Covering Class-3 Orthogonal Polygons with the Minimum Number of r-Stars

Leonidas Palios* and Petros Tzimas
Department of Computer Science, University of Ioannina
GR-45110 Ioannina, Greece
{palios, ptzimas}@cs.uoi.gr

Abstract

We consider the problem of covering simple orthogonal polygons with the minimum number of r-stars. A point q in an orthogonal polygon P is r-visible from a point $p \in P$ if the axis-parallel rectangle with p, q at opposite corners lies within P; then, an orthogonal polygon P is an r-star if there exists a point $p \in P$ such that every point in P is r-visible from p. The problem of covering a simple orthogonal polygon with the minimum number of r-stars has been considered by Worman and Keil [10] who described an $O(n^{17}\text{poly-}\log n)$-time algorithm where n is the size of the given polygon.

In this paper, we consider the above problem on simple class-3 orthogonal polygons; a class-3 orthogonal polygon is defined to have dents along at most 3 different orientations. By taking advantage of geometric properties of these polygons, we are able to provide an $O(n^2)$-time algorithm; this is the first purely geometric algorithm for this problem and it paves the way for obtaining algorithms for the problem on general simple orthogonal polygons that are faster than Worman and Keil's.

Keywords: orthogonal polygon, covering, decomposition, r-star, visibility

1 Introduction

Motivated by a question of Klee in 1973 and thanks to the work of Chvátal and Fisk (see [8]), the now-classic *Art Gallery Theorem* states that for an n-sided simple polygon, $\lfloor n/3 \rfloor$ immobile guards are sometimes necessary and always sufficient such that every point of the polygon is watched by at least one of the guards [8].

Since then, many variants have been considered making the field of Art Gallery problems a vibrant and large research area in combinatorial and computational geometry [8, 9]. The multitude of variants is in part due to the fact that getting the minimum number of guards to watch a given polygon is NP-complete as shown by Aggarwal [1]. This stimulated research in restricted types of polygons or with guards possessing different visibility or mobility characteristics.

Guarding problems have been considered on *orthogonal* polygons, i.e., polygons whose edges are either horizontal or vertical. It turns out that fewer guards (in terms of the size of the polygon) are needed for such a polygon since the art gallery theorem in this case states that $\lfloor n/4 \rfloor$ immobile guards are sometimes necessary and always sufficient such that every point of the polygon is watched by at least one of the guards [4].

* This research has been co-financed by the European Union (European Social Fund - ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the national Strategic Reference Framework (NSRF) - Research Funding Program: THALIS UOA (MIS 375891).

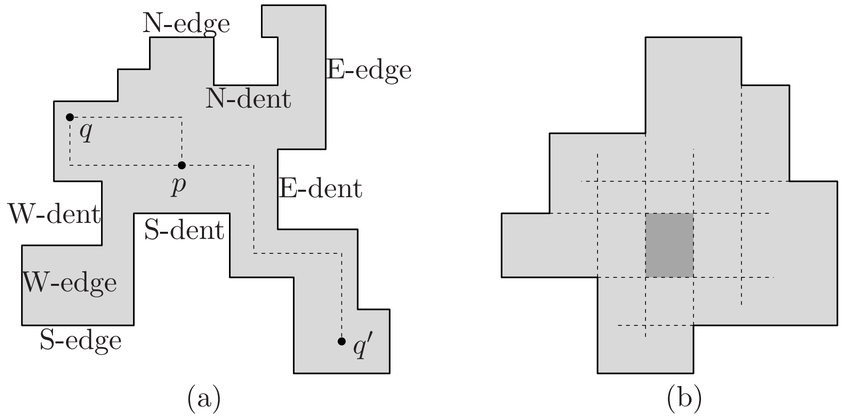

Figure 1: (a) Illustration of the main definitions; (b) an r-star with its kernel shaded.

