An optimization approach for radiosurgery treatment planning *

Michael C. Ferris[†] Jinho Lim [‡] David M. Shepard [§]

November 6, 2001

Abstract

We outline a new approach for radiosurgery treatment planning, based on solving a series of optimization problems. We consider a specific treatment planning problem for a specialized device known as the *Gamma Knife*, that provides an advanced stereotactic approach to the treatment of tumors, vascular malformations, and pain disorders within the head. The sequence of optimization problems involves nonlinear and mixed integer programs whose solution is required in a given planning time (typically less than 30 minutes). This paper outlines several modeling decisions that result in more efficient and robust solution. Furthermore, it outlines a new approach for determining starting points for the nonlinear programs, based on a skeletonization of the target volume. Treatment plans are generated for real patient data that show the efficiency of the approach.

1 Introduction

Radiation therapy is the treatment of cancer with ionizing radiation. This radiation, in the form of X-rays and gamma rays, damages the DNA of the cells in the area being treated, interfering with their ability to divide and grow. Cancerous cells are unable to repair this damage, so their growth is curtailed and the tumor shrinks. Healthy cells may also be damaged by the radiation, but they are more able to repair the damage and return to normal function. Radiation therapy may be used to treat solid tumors, such as cancers of the skin, brain,

*This material is based on research supported in part by the National Science Foundation Grant CCR-9972372, the Air Force Office of Scientific Research Grant F49620-01-1-0040, Microsoft Corporation and the Guggenheim Foundation.

[†]Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford, OX1 3QD. Permanent address: Computer Sciences Department, University of Wisconsin, 1210 West Dayton Street, Madison, Wisconsin 53706. email: ferris@cs.wisc.edu

[‡]Department of Industrial Engineering, University of Wisconsin, Madison, Wisconsin 53706. email: jin-ho@cs.wisc.edu

[§]Department of Radiation Oncology, University of Maryland School of Medicine, 22 South Green Street, Baltimore, MD 21201 email: dshep001@umaryland.edu

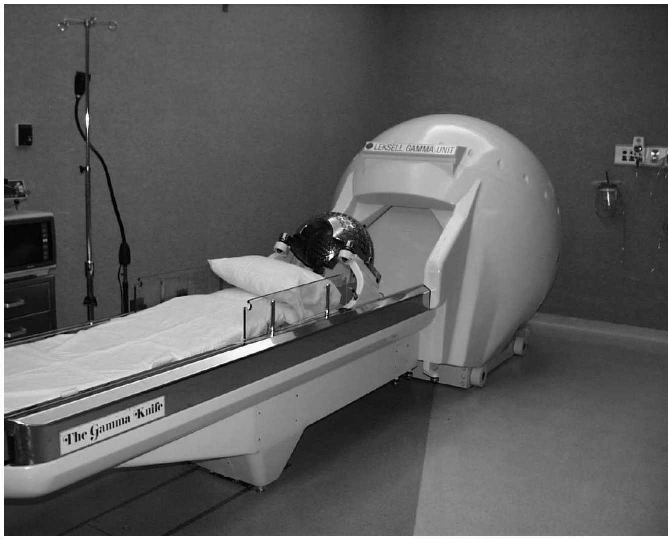

Figure 1: Gamma Knife treatment unit

and breast. It can attack cancer cells both on the surface of the body or deep within. It can be used as the sole form of treatment, or in conjunction with surgery (to shrink the tumor before surgery, or to kill remaining cancer cells after surgery) or chemotherapy.

Devices for delivering the radiation allow a significant amount of control over the characteristics of the radiation. Treatment plans, which specify the shapes of the applied radiation beams, times of exposure, etc., should be designed in a way that delivers a specified dose to the tumor while avoiding an excessive dose to the surrounding healthy tissue and, in particular, to any important nearby organs. The full potential of these devices to deliver optimal treatment plans has yet to be realized, due to the complexity of the treatment design process. This paper describes how to use advanced modeling techniques and state-of-the-art optimization algorithms for the design of treatment plans that fully exploit the capabilities of this new generation of technology.

Specifically, we consider treatment planning for a specialized device known as the *Gamma Knife*, which provides an advanced stereotactic approach to the treatment of tumors, vascular malformations, and pain disorders within the head [7]; see Figure 1. Inside a shielded treatment unit, beams from 201 cobalt-60 radioactive sources are focused so that they intersect at a certain point in space, producing a ellipsoidal region of high radiation dose referred to as a *shot*. A typical treatment consists of a number of shots, of possibly different sizes and different durations, centered at different locations in the tumor, whose cumulative effect is to deliver a certain dose to the treatment area while minimizing the effect on surrounding tissue.

The motivation for this problem, and the approaches that form the basis of this work have appeared elsewhere [5, 6, 14]. The key contributions of this paper are as follows:

1. The description and implementation of a heuristic approach to generate a good starting point for the nonlinear programs used to model the treatment planning approach (see Section 3). The approach is based on skeletonization ideas from computational graphics, is augmented using various optimization subproblems and leads to improved speed and quality of solutions (see Section 4).

2. Some practically motivated changes to the underlying models to improve robustness of the solution process and quality of the resulting treatment plan. In particular, several nonlinear programs have been replaced by a single (easy to solve) mixed integer program, some "hard constraints" have been remodeled using inexact penalization, and least squares optimization has been used for parameter estimation (see Section 2).

3. Tuning of the model parameters to improve solution speed and robustness (see Section 4).

The resulting tool provides solutions to the problem that are currently in use at the University of Maryland Medical School. The work described here has enabled the simple prototype to be enhanced to be usable (without optimization expert intervention) as a mechanism to robustly improve the operation of a complex medical system.

2 Models and solution process

The first step in generating a treatment plan is to model the dose delivered to the patient by a given shot centered at a given location. A nonlinear least squares model for this was developed in [5]. The dose delivered from any set of shots can be calculated as

$$Dose(i,j,k) = \sum_{(s,w)\in\mathcal{S}\times\mathcal{W}} t_{s,w} D_w(x_s,y_s,z_s,i,j,k), \tag{1}$$

where $D_w(x_s,y_s,z_s,i,j,k)$ is the dose delivered to the voxel (i,j,k) by the shot of size w centered at (x_s,y_s,z_s). Note that D_w is a complicated (non-convex) function of its arguments involving square roots and the error function.

