Determining an Optimal Penetration among Weighted Regions in Two and Three Dimensions

Danny Z. Chen,* Ovidiu Daescu,*† Xiaobo (Sharon) Hu,‡ Xiaodong Wu,* and Jinhui Xu*
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556, USA
{chen,odaescu,shu,xwu,jxu}@cse.nd.edu

Abstract

We present efficient algorithms for solving the problem of computing an optimal penetration (a ray or a line segment) among weighted regions in 2-D and 3-D spaces. This problem finds applications in several areas, such as radiation therapy, geological exploration, and environmental engineering. Our algorithms are based on a combination of geometric techniques and optimization methods. Our geometric analysis shows that the optimal penetration problem in d-D ($d = 2, 3$) can be reduced to solving $O(n^{2(d-1)})$ instances of certain special types of nonlinear optimization problems, where n is the total number of vertices of the regions. We also give implementation results of our 2-D algorithms.

1 Introduction

In this paper, we study the following geometric optimization problem (called optimal penetration problem): Given a subdivision R with a total of n vertices in 2-D or 3-D space, divided in m regions R_i, $i = 1, 2, \ldots, m$, find a ray L such that L originates from outside R and intersects a specified *target region* $T \in \{R_1, R_2, \ldots, R_m\}$ and such that the weighted sum

$$S(L) = \sum_{L \cap R_i \neq \phi} w_i * f_i(L)$$

is minimized, where $f_i(L)$ is a function associated with the pair (R_i, L) and w_i is a positive weight factor associated with R_i. We call such a ray L a *penetration*. The regions R_i, $i = 1, 2, \ldots, m$, are all simple polygons (resp., polyhedrons) in the 2-D (resp., 3-D) space, and the weights of T and the complement $\overline{R}$ of R are zero ($\overline{R}$ is the free space outside R). Let R_L denote the set of all regions of R intersected by a ray L and let d_i denote the length of L within $R_i \in R_L$. (The lengths of line segments are all based on the L_2 metric.) We consider two versions of this problem. For the first version (P1), $f_i(L)$ is either d_i or zero, depending on whether R_i is passed by L before or after L intersects the target region T (i.e., this models the case in which the ray stops when it hits the target). For the second version (P2), $f_i(L) = d_i$ for all $R_i \in R_L$. See Figure 1 for an example.

The problem we study finds applications in several areas such as radiation therapy, geological exploration, and environmental engineering. For example, in radiation therapy, the subdivision R may represent a portion of a human body while the regions of R may represent various organs within the human body. Different organs may have different characteristics. A region containing a tumor may correspond to the target region and should be treated by strong radiation, while healthy organs should suffer minimum exposure to radiation (various organs may have different degrees of tolerance

*This research was supported in part by the National Science Foundation under Grant CCR-9623585.

†The work of this author was supported in part by a fellowship from the Center for Applied Mathematics, University of Notre Dame, Notre Dame, Indiana, USA.

‡This research was supported in part by the National Science Foundation under Grant MIP-9701416 and by HP Labs, Bristol, England, under an external research program grant.

SCG'99 Miami Beach Florida

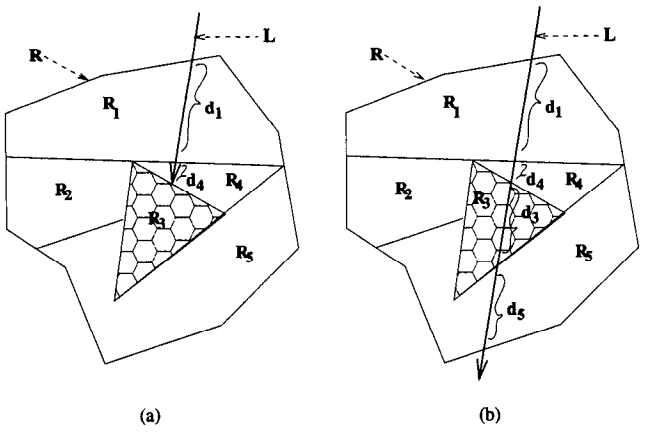

Figure 1: Illustrating the problems: (a) L "probes" (hits) the target region R_3 (P1); (b) L penetrates R_3 (P2).

to radiation, as indicated by their weight factors). Problem P1 can be used to model the process of delivering radiation by direct implantation of a radiation source into the target [13] (*branchytherapy*), while problem P2 is closely related to the use of a penetrating radiation ray (radiation beam) directed at the target from an external source [13] (*teletherapy*). Problem P1 also models the process of finding an optimal "probe" to a specified region T in the setting of R. Such a problem may appear in applications such as mineral exploration.

In radiation teletherapy, usually, three parameters need to be optimized: beam fluence distribution (the dose distribution to be delivered to the patient), beam energy (the radiation field that provides the dose distribution), and beam directions. Most efforts in the field have concentrated on optimizing the first two parameters for prespecified beam directions. In general, global optimization methods and probabilistic methods (e.g., simulated annealing, genetic algorithms) are used to compute the (approximate) optimal values for these two parameters. However, as pointed out in [18], in practice the optimal choice of beam directions is one of the most difficult problems of medical treatment optimization, and there should be as great an effort expended on optimizing beam directions [12]. While there are quite a few 2-D and 3-D results [5, 6, 7, 8, 9, 10, 11, 12, 13, 18, 19, 22, 23, 24, 26, 29, 34, 35] on treatment optimization with respect to one of the first two parameters, or even both, optimizing the beam directions has remained an elusive goal. The few papers which consider the optimization of the beam directions discretize the problem and use brute force methods (i.e., trying a large number of combinations of directions and selecting the best). In [13], a discretized ray space (i.e., a rectangular grid of points) and a two dimensional function (the 2-D case), defined on the grid of points in the discretized ray space, are used to obtain an approximate optimal beam direction. A discretized model is also used in [18] to optimize with respect to multiple criteria, including the beam direction. Clearly, such approaches do not guarantee yielding a true optimal solution.