Since the edges of an orthogonal polygon are either horizontal or vertical, we can characterize them using the compass directions (N, S, E, W); more specifically, an edge is a N-edge (S-edge, E-edge, and W-edge, resp.) if the vector normal to the edge and pointing outward is directed towards the North (South, East, and West, resp.). Of particular importance are edges whose both endpoints are reflex vertices of the polygon; such edges are called *dents* and as above they are characterized as N-dents, S-dents, E-dents, and W-dents (see Figure 1(a)). Orthogonal polygons can be classified in terms of the types of dents that they contain [2]: a *class-k* orthogonal polygon ($0 \leq k \leq 4$) is defined to have dents along at most k different orientations. Class-2 polygons can be further classified into class-2a where the 2 dent orientations are parallel (i.e., N and S, or E and W), and class-2b where the 2 dent orientations are perpendicular to each other.

Essential to a guarding problem is the notion of visibility of the guards used. According to the standard visibility definition on a polygon P, two points p and q of P are visible from one another if and only if the entire line segment pq belongs to P. Especially for orthogonal polygons, two more types of visibility have been defined and used, r-visibility and s-visibility: in an orthogonal polygon P, two points p, q of P are *r-visible* from one another if and only if the axis-parallel rectangle with p, q at opposite corners lies within P whereas p and q are *s-visible* from one another if and only if there exists a staircase path from p to q that lies entirely in P (a staircase path is a chain of axis-parallel edges with bends that alternate between exactly two orientations – in Figure 1(a) the staircase path from p to q' has alternating down-turns and right-turns). Then, a polygon P is an *r-star* (*s-star*, resp.) if there exists a point p of P such that every point $q \in P$ is r-visible (s-visible, resp.) from p; the set of all such points p in P is called the *kernel* of the r-star (s-star, resp.). Figure 1(b) shows an r-star with its kernel shaded.

Clearly, the problem of determining a minimum set of r-visibility (or s-visibility) guards to watch a simple polygon is equivalent to determining a minimum covering of the polygon by r-stars (or s-stars, respectively). A *covering* of a polygon P into a set S of pieces (or subpolygons or components) requires that the union of the pieces in S is equal to P. If additionally the pieces are required to be mutually disjoint (except along boundaries), then we have a *partition*. Obviously, a partition of a polygon also forms a covering of the polygon; thus, a minimum-size covering of a polygon involves at most as many pieces as a minimum-size partition of the polygon into the same type of pieces, and consequently coverings are better than partitions in terms of the number of pieces. On the other hand, covering problems prove to be harder than their corresponding partition problems and there are cases where the former are NP-hard whereas the latter admit polynomial solutions.

Covering by r-stars has been investigated early enough. Keil [5] described an $O(n^2)$-time algorithm to cover a class-2a orthogonal polygon by r-stars. Culberson and Reckhow [2] showed that Keil's algorithm is worst-case optimal if the r-stars need to be explicitly reported and presented an $O(n)$-time algorithm to count the number of r-stars needed; they also gave $O(n^2)$-time algorithms for minimally covering class-2a as well as class-2b orthogonal polygons. Soon afterwards, Motwani, Raghunathan, and Saran [7] studied s-star coverings. They showed a close relation between minimum-size coverings of orthogonal polygons by s-stars and coverings of perfect graphs with the minimum number of cliques; they took advantage of this very interesting idea to derive an $O(n^8)$-time algorithm for covering an orthogonal polygon by the minimum number of s-stars and an $O(n^3)$-time algorithm for the same problem in the case that the orthogonal polygon is class-3. Returning back to r-stars, Gewali, Keil, and Ntafos [3] considered the problem of covering class-2a orthogonal polygons by the minimum number of r-stars and they gave an $O(n)$-time algorithm to report the locations of a minimum-cardinality set of guards. Their algorithm was improved by Lingas, Wasylewicz, and Żyliński [6] who were able to perform the computations in the two passes of the algorithm of Gewali et al. into a single pass; they also reduced the space requirement (in addition to the space required to store the polygon) to linear in the number of guards required rather than linear in the size of the polygon. The problem of covering general orthogonal polygons with r-stars was addressed by Worman and Keil who took advantage of the graph-theoretic approach to describe an $O(n^{17}\text{poly-log}n)$-time algorithm [10].