Once the functional form of the dose calculation is known, a series of 5 optimization problems are solved to determine the treatment plan.

1. Conformity estimation. The treatment plan needs to be both conformal and homogeneous. It is easy to specify homogeneity in the models simply by imposing lower and upper bounds on the dose delivered to voxels in the target $\mathcal{T}$. Typically, however, the imposition of rigid bounds leads to plans that are overly homogeneous and not conformal enough, that is,

they provide too much dose outside the target. To overcome this problem, the notion of "underdose" was suggested in [5]. Underdose measures how much the delivered dose is below the prescribed dose on the target voxels and is described in more detail in Section 2.2. In all of our models, we either constrain the underdose to be less than a prespecified value, or attempt to minimize the total underdose.

The conformity of the plan is harder to deal with since it involves voxels outside of the target, of which there may be many. Furthermore, a reasonable conformity for a given patient plan is very hard to estimate a priori since it depends critically on the number of shots allowed and how the volume of the target interacts with the volumes of the allowable shots. In order to avoid calculating dose outside of the target, we solve an optimization problem on the target to estimate an "ideal" conformity for the particular patient for a given number of shots; details can be found in Section 2.1.

2. Coarse grid estimate. Given the estimate of conformity, we then specify a series of optimization problems whose purpose is to minimize the total underdose on the target for the given conformity. In order to reduce the computational time required to determine the plan, we first minimize the total underdose on a coarse grid subset of the target voxels.

3. Refined grid estimate. In order not to increase the number of voxels considered too much, we only add to the coarse grid those voxels on a finer grid for which the homogeneity (bound) constraints are violated. This procedure improves the quality of the plan without greatly increasing the execution time.

4. Shot reduction problem. To this point, our discussion has omitted the fact that we can only use a certain number of size/location combinations in the treatment plan. Choosing the particular shot size at each location is a discrete optimization problem that is treated by approximating the step function

$$H(t) = \begin{cases} 1 & \text{if } t > 0 \\ 0 & \text{if } t = 0 \end{cases}$$

by a nonlinear function,

$$H(t) \approx H_\alpha(t) := \tfrac{2\arctan(\alpha t)}{\pi} \quad .$$

For increasing values of α, H_α becomes a closer approximation to the step function H for $t \geq 0$. This process is typically called smoothing.

The set of shot sizes for a given number of shots n is chosen by imposing the constraint:

$$n = \sum_{(s,w)\in\mathcal{S}\times\mathcal{W}} H_\alpha(t_{s,w}). \tag{2}$$

This states that the total number of size/location combinations to be used is n. We have found it beneficial to use one or two more shot locations in the model than the number requested by the user, that is $\mathcal{S} := \{1, ..., n+2\}$, and allowing the optimization to choose not only useful sizes but also to discard the extraneous shot locations.

Note that it is possible for the optimization solution to suggest multiple shots to be centered at the same location. If, in addition, there are other locations that are not used at all in the solution at hand, we shift as many of the multiple shots as possible to these unused locations. This maintains the objective value of the current solution while giving any subsequent solves the ability to move the different size shots independently.

In the models outlined in 1-3, we use a small value of α, typically 6 to impose the constraint (2) in an approximate manner. In the fourth solve, we increase the value of α to 100 in an attempt to force the planning system to choose which size/location pairs to use. At the end of this solve, there may still exist some size/location pairs that have very small exposure times t. Also note that our solution technique does not guarantee that the shots are centered at locations within the target.

5. Fixed location model. The computed solution may have more shots used than the user requested and furthermore may not be implementable on the Gamma Knife since the coordinate locations cannot be keyed into the machine. Our approach to refine the optimization solution to generate implementable coordinates for the shot locations is to round the shot location values and then fix them. Once these locations are fixed, the problem becomes linear in the intensity values t. We reoptimize these values and force the user requested number of size/location pairs precisely using a mixed integer program. Further details can be found in Section 2.3.

Note that the starting point for each of the models is the solution point of the previous model. Details on how to generate an effective starting point for the first model are given in Section 3. All the optimization models are written in the GAMS [3] modeling language and solved using CONOPT [4] or CPLEX [10]. In the following subsections we give some more details of each of these optimization problems.

2.1 Conformity estimation

Our conformity index C is an estimate of the ratio of the dose delivered to the target, divided by the total dose delivered to the patient. The latter quantity is estimated by summing the (measured) dose delivered ($\bar{D}_w$) by a shot of size w for length $t_{s,w}$ to a "phantom". Thus C is calculated by the following expression.

$$C = \frac{\displaystyle\sum_{(i,j,k)\in\mathcal{T}} Dose(i,j,k)}{\sum_{(s,w)\in\mathcal{S}\times\mathcal{W}} \bar{D}_w t_{s,w}}$$

In previous work [5], we attempted to estimate C by minimizing the total dose to the target, subject to hard constraints on the amount of dose delivered at each voxel in the target. However, instead of enforcing these hard constraints, we now propose the following optimization model as a mechanism to determine C.

$$
\begin{array}{ll}
\min & \displaystyle\sum_{(s,w)\in\mathcal{S}\times\mathcal{W}} \bar{D}_w t_{s,w} \\
\text{subject to} & Dose(i,j,k) = \displaystyle\sum_{(s,w)\in\mathcal{S}\times\mathcal{W}} t_{s,w} D_w(x_s,y_s,z_s,i,j,k) \\
& \theta \leq UnderDose(i,j,k) + Dose(i,j,k) \\
& 0 \leq UnderDose(i,j,k) \\
& 0 \leq Dose(i,j,k) \leq 1, \quad \forall (i,j,k)\in\mathcal{T} \\
& \sum_{(i,j,k)\in\mathcal{T}} UnderDose(i,j,k) \leq \mathcal{N}\mathcal{P}_{\mathcal{U}} \\
& n = \displaystyle\sum_{(s,w)\in\{1,\ldots,n\}\times\mathcal{W}} H_\alpha(t_{s,w}) \\
& 0 \leq t_{s,w} \leq \bar{t}
\end{array}
\tag{3}
$$

The crucial constraint is the one involving both $\mathcal{N}$, the number of voxels in the target, and $\mathcal{P}_{\mathcal{U}}$ a user supplied estimate of the "average percentage" underdose allowable on the target. By increasing the value of $\mathcal{P}_{\mathcal{U}}$, the user is able to relax the homogeneity requirement, thereby reducing the total dose delivered to the patient. (The model from [5] forced the underdose to be zero at every voxel in the target.) Notice that reducing the total dose delivered to the patient typically increases C. Thus, C is essentially a monotone function of $\mathcal{P}_{\mathcal{U}}$. The upper bound on exposure time $\bar{t}$ is typically chosen as a large fraction of the maximum dose delivered to $\mathcal{T}$ (here assumed to be 1) for the purposes of improving solver performance.

