Guarding a Terrain by Two Watchtowers[*]

Pankaj K. Agarwal[†] Sergey Bereg[‡] Ovidiu Daescu[*] Simeon Ntafos[*]
Micha Sharir[§¶] Binhai Zhu[||]

Abstract

Given a polyhedral terrain T with n vertices, the *two-watchtower* problem for T asks to find two vertical segments, called *watchtowers*, of smallest common height, whose bottom endpoints (bases) lie on T, and whose top endpoints *guard* T, in the sense that each point on T is visible from at least one of them. There are three versions of the problem, *discrete*, *semi-discrete*, and *continuous*, depending on whether two, one, or none of the two bases are restricted to be among the vertices of T, respectively.

In this paper we present the following results for the two-watchtower problem in $\mathbb{R}^2$ and $\mathbb{R}^3$: (1) We show that the *discrete* two-watchtowers problem in $\mathbb{R}^2$ can be solved in $O(n^2 \log^4 n)$ time, significantly improving previous solutions. The algorithm works, without increasing its asymptotic running time, for the semi-continuous version, where one of the towers is allowed to be placed anywhere on T. (2) We show that the *continuous* two-watchtower problem in $\mathbb{R}^2$ can be solved in $O(n^3 \alpha(n) \log^3 n)$ time, again significantly improving previous results. (3) Still in $\mathbb{R}^2$, we show that the continuous version of the problem of guarding a finite set $P \subset T$ of m points by two watchtowers of smallest common height can be solved in $O(mn \log^4 n)$ time. (4) We show that the discrete version of the two-watchtower problem in $\mathbb{R}^3$ can be solved in $O(n^{11/3}\,\mathrm{polylog}(n))$ time; this is the first nontrivial result for this problem in $\mathbb{R}^3$.

[*]P.A. was supported by NSF under grants CCR-00-86013, EIA-01-31905, CCR-02-04118, and DEB-04-25465, by ARO grants W911NF-04-1-0278 and DAAD19-03-1-0352, and by a grant from the U.S.–Israel Binational Science Foundation. O.D. was supported by NSF award CCF-0430366. M.S. was supported by NSF Grant CCR-00-98246, by a grant from the U.S.-Israeli Binational Science Foundation (jointly with P.A.), by a grant from the Israeli Academy of Sciences for a Center of Excellence in Geometric Computing at Tel Aviv University, and by the Hermann Minkowski–MINERVA Center for Geometry at Tel Aviv University.

[†]Department of Computer Science, Duke University, Durham, NC 27708, USA. E-mail: pankaj@cs.duke.edu

[‡]Department of Computer Science, University of Texas at Dallas, Box 830688, Richardson, TX 75083, USA. E-mail: {besp,daescu,simeon}@utdallas.edu

[§]School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel. E-mail: michas@post.tau.ac.il

[¶]Courant Institute of Mathematical Sciences, New York University, New York, NY 10012, USA.

[||]Department of Computer Science, Montana State University, Bozeman, MT 59717-3880, USA. E-mail: bhz@cs.montana.edu

1 Introduction

A polyhedral terrain in $\mathbb{R}^d$ is the graph of a continuous, piecewise-linear $(d-1)$-variate function. Thus, a terrain in $\mathbb{R}^2$ is an x-monotone polygonal chain, while a terrain in $\mathbb{R}^3$ is an xy-monotone polyhedral surface. A *watchtower* is a vertical line segment whose bottom endpoint (*base*) lies on T. A point $w \in T$ is *seen* (or *guarded*) by a watchtower τ if the top endpoint of τ sees w, that is, the segment connecting it to w lies fully above T. Two watchtowers placed on T are said to guard T if each point on T is visible from the top endpoint of at least one of the towers.

In this paper we study the *two-watchtower* problem for polyhedral terrains in $\mathbb{R}^2$ and $\mathbb{R}^3$, which is defined as follows. Given a polyhedral terrain T with n edges, find the smallest height $h > 0$ for which there exist two points $u, v \in T$ such that the watchtowers of height h erected at u and v guard T.

Two versions of the problem have been studied in the literature. In the *discrete* version, the bases u and v are restricted to be among the vertices of T (or, for that matter, could belong to any prespecified finite point set). In the *continuous* version, u and v can be located anywhere on T. In this paper we study these two versions and also address a new version, called *semi-discrete*, in which the base of one tower is restricted to be among the vertices on T while the base of the other tower can be anywhere on T (see Figure 1). We further introduce a variant of the problem when not all of T needs to be guarded. Instead, we specify a finite set P of m "critical" points on T, and the goal is to find two watchtowers of minimum common height that together guard P (i.e., every point of P is visible from at least one of the towers). Again, we may consider the discrete, semi-continuous, or the continuous versions of this problem.

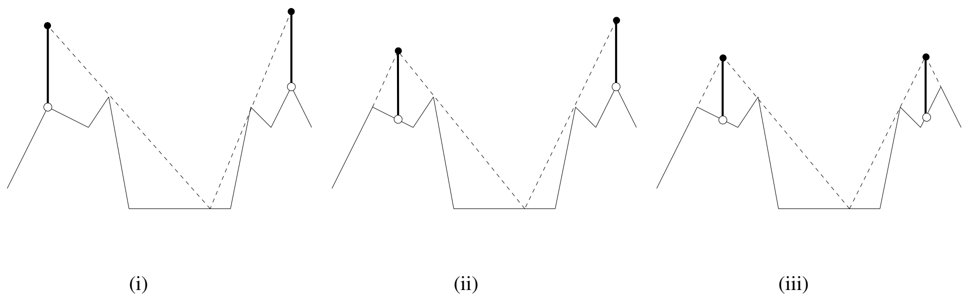

Figure 1. The three versions of the problem in $\mathbb{R}^2$: (i) discrete, (ii) semi-discrete, and (iii) continuous.

Of course, similar problems can be stated for any number of watchtowers, and one can also consider variants, such as the one where the height h of the watchtowers is specified, and one wishes to find the *smallest* number of watchtowers of that height that collectively guard T.

Related work. Guarding a terrain by watchtowers is a special case of the general class of visibility problems in two and three dimensions, known as *art gallery problems*, which have been extensively studied for more than two decades. These problems have numerous applications in surveillance, navigation, computer vision, modelling and graphics, GIS, and many more. See [23] for a recent survey of art gallery problems.

The problem of guarding a terrain in $\mathbb{R}^2$ by two watchtowers has been studied in several recent papers. Bespamyatnikh *et al.* [4] show that for the discrete case in $\mathbb{R}^2$ deciding whether there exist two watchtowers of given height h that guard T can be done in $O(n^3)$ time. Using parametric search [18], they obtain an $O(n^3 \log^2 n)$-time algorithm for the optimization problem. They also present an $O(n^4)$-time solution that avoids parametric search. Ben-Moshe *et al.* [2] also address the discrete version of the problem in $\mathbb{R}^2$ and give an $O(n^{2.688} \log^2 n)$-time algorithm, based on parametric search and on the computation of all dominances for a set of n points in $\mathbb{R}^n$. The continuous case for $\mathbb{R}^2$ is solved in [4] by an algorithm that takes $O(n^4 \log^2 n)$ time, using parametric search.

Much less is known about terrain guarding in $\mathbb{R}^3$. Early work on terrain guarding, due to Cole and Sharir [10], shows that the problem of finding the minimum number of guards is NP-complete, even if the guards are placed on the terrain (no elevation is allowed). However, the case of a single watchtower guarding the terrain has been shown by Sharir [21] to be solvable in $O(n \log^2 n)$ time. An $O(n \log n)$-time algorithm for this problem was later obtained by Zhu [24].

Attention has also been given to the problem of guarding a two-dimensional terrain with the minimum number of guards placed on the terrain. If the guards are at some fixed height, the minimum number of guards can be found in polynomial time [19]. Recently, Ben-Moshe, Katz, and Mitchell [3], and, independently, Clarkson and Varadarajan [8], discovered constant-factor approximation algorithms for the problem. Eidenbenz et al. [12] show that the related problem of guarding a simple polygon with a minimum number of guards is APX-hard. So for the problem of guarding polygons there is an ε such that it is NP-hard to obtain a $(1 + \varepsilon)$-approximation for the minimum number of guards.

Our results. we study the problem of guarding a terrain by two watchtowers in $\mathbb{R}^2$ and $\mathbb{R}^3$. For the planar case we obtain the following results.

(i) We show that the discrete two-watchtower problem can be solved in $O(n^2 \log^4 n)$ time, significantly improving the previous solutions cited earlier. The algorithm works, without affecting its asymptotic running time, for the semi-continuous version as well, in which one of the bases can be anywhere on T and the other has to be placed at a vertex of T.

(ii) We show that the continuous two-watchtower problem can be solved in $O(n^3 \alpha(n) \log^3 n)$ time, again significantly improving previous results.

(iii) We show that the continuous version of the problem of guarding a finite set $P \subset T$ of m points by two watchtowers can be solved in $O(mn \log^4 n)$ time.

We also study the problem of guarding a terrain by two watchtowers in $\mathbb{R}^3$, and present an $O(n^{11/3} \operatorname{polylog}(n))$ time algorithm for the discrete two-watchtower problem. This is the first nontrivial algorithm for the problem: A trivial solution for the discrete problem takes about $O(n^4)$ time.

All the results derived in this paper are based on the parametric-searching technique [18]. For each result, we first design a *decision procedure* that, given T and a real value $h > 0$, determines whether T can be guarded by two watchtowers of height at most h. Next, we apply the parametric searching technique to the decision procedure, to obtain an algorithm that finds where to place two watchtowers of smallest possible height.

The parametric searching step involves developing a parallel algorithm for the decision problem and simulating it generically at the unknown value of the smallest height. While the main contributions of the paper lie in developing the decision procedures, their generic implementations for parametric searching are also nontrivial and we describe them too in detail. (We note that previous results on the problem do not detail the parametric searching steps.)

Section 2 describes the algorithms for the discrete and semicontinuous two-watchtower problems in $\mathbb{R}^2$. Section 3 describes the algorithm for the continuous two-watchtower problem in $\mathbb{R}^2$. Section 4 discusses the continuous version in which only a given set of points on T need to be guarded. Section 5 describes the algorithm for the discrete two-watchtower problem in $\mathbb{R}^3$. We conclude the paper in Section 6 with a brief discussion and some open problems.

2 The Discrete and Semi-Continuous Problems in $\mathbb{R}^2$

Let T be an x-monotone polygonal chain in $\mathbb{R}^2$, with n edges, and let V be the set of its vertices. For a point $u \in T$ and $h > 0$, let $u(h)$ be the point that is vertically above u at distance h. We call the vertical segment $uu(h)$ *the watchtower* with base u and height h, but we often refer to the watchtower just by its top endpoint $u(h)$, for short.

The *discrete two-watchtower* problem asks for the smallest height $h > 0$ such that there exist two vertices $u, v \in V$, so that the two points $u(h)$ and $v(h)$ (the *tops* of the towers at height h erected at u and v, respectively), guard T. The *semi-continuous two-watchtowers* problem is the same, except that only one of u, v is required to be a vertex of T, and the second one can lie anywhere on T. We present algorithms that solve these two problems in time $O(n^2 \log^4 n)$. Both algorithms employ parametric searching [18], and therefore rely on a *decision procedure*, to guide the search for the optimum height.

Let T be a polygonal terrain in the plane with n edges, and let $h > 0$ be a fixed parameter. To solve the decision problem we need to determine whether there exist two points $u, v \in T$, with both $u, v \in V$ (discrete case) or at least one of u, v a vertex of T (semi-continuous case), such that the top endpoints $u(h)$, $v(h)$ of the watchtowers of height h at u, v guard T. We present a procedure that does much more than that: Given one tower $u(h)$, it finds the *shortest* second tower that can be placed *anywhere* on the terrain, so that both towers guard T. This serves as a decision procedure for both the discrete and the semi-continuous versions of the problem. We begin by describing a procedure that is a main step in the decision procedure. Next, we describe the decision procedure, and then discuss how to plug the parametric searching.

2.1 Computing visibility pairs

Let P be a set of points on T, and let V be the set of vertices of T. We want to find, for each vertex $v \in V$, a point $p \in P$ (if one exists), that satisfies the following conditions:

(i) p lies to the left of V;

(ii) p sees v; and

(iii) p is the rightmost point in P that satisfies (i) and (ii). Alternatively, among the points of P that satisfy (i) and (ii), the segment pv has the largest slope (it is easily verified that these alternative formulations are equivalent).

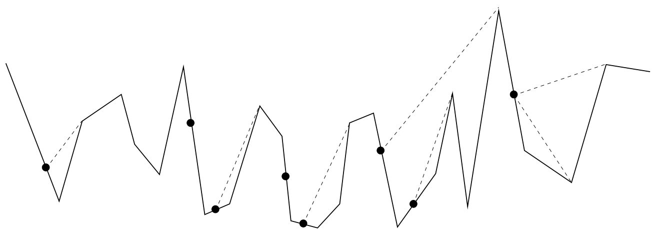

Figure 2. Some pairs of $\Pi(P, V)$.

