangles. If l_i, l_j, r_n forms a right turn but l_i, r_m, r_{l+1} forms a left turn, a guard on edge l_il_j sees all triangles. If both l_i, l_j, r_n and l_i, r_m, r_{l+1} are right turns, but r_n, r_m, r_l forms a left turn, a guard on edge r_lr_{l+1} sees all triangles. Finally if all the above are right turns, a guard on edge $l_{i-1}l_i$ sees all triangles. This follows from the fact that r_l, r_k, l_{i-1} must also form a right turn or $l_{i-1}r_l$ would be a cross-diagonal instead of l_ir_k.

If the side-triangle lies between l_i and l_j (see figure 36), there is again no single edge guaranteed to see all triangles in every situation. If l_i, l_j, r_n forms a left turn, a guard on edge $l_{i-1}l_i$ sees all triangles. If l_i, l_j, r_n forms a right turn but r_m, l_i, l_{i+1} forms a left turn, a guard on edge r_lr_m sees all triangles. If both l_i, l_j, r_n and r_m, l_i, l_{i+1} are right turns, but r_k, l_i, l_j is a left turn, a guard on edge $l_{i+1}l_j$ sees all triangles. Finally if all the above are right turns, a guard on edge edge r_mr_n sees all triangles. $\diamond$

Case 3.4.8 *Cross-diagonals are l_ir_k, l_ir_l, l_ir_m, l_jr_m and l_jr_n. Vertex l_i lies between r_l and*

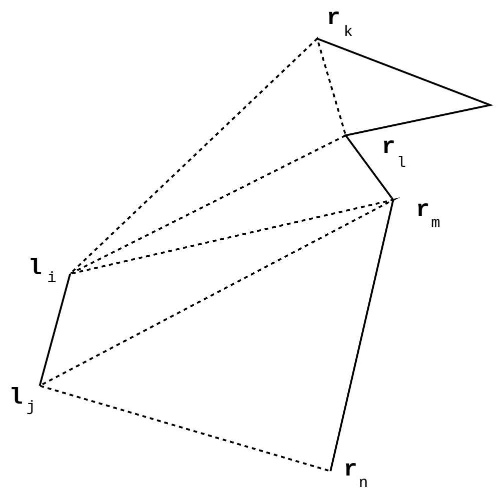

Figure 34: Case 3.4.7. A side-triangle lies between r_k and r_l.

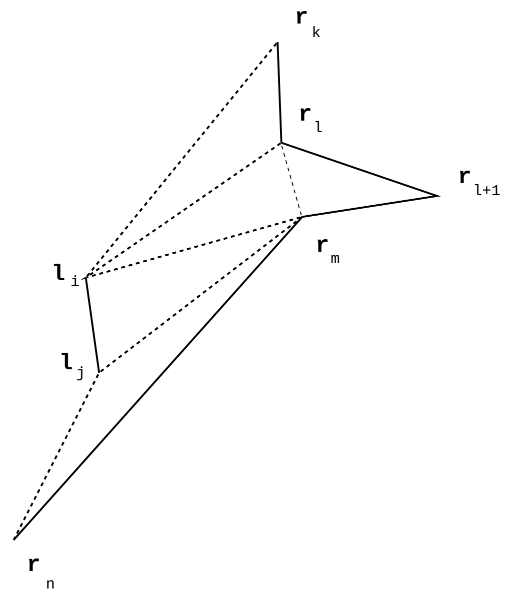

Figure 35: Case 3.4.7. A side-triangle lies between r_l and r_m.

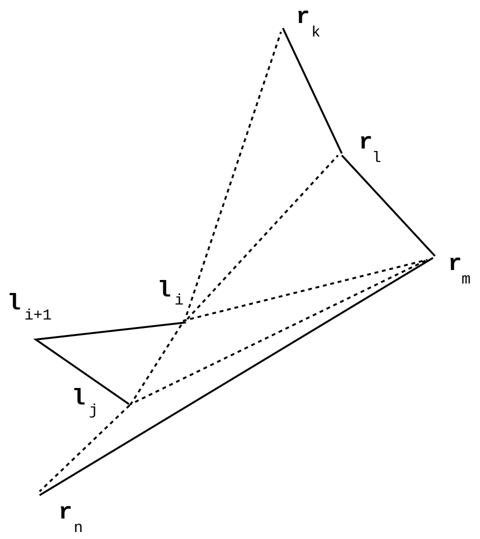

Figure 36: Case 3.4.7. A side-triangle lies between l_i and l_j.

r_m.

One side-triangle may lie between r_k and r_l or between l_i and l_j (see figure 37). Any number of side-triangles may lie between r_m and r_n. If vertex l_j lies above r_n, a guard on edge $r_l r_m$ sees the single side-triangle, all cross-triangles and one side-triangle between r_m and r_n if present. Any additional side-triangles between r_m and r_n are left for the next guard assignment. If vertex l_j lies below r_n and no side-triangle is present between r_k and r_l then a guard on edge $l_i l_j$ sees all triangles. If vertex l_j lies below r_n and a side-triangle is present between r_k and r_l then a guard on edge $r_l r_m$ sees this side-triangle, the cross-triangles and one side-triangle between r_m and r_n. If more than one side-triangle lies between r_m and r_n then the guard assignment continues below $l_j r_m$ with one less side-triangle counted between r_m and r_n. $\diamond$

Case 3.4.9 *Cross-diagonals are $l_i r_k$, $l_i r_l$, $l_i r_m$, $l_j r_m$ and $l_j r_n$. Vertex l_i lies between r_k and r_l.*

Vertex l_j must lie below r_m, since $l_i r_m$ is a diagonal. One side-triangle may lie above l_i or between r_l and r_m. If l_j lies below r_n, any number of side-triangles may lie between r_m and r_n. A guard on edge $l_i l_j$ sees all triangles. $\diamond$

Case 3.4.10 *Cross-diagonals are $l_i r_k$, $l_i r_l$, $l_i r_m$, $l_j r_m$ and $l_j r_n$. Vertex l_i lies above r_k.*

One side-triangle may lie anywhere on the right side (see figure 38). Additional side-triangles can lie between r_m and r_n. A guard on edge $l_i l_j$ sees all triangles. $\diamond$

Next let two cross-diagonals emanate from l_i and three from l_j. The cross-diagonals are $l_i r_k$, $l_i r_l$, $l_j r_l$, $l_j r_m$ and $l_j r_n$. At most one side-triangle is present above $l_j r_m$. Vertex l_i cannot

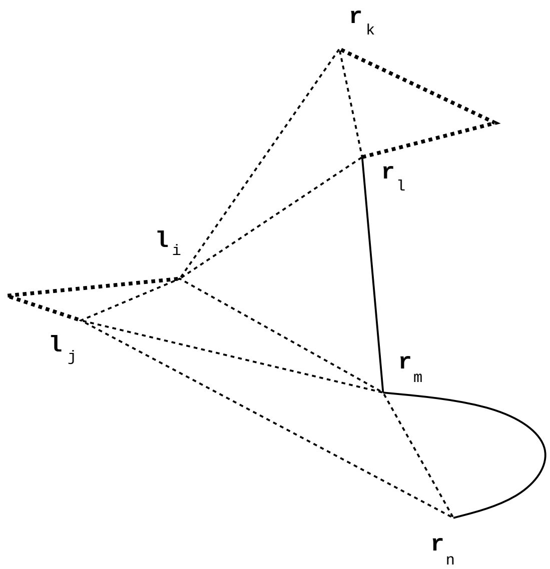

Figure 37: Case 3.4.8

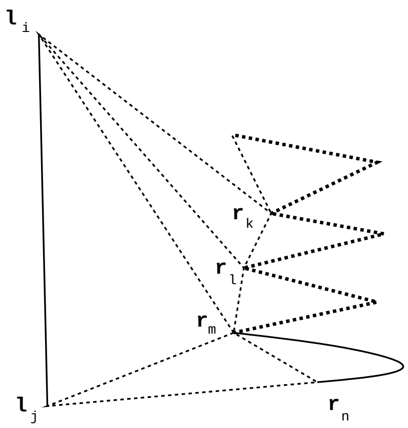

Figure 38: Case 3.4.10.

lie below r_m since $l_j r_l$ is a cross-diagonal and there are three possible cases of positions for l_i.