In this paper, we study the r-star covering problem for class-3 orthogonal polygons. We take advantage of geometric properties of these polygons and we describe an $O(n^2)$-time algorithm to report the locations of a minimum-cardinality set of r-visibility guards to watch the entire polygon by sweeping the polygon a single time. This is the first purely geometric algorithm for this problem and it paves the way for obtaining algorithms for the problem on general simple orthogonal polygons that are faster than Worman and Keil's.

2 Theoretical Framework

We consider simple orthogonal polygons; so, in the following, we will omit the adjective simple.

Consider an orthogonal polygon P that does not have N-dents. The intersection of such a polygon with a horizontal line L may consist of several line segments. Since P has no N-dents, these line segments correspond to *disjoint* parts of the polygon P below the line L; for convenience, we call each such part of P a *trouser*. Next, we give extensions of the notions of "level" and "grid segment" used in [3]: the *level* of a point or a horizontal edge of a polygon is its y-coordinate; the *grid segment* of a trouser T at a level ℓ is the intersection of T with a horizontal line $y = \ell$. We also use the notion of orthogonal projection given in [6]: the *orthogonal projection* $o(s)$ of a horizontal line segment s (which may be a grid segment or a horizontal edge) in an orthogonal polygon P onto the grid segment s_ℓ at level ℓ is the *maximal* subsegment of s_ℓ such that there exists a vertical line segment in P crossing both s and $o(s)$. Finally, we define the *x-range* of a horizontal edge e of a polygon to be the set of x-coordinates of the points of e. In other words, if the endpoints of a horizontal edge have x-coordinates x_l and x_r with $x_l < x_r$, then the x-range of the edge is (x_l, x_r); we note that although a polygon is considered a closed set, we consider edges to be open sets (i.e., they do not include their endpoints) and thus the x-ranges are open sets as well.

The following lemma provides three very important properties of class-3 orthogonal polygons.

Lemma 2.1 *Let P be a class-3 orthogonal polygon and assume that P has no N-dents. Then, the following hold.*

(i) The polygon P has a single topmost edge.

(ii) Consider sweeping the polygon from bottom to top. Each edge encountered is incident on the boundary of the swept polygon.

(iii) Consider levels ℓ, ℓ_1, and ℓ_2 such that $\ell_1 < \ell_2 \leq \ell$ and a trouser T while P is intersected by a horizontal line at level ℓ. The orthogonal projection of the grid segment of T at level ℓ_1 onto ℓ is a subset of the orthogonal projection of the grid segment of T at level ℓ_2 onto ℓ.

Proof: Statements (i) and (ii) easily follow from the lack of N-dents. Statement (iii) follows from the observation that the orthogonal projection of the grid segment of T at level ℓ_1 onto the level ℓ_2 is a subset of the grid segment of T at level ℓ_2. ∎

3 The Algorithm

Our algorithm applies the plane-sweep paradigm as do the algorithms in [3, 6]. We assume that the given class-3 polygon does not have N-dents and we sweep it from bottom to top (thus we can take advantage of Lemma 2.1); the algorithm reports the locations of a minimum-cardinality set of r-visibility guards to watch the entire input polygon. Below we discuss the key ingredients of our algorithm and subsequently we give its description in detail and analyze its complexity.

Maintaining and Processing Guards

We follow the convention in the algorithms of Gewali, Keil, and Ntafos [3] and later of Lingas, Wasylewicz, and Żyliński [6], according to which the guards are placed at the leftmost possible point of the highest possible level; thus each guard is located at the level of a N-edge. Then in order to find the appropriate locations of the guards, with each guard we maintain:

- a *location-range*, or *loc-range* for short, which is the range of x-coordinates of the points at which the guard can currently be placed;
- a *visibility-range*, or *vis-range* for short, which is the range of x-coordinates of the points *above* the current position of the sweep-line that are visible to the guard.

