A Faster Algorithm for 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 are interested in the problem of covering simple orthogonal polygons with the minimum number of r-stars. An orthogonal polygon P is an r-star if P is (orthogonally) convex and star-shaped. The problem of covering a simple orthogonal polygon with the minimum number of r-stars has been considered by Worman and Keil [13] 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 provide an $O(n \log n)$-time algorithm; this is the first purely geometric algorithm for this problem. Moreover, ideas in our algorithm may be generalized to yield exact algorithms for this problem that are faster than Worman and Keil's.

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

1 Introduction

Motivated by a question of Klee in 1973 and thanks to work of Chvátal and Fisk (see [11]), 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 guard [11].

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 [11, 12]. 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 (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).

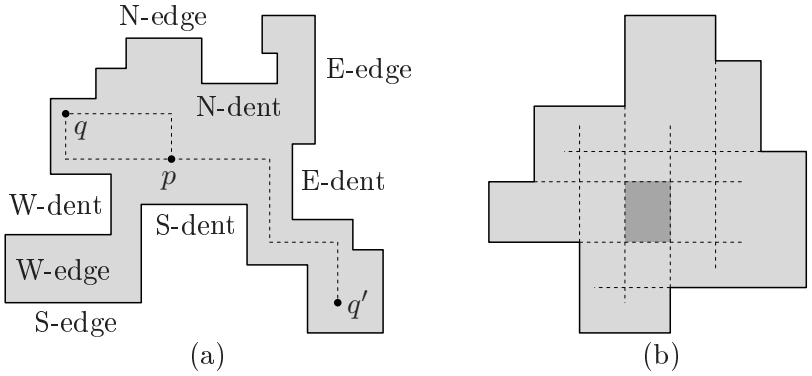

Figure 1: (a) Illustration of the main definitions; (b) an r-star with its kernel shaded.

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.

An orthogonal polygon is an *r-star* if it is (orthogonally) convex and star-shaped. The term r-star comes from its formal definition with respect to the *r-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 (Figure 1(a) shows two such points p and q); then, a polygon P is an *r-star* if there exists a point p of P such that every point $q \in P$ is r-visible from p while the set of all such points p in P is called the *kernel* of the r-star. Figure 1(b) shows an r-star with its kernel shaded. (For completeness, we mention that in orthogonal polygons another type of visibility, the *s-visibility*, is defined: two points p, q of an orthogonal polygon P 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) points p and q' are s-visible from one another.)

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 cover of the polygon by r-stars (or s-stars, respectively). A *cover* of a polygon P by 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 cover of the polygon; thus, a minimum-size cover 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 covers 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 (e.g., finding a minimum-size Steiner-free partition of a simple polygon into star-shaped polygons is known to be computable in polynomial time [5], whereas the corresponding covering problem is NP-complete [1]). Covers and partitions are very important as they can be used for decomposition into simpler pieces. Recent applications of

rectangulations include planar self-assembly with local information [7] and DNA self-assembly (M.Y. Kao and A. Sterling).

Covering by r-stars has been investigated early enough. Keil [6] 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 [10] studied s-star covers. They showed a close relation between minimum-size covers of orthogonal polygons by s-stars and covers 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 [8] 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 [13]. Very recently, a linear-time 3-approximation algorithm for general orthogonal polygons has been given by Lingas, Wasylewicz, and Żyliński [9].

In this paper, we study the r-star covering problem on class-3 orthogonal polygons. We take advantage of geometric properties of these polygons and we describe an $O(n \log n)$-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. Moreover, ideas in our algorithm may be generalized to yield exact algorithms for this problem 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 in a cartesian coordinate system. 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 "grid segment" and "level" used in [3]: a *grid segment* of P or a trouser T is a maximal (closed) horizontal line segment in P or T; the *level* of a point or a horizontal line segment (which may be a grid segment or a horizontal edge) is its y-coordinate. We also use the notion of orthogonal projection in an orthogonal polygon P given in [8]: the *orthogonal projection* $o(s)$ of a horizontal line segment s at level ℓ in P onto the grid segment s' at level $\ell' \geq \ell$ is the maximal subsegment of s' such that for each point a of $o(s)$ there exists a vertical line segment in P that goes through a and intersects s. Finally, for a horizontal line segment s (edge or grid segment) we define its *x-range* to be the set of x-coordinates of the points of s. (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 their x-ranges are open sets as well.)