Case 3.4.11 *Cross-diagonals are $l_i r_k$, $l_i r_l$, $l_j r_l$, $l_j r_m$ and $l_j r_n$. Vertex l_i lies between r_l and r_m.*

Any number of side-triangles can lie between r_m and r_n (see figure 39). One side-triangle

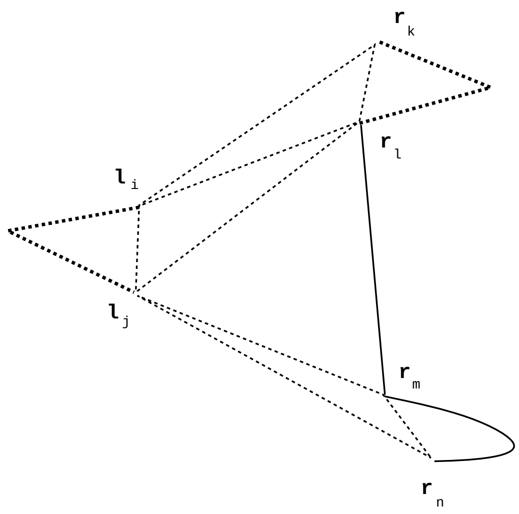

Figure 39: Case 3.4.11.

may be present between r_k and r_l or between l_i and l_j. A guard on edge $r_l r_m$ sees this triangle and all cross-triangles. It also sees one side-triangle below r_m. Any additional side-triangles below r_m are left for the next guard assignment. $\diamond$

Case 3.4.12 *Cross-diagonals are $l_i r_k$, $l_i r_l$, $l_j r_l$, $l_j r_m$ and $l_j r_n$. Vertex l_i lies between r_k and r_l.*

At most one side-triangle can lie above $l_j r_m$. If it lies above l_i (see figure 40), a guard on edge $l_i l_j$ sees the side-triangle and all cross-triangles.

The single side-triangle may instead lie between l_i and l_j. This is possible if l_j lies above r_l (see figure 41). We have a case similar to 3.4.7, where no single edge is guaranteed to see all triangles in every situation. If r_l, r_m, r_n forms a right turn, then a guard on edge $r_k r_l$ sees the side-triangle and all cross-triangles. If r_l, r_m, r_n forms a left turn, but r_l, l_j, l_{i+1} forms a right turn, a guard on edge $r_l r_m$ sees the side-triangle and all cross-triangles. If r_l, r_m, r_n and r_l, l_j, l_{i+1} are both left turns, but r_n, l_j, l_i is a right turn, a guard on edge $l_i l_{i+1}$ sees the side-triangle and all cross-triangles. Finally if all the above are left turns, a guard on edge $r_k r_l$ sees the side-triangle and all cross-triangles, since l_j, l_i, r_k must also be a left turn. Side-triangles between r_m and r_n are left for the next guard assignment.

A side-triangle may also lie between r_l and r_m. In this case the position of l_j is important. With l_j below r_m, a guard on edge $l_i l_j$ sees the side-triangles and all cross-triangles, but if

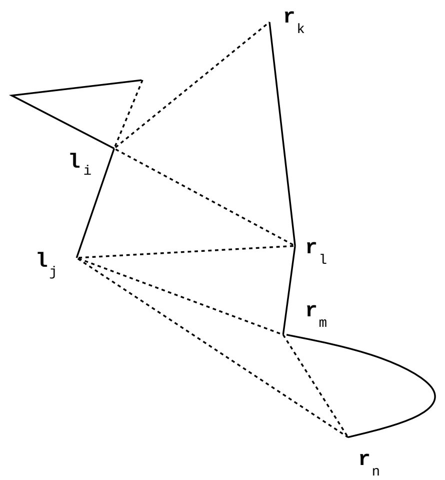

Figure 40: Case 3.4.12. A side-triangle lies above l_i.

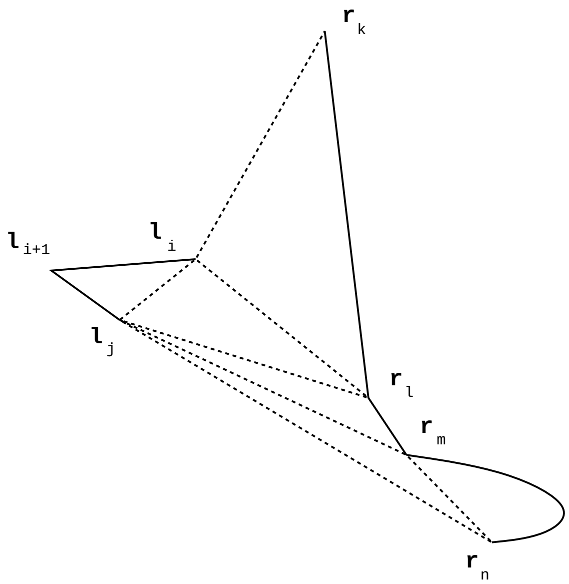

Figure 41: Case 3.4.12. A side-triangle lies between l_i and l_j.

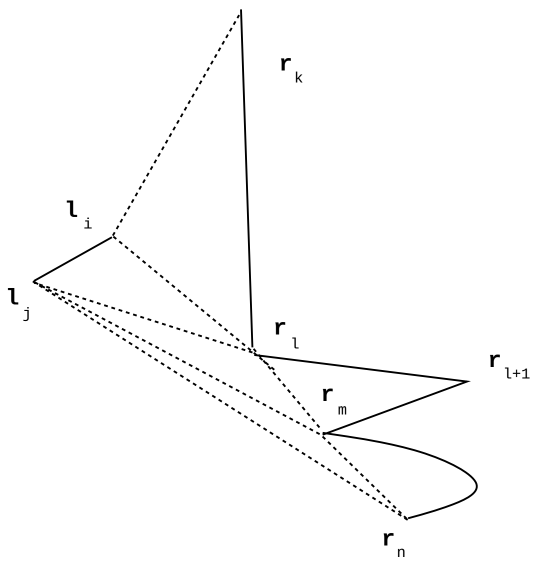

Figure 42: Case 3.4.12. A side-triangle lies between r_l and r_m.

l_j lies above r_l (see figure 42), again no single edge is guaranteed to see all triangles in every situation. If r_l, r_m, r_n forms a right turn, a guard on edge $r_k r_l$ sees the side-triangle and all cross-triangles. If r_l, r_m, r_n forms a left turn, but l_j, r_l, r_{l+1} forms a right turn, a guard on edge $l_i l_j$ sees the side-triangle and all cross-triangles. If r_l, r_m, r_n and l_j, r_l, r_{l+1} are both left turns, but r_k, r_l, r_m is a right turn, a guard on edge $r_{l+1} r_m$ sees the side-triangle and all cross-triangles. Finally it is possible that all the above are left turns. In this case no edge above $l_j r_n$ sees more than four of the triangles in the group to which we wish to assign a guard. We must use an edge from below $l_j r_n$. Chosen will be either an edge from which all the unguarded triangles above $l_j r_n$ can be seen or an edge from which the bottom cross-triangle l_j, r_m, r_n is seen and which in addition can be used as guard for the next group of triangles. In the latter case this guard together with a guard on edge $r_k r_l$ together cover all triangles in the two groups. Since the choice will depend on the configuration of triangles below $l_j r_n$, the discussion of how the specific edge is chosen is postponed till all cases of possible configurations have been examined.

Finally if all side-triangles lie between r_m and r_n, a guard on $r_l r_m$ sees one of these and all cross-triangles. Any additional side-triangles are left for the next guard assignment. $\diamond$

Case 3.4.13 *Cross-diagonals are $l_i r_k$, $l_i r_l$, $l_j r_l$, $l_j r_m$ and $l_j r_n$. Vertex l_i lies above r_k.*

Vertex l_j must lie below r_l, since $l_i r_l$ is a cross-diagonal. If l_j lies between r_l and r_m and there is a side-triangle above r_m (see figure 43), a guard on edge $l_i l_j$ sees this and all cross-triangles. If l_j lies between r_l and r_m and all side-triangles lie between r_m and r_n, a guard on edge $r_l r_m$ sees one of these and all cross-triangles. Any additional side-triangles between r_m and r_n are left for the next guard assignment. If l_j lies below r_m, a guard on edge $l_i l_j$ sees all triangles. $\diamond$

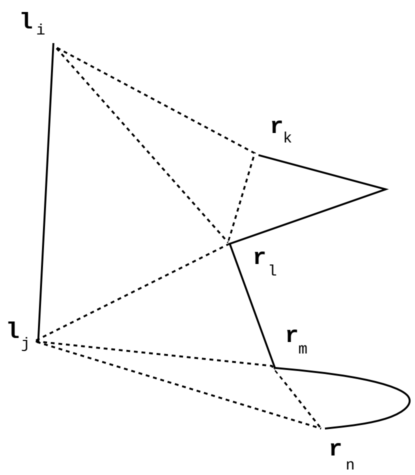

Figure 43: Case 3.4.13.

