Shortest m-Watchmen Routes for Histograms: The MinMax Case*

Bengt J. Nilsson† Sven Schuierer‡

Abstract

In this paper we consider the problem of computing an optimum set of watchmen routes in a histogram. A watchman, in the terminology of art galleries, is a mobile guard and in this version we want to minimize the length of the longest route in the solution. We give an $O(n^2 \log n)$ time algorithm to compute the MinMax optimum set of m watchmen in a histogram polygon and we extend the algorithm to solve the Weighted MinMax problem under the same time bound.

1 Introduction

The problem of placing guards in an art gallery so that every point in the gallery is visible to at least one guard has been considered by several researchers. If the gallery is represented by a polygon (having n vertices) and the guards are points in the polygon, then visibility problems can be equivalently stated as problems of covering the gallery with starshaped polygons. Chvàtal [5] and Fisk [7] proved that $\lfloor n/3 \rfloor$ guards are always sufficient and sometimes necessary. Many other results on art gallery problems and general visibility problems can be found in [10].

Chin and Ntafos [3, 4] consider the problem of finding shortest watchman routes for different classes of polygons. The route is the shortest closed curve traced by a moving guard who (in total) sees the whole polygon. They prove that the problem is NP-hard for polygons with holes and for 3-dimensional simple polyhedra. They also give a $\Theta(n)$-time algorithm that solves the problem for simple rectilinear polygons. In addition they provide a solution to the slightly different question of finding a shortest watchman route given a "door", a point on the boundary that the route must pass through. They give an $O(n^4)$-time algorithm that solves this problem for simple polygons without holes. (These bounds rely on the new linear time triangulation algorithm by Chazelle [2].)

The stationary art gallery problem and the watchman route problem can be thought of as special cases of the *m-watchmen routes problem.* Suppose that we have m mobile watchmen and we want to design routes for them so that each point in a polygon is visible to some point along a route. The objective is to minimize the total length of the m routes, the MinSum criterion. Clearly, the total length of an optimum solution increases (decreases) as the number of watchmen decreases (increases). At one limit, we have a situation where m is large enough so that the length of the optimum solution is zero (i.e., each watchman remains stationary) and we have the art gallery problem. At the other limit, if $m = 1$, we have the watchman route problem.

Carlsson *et al.* [1] prove that the m-watchmen routes problem is NP-hard but give a $\Theta(n^3 + n^2m^2)$ time algorithm for histogram polygons. Nilsson and Wood [9] treat the class of spiral polygons. They give a $\Theta(n^2)$ time algorithm for the m-watchmen routes problem.

*This work was supported by the Deutsche Forschungsgemeinschaft under Grant No. Ot 64/5–4.

†Department of Computer Science, Lund University, Box 118, 221 00 Lund, Sweden.

‡Institut für Informatik, Universität Freiburg, Rheinstr. 10–12, D-7800 Freiburg, Fed. Rep. of Germany.

Mitchell and Wynters [8] also consider multiple watchmen routes but use a different length criterion, the MinMax criterion, where the length of the longest watchman route is to be minimized. They prove that under rectangular visibility it is possible to find the optimum set of m watchmen in a rectilinear monotone polygon in $O(n^4 m)$ time. Additionally, they prove that, in general, already the 2-watchmen routes problem under the MinMax criterion is NP-hard. No such result exists for the MinSum criterion.

It turns out that a simple modification of the algorithm by Nilsson and Wood is sufficient to solve the MinMax version of the m-watchmen routes problem in spiral polygons. In this paper we consider the MinMax version in histogram polygons. Note that the result by Mitchell and Wynters does not immediately carry over since they consider rectangular visibility.

In the next section we give some definitions and state some properties of histogram polygons. In Section 3 we present an algorithm to compute a MinMax optimum set of m watchmen routes in $O(n^2 \log n)$ time and quadratic storage and in Section 4 we extend the algorithm to solve a weighted version of the problem in the same time and space bounds.

2 Definitions and Preliminary Results

We begin the section by defining the class of histogram polygons.

Definition 2.1 A *histogram* is a simple rectilinear polygon which is monotone with respect to the horizontal axis, with the additional property that one of the monotone chains forming its boundary is a single straight line segment.

We will henceforth assume that the lower monotone chain is a straight line segment; see Figure 1, and call this segment the *base* of the histogram. Edelsbrunner *et al.* [6] define a vertical histogram as an orthogonal polygon with a horizontal edge (the base) equal in length to the sum of the lengths of the other horizontal edges.