The following table indicates the motivation for this change. For a variety of patients, the estimate of C is essentially the same, but is has smaller standard deviation (indicated in parentheses) and smaller computing times. (For each of the patients, the starting point for the conformity problem was randomly perturbed by up to two voxels in each coordinate direction to generate the sample. The variance is calculated over a set of 30 runs.) Furthermore, it seems clear that the final objective values arising from the subsequent solves is better if these solves are seeded with the new conformity estimation model solutions.

2.2 Underdose models

The main models used in the treatment planning process are nonlinear programs, defined over a (grid) subset $\mathcal{G}$ of the voxels in the target $\mathcal{T}$.

Table 1: Comparison of conformity estimation models

Patient	Old Conformity Model			New Conformity Model		
	C	obj.val.	time	C	obj.val.	time
Patient 5	0.296	28.89	106.1	0.296	25.68	77.4
	(0.007)	(13.93)	(32.9)	(0.005)	(12.93)	(17.3)
Patient 6	0.246	17.81	397.0	0.247	14.89	358.3
	(0.011)	(14.54)	(90.5)	(0.009)	(13.21)	(56.2)
Patient 8	0.323	3.33	195.2	0.323	2.86	167.6
	(0.007)	(2.73)	(60.8)	(0.003)	(1.79)	(56.3)

$$
\begin{aligned}
\min \quad & \sum_{(i,j,k)\in\mathcal{G}} UnderDose(i,j,k) \\
\text{subject to} \quad & Dose(i,j,k) = \sum_{(s,w)\in\mathcal{S}\times\mathcal{W}} t_{s,w} D_w(x_s,y_s,z_s,i,j,k) \\
& \theta \le UnderDose(i,j,k) + Dose(i,j,k) \\
& 0 \le UnderDose(i,j,k) \\
& 0 \le Dose(i,j,k) \le 1, \quad \forall (i,j,k)\in\mathcal{G} \\
& C\frac{\mathcal{N}_{\mathcal{G}}}{\mathcal{N}} \le \frac{\sum_{(i,j,k)\in\mathcal{G}} Dose(i,j,k)}{\sum_{(s,w)\in\mathcal{S}\times\mathcal{W}} \bar{D}_w t_{s,w}} \\
& n = \sum_{(s,w)\in\{1,\ldots,n\}\times\mathcal{W}} H_\alpha(t_{s,w}) \\
& 0 \le t_{s,w} \le \bar{t}
\end{aligned}
\tag{4}
$$

Note that minimizing underdose is the objective and conformity is imposed as a constraint in contrast to (3). In practice, for solution performance, the constraint involving C is rearranged to be a linear constraint by rationalizing the denominator. (Note also that we normalize C using the number of voxels $\mathcal{N}_{\mathcal{G}}$ in the grid.) The mechanism to update both $\mathcal{G}$ and α is described above. These models are essentially the same as described in [5], except that an upper bound has been applied to the exposure times. While this upper bound was motivated by application specific considerations it also helps increase solution robustness.

2.3 Fixed location model

In order to implement the solution on the Gamma Knife, we round the location values from the fourth solve and fix them at $\bar{x}_s$, $\bar{y}_s$ and $\bar{z}_s$ respectively. The values of $D_w(\bar{x}_s, \bar{y}_s, \bar{z}_s, i, j, k)$ can then be calculated at each location (i, j, k) as data. The final optimization involves the following mixed integer linear optimization problem.

$$
\begin{array}{lll}
\min & \displaystyle\sum_{(i,j,k)\in\mathcal{G}} UnderDose(i,j,k) & \\
\text{subject to} & Dose(i,j,k) = \displaystyle\sum_{(s,w)\in\mathcal{S}\times\mathcal{W}} t_{s,w} D_w(\bar{x}_s, \bar{y}_s, \bar{z}_s, i, j, k) & \\
& \theta \leq UnderDose(i,j,k) + Dose(i,j,k) & \\
& 0 \leq UnderDose(i,j,k) & \\
& 0 \leq Dose(i,j,k) \leq 1, \quad \forall (i,j,k) \in \mathcal{G} & \qquad (5) \\
& C\dfrac{\mathcal{N}_{\mathcal{G}}}{\mathcal{N}} \displaystyle\sum_{(s,w)\in\mathcal{S}\times\mathcal{W}} \bar{D}_w t_{s,w} \leq \displaystyle\sum_{(i,j,k)\in\mathcal{G}} Dose(i,j,k) & \\
& 0 \leq t_{s,w} \leq \psi_{s,w} \bar{t} & \\
& \sum_{(s,w)\in\mathcal{S}\times\mathcal{W}} \psi_{s,w} \leq n & \\
& \psi_{s,w} \in \{0,1\} &
\end{array}
$$

This model was adapted from the work described in [6]. The key observation is the use of the binary variable $\psi_{s,w}$ to indicate whether a shot of size w is used at location s. The penultimate constraint in the model ensures that no more than n shots are used, while the upper bound on t ensures that no exposure time occurs if the corresponding shot is not used. In previous work [5], we had used increasing values of α coupled with the removal of small shots in a nonlinear programming approach. The current scheme is guaranteed to outperform this.

It may of course be possible to extend this model to include more locations, but this was not deemed necessary for our work.

3 Starting point generation

A good starting point is very important for nonlinear programs, especially if the problem is not convex. This section will explore some techniques to find an initial starting solution for our solution process. The main focus is to find a set of good shot locations and their corresponding sizes. We propose a shot location and size determination (SLSD) process based on 3D medial axis transformation. Our results show that it takes no more than 6 seconds to produce a good starting solution for all the three-dimensional data considered in our research.