Finally let all cross-diagonals except the first emanate from l_j. They are l_ir_k, l_jr_k, l_jr_l, l_jr_m and l_jr_n. Vertex l_i cannot lie below r_l, since l_jr_k is a cross-diagonal. Two cases are possible, either l_i lies between r_k and r_l or l_i lies above r_k. At most one side-triangle is present above l_jr_m.

Case 3.4.14 *Cross-diagonals are l_ir_k, l_jr_k, l_jr_l, l_jr_m and l_jr_n. Vertex l_i lies between r_k and r_l.*

Vertex l_j must lie above r_l, since l_jr_k is a cross-diagonal. A side-triangle may lie on the left chain above l_i (see figure 44), a guard on edge l_il_j sees this and all cross-triangles. Any

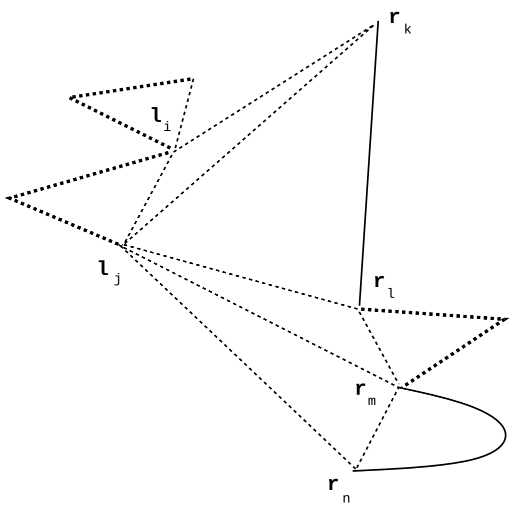

Figure 44: Case 3.4.14

side-triangles between r_m and r_n are left for the next guard assignment. If a single side-triangle lies between l_i and l_j or between r_l and r_m or if all side-triangles lie between r_m and r_n, a guard on edge $l_j l_{j+1}$ sees all triangles. $\diamond$

Case 3.4.15 *Cross-diagonals are $l_i r_k$, $l_j r_k$, $l_j r_l$, $l_j r_m$ and $l_j r_n$. Vertex l_i lies above r_k.*

A single side-triangle may lie anywhere on the right chain above r_m and below l_i or on the left chain between l_i and l_j. The last is possible if l_j lies above r_k. Any number of side-triangles can lie between r_m and r_n. If there is no side-triangle on the left chain, a guard on edge $l_i l_j$ sees any side-triangle present above l_j and all cross-triangles. If there are no side-triangles on the right chain above l_j, a guard on edge $l_j l_{j+1}$ sees all triangles. $\diamond$

Next consider the possibility that there are three endpoints on the left chain and three on the right chain. Let l_i, l_j and l_k be the endpoints on the left chain in order and r_l, r_m and r_n the same or the right chain. Let the cross-diagonals be $l_i r_l$, $l_i r_m$, $l_i r_n$, $l_j r_n$ and $l_k r_n$. We study four possible positions for vertex l_i. At most one side-triangle lies above $l_j r_n$.

Case 3.4.16 *Cross-diagonals are $l_i r_l$, $l_i r_m$, $l_i r_n$, $l_j r_n$ and $l_k r_n$. Vertex l_i lies below r_n.*

No side-triangle can lie on the left side immediately above l_i. A single side-triangle may lie between r_l and r_m, between r_m and r_n or between l_i and l_j (see figure 45). Any number

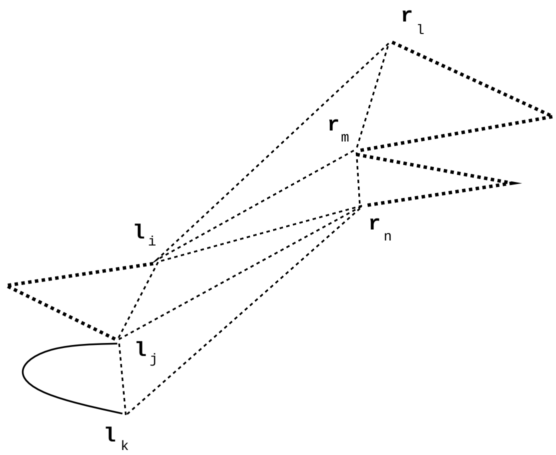

Figure 45: Case 3.4.16.

of side-triangles can lie between l_j and l_k.

This case is similar to symmetric case 3.4.12 where different guards are chosen in different subcases. $\diamond$

Case 3.4.17 *Cross-diagonals are l_ir_l, l_ir_m, l_ir_n, l_jr_n and l_kr_n. Vertex l_i lies between r_m and r_n.*

One side-triangle may lie between r_l and r_m or between l_i and l_j. The last can happen if l_j lies above r_n (see figure 46). A guard on edge r_mr_n sees this side-triangle and all cross-triangles. If vertex l_k lies above r_n this guard also sees any side-triangles between l_j and l_k. If l_j and l_k lie below r_n and all side-triangles are below l_j, a guard on edge l_il_j sees one of these side-triangles and all cross-triangles. Additional side-triangles between l_j and l_k are left unguarded. ◇

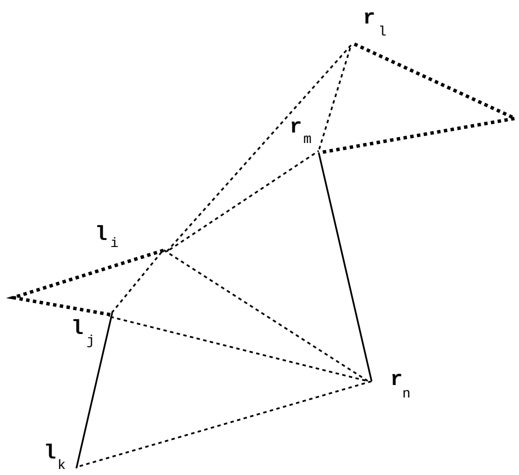

Figure 46: Case 3.4.17.

Case 3.4.18 *Cross-diagonals are l_ir_l, l_ir_m, l_ir_n, l_jr_n and l_kr_n. Vertex l_i lies between r_l and r_m.*

Vertex l_j must lie below r_n, since l_ir_n is a cross-diagonal. One side-triangle may lie on the left side immediately above l_i or on the right side between r_m and r_n (see figure 47). Any number of side-triangles may lie between l_j and l_k. A guard on edge l_il_j sees the single side-triangle, all cross-triangles and at least one side-triangle between l_j and l_k. Additional side-triangles between l_j and l_k are left for the next guard assignment. ◇

Case 3.4.19 *Cross-diagonals are l_ir_l, l_ir_m, l_ir_n, l_jr_n and l_kr_n. Vertex l_i lies above r_l.*

Vertex l_j must lie below r_n, since l_ir_n is a cross-diagonal. If a side-triangle is present above l_jr_n it must lie on the right side (see figure 48). A guard on edge l_il_j sees this side-triangle,

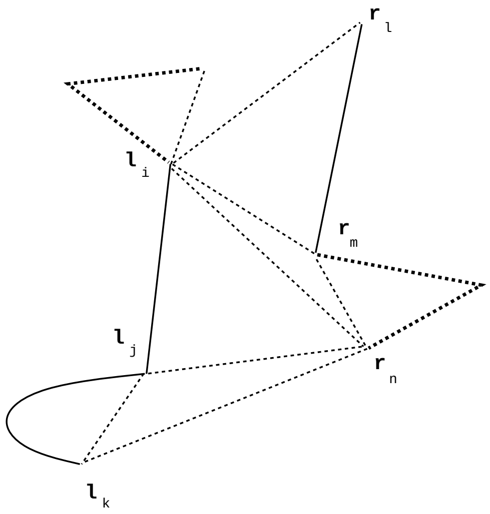

Figure 47: Case 3.4.18.