Note that the monotone chain opposite of the base consists of a sequence of pairs of staircase patterns, one up staircase followed by a down staircase. We call the part of the polygon under such a pair a *pyramid*. Sack [11] defines a pyramid to be horizontally convex vertical histogram. (Horizontal convexity is the same as monotonicity along a vertical line.)

The two staircases are connected by the *top edge*; see Figure 1. We will use the term *west edge* to denote a vertical edge that has the interior to the right of the edge. An *east edge* is defined similarly as a vertical edge having the interior to the left. Note that a pyramid is a starshaped polygon with the kernel under the top edge.

We define the *supporting line* of a point s, on the base, and a pyramid $\triangle$ to be the line that intersects s, at least one vertex of a staircase of $\triangle$, and either the top edge or the other staircase. Note that the supporting line divides the pyramid into one section which is visible from s and another section where there may exist points that are not seen from s, since the supporting line passes through a vertex. Should s lie below the top edge of the pyramid, we define the supporting line as the vertical line intersecting s.

First, we state some simple properties of histograms. The proofs of Lemmas 2.1–2.3 are given in [1].

Lemma 2.1 *If s and t are two points on the base of a histogram, s being to the right of t, then s sees at least as much of the part of the polygon that lies to the right of s as t does.*

We define the function *partner*$(s, \triangle)$ of a point s on the base and a pyramid $\triangle$ to be the furthest point from s, on the base, where s and the partner collectively see all of $\triangle$. If s already sees all of $\triangle$ the partner is undefined.

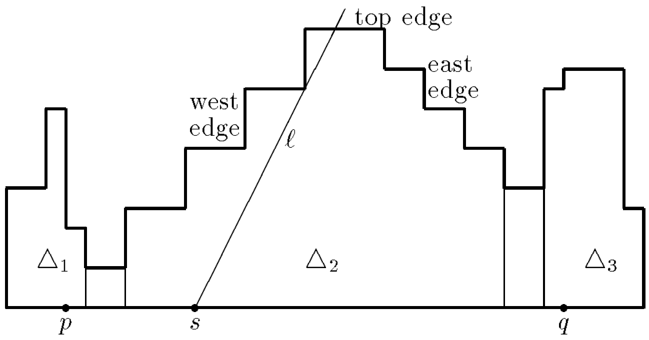

Figure 1: Histogram with three pyramids. ℓ is the supporting line of point s with respect to $\triangle_2$.

The following lemma shows how to compute the partner (when defined) of a given point with respect to some pyramid.

Lemma 2.2 *If s is a point not under the top edge of a pyramid $\triangle$, then one of the following holds:*

1. *The supporting line of $partner(s, \triangle)$ (w.r.t. $\triangle$) intersects the supporting line of s on the top edge of $\triangle$, or*

2. *$partner(s, \triangle)$ lies under the top edge of $\triangle$.*

Given a point s on the base and a pyramid $\triangle$ the supporting line can be easily computed. First determine whether s lies to the left or right of the top edge of $\triangle$. Assume it lies to the left of $\triangle$, then follow the left staircase down, maintaining the line through s and the vertex while remembering the line having minimum slope.

The point $partner(s, \triangle)$ can be computed by taking the intersection point of the supporting line of s and the top edge of $\triangle$, then by following the right staircase down, maintaining the line through the intersection point and the current point of the right staircase. The line having maximum slope is the supporting line of the partner, and hence the partner is the intersection of this supporting line with the base. If the supporting line of s does not intersect the top edge of $\triangle$, the partner is the point on the base under leftmost east edge of $\triangle$.

If s lies to the right of $\triangle$, a similar scheme can be used to compute the supporting line and the partner.

We define $rlp(s)$, the *limit point to the right of s* to be the closest partner to the right of s. More formally

$$rlp(s) = \min_{s \preceq \triangle_i} (partner(s, \triangle_i)),$$

where the value of a point is its distance from the leftmost edge of the histogram; $\triangle_i$, with $1 \leq i \leq k$, are the pyramids of the histogram; and $\preceq$ means "not to the right of".

Next we claim that the limit point to the right of a given point s is always the furthest point on the base which retains visibility between the two points.

Lemma 2.3 *If s is a point on the base of a histogram, then there does not exist any point to the right of $rlp(s)$, on the base, which together with s sees all of the top monotone chain between the two points.*

The right limit point of s can be computed in linear time by scanning the pyramids having their top edges to the right of s. This is done from left to right. For each pyramid the partner to s is computed and the leftmost partner found so far is remembered. The algorithm stops when the top edge of the next pyramid to be scanned is to the right of the leftmost partner found.