Our targets are collections of three-dimensional voxels. For the large scale problems of interest, the data manipulation and optimization solution times are much larger than allowable (typically 20-40 minutes is allowed for planning) and we must resort to data compression. One technique used extensively in computer vision and pattern recognition is the notion of a skeleton, a series of connected lines providing a simple representation of the object at hand[1, 8, 11, 15, 18]. Skeletons have been used by physicians and scientists to explore virtual human body organs with non-invasive techniques[9, 17]. The term skeleton was proposed in [1] to describe the axis of symmetry, based on the physical analogy of grassfire propagation, namely, the locus of centers of maximal disks (balls) contained in a two- (three-) dimensional shape.

Some applications require that the original object has to be reconstructed from the compact representation, and hence the normal measure of goodness is the error between the original and reconstructed object. However, in our case, we will just use the skeleton to quickly find good locations for starting points of the nonlinear program, so our objective is good location and speed of generation. Thus we adapt techniques from the literature to achieve these goals.

Our process is in three stages. First we generate the skeleton, then we place shots and choose their sizes along the skeleton to maximize a measure of our objective. After this, we choose the initial exposure times using a simple linear program. Finally, we apply the five stage optimization process outlined in Section 2 to improve upon the starting points found.

3.1 Skeleton generation

In this section, we introduce a 3D skeleton algorithm that follows similar procedures to that of [17]. The first step in the skeleton generation is to compute the contour map containing distance information from the voxel to a nearest target boundary. The ideal distance metric is Euclidean, but this is too time consuming to implement in a three-dimensional environment. To describe our simpler scheme, we first introduce some terminology.

Definition 1 *Considering a voxel i as a three-dimensional box, an adjacent voxel j is called an F-neighbor of i if j shares a face with i, an E-neighbor of i if j shares an edge with i and a V-neighbor of i if j shares a vertex with i.*

Our procedure is as follows:

1. Assign 0 to the non-target area, and let $v = 0$.
2. Assign $v + 1$ to any voxel that is unassigned and has an F-neighbor with value v.
3. Increment v by 1 and repeat until all voxels in the target area are assigned.

An example of a two-dimensional contour map generated is shown in Figure 2.

Note that if the maximum height in the contour map is less than 2, we terminate the skeleton generation process.

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	2	2	1	1	1	1	1	0	0	0	0	0	0	0
0	0	0	1	2	3	2	2	2	2	2	1	1	1	0	0	0	0
0	0	0	0	1	2	3	3	3	3	3	2	2	2	1	0	0	0
0	0	0	0	1	2	3	4	4	4	4	3	2	1	0	0	0	0
0	0	0	0	0	1	2	3	4	**5**	4	3	2	1	0	0	0	0
0	0	0	0	0	0	1	2	3	4	4	3	2	1	0	0	0	0
0	0	0	0	0	0	1	2	3	4	3	2	1	0	0	0	0	0
0	0	0	0	0	1	2	3	4	4	3	2	1	0	0	0	0	0
0	0	0	0	1	2	3	4	3	3	2	1	0	0	0	0	0	0
0	0	0	0	1	2	3	3	2	2	2	1	0	0	0	0	0	0
0	0	0	1	2	2	2	2	1	1	1	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2: A contour map on a two-dimensional example

Extracting an initial skeleton. Based on the contour map, there are several known skeleton extraction methods in the literature [17]: *Boundary Peeling* (also called thinning)[12], *Distance Coding* (distance transformation) [13] and *Polygon-based Voronoi Methods*[2]. Because it is simple and fast, we use the distance transformation method to generate a skeleton. In our terminology, this means that we define a *skeleton point* as a voxel whose contour map value is greater than or equal to those of its E-neighbors.

Refinement for connectivity of a thin skeleton. We say that two skeleton points are *connected* if they are V-neighbors. Unfortunately, not all the skeleton points generated will be connected, and thus we use a two stage process to connect the pieces of the skeleton together.

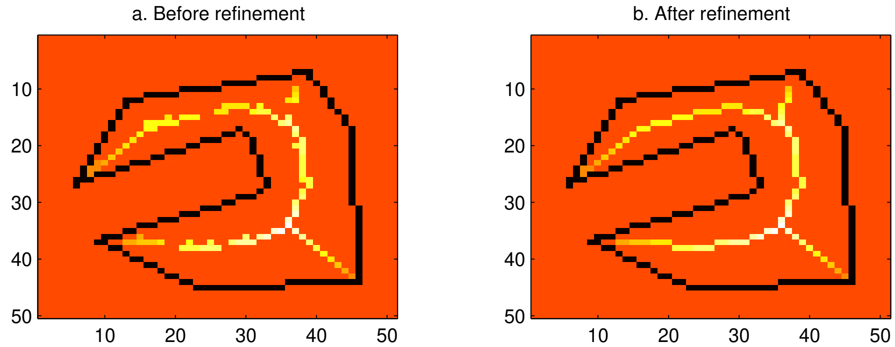

Figure 3: An example of skeleton refinement

For example, Figure 3(a) shows a raw skeleton with several disconnected components. We use two algorithms to join all the disconnected components. The first algorithm is a *directional search* algorithm. The second is the *shortest* *path* algorithm. After these refinements, we have a connected skeleton as seen in Figure 3(b).

We first use depth first search to label each skeleton point as belonging to a particular component of the skeleton. The first connection phase is a steepest ascent technique. Consider the contour map as a function f. We calculate an approximate gradient ∇f using coordinate-wise central divided differences. Thus, for each voxel (i, j, k), we use the values of f at each of its F-neighbors to generate a three-dimensional vector

$$\begin{aligned}\nabla f(i,j,k) := (&\mathrm{sgn}(f(i+1,j,k) - f(i-1,j,k)),\\ &\mathrm{sgn}(f(i,j+1,k) - f(i,j-1,k)),\\ &\mathrm{sgn}(f(i,j,k+1) - f(i,j,k-1)))\end{aligned}$$

and store these in a divided difference table. Given the voxel (i, j, k) we evaluate f at the V-neighbor $(i, j, k) + \nabla f(i, j, k)$, and accept the move if f does not decrease. We terminate the process if either f decreases or we move to a voxel in a different piece of the skeleton, thus connecting (i, j, k) to this piece. Including the paths generated in this fashion in the skeleton typically connects pieces that are close but not currently connected.