Since there are no N-dents, each of these ranges is a *single interval* of x-coordinates, and because we place guards so that they can see as much of the polygon above them as possible, it always holds that the loc-range of a guard is a subset of its vis-range.

For a guard g to be placed at level ℓ in a trouser T, initially its loc-range and its vis-ranges coincide with the range of x-coordinates of the grid segment of T at level ℓ. As the sweep-line moves upward, both ranges get clipped by N-edges whose x-ranges intersect them. Finally, when a N-edge e is encountered such that the (possibly clipped) loc-range of g is a subset of the closure of the x-range of e, then g is placed at the point (x_l, ℓ) where x_l is the left bound value of g's loc-range right before the N-edge e is encountered (in accordance with the convention followed by [3, 6]); g cannot see any points in the polygon P above the level of the edge e.

Figure 2 exhibits the changes in the loc-ranges and vis-ranges of a guard as we sweep the orthogonal polygon shown from bottom to top. Before we reach edge e_1, all the points in the swept portion of the polygon can be seen from guards (at appropriate locations) higher than the sweep-line location. This ceases to be true (consider the points below the N-edge e_1) when e_1 is reached; this implies that a guard needs to be placed at the level of e_1 with its loc-range and vis-range initialized to the x-range of the grid segment at the level of e_1. Both ranges are clipped after processing e_1 and clipped further after processing e_2. When the N-edge e_3 is reached, we see that a guard is needed to watch edge e_5; then, the loc-range of the guard is further clipped to the closure of the x-range of e_5 (the vis-range does not change). Again, both the loc-range and vis-range are clipped by the edge e_3 and they end up as shown in the figure at the level of e_3. Finally, the resulting loc-range is a subset of the closure of the x-range of the N-edge e_4 and then the guard g is positioned at point p as shown. (We note that a second guard needs to be placed so that the polygon in Figure 2 is watched in its entirety.)

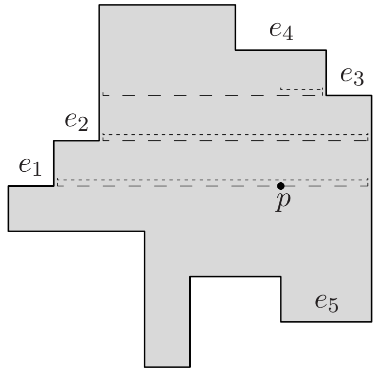

Figure 2: An example of the changes in the loc-range (shown dotted) and vis-range (shown dashed) of a guard; both ranges are shown *after* the processing of the N-edge at the same level.

Figure 3: (a) A guard needs to be placed no higher than the N-edge e_2 to watch the S-edge e; (b) the f-range (shown dotted) and the p-range (shown dashed) of the S-edge e.

Determining Where to Place a Guard

Consider any S-edge e of the given polygon; see Figure 3(a). As long as the x-range of no encountered N-edge intersects the x-range of e, then a guard at a level higher than the level of the N-edge can see the entire e; this is the case with the N-edge e_1 in Figure 3(a). However, if the x-range of a N-edge d intersects e's x-range, then a guard must be placed at a level *between (and including) the levels of* e *and* d since no guard at a level higher than the level of d can see the entire e; this is the case with the N-edge e_2 in Figure 3(a). Additionally, if ℓ is the level to place such a guard, the requirement that the guard sees the entire edge e implies that the guard needs to be placed at *the orthogonal projection of the grid segment at the level of* e *onto level* ℓ.

Therefore, in order to enforce the above observations, with each S-edge e we maintain:

- a *forcing-range*, or *f-range* for short, which is the x-range of the edge e;
- a *placement-range*, or *p-range* for short, which is the range of x-coordinates of the grid segment at the level of e.

Each of these ranges is a *single interval* of x-coordinates (the f-range is open, the p-range is closed), and it always holds that the f-range of a S-edge is a subset of its p-range. Figure 3(b) shows the f-range (shown dotted) and p-range (shown dashed) for the S-edge e.