3 MinMax Routes in Histograms

We define a *route* to be a closed curve interior to the polygon and a *set of m watchmen routes* to be m routes so that for every point in the polygon there is a point on a route such that the two points see each other. An *optimum set of m watchmen routes* is a set of m watchmen routes such that the length of the longest route is minimal. We denote the length of a route r by $\lambda(r)$ and if $\mathcal{R}$ is a set of routes where the length of the longest route is ℓ, we say that the length of $\mathcal{R}$ is $\lambda(\mathcal{R}) = \ell$.

We first prove that an optimum set of watchmen routes always lies on the base of the histogram.

LEMMA 3.1 *For any histogram, a MinMax optimum set of watchmen lies on the base of the histogram between the leftmost east edge and the rightmost west edge.*

PROOF: Take any solution $\mathcal{W}$ and project it orthogonally onto the base giving a set of routes $\mathcal{W}'$. Let point x be visible from some original watchman w. We need to show that the projected route w' also sees x. Since w sees x, there is a point s_w on w for which the segment between x and s_w is interior to the polygon. Project the points s_w and x orthogonally onto the base giving the points s'_w and x' respectively. The four points s'_w, x', x, and s_w define a convex area (trapezoid) and, hence, s'_w sees x. Since s'_w is a point on w', this watchman sees x. Also, since we made an orthogonal projection, $\lambda(w') \leq \lambda(w)$ for all the watchmen and it follows that $\lambda(\mathcal{W}') \leq \lambda(\mathcal{W})$.

Any watchman having points to the left of the leftmost east edge can have these points removed so that the leftmost point lies under the leftmost east edge. Similarly, the rightmost watchman can be made to stop under the rightmost west edge and this can never increase the size of the maximum length watchman. □

This proves that the optimum set of m watchmen routes consists of m disjoint line segments on the base. Note that the length of each route is twice the length of the line segment it consists of since a guard that moves along the route must go from one end point to the other and back again.

The points on the base under the leftmost east edge and the rightmost west edge are denoted p and q respectively; see Figure 1. Note also that any set of watchmen routes has to span the interval between p and q.

Consider a sequence of m routes (not necessarily watchmen routes) on the base $\mathcal{R} = \{r_1, \ldots, r_m\}$, $r_i = [a_i, b_i]$ all having length ℓ (i.e., $|r_i| = \ell/2$) and such that $a_1 = p$, $a_{i+1} = rlp(b_i)$, and $b_m \preceq q$. We call such a sequence the *longest spanning collection of routes having length ℓ.*

LEMMA 3.2 *Let a_i be the left end point and b_i be the right end point of the i^{th} route in the longest spanning collection of routes having length ℓ. Take any collection of routes having length at most ℓ and let a'_i be the left end point and b'_i be the right end point of the i^{th} route in this collection. We have that $a'_i \preceq a_i$ and $b'_i \preceq b_i$ for all i such that $1 \leq i \leq m$.*

PROOF: Consider a collection of m routes $\{r'_1, \ldots, r'_m\}$, $r'_i = [a'_i, b'_i]$ with $a'_1 \preceq p$ and $b'_m \preceq q$. Let ℓ be the length of the longest route in the collection. Construct a longest spanning collection of m routes $\{r_1, \ldots, r_m\}$, $r_i = [a_i, b_i]$ having length ℓ. Because of the visibility requirements it is

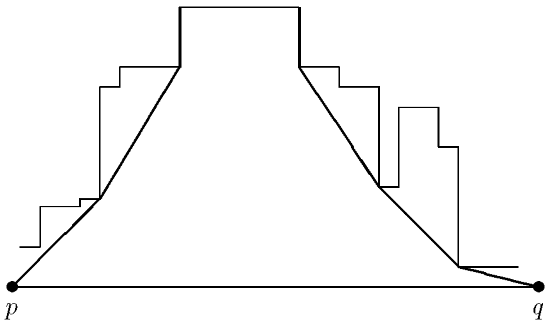

Figure 2: Pyramid with concavity chains.