The following lemma provides three 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:*

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

(ii) Consider sweeping the polygon from bottom to top. Each edge encountered other than the bottommost edge of each trouser is incident with the boundary of the swept polygon.

(iii) Let T be a trouser at the moment when P is intersected by a horizontal line at level ℓ, and let s_1 and s_2 be grid segments of T at levels ℓ_1 and ℓ_2, respectively, where $\ell_1 < \ell_2 \leq \ell$, such that there exists a vertical line segment in T intersecting both s_1 and s_2. Then, the orthogonal projection of s_1 onto ℓ is a subset of the orthogonal projection of s_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 s_1 onto level ℓ_2 is a subset of s_2 taking into account that $\ell_1, \ell_2 \leq \ell$. ∎

3 The Algorithm

Our algorithm applies plane-sweeping as do the algorithms in [3, 8]; we assume that the given class-3 polygon does not have N-dents and we sweep it from bottom to top stopping at each horizontal edge (thus we can take advantage of Lemma 2.1). The invariant that we maintain is that at any given time, the guards that have been placed watch all points of the swept polygon that cannot be watched by a guard located at a point above the sweep-line at its current position. In particular, at any S-edge we do some preparatory work but do not place guards as such edges can be watched by guards located at a higher level. N-edges may "cover" parts of the polygon from guards positioned higher; we check this and only if a guard is needed, it is located at the level of the N-edge (the x-coordinate of its location may not be set at the moment as we place guards so that they can see as much of the polygon above them as possible –details are given below). In the end, the algorithm reports the locations of a minimum-size set of r-visibility guards that watch the entire input polygon.

Determining When a Guard is Needed and Where to be Placed
Consider any S-edge e of the given polygon; see Figure 2(a). As long as the x-ranges of the encountered N-edges do not intersect the x-range of e, then a guard at a level higher than the level of the N-edge can see the entire e; see the N-edge e_1 in Figure 2(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; see the N-edge e_2 in Figure 2(a). Additionally, if such a guard is to be placed at level ℓ, it has to be placed at any point of *the orthogonal projection of the grid segment containing e onto level ℓ*, in order to watch e.

Therefore, in order to enforce the above observations, each S-edge e submits a *type-1 guard-request* with which we maintain:

- ▷ a *forcing-range*, or *f-range* for short, which is the x-range of the edge e (because a guard is needed to watch e if the x-range of a N-edge above e intersects e's f-range);
- ▷ a *placement-range*, or *p-range* for short, which is the range of x-coordinates of the grid segment containing e (because this is the initial range of x-coordinates of the guard's location).

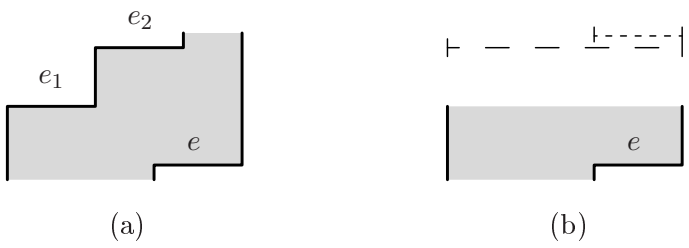

Figure 2: (a) A guard needs to be placed no higher than the N-edge e_2 to watch the entire S-edge e; (b) the f-range (shown dotted) and the p-range (shown dashed) of the S-edge e.