The directional search algorithm, while joining many of the disconnected pieces of the skeleton along ridges of the contour map may fail in cases where the contour map decreases in the gap between two disconnected pieces. Therefore, the second connection phase uses a shortest path algorithm to connect the skeleton (instead of using the *saddle point method* discussed in [17]).

Let $\mathcal{K}$ be the set of all skeletal points, divided into d disconnected components. In order to reduce the search space for the shortest path algorithm, we generate a cloud of voxels $\mathcal{C}$ in the target volume each of which are local maxima among their F-neighbors. Note that $\mathcal{C}$ contains $\mathcal{K}$ by definition, and can be thought of heuristically as a cloud of points encircling the skeleton. We will only join the disconnected components of $\mathcal{K}$ using points in $\mathcal{C}$.

Let each voxel in $\mathcal{C}$ be a node. An arc $(i, j) \in \mathcal{A} \subseteq \mathcal{C} \times \mathcal{C}$ is defined if voxels i and j are V-neighbors.

We choose an arbitrary voxel in an arbitrary component as the source node s. A representative node is chosen from each of the remaining components arbitrarily and joined to a dummy node t that will be the destination. The distance c_{ij} between voxels in a connected cluster is assigned 0, whereas other V-neighbors of a given voxel are at distance 1. We attempt to send $d - 1$ units of flow from s to t. We also add an arc from s to t directly with a high cost to allow for the fact that it may not be possible to join every component through $\mathcal{C}$. If this is the case, it will be signified by flow along these final arcs. The complete formulation of our problem follows.

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in\mathcal{A}} c_{ij}x_{ij} \\
\text{subject to} \quad & \sum_{\{j|(i,j)\in\mathcal{A}\}} x_{ij} - \sum_{\{j|(j,i)\in\mathcal{A}\}} x_{ji} = \begin{cases} (d-1) & if \quad i=s \\ -(d-1) & if \quad i=t \\ 0 & otherwise \end{cases} \\
& 0 \le x_{ij}, \quad \forall (i,j)\in\mathcal{A}.
\end{aligned}
$$

Typically, this problem is solved very quickly by standard linear programming algorithms, even though specialized network flow algorithms could be applied.

3.2 Shot placement

At this stage, we recall that our goal is to determine where to place shots and how large to make them initially. The skeleton generation is a data reduction technique to facilitate this goal. We restrict our attention to points on the skeleton. This is reasonable, since the dose delivered (1) looks ellipsoidal in nature and hence being centrally located within the target (that is, on the skeleton) is preferable.

Our approach moves along the skeleton evaluating whether the current point is a good location to place a shot. There are two special types of skeleton points that help determining the shot size and the location, see Figure 4.

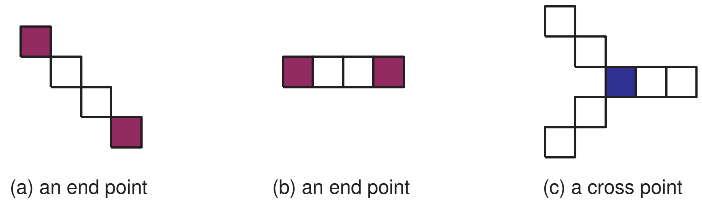

Figure 4: Examples of end-points and a cross-point

Definition 2 *A voxel is an* end-point *if*

1. *It is in the skeleton.*
2. *It has only one V-neighbor in the skeleton.*

A voxel is a cross-point *if*

1. *It is in the skeleton.*
2. *It has at least three V-neighbors.*

3. It is a local maximum in the contour map.

These points are respectively the start and finish points for our heuristic.

Let $\mathcal{K}$ be a set of skeletal points in the target volume. The first phase of the methods determines all end-points in the current skeleton. Given an end-point $(x, y, z) \in \mathcal{K}$, we carry out the following steps to generate a stack for the end-point.

1. Calculate a merit value at the current location. Save the location information, the best shot size, and the merit value on a stack.

2. Find all V-neighbors of the current point in the skeleton not in the stack. If there is exactly one neighbor, make the neighbor the current location and repeat these two steps. Otherwise, the neighbor is a cross-point, or an end-point, and we terminate.

If the length of the stack is less than 3, then we discard these points from the skeleton. Otherwise, we choose the shot location and size determined by the smallest merit value on the stack. This shot will cover a subset of the voxels in the target; these voxels are removed from the target at this stage.

We then move to the next end-point and repeat the above process. Once all end-points have been processed, we attempt to generate a new skeleton based on the remaining (uncovered) voxels in the target. We then repeat the whole process with the new skeleton.

The key to this approach is the merit function. Ideally, we would like to place shots that cover the entire region, without overdosing within (or outside) of the target. Overdosing occurs outside the target if we choose a shot size that is too large for the current location, and hence the shot protrudes from the target. Overdosing occurs within the target if we place two shots too close together for their chosen sizes.

Thus, if we label *height* as the approximate Euclidean distance to the target boundary, *spread* as the minimum distance between the current location and the end-point at which we started, and w as the shot size, we would like to ensure that all three of these measures are as close as possible. Therefore, we choose an objective function that is a weighted sum of squared differences between these three quantities.

1. $\Phi_{sh}(x, y, z) := (spread(x, y, z) - height(x, y, z))^2$

2. $\Phi_{sw}(x, y, z, w) := (spread(x, y, z) - w)^2$

3. $\Phi_{hw}(x, y, z, w) := (height(x, y, z) - w)^2$

The first function ensures that we pack the target volume as well as possible, that is the current spread between shots should be close to the distance to the closest target boundary. The second function is used to choose a helmet size that fits the skeleton best for the current location. The third function favors a location that is the appropriate distance from the target boundary for the current shot size.

Our objective function Φ is defined as a linear combination (with weights λ) of these penalty functions and a fourth $(18 - w)^2$, that is designed to favor large shot sizes. The weights can be adjusted based on a user's preference. In practice we use 1/3 for the first three objective weights, and 1/2 for the fourth.