Denote by $\Pi(P, V)$ the set of these pairs. See Figure 2. We assume that $|P| = O(n)$, which will be the case in our application. We solve this problem using the following divide-and-conquer technique. Consider the set $P \cup V$ and let ℓ be the vertical line that splits it into two subsets of equal size. Let P_L, V_L (resp., P_R, V_R) denote the subsets of P and V that lie to the left (resp., right) of ℓ. We clearly have

$$\Pi(P, V) = \Pi(P_L, V_L) \cup \Pi(P_R, V_R) \cup \Pi(P_L, V'_R),$$

where V'_R is the subset of all points $v \in V_R$ for which there is no point $p \in P_R$ that forms with v a pair in $\Pi(P_R, V_R)$. Let n' denote the size of V'_R. We compute $\Pi(P_L, V_L)$ and $\Pi(P_R, V_R)$ recursively, extract the subset V'_R, and compute $\Pi(P_L, V'_R)$ in the following direct way.

Without loss of generality, assume that ℓ is the y-axis. We pass to the dual plane, using a duality transform that maps a point (m, p) to the line $y = mx + p$. For each $p \in P_L$, let γ_p denote the dual representation of the locus of all lines λ that pass through p, such that p sees the intercept $\lambda \cap \ell$. In the dual plane, γ_p is a rightward directed ray, contained in the line dual to p and emanating from the point w_p that represents the shallowest line through p with the above property. Note that the rays γ_p are pairwise openly disjoint, because an intersection point of two such rays γ_p, $\gamma_{p'}$ is dual to a line that passes through p and p', so that both points see the intercept. Since both points lie on T, this is impossible (except at an endpoint of one of the rays, namely, the ray corresponding to the leftmost point among p, p').

In complete analogy, for each point $q \in V'_R$, let δ_q denote the dual representation of the locus of all lines λ that pass through q, such that q sees the intercept $\lambda \cap \ell$ (here q sees the intercept to its left). In the dual plane, δ_q is a leftward directed ray, emanating from the point z_q that represents the shallowest line through q with the above property. Here too the rays δ_q are pairwise openly disjoint. See Figure 3(i).

Let $q \in V'_R$, and let $p \in P_L$ be the point that satisfies $(p, q) \in \Pi(P_L, V'_R)$ (assuming that such a point exists). The point dual to the line that passes through p and q is a point that lies on both rays γ_p, δ_q, and is the *rightmost* point of intersection of δ_q with some ray γ_p (it corresponds to the line of largest slope in condition (iii)). Let Q denote the unbounded (and degenerate) 'comb-like' simple polygon, whose boundary consists of all the rays γ_p, and of a vertical line ℓ_0 at $x = +\infty$.

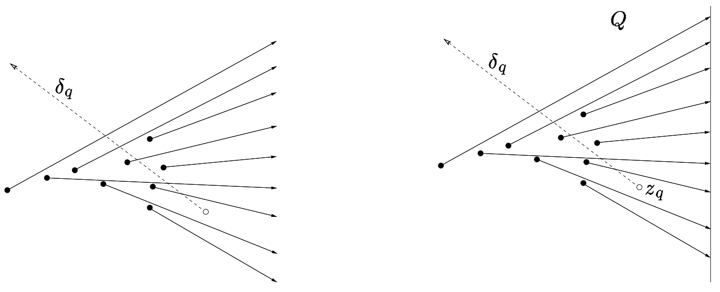

Figure 3. (i) The rays γ_p and a query ray δ_q. (ii) The polygon Q.

(In practice, we place ℓ_0 at a sufficiently large x-coordinate, and trim the rays γ_p at their intercepts with ℓ_0; see Figure 3(ii).) Then the point $p \in P_L$ that forms with q a pair in $\Pi(P_L, V'_R)$ is simply the output of a ray-shooting query inside Q along δ_q.

We first construct Q. For this, we need to compute the (apices of the) rays γ_p, for $p \in P_L$. Let T_L (resp., T_R) denote the region consisting of all points that lie above T and to the left (resp., right) of ℓ. Let b_ℓ denote the intersection point $T \cap \ell$. We construct the shortest-path map inside T_L with b_ℓ as a source, as in [13]. This allows us to find, in linear time, the terminal segments of all the shortest paths from b_ℓ to the points of P_L. For each $p \in P_L$, the apex of γ_p is then the dual of the line containing the respective terminal segment pa; see Figure 4. The total cost of this step is linear.

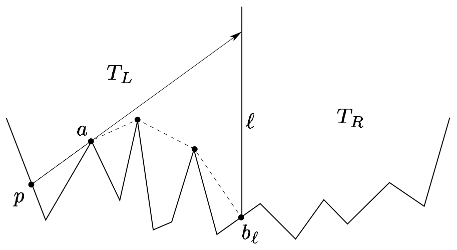

Figure 4. Constructing the rays γ_p from the shortest-path map from b_ℓ.

To complete the construction of Q, we need to sort the rays γ_p by their intercepts with ℓ_0. However, this order is equal to the order of the slopes of these rays, which in turn is the same as the order of the abscissas of the points of P_L, and this order can be computed prior to the beginning of the divide-and-conquer process (the compuation of the order of abscissas over P requires $O(n \log n)$ time). Hence, this substep also takes linear time.

Using the algorithm of Guibas *et al.* [13] (see also [5]), we next preprocess Q in linear time for ray shooting queries that take logarithmic time each. We then construct the (apices of the) rays δ_q, for $q \in V'_R$. This is done, in a symmetric manner to the construction of the rays γ_p, using the region T_R instead of T_L. This step also runs in linear time. We then perform ray shooting queries in Q with the rays δ_q, for $q \in V'_R$, thereby obtaining the set $\Pi(P_L, V'_R)$. Hence, the overall cost of the 'merge' step of the recursion is $O(n \log n)$ time, so the entire procedure takes $O(n \log^2 n)$ time.

In summary, we have an algorithm that computes the set $\Pi(P, V)$ in $O(n \log^2 n)$ time.

2.2 The decision procedure

Let H be the region of the plane that lies above T. In the decision procedure, we fix a vertex u of T, erect the first tower, of height h, over u, and compute the *visibility polygon* $W(u(h))$, which is the portion of H that is visible from $u(h)$. The polygon $W(u(h))$ can be computed in linear time [13]. Let $\partial W(u(h))$ denote the boundary of $W(u(h))$. The portion of $\partial W(u(h))$ that lies on T consists of subsegments of the edges of T, at most one subsegment for each edge, which are delimited either by the vertices of T, or by intercepts of visibility rays that emanate from $u(h)$ and pass through some vertex of T. In either case, at least one endpoint of each visibility segment is a vertex of T. See Figure 5.

Let $P = P(u, h)$ denote the set of all the endpoints of the portions of the edges of T that are not visible from $u(h)$. Clearly, $u(h)$ and another tower top $v(h')$ (with a possibly different height h') guard T if and only if $v(h')$ sees all the points of P. This follows from the easy observation that if a point $v(h')$ above T sees both endpoints of a subsegment of some edge of T then it sees the entire subsegment.

The following lemma provides the crucial geometric property on which our algorithm relies.

Lemma 2.1. *For a point v on T and height h', the tower top $v(h')$ sees all the points of P that lie to the left of v if and only if $v(h')$ lies above all the lines pw such that $(p, w) \in \Pi(P, V)$ and w lies to the left of (or coincides with) v.*

Equivalently, $v(h')$ sees all the points of P that lie to the left of v if and only if, for each $p \in P$ to the left of v, $v(h')$ lies above the steepest line pw such that $(p, w) \in \Pi(P, V)$ and w lies to the left of (or coincides with) v.

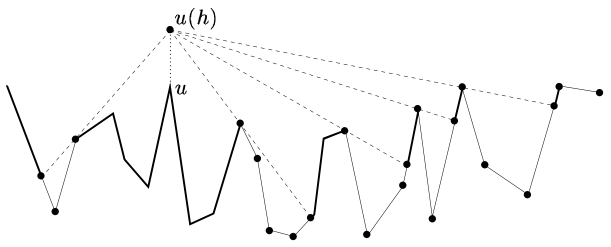

Figure 5. The visible portions of T from $u(h)$, and the set $P(u, h)$ (highlighted along T).

Proof: If $v(h')$ sees all the points of P that lie to the left of v then it must lie above all the lines pw as in the first statement of the lemma, or else the corresponding point p would not be visible from $v(h')$. Conversely, suppose that $v(h')$ lies above all these lines, and let p be a point in P to the left of v. If $v(h')$ does not see p then the shortest path from p to $v(h')$ that lies above T must bend at some vertices of T. Let w be the leftmost such vertex. If $(p, w) \in \Pi(P, V)$, then (p, w) satisfies the conditions in the lemma, and $v(h')$ passes below the line pw, contrary to our assumption. If $(p, w) \notin \Pi(P, V)$ then there must exist another point $p' \in P$ that lies between p and w so that

$(p', w) \in \Pi(P, V)$. The line $p'w$ is even steeper than pw, and $v(h')$ thus lies below it, again a contradiction. The second statement of the lemma is obvious. $\Box$

Lemma 2.1 has a symmetric version, which handles visibility from a tower top $v(h')$ to the points of P to its right. For this version we need a symmetric version of $\Pi(P, V)$, involving pairs $(p, v) \in P \times V$ with p lying to the right of v, which is defined and constructed in a fully symmetric manner.

Next, we sweep the plane with a vertical line ℓ from left to right, starting with the leftmost vertex of T, and construct a *height function* h_L, so that, for each $q \in T$, $h_L(q)$ is the height of the shortest tower erected over q that sees all the points of P to its left. Equivalently, in view of Lemma 2.1, $h_L(q)$ is the distance from q to the *upper envelope* E of the lines pw, for all the pairs $(p, w) \in \Pi(P, V)$ with w lying to the left of (or coinciding with) q.

As we sweep ℓ, $h_L(q)$ remains a linear function, equal to the vertical distance between an edge of E and an edge of T, until ℓ reaches a vertex of either E or T. If it reaches a vertex of E, we pass to another edge of E, and the corresponding update of h_L is easy and obvious. If ℓ reaches a vertex v of T, in addition to the obvious local update of h_L, we need to dynamically update E, by adding the line pv, for the unique pair $(p, v) \in \Pi(P, V)$, to the set of lines that form E, or do nothing if such a pair does not exist. This has the effect of either leaving E unchanged or, since E is convex, creating two new vertices of E and deleting the old vertices between them, and can easily be done in overall time $O(\log n)$. After each update is performed, we also need to find the next vertex of E to the right of the current position of ℓ, which can be done in $O(1)$ time per update. Hence, the construction of h_L can be done in $O(n \log n)$ time (after the preprocessing stage of constructing $\Pi(P, V)$, which takes $O(n \log^2 n)$ time). Note that this algorithmic analysis also implies that the combinatorial complexity (number of vertices of the graph) of the function h_L is $O(n)$.

We then apply a symmetric process, in which we sweep the plane from right to left, starting with the rightmost vertex of T, and construct the symmetrically defined height function h_R. We then construct the upper envelope h^* of h_L and h_R, in time $O(n)$, using a standard merging procedure. The global minimum of h^* is the shortest height of a second tower, erected anywhere on T, that sees, together with $u(h)$, the entire terrain.

Since we need to repeat this step for each vertex u of T, the entire decision procedure runs in $O(n^2 \log^2 n)$ time. In summary, we thus have:

Lemma 2.2. *(a) Given a terrain T in the plane with n edges, a vertex u of T, and a height $h \geq 0$, one can find, in $O(n \log^2 n)$ time, the shortest second tower, erected anywhere on T, that sees, together with $u(h)$, the entire terrain.*

(b) Given a terrain T in the plane with n edges, and a height $h \geq 0$, one can determine, in $O(n^2 \log^2 n)$ time, whether h is smaller than, equal to, or greater than the optimum height for the discrete or the semi-continuous two-towers problem for T.

Remark: An obvious open problem is to improve the efficiency of the construction of the set $\Pi(P, V)$. Can it be done in $O(n \log n)$ time? This is the bottleneck step in the decision procedure, and the above improvement, if possible, would have reduced the total running time to $O(n^2 \log n)$.

2.3 Plugging the parametric search

In this section we show how to find the optimum height in the discrete and semi-continuous cases by applying the parametric searching technique [18]. That is, we run a generic version of the decision procedure, in which h is left as an unknown parameter. Comparisons that depend on h are resolved by finding the few critical values of h at which the answer to the comparison may change, and by running (a concrete version of) the decision procedure at these critical heights, thereby finding the noncritical range that contains the optimum height h^* and is delimited by two consecutive critical heights. This determines the outcome of the comparison, and at the same time narrows down the range I that is known to contain h^*. We proceed in this manner through the execution of the generic decision procedure. If h^* is found during one of these comparison resolutions, we stop and report it. Otherwise, upon termination, we output the smaller endpoint of the final interval I. Since the function $f(h)$ corresponding to the portion of T visible from two towers of height h is a monotonically increasing function, the overall problem can be casted as a monotonic root finding problem, for which the parametric searching procedure is guaranteed to produce the optimal solution h^*.

To reduce the cost of the generic execution, we need to run a parallel version of it. More precisely, the only steps in the algorithm that need be parallelized are comparisons that depend on h. All such comparisons that arise in a single parallel step of the algorithm are resolved simultaneously, by running a binary search on all the resulting critical heights. If the algorithm runs in T_π parallel steps and uses p processors, then its overall cost is $O(T_\pi(p + D\log p))$, where $D = O(n^2 \log^2 n)$ is the cost of the decision procedure.