In [32], the authors use a different approach for finding all feasible beam directions for a special 3-D case of problem P2. For this case, the target region is modeled as a 3-D ball, the central axis of each beam is required to go through the center of the target region ball, and the healthy organs under consideration are all treated as obstacles to the radiation beams. Their approach is based on computational geometry techniques developed for robot motion planning problems [21] and uses topological sweep [17] to compute the arrangement of $O(n)$ great circle arcs on a 3-D sphere.

In computational geometry, there are results in the context of computing approximate geodesic shortest paths among weighted regions [1, 20, 25, 27] that are somewhat related to the optimal penetration problems. Although the geodesic shortest path problem and our optimal penetration problem both deal with weighted regions, there are several different aspects between these problems: (1) while a geodesic shortest path connects two specified points, an optimal penetration may connect any two points in two specified regions, and (2) while a geodesic shortest path is in general a polygonal line, a penetration can only be a ray or a line segment. Due to such differences, the techniques that we use in solving the optimal penetration problem are quite different from those of [1, 20, 25, 27].

Our work represents a step towards solving the optimal beam direction problem (the general beam direction problem may involve multiple beams and may be tangled with other constraints and optimization parameters such as dose distribution and

beam energy).

Our main results are as follows. We solve both P1 and P2 in 2-D and 3-D spaces. We show that the ray space for the 2-D (resp., 3-D) cases can be partitioned into a set S_C of $O(k)$ cells, where k is $O(n^2)$ (resp., $O(n^4)$) in the worst case, such that the function $S(L)$ to be minimized has a similar description for all the rays L corresponding to any such cell. To obtain the cells in the ray space, we transform P1 and P2 to certain visibility problems. For each cell $C \in S_C$, we find a ray L that optimizes $S(L)$ over C, and thus the sought solution is a ray that gives the smallest value for $S(L)$ over all cells in S_C. We present output-sensitive algorithms for solving P1 and P2 in 2-D, which take $O(k)$ time and $O(n)$ space to generate the cells of S_C, where k is $O(n^2)$ in the worst case. Further, we present an output-sensitive $O((n^3 + k)\log n)$ time algorithm for constructing the cells of S_C for P1 and P2 in 3-D, where k is $O(n^4)$ in the worst case. The function to be optimized over each cell, which in general is non-linear and non-convex, has a quite complicated form. Hence we use an optimization software to compute the optimal solution for each cell. We have implemented our $O(n^2)$ time, $O(n)$ space algorithms for the 2-D case in LEDA, and provide an analysis of the overall performance of the algorithms.

2 The 2-D Case

In this section, we present our algorithms for solving problems P1 and P2 in 2-D. We divide each problem into two subproblems: (1) partition the ray space into a set S_C of cells, such that the function $S(L)$ to be minimized has a similar description for all rays corresponding to any such cell, and (2) for each cell $C \in S_C$, find a ray that optimizes $S(L)$ over C.

Since our approaches are based on geometric techniques such as the visibility complex, topological walk, and topological sweep, we first review some useful structures.

Let $H = \{l_1, l_2, \ldots, l_n\}$ be a set of n straight lines on a plane. The lines in H partition the plane into a subdivision, called the *arrangement* $A(H)$ of H, that consists of a set of convex regions (cells), each bounded by some line segments of the lines in H. In general, $A(H)$ consists of $O(n^2)$ cells, edges and vertices. Edelsbrunner and Guibas [17] introduced a technique called *topological sweep* that allows to construct and report $A(H)$ in $O(n^2)$ time and $O(n)$ space, by sweeping the plane with an unbounded simple curve that is monotone with respect to the y-axis and that intersects each line of H exactly once. The technique was later extended to 3-D by Anagnostou, Guibas and Polimenis [2]. Asano, Guibas and Tokuyama [3] developed another approach, called *topological walk*, for constructing and reporting $A(H)$ in $O(n^2)$ time and $O(n)$ space. Essentially, a topological walk traverses $A(H)$ in a depth-first search fashion [3, 4].

For a geometric scene with "opaque" objects, the visibility computation involves determining the objects seen along certain directions of vision (i.e., rays). Pocchiola and Vegter [28] introduced a data structure, called *visibility complex*, representing collections of line segments (or rays) in the free space of the scene that have the same visibility properties. They gave an output-sensitive $O(n \log n + k)$ time and $O(k)$ space algorithm, where k is the complexity of the visibility complex and can be $O(n^2)$ in the worst case, to construct the visibility complex of a scene consisting of n convex curved objects. Rivière [31] presented an output-sensitive $O(n \log n + k)$ time and $O(n)$ space algorithm, where $k = O(n^2)$ in the worst case, that optimally constructs the visibility complex of a polygonal scene with a total of n vertices.