Here is how these two ranges of an S-edge e are used: During the sweeping, as long as we encounter N-edges whose x-ranges do not intersect either range, no change occurs. If we encounter a N-edge whose x-range intersects the p-range of e, then the p-range simply gets clipped. However, if we encounter a N-edge d whose x-range intersects the f-range of e, then a

Figure 4: The second kind of guard-requests (f-range shown dotted, p-range shown dashed).

guard is needed immediately in order to watch the S-edge e (as we mentioned while discussing Figure 3(a)). Any guard meeting the following 2 conditions will do:

- the guard should be located at a level between (and including) the levels of e and d, and
- the guard's loc-range should intersect the p-range of the edge e.

Then, the loc-range of the guard chosen to watch the S-edge e is set equal to the intersection of the current value of the loc-range with the p-range of e; in this way, the guard will be able to watch both e and as much of the *unseen* polygon as possible (this is the reason for the clipping of the p-range of e). Since the f-range and p-range of a S-edge help us determine the need for a guard to watch this edge and the range of x-coordinates of the guard's location, then for each S-edge we produce one *guard-request* associated with the f-range and p-range of the edge.

In fact, there is one more case in which we need a guard-request; see Figure 4. While processing the N-edge e, a guard g gets positioned as shown to watch the rightmost S-edge. The same guard watches the S-edge e' which justifies the removal of the guard-request produced due to e'; however, if we do not do anything else, no need will be recorded for a guard to watch the orthogonal projection of e' onto a level slightly above the level of e. This clearly leads to an error in the case of Figure 4 (right).

Therefore, if the x-range of a N-edge e intersects but does not contain the f-range in a guard-request r of a S-edge e', a guard is needed to watch the edge e' as explained earlier. Additionally, we check whether this or any other guard watches the projection $o(e')$ of e' at a level slightly higher than e's and if not, we create a guard-request with f-range equal to the x-range of $o(e')$, i.e., the difference of the f-range of r minus the closure of the x-range of e. The p-range is as shown in Figure 4 since any guard at a level higher than e's with x-coordinate in that interval can watch the projection $o(e')$.

The above imply the following observation.

Observation 3.1 *At any time the f-ranges of the current guard-requests do not intersect except at their ends.*

Selecting a Guard to Watch an Edge

Many guards at different levels in the polygon may watch a S-edge e' when the f-range in the guard-request of e' is intersected by the x-range of a N-edge. In order to make a good choice among them, we apply the policy suggested in the following lemma.

Lemma 3.1 *Whenever a guard-request needs to be fulfilled, among all guards that can fulfill it, it suffices to choose the lowermost one.*

The proof takes advantage of Lemma 2.1(iii). Consider two guards g_1 and g_2 at levels ℓ_1 and ℓ_2, respectively, with $\ell_1 < \ell_2$. Recall that the vis-range of a guard is initialized to the x-range of the

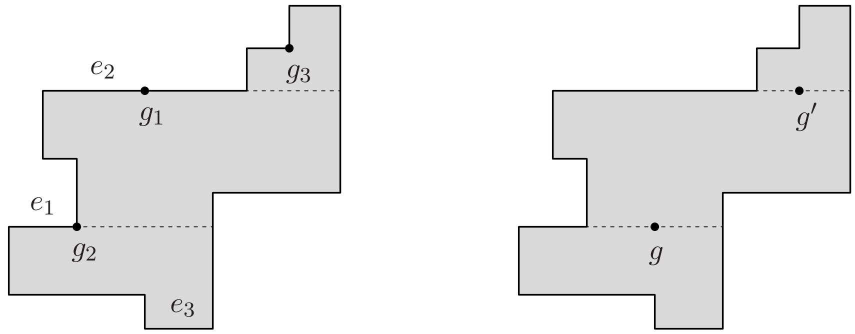

Figure 5: Not selecting the lowermost candidate guard may lead to a non-minimum number of guards.

grid segment at the level of the guard and is subsequently clipped by N-edges encountered; thus, at a level ℓ higher than its level, the guard can see all the points in the orthogonal projection of its initial vis-range onto the level ℓ. But then, in accordance with Lemma 2.1(iii), the set of points at level $\ell > \ell_2$ seen by guard g_1 is a subset of the set of points at level ℓ seen by g_2.