We describe two such generic parallel implementations of the decision procedure, one for the discrete problem and one for the semi-continuous problem. Both implementations require $O(\log n)$ parallel steps. The first algorithm requires $O(n^2 \log n)$ processors, and the second requires $O(n^2 \log^3 n)$ processors. Hence, both result in algorithms with $O(n^2 \log^4 n)$ running time.

We begin by preprocessing T as follows. For each vertex u of T, we compute the visibility map from the vertical halfline above u, with an endpoint at u, using the algorithm of [13]. This takes overall time $O(n^2)$, since computing the visibility map from the vertical halfline above u takes $O(n)$ time. For each u, the output contains all mutually visible pairs (v, v') of vertices, such that some point above u sees both v and v' along the line vv'. In addition, we also have the first point along that line, as we trace it from the point above u towards v and v', where it crosses T into the region below T (which may happen at the farthest vertex among v and v' or at another further point of T). As shown in [13], the number of such critical events is linear for each u. We thus obtain $O(n)$ critical heights over u, one for each of these visibility events (see Figure 6). Repeating this step for each vertex u, we obtain a set H_0 of a total of $O(n^2)$ critical heights, and a corresponding set $\mathcal{I}$ of intercepts along the edges of T, where the above visibility rays vv' 'enter' the region below T.

We run a standard binary search through H_0, using the decision procedure described above. After $O(\log n)$ calls to that procedure, with overall cost $O(n^2 \log^3 n)$, we obtain an initial interval I_0 that contains the optimum height h^*. For each $u \in V$ and for each $h \in I_0$, the visibility polygon $W(u(h))$ has a fixed combinatorial structure. In particular, for each visible portion of an edge e of T, the nature of its endpoints is fixed, in the sense that each of them is either a fixed vertex of T, or the intercept along e of a visibility ray that emanates from $u(h)$ and passes through a fixed

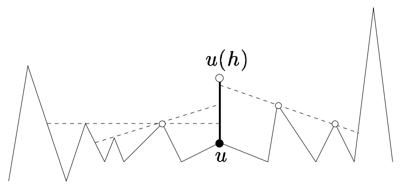

Figure 6. The critical heights over u.

vertex of T before hitting e. Let $P(u, h)$ denote the set of those endpoints of the visibility segments of $W(u(h))$ that are not vertices of T, and set $P(h)$ to be the union of these sets, over all vertices u. Clearly, $P(h)$ has size $O(n^2)$. The parameter h reminds us that these are parametric points that depend on h.

Our next step is to locate the points of $P(h)$ among the points of $\mathcal{I}$. We run the following process iteratively. At each stage we have a partition S of the edges of T into subsegments, each delimited by points of $\mathcal{I} \cup V$, an interval $I_S \subseteq I_0$, and a corresponding partition of $P(h)$ into subsets, each known to be contained in a fixed segment s of S, for all $h \in I_S$. Initially, S is the set of edges of T, $I_S := I_0$, and the sets $P_s(h) := P(h) \cap s$, for s an edge of T, form the desired partition of $P(h)$. (Recall that, for $h \in I_0$, the points of $P_s(h)$ move continuously along s, and no point enters or leaves this set.) Each iterative step refines S as follows.

For each segment $s \in S$, set $\mathcal{I}_s := \mathcal{I} \cap \mathrm{int}(s)$. Segments $s \in S$ with $\mathcal{I}_s = \emptyset$ are removed from S, and we report the corresponding sets $P_s(h)$ as part of the output of this stage. For any remaining segment s, let q_s denote the median point of $\mathcal{I}_s$. For each $p \in P_s(h)$, let h_p be the (unique) height at which p coincides with q_s. If there is no such height, or if h_p lies outside the range I_S, then we know which half of s contains p, and we ignore p in the remainder of the present step. Repeating this over all segments $s \in S$, we obtain a set H_S of $O(n^2)$ critical heights, and we run a binary search through them, as above, to obtain a subinterval $I' \subseteq I_S$, so that for each $h \in I'$, $s \in S$, and $p \in P_s(h)$, we know which half of s contains p. The next partition S' is then obtained by splitting each surviving segment $s \in S$ at the median point q. We set $I_{S'} := I'$, and get the new partition of $P(h)$ by assigning each point p to the new half-segment it belongs to. After at most $O(\log n)$ steps, all segments of S are removed, and each point $p \in P(h)$, for any h in the final range I_S, is located between two successive points of $\mathcal{I} \cup V$, as required. The overall cost of this step is $O(n^2 \log^4 n)$.

Let I_1 denote the final range I_S. For each $h \in I_1$ and for each $u \in V$, the set $\Pi(P(u, h), V)$ is now fixed, as is easily seen. This follows from the observation that whenever $\Pi(P(u, h), V)$ changes, some point in $P(h)$ must coincide with a point in $\mathcal{I} \cup V$. This allows us to run the left-to-right sweeping procedure, for each fixed $u \in V$, without having to fix the value of h, and obtain, for each swept vertex v, the set L_v of all the *active* lines pw, for $(p, w) \in \Pi(P(u, h), V)$, namely, the lines corresponding to those pairs (p, w) with w lying to the left of v. In the discrete problem, we need to find which of these lines attain the maximum height at v, and assert that this maximum height is no more than h^*; see below for more details. Handling the semi-continuous case is somewhat more involved, and will be discussed later.

To parallelize this step, we store the vertices of T at the leaves of a minimum-height binary tree τ. Each internal node ξ of τ represents the set $V(\xi)$ of all vertices stored at the leaves of the subtree

$\tau(\xi)$ rooted at ξ. Let $L(\xi)$ denote the set of all the lines pw, for $(p, w) \in \Pi(P, V)$, and $w \in V(\xi)$, and let $E(\xi)$ denote the upper envelope of the lines in $L(\xi)$. Note that $E(\xi)$ is an envelope of $|L(\xi)|$ lines, so its complexity is $O(|L(\xi)|)$. We have $\sum_{\xi\in\tau} |L(\xi)| = O(n \log n)$. Hence, the overall complexity of all the envelopes $E(\xi)$ is $O(n \log n)$, for a fixed first tower base u. We construct each of the envelopes $E(\xi)$ as follows. We dualize the respective lines pw to points in the dual plane, and run the parallel algorithm of [1] for constructing the upper convex hull of these (parametric) points. For a set of m points, this algorithm uses $O(m)$ processors and runs in $O(\log m)$ time. Each generic comparison that the algorithm performs is either between the x-coordinates of two dual points (that is, between the slopes of two input lines), or a left-turn test involving three dual points. Clearly, all the envelopes $E(\xi)$, over all the nodes ξ of τ and over all the first tower bases u in V, can be constructed in parallel, using a total of $O(n^2 \log n)$ processors and $O(\log n)$ parallel steps. This yields an overall parametric searching stage that runs in time $O(n^2 \log^4 n)$, as follows from the general time bound for parametric searching given above. Note that the output envelopes are still parametric and vary with h. However, their combinatorial structure is fixed (within the restricted subrange of h computed so far): For each envelope we know the sequence of lines (or, rather, pairs (p, w) defining those lines) that attain the envelope from left to right, and thus also know the nature of each breakpoint of the envelope.

We next apply another parallel stage, in which we locate, for each envelope $E(\xi)$ and each of its breakpoints q, the edge of T that lies vertically above or below q. We process these breakpoints in parallel, and for each breakpoint q we run a binary search with its (parametric) x-coordinate among the x-coordinates of the vertices of T. This stage uses $O(n^2 \log n)$ processors and runs in $O(\log n)$ parallel steps, so its overall cost is also $O(n^2 \log^4 n)$.

This initial part of the algorithm applies to both the discrete and the semi-continuous cases.

Proceeding with the discrete case is now easy: Keeping the first tower base u fixed, we process all the vertices v of T in parallel. For each vertex v, we obtain the set of vertices that lie to the left of v as the (disjoint) union of $O(\log n)$ subtrees of τ. For each of these trees $\tau(\xi)$, we locate v among the vertices of $E(\xi)$, using binary search. Since we have already located the vertices of each $E(\xi)$ among the vertices of T in their left-to-right order, these binary searches can be performed explicitly, without having to use generic parametric searching. Hence this step can be performed in overall $O(n^2 \log^2 n)$ time. We now have, for each pair u, v, a set of $O(\log n)$ lines, each attaining a respective subenvelope $E(\xi)$ at v, and we compute their maximum height at v. This is still a parametric step, which can be easily performed in parallel, using $O(n^2 \log n)$ processors and $O(\log n)$ parallel depth, so that its overall cost is, as above, $O(n^2 \log^4 n)$. To end the algorithm, we output a pair $u, v \in V$ for which the height computed at v, as described above, is no more than h. This step is also parametric, and can be performed within the time bound for the preceding step. It narrows down the range of h to its final value, and we terminate by returning the minimum of this interval (unless the optimum height h^* has already been detected in one of the comparison resolution steps). Hence we obtain:

Theorem 2.3. *The discrete two-watchtower problem for a terrain in $\mathbb{R}^2$ with n edges can be solved in $O(n^2 \log^4 n)$ time.*

We next handle the semi-continuous case. We proceed through the preceding stages, up to the point where all the envelopes $E(\xi)$ have been constructed, and their vertices have been located over the edges of T. We then proceed as follows. Consider a fixed first tower base u. For each edge $e = v'v$ of T, take its right endpoint v and obtain, as above, the set of vertices that lie to the left of v as the (disjoint) union of $O(\log n)$ subtrees of τ. We need to find the shortest vertical distance from e to the upper envelope of the $O(\log n)$ envelopes $E(\xi)$, over the corresponding subtrees $\tau(\xi)$. See Figure 7.

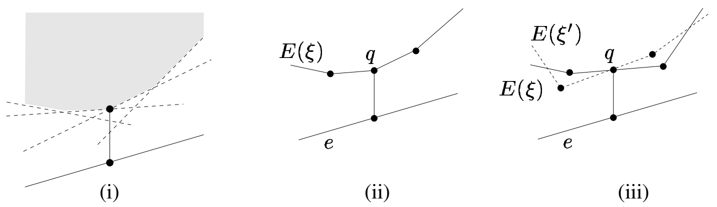

Figure 7. (i) The shortest vertical distance from an edge of T to the corresponding upper envelope. (ii) The shortest vertical distance from e to the envelope is attained at a vertex of a single subenvelope. (iii) The shortest distance is attained at an intersection point between two subenvelopes.

Note that the shortest vertical distance is attained (i) either at an endpoint of e, (ii) at a vertex q of one of the envelopes $E(\xi)$, or (iii) at a point q of intersection between two envelopes $E(\xi)$, $E(\xi')$. In case (ii), the edge of $E(\xi)$ incident to q and lying to its left (resp., right) must have slope smaller than (resp., larger than) that of e. In case (iii), q is incident to an edge of $E(\xi)$ and to an edge of $E(\xi')$, so that the slope of e lies in between the slopes of these two edges. See Figure 7 (iii).

Consider such a shortest vertical distance of type (iii). Both subenvelopes $E(\xi)$, $E(\xi')$ correspond to subtrees $\tau(\xi)$, $\tau(\xi')$ that are left children of nodes along the path in τ from the leaf storing v to the root. Without loss of generality, suppose that ξ is deeper in the tree than ξ'. Then if $\tau(\xi)$ is part of the output for a vertex v of T, $\tau(\xi')$ must also be part of that output. We will refer to this case by saying that ξ' is a *left great uncle* of ξ. In other words, we have argued that, independent of v, a subenvelope $E(\xi)$ can form intersection points of type (iii) with only $O(\log n)$ other subenvelopes $E(\xi')$ for which ξ' is higher in τ, and all these nodes τ' are left great uncles of ξ. See Figure 8.

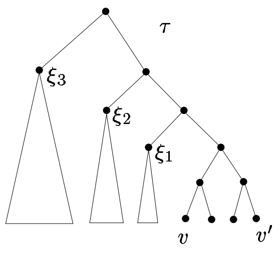

Figure 8. The tree τ. If a subtree $\tau(\xi_1)$ is part of the output for a vertex v, all subtrees that are left children of nodes on the path from ξ_1 to the root are also part of that output.

We therefore proceed as follows. We only fix the first tower base u, and obtain the corresponding set $P = P(u, h)$ and tree τ. We fix a pair (ξ, ξ') of nodes of τ such that ξ' is a left great uncle of ξ. We construct an implicit representation of the upper envelope of $E(\xi) \cup E(\xi')$, by merging the vertices and edges of $E(\xi)$ into $E(\xi')$, as follows. For each vertex q of $E(\xi)$, locate it among the vertices of $E(\xi')$, with respect to the x-coordinate, using binary search (which is parametric). Next, for each edge e of $E(\xi)$, find its intersections, if any, with $E(\xi')$; since $E(\xi')$ is a convex polygonal chain, e can meet it in at most two points, and they can be found using (parametric) binary search. We assign a processor to each vertex and edge of $E(\xi)$ for each such pair (ξ, ξ'). Since the overall size of all the envelopes $E(\xi)$ is $O(n \log n)$, and each has only $O(\log n)$ left great uncles, we need a total of $O(n \log^2 n)$ processors, and $O(\log n)$ parallel steps. We run this procedure for all first tower bases u in parallel, and thus use $O(n^2 \log^2 n)$ processors and $O(\log n)$ parallel steps, so the overall cost of this parametric searching step is, as above, $O(n^2 \log^4 n)$.