Let P denote the 2-D plane and consider the duality transform [30] that maps a line $l : y = mx + p$ on P onto the point $l^* : (m, p)$ on the dual plane $\overline{P}$. Using this duality, the set of lines passing through a point $q : (a, b)$ on P corresponds to the line $q^* : y = -ax + b$ on $\overline{P}$ [30].

For a ray L on P, we define a *point order* $<_L$ on L such that, for two distinct points p and q on L, $p <_L q$ if and only if (iff) the ray originating at p and passing through q has the same direction as L. Let L_S be the set of line segments of the subdivision R intersected by L. We say L_S is sorted if the sequence of segments in L_S is consistent with the point order of their intersection points with L along L (i.e., let $p_i = L \cap s_i$ and $p_j = L \cap s_j$ for two segments s_i and s_j in L_S; then $p_i <_L p_j$ iff s_i appears in L_S before s_j). When L changes (rota-

tion and/or translation), the segment sequence in L_S changes only when a segment is removed from L_S or a new segment is inserted into L_S. In these situations, we have an *event* that updates L_S. As long as no event occurs while changing L, we say that the function $S(L)$ has a *similar description* (the exact description of $S(L)$ is given later).

Our algorithms for problems P1 and P2 generate collections of rays such that for each ray collection (cell), $S(L)$ has a similar description. This is achieved by efficiently updating the L_S sequences to produce the ray collections one by one.

2.1 A Visibility Approach

In this subsection, we show that the computation of the cells in the ray space (on each of the cells $S(L)$ has a similar description) can be done by transforming P1 and P2 into visibility problems. Since the transformations for P1 and P2 are similar, we mainly describe the transformation for P2.

Let P be the 2-D plane containing the subdivision R and let S_R be the set of line segments bounding $\{R_1, R_2, \ldots, R_m\}$. Let L_C be the set of rays that originate in the free space $\overline{R}$ outside R, hit the target region T, and intersect a same subset L_S of line segments of R such that the segments in L_S are in the same point order along every $L \in L_C$.

Observation 1 *The union of rays in L_C forms an hourglass in the plane P and a convex cell in its dual plane $\overline{P}$, such that each line segment on the boundary of the hourglass (resp., each vertex of the cell) is defined by a line passing through two vertices of the subdivision R.*

For a segment $s \in L_S$, let s_l and s_r be the left and right endpoints of s, and l_s be the line containing s. On plane P, we replace each segment $s \in L_S$ by two semilines on l_s (see Figure 2 (a) and (b)), such that the first semiline has its right endpoint at s_l and the second one has its left endpoint at s_r (i.e., removing s from l_s gives the two semilines). Let L'_S denote the set of semilines defined by L_S in this manner. The hourglass bounding L_C remains unchanged after this transformation, but L_C now corresponds to a set of rays with the same backward and forward views. That is, if we assume that R is enclosed by a large circle C_R and consider $(S_R \setminus L_S) \cup L'_S$ as opaque objects, then each

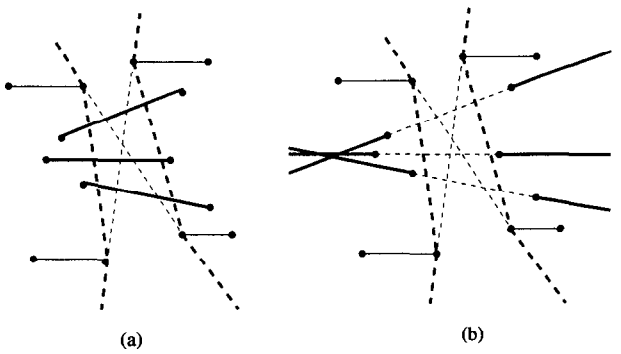

Figure 2: (a) An hourglass (bold dashed line) corresponding to the set of rays intersecting three line segments (bold), and (b) transformation to a visibility problem.

ray $L \in L_C$ originates on C_R, goes through the free space of the hourglass, and ends on C_R (we may refer to C_R as an object at infinity). Using the duality transformation, the hourglass for L_C on P is mapped onto a convex cell C on $\overline{P}$ [31].

Observation 2 *The cell C corresponds to a face of the visibility complex of $(S_R \setminus L_S) \cup L'_S$ with C_R as forward and backward views.*

Similarly, for problem P1, a cell C on $\overline{P}$ corresponds to a face of the visibility complex which has C_R as backward view and a line segment on the target region as forward view.

For a cell C and a ray $L \in C$, let L intersect the region set $R_L = \{R_1, R_2, \ldots, R_{k-1}\}$, and let $L_S = \{s_1, s_2, \ldots, s_k\}$ such that L_S is sorted in the point order. For problem P2, we have $S(L) = \sum_{i=1}^{k-1} w_i * d_i$, where d_i is the length of L inside R_i (i.e., the Euclidean distance along L between the two consecutive segments s_i and s_{i+1} of L_S that are on the boundary of R_i). Clearly, the distance d_i depends on the descriptions of s_i, s_{i+1}, and the ray L. Note that the descriptions of s_i and s_{i+1} are fixed while that of L is not. Hence, s_i and s_{i+1} contribute only to the constant coefficients of the expression for d_i. If we parameterize L by, say, its slope m and intercept p of the y-axis, then m and p are the only variables in the expression of d_i. Thus $S(L)$ has the same expression (or similar description) on all rays of C.