In fact, there are cases where by choosing a guard other than the lowermost available we get an incorrect result; see Figure 5. When encountering the N-edges e_1 and e_2, we realize that guards are needed at these levels. If when assigning a guard to watch the S-edge e_3, we select a guard at the level of e_2 (see guard g_1 in the polygon at left), then a third guard g_3 will also be needed; yet, two guards suffice to watch the entire polygon as shown at right.

Description of the Algorithm

As mentioned, we will sweep the given class-3 orthogonal polygon from bottom to top maintaining information on the current trousers (at the current location of the sweep-line). Since we need to be able to insert new trousers, to delete trousers, and to search the current trousers to locate the one incident on an edge (see Lemma 2.1(ii)), we maintain the trousers in a balanced binary search tree storing them in order from left to right. Along with each trouser T, we store the guard-requests associated with T (their f-ranges ordered from left to right in a doubly-linked list in light of Observation 3.1), and lists for the guard-requests' p-ranges and the guards' loc-ranges and vis-ranges. We also maintain two sets, $Available(T)$ and $Positioned(T)$, storing the guards in T whose loc-ranges are not a point and whose loc-ranges are a point, respectively.

During the sweeping, we stop at each horizontal edge e and process it. If e is a S-edge, we update the trouser information and set up and insert a corresponding guard-request. If e is a N-edge, we process the guard-requests whose f-ranges are intersected by e's x-range, conditionally setting up and inserting a guard-request of the 2nd kind, we position the guards whose loc-ranges are subsets of the closure of e's x-range, and we clip the p-ranges of all the guard requests and the loc- and vis-ranges of the guards. After all the edges have been processed, the resulting guard set *Positioned* gives us the locations of a minimum-cardinality set of r-visibility guards.

Next, we give a detailed description of the algorithm in pseudocode (the ranges of a guard g are denoted by g.loc-range and g.vis-range, the ranges of a guard-request r by r.f-range and r.p-range, and the x-range of an edge e by $e.x$-range).

Algorithm Class3_rStar_Cover(P)
Input : a simple class-3 orthogonal polygon P (no N-dents)
Output: a minimum set of r-visibility guards

1. sort the N- and S-edges of P by non-decreasing y-coordinate;
   create an empty data structure D_t to store the trousers;

2. {*sweep from bottom to top maintaining the trousers*}
   for each N- or S-edge e in order **do**
       if e is a S-edge
       then create the corresponding guard-request, say, r;
           locate e in the data structure of the trousers;
           if e does not belong to any of the current trousers
           then create a record for the new trouser T (containing only the edge e) and insert it in the data structure D_t;
               insert r in the guard-requests data structure of the trouser T;
               $Positioned(T) \leftarrow \emptyset$;   {*contains guards whose loc-ranges are points*}
               $Available(T) \leftarrow \emptyset$;   {*contains guards whose loc-ranges are not points*}
           else if e is a S-dent
           then   {*merge the two trousers T_1 and T_2 on either side of e*}
               remove the records of T_1 and T_2 from D_t and insert a new trouser T;
               merge the guard sets and guard-requests data structures associated with T_1 and T_2 (inserting r in the (merged) requests data structure), and associate them with T;
           else   {*e belongs to a single trouser T*}
               insert r in T's guard-requests data structure;
       else   {*e is a N-edge*}
           locate e in the data structure of the trousers and suppose that e is incident on the boundary of trouser T;