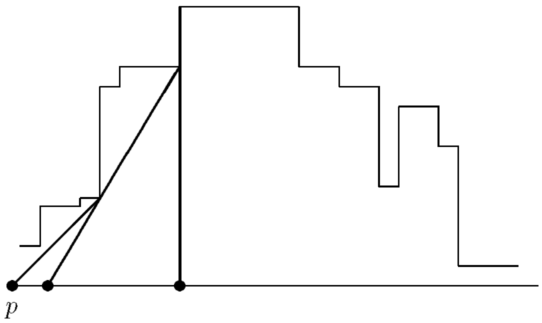

Figure 3: Primary event points from the left concavity chain.

clear that $a_1' \preceq a_1$ and, hence, $b_1' \preceq b_1$. By induction and Lemma 2.3 it follows that $a_i' \preceq a_i$ and $b_i' \preceq b_i$. □

The lemma implies that we can restrict ourselves to computing longest spanning routes.

In [1] it is proven that $rlp(s+\varepsilon) = rlp(s) + c\varepsilon$, i.e., the gap between a point s and its right limit point increases or decreases as a linear function within a neighbourhood of s. The value c is called the *sliding constant* of s. We want to find the set of points on the base where c changes its value. In [1] a subset of these points, called the *event points,* is used to find the MinSum optimum set of m watchmen. The definition of these points is based on the notion of concavity chains.

DEFINITION 3.1 The *concavity chains* of a pyramid $\triangle$ are the two reflex chains, one starting at the point p on the base and ending at the left end point of the top edge of $\triangle$ and the other starting at the point q and ending at the right end point of the top edge, such that each chain is the shortest path between the two points; see Figure 2.

There are two kinds of event points. The *primary* and the *secondary* event points. The primary event points of a pyramid $\triangle$ are:

1. The intersection points of the base and the extended links of the left concavity chain that also intersect the top edge of $\triangle$; see Figure 3, and

2. the points $partner(s, \triangle)$ where s is an intersection point of the base and one of the extended links of the right concavity chain that also intersects the top edge of $\triangle$; see Figure 4.

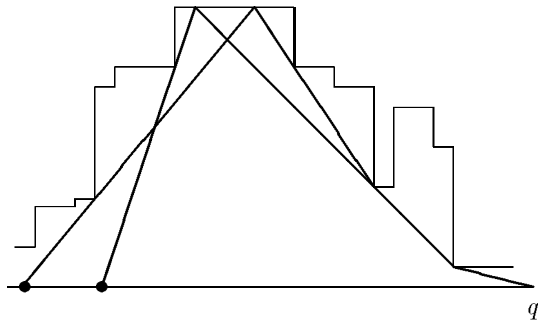

Figure 4: Primary event points from the right concavity chain.

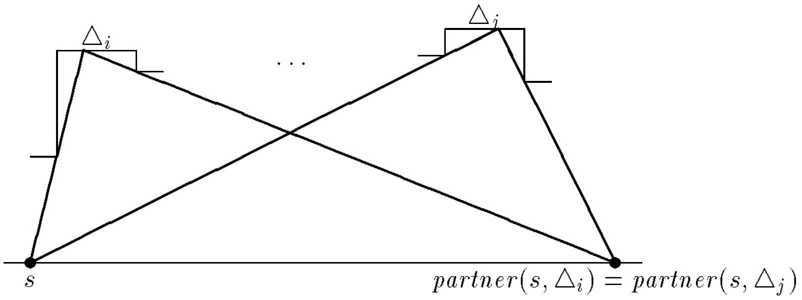

Figure 5: Secondary event points define local extrema of the distance function between a point and its partner.

The secondary event points are all points s on the base such that $partner(s, \triangle_i) = partner(s, \triangle_j)$ for any two pyramids $\triangle_i$ and $\triangle_j$, s lies to the left of $partner(s, \triangle_i)$, and such that $rlp(s) = partner(s, \triangle_i)$; see Figure 5.

The number of event points is bounded by $\Theta(n^2)$ and a superset, denoted S, can be computed in $O(n^2)$ time as in [1]. The superset contains the primary event points, the set S_1, and the points s such that $partner(s, \triangle_i) = partner(s, \triangle_j)$, the set S_2.