Therefore, our algorithms for solving P1 and

P2 have the following outline: (1) Generate a cell C and compute L_S, where L is any ray in C; (2) optimize $S(L)$ over C (i.e, find $L \in C$ such that $S(L)$ is minimized); (3) compute a new cell while updating L_S accordingly, and go to step (2), until we are done.

The next two subsections show how to generate the cells while efficiently maintaining L_S, and discuss a few different ways for generating the instances of the special optimization problem.

2.2 Computing the Cells of S_C

Let S_C denote the set of cells in the ray space $\overline{P}$ such that for each cell $C \in S_C$, $S(L)$ has a similar description over C, and the rays in C all intersect the target region T. Let S_T denote the set of segments on the boundary of T.

The visibility approach introduced in Subsection 2.1 appears to suggest that, for each cell $C \in S_C$, we need to identify its corresponding segment set L_S and perform a visibility transformation to obtain L'_S, before being able to compute C. But that would not yield an efficient approach for computing the cells of S_C. Instead, observe that, essentially, only the endpoints of the line segments in S_R are actually involved in the visibility transformation. Thus, to compute S_C for problem P1, for each segment $s \in S_R \setminus S_T$, we discard the interior of s and treat its endpoints s_l and s_r as point obstacles. We further consider each line segment in S_T as a distinct obstacle in the transformed scene. Then, we can obtain S_C for P1 by computing the faces F of the visibility complex of the transformed scene that have the object at infinity as the backward view and some line segment in S_T as the forward view. To compute S_C for P2, we treat the endpoints of all segments $s \in S_R$ as point obstacles and compute the faces F of the visibility complex of the scene that have the object at infinity as both the backward and forward views. We can then consider each face $f \in F$ such that every point $p \in f$ dualizes to a line that penetrates the target region T.

The cells of S_C for both P1 and P2 can be obtained in $O(n \log n + k)$ time and $O(n)$ space, where $k = O(n^2)$ in the worst case, by using Rivière's visibility complex algorithm [31]. Observe that we need not maintain explicitly the whole visibility complex of the transformed scene. Instead, we only need to produce the faces of the complex as they are used by our computation. Once the computation on a face is done, that face is discarded.

There are other ways to construct S_C, and in fact we choose to use one of these approaches in the implementation of our algorithms. That approach is based on the following observation.

Observation 3 *The cells of S_C are a subset of the cells of the arrangement for the lines on $\overline{P}$ that are the duals of the vertices of the subdivision R.*

Based on the observation above, we dualize the vertices of R and obtain a set of n lines $H = \{l_1, l_2, \ldots, l_n\}$ on $\overline{P}$. Let $A(H)$ denote the arrangement of these lines. It is well known that each cell of $A(H)$ corresponds to a set of lines that intersect a same sequence of line segments of R on P. Thus, we can use topological walk [3, 4] to compute the cells of $A(H)$ and to identify those cells C such that each point $p \in C$ dualizes to a line that penetrates T. A similar result can also be obtained by using topological sweep [17].

The algorithms for constructing the visibility complex, as well as the ones based on topological walk and topological sweep, allow us to maintain the sequence L_S (and its point order) of intersected segments in amortized constant time per cell over all cells in $A(H)$ [3, 17, 31]. Then, we have the following lemma.

Lemma 1 *The set S_C of cells for problems P1 and P2 in 2-D can be computed in $O(n^2)$ time and $O(n)$ space. The sequences of line segments of R intersected by a line L in each cell $C \in S_C$ can also be computed in $O(n^2)$ time and $O(n)$ space, for all cells of S_C.*

2.3 Generating Optimization Problem Instances

In this subsection, we present and analyze our algorithm for generating the function $S(L)$ that is to be minimized over each cell $C \in S_C$, for both P1 and P2.

Let $R_L = \{R_1, R_2, \ldots, R_{k-1}\}$ and $L_S = \{s_1, s_2, \ldots, s_k\}$, such that L_S is sorted in the point order and the segments s_i and s_{i+1} of L_S are on the

boundary of R_i, for $i = 1, 2, \ldots, k-1$. For P2, the function $S(L)$ to be minimized over a cell C has the form $S(L) = \sum_{i=1}^{k-1} w_i * d_i$, where d_i is the length of L inside R_i. Thus, to compute $S(L)$ on C, we need to compute the distances d_i, $i = 1, 2, \ldots, k-1$, that change only when $L \in C$ changes. Our goal is to obtain an expression for $S(L)$ whose form is as simple as possible, so that its computation is efficient and stable. Without loss of generality (WLOG), we assume that R is in the first quadrant of the plane P.