           {*process T's guard-requests whose f-ranges are "intersected" by e's x-range*}
           for each guard-request r in T such that r.f-range $\cap$ $e.x$-range $\neq \emptyset$ **do**
               {*e is the first edge whose x-range intersects r.f-range*}
               if $\exists$ guards $\in Available(T) \cup Positioned(T)$ whose loc-range is a subset of r.p-range
               then $g \leftarrow$ lowermost such guard;
               else **if** $\exists$ guards $\in Available(T)$ whose loc-ranges intersect r.p-range
                   then $g \leftarrow$ lowermost such guard;
                   else use a new guard g at e's level and insert it in $Available(T)$;
                       g.vis-range $\leftarrow$ x-range of grid segment at e's level;
               g.loc-range $\leftarrow$ g.loc-range $\cap$ r.p-range;

               $I \leftarrow r$.f-range - closure($e.x$-range);
               if $I \neq \emptyset$ **and** no guard $\in Available(T)$ sees the points with x-coordinates in I at a level higher than e's
               then create a new guard-request r';   {*2nd kind of guard-requests*}
                   r'.f-range $\leftarrow I$;   {*= r.f-range - closure(e.x-range)*}
                   r'.p-range $\leftarrow$ x.range of grid segment at the level of e;
                   insert guard-request r' in the requests data structure of T;
               remove r from the requests data structure of T;

{*process T's guards whose loc-ranges are "covered" by the x-range of e*}
for each guard g such that g.loc-range $\subseteq$ closure($e.x$-range) **do**
  $x_g \leftarrow x$-coordinate of left endpoint of g.loc-range;
  position g at (x_g, y_g) where y_g is the level of g;
  remove g from the set $Available(T)$ and insert it in the set $Positioned(T)$;

clip the loc-ranges and vis-ranges (whenever needed) of guards $\in Available(T)$;
clip the p-ranges (whenever needed) of the guard-requests of T;

3. report the locations of the guards in the resulting set *Positioned*.

The correctness of Algorithm Class3_rStar_Cover follows from Lemmas 2.1 and 3.1, and the discussion preceding the pseudocode.

Time and Space Complexity

Let n be the number of vertices of the given class-3 polygon. Then, the number of trousers is $O(n)$ and so are the number of guard-requests (at any given time we have at most 1 guard-request for each of the S-edges encountered – even in the case of the 2nd kind of guard requests) and the number of guards (note that by placing a guard on each N- and S-edge, we can watch the entire polygon).

As mentioned earlier, the trousers are stored in a balanced binary search tree; since the number of trousers is $O(n)$, then so is the size of the tree and thus every insertion, deletion, and search operation takes $O(\log n)$ time. Since the f-ranges of the guard-requests do not intersect except at their endpoints (Observation 3.1), we store them in a doubly-linked list (with pointers at both ends) which allows for traversal from both ends as well as for constant time merging. The sets *Available* and *Positioned* of guards can also be stored as lists with pointers at both ends so that they can be merged in constant time.

Step 1 of the algorithm clearly takes $O(n \log n)$ time. In Step 2, for each S-edge, we need to locate the edge with respect to the existing trousers, do at most one insertion and at most two deletions of trousers, and update the information stored in the corresponding trouser record(s); this takes $O(\log n)$ total time per edge ($O(\log n)$ time for the insertion, deletion, and search operations on the data structure D_t and constant time for the updates on the record(s) in D_t).

Let us now consider the processing of each N-edge e in Step 2. The location of the trouser T whose boundary e extends can be done in $O(\log n)$ time using the data structure D_t. Since e is either at the left or at the right end of the boundary of T, *exactly the useful* guard-requests (i.e., those whose f-ranges intersect $e.x$-range) can be found by traversing the guard-requests in their doubly-linked list either from left or from right, respectively (recall Lemma 2.1(ii)). For each such guard-request r, we spend $O(n)$ time to ensure that a guard watches the edge associated with r and to update the guard's ranges; r is removed and we will not handle it again. There exists *at most one* guard-request whose f-range is not a subset of the closure($e.x$-range); in such a case it takes $O(n)$ time to check if a guard-request of the 2nd kind is needed (see Figure 4) and constant time to set new request up and insert it in the guard-requests data structure of T. Processing each guard whose loc-range is a subset of the closure of $e.x$-range takes constant time per guard; this happens *exactly once* for each guard. Finally, clipping the ranges of the guards and the p-ranges of the guard requests (whenever needed) can be clearly done in $O(n)$ time. In summary, if we ignore the time to process the guard-requests whose f-range is a subset of the x-range of e and the time to process the guards whose loc-range is a subset of $e.x$-range, the processing of each N-edge e takes $O(n)$ time.