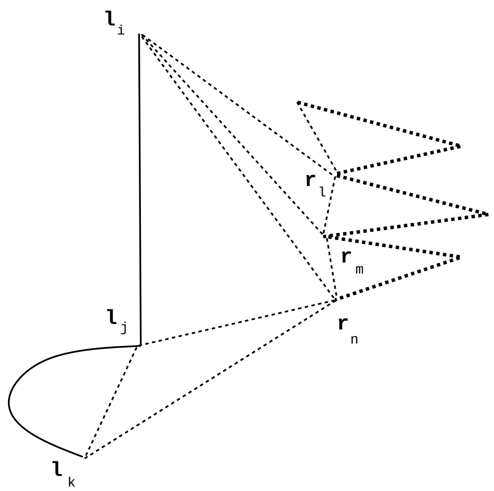

Figure 48: Case 3.4.19.

all cross-triangles and at least one of the side-triangles that may lie below l_j. Additional side-triangles below l_j are left for the next guard assignment. $\diamond$

Next let the cross-diagonals be l_ir_l, l_ir_m, l_jr_m, l_jr_n and l_kr_n. At most one side-triangle lies above l_jr_n. Since l_jr_m is a cross-diagonal, vertex l_i cannot lie below r_n. We are left with three possible positions of vertex l_i.

Case 3.4.20 *Cross-diagonals are l_ir_l, l_ir_m, l_jr_m, l_jr_n and l_kr_n. Vertex l_i lies between r_m and r_n.*

Since l_jr_m is a cross-diagonal, vertex r_n must lie below l_j. The single side-triangle above l_jr_n may lie between r_l and r_m or between l_i and l_j (see figure 49). A guard on edge r_mr_n

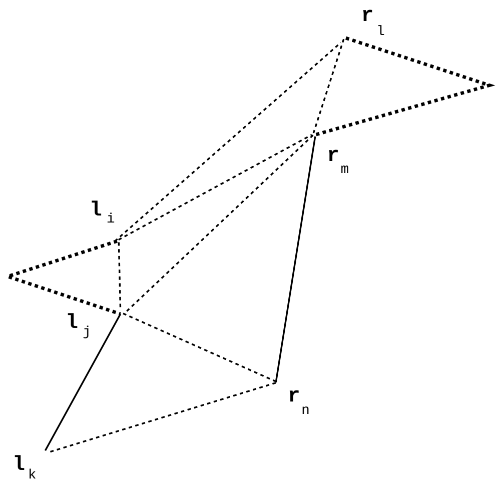

Figure 49: Case 3.4.20.

sees all triangles. $\diamond$

Case 3.4.21 *Cross-diagonals are l_ir_l, l_ir_m, l_jr_m, l_jr_n and l_kr_n. Vertex l_i lies between r_l and r_m.*

Vertex l_k must lie below r_m, since l_jr_n is a cross-diagonal. Furthermore if l_j lies above r_m, vertex l_k must lie below r_n. If this is the case, no side-triangles can be present between l_j and l_k (see figure 50). Exactly one side-triangle is in the group. This may lie above l_i, between l_i and l_j, or between r_m andr_n. This case is symmetric to case 3.4.7.

If l_j lies between r_m and r_n and no side-triangle is present above l_jr_n, a guard on edge r_mr_n sees all triangles.

If l_j lies between r_m and r_n and a side-triangle above l_jr_n is present, or if l_j lies below r_n (see figure 51), a guard on edge l_il_j sees a side-triangle above l_jr_n, all cross-triangles and one of the side-triangles that may lie below l_j. If l_j and l_k lie below r_n, additional side-triangles

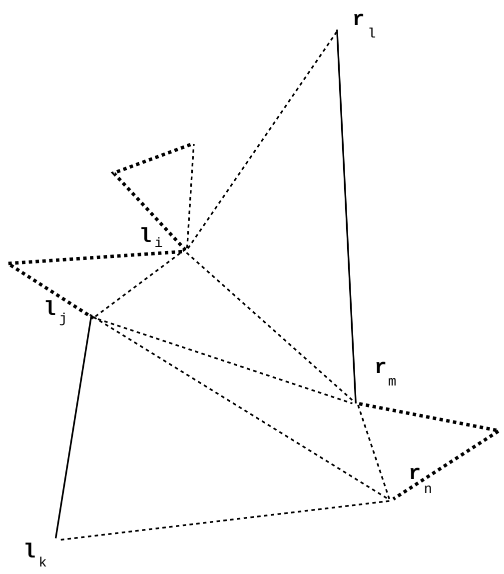

Figure 50: Case 3.4.21. Vertex l_j lies above r_m.

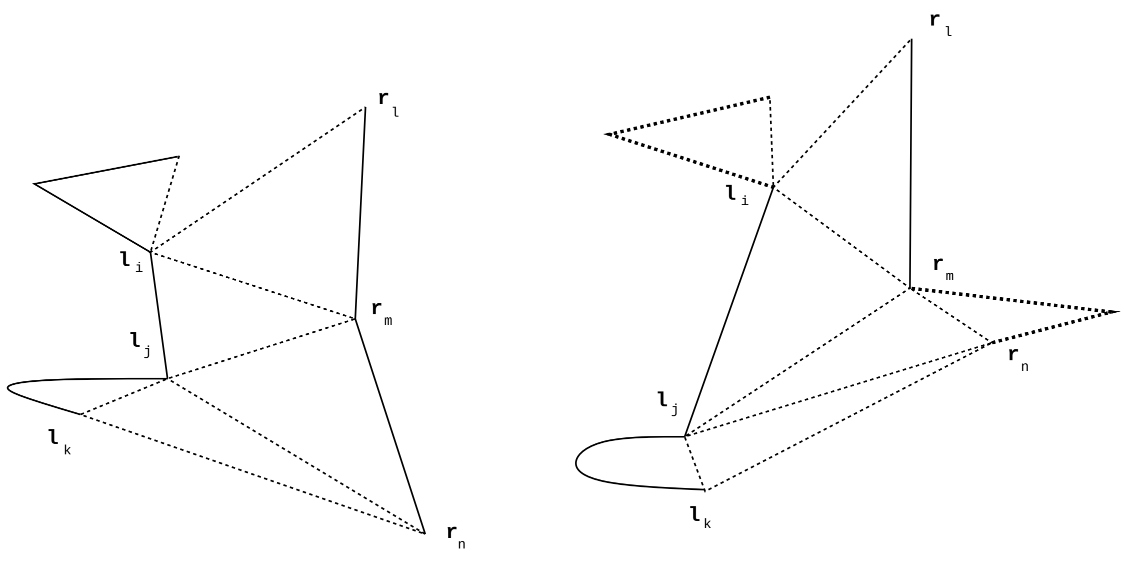

Figure 51: Case 3.4.21. Vertex l_j lies below r_m.

between them are left for the next guard assignment. On the other hand if l_j and l_k lie above r_n, the guard assignment continues below l_jr_n with one less side-triangle counted between the l_j and l_k. $\diamond$

Case 3.4.22 *Cross-diagonals are l_ir_l, l_ir_m, l_jr_m, l_jr_n and l_kr_n. Vertex l_i lies above r_l.*

Since l_ir_m is a cross-diagonal, vertex l_j must lie below r_m. If l_j is also below r_n, a guard on edge l_il_j sees any side-triangle above l_jr_n, all cross-triangles and one of the side-triangles that may be present between l_j and l_k (see figure 52). Additional side-triangles between l_j and l_k are left for the next guard assignment. If vertex l_j lies above r_n and no side-triangles lie above l_jr_n, a guard on edge r_mr_n sees all triangles. If vertex l_j lies above r_n and a side-triangle is present above l_jr_n, a guard on edge l_il_j sees this side-triangle the cross-triangles and one of the side-triangles that may lie on the left chain between l_j and l_k. If two or more side-triangles are present between l_j and l_k, the guard assignment continues below l_jr_n with one less side-triangle counted between l_j and l_k. $\diamond$

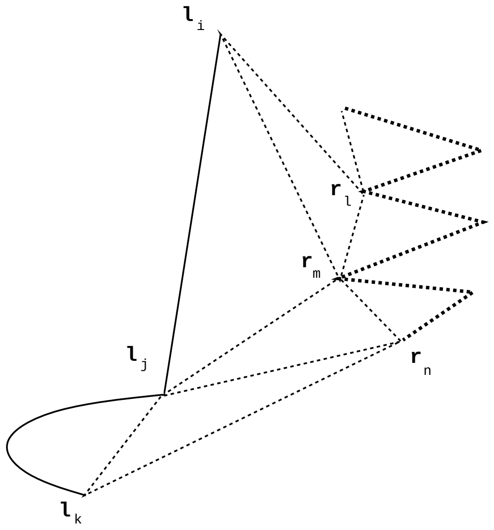

Figure 52: Case 3.4.22.