We next construct the set B of all breakpoints of the individual envelopes $E(\xi)$, of the breakpoints of envelopes of pairs of envelopes $E(\xi)$, $E(\xi')$, and of the vertices of T, over all the first tower bases u. The preceding analysis implies that the size of B is $O(n^2 \log^2 n)$. Using the same parametric binary search as in a previous stage, we locate the new breakpoints (of envelopes of pairs of envelopes) among the vertices of T. This step too takes $O(n^2 \log^4 n)$ time.

For each fixed first tower base u, we next process each edge e of T. We query in T with the right vertex v of e and obtain the set $L(v)$ of lines pw, as defined above, as the union of $O(\log n)$ disjoint subtrees of τ. Let $B(v)$ denote the subset of B consisting of the two endpoints of e, and of all the breakpoints of subenvelopes and of pairs of subenvelopes that correspond to the above $O(\log n)$ subtrees, with the additional requirement that they lie above or below e. Except for vertices of T, the sets $B(v)$ are pairwise disjoint, and their overall size is at most twice the size of B. All the sets $B(v)$, over all tower bases u, can be retrieved in $O(|B|) = O(n^2 \log^2 n)$ time (note that this step is no longer parametric).

With u and v fixed, we now assign a processor to each pair of a point $q \in B(v)$ and a subenvelope $E(\xi'')$ in the output of v. The overall number of processors is $O(n^2 \log^3 n)$. The processor assigned to q and $E(\xi'')$ has to determine whether q lies below $E(\xi'')$, which it can do using (parametric) binary search over the vertices of $E(\xi'')$. Hence, with $O(n^2 \log^3 n)$ processors and $O(\log n)$ parallel depth, we can collect all points $q \in B$ that are not hidden from above by another subenvelope. It is easily verified that, for each fixed u and v, among the points that pass these tests, exactly one point q has the property that the slope of e lies between the slopes of the two envelope edges incident to q. This surviving point yields the desired shortest vertical distance from e to the envelope. We find this point by testing the slope condition at each of the surviving points.

To end the algorithm, we need to test (parametrically) that among the surviving points there exists one for which the corresponding vertical distance is no more than h. This step narrows down the range of h to its final value, and we terminate by returning the minimum of this interval.

The overall running time of this algorithm is $O(n^2 \log^4 n)$, which leads to the following result.

Theorem 2.4. *The semi-continuous two-towers problem, for a terrain in $\mathbb{R}^2$ with n edges, can be solved in $O(n^2 \log^4 n)$ time.*

3 The Continuous Two-Tower Problem

In this section we consider the continuous version of the problem, where the two towers can be erected anywhere on the terrain. We first describe the decision procedure, in which we are given height h, and wish to determine whether there exist two towers of height h that together see the entire terrain. We give the full algorithm, including the parametric searching part, in Section 3.2.

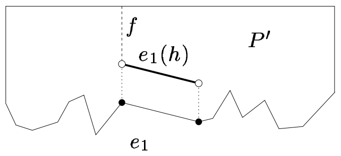

Figure 9. Creating a simple polygon P' from T and $e_1(h)$.

3.1 The decision procedure

Let e_1, e_2 be two fixed distinct edges of T, and consider the subproblem of determining whether there exist points $p \in e_1$, $q \in e_2$ such that the two tower tops $p(h), q(h)$ see together the entire T. We parametrize the locations of p and q by their respective x-coordinates s and t.

Let $e_1(h)$ denote the segment obtained by translating e_1 upwards by distance h. Compute the *visibility structure* of T from $e_1(h)$. To do this efficiently, since $e_1(h)$ is not an edge of T, we form from $e_1(h)$ and T a simple polygon P' in two steps. We first trim the halfplane above T to a bounded region, using two vertical segments and one horizontal segment, placed sufficiently far. Next we 'hook' $e_1(h)$ to the ceiling of the new region by a vertical segment f that connects an endpoint of $e_1(h)$ to the ceiling. This yields the desired simple polygon P', in which f and $e_1(h)$ are regarded as double edges. See Figure 9.

Applying the algorithm of [6] (see also [13]) to P', we obtain, in $O(n \log n)$ time, the visibility structure of P' from $e_1(h)$. This is not quite what we want, though, because f may block some visibility rays that emanate from $e_1(h)$ and otherwise reach T. We can overcome this problem by creating a second simple polygon P'', by hooking the other endpoint of $e_1(h)$ to the ceiling, and by computing the visibility structure from $e_1(h)$ in P''. Merging the two resulting visibility structures, we obtain the visibility of T from $e_1(h)$. More specifically, this involves a partition of $e_1(h)$ into $O(n)$ intervals, delimited by points that see two vertices of T along a common ray. We extend each such ray to the point where it first crosses T and enters the region below it. The sequence of all these crossing points that lie on a specific edge g of T is denoted by $\Sigma(e_1, g)$. If the line containing e_1 intersects T on g, we add this intersection point to $\Sigma(e_1, g)$. (This adds at most two points to the union of $\Sigma(e_1, g)$ over all $g \in T$.)

As the point p moves along e_1 from left to right, the corresponding tower top traces the segment $e_1(h)$. For each value s of the x-coordinate of p, denote the point p as $p(s)$, and the corresponding tower top as $p(s, h)$. Let g be another edge of T. The point $p(s, h)$ sees a portion of g which, if not empty, is delimited by the endpoint of g farthest from e_1, and by a point $z(s) = z_g(s)$ that moves continuously with p. As long as $p(s, h)$ does not cross a critical point of the visibility structure, $z(s)$ is either the endpoint of g nearest to e_1, or is the intercept of a visibility ray that emanates from $p(s, h)$ and passes through a fixed vertex v of T. When $p(s, h)$ crosses a critical point, and $z_g(s)$ crosses the matching point in $\Sigma(e_1, g)$, the 'pivot' vertex v may change, but the motion of $z(s)$ remains continuous.

The motion of $z_g(s)$ is also *monotone*. The direction of motion of $z(s)$ depends on whether g

lies to the left or to the right of e_1, and on whether the pivot vertex through which the segments $p(s,h)z_g(s)$ pass lies above or below the line containing $e_1(h)$. It is easily seen that when the pivot vertex changes, the new vertex continues to lie on the same side of the line through $e_1(h)$, and thus the motion of $z_g(s)$ continues in the same direction. See Figure 10 for some examples.

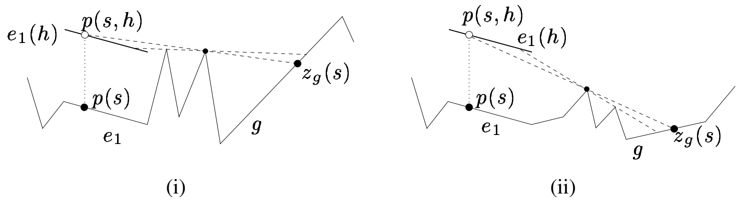

Figure 10. As $p(s,h)$ traces $e_1(h)$ from left to right, the corresponding point $z_g(s)$ moves to the right in (a), and to the left in (b).

Abusing the notation slightly, let us denote by $z_g(s)$ the x-coordinate of this point. Then the collection of functions $z_g(s)$, over all edges g of T, is a collection of continuous piecewise linear rational functions of s.

To see this, refer to Figure 11. Let a and b be two points on $e_1(h)$, and let c and d be the endpoints of the two corresponding visibility subintervals on g, where the visibility segments ac and bd pivot about the same vertex o of T. Regarding o as the origin, we can write $c = -\xi a$, $d = -\eta b$, for positive scalars ξ, η. Take a point $p = p(s,h)$ on $e_1(h)$, and let $z = z_g(s)$ be the corresponding endpoint of the subinterval of g visible from p. Write $p = \lambda a + (1-\lambda)b$, for $\lambda \in (0,1)$, and $z = \mu c + (1-\mu)d$, for $\mu \in (0,1)$. Since p, o, and z are collinear, the vectors $\lambda a + (1-\lambda)b$ and $\mu c + (1-\mu)d = -\mu\xi a - (1-\mu)\eta b$ are parallel and, since a and b are affinely independent, we must have

$$\frac{\mu\xi}{\lambda} = \frac{(1-\mu)\eta}{1-\lambda},$$

or

$$\mu\xi(1-\lambda) = (1-\mu)\eta\lambda,$$

or

$$\mu = \frac{\eta\lambda}{\xi(1-\lambda)+\eta\lambda}.$$

Since λ is a linear function of s and μ is a linear function of z, it follows that $z_g(s)$ is indeed a linear rational function of s, for the subinterval $[a,b]$ of $e_1(h)$.

The overall number of linear rational portions of these functions, over all edges g, is $O(n)$. Indeed, such a portion ends either at an endpoint of $e_1(h)$, or at a critical visibility point on $e_1(h)$ that sees two vertices of T along a common ray, and the number of such points on $e_1(h)$ is only $O(n)$.

We apply an analogous construction to the edge e_2, denote the point that moves along e_2 by $q = q(t)$, the corresponding tower top by $q(t,h)$, and the collection of functions that trace the (x-coordinates of the) endpoints of the visibility subsegments of edges g of T, as seen from $q(t,h)$, by $\{w_g(t)\}$. Let C denote the rectangle $e_1^* \times e_2^*$ in the st-plane, where e_1^*, e_2^* denote respectively the x-projections of e_1, e_2. Partition C, only for the purpose of analysis, into $O(n^2)$ subrectangles, by the vertical and horizontal lines corresponding to the x-coordinates of the critical points of the visibility structures on $e_1(h)$ and $e_2(h)$, respectively.

Fix an edge g of T, different from e_1, e_2. Let β_g denote the curve $z_g(s) = w_g(t)$, drawn in C. Within each subrectangle of C that β_g crosses, it is a hyperbolic arc. Indeed, by what we have just argued, the equation of β_g is of the form

$$\frac{\alpha_1 s + \beta_1}{\gamma_1 s + \delta_1} = \frac{\alpha_2 t + \beta_2}{\gamma_2 t + \delta_2},$$

for appropriate coefficients $\alpha_i, \beta_i, \gamma_i, \delta_i$, $i = 1, 2$. This equation can be rewritten as $Ast + Bs + Ct + D = 0$, and this defines a hyperbola in the st-plane.

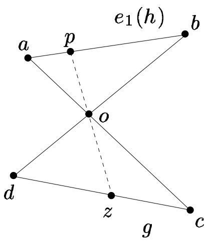

Figure 11. The relation between the point $p(s, h)$ on $e_1(h)$ and $z_g(s)$ on g. Here o is the pivot vertex of T.

Moreover, β_g is a connected curve which is both s- and t-monotone, and its endpoints lie on ∂C. Indeed, for each fixed s, there is at most one point t for which $w_g(t) = z_g(s)$, because the monotonously moving point $w_g(t)$ can sweep at most once through the stationary point $z_g(s)$. A symmetric argument applies to any fixed t. It is also easily seen that β_g cannot have an endpoint in the interior of C and that it remains connected when crossing from one subrectangle of C to another subrectangle.

The preceding argument implies that the number of hyperbolic arcs that constitute the curves β_g, over all edges g of T, is only $O(n)$. Indeed, an endpoint of such an arc that lies in the interior of the corresponding rectangle C is such that either its s-coordinate or its t-coordinate represents a critical visibility event on either $e_1(h)$ or on $e_2(h)$, and the overall number of such events is $O(n)$. Moreover, only one curve β_g can have such a transition point at the same s or t-coordinate, namely, the curve β_g whose edge g is hit by the critical visibility ray corresponding to the critical event. This implies the asserted linear bound on the number of pieces.

Each curve β_g thus partitions C into two portions, one of which, denoted MV_g, consists of all points (s, t) that represent placements of two towers on e_1 and e_2 that guard g, whereas the complement of MV_g consists of points representing placements of towers where not all of g is visible. Figure 12 exhibits four types of such portions of C. The classification depends (i) on the left-to-right order of the edges e_1, e_2, g, (ii) on whether the pivot vertex through which the segments $p(s, h)z_g(s)$ pass lies above or below the line containing $e_1(h)$, and (iii) on whether the pivot vertex through which the segments $q(t, h)w_g(t)$ pass lies above or below the line containing $e_2(h)$. As already remarked, when either of these pivot vertices changes, the new vertex continues to lie on the same side of the line through $e_1(h)$ or $e_2(h)$.

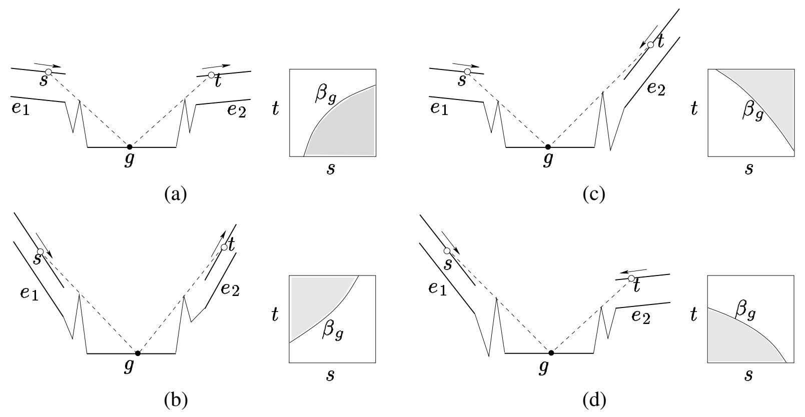

Figure 12. The four types of mutual visibility of an edge g from a pair of moving towers, and the corresponding st-regions where the entire g is guarded.