Lemma 3.3 *Let s be a point and $\triangle_i$, $\triangle_j$ two pyramids such that $partner(s, \triangle_i) = partner(s, \triangle_j)$. If $rlp(s)$ lies to the left of $partner(s, \triangle_i)$, then the sliding constant to the left of s is the same as the sliding constant to the right of s.*

Proof: Let e be the event point to the left of s and let c be the sliding constant to the right of e. We have by the assumption that $rlp(s) = partner(s, \triangle_k)$ for some pyramid $\triangle_k$ different from $\triangle_i$ and $\triangle_j$. As proven in [1] the sliding constant c depends only on $\triangle_k$ and, hence, s is not an event point where the sliding constant changes. $\square$

To remove those points of S that are not event points, first sort the primary event points and find the sliding constant to the right of the event point and the right limit point in linear time per point. Sort the points in S_2, the superset of secondary event points, and find the associated sliding constants in constant time per point as in [1]. We will construct the sorted set E consisting of only the primary and secondary event points. Insert the primary event points in E. For each point s from left to right in S_2, apply binary search on the current set of points in E to find the event point e to the left of s. If c is the sliding constant to the right of e we

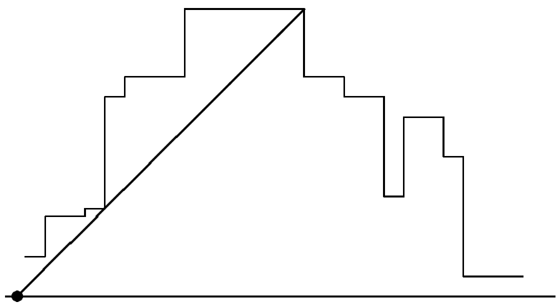

Figure 6: The extra event points.

have $rlp(s) = rlp(e+\delta) = rlp(e) + c\delta$ where $\delta = s - e$ since there cannot be an event point between e and s. Hence, we can compute the right limit point of s in $O(\log n)$ time and if $rlp(s) = partner(s, \triangle_i)$, where $\triangle_i$ is the pyramid as specified above, we add s to E, otherwise s is discarded since by Lemma 3.3 the sliding constants to the left and to the right of s are the same. Thus, in a total of $O(n^2 \log n)$ time we have computed the event points in sorted order, the sliding constant to the right of each event point, and the right limit point of each event point.

However, the event points as defined above are not all the points where the sliding constant can change its value. The only possible points are those points s which lie to the left of the leftmost event point associated to a pyramid. Consider the point t on the base under the leftmost east edge of a pyramid. The point t sees all of the pyramid so $partner(t, \triangle)$ is undefined. Let s be a point such that $t = rlp(s)$. What happens when s moves to the right? Answer, $rlp(s+\varepsilon) = t$ and, hence, the sliding constant is 0 until $s+\varepsilon$ is the point for which the supporting line intersects the right end point of the top edge. From then on and further to the right the sliding constant is greater than 0. Thus, we have to add these extra points (there are $O(n)$ of them) to the set of event points; see Figure 6. The sliding constants and right limit points of these points can easily be computed in $O(n^2)$ time.

We continue by taking a more detailed look at the structure of routes. Consider a longest spanning set of routes $\mathcal{R} = \{r_1, \ldots, r_m\}$, $r_i = [a_i, b_i]$, with $\lambda(\mathcal{R}) = \ell$ such that b_m lies to the left of q. This means that not everything to the right of b_m is seen from the routes and therefore the routes have to be extended by some length. If we define $rlp^0(s,\ell) = s$ and $rlp^i(s,\ell) = rlp(rlp^{i-1}(s,\ell)+\ell)$ when $i > 0$, we get $a_i = rlp^{i-1}(p,\ell)$ and $b_i = rlp^{i-1}(p,\ell)+\ell$. Hence, we can view the point b_i as a function on the length $\lambda(\mathcal{R}) = \ell$ which we write as $b_i(\ell)$.

We construct a new set of longest spanning routes $\mathcal{R}' = \{r'_1, \ldots, r'_m\}$, with $\lambda(\mathcal{R}') = \ell + \varepsilon$ and $r'_i = [a'_i, b'_i]$, and observe how the routes change when ε is small. From the definition we have that $a'_1 = a_1 = p$, together with $b'_i = a'_i + \ell + \varepsilon$, and $a'_{i+1} = rlp(b'_i) = rlp(a'_i + \ell) + c\varepsilon$ for some nonnegative constant c which only depends on the event point e immediately to the left of b'_i.

For the new left end points we can show

$$a'_i = rlp^{i-1}(p, \ell+\varepsilon) = rlp^{i-1}(p,\ell) + \sum_{v=1}^{i-1} c_{j_v} \cdots c_{j_{i-1}} \varepsilon = a_i + \sum_{v=1}^{i-1} c_{j_v} \cdots c_{j_{i-1}} \varepsilon$$

by a simple induction and for the new right end points we get

$$b'_i = a'_i + \ell + \varepsilon = a_i + \ell + \sum_{v=1}^{i-1} c_{j_v} \cdots c_{j_{i-1}} \varepsilon + \varepsilon = b_i + (\sum_{v=1}^{i-1} c_{j_v} \cdots c_{j_{i-1}} + 1)\varepsilon$$

and, hence, using the functional notation

$$b_i(\ell+\varepsilon) = b_i(\ell) + (\sum_{v=1}^{i-1} c_{j_v}\cdots c_{j_{i-1}} + 1)\varepsilon$$

for sufficiently small values of ε.