Thus, Step 2 takes $O(n^2)$ time since the f-range of a guard-request and the loc-range of a guard will be a subset of the x-range of a N-edge *exactly once*.

Since Step 3 takes $O(n)$ time, we have:

Theorem 3.1 *Let P be a simple class-3 orthogonal polygon with n vertices. Then, Algorithm Class3_rStar_Cover computes a minimum-cardinality set of r-visibility guards in $O(n^2)$ time and $O(n)$ space.*

We note that by using appropriate data structures we can in constant time find the lowermost guard for a guard-request and we can in constant time check whether there is a need for a guard-request of the 2nd kind. Unfortunately, clipping is more difficult to be sped up. What is needed is a data structure which primarily will do the clipping implicitly (explicit clipping yields $\Omega(n^2)$-time algorithms) and which will either be mergeable (since trousers get merged) or will allow implicit clipping of a connected subset of the intervals stored.

4 Concluding Remarks

In this paper we considered the problem of covering a class-3 orthogonal polygon with the minimum number of r-stars and described an $O(n^2)$-time algorithm where n is the number of vertices of the given polygon. Our algorithm paves the way to algorithms for the above problem on general simple orthogonal polygons that will be much faster than the $O(n^{17}\text{poly-}\log n)$-time algorithm of Worman and Keil [10]. Therefore, an immediate open question is to investigate how ideas from this work can be generalized to yield algorithms for the problem of r-star covering a general simple orthogonal polygon.

Another interesting open question is to try to obtain faster algorithms for the s-star covering problem on general simple orthogonal polygons; the current fastest algorithm requires $O(n^8)$ time [7] and is based on the graph-theoretic approach.

Finally, it would also be interesting to try to improve the time complexity of our algorithm. We believe that appropriate specialized data structures that will allow us to handle the clipping faster, will enable us to improve the time complexity to $O(n \log n)$.

References

[1] A. Aggarwal, *The Art Gallery Theorem: its Variations, Applications, and Algorithmic Aspects*, PhD Thesis, Department of Electrical Engineering and Computer Science, John Hopkins University, 1984

[2] J. Culberson and R.A. Reckhow, Orthogonally convex coverings of orthogonal polygons without holes, *J. Comput. Systems Science* 39(2), 166-204, 1989

[3] L. Gewali, M. Keil, and S.C. Ntafos, On covering orthogonal polygons with star-shaped polygons, *Information Sciences* 65, 45-63, 1992

[4] J. Kahn, M. Klawe, and D. Kleitman, Traditional galleries require fewer watchmen, *SIAM J. Algebraic Discrete Methods* 4(2), 194-206, 1983

[5] J.M. Keil, Minimally covering a horizontally convex orthogonal polygon, *Proc. 2nd Annual ACM Symp. Computational Geometry*, 43-51, 1986

[6] A. Lingas, A. Wasylewicz, and P. Żyliński, Note on covering orthogonal polygons with star-shaped polygons, *Information Processing Letters* 104(6), 220-227, 2007

[7] R. Motwani, A. Raghunathan, and H. Saran, Covering orthogonal polygons with star polygons: the perfect graph approach, *J. Comput. Systems Science* 40, 19-48, 1990

[8] J. O'Rourke, *Art Gallery Theorems and Algorithms*, Oxford University Press, 1987

[9] J. Urrutia, Art gallery and illumination problems, *Handbook of Computational Geometry*, Elsevier Science, Amsterdam, 973-1027, 2000

[10] C. Worman and J.M. Keil, Polygon decomposition and the orthogonal art gallery problem, *International Journal of Computational Geometry & Applications* 17(2), 105-138, 2007