The decision procedure then continues as follows. Keeping the pair of edges e_1, e_2 fixed, we first construct, in linear time, the regions MV_g, or rather the curves β_g, over all the edges g of T. This is done by merging the sequences $\Sigma(e_1, g)$ and $\Sigma(e_2, g)$ into a common sequence $\Sigma(e_1, e_2, g)$. We then process the subintervals of g delimited by the points of $\Sigma(e_1, e_2, g)$. Each such interval I defines a piece of β_g, in which the common visible point $z_g(s) = w_g(t)$ moves in I. Clearly, only intervals I that lie in the range of both functions contribute nonempty portions to β_g. The computation of all the regions MV_g takes $O(n \log n)$ time for fixed e_1 and e_2.

Next, we compute the intersection $\bigcap_g MV_g$ of these regions. If this intersection is nonempty, any point (s, t) in it represents a placement of two towers on e_1 and e_2 that guard T. Conversely, if the intersection is empty, no such placement exists. Since each region is bounded by a curve that is s-monotone, $\bigcap_g MV_g$ is a *sandwich region* between the upper envelope of the curves β_g, for which MV_g lies above β_g (in the t-direction), as depicted in Figure 12(b,c), and the lower envelope of the curves β_g, for which MV_g lies below β_g, as depicted in Figure 12(a,d). Since any pair of hyperbolic arcs, with equations of the form $Ast + Bs + Ct + D = 0$, intesect at most twice, it follows that the complexity of either envelope, and thus also of the sandwich region, is $O(\lambda_4(n))$, and that it can be computed in time $O(\lambda_3(n) \log n)$, using the algorithm of Hershberger [15, 22]. We repeat this procedure for every pair e_1, e_2 of edges of T. We stop as soon as we find a pair for which the intersection $\bigcap_g MV_g$ is nonempty, and then report a corresponding placement of the two towers. If all these intersections are found to be empty, we report that no pair of towers of height h can see the entire T. Hence, the overall cost of the decision procedure is $O(n^3 \alpha(n) \log n)$.

3.2 Plugging the parametric search

As in the preceding discrete and semi-continuous problems, we solve the optimization problem by applying the parametric searching paradigm to the decision procedure just described. That is, we run it generically at the unknown optimal height h^*, parallelizing as much as possible comparisons that depend on h, and resolving them by binary search over the corresponding list of critical values of h at which some comparison outcome changes. While running the generic simulation, we progressively narrow down the range that contains the optimum h^*, and h^* is output either during one of the comparison resolution stages, or as the minimum of the final range.

For each vertex v of T, we compute the visibility structure from the vertical ray ρ_v emanating upwards from v. This takes a total of $O(n^2)$ time [13]. For each edge $e = uv$ of T, let τ_e denote the semi-unbounded trapezoid bounded by e, ρ_u and ρ_v. There are only $O(n)$ critical visibility rays, namely rays that pass through two vertices of T, which cross either ρ_u or ρ_v, and thus τ_e. We construct the arrangement of these rays, and clip it to within τ_e. With each vertex q of the arrangement we associate a critical height h, equal to the vertical distance of q from e. Repeating this procedure to each edge e, we obtain a collection of $O(n^3)$ critical heights. We add to these heights $O(n^2)$ additional critical heights, at which a shifted edge $e(h)$ becomes collinear with a vertex of T. We run a standard binary search, guided by the decision procedure, through these critical heights, to locate an initial interval I_0 between two successive critical heights that contains the optimum height h^*. This step takes a total of $O(n^3\alpha(n)\log^2 n)$ time. The visibility structure of $e(h)$, for any edge e, is now combinatorially determined, for any $h \in I_0$. That is, the nature of each critical event along $e(h^*)$, and their left-to-right order, are now determined and are the same for all $h \in I_0$.

In addition, for any pair of edges e, g, the critical visibility rays that emanate from points on $e(h)$, pass through a pivot vertex of T, and then hit g, are all determined combinatorially, and the order of their intercepts with g is fixed, for any $h \in I_0$. As above, we denote the sequence of these intercepts by $\Sigma(e, g)$. In the next step, we take each edge g, and collect the intercepts along g with the critical rays from all shifted edges $e(h)$. For each pair e_1, e_2 of edges, we merge the two sequences $\Sigma(e_1, g)$, $\Sigma(e_2, g)$ into a common sequence $\Sigma(e_1, e_2, g)$. We implement this step by a sorting network of depth $O(\log n)$. The total number of processors that are needed is

$$\sum_{e_1,e_2,g} \Big(|\Sigma(e_1, g)| + |\Sigma(e_2, g)| \Big) \le 2n \cdot \sum_{e_1,g} |\Sigma(e_1, g)| = O(n^3).$$

Hence, using Cole's improvement of the parametric searching method [9], this step takes a total of $O(n^3\alpha(n)\log^2 n)$ time.

In the next step, we process in parallel all pairs of edges of T, and construct, for each such pair e_1, e_2, the curves β_g, for all other edges g of T. Given e_1, e_2, and g, we process the subintervals of g delimited by the points of $\Sigma(e_1, e_2, g)$. Each such interval I defines a piece of β_g, in which the common visible point $z_g(s) = w_g(t)$ moves in I. We construct in parallel all these pieces, each with its explicit linear rational equation. Since $\sum_{e_1,e_2,g} |\Sigma(e_1, e_2, g)| = O(n^3)$, this step takes $O(n^3)$ time, and is in fact non-parametric.

In the next step, we process in parallel all pairs of edges of T, and construct, for each such pair (e_1, e_2), the sandwich region between the upper envelope of an appropriate subcollection of the curves β_g and the lower envelope of the complementary subcollection.

We compute the lower envelope of m hyperbolic arcs by running in parallel the standard divide-and-conquer algorithm for constructing envelopes (see [22]). The divide-and-conquer process has $O(\log n)$ parallel depth, and each stage of it requires merging two sequences of envelope break-points into a common sequence, which can be done in $O(\log n)$ parallel steps. Overall, the parallel depth is $O(\log^2 n)$, so this simulation, enhanced with Cole's technique [9], entails an overall cost of $O(n^3\alpha(n)\log^3 n)$. This also subsumes the cost of the subsequent stage that 'merges' the upper and the lower envelopes to produce the sandwich region between them.

This completes the description of the generic parallel simulation of the decision procedure, and thus yields the following result:

Theorem 3.1. *The continuous two-tower problem, for a planar terrain with n edges, can be solved in $O(n^3\alpha(n)\log^3 n)$ time.*

4 Guarding a Fixed Set of Points

Let T be a terrain with n vertices, and let $P = \{p_1, \ldots, p_m\}$ be a set of m points, all lying on T, and sorted in this left-to-right order along T. We assume that m is polynomial in n, so that $\log m = O(\log n)$. The goal is to place two towers $u(h), v(h)$ of the smallest possible height *anywhere* on T, so that they guard the entire set P. We show that this problem can be solved in time close to $O(mn)$.

We first develop a decision procedure that for a fixed h finds placements for two towers of height h over a terrain that together cover the entire set P, or determines that no such pair of towers exists, and then apply the paramatric search technique.

We denote by T_h the terrain T shifted up by distance h in the y-direction, such that each point (x, y) on T maps to the point $(x, y + h)$ on T_h.

4.1 The decision procedure

For each point $u \in T_h$, let $r(u)$ (resp., $t(u)$) denote the leftmost (resp., rightmost) point $p \in P$ that lies to the right of u and is not visible from u, or $+\infty$ (resp., $-\infty$) if there is no such point. For each point $u \in T_h$, let $P_u \subset P$ be the set of points that u does not see.

Lemma 4.1. *Let $u, v \in T$ with u to the left of v. Then $u(h)$ and $v(h)$ guard the set $P^+ \subseteq P$ of all points of P to the right of v if and only if $r(v) > t(u)$.*

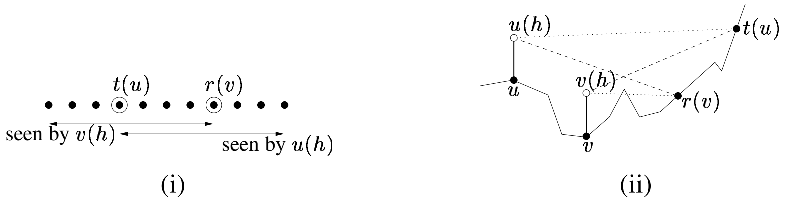

Figure 13. (i) Proof of the "if" part. (ii) The case $r(v) < t(u)$.

Proof: The 'if' part is trivial; see Figure 13 (i). Suppose then that $r(v)$ is not to the right of $t(u)$. The case $r(v) = t(u)$ is also trivial: neither $u(h)$ nor $v(h)$ sees this common point in P. Assume then that $r(v) < t(u)$. If $u(h)$ does not see $r(v)$ or if $v(h)$ does not see $t(u)$ then we are done: one of $r(v), t(u)$ is not guarded. Hence we may assume that $u(h)$ sees $r(v)$ and $v(h)$

sees $t(u)$; see Figure 13 (ii). Since $v(h)$ and $r(v)$ are not mutually visible, it easily follows that $v(h)t(u)$ lies counterclockwise to $v(h)r(v)$, and $r(v)u(h)$ lies clockwise to $r(v)v(h)$. Hence the segments $u(h)r(v)$ and $v(h)t(u)$ must intersect, from which it easily follows that $u(h)$ sees $t(u)$, a contradiction that completes the proof. $\square$

For each point $p_i \in P$, let $W_i = \partial W(p_i)$ denote the boundary of the visibility polygon of p_i with respect to the terrain T. As discussed above, W_i is a sequence of connected portions of T, interleaved with segments qr, where q is a vertex of T, $r \in T$, and the ray emanating from p_i towards q sees both q and r and crosses T at r. See Figure 14.

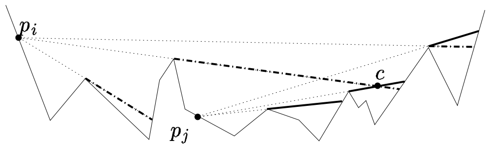

Figure 14. Two points p_i, p_j and their visibility polygons W_i and W_j. Only the right portions of these polygons are drawn. The portions $\overline{W}_i$, $\overline{W}_j$ of the boundaries of these polygons that lie strictly above T are drawn as dashed-dotted and solid, respectively.

Lemma 4.2. *Let s be a segment that lies fully above T. Then s intersects W_i at most once to the left of p_i and at most once to its right.*

Proof: Suppose without loss of generality that s lies fully to the left of p_i, and let q be a point in $s \cap W_i$. Then any point on s that lies above the line p_iq must be visible from p_i, as is easily checked. This implies that assertion of the lemma. $\square$

Let E_i be the set of points where T_h intersects W_i. By Lemma 4.2, each edge of T_h intersects W_i at most once, with the possible exception of the edge that lies directly above p_i, which may intersect W_i twice. It thus follows that $|E_i| \leq n + 1$. Let $E = \bigcup_{i=1}^{m} E_i$. Let u and v be two consecutive points of E along T_h. Clearly, from any point z on T_h that lies between u and v, we see the same subset of P. Therefore if there are two tower-tops on T_h that guard P, then there are two points in E that do the same. Our algorithm will therefore determine whether there exist two points $u, v \in E$ that guard P.

The algorithm first computes W_i, for $i = 1, \ldots, m$ using the algorithm of [13]. This takes a total of $O(mn)$ time. Then we compute the sets E_i, for $1 \leq i \leq m$, by traversing in parallel W_i and T_h from left to right, locating intersections between edges of T_h and W_i whose x-projections overlap. Since both T_h and W_i are connected x-monotone polygonal chains with $O(n)$ edges each, this step takes $O(n)$ time for each W_i, for a total of $O(mn)$ time. We organize E in a list sorted by x-coordinate, represented as a balanced search tree E^*.

We next compute the pointers $r(u)$ and $t(u)$ for every $u \in E$ by traversing the points in E from left to right along T_h, while maintaining the subset $P_u \subseteq P$ of points that are not visible from the current point u. We start out from $-\infty$, with $P_{-\infty} = \emptyset$. When we advance from a point $u \in E$ to the next point v on T_h, we add and/or delete a point to/from P_u to obtain P_v as follows. Let p_j be the point such that u is an intersection of W_j with T_h, and let p_k be the point such that v is an intersection of W_k with T_h. If to the right of u we cannot see p_j, then we add p_j to P_u. Similarly if to the right of v we see p_k then we delete p_k from P_u. Note that if $j = k$ and between u and v we cannot see p_j then $P_v = P_u$ and we do not have to delete and add p_j. After these updates we have the set P_v, and we set $r(v)$ (resp., $t(v)$) to be the leftmost (resp., rightmost) point in P_v which is to the right of v, or to $+\infty$ (resp., $-\infty$) if P_v is empty. We perform this computation in $O(mn \log n)$ time, by maintaining P_u in a search tree, sorted by increasing x-coordinate.