A straightforward approach to compute d_i is to parameterize L by its slope m and intercept p of the y-axis on the plane P. When L is specified by equation $y = mx + p$, these two parameters define L's dual point $(m, p) \in C$ on $\overline{P}$, and the feasible region for $S(L)$ is exactly the convex cell C. The Euclidean distance between the two consecutive intersection points of L with the boundary of R_i, lying on segments s_i and s_{i+1} of L_S, is d_i (see Figure 1). Thus, we obtain the following form for $S(L)$:

$$S(L) = \sqrt{1+m^2} \sum_{i=1}^{k-1} w_i |x_{i+1} - x_i|$$

After computing the form for x_i, we have:

$$S(L) = \sqrt{1+m^2} \sum_{i=1}^{k-1} w_i \left| \frac{p_{i+1} - p}{m - m_{i+1}} - \frac{p_i - p}{m - m_i} \right|$$

where x_i (resp., x_{i+1}) is the x-coordinate of the intersection point v_i (resp., v_{i+1}) of L and s_i (resp., s_{i+1}), and m_i and p_i (resp., m_{i+1} and p_{i+1}) are the slope and intercept of the function defining the line l_{s_i} (resp., $l_{s_{i+1}}$).

One drawback of this form is that the computation of $S(L)$ may not be very stable in certain situations. The instability may occur when the cell C contains a line L that is (almost) vertical. Thus, if the sought optimal line lies in a narrow sector that contains such a line L, it may be difficult to capture the line correctly by common optimization software, due to numerical errors. Another drawback of this form is that it applies the absolute-value function to each term of the summation, since the sign of each term may change over the feasible domain C. Therefore, function $S(L)$ in this form is in general not smooth and may not be suitable for optimization techniques and software that expect smooth objective functions. Certainly, there are optimization approaches that also accommodate non-smooth functions, but it is likely to yield a lesser implementation performance. The main advantage of this expression is the simple form for $S(L)$ (i.e., each term in the summation is a linear fractional).

The following lemma suggests a way to remove the absolute-value function in $S(L)$.

Lemma 2 *Let $List_C$ be the list $x_1, x_2, \ldots, x_k$ of the x-coordinates of the k intersection points of L and L_S in the point order. Then, $List_C$ is either an increasing list or a decreasing list, and the monotonicity changes when L sweeps through a vertical line.*

Note that the points of the dual plane $\overline{P}$ use m and p as their two coordinates. Based on Lemma 2, splitting the cell C on $\overline{P}$ along the dual curve of the vertical lines on P yields at most two convex subregions C_1 and C_2, and in each subregion C_r, $r = 1, 2$, the slopes of all penetration lines have the same sign. Thus, in each such subregion, the absolute-value function can be removed from the corresponding objective function, and the new form of $S(L)$ is as follows:

$$S(L) = \sqrt{1+m^2} \sum_{i=1}^{k} \frac{a_i p + b_i}{m - m_i}$$

where a_i and b_i are constants that depend on w_{i-1}, w_i, and p_i.

As the reader may have observed, the duality transformation proposed above does not dualize vertical lines. However, this cell splitting is implicitly done by the dualization process (i.e., if the set of lines on P for a given segment sequence L_S includes vertical lines, then its dual is a pair of cells in $\overline{P}$). Let one such cell be denoted as C_V, and observe that there can be at most $O(n)$ such cells on $\overline{P}$ (since the total number of edges of the unbounded cells of $A(H)$ is $O(n)$ [33]). To resolve the instability issue associated with vertical lines, we identify each such cell C_V on $\overline{P}$, as it is generated by topological walk. Starting with C_V, we next identify the vertices of the corresponding hourglass on P (this can be easily done in $O(n^2)$ time over all

such cells C_V). (Note that C_V does not maintain sufficient information for restoring its corresponding hourglass since such an hourglass is associated with two unbounded cells.) Finally, we use a different dualization, obtained from the one above by exchanging the labels of the coordinate axes of P (i.e., rotating the coordinate system by 90 degrees), to compute $S(L)$ and the feasible domain C.

This approach retains the simplicity of the form of $S(L)$ as given above (i.e., the terms in the summation are all linear fractionals), and gains stability in computing $S(L)$ as well. Also, there is no need to use the absolute-value function in the expression of $S(L)$. This is possible since the sign of $S(L)$ and the sign of each term in $S(L)$ do not change over C. Thus, the sign of each term in $S(L)$ for the first encountered cell C by topological walk can be computed in advance and "hidden" in the values of the constant coefficients of the numerator.

Before solving the instance of the optimization problem on each cell $C \in S_C$, we need to compute the feasible domain and the constant coefficients of the terms in $S(L)$. As stated earlier, these coefficients depend only on the segments in L_S and the weights of the regions in R_L. When changing from one problem instance to another (i.e., going from one cell of S_C to the next), a new $S(L)$ and a new feasible domain need to be computed. The new domain can be produced by (say) topological walk. Further, there is no need to recompute the entire set of constant coefficients of the new $S(L)$: it suffices to update the previous set, by removing the coefficients for the terms that no longer appear in $S(L)$ and inserting the ones for the new terms of $S(L)$. As stated earlier, this updating can be done in amortized constant time per cell of S_C.

3 The 3-D Case

In this section, we sketch our algorithms for problems P1 and P2 in 3-D. As for the 2-D cases, we divide each problem into two subproblems: Find the cell set S_C and, for each cell $C \in S_C$, find the ray that optimizes $S(L)$ over C, maintaining the ray that gives the best value for $S(L)$ over all cells $C \in S_C$.

Consider a line L in 3-D. L can be described by four parameters: the two spherical coordinates (θ, ϕ) of the direction vector of L and the projection (u, v) onto the plane orthogonal to L and containing the origin [16]. Hence, a cell $C \in S_C$ is a point set in the 4-D dual space.