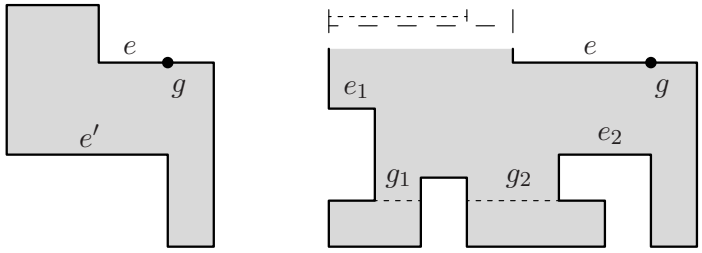

Figure 3: Type-2 guard-requests (f-range shown dotted, p-range shown dashed).

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 2(b) shows the f-range (shown dotted) and p-range (shown dashed) for the S-edge e.

In fact, there is one more case in which we need a guard-request. See Figure 3 (left). While processing the N-edge e, a guard g gets positioned as shown to watch the lowermost 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 3 (left) as no guard other than g is placed.

Therefore, at each N-edge e (of a trouser T), we investigate the need to place a *type-2 guard-request*. Let I be the grid-segment of T at a level slightly above e's level. If the entire I is watched, no guard-request is needed. Otherwise, a guard-request r is submitted with p-range equal to I and f-range equal to (x_l, x_r) where x_l (x_r, resp.) is the x-coordinate of the leftmost (rightmost, resp.) point in I not watched by any of the currently placed guards (see Figure 3 (right)).

Here is how the f- and p-range of a guard-request r submitted by an 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 r, then the p-range simply gets clipped. However, if we encounter a N-edge d whose x-range intersects the f-range of r, then a guard is needed immediately; any guard located at a level between (and including) the levels of e and d, which can be positioned at a point with x-coordinate in the p-range of r will do.

Maintaining and Processing Guards

In order to be able to manage the guards, with each guard we maintain:

- its *level* (i.e., the y-coordinate of its location),
- its *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;

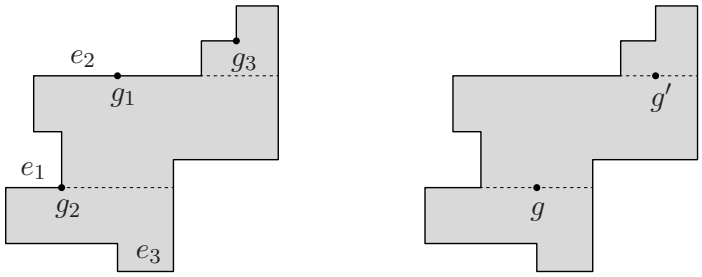

Figure 4: Not selecting the lowermost candidate guard may lead to a non-minimum number of guards.

- its *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 r-visible to the guard.

Since there are no N-dents, each of these ranges is a *single interval* of x-coordinates, and 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 a grid segment s_ℓ at level ℓ in a trouser T, initially its loc-range and its vis-ranges coincide with the x-range of s_ℓ. As the sweep-line moves upward, both ranges get clipped by N-edges whose x-ranges intersect them. If g is chosen to fulfill a guard-request r (then g's loc-range must intersect r's p-range), the loc-range of g is set equal to its intersection with the p-range of r; in this way, the guard will be able to watch both the edge that submitted r and as much of the *unseen* polygon as possible. Finally, when a N-edge d is encountered such that the (possibly clipped) loc-range of g is a subset of d's x-range, then g is placed at the point (x_l, ℓ) where x_l is the left bound value of g's (clipped) loc-range right before the N-edge d is encountered (in accordance with the convention followed by [3, 8]); moreover, g cannot see any points in the polygon P above the level of the edge d.

Selecting a Guard to Watch a S-Edge
Many guards at different levels in the polygon may be able to watch a S-edge e' when the f-range in the guard-request submitted by 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 *There exists a minimum-size set of guards such that whenever a guard-request needs to be fulfilled, among all guards that can fulfill it, the lowermost one is chosen.*

In other words, among the guards fulfilling the guard-request, we choose a guard g that has the smallest vis-range, saving guards with larger vis-ranges to possibly watch portions of the polygon that g cannot see. The proof of Lemma 3.1 relies on Lemma 2.1(iii). Recall that the vis-range of a guard at a level ℓ is initialized to the x-range of the grid segment at level ℓ and is subsequently clipped by N-edges encountered; thus, at a level $\ell' > \ell$, the guard can see all the points in the orthogonal projection of its initial vis-range onto ℓ'.