Let W_i^r (resp., W_i^ℓ) be the portion of W_i that lies to the right (resp., left) of p_i. It is easily checked that W_i^r has the following *order property*: Let $a, b \in W_i^r$, with a lying to the left of b. Then the slope of $p_i a$ is smaller than or equal to the slope of $p_i b$. (A symmetric property holds for W_i^ℓ, but we shall not use it.)

The core of our algorithm is based on the following lemma and its corollary.

Lemma 4.3. *Let $i < j$, and let u be the leftmost intersection point of W_i^r and W_j^r. Then to the left of u, W_j^r lies below W_i^r, and to the right of u, W_j^r lies above or overlaps with W_i^r.*

Proof: Both segments $p_i u$ and $p_j u$ are contained in the region of the plane bounded from below by T. We refer to this region as the closed halfplane lying above T. This implies that p_j lies below $p_i u$, unless $p_j = u$. In the former case, p_j is invisible from p_i and thus lies below W_i^r. The continuity of W_i^r and W_j^r implies that the entire portion of W_j^r between p_j and u lies strictly below W_i^r.

It therefore remains to consider the portion of W_j^r to the right of u, and to show that this portion lies above, or partially overlaps with, W_i^r. In other words, we need to show that every point $z \in W_j^r$ to the right of u is visible from p_i. To see this we note that, by the order property of W_j^r, $p_j z$ must lie counterclockwise to $p_j u$, which is easily seen to imply that the segments $p_i u$ and $p_j z$ intersect at some point q; see Figure 15. But then the polygonal chain $p_i q z$ lies above T, and thus $p_i z$ also lies above T, implying that z is visible from p_i, as claimed. This shows that, to the right of u, W_i^r lies below, or partially overlaps with, W_j^r.

The case when $p_j = u$ follows immediately from above. □

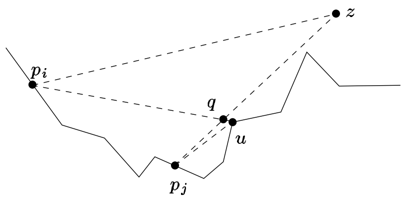

Figure 15. Points $z \in W_j^r$ to the right of u are visible from p_i.

As a consequence of Lemma 4.3, we obtain:

Corollary 4.4. *Let $p_{i_1}, \ldots, p_{i_k}$ be a subset of the points of P, ordered from left to right. Then the upper envelope of $W_{i_1}^r, \ldots, W_{i_k}^r$ is a concatenation of connected portions of $W_{i_1}^r, \ldots, W_{i_k}^r$, such that, for each $i = 1, \ldots, k-1$, the portion of $W_{i_j}^r$ precedes the portion of $W_{i_{j+1}}^r$. Some of the portions may be empty.*

Proof: We use the convention that when a point z belongs to several of the chains $W_{i_1}^r, \ldots, W_{i_k}^r$, we regard it as belonging to the chain with the largest index i_j. The assertion is now immediate from Lemma 4.3. □

For each point $u \in E$, recall that $P_u \subset P$ is the set of points that u does not see. As in the computation of $r(u)$ and $t(u)$, our algorithm traverses the points of E from left to right along T_h, while maintaining the sets P_u incrementally. For each point u that we traverse, we decide whether there is another point $v \in E$ *to the right* of u, such that all points in P_u are visible from v. To that end we also maintain the upper envelope $\mathcal{E}(u)$ of $\{W_i^r \mid p_i \in P_u\}$. By definition, any point of E on or above $\mathcal{E}(u)$ sees all the points of P_u *to its left.* Let $v \in E$ be the point on or above $\mathcal{E}(u)$ that lies to the right of u and has maximum $r(v)$. By Lemma 4.1, if $r(v) > t(u)$, then u and v cover the entire set P, and otherwise there is no point in E that can sees the entire P_u.

We maintain P_u as a binary search tree $\mathcal{T}$. A leaf in $\mathcal{T}$ corresponds to W_i^r, for some $p_i \in P_u$. Each internal node x of $\mathcal{T}$ represents the upper envelope $\mathcal{E}_x$ of the chains W_i^r, for all points p_i in its subtree. It follows that if r is the root of $\mathcal{T}$ then $\mathcal{E}_r = \mathcal{E}(u)$. For $x \in \mathcal{T}$, the envelope $\mathcal{E}_x$ is represented as a search tree $\mathcal{T}_x$. Each node $\alpha \in \mathcal{T}_x$ represents an edge e_α of $\mathcal{E}_x$. $\mathcal{T}_x$ is organized such that if we traverse it in symmetric order we obtain the edges on the envelope from left to right. We associate with each node $\alpha \in \mathcal{T}_x$ another search tree $\mathcal{T}_{x,\alpha}$, that represents the points of E that lie on or above the corresponding edge, ordered from left to right.

The representation of $\mathcal{T}_{x,\alpha}$ is as follows. Recall that the points of E are stored in a static search tree, sorted from left to right along T_h. T_h intersects e_α at points that belong to E. These points delimit contiguous subsequences of E, which alternate between lying above e_α and lying below it. We collect the subsequences that lie above e_α, and represent each of them by a leaf of $\mathcal{T}_{x,\alpha}$, where these leaves are sorted from left to right in $\mathcal{T}_{x,\alpha}$. Each such leaf simply stores pointers to the two nodes of the E-tree that delimit the corresponding subsequence. In addition, each leaf of $\mathcal{T}_{x,\alpha}$ stores the maximum value $r(v)$ of the points v in the subsequence of E that it represents, and the point v where this maximum is attained. Each internal node $\xi \in \mathcal{T}_{x,\alpha}$ stores the maximum value $r(v)$ of the points $v \in E$ which are above e_α in all the subsequences represented by the leaves of the subtree rooted at ξ, as well as the point v where this maximum is attained.

We propagate points with maximum $r(v)$ value further up in $\mathcal{T}_x$ as well. Each node $\alpha \in \mathcal{T}_x$ has, in the root of $\mathcal{T}_{x,\alpha}$, the maximum value $r(v)$ of the points $v \in E$ above e_α. We also store in each node $\alpha \in \mathcal{T}_x$ the maximum $r(v)$ value of a point $v \in E$ which is above any of the edges of $\mathcal{E}_x$ that reside in the subtree rooted by α.

To support updates, at each node of the tree representing all points in E, sorted along T_h, we also store the maximum $r(v)$ value of a point v in its subtree, together with the point v where this maximum is attained.

As we move from a point $u \in E$ to the next point $v \in E$ to the right along T_h, we update $\mathcal{T}$ so that it stores $\mathcal{E}(v)$ — the upper envelope of the functions W_i^r, for $p_i \in P_v$, in the manner just described. Let p_j be the point such that u is an intersection of W_j with T_h, and let p_k be the point such that v is an intersection of W_k with T_h. If to the right of u we cannot see p_j, then we have to add W_j^r to the tree $\mathcal{T}$. Similarly if to the right of v we see p_k then we delete W_k^r from $\mathcal{T}$. Note that if $j = k$ and between u and v we cannot see p_j then we can just leave $\mathcal{T}$ unchanged since $P_v = P_u$.

Consider $\mathcal{T}$ when we are at a point u as we move along T_h. The root r of $\mathcal{T}$ stores in $\mathcal{T}_r$ the upper envelope $\mathcal{E}(u)$ of W_i^r for all point $p_i \in P_u$. Let v be the point of E that (i) lies to the right of u, (ii) lies above all chains W_i^r, for $p_i \in P_u$ (which is equivalent to v seeing all the points of P_u to its left), and (iii) has the maximum value $r(v)$ among all such points. Then v can be retrieved from

$\mathcal{T}_r$ as follows.

Search in $\mathcal{T}_r$ with the x-coordinate of u. The node α where this search path ends stores the edge e_α of $\mathcal{E}(u)$ that lies vertically above or below u. We examine each of the subtrees of $\mathcal{T}_r$ that hang to the right of the search path to α, and extract from each of them the point v with maximum $r(v)$ among all the points of E that are stored in that subtree. Let v_1 be the point that attains the maximum r-value among all these points. Finally, we go to the tertiary tree $\mathcal{T}_{x,\alpha}$, and find there the leaf ξ that stores the subsequence E_ξ of $\mathcal{E}(u)$ that lies above e_α and contains u, if u does indeed lie above e_α, or else the subsequence E_ξ of $\mathcal{E}(u)$ that lies above e_α immediately to the right of u. In either case, we examine each of the subtrees of $\mathcal{T}_{x,\alpha}$ that hang to the right of the search path, extract from each of them the point v with maximum $r(v)$ among all the points of E that are stored in that subtree, and let v_2 denote the point with largest r-value among these points. If the subsequence E_ξ lies above e_α immediately to the right of u we extract the point $v_3 \in E_\xi$ with maximum r-value. If the subsequence E_ξ lies above e_α and contains u we search in the E-tree for the point v_3 of E_ξ that lies to the right of u and has maximum r-value. We now return the point v among v_1, v_2, v_3 that attains the maximum r-value.

By what has been argued above, we test whether $r(v) > t(u)$. If so, u and v cover the entire set P, and we stop and report them. Otherwise there is no point in E that, together with u, covers P, and then we move to the next point of E.

We now describe how to update $\mathcal{T}$. That is, we describe how to add a chain W_j^r to the set of chains that determine the upper envelope of $\mathcal{T}$, or how to delete a chain from that set. To add a chain W_j^r, we insert a new leaf into $\mathcal{T}$ using a standard INSERT operation for binary search trees. This insertion requires to update the envelopes of the nodes on the path in $\mathcal{T}$ from the root to the new leaf. We proceed to describe how to update these envelopes during an insertion. Deletion of a chain W_j^r from $\mathcal{T}$ is carried out analogously.

Let x be the left child and y be the right child of a node z in $\mathcal{T}$. To obtain $\mathcal{E}_z$ from $\mathcal{E}_x$ and $\mathcal{E}_y$, we first locate the leftmost intersection point b of $\mathcal{E}_x$ and $\mathcal{E}_y$. Then we split $\mathcal{E}_x$ at b into two connected pieces $\mathcal{E}_x^\ell$ and $\mathcal{E}_x^r$, containing the portions of $\mathcal{E}_x$ that lie to the left and to the right of b, respectively. Similarly, we split $\mathcal{E}_y$ at b into its left portion $\mathcal{E}_y^\ell$ and its right portion $\mathcal{E}_y^r$. By Corollary 4.4, $\mathcal{E}_z$ is the concatenation of $\mathcal{E}_x^\ell$ and $\mathcal{E}_y^r$.

We find the leftmost intersection b of the two envelopes $\mathcal{E}_x$ and $\mathcal{E}_y$ as follows. We first find the edge e of $\mathcal{E}_x$ that contains b. We start with the edge e' at the root of $\mathcal{T}_x$, the tree that represents $\mathcal{E}_x$, and check whether its endpoints are above or below $\mathcal{E}_y$. Each of these checks can be done in logarithmic time. If both endpoints of e' are above $\mathcal{E}_y$, we continue the search in the right child of e'. If both endpoints of e' are below $\mathcal{E}_y$ we continue with the left child of e'. If the left endpoint of e' is above $\mathcal{E}_y$ and the right endpoint of e' is below $\mathcal{E}_y$ then $e = e'$ and we stop the search. By Corollary 4.4, the symmetric positions of the endpoints of e' with respect to $\mathcal{E}_y$ are impossible. Handling cases where one or both endpoints of e' lie on $\mathcal{E}_y$ is done similarly, by regarding those endpoints as lying below $\mathcal{E}_y$. We locate the edge $f \in \mathcal{E}_y$ such that $b \in f$ analogously. After having found e and f, b is easily computed in constant time.

Next we describe how to split $\mathcal{E}_x$ and $\mathcal{E}_y$ at b. We implement each of these splits using a standard SPLIT operation on search trees. We also split at b the secondary trees $\mathcal{T}_{x,\alpha_e}$, $\mathcal{T}_{y,\alpha_f}$ associated with the edges e and f, respectively, that contain the points of E on or above these edges. The left part of the tree $\mathcal{T}_{x,\alpha_e}$ is associated with the portion of e on $\mathcal{E}_x^\ell$, and the right portion of the tree $\mathcal{T}_{y,\alpha_f}$ is associated with the portion of f on $\mathcal{E}_y^r$.

When we split a tertiary tree $\mathcal{T}_{x,\alpha_e}$ at b and b falls within the range of a leaf ξ of $\mathcal{T}_{x,\alpha_e}$, we have to split ξ into two leaves ξ_1 and ξ_2. Let the leaf ξ represent the sequence of points of E above T_h from e_1 to e_2. The leaf ξ_1 represents the subsequence of E from e_1 to the point of E just before b. The leaf ξ_2 represents the subsequence of E from the point right after b to e_2. The leaf ξ_1 is the rightmost in the left tree produced by the split of $\mathcal{T}_{x,\alpha_e}$, and the leaf ξ_2 is the leftmost in the right tree produced by the split of $\mathcal{T}_{x,\alpha_e}$. We find the maximum $r(v)$ value to store in ξ_1 by taking the maximum among the $r(v)$ values of the subtrees which are to the left of the search path to b and to right of the search path to e_1 in the tree containing all the points in E. Similarly, we find the maximum $r(v)$ value to store at ξ_2.