The previous calculations suggest the following algorithm for computing the longest spanning watchmen routes between points p and q. Let $\mathcal{E}$ be the set of primary, secondary, and extra event points. Add the points p and q to $\mathcal{E}$. We first try to compute the length ℓ of the longest spanning set of routes that see the histogram. We do this by application of a binary search approach to narrow down the possible values of ℓ. The claim is that once we have the correct length we can compute the longest spanning routes efficiently.

We state the algorithm in detail below. It consists of a loop that keeps the following invariant: After the loop has been executed i times we have two lengths $min\ell$ and $max\ell$ such that $min\ell \leq \ell \leq max\ell$, there is no event point in an interval $[b_j(min\ell), b_j(max\ell)]$, for any value $1 \leq j \leq i$, and q lies in the interval $[b_m(min\ell), b_m(max\ell)]$. Thus, after m executions of the loop the length of the optimum set of m watchmen routes can be computed by a simple application of the definition and given this length the routes can be easily found.

Algorithm *Longest-Spanning-Watchmen-Routes*
Input: A histogram polygon P and a positive integer m
Output: The longest spanning set of m watchmen for P
Step 1 Compute the set $\mathcal{E} = \{p = e_1, \ldots, e_k = q\}$ in sorted order, compute the right limit points $rlp(e_1), \ldots, rlp(e_k)$, and find the sliding constants $c_1, \ldots, c_{k-1}$ to the right of each event point (except the last)
$min\ell := 0$, $max\ell := q - p$, $left := 1$, $right := k$, $C_1 := 0$
Step 2 **for** $i := 1$ **to** m **do**
Step 2.1 **while** $right - left > 1$ **do**
$mid := \lfloor (left + right)/2 \rfloor$
Step 2.1.1 $\ell := \mathit{Find\text{-}Length}(i, mid, min\ell)$
Compute the longest spanning set of m routes having length ℓ
if $b_m(\ell) \preceq q$ **then**
$left := mid$ and $min\ell := \ell$
/* ℓ too short */
else
$right := mid$ and $max\ell := \ell$
/* ℓ too long */
endif
endwhile
Step 2.2 $C_{i+1} := c_{left} \cdot C_i + c_{left} = \sum_{v=1}^{i} c_{j_v} \cdots c_{j_i}$
Step 2.3 Compute the two longest spanning sets of m routes having length $min\ell$ and $max\ell$ respectively
$left :=$ index of the event point lying immediately to the left of $b_{i+1}(min\ell)$
$right :=$ index of the event point lying immediately to the right of $b_{i+1}(max\ell)$
endfor
Step 3 $\ell := \mathit{Find\text{-}Length}(m, k, min\ell)$
Step 4 Compute the longest spanning set of m routes having length ℓ
return the routes
End *Longest-Spanning-Watchmen-Routes*

In Step 1 the set of event points is computed in sorted order and the associated sliding constant and right limit point to each event point is found, taking $O(n^2 \log n)$ time.

In Step 2.1 we apply binary search to find the two lengths $min\ell$ and $max\ell$ such that the invariant holds, implying that the loop is executed $O(\log n)$ times. During each loop, Step 2.1.1, the correct length ℓ of the longest spanning set of i routes ending at event point e_{mid} is found by the *Find-Length* routine. This takes linear time as we shall see subsequently. Next, we compute the longest spanning set of m routes having length ℓ and check whether the right end point of the m^{th} route lies to the left or to the right of q. Depending on the case the number of event points in the span we consider (between *left* and *right*) is halfed accordingly. The computation of the longest spanning set of routes takes linear time using a simple extension of the static guard algorithm presented in [1]. Step 2.1 therefore uses $O(n \log n)$ time. In Step 2.3 the interval of event points is updated for the next application of Step 2.1. This takes, as stated, linear time. Hence, the total running time of Step 2 is $O(mn \log n)$.

Steps 3 and 4 take linear time each, giving a total running time for the algorithm of $O(n^2 \log n + mn \log n) = O(n^2 \log n)$, since $m = O(n)$.