The cases where only one cross-diagonal emanates from l_i are symmetric to the above cases. We have described twenty two cases of possible configurations with exactly four cross-triangles present. These cases and their symmetric counterparts exhaust the possibilities in this category.

3.5 Exactly five cross-triangles are present.

The six cross-diagonals have endpoints distributed with one, two or three endpoints on one chain and six, five or four endpoints on the opposite chain. Since there are five cross-triangles included in the group, any included side-triangle must lie between the last two cross-diagonals.

First assume that one endpoint l_i lies on the left chain and the six endpoints on the right chain are in order r_j, r_k, r_l, r_m, r_n and r_p. We have two cases.

Case 3.5.1 *Cross-diagonals are l_ir_j, l_ir_k, l_ir_l, l_ir_m, l_ir_n and l_ir_p. Vertex l_i lies below r_n.*

As for case 3.2.1, a guard on edge $l_{i-1}l_i$ sees all triangles. ◇

Case 3.5.2 *Cross-diagonals are l_ir_j, l_ir_k, l_ir_l, l_ir_m, l_ir_n and l_ir_p. Vertex l_i lies above r_n.*

Any side-triangle must lie on the right side between r_n and r_p. A guard on edge l_il_{i+1} sees all triangles. ◇

Next assume there are two endpoints l_i and l_j on the left chain and five endpoints r_k, r_l, r_m, r_n and r_p on the right chain. Either l_ir_p or l_jr_n but not both must be a cross-diagonal. Similarly either l_ir_l or l_jr_k but not both is a cross-diagonal. We study four exhaustive cases. If diagonal l_ir_p is present then vertex l_i can lie below r_n or above r_n. If diagonal l_ir_p is not present then l_jr_n must be a generated diagonal and either diagonal l_ir_l or diagonal l_jr_k is present.

Case 3.5.3 *Cross-diagonal l_ir_p is one of the generated diagonals. Vertex l_i lies below r_n.*

Any side-triangle must lie between l_i and l_j (see figure 53). A guard on edge $l_{i-1}l_i$ sees one such side-triangle and all cross-triangles. If there is more than one side-triangle present, the guard on edge $l_{i-1}l_i$ sees one side-triangle and the four top cross-triangles. In the latter case the guard assignment continues below l_ir_p with one less side-triangle counted between l_i and l_j. ◇

Case 3.5.4 *Cross-diagonal l_ir_p is one of the generated diagonals. Vertex l_i lies above r_n.*

Vertex l_j must lie below r_p, since l_ir_p is a cross-diagonal. Thus no side-triangles can exist between l_i and l_j and a guard on edge l_il_j sees all triangles. ◇

Case 3.5.5 *Cross-diagonal l_jr_n but not l_jr_k is one of the generated cross-diagonals.*

Any side-triangle must lie between r_n and r_p (see figure 54). The guard chosen is the same for all possible positions of l_i. A guard on edge l_il_j sees all cross-triangles. If l_j lies below r_p this guard also sees any present side-triangles. If l_j lies between r_n and r_p, no side-triangles are possible. If l_j lies above r_n, any present side-triangles are let for the next guard assignment. ◇

Case 3.5.6 *Cross-diagonals l_jr_n and l_jr_k are both generated.*

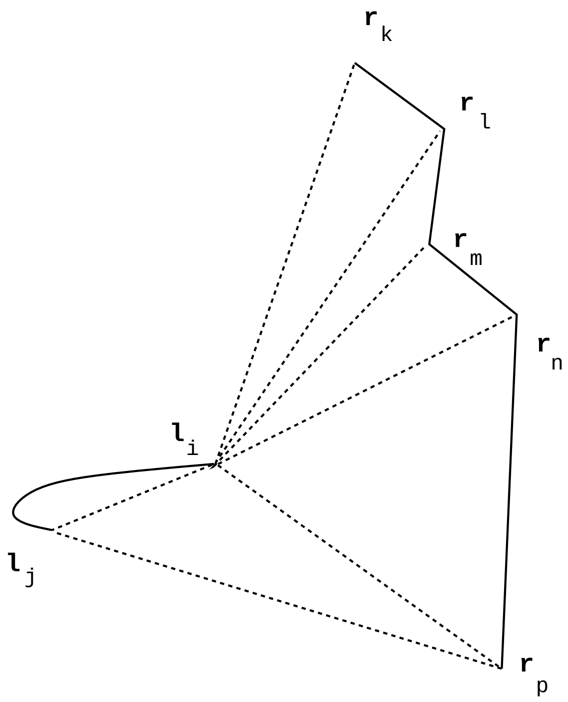

Figure 53: Case 3.5.3.

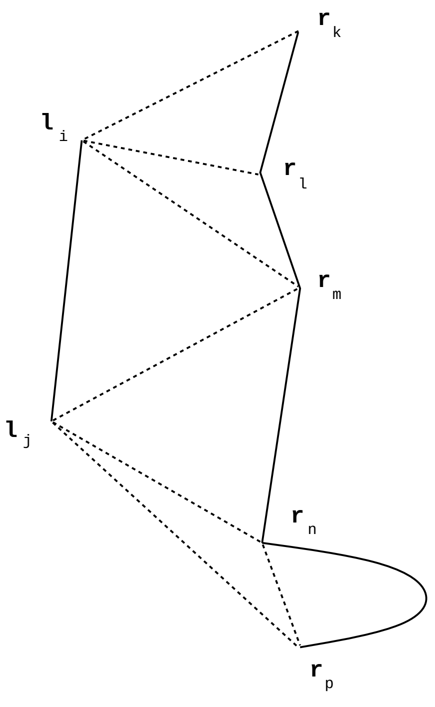

Figure 54: Case 3.5.5.

The choice of guard is the same for all possible positions of l_i. If l_j lies below r_p, a guard on edge $l_i l_j$ sees all triangles. If l_j lies above r_p, a guard on edge $l_j l_{j+1}$ sees all triangles. ◇

Finally assume that there are three endpoints on one chain and four endpoints on the other chain. These endpoints are in order l_i, l_j and l_k on the left chain and r_l, r_m, r_n and r_p on the right chain. Symmetric cases exist when there are four endpoints on the left chain and three endpoints on the right chain. The generated cross-diagonals must include either $l_j r_p$ or $l_k r_n$ but not both and $l_i r_m$ or $l_j r_l$ but not both. We have a total of three cases. Our first case handles all configurations where diagonal $l_j r_p$ is present. For configurations where instead $l_k r_n$ is a diagonal we have two cases, depending on which of the diagonals $l_i r_m$ and $l_j r_l$ is present.

Case 3.5.7 *Cross-diagonal $l_j r_p$ is one of the generated diagonals.*

Any side-triangle present must lie between l_j and l_k (see figure 55). A guard on edge $l_i l_j$

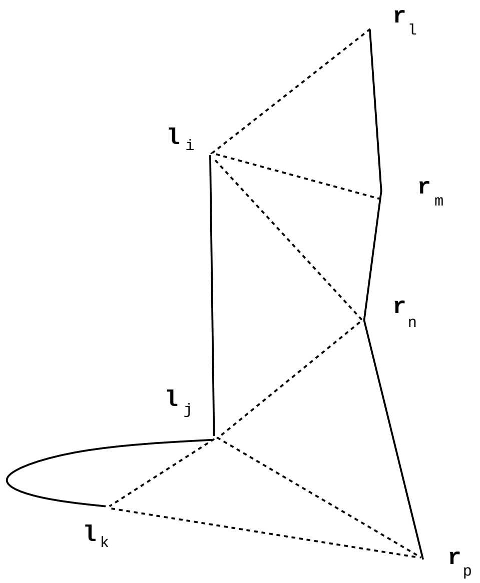

Figure 55: Case 3.5.7.

sees all cross-triangles and one side-triangle below l_j if such is present. If more than one side-triangle is present, the guard assignment continues below $l_j r_p$ with one less side-triangle counted between l_j and l_k. ◇