In fact, there are cases where by choosing a guard other than the lowermost available we get an incorrect result; see Figure 4. 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 Figure 4 (left), then a third guard g_3 will also be needed; yet, two guards suffice to watch the entire polygon as shown at Figure 4 (right).

Description of the Algorithm
As mentioned, we sweep the given class-3 orthogonal polygon P from bottom to top maintaining information on the current trousers (at the current position of the sweep-line), and the ranges

of the guards and of the guard-requests. With each trouser T, we also maintain T's guards partitioned into two sets, $Available(T)$ and $Positioned(T)$, storing the guards in T that can watch points in P above the current position of the sweep-line or not, 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 type-1 guard-request. If e is a N-edge, we process the guard-requests whose f-ranges are intersected by e's x-range, position the guards whose loc-ranges are subsets of e's x-range, clip the guard-requests' p-ranges and the guards' loc- and vis-ranges, and conditionally set up and insert a type-2 guard-request. 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.

Below, we give a detailed description of the algorithm in pseudocode when applied on a class-3 orthogonal polygon P (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 type-1 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 (involving only e) and insert it in the data structure D_t;
   insert r in T's guard-requests data structure;
   $Available(T) \leftarrow \emptyset$; $Positioned(T) \leftarrow \emptyset$;
   else if e is a S-dent
   then {*merge the two trousers T_1 and T_2 on either side of e*}
   remove 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 and associate them with T;
   insert r in the (merged) requests data structure;
   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 D_t of the trousers and let T be the trouser whose boundary is incident with e;
   {*process T's guard-requests whose f-ranges intersect e's x-range*}
   for each guard-request r in T s.t. r.f-range $\cap$ $e.x$-range $\neq \emptyset$ **do**
   {*r.f-range not intersected before by x-range of a N-edge*}
   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;
g.loc-range $\leftarrow$ g.loc-range $\cap$ r.p-range;
else use a new guard g and insert it in $Available(T)$;
g.level $\leftarrow$ level of e;
g.vis-range $\leftarrow$ x-range of the grid segment of T containing e;
g.loc-range $\leftarrow$ r.p-range;
remove r from T's guard-requests data structure;

{*process T's guards whose loc-ranges are "covered" by e*}
for each guard g such that g.loc-range $\subseteq$ $e.x$-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 $Available(T)$ and insert it in $Positioned(T)$;

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

{*check if a type-2 guard-request is needed*}
$I \leftarrow$ x-range of closure((grid segment at the level of e) $- e$);
if I is not entirely watched
then create a new guard-request r'; {*type-2 guard-request*}
r'.p-range $\leftarrow I$;
$x_l \leftarrow$ x-coordinate of leftmost non-watched point in I;
$x_r \leftarrow$ x-coordinate of rightmost non-watched point in I;
r'.f-range $\leftarrow (x_l, x_r)$;
insert r' in the guard-requests data structure of T;

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

The correctness of the algorithm follows from the fact that we use a new guard only when we have located a portion of the polygon that is not watched by any of the currently used guards and by any guard above the sweep-line at its current position, from Lemmas 2.1 and 3.1, and from the preceding discussion.

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 is the number of guard-requests (we have at most 1 guard-request for each of the S-edges (type-1 request) and each of the N-edges (type-2 request) encountered), and the number of guards (note that by placing a guard on each N- and S-edge, we can watch the entire polygon).

Data Structures. Let us now discuss the data structures used. Since we need to be able to insert new trousers, to delete trousers, and to search the current trousers to locate the one incident with an edge (see Lemma 2.1(ii)), we maintain the trousers in a balanced binary search tree D_t storing them in order from left to right; then every insertion, deletion, and search operation takes $O(\log n)$ time.