The storage bound is $O(n^2)$ since the set $\mathcal{E}$ of event points, the right limit points, and the sliding constants have to be maintained.

The *Find-Length* routine finds the lengths of the longest spanning set of i routes with the rightmost end point lying on e_z under the assumption that the sliding constants $c_{j_1}, \ldots, c_{j_{i-1}}$ are known and we are given a length len such that the sliding constants do not change when the rightmost end point slides from $b_i(len)$ to e_z. These assumptions hold when the routine is called in Step 2.1.1 with $len = min\ell$ and $z = mid$ since e_{mid} lies to the right of $b_i(min\ell)$ and none of the intervals $[b_j(min\ell), b_j(max\ell)]$, with $1 \leq j \leq i-1$, contain any event points by the invariant.

From the calculations above we therefore have

$$e_z = b_i(len + \varepsilon) = b_i(len) + (\sum_{v=1}^{i-1} c_{j_v} \cdots c_{j_{i-1}} + 1)\varepsilon = b_i(len) + (C_i + 1)\varepsilon$$

from which we compute the extension length ε in Step 2.

Algorithm	*Find-Length*
Input:	Two positive integers i and z and the current length len
Output:	The new length $len + \varepsilon$
Step 1	Compute the longest spanning set of i routes having length len
Step 2	$\varepsilon := (e_z - b_i(len))/(C_i + 1)$
	return $len + \varepsilon$
End *Find-Length*	

The linear time bound for the *Find-Length* routine follows from Step 1. The routine computes the longest spanning set of routes given a length in the same way as above.

We have proved the following theorem.

THEOREM 1 *The algorithm Longest-Spanning-Watchmen-Routes finds a MinMax optimum set of watchmen routes in $O(n^2 \log n)$ time and $O(n^2)$ storage.*

PROOF: The time and storage bounds follow from the discussion above.

The algorithm returns m watchmen each having length ℓ. We must show that there does not exist any other collection of m watchmen having length shorter than ℓ such that the histogram is completely seen. Assume there is such a collection. This means that the right end point b'_m of the rightmost watchman cannot lie to the left of q. Extend the rightmost watchman so that it has length ℓ giving that the new right end point lies to the right of q. This contradicts Lemma 3.2 since b_m, the right end point of the rightmost watchman in the longest spanning collection of m watchmen having length ℓ, lies on q. Hence, the longest spanning collection of watchmen routes is also a MinMax optimum set of watchmen routes. □

4 The Weighted Case

In Section 3 we studied the case when m watchmen are moving along routes each with the same velocity. We want to ensure that all points of the polygon are seen as often as possible which leads to the MinMax optimization criterion. Consider the extended version of this problem where the watchmen move with different velocities. We call this problem the *weighted MinMax routes problem*. Assume that to each watchman w_i there is associated a positive weight α_i specifying the time it takes for the watchman to move a unit distance. Hence, the total time for the watchman to traverse its route is $2\alpha_i|w_i|$ and we want routes such that $\max_{1\leq i\leq m} 2\alpha_i|w_i|$ is minimized.

We can define the *weighted longest spanning* collection of m routes of weight ℓ in a similar manner as before. $\mathcal{R} = \{r_1,\ldots,r_m\}$, $r_i = [a_i, b_i]$ such that $2\alpha_i|r_i| = \ell$, $a_1 = p$, $a_{i+1} = rlp(b_i)$, and, $b_m \preceq q$. We can prove a companion lemma to Lemma 3.2 saying that $a'_i \preceq a_i$ and $b'_i \preceq b_i$ where a'_i and b'_i are the left and right end points (respectively) of the i^{th} route of *any* collection of m routes such that each route has at most weight ℓ.

Define $wrlp^0(s,\ell) = s$ and $wrlp^i(s,\ell) = rlp(wrlp^{i-1}(s,\ell) + \ell/\alpha_i)$ for $i > 0$. We have that $a_i = wrlp^{i-1}(p,\ell)$ and $b_i = wrlp^{i-1}(p,\ell) + \ell/\alpha_i$.