Case 3.5.8 *Cross-diagonal $l_k r_n$ but not $l_i r_m$ is one of the generated diagonals.*

Any side-triangles must lie between r_n and r_p (see figure 56). A guard on edge $l_j l_k$ sees all cross-triangles. If l_k lies below r_p this guard also sees all side-triangles. If l_k lies above r_p any present side-triangles are left for the next guard assignment. ◇

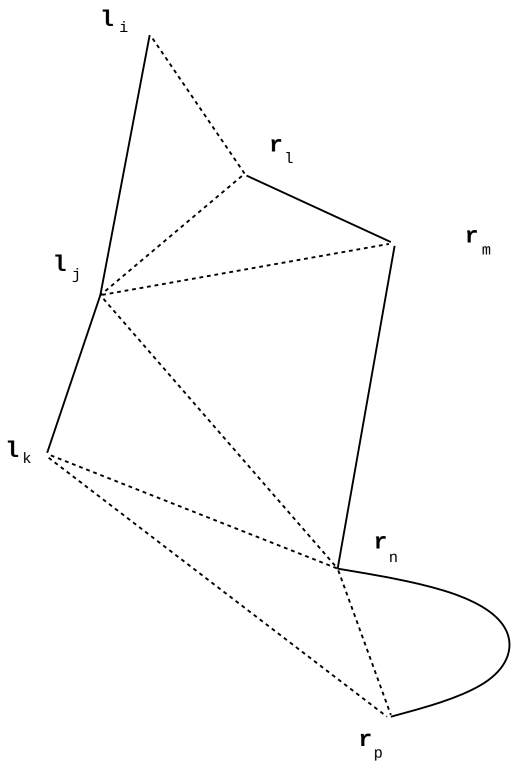

Figure 56: Case 3.5.8.

Case 3.5.9 *Cross-diagonals $l_k r_n$ and $l_i r_m$ are both among the generated diagonals.*

Any side-triangle must lie between r_n and r_p. A guard on edge $r_m r_n$ sees one such side-triangle and all cross-triangles. If more than one side-triangle is present, the guard assignment continues below $l_k r_n$ with one less side-triangle counted between r_n and r_p. $\diamond$

The above cases and their symmetric counterparts exhaust all possible configurations of triangles in a group. It remains to finish the discussion postponed from case 3.4.12. Here a stop was made when cross-diagonal $l_j r_n$ was generated, but no edge above the cross-diagonal can see more than four triangles (see figure 42). We must use an edge below $l_j r_n$. In the continued triangulation the next cross-diagonal will have an endpoint at either l_j or r_n, i.e. the next cross-diagonal is either $l_{j+1} r_n$ or $l_j r_p$, where r_p is a vertex on the right chain. Consider first the possibility that cross-diagonal $l_{j+1} r_n$ is generated. If r_k, l_i, l_{j+1} does not form a right turn, then a guard on edge $l_j l_{j+1}$ sees all the triangles in the group above $l_j r_n$ including any side-triangles between r_m and r_n (see figure 57). It also sees cross-triangle

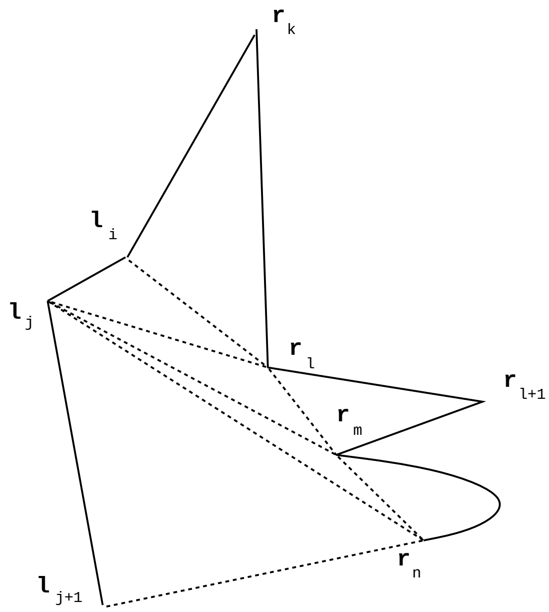

Figure 57: Case 3.4.12. A guard on edge $l_j l_{j+1}$ sees all triangles.

l_j, l_{j+1}, r_n. After assigning this guard the triangulation and guard assignment continues below $l_{j+1} r_n$. If r_k, l_i, l_{j+1} forms a right turn, then since r_k, r_l, r_m is a left turn, r_l, r_m, l_{j+1} must form a right turn and l_{j+1} sees r_m (see figure 58). Let $l_{j+1} r_m$ replace $l_j r_n$ as cross-diagonal, a guard on edge $r_k r_l$ sees the five triangles above $l_{j+1} r_m$. The triangulation continues below $l_{j+1} r_m$ with $l_{j+1} r_n$ as next cross-diagonal. Next consider the possibility that $l_j r_p$ is the generated cross-diagonal (see figure 59). Look at the cases that may be applicable to the next group of triangles. Any such case must have a top endpoint on one chain followed by two endpoints

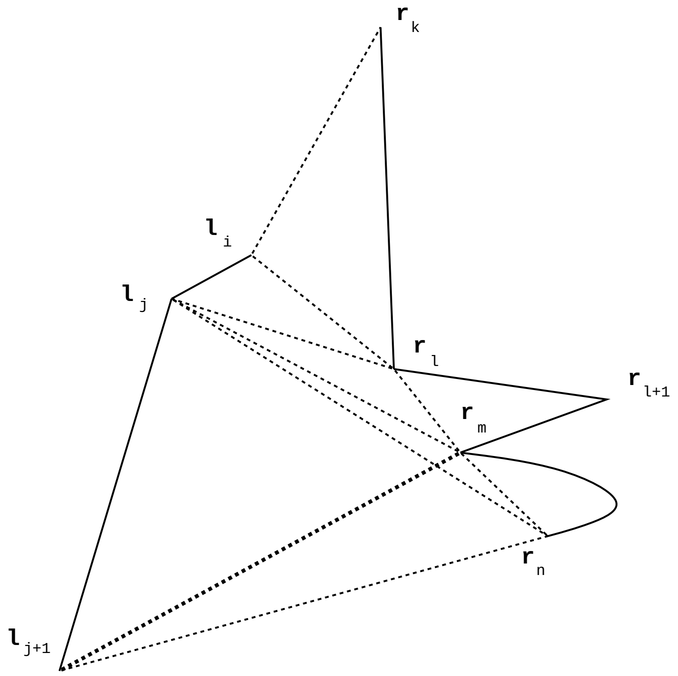

Figure 58: Case 3.4.12. Cross-diagonal $l_j r_n$ can be replaced by $l_{j+1} r_m$.

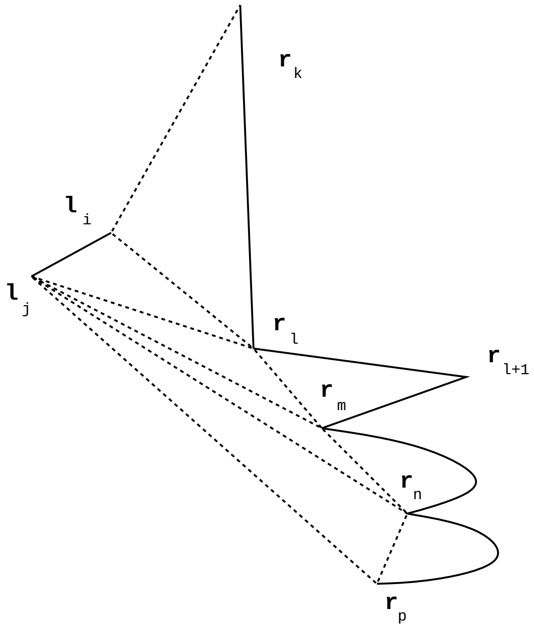

Figure 59: Case 3.4.12. Edges $l_j l_{j+1}$, $r_n r_{n+1}$ and $r_p r_{p+1}$ all see triangle l_j, r_m, r_n.