Finally we concatenate $\mathcal{E}_x^\ell$ and $\mathcal{E}_y^r$ to obtain $\mathcal{E}_z$, using a standard CONCATENATE operation on binary search trees. It is straightforward to maintain the maxima of the r-values while doing rotations in binary search trees. Since split and concatenate of binary search trees change the tree only via rotations, it is easy to maintain these maxima of the r-values during split and concatenate.

It is easy to see that each update (insertion or deletion of a chain W_j^r) to $\mathcal{T}$ takes $O(\log^3 n)$ time: (i) the search path in $\mathcal{T}$ visits $O(\log n)$ nodes, (ii) for each such node the corresponding paths in $\mathcal{T}_x$ and $\mathcal{T}_y$ have $O(\log n)$ nodes, and (iii) at each node in $\mathcal{T}_x$ (or $\mathcal{T}_y$) we need $O(\log n)$ time to perform the required computation (see above). Thus, the entire algorithm runs in $O(mn \log^3 n)$ time. That is, we have shown:

Theorem 4.5. *Let T be a terrain with n edges in the plane, P a set of m points lying on T, and $h > 0$ a parameter. We can determine, in $O(mn \log^3 n)$ time, whether there exist two points $u, v \in T$ such that each point of P is visible from either $u(h)$ or $v(h)$.*

4.2 Plugging the parametric search

Next we apply parametric searching to the decision procedure just described, to obtain an algorithm that finds the smallest height h^* of two towers that cover the entire set P. As usual, this is done by running a parallel generic simulation of the decision procedure at the unknown optimal height h^*, resolving comparisons that depend on h^* by finding their critical h-values and running a binary search through them, thereby progressively shrinking the range where h^* has to lie.

Fortunately, most of the steps in the decision procedure are *independent* of the value of h. Specifically, after E has been computed and sorted along T_h, all subsequent steps are independent of h, since they mostly maintain structures that depend on the visibility polygons $W(p_i)$, which do not depend on h. Hence, after the generic simulation reaches the stage where E has been computed and sorted, we can stop, take the smallest h in the current range, which we know to be equal to h^*, call the decision procedure with this specific h^*, and output the two resulting towers.

Since the visibility polygons W_i do not depend on h, we start the algorithm by computing them explicitly, in a total of $O(mn)$ time. We then process all the W_i's in parallel. For each W_i, we merge the sequence of its vertices, sorted in left-to-right order, with the similarly sorted sequence of the vertices of T_h. Note that the x-coordinates of the vertices of T_h coincide with those of the vertices of T, and thus are independent of h, so all the merges can be done explicitly.

Consider a fixed W_i, and let I be an interval between two consecutive vertices in the merged sequence. I is contained in the x-projection of a single edge g of W_i and of a single edge e of T_h. There are two critical values $h_i^- < h_i^+$, so that e and g intersect when $h_i^- \le h \le h_i^+$, and they are disjoint when h lies outside that range. By binary search over all these critical values (using, as usual, the decision procedure to guide the search), we identify all the points of E, each represented as the intersection point of a specific edge of T_h with a specific edge of some W_i. The total cost of this step is $O(mn \log^4 n)$.

Next, we need to sort the points of E from left to right. For this we use an $O(\log n)$-depth sorting network with $O(mn)$ processors, and apply the speed-up technique of Cole [9], to accomplish this step also in $O(mn \log^4 n)$ time. As argued above, we can terminate the parametric search after this stage, and complete the algorithm with a call to the decision procedure with the concrete value of h equal to the minimum of the feasible h-range.

In summary, we thus have:

Theorem 4.6. *Given a terrain T with n edges in the plane, and a set $P \subset T$ of m points, we can find, in $O(mn \log^4 n)$ time, two towers of smallest height that can be placed anywhere on T and together cover the entire set P.*

5 The Discrete Two-Tower Problem in 3-Space

Let T be a polyhedral terrain in $\mathbb{R}^3$ with n edges, and let $h > 0$ be a real parameter. Without loss of generality we can assume that each facet of T is a triangle. We wish to determine whether there exist two vertices u, v of T, so that the watchtowers $u(h), v(h)$ of height h erected at u, v guard the entire terrain, as defined in the introduction. We call any such pair (u, v) a *guarding pair* of T. Let V denote the set of vertices of T. For each $v \in V$ and a facet f of T, let $H_v(f)$ denote the portion of f that is invisible from $v(h)$; we call it the *invisibility region* of v in f. The regions $H_v(f)$ can be constructed as follows.

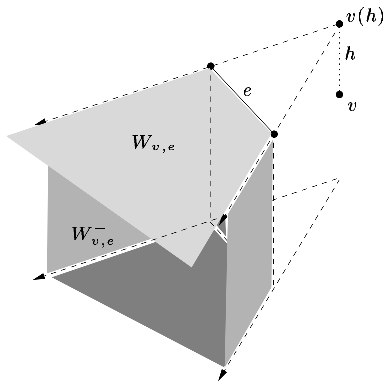

Figure 16. The truncated wedge $W_{v,e}$ and the prism $W^-_{v,e}$.

For each vertex $v \in V$ and for each edge e of T not adjacent to v, let $W_{v,e}$ denote the truncated planar wedge that is the union of all rays emanating from $v(h)$ and passing through e, minus the triangle spanned by $v(h)$ and e. Let $\pi_{v,e}$ denote the plane containing $W_{v,e}$, and let $W_{v,e}^-$ denote the unbounded prismatic region consisting of all points that lie vertically below $W_{v,e}$. The prism $W_{v,e}^-$ is bounded by $W_{v,e}$, and by three vertical walls (semi-unbounded vertical strips) bounded from above by the three edges of $W_{v,e}$. Informally, $W_{v,e}^-$ is the portion of $\mathbb{R}^3$ that is invisible from $v(h)$ if T degenerates to the single vertical wall bounded from above by e (see Figure 16).

For each vertex $v \in V$, let $\mathcal{W}_v$ denote the set of all wedges $W_{v,e}$, for edges e of T that are not adjacent to v. Let Π_v and $\mathcal{W}_v^-$ denote the set of all corresponding planes $\pi_{v,e}$ and prisms $W_{v,e}^-$, respectively. Put $\mathcal{W} := \bigcup_{v\in V} \mathcal{W}_v$, $\Pi := \bigcup_{v\in V} \Pi_v$, and $\mathcal{W}^- := \bigcup_{v\in V} \mathcal{W}_v^-$.

For each facet f of T and each vertex $v \in V$ not adjacent to f, we can construct $H_v(f)$ by intersecting each prism $W_{v,e}^- \in \mathcal{W}_v^-$ with f, and by taking the union of all the resulting regions. The complement $f \setminus H_v(f)$ is the *visibility region* of v (in f). The set of visibility regions, over all facets f of T, is the so-called *visibility map* from $v(h)$; its complexity is $\Theta(n^2)$ in the worst-case, and it can be computed in $O(n^2)$ time [22].

It is easy to establish the following properties of $H_v(f)$. The intersection of $H_v(f)$ with any vertical halfplane bounded by the vertical line λ_v through v, is a (possibly empty) line segment contained in f, and having one endpoint (the one nearer to v) on ∂f. As the halfplane rotates about λ_v, the other endpoint of the invisibility segment traces a polygonal path $\gamma_v(f) \subset f$, which is monotone with respect to the horizontal polar orientation θ of the halfplane about λ_v. The edges of $H_v(f)$ that lie in $\mathrm{int}(f)$ are portions of intersections of wedges $W_{v,e}$ with f. Moreover, $\gamma_v(f)$ can be interpreted as the upper envelope of these intersection edges, in an appropriate polar coordinate system within the plane containing f. Hence, the combinatorial complexity of $H_v(f)$ is $O(n\alpha(n))$ [22]. The overall complexity of all these regions, for a fixed vertex v, is $O(n^2)$, as each of its vertices corresponds to a vertex of the visibility map of T from $v(h)$; as noted above, all the invisibility regions $H_v(f)$, over all facets f of T, can be computed in $O(n^2)$ time.

By definition of $H_v(f)$, two vertices u, v of T form a guarding pair of T if and only if $H_u(f) \cap H_v(f) = \emptyset$ for every facet f of T, or equivalently, $H_v(f)$ is fully visible from $u(h)$ for every f. Moreover, as is easily checked, $H_v(f)$ is entirely visible from $u(h)$ if and only if $u(h)$ sees every point on its boundary $\partial H_v(f)$.

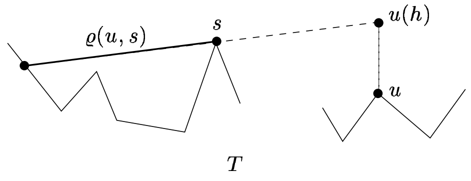

Figure 17. The segment $\varrho(u, s)$.

For a pair $u, s \in V$, if s is visible from $u(h)$, let $\varrho(u, s)$ be the segment whose endpoints are s and the first intersection point with T of the ray from s in direction $\overrightarrow{u(h)s}$, as illustrated in Figure 17. If s is not visible from $u(h)$, $\varrho(u, s)$ is not defined.

Lemma 5.1. *Let $u, v \in V$, and let f be a facet of T. $\partial H_v(f)$ is entirely visible from $u(h)$ if and only if the following two conditions hold:*

(V1) $u(h)$ sees every vertex of $H_v(f)$.

(V2) There does not exist a vertex $s \in V$ such that the segment $\varrho(u,s)$ passes vertically above an edge of $H_v(f)$.

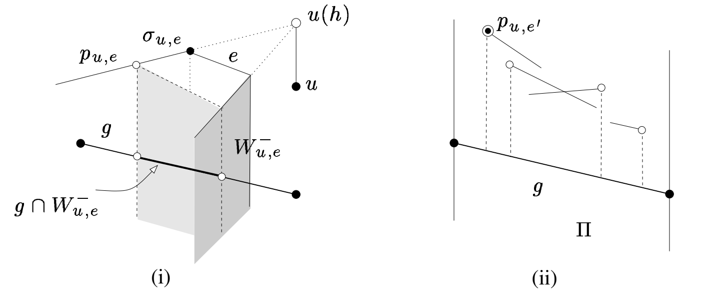

Figure 18. (i) Intersection of g and a prism $W^-_{u,e}$. (ii) The vertical strip Π and its intersection with the wedges $W_{u,e}$.

Proof. Indeed, if $u(h)$ sees the entire boundary $\partial H_v(f)$ then (V1) and (V2) hold. Conversely, suppose for the sake of contradiction that (V1) and (V2) hold but an edge g of $H_v(f)$ is not fully visible from $u(h)$. Let Π be the vertical strip containing g and bounded by the vertical lines passing through its endpoints. Since g is not fully visible from $u(h)$ it must intersect some prism $W^-_{u,e}$. Moreover, the endpoints of g do not lie inside $W^-_{u,e}$ because (V1) holds, therefore g intersects the vertical boundaries of $W^-_{u,e}$; see Figure 18 (i). Let $p_{u,e}$ be the point $W_{u,e} \cap \Pi$ of the largest vertical distance from g; $p_{u,e}$ lies on the ray emanating from an endpoint $\sigma_{u,e}$ of e in direction $\overrightarrow{u(h)\sigma_{u,e}}$. Among the prisms $W^-_{u,e}$ intersecting g let $W^-_{u,e'}$ be the one so that the (vertical) distance of $p_{u,e}$ from g is the largest; see Figure 18 (ii). Since no segments of $W_{u,e} \cap \Pi$ lies above $p_{u,e'}$, and thus $p_{u,e'}$ does not lie in any prism $W^-_{u,e}$, the ray $\overrightarrow{u(h)\sigma_{u,e'}}$ intersects the terrain after the point $p_{u,e'}$. Hence, the segment $\varrho(u, \sigma_{u,e'})$ passes vertically above g, thereby violating (V2). $\square$

Conditions (V1) and (V2) are equivalent to the following respective conditions.

(V1') There does not exist a wedge $W_{u,e}$, for an edge e of T not adjacent to u, such that $W^-_{u,e}$ contains a vertex of $H_v(f)$.

(V2') There does not exist a vertex $s \in V$ such that the xy-projections of the segment $\varrho(u,s)$ and of some edge of $H_v(f)$ intersect.

The equivalence of (V2) and (V2') follows from the fact that the segments $\varrho(u,s)$ lie above T, whereas all the edges of $H_v(f)$ lie on T. Recall that conditions (V1') and (V2') are formulated for a fixed triple u, v, f of two vertices and a facet of T. The algorithm has to determine whether there exists a pair u, v for which (V1') and (V2') hold for every facet f of T.

The algorithm proceeds in two stages. The first (resp., second) stage reports the set $\mathcal{N}_1$ (resp., $\mathcal{N}_2$) of all pairs $(u,v) \in V \times V$ for which condition (V1') (resp., (V2')) is violated for some facet f. Any pair of vertices $(u,v) \notin \mathcal{N}_1 \cup \mathcal{N}_2$ is a guarding pair of T, and thus constitutes a solution to the decision procedure. If all pairs of vertices are disqualified, the decision procedure has a negative answer.