Now we examine what happens when the weighted longest spanning set of routes having weight ℓ is extended by weight ε, i.e., route r_i is extended by length ε/α_i giving a new set of routes $\mathcal{R}' = \{r'_1,\ldots,r'_m\}$, $r'_i = [a'_i, b'_i]$. We get the left end points

$$\begin{aligned} a'_i = wrlp^{i-1}(p,\ell+\varepsilon) &= wrlp^{i-1}(p,\ell) + \sum_{v=1}^{i-1} \frac{c_{j_v}\cdots c_{j_{i-1}}}{\alpha_v\cdots\alpha_{i-1}}\varepsilon \\ &= a_i + \sum_{v=1}^{i-1} \frac{c_{j_v}\cdots c_{j_{i-1}}}{\alpha_v\cdots\alpha_{i-1}}\varepsilon \end{aligned}$$

and the right end points

$$\begin{aligned} b'_i = a'_i + (\ell+\varepsilon)/\alpha_i &= a_i + \ell/\alpha_i + \sum_{v=1}^{i-1} \frac{c_{j_v}\cdots c_{j_{i-1}}}{\alpha_v\cdots\alpha_{i-1}}\varepsilon + \varepsilon/\alpha_i \\ &= b_i + (\sum_{v=1}^{i-1} \frac{c_{j_v}\cdots c_{j_{i-1}}}{\alpha_v\cdots\alpha_{i-1}} + \frac{1}{\alpha_i})\varepsilon. \end{aligned}$$

Hence, only a few small changes of the *Longest-Spanning-Watchmen-Routes* algorithm need to be made to solve the weighted problem. In Step 2.2 the constants should be computed by $C_{i+1} := (c_{left}/\alpha_i)C_i + c_{left}/\alpha_i$ and furthermore all instances of $rlp^i(p,\ell) + \ell$ should be exchanged for $wrlp^i(p,\ell) + \ell/\alpha_{i+1}$.

The running time and storage requirements are the same as for the MinMax case as is the proof of correctness, giving us the following theorem.

THEOREM 2 *There is an algorithm which solves the Weighted MinMax optimum watchmen routes problem for histogram polygons in $O(n^2\log n)$ time and $O(n^2)$ storage.*

5 Conclusions

We have solved the problem of computing a MinMax and a Weighted MinMax optimum set of m watchmen routes in a histogram polygon. The algorithms run in $O(n^2\log n)$ time and use $O(n^2)$ storage.

Other problems related to the problem of finding "best" watchmen for any number of watchmen is that of finding the "best" watchmen for a fixed, constant, number of watchmen, e.g., is there a polynomial time algorithm to compute the "best" 2-watchmen routes for simple polygon? As stated, this problem is NP-hard under the MinMax criterion but still open for the MinSum criterion. Even for the single watchman route case, the problem has not been completely answered although an optimal linear time algorithm exists for rectilinear polygons.

Acknowledgements

We thank Prof. Hartmut Noltemeier for suggesting the Weighted MinMax problem.

References

[1] S. Carlsson, B.J. Nilsson, S. Ntafos. Optimum Guard Covers and m-Watchmen Routes for Restricted Polygons. *International Journal of Computational Geometry and Applications*, 3(1):85–105, 1993.

[2] B. Chazelle. Triangulating a Simple Polygon in Linear Time. In *Proc. 31st Symposium on Foundations of Computer Science*, pages 220–230, 1990.

[3] W. Chin, S. Ntafos. Optimum Watchman Routes. *Information Processing Letters*, 28:39–44, 1988.

[4] W. Chin, S. Ntafos. Shortest Watchman Routes in Simple Polygons. *Discrete and Computational Geometry*, 6(1):9–31, 1991.

[5] V. Chvátal. A Combinatorial Theorem in Plane Geometry. *Journal of Combinatorial Theory B*, 13(6):395–398, 1975.

[6] H. Edelsbrunner, J. O'Rourke, E. Welzl. Stationing Guards in Rectilinear Art Galleries. *Comput. Vision, Graphics, Image Processing*, 27:167–176, 1984.

[7] S. Fisk. A Short Proof of Chvátal's Watchman Theorem. *Journal of Combinatorial Theory B*, 24:374, 1978.

[8] J.S.B. Mitchell, E.L. Wynters. Watchman Routes for Multiple Guards. In *Proc. 3rd Canadian Conference on Computational Geometry*, pages 126–129, 1991.

[9] B.J. Nilsson, D. Wood. Watchmen Routes in Spiral Polygons. Technical Report LU-CS-TR:90-55, Dept. of Computer Science, Lund University, 1990. An extended abstract of a preliminary version was presented at the *2nd Canadian Conference on Computational Geometry*, pages 269–272, 1990.

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

[11] J.-R. Sack. *Rectilinear Computational Geometry.* PhD thesis, School of Computer Science, McGill University, 1984.