3.3 Modifying the number of shots used

Often, the application expert knows how many shots will be needed to treat a specific tumor based upon experience. The planning tool accepts this information as input. However, the SLSD procedure only uses target information and it might suggest using fewer or more shots.

If the number of shots generated by SLSD is too large, the first $n + 2$ shots are used as the starting point. We allow the nonlinear program to adjust the locations further and remove the least useful shots during the solution process.

If the number of shot locations obtained from the SLSD procedure is less than the requested number, we add extra shot locations using the following (SemiRandom) heuristic. The key idea is to spread out the shot center locations with appropriate shot sizes over the target area.

We assume that we are given ρ, an estimate of the conformity that we require from any shot. In practice, we choose this value as 0.2. We then generate k different shot/size combinations as follows. First, a random location s is generated from the target area that is not covered by the current set of shots. Secondly, a random shot size w for the specific location is generated within the set of different shots available $\mathcal{W}$. For each shot/size combination we calculate the fraction $f(s, w)$ of the dose that hits the target by taking the ratio of the number of voxels that it hits in the target to the total number of voxels in a shot of the given size.

We decide the location and size (s, w) to use as follows. If $\max f(s, w) \leq \rho$, then we choose the combination that maximizes $f(s, w)$. Otherwise, amongst all those combinations that are acceptable (i.e. $f(s, w) \geq \rho$), we choose the largest one (i.e. the one that maximizes w among these).

Note that the SemiRandom scheme can be used in cases where the SLSD procedure fails, and also as an alternative scheme for locating starting points. In practice we use $k = 5$.

4 Computational results

In this section, we demonstrate how to use the techniques outlined above on two-dimensional testing problems as well as real patient data.

4.1 Examples on two-dimensional problems

We start with some simple two-dimensional examples that show the types of skeletons that are produced and portray the resulting optimization solutions.

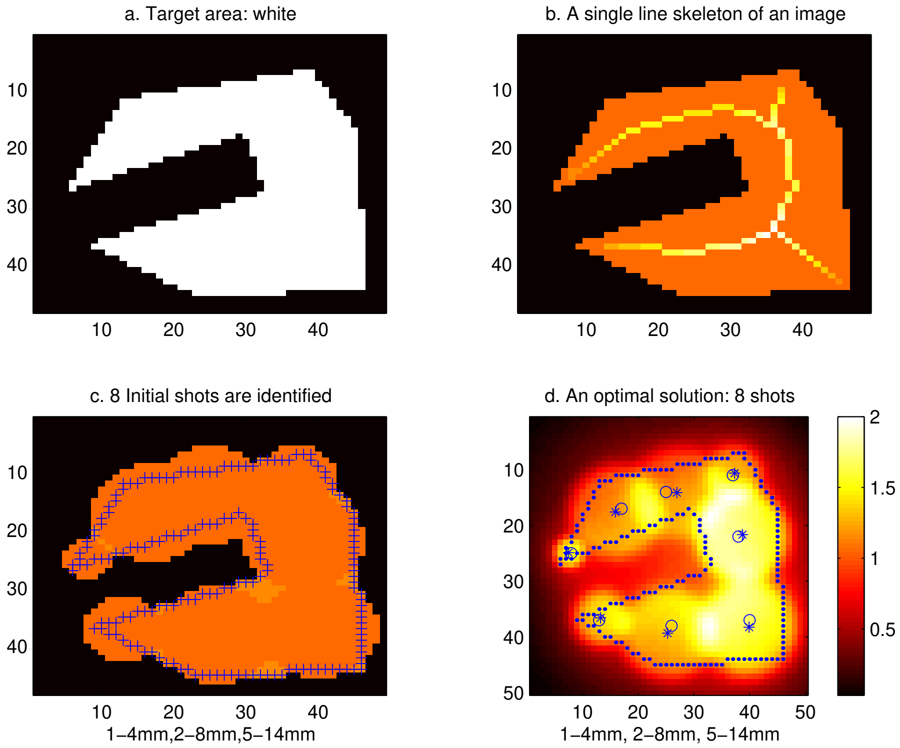

Figure 5: Computational results on a two-dimensional example

Figure 5(a) depicts a particular target (*tumor*) area for our problem as white space. This tumor is approximately 3 inches square. The shape is not convex. It has a indentation that makes it difficult for a normal optimization model to obtain an acceptable plan. Figure 5(b) shows a thin line skeleton generated from the image. The skeleton generation process takes less than 1 second on Pentium III 800MHz workstation. We then apply the SLSD process to obtain the starting solution for the NLP model as shown in Figure 5(c). Eight shots of radiation are used for this example; one 4 mm, two 8 mm and five 14 mm shots. We use 0.9 as the initial exposure times. The solution covers the target area well. We solve the conformity estimation optimization model using CONOPT2 interface with the starting solution, finding an optimal solution of 8 shots in 61 seconds of execution time. Figure 5(d) shows the resulting plot using MATLAB image toolbox. The circles are the starting solution and the stars are the optimal solution from CONOPT. They are almost identical in shot center locations. The SLSD process outperforms a random starting solution. Given 8 shots to use, the NLP model using a random starting solution finds an optimal solution in 1122 seconds.

We show two more results on other examples in Figure 6. Figure 6(a) is a rectangular shaped target for which three shots are used. The optimization model finds the solution of two $4mm$ and one $14mm$ shots depicted in Figure 6(b). The total time to produce the solution is about 15 seconds. Another example is given in Figure 6(c,d). This is a small tumor (less than 1 in^2) for which three shots are again used. The SLSD model takes 1.5 seconds to generate the starting solution. The NLP model finds an optimal solution of two $4mm$ and one $8mm$ shots in 6 seconds.

4.2 Application to real patient data

We have tested our techniques on seven targets arising from real patient cases. The seven targets are radically different in size and complexity. The tumor volumes are ranging from 30 voxels to 36088 voxels. In fact, the tool that implements the process described in this paper is in use at the University of Maryland Medical School. Since our problems are not convex, the choice of parameters in their solution can also have dramatic effects. In this section, we demonstrate how to choose good parameters for the NLP models. Some further description of the medical implications of these results are given in [14].

We generate good initial shot center locations and sizes by running SLSD. This is a starting solution for the NLP model with an exception of shot exposure times. These times $t_{s,w}$ are estimated using the following simple linear program.