on the opposite chain with cross-diagonals from the latter two running to the first. The possible edges to which next group's guard can be assigned are $l_j l_{j+1}$, $r_n r_{n+1}$ and $r_p r_{p+1}$. The guards on edges $l_j l_{j+1}$ and $r_n r_{n+1}$ are both guaranteed to see cross-triangle l_j, r_m, r_n in addition to its own triangles. If r_m, r_n, r_p forms a right turn, the same is true for a guard on edge $r_p r_{p+1}$. Thus a guard on edge $r_k r_l$ together with next group's guard see all triangles above $l_j r_n$ and the triangles in the next group in these cases. It remains to choose a guard when $l_j r_p$ is a cross-diagonal, next group's guard would be assigned to $r_p r_{p+1}$ and r_m, r_n, r_p does not form a right turn. To do so we check if $l_{j+1} r_p$ is a cross-diagonal. If it is then either r_k, l_i and l_{j+1} form a left turn and a guard on edge $l_j l_{j+1}$ sees all triangles above $l_{j+1} r_p$ or r_k, r_i and l_{j+1} form a right turn, cross-diagonals $l_j r_n$ and $l_j r_p$ can be replaced by $l_{j+1} r_m$ and $l_{j+1} r_n$ respectively and a guard on edge $r_k r_l$ sees all triangles in the top group (see figure 60). Finally it is possible that r_m, r_n, r_p does not form a right turn, edge $r_p r_{p+1}$ is assigned a guard

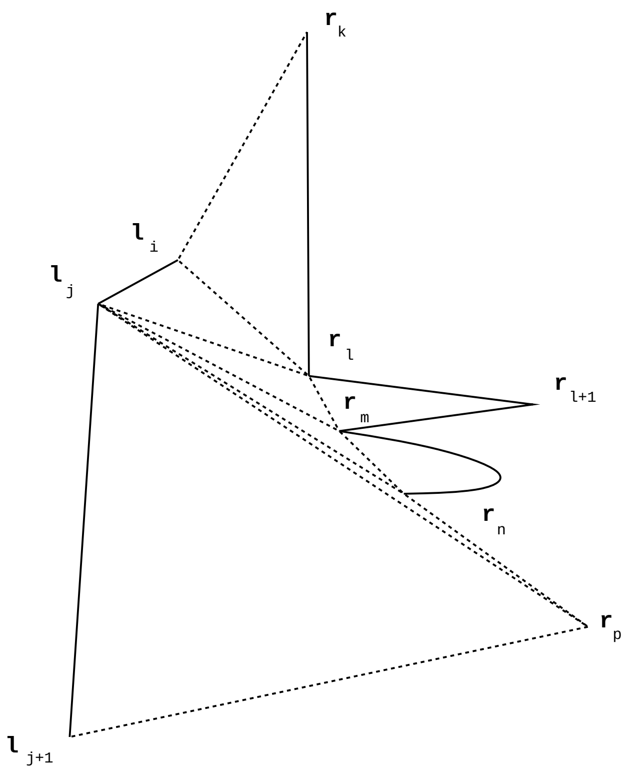

Figure 60: Case 3.4.12. Cross-diagonal $l_{j+1} r_p$ exists.

and $l_{j+1} r_p$ is not a cross-diagonal (see figure 61). This situation occurs only in cases 3.5.8 and 3.5.9. Here cross-diagonal $l_{j+1} r_{p+1}$ definitely exists. Using the same reasoning as above we see that if r_m, r_n, r_{p+1} forms a right turn a guard on edge $r_p r_{p+1}$ sees also cross-triangle $l_j r_m r_n$ and if r_m, r_n, r_{p+1} does not form a right turn then either edge $l_j l_{j+1}$ sees all triangles above $l_{j+1} r_{p+1}$ or cross-diagonals $l_j r_n$, $l_j r_p$ and $l_j r_{p+1}$ can be replaced by $l_{j+1} r_m$, $l_{j+1} r_n$ and $l_{j+1} r_p$ respectively and a guard on edge $r_k r_l$ sees all five triangles above $l_{j+1} r_m$.

Some of the configurations discussed above resulted in a guard assignment where the top

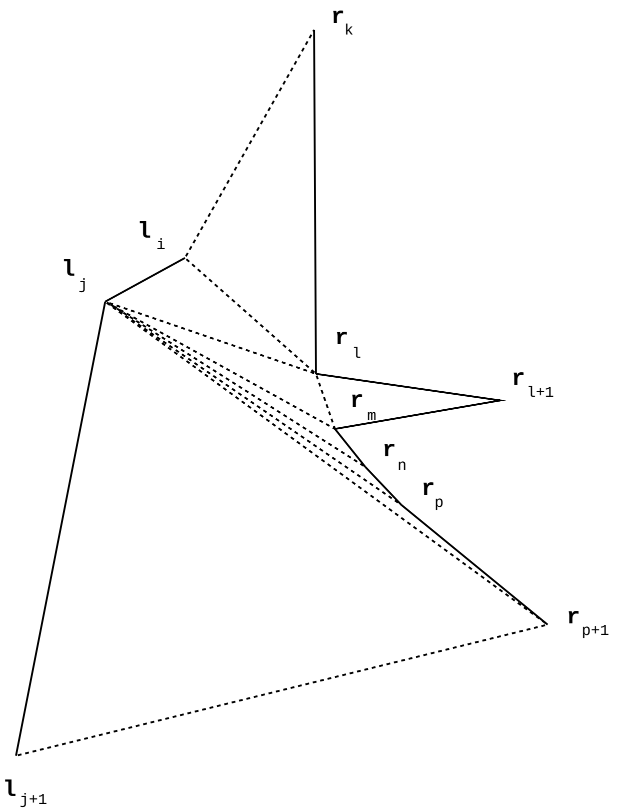

Figure 61: Case 3.4.12. Cross-diagonal $l_j r_{p+1}$ exists.

group has only four triangles. The fifth triangle is covered by the guard in the group below and as long as this group is not the last group in the polygon the two groups together contain at least ten triangles. If the group below $l_j r_n$ is the last group of triangles in the polygon, the two groups may have less than ten triangles. However, if there is at least one triangle below $l_j r_n$, the two guards chosen as above are still within the $\lceil (n-2)/5 \rceil$ bound. If no triangles at all lie below $l_j r_n$ then no more than one guard may be used. However, in this case $l_j r_n$ is in fact an edge of the polygon. A guard on this edge sees all the triangles in the group above.

The assignment of guards as described in all the cases above ensures that no more than $(n-2)/5$ guards are used when the number of triangles is a multiple of five. With one additional guard assigned to any remaining triangles the number of guards is at most $\lceil (n-2)/5 \rceil$. Given the lower bound from section 2, we have the following theorem:

Theorem 1 $\lceil (n-2)/5 \rceil$ *edge guards are sometimes necessary and always sufficient to cover a monotone polygon of n vertices.*

4 Algorithmic aspects

In the proof of the upper bound the polygon is triangulated using an existing algorithm for monotone polygons [4]. This algorithm produces a triangulation with a very predictable structure. Taking advantage of this structure, we modify the algorithm to enable it to assign edge guards.

The main modification is the addition of a conditional statement, where each case results in the output of an edge guard. We also include a count, which keeps track of the number of unguarded triangles above the most recently generated cross-diagonal. The conditional statement is run through each time the count is at least five.

To distinguish between the cases we need to know what the configuration of the currently unguarded triangles looks like. Specifically we need to know the vertices of the cross-triangles, the number of side-triangles above each such vertex and the generated cross-diagonals. Therefore where the original algorithm outputs a cross-diagonal the modified algorithm instead stores this diagonal in an array CD. The left and right endpoints are also stored in two separate arrays L and R. Where the original triangulation algorithm outputs a diagonal with both endpoints on the same chain, the modified algorithm increases by one a count of the number of side-triangles for the vertex which is the bottom vertex of the diagonal. The number of side-triangles for each vertex is kept in the array ST. In a collection of adjacent side-triangles the last diagonal to be generated has endpoints that are vertices on a cross-triangle. Each time a cross-diagonal is generated, the count of side-triangles associated with the bottom vertex is added to the count of unguarded triangles. Since the information for a particular group of triangles is no longer needed after a guard for the group has been assigned, the arrays L, R and CD are of constant size. The algorithm also maintains a stack $v_o, ..., v_t$ with v_0 on the bottom and v_t on the top of the stack. The variables l, r and cd are the indices of the arrays L, R and CD respectively.