Each of the guard sets $Available(T)$ and $Positioned(T)$ associated with a trouser T is stored with T in a doubly-linked list with pointers at both ends so that insertion, deletion, and list concatenation can be done in constant time.

In order to store the f-ranges of all the guard-requests (we do not distinguish them depending on the trouser to which they belong since the f-ranges of guard-requests from different trousers

do not overlap), we use two threaded balanced binary search trees T_l and T_r storing the f-ranges in their leaves (since the trees are threaded, their leaves are linked in order from left to right in the tree) with each pair of corresponding leaves in T_l and T_r linked to each other: T_l (T_r, resp.) stores the f-ranges in increasing order of their left (right, resp.) endpoint and in case of ties in decreasing order of the level of the edge that caused the guard-request. The size of each tree is again $O(n)$ and thus, inserting and deleting an f-range in both trees can be done in $O(\log n)$ time. In order to find all the f-ranges intersected by the x-range of a N-edge e, we work as follows: if e has its left endpoint on the boundary of a trouser defined by the sweep-line at its current position (see Lemma 2.1(ii)), we use T_l to locate all the f-ranges, if any, with their left endpoint identical to e's left endpoint, and move rightward from leaf to leaf using the thread pointers until all the f-ranges intersecting e's x-range are located; if e has its right endpoint on the boundary of a trouser, we work similarly with T_r. In summary, this can be done in $O(t + \log n)$ time where t is the number of f-ranges accessed (and which are deleted).

Clipping on the guards' vis-ranges is done in an implicit way; thus, the vis-ranges are stored in a special doubly-linked list as shown in Figure 5. Each node corresponds to a vertical edge (which defined the endpoint of a vis-range or which clipped a previously defined vis-range) and stores the x-coordinate of that edge and a y-ordered sublist of vis-ranges (with pointers at both ends) ending at that vertical edge; the levels of the vis-ranges stored at the sublists of two nodes enable us to compare them along the y-axis. If the clipping affects only the first or last node in the list, then we simply update the x-coordinate stored in the node in $O(1)$ time. If the clipping affects more nodes, then their sublists are concatenated (maintaining their y-ordering) and again the x-coordinate stored in the first or last node of the resulting list gets updated; the $O(1)$-time concatenation of the sublists of two consecutive nodes t_1, t_2 is *charged* to the horizontal edge incident on the top endpoint of the vertical edge corresponding to the lowermost node between t_1 and t_2 (the sublist of the lowermost node gets linked to the sublist of the other one).

A similar data structure is used to store the guard-requests' p-ranges together with the guards' loc-ranges; all these are linked together in y-ordered sublists which also have extra pointers doubly-linking only the p-ranges. Clipping is done as above. Getting the lowermost guard to fulfill a guard-request r involves getting to the sublist node for r (through pointers from T_l and T_r) in the first or last node of the main list and then moving upwards in the sublist until a guard's loc-range is found; if a guard is found, then all the traversed guard-requests will be fulfilled by that guard and they are removed (we may remove some guard-requests whose p-ranges are not intersected by the x-range of the currently processed N-edge but this does not cause an error), whereas if no guard is found then a new is used who again fulfills all the traversed guard-requests. Assigning the loc-range of the guard to the p-range of the guard-request is done by using the representation of the guard-request for the guard's loc-range and updating the information and pointers for the p-ranges linking.

Complexity. Sorting the N- and S-edges (by y-coordinate) takes $O(n \log n)$ time. Then, for each S-edge e, we need to locate e with respect to the existing trousers in D_t, do at most one insertion and at most two deletions of trousers (in $O(\log n)$ time), and update the information stored in the corresponding trouser (in $O(1)$ time).