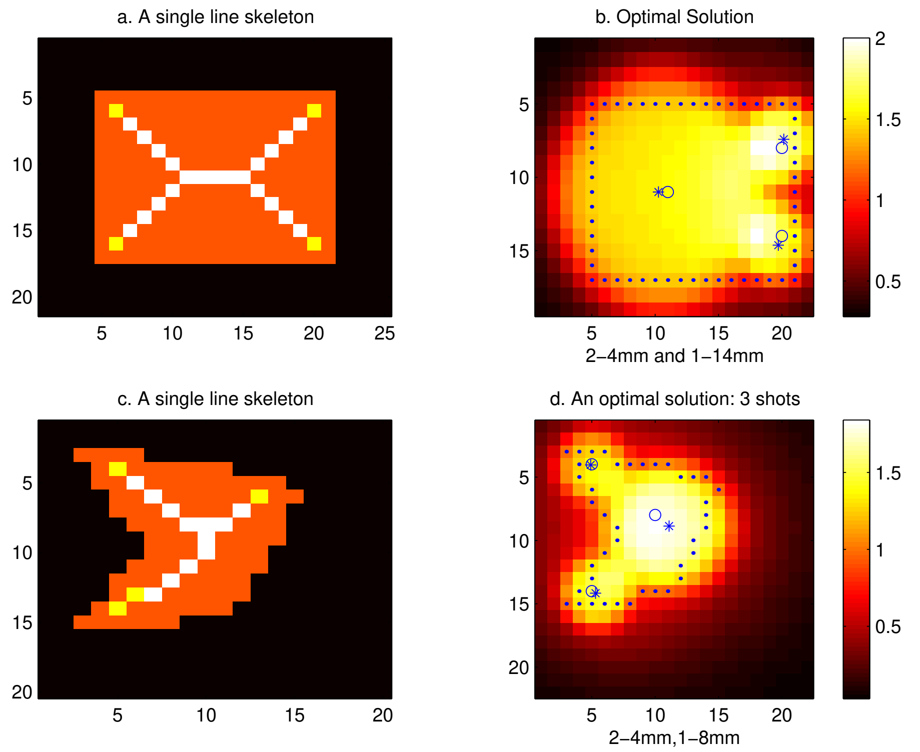

Figure 6: Two-dimensional examples: a rectangular target(a,b) and a small target(c,d)

$$
\begin{array}{lll}
\min & \displaystyle\sum_{(i,j,k)\in\mathcal{G}} UnderDose(i,j,k) & \\
\text{subject to} & Dose(i,j,k) = \displaystyle\sum_{(s,w)\in\mathcal{S}\times\mathcal{W}} t_{s,w} D_w(\bar{x}_s, \bar{y}_s, \bar{z}_s, i, j, k) & \\
& \theta \leq UnderDose(i,j,k) + Dose(i,j,k) & (6) \\
& 0 \leq UnderDose(i,j,k) & \\
& 0 \leq Dose(i,j,k) \leq 1, \quad \forall (i,j,k) \in \mathcal{G} & \\
& \underline{t} \leq t_{s,w} \leq \bar{t} &
\end{array}
$$

Note that we fix the locations of the shots at the points suggested by SLSD and only update the exposure times. Furthermore, we ensure that every size shot has positive weight in an initial solution by enforcing a lower bound (typically 0.1) on the exposure lengths.

The procedure for varying α (controlling the enforcement of the discrete choices) can have dramatic on solution quality and times. We generated solutions for a variety of patients under a number of different choices of α. These solutions were analyzed by an application expert. Based on his feedback, we suggest using values of α between 4 and 8.

Table 2 shows average objective values of three different starting solution generation techniques: Random, SemiRandom, and SLSD. The objective value represents the total average underdose of the target when the solution is applied. The numbers in the parentheses are the standard deviations from a batch of 50 perturbed runs. (In each run, the set of initial solution locations (x, y, z) were perturbed voxel by voxel by a distance of no more than two voxels.) We compare the techniques based on the final objective value and the run time. By fixing $\alpha = 6$, 50 perturbed runs were made for each patient-method pair on a Pentium III workstation. In each run, we generated initial locations randomly within the target for the random scheme, while location perturbation was used for SemiRandom and SLSD. The tumor was so small for Patient 1 that SLSD failed to generate a skeleton (maximum height in map was less than 2).

Using standard statistical tests, the pairwise p-value[16] between Random and Semi-Random was 0.013, between Random and SLSD was 0.0006 and between Semi-Random and SLSD was 0.078. This leads to the conclusion that these results are significantly different at the 90% confidence level.

Table 2 also shows average run time of the entire model for the seven different patients. Although a gain of speed using SLSD depends on the shape and size of the tumor, the table shows that the model execution time can be substantially reduced using SLSD over the other two techniques regardless of the size of tumor. Again, these results are significantly different at the 90% confidence level. The pair-wise p-value between Random and Semi-Random was 0.017, between Random and SLSD was 0.0006 and between Semi-Random and SLSD was 0.063.

Table 2: Average optimal objective value and solution times in seconds for different tumors

Patient	Objective			Time		
(size)	Random	Semi-rand	SLSD	Random	Semi-rand	SLSD
1	2.17	0.88	NA	0.3	0.3	NA
(28)	(0.86)	(0.29)	NA	(0.05)	(0.03)	NA
2	14.70	8.21	6.64	32	30	26
(2144)	(6.90)	(4.68)	(2.61)	(6)	(9)	(9)
3	27.53	19.22	14.43	89	67	52
(3279)	(19.07)	(8.87)	(14.99)	(25)	(16)	(9)
4	16.55	12.89	9.85	97	94	84
(3229)	(4.45)	(6.70)	(4.88)	(18)	(22)	(19)
5	34.87	34.53	23.85	153	128	77
(4006)	(16.36)	(17.26)	(13.84)	(40)	(30)	(17)
6	33.32	28.49	15.00	556	513	355
(6940)	(17.25)	(13.09)	(13.22)	(103)	(100)	(52)
7	35.45	29.97	31.03	590	460	343
(10061)	(12.63)	(11.16)	(13.65)	(228)	(100)	(75)
8	9.31	3.22	2.78	887	240	168
(22124)	(2.73)	(2.80)	(1.72)	(157)	(68)	(56)
9	45.05	35.18	31.05	874	629	498
(24839)	(18.10)	(7.11)	(10.25)	(425)	(166)	(99)
10	18.55	11.57	8.59	3568	937	695
(36088)	(11.20)	(11.83)	(6.71)	(589)	(108)	(79)