Algorithm

Sort vertices by decreasing y-coordinate, resulting in $P[0], P[1], ..., P[n-1]$.

```
For i = 1 to n − 1 let ST[i] ← 0
Push 0. The index of a vertex is pushed, not the vertex itself.
Push 1.
If P[1] lies on left side then L[0] ← 1, R[0] ← 0 and CD[0] ← 0, 1
else L[0] ← 0, R[0] ← 1 and CD[0] ← 0, 1
l ← r ← cd ← 1
i ← 2
while i < n do
        if P[i] is adjacent to P[v_0] then
                if P[i] lies on left side then
                        L[l] ← i; l ← l + 1;
                        CD[cd] ← i, v_1; cd ← cd + 1;
                else R[r] ← i; r ← r + 1;
                        CD[cd] ← v_1, i; cd ← cd + 1;
                endif
                count ← count + 1 + ST[i];
                if count ≥ 5 then place-a-guard;
                j ← 1;
                while v_j ≠ v_t do
                        begin j ← j + 1;
                                if P[i] lies on left side then
                                        R[r] ← v_j; r ← r + 1;
                                        CD[cd] ← i, v_j; cd ← cd + 1;
                                else L[l] ← v_j; l ← l + 1;
                                        CD[cd] ← v_j, i; cd ← cd + 1;
                                endif
                                count ← count + 1 + ST[v_j];
                                if count ≥ 5 then place-a-guard;
                        endwhile
                        replace stack by v_j, i;
                end
        else if P[i] is adjacent to P[v_t] then
                while t > 1 and angle at P[v_t] is not reflex do
                        ST[i] ← ST[i] + 1 + ST[v_t];
                        pop v_t;
                        t ← t − 1;
                endwhile
                if angle at P[v_t] is reflex then
                        push i;
                else if P[i] lies on the left side then
```

```
                $L[l] \leftarrow i; l \leftarrow l + 1;$
                $CD[cd] \leftarrow i, v_0; cd \leftarrow cd + 1;$
            else $R[r] \leftarrow i; r \leftarrow r + 1;$
                $CD[cd] \leftarrow v_0, i; cd \leftarrow cd + 1;$
            pop stack
            count ← count + 1 + ST[i];
            if count ≥ 5 then place-a-guard;
            push i;
    end
i ← i + 1;
endwhile
if count > 0 then place-a-guard;
end
```

The routine place-a-guard is a conditional statement, which distinguishes between the five main categories of cases. It also handles the case where one group has four triangles and a following group has six or more triangles. When this happens we may need to collect as many as seven cross-triangles before the choice of guard for the first group can be made. The index cd is always one larger than the number of cross-triangles present in the group for which a guard is sought.

```
place-a-guard
if cd = 2 then category1;
else if cd = 3 then category2;
else if cd = 4 then category3;
else if cd = 5 then category4;
else if cd = 6 then category5;
else if cd = 7 then category6;
else if cd = 8 then category7;
end
```

Within each category the number of vertices on the left side, the relative position of vertices in L and R and the number of side-triangles above each such vertex are used to further distinguish between the cases. For example for the category where exactly one cross-triangle is present we have:

```
category1
if l = 1 then One endpoint is on the left side and two endpoints are on the right.
    if vertex with index L[0] lies below vertex with index R[1] then
        output edge which ends at vertex with index L[0];
    else if vertex L[0] lies below vertex R[0] then
        output edge above vertex R[1];
    else output edge below vertex L[0];
```

else

Two endpoints are on the left side and one endpoint on the right side.
The cases are symmetric to those above.
...

endif
$leftovers \leftarrow 0$
$L[0] \leftarrow L[l-1]$;
$R[0] \leftarrow R[r-1]$;
$CD[0] \leftarrow CD[cd-1]$;
$l \leftarrow r \leftarrow cd \leftarrow 1$;
$count \leftarrow 0$
end

In category 1, no triangles in the group are left unguarded after the guard has been assigned. Thus the count can simply be reset to 0. In category 2 this is no longer true and the variable $leftovers$ is used to keep track of any such triangles. In this category we also need to check entries in the array CD.

```
category2
if l = 1 then
    if vertex with index L[0] lies below vertex with index R[1] then
        output edge above vertex with index L[0];
        leftovers ← 0;
    else if vertex with index L[0] lies below vertex with index R[0] then
        if leftovers ≥ 2 then
            output edge below vertex with index R[0];
            leftovers ← ST[R[2]] − 1;
        else output edge below vertex with index L[1];
            leftovers ← 0;
    else output edge below vertex with index L[1];
        leftovers ← 0;
else if l = 2 then
    if CD[1] = 0, 1 then
        if vertex with index L[0] lies below vertex with index R[1] then
            if ST[R[1]] ≥ 2 then
                output edge above vertex with index L[0];
                leftovers ← ST[L[1]] − 1;
            else output edge below vertex with index R[1]];
                leftovers ← 0;
            endif
        else if vertex with index L[0] lies below vertex with index R[0] then
            output edge above vertex with index R[1];
            leftovers ← 0;
```

else output edge above vertex with index $L[1]$;
    $leftovers \leftarrow 0$;
endif

else

The cross-diagonal runs between the vertices with indices $L[1]$ and $R[0]$.
The cases are symmetric to those above.
...

else

There are three endpoints on the left side.
These cases are symmetric to those for $l = 1$ above.
...

end

The routines for the other categories are similar. Checking the number of cross-triangles in the group and the number of vertices in L (i.e. checking cd and l) takes constant time. So does comparing the entries in CD against those present in a particular case, making any needed comparisons between vertices in L and R and looking up the number of side-triangles associated with a vertex. Since there is a constant number of cases, it takes constant time to output one edge guard once the diagonals for the group have been generated. The triangulation algorithm takes $O(n)$ time to generate all the diagonals, where n is the number of vertices. The number of edge guards assigned is also $O(n)$. With constant time taken by the assignment of each guard, total time taken is $O(n)$.

5 Conclusions

In this paper we studied the problem of assigning edge guards in monotone polygons. We showed a tight bound for the number of guards required and gave a linear time algorithm for placing the guards. The upper bound on the number of edge guards was arrived at by triangulating the polygon, grouping the triangles into groups of at least five and assigning one edge guard per group. The algorithm was derived from the constructive proof.

The same method, using convex quadrilaterals instead of triangles, has been used to show that $\lceil (n-2)/6 \rceil$ is a tight bound on the number of edge guards required to cover any rectilinear monotone polygon [2] and to provide a linear time algorithm for placing these guards.

The problem of finding a tight bound for the number of edge guards required in any arbitrarily shaped simple polygon remains unsolved. The best known upper bound is $\lfloor n/3 \rfloor$ edge guards. This is arrived at by simply taking the $\lfloor n/3 \rfloor$ vertex guards known to cover the polygon and assigning the edges incident to these vertices as edge guards. A general example of a polygon with n vertices that requires as many as $\lfloor n/4 \rfloor$ edge guards is shown in [5]. This example is due to Toussaint, who conjectures that $\lfloor n/4 \rfloor$ is a tight bound on the number of guards, with the qualification that certain polygons with a small number of vertices are excluded. Of the two excluded polygons, discovered by Paige and Shermer [6], one has seven vertices and requires two edge guards while the other has eleven vertices and requires three edge guards. Thus if these polygons are not excluded the lower bound for the number of edge guards required in an arbitrarily shaped simple polygon is $\lfloor (n+1)/4 \rfloor$. If the method of proof used for monotone polygons can be shown to work for arbitrary polygons, i.e. if it can be shown that the triangles of the triangulated polygon can be grouped into groups of at least four with one edge guard assigned per group, the resulting upper bound would be $\lceil (n-2)/4 \rceil$, which of course is the same as $\lfloor (n+1)/4 \rfloor$.

References

[1] Aggarwal, A. "The Art Gallery Theorem: Its Variations, Applications and Algorithmic Aspects," Ph.D. thesis, Johns Hopkins University, 1984.

[2] Bjorling-Sachs, I. "A Tight Bound for Edge Guards in Rectilinear Monotone polygons," In preparation.

[3] Chvatal, V. "A Combinatorial Theorem in Plane Geometry," *J. Combinatorial Theory Series* B 18 (1975), 39-41.

[4] Garey, Johnson, Preparata and Tarjan, "Triangulating a Simple Polygon," *Info. Proc. Letters* 7 (1978), 175-180.

[5] O'Rourke, J. "Art gallery problems and algorithms," Oxford University Press (1987).

[6] Shermer, T. "Recent Results in Art Galleries," Simon Fraser University Technical Report, October 1990.