To build S_C, we use a visibility approach similar to that for the 2-D cases. Let S_R be the set of 1-faces (i.e., edges) of the polyhedral subdivision R and S_T be the set of 1-faces of the target region T in 3-D. To generate S_C for P1, we consider the line segments in $S_R \setminus S_T$ (i.e., discard the interior of the 3-faces and 2-faces of $\{R_1, R_2, \ldots, R_m\}$) and the 2-faces of the target region T as the objects of the scene, and compute the faces F of the 3-D visibility complex that have as the backward view the object at infinity (a large sphere enclosing R) and as the forward view some face of T. To generate S_C for P2, we consider all line segments in S_R as the objects of the scene, and compute the faces F of the visibility complex that have the object at infinity as both the backward and forward views. We further consider each face $f \in F$ such that every point $p \in f$ dualizes to a line that penetrates the target region T.

For both P1 and P2, S_C can be computed in an output-sensitive $O((n^3 + k)\log n)$ time, where k is the size of the visibility complex and is between $\Omega(n)$ and $O(n^4)$, by using the algorithm in [16]. We are not aware of any topological walk or topological sweep algorithm for the 4-D case, although some results were given for the 3-D case.

The parameterization of L as given above (i.e., the quadruple (u, v, θ, ϕ)) may result in a rather complicated expression for the terms in $S(L)$. To obtain a simple expression for each term in $S(L)$, we choose to use a slightly different parameterization of L. Let L_z and L_y be the projections of L onto the planes $z = 0$ and $y = 0$, respectively. We describe L by the quadruple (u, v, θ, ϕ), where u and v are the coordinates of the intersection point of L with the plane $x = 0$, and θ and ϕ are the slopes of L_z and L_y. Thus, we obtain the following form for $S(L)$:

$$S(L) = \sqrt{1 + \theta^2 + \phi^2} \sum_{i=1}^{k-1} w_i |x_{i+1} - x_i|,$$

where the form for x_i is

$$x_i = \frac{a_i - b_i u - v}{\phi - c_i \theta - e_i}$$

and a_i, b_i, c_i and e_i are constants that depend on the planes supporting the facets (2-faces) of the 3-D regions that are intersected by L.

As for the 2-D case, instability and sign changes of the terms in $S(L)$ may occur when the feasible domain contains lines that are in planes parallel to the plane $x = 0$. To fix these problems, we proceed like in the 2-D case: identify the corresponding 3-D hourglass and use a different dualization. Then, we have the following lemma.

Lemma 3 *The set S_C of cells for problems P1 and P2 in 3-D and the sequences of 2-faces of R intersected by a line L in each cell $C \in S_C$ can be computed in $O(n^4 \log n)$ time in the worst case, for all cells $C \in S_C$.*

4 Experimental Results

In this section, we give some preliminary experimental results for solving problems P1 and P2 in 2-D. We implemented our 2-D algorithms in LEDA and ran the experiments on a SUN SPARC 5 computer. Since real data were not available to us at the time we did the experiments, we generated planar maps as the subdivision R, using the LEDA GraphWin class. In the full paper, we hope to be able to include experiments based on real data sets from the field of radiation therapy.

Our implementations use some robustness techniques, together with topological walk, to compute the cells of S_C and to maintain L_S over S_C. This helps our algorithms in achieving robustness and efficiency. Topological walk, as a key component of our algorithms, has been carefully implemented as a stand-alone procedure, and thus may be incorporated into other applications (e.g., see [3, 14]). One issue we encountered during the implementation was how to deal with numerical errors, which may cause some LEDA functions (e.g., test of intersection of line segments, test of a point on a line, etc.) to return erroneous results. We overcome this by enhancing our data structures with additional information for checking the correspondence between the computed values in the dual space (e.g., when topological walk computes the intersection point p of two lines) and their dual values in the original scene (e.g., the vertices of the scene that define the line dual to p).

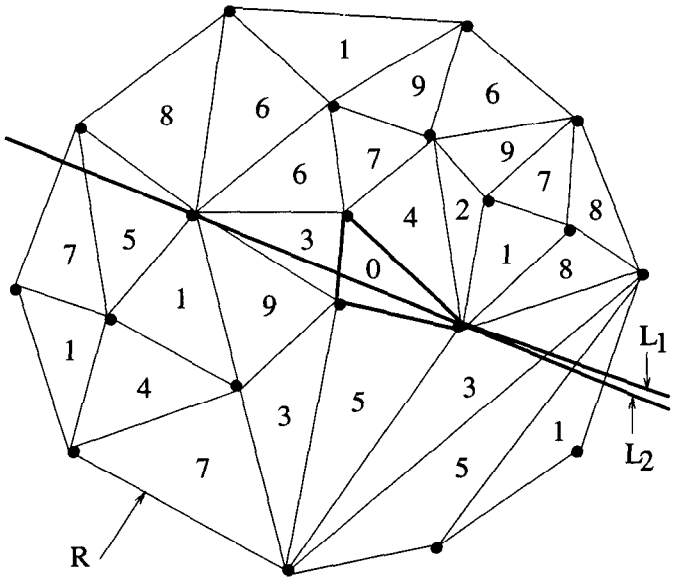

Figure 3: A subdivision R and the optimal rays for P1 (L_1) and P2 (L_2).