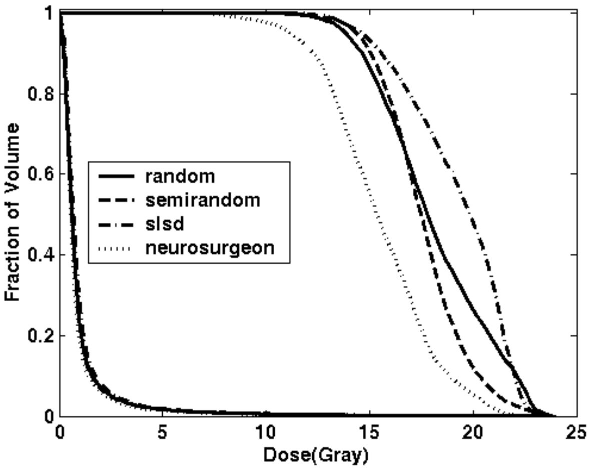

Figure 7: A dose volume histogram for patient 6

To conclude this section, we show a dose volume histogram relating various plans that were generated for patient 6. The histogram depicts the fraction of the volume that receives a particular dose for both the skull, and the target volumes. The curves on the right depict information related to the target, while on the left they ref to the skull. On the target, the curves that extend furthest to the right receive more dose. Since this can be effected by just delivering more dose to the patients skull, the lines to the left show the fraction of the skull that receives a particular dosage. The figure compares the three techniques outlined here, along with the actual plan used on the patient example. Clearly, all of the automatic plans are better than the neurosurgeons plan, while the SLSD approach appears preferable to the other two automatic plans in quality.

5 Conclusion and future directions

We have used a variety of optimization techniques in this paper to develop an approach for solving a planning problem for medical treatment. While our approach has been tailored to the specific application, we believe the methods and approaches used here can be effectively adapted to many other problem classes.

The work described in this paper was motivated by feedback received from an initial prototype use of our planning tool at the University of Maryland Medical School. The key features that needed improvement were the speed and robustness of the process. This paper has addressed both issues by using a variety of different optimization models and computational techniques. In particular, the speed of solving the sequence of nonlinear programming models has been substantially reduced by using the skeleton based starting point generation technique. Statistically, we have shown that SLSD outperforms two other heuristics for generating starting points. Furthermore, the use of an improved conformity estimation model, coupled with a "clean-up" mixed integer programming model, ensures the solutions generated are clinically acceptable and conform to the input specifications of the user. The modified tool is now in use at the hospital without intervention from any of the authors.

Our future work involves predicting the number of shots that can be used for a particular patient.

References

[1] H. Blum. A transformation for extracting new descriptors of shape. In W. Wathen-Dunn, editor, *Models for the Perception of Speech and Visual Form*, pages 362–380. MIT Press, 1967.

[2] J. W. Brandt and V. R. Algazi. Continuous skeleton computation by voroni diagram. *CVGIP:Graph Models Image Processing*, 55:329–338, 1992.

[3] A. Brooke, D. Kendrick, and A. Meeraus. *GAMS: A User's Guide*. The Scientific Press, South San Francisco, California, 1988.

[4] A. Drud. CONOPT: A GRG code for large sparse dynamic nonlinear optimization problems. *Mathematical Programming*, 31:153–191, 1985.

[5] M. C. Ferris, J.-H. Lim, and D. M. Shepard. Radiosurgery treatment planning via nonlinear programming. Data Mining Institute Technical Report 01-01, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, 2001.

[6] M. C. Ferris and D. M. Shepard. Optimization of gamma knife radiosurgery. In D.-Z. Du, P. Pardalos, and J. Wang, editors, *Discrete Mathematical Problems with Medical Applications*, volume 55 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 27–44. American Mathematical Society, 2000.

[7] J. C. Ganz. *Gamma Knife Surgery*. Springer-Verlag Wien, Austria, 1997.

[8] Y. Ge and J. M. Fitzpatrick. On the generation of skeletons from discrete euclidean distance maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(11):1055–1066, 1996.

[9] L. Hong, A. Kaufman, Y. Wei, A. Viswambharn, M. Wax, and Z. Liang. 3D virtual colonoscopy. *Proceedings of 1995 Symposium on Biomedical Visualization Atlanta Ga.*, pages 26–32, 1995.

[10] ILOG CPLEX Division, 889 Alder Avenue, Incline Village, Nevada. *CPLEX Optimizer*. http://www.cplex.com/.

[11] F. Leymarie and M. D. Levine. Fast raster scan distance propagation on the discrete rectangular lattice. *Computer Vision Graphics Image Processing*, 55(1):84–94, 1992.

[12] C. M. Mao and M. Sonka. A fully parallel 3D thinning algorithm and its applications. *Computer Vision Image Understanding*, 64:420–433, 1996.

[13] C. W. Niblack, P. B. Gibbons, and D. W. Capson. Generating skeletons and centerlines from the sitance transform. *CVGIP:Graph Models Image Processing*, 54:420–437, 1992.

[14] D. M. Shepard, M. C. Ferris, R. Ove, and L. Ma. Inverse treatment planning for gamma knife radiosurgery. *Medical Physics*, 27:12, 2000.

[15] Q. J. Wu and J. D. Bourland. Morphology-guided radiosurgery treatment planning and optimization for multiple isocenters. *Medical Physics*, 26(10):2151–2160, 1999.

[16] B. S. Yandell. *Practical Data Analysis for Designed Experiments*. Chapman and Hall, London, 1997.

[17] Y. Zhou, A. Kaufman, and A. W.Toga. Three dimensional skelton and centerline generation based on an approximate minimum distance field. *Visual Computers*, 14:303–314, 1998.

[18] Y. Zhou and A. W. Toga. Efficient skeletonization of volumetric objects. *IEEE Transaction on Visualization and Computer Graphics*, 5(3):196–209, 1999.