Let us now consider the processing of each N-edge e. Locating *all* the $O(n)$ N-edges in D_t takes $O(n \log n)$ time. Processing *all* the guard-requests whose f-ranges intersect the x-ranges of all the N-edges requires $O(n \log n)$ time for searching T_l, T_r, and $O(n)$ for the deletion of guard-requests. Processing *all* the guards whose loc-ranges are covered by N-edges is done using the loc-ranges list and takes $O(1)$ time per guard since the guard-requests in the same node have already been processed and removed. Clipping is done in $O(n)$ time in total, since the clipping vertical edge is charged for the $O(1)$-time information updates while a different horizontal edge is charged for each $O(1)$-time sublist concatenation. Handling all type-2 guard-requests takes

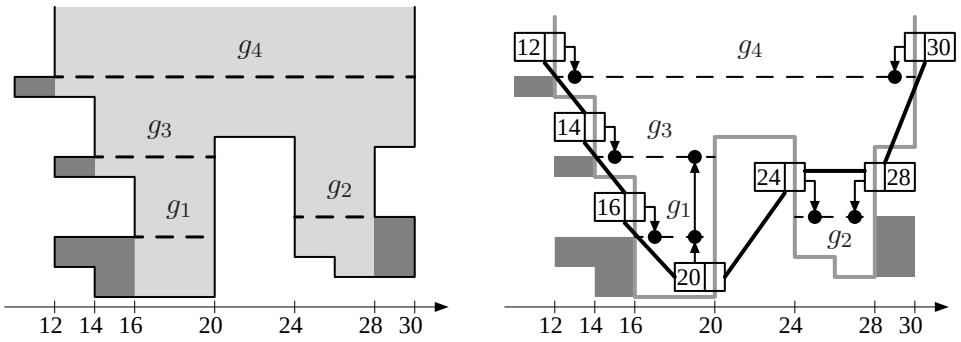

Figure 5: The data structure for the guards' vis-ranges (rectangles connected by thick lines indicate nodes of the main doubly-linked list, dark circles indicate sublist nodes).

$O(n \log n)$ time; for each such request r', locating the leftmost and rightmost non-watched points can be done in $O(1)$ time using the vis-ranges data structure, inserting and linking r' in the p-ranges (and loc-ranges) data structure is done in $O(1)$ time as well, whereas inserting r' in T_l and T_r takes $O(\log n)$ time. In summary, processing all the N-edges takes $O(n \log n)$ time.

Since reporting the guards takes $O(n)$ time, we have:

Theorem 3.1 *Let P be a simple class-3 orthogonal polygon with n vertices. Then, a minimum-cardinality set of r-visibility guards watching the entire P can be computed in $O(n \log n)$ time and $O(n)$ space.*

4 Concluding Remarks

We presented an $O(n \log n)$-time algorithm for computing a minimum r-star cover of a class-3 orthogonal polygon on n vertices. It would be interesting to make our algorithm output-sensitive. We believe that a more conservative policy on collecting guard-requests will help improve the time complexity to $O(n + k \log k)$ where k is the minimum number of guards needed to watch the given class-3 orthogonal polygon.

We leave as open problems the following on minimum r-star covers: obtaining faster algorithms for general simple orthogonal polygons compared to the algorithm of Worman and Keil [13], investigating the complexity of the problem on orthogonal polygons with holes, and studying extensions of the problem in three dimensions.

Finally, it would also be interesting to obtain faster algorithms for the s-star covering problem on general simple orthogonal polygons; the current fastest algorithm takes $O(n^8)$ time [10] and is based on the graph-theoretic approach.

References

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

[2] J. Culberson and R.A. Reckhow, Orthogonally convex coverings of orthogonal polygons without holes, *J. Comput. Syst. Sci.* 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, Decomposing a polygon into simpler components, *SIAM J. Computing* 14, 799-817, 1985

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

[7] G. Li and H. Zhang, A rectangular partition algorithm for planar self-assembly, *Proc. IEEE/RSJ Int. Conference on Intelligent Robots and Systems*, 3213-3218, 2005

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

[9] A. Lingas, A. Wasylewicz, and P. Żyliński, Linear-time 3-approximation algorithm for the r-star covering problem, *International Journal of Computational Geometry & Applications* 22(2), 103-141, 2012

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

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

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

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