Another key component of our implementations is the software for optimizing $S(L)$. To compute a ray L that minimizes $S(L)$ over a cell $C \in S_C$, we used a non-linear optimization software called CFSQP [15], developed at the University of Maryland. CFSQP is a set of C functions, and we were able to build an interface between this software and our LEDA programs.

We measure the following parameters of our algorithms: (1) the total number of vertices of the subdivision R; (2) the total number of iterations performed by CFSQP, for finding an optimal value of $S(L)$ over all the cells of S_C (an iteration corresponds to the amount of computation performed by the software in order to advance from the current feasible point to the next one, thus giving a better value for $S(L)$ inside the same cell); (3) the user time, as measured by the *getrusage* function.

In the example of Figure 3, we use a subdivision R with twenty vertices, and show the ray L_1 that optimizes P1 and the ray L_2 that optimizes P2, as given by the output of our program. The boundary of the target region is in bold line and the target has associated a zero weight (all other weights are positive).

Figure 4 presents a plot of the average, over multiple runs, of the user time (in seconds) ver-

sus the number of vertices of the subdivision R, for both P1 (continuous line) and P2 (dotted line). The plot in Figure 4 indicates that the running time of our implementation increases quadratically as the number of vertices increases, which is consistent with the theoretical analysis.

Figure 5 depicts the average, over multiple runs, of the total number of iterations performed by the optimization software for finding the optimal solutions for P1 (continuous line) and P2 (dotted line) versus the number of vertices of R. The plot in Figure 5 seems to suggest that the contribution of the optimization software to the total running time is likely to decrease as the number of vertices of the subdivision R increases.

One conclusion we draw from our experiments is that the number of iterations for finding an optimal ray over a cell has high variations from one cell to another, even when the feasible domain is topologically the same (e.g., a triangle). Another thing to notice is that the optimization software allows the user to specify the error tolerance in computing the optimal solution. Hence, for example, when the target region is surrounded by critical healthy organs (and thus high precision is required), we can use a smaller error tolerance for computing the optimum, paying the price of having a longer running time. But if there are only sparsely distributed critical organs around the target, we can use a bigger error tolerance to reduce the running time.

Acknowledgement

The authors are very grateful to Dr. Cedric Yu, Department of Radiation Oncology, School of Medicine, University of Maryland at Baltimore, for discussing the problems with us and for providing many useful references. We also thank Drs. Craig Lawrence, Jian L. Zhou, and André L. Tits for making the CFSQP software available to us.

References

[1] Aleksandrov, L., Lanthier, M., Maheshwari, A., Sack, J.-R.: An ϵ-approximation algorithm for weighted shortest paths on polyhedral surfaces. *Proc. of the 6th Scandinavian Workshop on Algorithm Theory* (1998) 11–22.

[2] Anagnostou, E.G., Guibas, L.J., Polimenis, V.G.: Topological sweeping in three dimensions. *Proc. SIGAL Int. Symp. on Algorithms* (1990) 310–317.

[3] Asano, T., Guibas, L.J., Tokuyama, T.: Walking in an arrangement topologically. *Int. J. of Computational Geometry and Applications* **4** (1994) 123–151.

[4] Asano, T., Tokuyama, T.: Topological walk revisited. *Proc. 6th Canadian Conf. on Comp. Geometry* (1994) 1–6.

[5] Bahr, G.K., Kereiakes, J.G., Horowitz, H., Finney, R., Galvin, J., Goode, K.: The method of linear programming applied to radiation treatment planning. *Radiology* **91** (1968) 686–693.

[6] Bortfeld, T., Bürkelbach, J., Boesecke, R., Schlegel, W.: Methods of image reconstruction from projections applied to conformation radiotherapy. *Phys. Med. Biol.* **38** (1993) 291–304.

[7] Bortfeld, T., Schlegel, W.: Optimization of beam orientations radiation therapy: some theoretical considerations. *Phys. Med. Biol.* **35** (1990) 1423–1434.

[8] Boyer, A.L., Bortfeld, T.R., Kahler, L., Waldron, T.J.: MLC modulation of x-ray beams in discrete steps. *Proc. of the 11th Conf. on the Use of Computers in Radiation Therapy* (1994) 178–179.

[9] Boyer, A.L., Desobry, G.E., Wells, N.H.: Potential and limitations of invariant kernel conformal therapy. *Med. Phys.* **18** (1991) 703–712.

[10] Brahme, A.: Optimization of stationary and moving beam radiation therapy techniques. *Radiother. Oncol.* **12** (1988) 129–140.

[11] Brahme, A.: Inverse radiation therapy planning: principles and possibilities. *Proc. of the 11th Conf. on the Use of Computers in Radiation Therapy* (1994) 6–7.

[12] Brahme, A.: Optimization of radiation therapy. *Int. J. Radiat. Oncol. Biol. Phys.* **28** (1994) 785–787.

[13] Censor, Y., Altschuler, M.D., Powlis, W.D.: A computational solution of the inverse problem in radiation-therapy treatment planning. *Applied Math. and Computation* **25** (1988) 57–87.

[14] Chen, D.Z., Daescu, O., Hu, X.S., Xu, J.: Finding an optimal path without growing the tree. *Procs. of the 6th Annual European Symposium on Algorithms* (1998) 356–367.

[15] Craig, L., Zhou, J.L., Tits, A.L.: User's guide for CFSQP Version 2.5. Electrical Eng. Dept. and Institute for Systems Research, University of Maryland, TR-94-16r1, 1994.