Computing $\mathcal{N}_1$. We wish to report all pairs $(u, v) \in V \times V$ for which there exist an edge $e \in T$ and a facet $f \in T$ such that $W^-_{u,e}$ contains a vertex of $H_v(f)$. A vertex $\chi \in H_v(f)$ lies in $W^-_{u,e}$ if and only if (i) the xy-projection χ^* of χ lies inside the xy-projection $W^*_{u,e}$ of $W_{u,e}$, and (ii) χ lies below the plane $\pi_{u,e}$ containing $W_{u,e}$.

We fix a vertex $v \in T$ and report all pairs $(u, v) \in \mathcal{N}_1$ in two stages. Let Ξ be the set of vertices in $H_v(f)$, and let $\mathcal{W}$ be the set of all wedges $W_{u,e}$, as defined in the beginning of the section. The first stage reports all pairs in $\Xi \times \mathcal{W}$ that satisfy (i). The second stage reports a pair $(u, v) \in V \times V$ if there is a pair $(\chi, W_{u,e})$ reported in the first stage for which χ lies below $\pi_{u,e}$.

In more detail, let $\Xi^* = \{\chi^* \mid \chi \in \Xi\}$, and let $\mathcal{W}^* = \{W^* \mid W \in \mathcal{W}\}$. We have $|\Xi^*|, |\mathcal{W}^*| = O(n^2)$. Each W^* is an unbounded triangle in the xy-plane. Using a triangle range-searching data structure [17], we report, in $O(n^{8/3} \operatorname{polylog}(n))$ time, all pairs $(\chi, W) \in \Xi \times \mathcal{W}$ such that $\chi^* \in W^*$, as the disjoint union of complete subgraphs. That is, we report a family $\mathcal{F} = \{(\Xi_1, \mathcal{W}_1), \ldots, (\Xi_r, \mathcal{W}_r)\}$ where (i) $\Xi_i \subseteq \Xi$ and $\mathcal{W}_i \subseteq \mathcal{W}$, (ii) for any $(\chi, W) \in \Xi_i \times \mathcal{W}_i$, $\chi^* \in W^*$, and for every pair $(\chi, W) \in \Xi \times \mathcal{W}$ such that $\chi^* \in W^*$, there is an index i with $\chi \in \Xi_i$ and $W \in \mathcal{W}_i$. Moreover, $\sum_i (|\Xi_i| + |\mathcal{W}_i|) = O(n^{8/3} \operatorname{polylog}(n))$.

Fix a complete subgraph $(\Xi_i, \mathcal{W}_i) \in \mathcal{F}$. We preprocess Ξ_i, in $O(|\Xi_i| \log |\Xi_i|)$ time, into a data structure so that a halfspace-emptiness query (i.e., determine whether a query halfspace contains any point of Ξ_i) can be answered in $O(\log |\Xi_i|)$ time. This can be accomplished by constructing the Dobkin-Kirkaptrick hierarchy [11] on the convex hull of Ξ_i. For each $W_{u,e} \in \mathcal{W}_i$, we query the data structure with the halfspace $\pi^-_{u,e}$ lying below $\pi_{u,e}$. If $\pi^-_{u,e} \cap \Xi_i \neq \emptyset$, we add the pair (u, v) to $\mathcal{N}_1$. The total time spent over all complete bipartite graphs of $\mathcal{F}$ is $O(n^{8/3} \operatorname{polylog}(n))$. Repeating this procedure for all vertices $v \in T$, we construct the set $\mathcal{N}_1$ in $O(n^{11/3} \operatorname{polylog}(n))$.

Computing $\mathcal{N}_2$. Let $\mathcal{R}$ be the set of segments $\varrho(u, s)$ over all pairs $u, s \in V$ where s is visible from u. We compute $\mathcal{R}$ as follows. We fix a vertex $u \in V$ and preprocess T in $O(n \log n)$ time into a data structure, so that the first intersection point of T with a ray emanating from $u(h)$ can be computed in $O(\log n)$ time [22]. For each vertex $s \in V \setminus \{u\}$, we determine the first point ξ hit by the ray $u(h)s$. If ξ lies between s and $u(h)$ on the ray, $\varrho(u, s)$ is not defined; otherwise we set $\varrho(u, s) = s\xi$. We repeat this procedure for all vertices $u \in T$. The total time spent in this step is $O(n^2 \log n)$.

Fix a vertex $v \in V$. We compute in $O(n^2)$ time the visibility map of T from $v(h)$ and thus the set $\mathcal{E}$ of edges of all the regions $H_v(f)$; $|\mathcal{E}| = O(n^2)$. For a geometric object γ in $\mathbb{R}^3$, let γ^* denote, as above, its xy-projection. Set $\mathcal{R}^* = \{\varrho^* \mid \varrho \in \mathcal{R}\}$ and $\mathcal{E}^* = \{\gamma^* \mid \gamma \in \mathcal{E}\}$. Each of $\mathcal{R}^*$ and $\mathcal{E}^*$ is a set of $O(n^2)$ segments in $\mathbb{R}^2$. Using the algorithm described in [17], we compute, in $O(n^{8/3} \operatorname{polylog}(n))$ time, the set of all intersecting pairs $(\varrho^*, e^*) \in \mathcal{R}^* \times \mathcal{E}^*$ as the disjoint union of complete bipartite graphs, so that the overall size of their vertex sets is $O(n^{8/3} \log n)$.

For each complete bipartite subgraph $\mathcal{R}^*_i \times \mathcal{E}^*_i \subseteq \mathcal{R}^* \times \mathcal{E}^*$ in the output, we output all pairs (u, v), such that $\mathcal{R}^*_i$ contains (the projection of) a segment $\varrho(u, s)$, for some $s \in V$. The total cost of this step is $O(n^{8/3} \operatorname{polylog}(n))$, and we repeat it for each $v \in V$, to obtain an output collection $\mathcal{N}_2$ of all pairs $(u, v) \in V \times V$, with the property that the projection of some segment of the form $\varrho(u, s)$ intersects the projection of an edge of some $H_v(f)$. In view of the preceding discussion, no pair (u, v) in $\mathcal{N}_2$ is a guarding pair of T. The total cost of this step is $O(n^{11/3} \operatorname{polylog}(n))$.

Putting everything together, we can find in $O(n^{11/3}\,\mathrm{polylog}(n))$ time whether T can be guarded by two watchtowers of height at most h placed at two vertices of T.

The parametric search. To obtain the full algorithm, we apply parametric searching to the decision procedure just described. We briefly sketch the generic parallel implementation of this procedure. We first construct the invisibility regions $H_v(f)$. Each such region is computed as the upper envelope of $O(n)$ segments, which is easy to do in polylogarithmic parallel time (see the parametric searching steps for the 2-dimensional problems). Next we compute the segments $\rho(u,s)$. This can be done for each pair u, s in parallel, by considering a vertical planar cross-section of T through u and s (again, see the corresponding routines in the 2-dimensional problems). Both of these steps can be implemented in near-cubic time, and are thus far from being bottleneck steps.

Next, we simulate the construction of the sets $\mathcal{N}_1$ and $\mathcal{N}_2$. Each of these constructions uses a collection of two-level range searching structures, each of which can be constructed in polylogarithmic parallel depth; we omit details of these standard constructions, which are mostly routine. We can thus conclude the following.

Theorem 5.2. *For a polyhedral terrain in $\mathbb{R}^3$ with n edges, the discrete version of the two-watchtower problem can be solved in $O(n^{11/3}\,\mathrm{polylog}(n))$ time.*

6 Conclusion

In this paper we have presented efficient algorithms for many variants of the two-watchtower problem. There are of course many additional variants and extensions that could be studied, such as guarding a terrain with three or more guards, guarding with various visibility constraints or costs, maximizing the portion of the terrain that can be guarded by two (or any other number of) guards of a fixed height, guarding more general 3-dimensional polyhedral scenes, and so on.

The obvious immediate open problems are to improve the running time of the algorithms. In particular, (i) Can the 2-dimensional continuous version of the problem be solved in sub-cubic time? (ii) Can the 3-dimensional problem be solved by a faster algorithm? say, by a near-cubic algorithm?

The bottleneck in improving the algorithm for the 3-dimensional problem seems to be in the analysis of visibility along edges of the terrain: We have $O(n)$ edges, and each of them has $O(n)$ collections of invisibility intervals, where each collection is induced by some vertex of the terrain, and consists of $O(n)$ intervals. If we could find, in near-quadratic time, the set of all pairs of vertices that have a common point of invisibility along a fixed edge of the terrain, then we could have solved the whole problem in near-cubic time.

This 1-dimensional subproblem seems to be very basic, and is a special case of *generalized (or colored) intersection searching*, as studied by Gupta et al. and others (see the survey [14]). Recent progress on this problem has been made by Kaplan et al. [16], but it does not lead to an improved solution in our special setting. (It yields an algorithm with running time $O^*(n^{(5+\omega)/2}) \approx O(n^{3.688})$, where $\omega < 2.376$ is the exponent for matrix multiplication, which is just slightly worse than our solution.)

References

[1] A. Aggarwal, B. Chazelle, L. Guibas, C. Ó'Dúnlaing and C. Yap, Parallel computational geometry, *Algorithmica* 3 (1988), 293–328.

[2] B. Ben-Moshe, P. Carmi, and M. J. Katz, Computing all large sums-of-pairs in $\mathbb{R}^n$ and the discrete planar two-watchtower problem, *Inform. Process. Lett.* 89 (2004), 137–139,

[3] B. Ben-Moshe, M. J. Katz, and J.S.B. Mitchell, A constant-factor approximation algorithm for optimal terrain guarding, in *Proc. 16th Annu. ACM-SIAM Sympos. Discrete Algo.*, 2005, pp. 515–524.

[4] S. Bespamyatnikh, Z. Chen, K. Wang, and B. Zhu, On the planar two-watchtower problem, In *Proc. 7th Annu. Internat. Conf. Computing and Combinatorics*, 2001, pp. 121–130.

[5] B. Chazelle and L. Guibas, Visibility and intersection problems in plane geometry, *Discrete Comput. Geom.* 4 (1989), 551–581.

[6] D.Z. Chen and O. Daescu, Maintaining visibility of a polygon with a moving point of view, *Inform. Process. Lett.*, 65 (1998), 269–275.

[7] D.Z. Chen, V. Estivvill-Castro and J. Urrutia, Optimal guarding of polygons and monotone chains, *Proc. 7th Canadian Conf. Comput. Geom.*, 1995, pp. 133–138.

[8] K. Clarkson and K. Varadarajan, Improved approximation algorithms for geometric set cover, in *Proc. 21st Annu. Sympos. Comput. Geom.*, 2005, pp. 135–141.

[9] R. Cole, Slowing down sorting networks to obtain faster sorting algorithms, *J. ACM* 34 (1987), 200–208.

[10] R. Cole and M. Sharir, Visibility problems for polyhedral terrains, *J. Symbolic Comput.*, 7 (1989), 11–30.

[11] D. P. Dobkin and D. G. Kirkpatrick, Fast detection of polyhedral intersection, *Theoret. Computer Sci.* 27 (1983), 241–253.

[12] S.J. Eidenbenz, C. Stamm, and P. Widmayer, Inapproximability results for guarding polygons and terrains. *Algorithmica* 31 (2001), 79–113.

[13] L. Guibas, J. Hershberger, D. Leven, M. Sharir and R. Tarjan, Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons, *Algorithmica* 2 (1987), 209–233.

[14] P. Gupta, R. Janardan and M. Smid, Computational geometry: Generalized intersection searching, in *Handbook of Data Structures and Applications* (D. P. Mehta and S. Sahni, eds.), CRC Press, 2005, pp. 64.1–64.17,

[15] J. Hershberger, Finding the upper envelope of n line segments in $O(n \log n)$ time, *Inform. Process. Lett.* 33 (1989), 169–174.

[16] H. Kaplan, M. Sharir and E. Verbin, Colored intersection searching via sparse rectangular matrix multiplication, in *Proc. 22nd Annu. Sympos. Comput. Geom.*, 2006, to appear.

[17] J. Matoušek, Range searching with efficient hierarchical cuttings, *Discrete Comput. Geom.*, 10 (1993), 157–182.

[18] N. Megiddo, Applying parallel computation algorithms in the design of serial algorithms, *J. ACM* 30 (1983), 852–865.

[19] B. Nilsson, *Guarding art galleries: Methods for mobile guards*, Doctoral Thesis, Dept. Comput. Sci., Lund University, 1994.

[20] M.H. Overmars and J. van Leeuwen, Maintenance of configurations in the plane, *J. Comput. System Sci.* 23 (1981), 166–204.

[21] M. Sharir, The shortest watchtower and related problems for polyhedral terrains, *Inform. Process. Lett.*, 29(5) (1988), 265–270.

[22] M. Sharir and P.K. Agarwal, *Davenport Schinzel Sequences and Their Geometric Applications*, Cambridge University Press, New York, 1995.

[23] J. Urrutia, Art gallery and illumination problems, in *Handbook of Computational Geometry* (J.R. Sack and J. Urrutia, eds.), Elsevier Science Publishers, 2000, pp. 973–1026.

[24] B. Zhu, Computing the shortest watchtower of a polyhedral terrain in $O(n \log n)$ time, *Comput. Geom. Theory Appls.*, 8 (1997), 181–193.