[16] Durand, F., Drettakis, G., Puech, C.: The 3D visibility complex, a new approach to the problems of accurate visibility. *Proc. of the 7th Eurographic Workshop on Rendering* (1996) 245–257.

[17] Edelsbrunner, H., Guibas, L.J.: Topologically sweeping an arrangement. *Journal of Computer and system Sciences* **38** (1989) 165–194.

[18] Gustafsson, A., Lind, B.K., Brahme, A.: A generalized pencil beam algorithm for optimization of radiation therapy. *Med. Phys.* **21** (1994) 343–356.

[19] Holmes, T., Mackie, T.R.: A comparison of three inverse treatment planning algorithms. *Phys. Med. Biol.* **39** (1994) 91–106.

[20] Lanthier, M., Maheshwari, A., Sack, J.-R.: Approximating weighted shortest paths on polyhedral surfaces. *Proc. of the 13th ACM Symp. on Comp. Geometry*

(1997) 274–283.
[21] Latombe, J.C.: *Robot Motion Planning*, Kluwer Academic Publishers, Boston, MA, 1991.
[22] Legras, J., Legras, B., Lambert, J.P., Aletti, P.: The use of a microcomputer for non-linear optimization of doses in external radiotherapy. *Phys. Med. Biol.* **31** (1986) 1353–1359.
[23] Lind, B.K.: Properties of an algorithm for solving the inverse problem in radiation therapy. *Proc. of the 9th Intl. Conf. on the Use of Computers in radiation Therapy* (1987) 235–239.
[24] Lind, B.K., Brahme, A.: Optimization of radiation therapy dose distributions with scanned photon beams. *Inv. Prob.* **16** (1990) 415–426.
[25] Mata, C., Mitchell, J.S.B.: A new algorithm for computing shortest paths in weighted planar subdivisions. *Proc. of the 13th ACM Symp. on Comp. Geometry* (1997) 264–273.
[26] McDonald, S.C., Rubin, P.: Optimization of external beam radiation therapy. *Int. J. Radiat. Oncol. Biol. Phys.* **2** (1977) 307–317.
[27] Mitchell, J.S.B., Papadimitriou, C.H.: The weighted region problem: Finding shortest paths through a weighted planar subdivision. *Journal of the ACM* **38** (1991) 18–73.
[28] Pocchiola, M, Vegter, G.: The visibility complex. *Int. J. of Computational Geometry and Applications* **6** (1996) 279–308, a special issue devoted to ACM-SCG'93.
[29] Powlis, W.D., Altschuler, M.D., Censor, Y., Buhle, E.L.: Semi-automated radiotherapy treatment planning with a mathematical model to satisfy treatment goals. *Int. J. Radiat. Oncol. Biol. Phys.* **16** (1989) 271–276.
[30] Preparata, F. P. and Shamos, M. I.: *Computational Geometry: An Introduction*, Springer-Verlag, New York, 1985.
[31] Rivière, S.: Visibility computations in 2D polygonal scenes. Ph.D. thesis, Université Joseph Fourier, Grenoble, France (1997).
[32] Schweikard, A., Adler, J.R. and Latombe, J.C.: Motion planning in stereotaxic radiosurgery. *IEEE Trans. on Robotics and Automation* **9** (1993) 764–774.
[33] M. Sharir and P.K. Agarwal, *Davenport-Schinzel Sequences and Their Geometric Applications*, Cambridge University Press,1995.
[34] Webb, S.: Optimization of conformal radiotherapy dose distributions by simulated annealing. *Phys. Med. Biol.* **34** (1989) 1349–1369.
[35] Webb, S.: Optimizing the planning of intensity-modulated radiotherapy. *Phys. Med. Biol.* **39** (1994) 2229–2246.

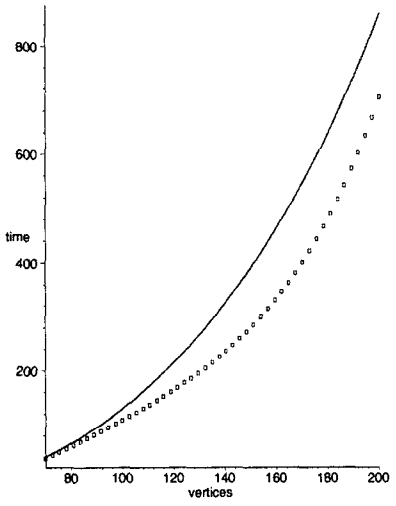

Figure 4: The user time versus the number of vertices in R.

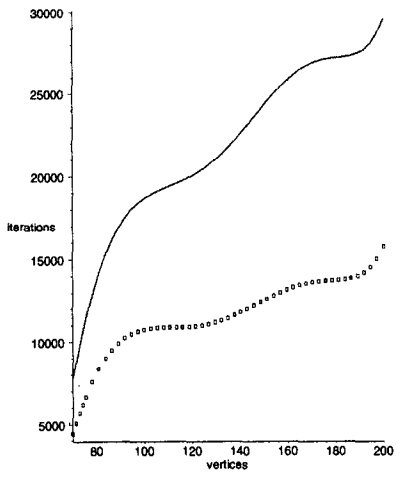

Figure 5: The number of iterations versus the number of vertices in R.